



SAPIENZA
UNIVERSITÀ DI ROMA

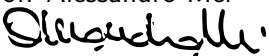
NFT e Smart contract in Algorand

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Corso di Laurea Magistrale in Informatica

Candidato

Alessandro Di Patria
Matricola 1844538

Relatore

Prof. Alessandro Mei


Correlatore

Dott. Massimo La Morgia

Anno Accademico 2021/2022

Tesi non ancora discussa

NFT e Smart contract in Algorand

Tesi di Laurea Magistrale. Sapienza – Università di Roma

© 2021 Alessandro Di Patria. Tutti i diritti riservati

Questa tesi è stata composta con L^AT_EX e la classe Sapthesis.

Versione: 9 ottobre 2021

Email dell'autore: dipatria.1844538@studenti.uniroma1.it

Indice

1	Introduzione	1
2	Gli NFT	3
2.1	Definizione	3
2.2	NFT, funzionalità e caratteristiche	3
2.3	Il loro utilizzo	4
2.4	NFT, alcuni esempi	5
2.4.1	Criptopunks	5
2.4.2	Criptokitties	6
3	Gli smart contract	7
3.1	Definizione di Smart contract	7
3.1.1	Ciclo di vita degli smart contract	8
3.2	Smart contract in Ethereum	9
3.2.1	Anatomia di uno smart contract Ethereum	9
3.3	ERC-20	10
3.4	ERC-721	10
3.4.1	Funzioni	10
3.4.2	Eventi	12
3.4.3	Metadati	12
3.5	Algorand smart contract	13
3.5.1	Stateful Smart Contract	13
3.5.2	Stateless Smart Contract	16
3.5.3	La logica Teal	18
3.5.4	Le transazioni	18
4	Creare e distribuire NFT	21
4.1	Creare e distribuire NFT in Algorand	21
4.1.1	Algorand Standard Asset	21
4.1.2	NFT con smart contract stateless	23
4.1.3	NFT con Smart Contract Stateful	23
5	Creare applicazioni personalizzate con Algorand	25
5.1	Atomic Transfer	25
5.2	Collegare stateful a stateless smart contract	26

6 Logica per la gestione e scambio di NFT	29
6.1 Il problema	29
6.2 La Logica	29
6.3 Usa Case	30
6.4 Requisiti	30
6.5 Architettura	31
6.5.1 Componente stateful	31
6.5.2 Creazione NFT	33
6.5.3 Stateless Smart contract	33
6.6 Implementazione	35
7 Conclusione e sviluppi futuri	37
A Componente stateful	39
A.1 clear state	39
A.2 Approval program	39
A Componente stateless	43

Capitolo 1

Introduzione

Negli ultimi mesi si è sentito parlare spesso nei telegiornali e mezzi di informazione di opere d'arte digitali, riferendosi ad esse con il termine NFT, vendute per milioni di euro. Ad esempio, la casa d'aste *Christie's* ha venduto per 69 milioni di dollari un quadro digitale [10]. L'NBA (National Basketball Association) attraverso il servizio *NBA top shop* commercializza filmati riguardanti il basket per un volume di affari di oltre 250 milioni di dollari [15]. Questi appena citati sono solo alcuni degli esempi relativi alla commercializzazione di NFT.

Con il termine NFT si fa riferimento ad un acronimo della parola inglese *Non Fungible Token*; gli NFT possono essere pensati come certificati che attestano l'autenticità, l'unicità e la proprietà di un file digitale. Queste proprietà vengono garantite grazie all'impiego della blockchain. Gli NFT vengono considerati da taluni il giusto e tanto atteso modo per dare valore e unicità ai contenuti digitali che altrimenti faticherebbero a trovarne. Opposto è invece il parere di chi li considera poco più che una bolla speculativa.

La blockchain di Ethereum è attualmente la rete più utilizzata per la creazione e la distribuzione degli NFT. Tuttavia, i costi di transazione elevati ed i problemi di scalabilità della rete Ethereum possono rappresentare una barriera di accesso alla diffusione di massa di questo particolare tipo di prodotti. Esistono tuttavia alcune blockchain che mirano ad eliminare questi problemi. Una di queste è Algorand la quale mira ad essere sicura, scalabile e decentralizzata.

L'obiettivo di questa tesi è quello di analizzare gli standard utilizzati sulla rete Ethereum (ERC-721) per la creazione degli NFT (Capitolo 2), illustrandone la struttura, le caratteristiche, e le relative problematiche ed analizzare la possibilità di replicarli sulla rete Algorand. Passeremo poi all'analisi degli smart contract (Capitolo 3) che permettono la creazione e lo scambio di NFT. Analizzeremo anche come combinare le varie tecnologie di Algorand per creare qualcosa di unico (Capitolo 5). Per giungere infine alla parte progettuale in cui verrà presentata una logica personalizzata che consente la gestione e lo scambio di NFT con altri utenti della rete (Capitolo 6). Infine, nelle conclusioni verranno illustrati i risultati raggiunti in questa tesi.

Capitolo 2

Gli NFT

In questo capitolo verranno presentate le principali caratteristiche che compongono questi particolari token crittografici. Inoltre verranno presentati alcuni progetti che hanno amplificato il fenomeno degli NFT: *I Criptokitties* e *I Criptopunks*.

2.1 Definizione

Un NFT è un record contenente dei dati che viene archiviato in un registro digitale chiamato blockchain. La decentralizzazione della blockchain garantisce che quel record memorizzato non possa essere manomesso o cancellato. I dati contenuti all'interno del record servono per certificare che un determinato asset è unico. Un asset è un bene digitale in formato binario che viene fornito con il diritto d'uso¹. Tra gli esempi di asset digitali troviamo documenti, immagini, video e canzoni. Un NFT viene inserito nella blockchain come prova di proprietà dell'asset. Chi acquista un NFT ne acquista la proprietà, l'autenticità e l'unicità. Per capire davvero cosa è un NFT, è utile familiarizzare con il concetto economico di fungibilità. Gli oggetti fungibili possono essere scambiati tra loro con facilità perché il loro valore non è legato alla loro unicità. Ad esempio, se puoi scambiare una banconota da 1 dollaro con un'altra da 1 dollaro e avere ancora 1 dollaro, con oggetti non fungibili ciò invece non accade poiché non sono intercambiabili. Con gli NFT, ogni token ha proprietà uniche e non vale lo stesso importo di altri token simili.

2.2 NFT, funzionalità e caratteristiche

Come detto in precedenza gli NFT sono un tipo di file memorizzato nella blockchain. In questo file non viene inserito alcun contenuto multimediale poiché finirebbe per appesantire l'intera rete. Per evitare ciò viene inserito un puntatore ad un file esterno. Questo file contiene i cosiddetti metadati dell'NFT. I metadati sono dati che forniscono informazioni su altri dati; nel caso di un NFT i metadati descrivono le proprietà essenziali di tale NFT, incluso il suo nome, la descrizione e qualsiasi altra cosa il suo creatore ritenga importante. In molti casi, i metadati di un NFT contengono anche collegamenti ad altre risorse digitali (immagini, filmati, etc...) che

¹Con diritto d'uso si intende il diritto di servirsi di un bene, e se fruttifero, di raccoglierne i frutti

danno a un NFT il suo vero valore.

Solitamente questi metadati sono file che vengono memorizzati in sistemi di cloud distribuiti chiamati anche IPFS. L'InterPlanetary File System (IPFS) è un protocollo di comunicazione e una rete peer-to-peer per l'archiviazione e la condivisione di dati in un file system distribuito. I marketplace di NFT e altre applicazioni sfruttano i metadati per mostrare gli NFT ad acquirenti e venditori. Per questo è importante che i metadati siano in un formato comprensibile per i marketplace. Per rendere gli NFT compatibili con l'ecosistema di mercati, portafogli e altri strumenti NFT si dovrebbe adottare uno standard di metadati esistente e, se necessario, adattarlo alle esigenze specifiche.

Il formato più comune per i metadati NFT è JSON ^[2], un formato leggero, definito per primo dal linguaggio JavaScript.

Alcuni sviluppatori di NFT su blockchain Ethereum hanno definito uno standard di metadati che può essere esteso con altre funzionalità ^[16]. Nella figura sottostante viene presentato un esempio di metadati.

La logica che governa lo scambio, la distribuzione e le funzionalità di un NFT è lo smart contract. Lo smart contract è un programma realizzato con un particolare linguaggio di programmazione che viene compilato e distribuito nella blockchain. Interagendo con gli smart contract attraverso delle transazioni è possibile coniare NFT.

```
{
  "title": "Asset Metadata",
  "type": "object",
  "properties": {
    "name": {
      "type": "string",
      "description": "Identifies the asset to which this NFT represents"
    },
    "description": {
      "type": "string",
      "description": "Describes the asset to which this NFT represents"
    },
    "image": {
      "type": "string",
      "description": "A URI pointing to a resource with mime type image/* representing the asset to which this NFT represents"
    }
  }
}
```

Figura 2.1. La figura illustra un esempio di metadati in formato JSON.

2.3 Il loro utilizzo

Quello dell'arte è il caso d'uso più famoso degli NFT. Non solo vengono scambiati NFT di immagini ma anche di canzoni e di oggetti da collezione. Nel settore musicale vi è una vera e propria corsa al digitale. Uno degli esempi più celebri è quello del noto dj *Steave Aoki* ^[21] il quale ha venduto una serie di NFT. Tra questi vi è un'animazione di mezzo minuto accompagnata da una sua traccia musicale che, da sola, ha fruttato oltre 800 mila dollari. Anche gli oggetti da collezione non sono da meno, una delle collezioni più importanti è quella dei "Criptokitties". I "Criptokitties" sono degli avatar di gattini digitali collezionabili che possono essere acquistati o scambiati. Il loro utilizzo, però, può riguardare molti altri campi. La tecnologia degli NFT può essere utilizzata per la gestione e la tutela dei diritti d'autore. Un

²<https://www.json.org/json-en.html>

artista potrebbe mettere in vendita i propri biglietti digitali per il suo concerto e guadagnare con i diritti d'autore. La società italiana *SIAE* [17] impegnata nella tutela dei diritti di autore, sta implementando un sistema, attraverso la blockchain di Algorand, che consente di tutelare i diritti degli artisti in maniera completamente digitale.

2.4 NFT, alcuni esempi

I progetti più interessanti che hanno rivoluzionato ed ispirato il mondo degli NFT sono stati i "Crypto Punks" e i "Crypto Kitties".

2.4.1 Criptopunks



Figura 2.2. L'immagine di presentazione dei criptopunks, una collezione di personaggi digitali

CryptoPunks è stato il primo progetto NFT sulla blockchain Ethereum [3]. Come riporta la foto sovrastante sono delle semplicissime immagini raffiguranti volti stilizzati di esseri umani o alieni presenti in soli 10 mila esemplari. Due di loro, i numeri 3100 e 7804, sono stati venduti nello scorso marzo a 4200 ETH, circa 7,58 milioni di dollari. Questo evento ha avuto un forte impatto nella vendita degli NFT. Ognuno di questi volti è stato generato tramite un algoritmo per questo non esistono due volti esattamente uguali, sebbene alcuni tratti siano più rari di altri. I CryptoPunk, che rimarranno sempre 10.000, sono stati originariamente rilasciati gratuitamente e potrebbero essere richiesti da chiunque abbia un portafoglio Ethereum. I Crypto Punk si basano sullo smart contract ERC-721, elaborato dagli sviluppatori responsabili del progetto allo scopo di produrre degli oggetti non fungibili. Questo è divenuto lo standard per la creazione di contratti intelligenti riguardanti NFT.

Dato che un'immagine è normalmente troppo grande per poter essere memorizzata sulla blockchain è stato necessario creare un'immagine di riferimento. Allo scopo, l'azienda creatrice del progetto ha creato un'immagine contenente tutti e 10.000 i CryptoPunk in cui ciascun Punk viene identificato tramite i metadati presenti nel suo token, i metadati si riferiscono infatti alla sua posizione nell'immagine composita. I Crypto Punks servono a poco altro se non a scopi di collezione, anche se qualcuno ha deciso di stampare i propri personaggi e venderli all'asta.

2.4.2 Crip tokitties

I CryptoKitties sono gattini digitali da collezione costruiti sulla blockchain di Ethereum. [14] L'immagine sottostante mostra l'aspetto di questi asset digitali. Possono essere acquistati e venduti usando ETH, la moneta di Ethereum. Al momento del lancio sono stati pensati 50.000 gatti denominati "Clock Cats" che sono stati archiviati in uno smart contract sulla blockchain di Ethereum. Questi Clock Cats vengono distribuiti automaticamente tramite contratto intelligente al ritmo di un gattino ogni 15 minuti. Ogni gatto viene venduto all'asta. I CryptoKitties hanno un aspetto unico, con un aspetto visivo distinto (fenotipo) determinato dai suoi geni immutabili (genotipo) che sono memorizzati nel contratto intelligente. Lo smart contract attraverso delle funzioni interne permette ai Crip tokitties di riprodursi. Difatti due Crypto Kitties possono riprodursi per creare un nuovo gatto che è la combinazione genetica dei suoi genitori. I Crypto Kitties sono infiniti: ogni coppia di gatti si può accoppiare generando nuove e a volte rare composizioni genetiche di gattini. Tutto questo rende i Crypto Kitties una comunità autosufficiente in cui gli utenti possono creare nuovi oggetti da collezione facendo accoppiare questi personaggi oppure scambiarli sulla blockchain di Ethereum.

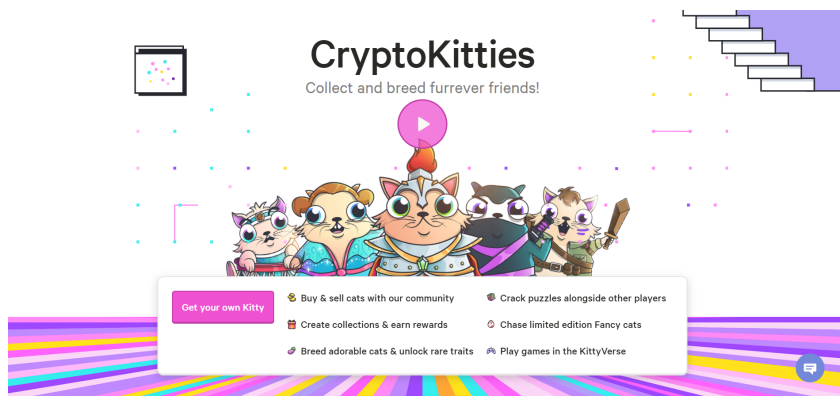


Figura 2.3. L'immagine di presentazione dei crip tokitties, la collezione di gattini digitali

Capitolo 3

Gli smart contract

In questo capitolo vengono spiegate le meccaniche fondamentali che governano e distribuiscono NFT: gli smart contract. Inoltre viene presentato il funzionamento e l'implementazione di vari tipi di smart contract sia su blockchain Ethereum sia su blockchain Algorand.

3.1 Definizione di Smart contract

Uno smart contract è un software implementato su una tecnologia blockchain. Il compito di questi contratti digitali è di eseguire delle azioni automatiche al verificarsi di determinate condizioni [6]. Questi contratti possono definire regole, come un normale contratto, e farle applicare automaticamente. Gli smart contract permettono l'esecuzione di transazioni credibili senza la presenza di un'autorità di terze parti. Gli utenti che vogliono interagire con uno smart contract devono chiamarlo inviando una transazione in rete. In questo modo un utente può usufruire delle sue funzionalità.

Dato che gli smart contracts sono archiviati nella blockchain, ne ereditano alcune interessanti proprietà. Essi sono immutabili e distribuiti. Uno smart contract è immutabile perché, dopo essere stato creato, non potrà mai essere modificato quindi nessuno potrà manometterne il codice. Mentre è distribuito in quanto un contratto è validato da ogni nodo della rete. Quindi un singolo utente non può forzare il contratto e rilasciare i fondi perché altre persone della rete individueranno questo tentativo e lo marcheranno come invalido. L'uso degli smart contracts non è limitato alla raccolta fondi, le banche potrebbero utilizzarli per emettere prestiti o per offrire pagamenti automatici oppure, come vedremo, possono essere usati per la creazione di Non-Fungible Token.

Uno smart contract può essere paragonato ad un distributore automatico: come un distributore automatico elimina la necessità di un dipendente per fornire le bibite così i contratti intelligenti possono sostituire eventuali intermediari in molti settori.

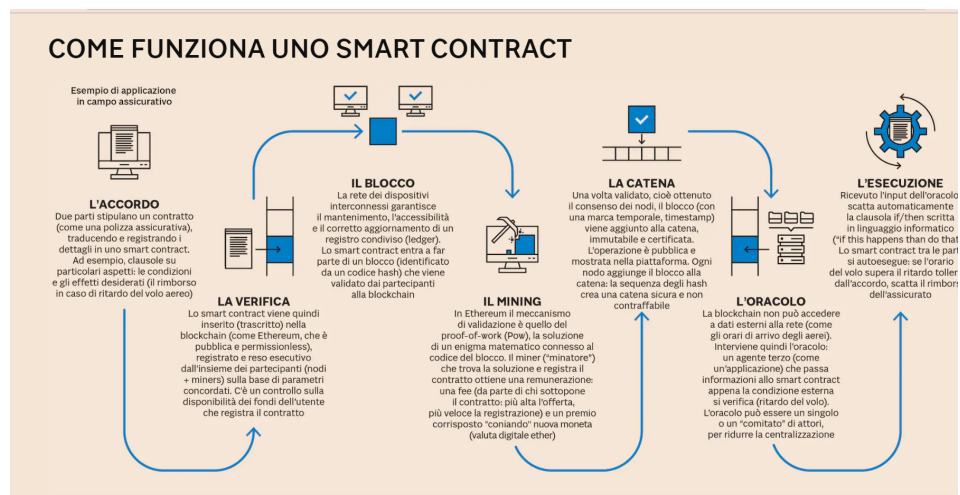


Figura 3.1. Ciclo di implementazione di uno smartcontract

3.1.1 Ciclo di vita degli smart contract

Una volta stipulato un contratto tra le parti ed inseriti i dettagli all'interno di uno smart contract, [18] questo viene inviato alla rete e trascritto nella blockchain. Lo smart contract quindi entra a far parte di un blocco che viene validato dai partecipanti o "miners" della blockchain e memorizzato sulla catena. Inoltre gli smart contract da soli non possono ottenere informazioni sugli eventi "reali" perché non possono inviare richieste HTTP. Fare affidamento su informazioni esterne potrebbe compromettere il consenso fondamentale per la sicurezza e il decentramento. Per aggirare questo problema si usano gli "oracoli", applicazioni affidabili che collegano la blockchain ad informazioni off-chain.

La figura 3.1 mostra l'intero ciclo di vita di uno smart contract.

3.2 Smart contract in Ethereum

Ethereum è stata la prima piattaforma ad accettare Smart Contract. Esistono diversi tipi di contratti che sono stati realizzati nel corso del tempo. Queste tipologie di contratti sono diventati dei veri e propri standard per Ethereum [5]. I più famosi in questo caso sono lo standard ERC-20 utilizzato per gestire token "fungibili" ed l'ERC-721 utilizzato per gestire token non fungibili. Questi contratti sono scritti in solidity, un particolare linguaggio di programmazione creato appositamente per gli smart contract Ethereum. I contratti intelligenti sono pubblici su Ethereum e possono essere pensati come API aperte. Ciò significa che è possibile chiamare altri contratti intelligenti nel proprio contratto intelligente per estendere notevolmente le sue funzionalità. Lo smart contract viene salvato nella blockchain sotto forma di bytecode (un linguaggio di programmazione astratto). Per interagire con uno smart contract, una volta distribuito, ci serviamo di un ABI (Contract Application Binary Interface) questa funge da interfaccia tra l'utente e il contratto. L'ABI definisce le funzioni che puoi invocare e garantisce che le funzioni ritornino i dati nel modo che ci si aspetta.

3.2.1 Anatomia di uno smart contract Ethereum

Ogni smart contract è composto da funzioni, dati ed eventi [1]. I dati vengono memorizzati come variabili di stato e vengono salvati in maniera permanente nella blockchain. I valori di uno smart contract sono *address*, *boolean*, *integer*, *fixedPoint number*, *fixed size byte array*, *dynamically seize byte array*, *Rational and integer literals*, *hexadecimals literals*, *enums*. Esistono anche dei valori che vengono archiviati per la durata dell'esecuzione di una funzione del contratto, queste funzioni si chiamano variabili di memoria. Queste sono molto più economiche da usare perché non sono salvate in modo permanente nella blockchain. Oltre alle variabili che vengono definite nel contratto esistono delle variabili speciali. Esse sono principalmente usate per fornire informazioni alla blockchain o sulla transazione corrente.

Le funzioni nei contratti Ethereum possono ottenere delle informazioni o impostare valori in risposta alle transazioni in entrata.

Esistono due tipi di chiamate a funzione:

- Internal: funzioni e variabili di stato accessibili solo all'interno del contratto.
- External: funzioni esterne che fanno parte dell'interfaccia del contratto, ciò significa che possono essere chiamate da altri contratti e tramite transazioni.

Le funzioni possono anche essere *public* o *private*. Le funzioni public possono essere chiamate internamente al contratto ma anche esternamente tramite messaggi. Le funzioni private sono visibili solo per il contratto per cui sono definite. Le funzioni dette "getter" ottengono solo informazioni riguardo lo smart contract. Invece alcune funzioni possono modificare le variabili di stato di uno smart contract.

Esistono altri tipi di funzioni, una di queste è la *constructor*. Questa viene eseguita solo alla prima distribuzione dello smart contract e ha il compito di inizializzare le variabili.

Quando una transazione viene effettuata, i contratti intelligenti possono emettere eventi e scrivere informazioni sulla blockchain.

3.3 ERC-20

In questo paragrafo si descrive brevemente un tipo di contratto molto comune nella blockchain Ethereum: *ERC-20*. ERC -20, il contratto Ethereum che introduce uno standard per i token fungibili [8]. Tale contratto definisce il concetto di fungibilità. Tutti i token dello stesso tipo sono uguali e hanno tutti lo stesso valore, proprio come una valuta come il dollaro. Esempi di questi token possono essere asset finanziari come le valute. Questo tipo di contratto fornisce delle funzionalità standard come il trasferimento di token da un account all'altro, il bilancio di un determinato account o il numero totale di token e molte altre. In questa tesi ci focalizzeremo maggiormente nello standard per gli NFT che è proprio l'ERC-721.

3.4 ERC-721

In questo paragrafo viene definita l'anatomia e le funzionalità di uno smart contract che è ormai diventato uno standard per creare Non-Fungible Token: ERC-721. ERC-721 è uno standard [4] che definisce un'interfaccia minima che un contratto intelligente deve implementare per consentire la gestione, la proprietà e lo scambio di Non-Fungible Token. Non impone uno standard per i metadati né limita l'aggiunta di funzioni supplementari. Tutti gli NFT hanno una variabile di tipo uint256 chiamata tokenId, quindi per qualsiasi contratto ERC-721, la coppia indirizzo dello smart contract e uint256 tokenId deve essere univoca a livello globale. Questo è ciò che rende un NFT unico nella blockchain Ethereum.

Questo contratto fornisce funzionalità standard come trasferire un asset da un account a un altro, ottenere il saldo token corrente di un account, ottenere il proprietario di un token specifico, la fornitura totale del token disponibile sulla rete e molte altre. Se uno Smart Contract implementa i seguenti metodi ed eventi, può essere chiamato ERC-721 Non-Fungible Token Contract.

3.4.1 Funzioni

Questo paragrafo spiega tutte le funzionalità che un contratto deve implementare per essere considerato un ERC-721. Le funzioni elencate sono visibili nel dettaglio nelle immagini sotto riportate.

- function balanceOf()
Conta il numero di NFT che sono stati assegnati ad un proprietario. "owner" è l'indirizzo da interrogare per conoscere la quantità di asset posseduti.
- function ownerOf()
Dato in input l'identificativo di un asset la funzione ritorna l'indirizzo del proprietario di quell'nft.
- function safeTransferFrom()
Trasferisce la proprietà di un Nft da un indirizzo ad un altro.

```

function balanceOf(address _owner) external view returns (uint256);
function ownerOf(uint256 _tokenId) external view returns (address);
function safeTransferFrom(address _from, address _to, uint256
_tokenId, bytes data) external payable;
function safeTransferFrom(address _from, address _to, uint256
_tokenId) external payable;
function transferFrom(address _from, address _to, uint256 _tokenId)
external payable;
function approve(address _approved, uint256 _tokenId) external
payable;
function setApprovalForAll(address _operator, bool _approved)
external;
function getApproved(uint256 _tokenId) external view returns
(address);
function isApprovedForAll(address _owner, address _operator)
external view returns (bool);

```

Figura 3.2. La figura illustra gli eventi che uno smart contract deve implementare per essere ritenuto uno standard.

- `function safeTransferFrom()`
Trasferisce la proprietà di un NFT da un indirizzo a un altro indirizzo. Funziona in modo identico all'altra funzione con un parametro dati aggiuntivo tranne che questa funzione imposta semplicemente i dati su "".
- `function transferFrom()`
Indica il trasferimento di proprietà di un NFT. Se il destinatario non è in grado di ricevere l'NFT questo viene perso per sempre.
- `function approve()`
Modifica o ri-conferma l'indirizzo per un NFT. Il parametro `approved` rappresenta il nuovo token da approvare mentre `tokenId` rappresenta l'identificativo dell'NFT.
- `function setApprovalForAll()`
Abilita o disabilita l'approvazione di un operatore a gestire tutte le risorse del mittente.
- `function getApproved()`
Dato un identificativo per un NFT la funzione ritorna l'indirizzo per cui è stato approvato quel token.
- `function isApprovedForAll()`
Interroga se un indirizzo "operator" è un operatore autorizzato per gestire l'account del proprietario. Ritorna `true` se si verifica `false` altrimenti.

3.4.2 Eventi

```
1     event Transfer(address indexed _from, address indexed _to, uint256
      indexed _tokenId);
2     event Approval(address indexed _owner, address indexed _approved,
      uint256 indexed _tokenId);
3     event ApprovalForAll(address indexed _owner, address indexed
      _operator, bool _approved);
4
```

Figura 3.3. La figura illustra gli eventi che uno smart contract deve implementare per essere ritenuto uno standard

- event Transfer();
Questo evento viene emesso quando la proprietà di un qualsiasi NFT cambia per qualsiasi causa. Viene emesso quando vengono creati NFT e distrutti.
- event Approval();
Viene emesso quando l'indirizzo approvato per un NFT viene modificato o riaffermato. L'indirizzo zero indica che non esiste un indirizzo approvato.
- event ApprovalForAll();
Viene emesso quando un operatore può gestire tutti gli NFT del proprietario.

3.4.3 Metadati

Nello standard ERC-721 di ethereum l'estensione dei metadati è facoltativa. L'estensione consente di visualizzare il nome e i dettagli sulle risorse che il contratto rappresenta.

```
interface ERC 721Metadata
function name() external view returns (string _name);\newline
function symbol() external view returns (string _symbol);
function tokenURI(uint256 _tokenId) external view returns (string);
```

La prima funzione name() viene invocata per restituire il nome di un NFT. La seconda funzione symbol () restituisce il suo simbolo. Infine tokenURI è una funzione che dato in input un identificativo chiamato anche tokenId, restituisce un url. Questo indirizzo è un puntatore verso un file JSON. Il file serve per memorizzare i metadati più complessi come ad esempio le caratteristiche di un particolare NFT oppure la sua immagine.

3.5 Algorand smart contract

A differenza di Ethereum gli Algorand Smart Contract (ASC1) sono programmi molto leggeri. Gli ASC1 sono scritti in un linguaggio di programmazione chiamato Transaction Execution Approval Language (TEAL). TEAL è un linguaggio di tipo assembly interpretabile da qualsiasi nodo di Algorand. In Algorand abbiamo solo due categorie di smart contract: gli *Stateless Smart Contract* e *Stateful Smart Contract* [13].

3.5.1 Stateful Smart Contract

Gli Stateful Smart Contract vengono memorizzati nella blockchain. La loro funzione è tener traccia e manipolare valori locali e globali definiti all'interno della logica del contratto. I valori globali sono associati al contratto stesso mentre i valori locali sono associati agli utenti che interagiscono con il contratto. Per valori locali si intende il salvataggio dei valori all'interno di un record relativo al saldo dell'account se quest'ultimo ha partecipato allo Smart Contract. Invece i valori globali vengono salvati direttamente nella blockchain.

I contratti intelligenti con stato vengono implementati utilizzando due programmi TEAL; *L'Approval Program* e il *Clear program* [12]. L'Approval Program è responsabile dell'elaborazione di tutte le chiamate di applicazione al contratto. Questo programma è responsabile dell'implementazione della maggior parte della logica di un'applicazione.

Clear State Program si occupa di gestire delle chiamate al contratto effettuate dagli utenti per eliminare gli Stateful Smart Contract dai record relativi al saldo. L'utilizzo di due programmi permette al singolo account di rimuovere lo smart contract dal record relativo al suo saldo sia nel caso in cui Approval Program ritorna false, quindi fallisce, sia nel caso in cui ritorna true. Qualora la logica TEAL dovesse ritornare il valore false, allora tutte le modifiche apportate ai valori locali e/o globali non sarebbero definitive. Le chiamate ai contratti intelligenti con stato vengono effettuate utilizzando le *Application call*, uno specifico tipo di transazione. Questi tipi di transazione si dividono in 6 categorie:

- No Op: chiamate generiche utilizzate per eseguire l'approval program. Queste sono le transazioni più frequenti.
- OptIn: gli account utilizzano questa transazione per aderire al contratto intelligente.
- DeleteApplication: transazione per eliminare l'applicazione cioè lo stateful smart contract.
- UpdateApplication: transazione per che permette di aggiornare i programmi TEAL per uno smartcontract.
- CloseOut: gli account utilizzano questa transazione per chiudere la loro partecipazione al contratto. Questa chiamata può fallire in base alla logica TEAL, impedendo all'account di rimuovere il contratto dal suo record di saldo.

- `ClearState`: simile a `CloseOut`, ma la transazione cancellerà sempre un contratto dal record del saldo del conto se il programma ha esito positivo o negativo. L'utilizzo di questo tipo di transazione è consigliato quando si vuole avere la certezza di rimuovere la propria partecipazione da uno Stateful Smart Contract.

Il numero di valori globali e/o locali che uno Stateful Smart Contract può scrivere viene specificato durante la sua creazione. Ogni valore viene rappresentato attraverso l'utilizzo di una coppia chiave-valore. Il limite massimo per la chiave e il valore è di 64 bytes, sia che si parli di valori globali oppure di valori locali.

Ogni account che decide di partecipare allo Stateful Smart Contract è soggetto alla modifica dei propri valori locali se esegue una transazione indirizzata ad uno smart contract. Inoltre ogni volta che un account esegue una transazione indirizzata ad uno Stateful Smart Contract questa può comportare la modifica dei valori locali di ulteriori quattro account. Qualsiasi Stateful Smart Contract ha la possibilità di leggere il saldo di ogni account. Lo smart contract non può modificare i dati di un account a meno che l'account non abbia eseguito precedentemente una transazione `OptIn` verso quel determinato Stateful Smart Contract. Per la lettura dei valori globali ogni Stateful Smart Contract può visualizzare i valori globali di massimo altri due smart contract.

Per quanto riguarda la modifica dei valori globali degli Stateful Smart Contract è possibile effettuarla attraverso transazioni indirizzate allo smart contract a cui appartengono i valori. Questo vuol dire che solo lo Smart Contract che ha creato i valori globali può modificarli. Inoltre per la lettura dei valori globali ogni Stateful Smart Contract può visualizzare i valori globali di massimo altri due Smart Contract. Per scrivere i valori globali si utilizza l'opcode `"app-global-put"` mentre per la scrittura di valori locali si utilizza l'opcode `"app-local-put"`. Per quest'ultimo va specificato l'account per il quale si sta eseguendo l'operazione. Naturalmente l'opcode `app-local-put` ha esito positivo se e solo se l'account ha eseguito in precedenza una transazione `OptIn` con lo Stateful Smart Contract.

Per leggere le variabili è possibile eseguire quattro diversi opcode: `"app-local-get"`, `"app-global-get"`, `"app-local-get-ex"` e `"app-global-get-ex"`. I primi due opcode servono a leggere valori locali dell'account specificato e i valori globali dello Stateful Smart Contract verso il quale si effettua la transazione. Invece `"app-local-get-ex"` e `"app-global-get-ex"` sono utilizzati per leggere i valori locali e i valori globali di Stateful Smart Contract esterni all'account che esegue la transazione. Questi due opcode ritornano due valori. Il primo valore ritornato può essere 0 oppure 1. Viene ritornato 0 quando il valore richiesto viene ritornato in maniera corretta, mentre viene ritornato 1 in caso contrario. Naturalmente, se il primo valore ritornato è 1, il secondo valore ritornato è il valore richiesto.

Si possono passare degli argomenti alle transazioni che vengono indirizzate allo Stateful Smart Contract. I parametri che si possono passare sono di tipo stringa, intero, data codificati in base64 e indirizzi di account di Algorand. Una volta passati gli argomenti attraverso la transazione, questi vengono caricati in un array e possono essere elaborati attraverso l'opcode *Application Args*. L'ammontare del peso degli argomenti passati ad uno Stateful Smart Contract non può essere superiore ai 2kb. È possibile passare un set di array con qualsiasi transazione verso lo smart contract che permette di caricare dati aggiuntivi. Questi array sono l'array delle applicazioni,

l'array degli account, l' array delle risorse e l' array degli argomenti.

- L'array dell'applicazione viene utilizzato per passare altri smart contract in modo tale da leggere il loro stato.
- L'array degli account consente di trasferire account aggiuntivi al contratto per le informazioni sul saldo e l'archiviazione locale.
- L'array delle risorse viene utilizzato per passare un elenco di asset. Questi possono essere utilizzati per recuperare informazioni sulla configurazione e sul saldo degli asset. L'array degli argomenti viene utilizzato per passare argomenti standard al contratto. Questo è limitato a 16 argomenti con un limite di dimensione di 2 KB.

Gli altri tre array sono limitati a 8 valori totali combinati e, di questi, l'array account non può avere più di quattro valori. I valori passati all'interno di questi array possono cambiare per ogni transazione verso lo smart contract . Gli arrays account e applicazioni contengono il mittente della transazione e l'ID dell'applicazione corrente nella posizione 0 del rispettivo array. Questo sposta il contenuto di questi due array di uno slot.

Il creatore di uno Stateful Smart Contract è l'account che firma ed esegue la transazione per la creazione dello smart contract. Attraverso questa transazione viene specificato il numero di valori globali e di valori locali che si vogliono creare. Così facendo, si indica la quantità di spazio della blockchain richiesta per la corretta esecuzione dello Stateful Smart Contract. Una volta che vengono dichiarate le quantità di spazio è impossibile modificarlo. Al termine della creazione dello Stateful Smart Contract viene ritornato un ApplicationID. Attraverso questo id univoco nella blockchain è possibile indirizzare nel modo corretto qualsiasi tipo di transazione Application Call.

E' possibile creare fino ad un massimo di 64 coppie chiave-valore per quanto riguarda i valori globali, mentre per i valori locali è concessa la creazione di un massimo di 16 coppie chiave-valore. Il primo passo per effettuare transazioni Application Call verso uno Stateful Smart Contract con variabili locale è quello di firmare ed inviare una transazione OptIn verso lo Smart Contract desiderato. Da notare che questa operazione, oltre a qualsiasi account interessato, la deve eseguire anche l'account creatore dello Stateful Smart Contract. Una volta inviate nel network la transazione OptIn, l'Approval Program viene eseguito e se ritorna true allora l'account mittente della transazione OptIn può partecipare allo Stateful Smart Contract

Il saldo minimo è la quantità minima di Algo che un Account deve possedere per effettuare una transazione qualsiasi. Quando un utente crea uno Statful Smart Contract oppure esegue una transazione di tipo OptIn verso un contratto, il saldo minimo che un account deve possedere (normalmente è di 100000 microalgo) viene incrementato. L'aumento del saldo minimo è direttamente proporzionale alla quantità di memoria utilizzata nella blockchain da uno Stateful Smart Contract.

3.5.2 Stateless Smart Contract

Questi tipi di contratto approvano o rifiutano transazioni di spesa. Vengono anche impiegati per firmare una transazione attraverso una firma logica chiamata Logic Sig che viene creata dalla computazione di uno smart contract. Le firme logiche, indicate come LogicSig, sono strutture che contengono le quattro parti seguenti:

- Logic, sono i byte della logica dello smart contract stateless.
- Sig, contiene una firma valida applicata alla logica per conto dell'account che sta mandando la transazione.
- Msig, la firma di un Multisignature Account applicata alla logica TEAL
- Args, array di argomenti passati al programma.

La logic sig è considerata valida se si verificano una delle condizioni qui sotto :

- Sig contiene una firma valida dell'account che ha sottomesso la transazione.
- Msig contiene una Multi-firma valida per conto del multi-signature account che invia la transazione.
- L'hash del programma è uguale all'indirizzo del mittente.

I primi due casi sono esempi di delega [11]. Il titolare di un account può dichiarare che per suo conto la logica firmata può autorizzare le transazioni. Questi account possono essere account a firma singola o multi-firma. Il terzo caso è un account interamente governato dal programma. Il programma non può essere modificato. Una volta che Algos o asset sono stati inviati a quell'account, questi posso essere trasferiti solo quando c'è una transazione che approva la logica.

Gli stateless smart contract possono essere suddivisi in due categorie: il contract account e il Signature Delegation

Contract Account

Nel primo caso lo Smart Contract Stateless può essere utilizzato come un vero account, infatti possiede un proprio indirizzo Algorand. Per utilizzare un programma TEAL come Contract Account, vengono inviati Algo (la valuta di algorand) all'indirizzo del contratto per trasformarlo in un conto con un saldo. La logica del contratto decide se approvare o non approvare le varie transazioni. Attraverso la logica TEAL è possibile eseguire diversi controlli che vengono eseguiti al momento dell'invio di una transazione verso il Contract Account. Quindi la logica TEAL decide se il Contract Account può inviare oppure no ALGO. Per effettuare un pagamento dal Contract Account ad un altro account viene eseguita una transazione che porta la logica TEAL dello smart contract a ritornare il valore true. Il mittente della transazione deve essere il Contract Account. Chiunque può eseguire delle transazioni che portano lo smart contract a trasferire asset ad un altro account, l'unico requisito è possedere la logica TEAL compilata così da poterla applicare come LogicSig (firma) per la transazione. I conti contrattuali sono ottimi per creare conti in stile deposito a garanzia (Escrow) in cui si desidera limitare i prelievi o si desidera effettuare pagamenti periodici. Per fare un esempio, Alice crea un conto contrattuale e lo

finanzia con ALGO. La logica TEAL è scritta per consentire a Bob di rimuovere fino a 200 ALGO ogni periodo di tempo dall'account del contratto. Bob invia una transazione per rimuovere 201 ALGO dall'account del contratto e la transazione viene rifiutata. Bob invia una transazione per rimuovere 150 ALGO e la transazione ha esito positivo. Infine Bob invia nuovamente la stessa transazione ma questa viene rifiutata poichè non ha aspettato il periodo di tempo prefissato.

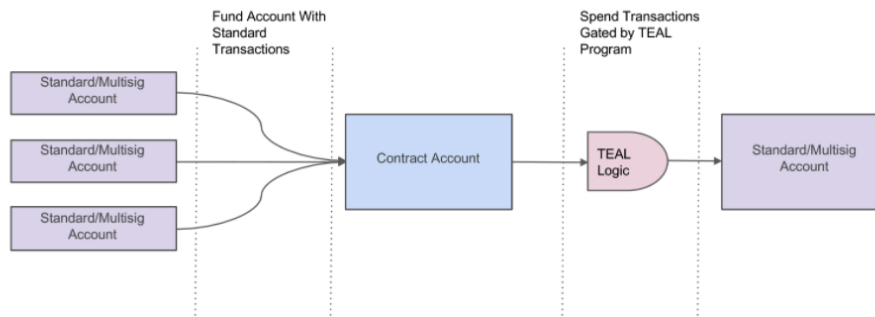


Figura 3.4. La figura illustra il ciclo di vita di un contract account

Signature Delegation

Gli smart contract senza stato possono essere utilizzati anche per delegare le firme. Questo significa che un account può firmare un programma TEAL e l'output risultante può essere utilizzato come firma nelle transazioni per conto dell'account. Il titolare del conto delegato può condividere questa firma logica, consentendo a chiunque di spendere fondi dal proprio conto secondo la logica all'interno del programma TEAL. Uno schema di queste operazioni viene illustrato nella figura 3.6. Ad esempio, se Alice desidera impostare un pagamento ricorrente con la sua società di servizi di pubblica utilità per un massimo di 200 Algo ogni mese, crea un contratto TEAL che codifica questa logica, lo firma con la sua chiave privata e lo dà alla società di servizi. La società di servizi pubblici utilizza quella firma logica nella transazione che invia ogni mese per riscuotere il pagamento da Alice.

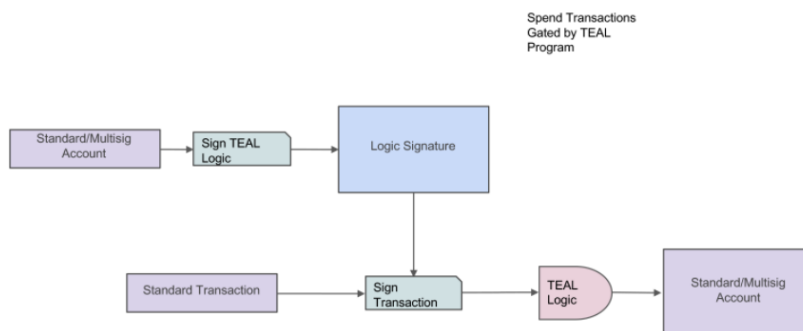


Figura 3.5. La figura illustra il funzionamento di un contratto di tipo Signature Delegated

3.5.3 La logica Teal

I contratti intelligenti senza stato e con stato sono scritti in Transaction Execution Approval Language (TEAL), un linguaggio assembly. Il linguaggio è di tipo Turing Completo che non supporta loop ma le branch. I programmi TEAL vengono elaborati una riga alla volta inserendo (push) ed eliminando (pop) i valori dentro e fuori dallo stack. Questi valori dello stack sono interi a 64 bit senza segno o stringhe di byte. Inoltre ha uno spazio di lavoro diverso dallo stack per memorizzare temporaneamente valori da utilizzare successivamente nel programma. Il linguaggio TEAL offre la possibilità di utilizzare numerosi operational codes (opcodes) [7], attraverso i quali la logica TEAL può avere accesso agli argomenti passati allo smart contract. Non tutti gli opcodes sono comuni ad entrambi i tipi di smart contract. Alcuni sono ad uso esclusivo degli Stateless Smart Contract, altri solo a Stateful Smart Contract mentre altri ancora sono in comune.

Questo linguaggio ritorna sempre un valore booleano, true o false e non sono ammessi altri valori di ritorno. Ritorna true se e solo se esiste un solo valore positivo sulla pila alla fine del processo di esecuzione della logica TEAL. In tutte le altre condizioni, il programma ritorna il valore booleano false. Si ricorda inoltre che, l'ordine con cui gli argomenti vengono passati al programma è fondamentale visto che tutti i valori sono inseriti in uno stack. Inoltre questo linguaggio permette di accedere alle proprietà delle transazioni tramite l'opcode *txn* oppure con *gtxn* che consente di accedere a quelle raggruppate insieme.

3.5.4 Le transazioni

Algorand offre la possibilità di creare diversi tipi di transazioni. Esistono sei tipi di transazioni principali. Nello specifico, i tipi di transazione che si possono creare in Algorand sono: Asset Configuration, Asset Transfer, Asset Freeze, Key Registration, Payment e Application Call.

Asset Configuration

Le transazioni di tipo Asset Configuration servono per creare, modificare oppure distruggere un Algorand Standard Asset. Una transazione per distruggere un Algorand Standard Asset è possibile solo quando l'account possiede tutte le unità e colui che invia e autorizza la transazione è l'account manager. Il creatore e il manager, sono due account di Algorand impostati attraverso la transazione con il quale si crea l'opcode relativo all'asset configuration è **acfg**.

Asset Transfer

Questo tipo di transazione viene utilizzata per accettare di ricevere un determinato Algorand Standard Asset, per trasferire un Algorand Standard Asset da un account ad un altro oppure per rimuovere un Algorand Standard Asset ad un account specifico. L'opcode relativo a questa transazione è **xfer** [9].

Asset Freeze

Una transazione Asset Freeze viene utilizzata per far perdere oppure concedere nuovamente ad un account la possibilità di inviare o ricevere un determinato Algorand

Standard Asset. L'opcode relativo a questa transazione è **afrz**

Key Registration

Lo scopo di questo tipo di transazione è quello di registrare un account per permettergli di partecipare al processo per il raggiungimento del consenso. L'opcode rappresentativo è **Keyreg**.

Payment Una transazione di tipo Payment è una transazione attraverso la quale un account, il mittente, invia ad un altro account una certa quantità di ALGO. La quantità di ALGO inviata non può eccedere quella posseduta dal mittente al momento dell'invio della transazione. L'opcode per questa transazione è **Pay**

Application Call

Sono tutte quelle transazioni che coinvolgono lo Smart Contract Stateful. L'opcode relativo a queste transazioni è **appl**. I vari campi che possono essere passati ad un application call sono: Application ID, OnComplete, Accounts, Approval Program, App Arguments, Clear State Program, Foreign Apps, Foreign Assets, GlobalStateSchema, LocalStateSchema, ExtraProgramPages.

- Application ID, corrisponde all'ID dell'applicazione in fase di configurazione o vuoto se in fase di creazione. Questo Id è unico nella rete.
- OnComplete, Definisce quali azioni aggiuntive si verificano con la transazione.
- Accounts, elenco degli account oltre al mittente a cui è possibile accedere dal programma di approvazione e dal programma di cancellazione dell'applicazione.
- Approval program, è la logica eseguita per ogni transazione dell'applicazione, tranne quando OnComplete è impostato su "clear". Può leggere e scrivere lo stato globale per l'applicazione, nonché lo stato locale specifico dell'account. I programmi di approvazione possono rifiutare una transazione.
- App Arguments, sono argomenti specifici della transazione a cui si accede dall'Approval program.
- Clear State Program: logica eseguita per le transazioni dell'applicazione con onCompletion impostato su "clear". Può leggere e scrivere lo stato globale per l'applicazione, nonché lo stato locale specifico dell'account. I programmi di stato clear non possono rifiutare la transazione.
- Foreign Apps: elenca altre applicazioni i cui stati globali sono accessibili dalla logica approval e clear. L'accesso è di sola lettura.
- Foreign Assets: elenca le risorse a cui è possibile accedere dal programma di approvazione e dal programma clear-state di questa applicazione. L'accesso è di sola lettura.
- GlobalStateSchema: contiene il numero massimo di valori di stato globali
- LocalStateSchema: contiene il numero massimo di valori di stato locali.

- Extra Program Pages: numero di pagine aggiuntive allocate per l'Approval Program e Clear program.

Capitolo 4

Creare e distribuire NFT

4.1 Creare e distribuire NFT in Algorand

Algorand supporta la tokenizzazione di qualsiasi risorsa, fungibile o non-fungibile. Questo in genere può essere fatto senza scrivere alcuna riga di codice per uno smart contract. Algorand supporta anche smart contract. Questi contratti possono essere utilizzati per creare vari tipi di applicazioni e possono essere integrati o estendere token non fungibili. Algorand fornisce molte funzionalità tra cui Algorand Standard Assets (ASA), Atomic Transfers e smart contracts. Queste funzionalità sono potenti e possono essere configurate per funzionare tra loro in modi unici.

4.1.1 Algorand Standard Asset

Il metodo principale che uno sviluppatore o un utente Algorand può utilizzare per creare e distribuire un NFT consiste nell'utilizzare le funzionalità di ASA. Questo consente di creare un NFT o un token fungibile in pochi secondi senza scrivere alcuno smart contract. Basta inviare una transazione sulla blockchain passando come argomenti le varie proprietà dell'asset che possono essere di due tipi: mutabili ed immutabili.

Proprietà immutabili

Le proprietà immutabili vengono configurate una volta e non possono essere modificate una volta creato l'NFT [24]. I parametri richiesti sono:

- **creator**: il nome dell'utente che vuole creare un asset.
- **total**: il numero di asset che si vogliono coniare.
- **decimal**: che consente la suddivisione di un asset in sue sottounità.
- **DefaultFrozen**: specifica se un asset può essere scambiato o meno. Impostando questo campo a true nessun utente potrà scambiare l'asset nemmeno il creator.

Il numero totale di un asset per coniare un NFT deve essere impostato ad uno ed il campo decimal deve essere impostato a 0. Questo viene fatto per garantire l'unicità

dell'asset, difatti il token creato è unico ed indivisibile. Le proprietà immutabili facoltative includono:

- **Asset Name:** una stringa fino a 32 byte che può essere utilizzata per dare un nome all'asset.
- **UnitName:** una stringa di 8 byte che specifica il nome di un'unità dell'asset (ad esempio USDC)
- **URL:** una stringa di 32 byte che consente di aggiungere un URL all'asset che può essere utilizzato per specificare una posizione fuori catena con dettagli aggiuntivi sull'asset.
- **MetaData Hash:** un campo di 32 byte destinato ad essere utilizzato per memorizzare un hash di alcuni dati fuori catena come il certificato di proprietà.

Come già visto con ethereum potremo utilizzare il campo URL per archiviare alcuni dati off-chain come un file JSON contenente tutte le caratteristiche di un nft (una descrizione , un immagine, ecc...)

Proprietà mutabili

Le proprietà mutabili consistono in un insieme di quattro indirizzi che consentono di configurare un l'NFT. Questi includono :

- **Manager Address** che può essere utilizzato per modificare gli altri tre indirizzi, gestire l'asset o distruggerlo.
- **Reserve Address** che può essere utilizzato per contenere riserve di token non ancora conati.
- **Freeze Address** che può essere utilizzato per bloccare o sbloccare i beni posseduti per un account specifico. Quando un account è bloccato, non può inviare o ricevere l'asset.
- **Clawback Address** che consente di revocare beni da un conto quando questo viola determinati obblighi contrattuali. Ognuno di questi indirizzi può essere modificato anche dopo la creazione dell'asset. Questo può essere fatto se e solo se l'indirizzo dell'account che modifica l'asset coincide con il Manager address.

Principalmente utilizzare solamente le funzionalità offerte da ASA per creare NFT può essere un limite. Questo è dovuto al fatto che l'asset coniato presenta delle caratteristiche standard che non possono essere estese. ASA non consente di coniare collettibile come invece ERC-721 è predisposto. Un esempio può essere lo smart contract che regola il collectible Cryptokitties. Esso implementa anche le funzioni di genetica e di nascita all'interno dello smart contract. Senza estendere ASA con uno smart contract è difficile implementare queste funzionalità. Creando uno NFT con ASA in realtà creiamo un asset fungibile che però avendo una sola unità ed essendo indivisibile può essere considerato un NFT. Questo processo per creare NFT può assumere maggiore credibilità dal momento in cui si inseriscono dei metadati nel capo note o url dell'asset creato. Questo passaggio lo rende molto più simile ad un

NFT standard. Un altro limite è che ogni singolo account può creare con ASA un massimo di 1000 risorse. Questo può essere un problema per utenti particolarmente attivi.

Perciò per estendere una logica potremmo servirci di uno smart contract e combinarlo con ASA.

ASA consente di creare un NFT in pochi secondi senza scrivere alcuna riga di codice. Questo è utile se si ha bisogno di coniare semplici NFT. Inoltre le fees per distribuire questi NFT nella rete blockchain di Algorand sono davvero basse rispetto ad Ethereum.

4.1.2 NFT con smart contract stateless

I contratti stateless possono essere compilati per produrre un indirizzo Algorand, che funziona come qualsiasi altro indirizzo. Questo conto può contenere NFT, asset fungibili o ALGO. Può infatti riceverne liberamente da qualsiasi altro account senza che venga valutata la logica del contratto. Quando viene emessa una transazione dal contratto stateless, la logica viene valutata e, in caso di esito positivo, consentirà il completamento della transazione. Se la logica fallisce, anche la transazione fallisce. Questi tipi di contratti vengono spesso utilizzati per creare conti in stile escrow in cui i prelievi possono essere controllati dalla logica. I contratti stateless possono essere utilizzati come uno qualsiasi degli indirizzi mutevoli che controllano un NFT. Si ricorda che gli indirizzi mutevoli di un asset sono il *clawback address*, il *reserve address*, *freeze address* e il *menager address*. Ad esempio, il reserve address potrebbe essere impostato su un contratto senza stato. Se questo indirizzo viene utilizzato per coniare un NFT, la logica verrà valutata ogni volta che un NFT lascia il conto di riserva. Se la logica fallisce, l’NFT non viene coniato. Se la logica ha successo, l’NFT viene coniato. Lo stesso può essere fatto per congelare e sbloccare un NFT con un contratto stateless. L’indirizzo di clawback NFT può anche essere impostato su un contratto senza stato. Questo consente allo smart contract stateless di revocare un asset da un account e di trasferirlo ad un altro.

4.1.3 NFT con Smart Contract Stateful

I contratti intelligenti con stato possono agire in modo molto simile a Algorand Standard Assets. In effetti, la ricreazione di Algorand Standard Asset è stata eseguita con un contratto stateful. Uno smart contract può modificare le politiche di scambio e di gestione di un NFT. L’implementazione di un NFT in uno Smart Contract Stateful può essere eseguita creando una risorsa univoca con un set di proprietà personalizzate con ASA. Questa risorsa verrebbe quindi archiviata nello stato globale o locale dello smart contract. Questo NFT può quindi essere passato ai singoli account. Questo viene memorizzato nella memoria locale dell’utente. Se questo NFT viene trasferito su un altro account questo viene spostato dall’archivio locale del mittente all’archivio locale del destinatario. Ad esempio, lo stato globale potrebbe essere utilizzato per presentare un NFT come in vendita. Una volta venduto, viene spostato dallo stato globale allo stato locale del destinatario.

Capitolo 5

Creare applicazioni personalizzate con Algorand

Grazie alla combinazione di tecnologie come Atomic Transfer, Stateful e Stateless Smart Contract è possibile creare applicazioni con una logica unica.

5.1 Atomic Transfer

Gli Stateful Smart Contract non approvano di per sé le transazioni di spesa, ma solo le transazioni che riguardano altri contratti intelligenti con stato. Tuttavia possono essere collegati a transazioni di spesa e approvare o rifiutare efficacemente una transazione di spesa. Questo viene fatto usando il trasferimento atomico di Algorand. Nella finanza tradizionale, il trading di asset richiede generalmente un intermediario di fiducia (una banca o una borsa valori) per assicurarsi che entrambe le parti ricevano ciò che è stato concordato. Sulla blockchain di Algorand, questo tipo di scambio è implementato all'interno del protocollo chiamato Atomic Transfer. Ciò significa semplicemente che le transazioni che fanno parte del trasferimento vengono o tutte accettate o tutte rifiutate. I trasferimenti atomici consentono ad utenti di scambiare beni senza la necessità di un intermediario di fiducia, il tutto garantendo che ciascuna parte riceverà ciò che ha concordato.

Le transazioni possono contenere ALGO o Algorand Standard Assets e possono anche essere regolate da smart contracts. I principali casi d'uso di questa funzionalità sono:

- Il pagamento circolare, nel quale un account A paga B se e solo se B paga C e se solo se C paga A.
- Group Payments, tutti devono pagare oppure nessuno paga.
- Scambi decentralizzati: scambiare un asset per un altro senza passare per un intermediario.
- Pagamenti Distribuiti, pagamenti a più destinatari.
- Commissione di pagamento combinate: una transazione paga le fees per altre

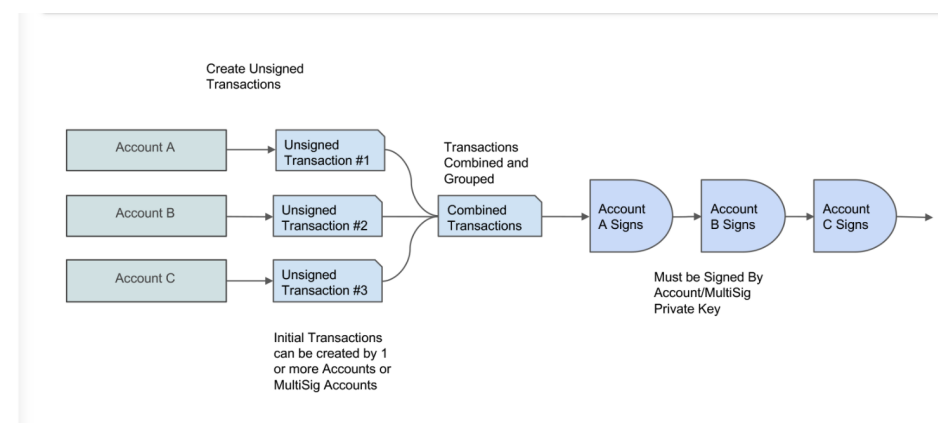


Figura 5.1. La figura illustra come vengono raggruppate le transazioni in un atomic transfer.

Per implementare un trasferimento atomico [2], bisogna generare tutte le transazioni che saranno coinvolte nel trasferimento e quindi raggrupparle insieme. Il risultato del raggruppamento è che a ogni transazione viene assegnato lo stesso ID di gruppo. Una volta che tutte le transazioni contengono questo ID, possono essere suddivise e inviate ai rispettivi mittenti per essere autorizzate. Una sola parte può quindi raccogliere tutte le transazioni autorizzate e inviarle insieme alla rete. La figura 5.1 mostra l'intero processo.

5.2 Collegare stateful a stateless smart contract

La potente funzionalità del Atomic Transfer consente ai contratti intelligenti con stato di rifiutare in modo efficace una transazione di spesa. Infatti, quando si utilizza la funzione di Atomic transfer, i contratti intelligenti senza stato o con stato possono interrogare tutte le proprietà di una qualsiasi transazione nel gruppo. Prendiamo come esempio un'applicazione di crowdfunding. Essa include un indirizzo per depositare i fondi, uno smart contract stateful e l'utilizzo di Atomic transaction. Per depositare i fondi un utente dovrà raggruppare due transazioni, firmarle e inviarle in rete. La prima è una chiamata allo Smart Contract Stateful che verifica che l'importo che si vuole donare sia maggiore dell'importo minimo. La seconda transazione raggruppata è quella di invio del denaro all'indirizzo predefinito. Se una delle due dovesse fallire l'intera operazione verrà abortita. Utilizzando i trasferimenti atomici, è anche possibile raggruppare molti tipi diversi di transazioni, inclusi contratti intelligenti senza stato.

Nell'esempio del crowdfunding, il conto di fondi mostrato nell'esempio precedente potrebbe essere uno smart contract senza stato che funge da deposito a garanzia (Escrow). Le donazioni vengono trattenute in questo conto vincolato fino al termine del periodo di raccolta fondi. La data di fine del fondo è memorizzata nello stato globale di un contratto stateful. Ciò significa che non si desidera che i fondi lascino il contratto stateless fino a quando non è trascorsa la data di fine del fondo. Lo stateless non ha accesso alla data di fine memorizzata nello stato globale del contratto intelligente con stato. Questo può essere aggirato aggiungendo una logica al contratto

di deposito a garanzia [23]. Questa logica verifica che il contratto intelligente con stato venga chiamato nello stesso momento in cui qualcuno sta tentando di trasferire fondi allo stateless. Per fare ciò, il contratto intelligente con stato deve conoscere il contratto senza stato e viceversa. Ci sono diversi modi in cui questo può essere fatto. Quando viene distribuito uno smart contract con stato, viene generato un ID univoco per quel contratto. Questo ID applicazione, come mostra la figura 5.2, può essere verificato nel contratto senza stato.

Vengono raggruppate due transazioni ; la prima è una chiamata allo stateless smart contract e la seconda è una transazione di pagamento verso il deposito di garanzia (donazione). La logica dello stateless verifica che la prima transazione sia una chiamata di Applicazione e che l'id del contratto chiamato sia proprio lo Stateful. Questo garantisce che se non verrà chiamato il contratto corretto l'intero gruppo di transazioni verrà abortito e quindi la donazione non verrà eseguita.

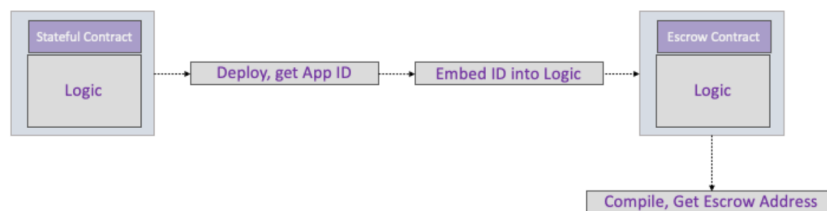


Figura 5.2. La figura illustra il processo per collegare uno smart contract stateful ad uno stateless

Per collegare lo Smart Contract Stateless a quello Stateful sarà un necessario un procedimento simile. La figura 5.2 mostra l'intero processo. E' noto che la logica stateless una volta compilati possono essere identificati tramite un indirizzo. Questo indirizzo verrà archiviato in una variabile globale dello Stateful Smart Contract.

Nella logica del crowdfunding lo Stateful Smart Contract verifica che la prima transazione sia una chiamata al contratto ed infine verifica che la transazione di pagamento abbia come mittente proprio lo smart contract stateless che ha memorizzato nella variabile globale. Per recuperare una variabile globale lo Stateful utilizza l'opcode *app-global-get* e la compara con il mittente della seconda transazione. Se le due transazioni vanno a buon fine avviene la donazione.

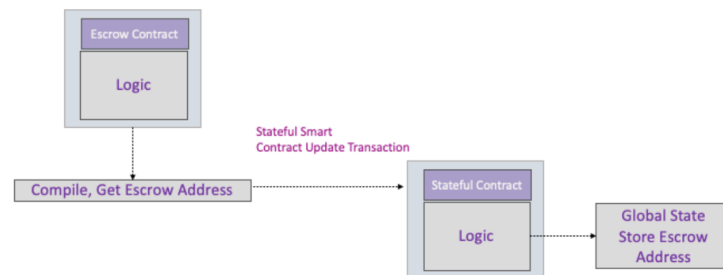


Figura 5.3. La figura illustra il processo per collegare uno smart contract stateless ad uno stateful

Capitolo 6

Logica per la gestione e scambio di NFT

In questo capitolo viene descritto il progetto per la realizzazione di una logica riguardante lo scambio di NFT. Utilizzeremo molte delle informazioni contenute nel capitolo 5 per sviluppare una logica personalizzata che usa tecnologie di Algorand combinate.

6.1 Il problema

Gli NFT sono oggetti utilizzati per gestire i diritti di proprietà dimostrabili di un oggetto digitale oppure di beni fisici unici. Gli NFT hanno uno stretto legame anche con il concetto dei diritti d'autore. I diritti d'autore sono generalmente un riconoscimento che viene dato all'autore relativo ad una determinata opera ogni volta che questa viene, ad esempio, venduta o usata a scopi pubblicitari. In tal modo ogni volta che un NFT viene scambiato tra gli utenti l'autore riceve un guadagno dalla vendita dell'oggetto.

6.2 La Logica

Il progetto prevede la realizzazione di una logica personalizzata per vincolare lo scambio dell'NFT e gestire i diritti d'autore. Questa permetterà lo scambio e la revoca di un asset da un wallet ad un altro quando verranno effettuati due pagamenti. Il primo pagamento rappresenta una donazione che viene effettuata al reward pool.¹ Nella logica viene salvata l'ultima donazione effettuata e solo pagando di più dell'ultima donazione è possibile richiedere l'NFT, tale meccanismo incentiva gli utenti ad effettuare donazioni sempre più elevate. La seconda transazione rappresenta il pagamento dei diritti d'autore al creatore da parte dell'utente che vuole reclamare l'NFT. L'importo versato al creator, per come la logica è stata implementata, dovrà essere minore della donazione. Quindi ogni volta che questo NFT passa da un account ad un altro il creator riceverà una determinata somma di ALGO. Una volta che l'utente effettua i pagamenti richiesti dalla logica questa

¹Reward Pool Address: Indirizzo che viene utilizzato dalla fondazione Algorand per pagare le ricompense ai miners e ai possessori di Algo.

revoca l'asset dall'attuale proprietario e lo trasferisce al nuovo. Inoltre la logica trascrive nella blockchain il nuovo proprietario dell'asset. Infine è stata aggiunta un ulteriore features: una volta che un giocatore reclama l'NFT viene fatto partire un timer. Fino alla scadenza di questo timer nessuno potrà interagire con la logica in tal modo il giocatore avrà a disposizione un tempo maggiore con il suo nuovo asset.

6.3 Usa Case

Questa logica può essere usata per realizzare una serie di applicazioni interessanti.

- La prima analizzata riguarda lo sviluppo di un gioco. Difatti si potrebbe pensare all'NFT come un oggetto magico che permette di avere un riconoscimento nel gioco. I giocatori entreranno in competizione per contendersi l'NFT, chi lo possiede potrà accedere a funzionalità uniche come ad esempio una armatura speciale oppure un baule magico in un videogioco.
- Un altro utilizzo di questa logica è la realizzazione di una applicazione per la beneficenza (crowdfunding). In questo caso l'indirizzo di reward pool address dovrà essere sostituito con l'indirizzo di una fondazione. I vari finanziatori potrebbero essere incentivati a donare poichè l'applicazione mette a disposizione del donatore più generoso un NFT. L'asset conteso dai vari partecipanti dell'applicazione potrebbe essere realizzato da un artista famoso. Questo potrebbe incentivare ulteriormente i finanziatori a possedere il bene facendo una donazione.

6.4 Requisiti

La progettazione del gioco inizia andando a definire i personaggi e le varie funzionalità che esso comprende. Gli attori principali sono :

- Gli utenti che interagiscono con l'applicazione per contendersi un oggetto che in questo caso è un NFT.
- L'NFT stesso che consente agli utenti di detenere un titolo speciale.
- Il clawback account rappresentato dallo stateless smart contract revoca l'NFT dal vecchio proprietario all'attuale proprietario.

. I requisiti per il minigioco sono i seguenti :

- L'applicazione deve consentire agli utenti che vogliono reclamare il titolo di fare una donazione per possedere l'Nft.
- L'applicazione deve bloccare una donazione se il timer della vecchia donazione non è ancora scaduto.
- L'applicazione deve bloccare la donazione se il creator dell'NFT non ha ricevuto una pagamento dei diritti. L'applicazione deve revocare il bene dall'attuale proprietario al nuovo proprietario solo se l'account che l'ha richiesto ha effettuato correttamente il pagamento.

6.5 Architettura

L'obiettivo del progetto è vincolare lo scambio di NFT attraverso la combinazione di Atomic Transfer, Smart Contract Stateful e Stateless, Algorand Standard Asset .

- Algorand Standard Assets : implementa l'NFT controllato dalla logica. La revoca dell'NFT viene gestita dallo Smart Contract Stateless. Inserendo nell'indirizzo di clawback dell'asset quello dello smart contract consentirà alla logica di avere il pieno controllo dell'asset e revocarlo da un account per trasferirlo in un altro.³
- Atomic Transfer, forza l'interazione simultanea tra smart contract stateful e stateless. Permette di combinare la logica stateless che mantiene una memoria nella blockchain con una logica stateful che consente di approvare delle transazioni di spesa.
- Stateful Smart Contract, implementa la logica dell'intera applicazione. Gestisce lo stato globale che tiene traccia degli importi delle donazioni, l'attuale possessore del titolo e il timer.
- Stateless Smart Contract Verifica se tutte le donazioni sono state eseguite correttamente. Approva le transazioni di clawback NFT in base alla politica dei diritti d'autore. Permette di trasferire quindi l'asset da un account ad un altro.

Le dipendenze tra le funzionalità Algorand implicano una sequenza di distribuzione specifica per i componenti richiesti:

- Distribuzione dell'applicazione stateful.
- Creazione dell' NFT.
- Distribuzione dell'account del contratto senza stato.
- Configurazione dell'NFT.
- Configurazione dell'applicazione con stato.

6.5.1 Componente stateful

Lo scopo dello smart contract stateful è quello di leggere e scrivere variabili globali nella blockchain. In figura si possono visionare tutte le variabili globali che vengono utilizzate dallo stateful smart contract.

- "Algolaw": viene inserito all'interno di questa variabile l'indirizzo dello Stateless Smart Contract in modo da collegare i due contratti e quindi combinare le due logiche.

³Il clawback address rappresenta un account a cui è consentito trasferire asset da e verso qualsiasi titolare di asset. Viene usato generalmente per revocare beni da un conto. Permette di avere il pieno controllo sull'asset

Nome	Dominio	Tipo
AlgoLaw	globale	ByteSlice
Donations	globale	uint
Royalty	globale	uint
Round	globale	uint
Owner	globale	ByteSlice

Figura 6.1. Le variabili globali utilizzate dallo smart contract stateful.

- "Donations": Variabile globale che viene modificata ogni volta che un nuovo utente vuole fare un'offerta più generosa della precedente rivendicando il titolo di "Owner".
- Royalty: Indica l'ammontare (in percentuale rispetto alla donazione) di Algo che il creatore dell'asset si aspetta, ogni volta che un utente vuole reclamare l'NFT. L'ammontare deve essere superiore al 10 percento dell'importo donato.
- "Round": indica il numero di blocchi validati nella blockchain dalla creazione di Algorand. Questo contatore serve per aumentare il tempo del gioco. Difatti ogni volta che un giocatore reclama l'NFT e ne prende il possesso, nessuno per 1000 round potrà fare una donazione e reclamare il titolo. Un round ha una durata di circa 4,5 sec. Facendo i calcoli un giocatore non può reclamare il titolo prima di un ora dall'ultimo reclamo.
- Owner: è la variabile nella quale viene inserito l'attuale possessore dell'NFT.

Lo Stateful Smart Contract è composto da due programmi : L'Approval Program e il Clear Program. Entrambi i programmi sono descritti nell'appendice A.

Approval program

Consente agli utenti di interagire con lo smart contract in due modi:

- Creazione del gioco, impostando le variabili globali.
- Effettuando azione sul gioco come rivendicare il titolo.

Nella prima parte del programma vengono inizializzate le variabili globali e locali all'interno dello smart contract.

Si inizializzano gli importi delle donazioni, l'attuale possessore dell'NFT, il global round e le tasse per i diritti d'autore. Si imposta il valore 10 nel valore Royalties perchè colui che reclama l'NFT dovrà pagare il 10 percento della cifra donata al creator. Infine viene inserito il global round attuale. Quando lo smart contract verrà distribuito in rete attraverso la transazione di creazione, verranno passate come argomenti anche le variabili globali. All'interno della logica l'inizializzazione delle variabili viene eseguito grazie all'opcode app-global-put.

Esistono due Application Call che possono essere fatte allo smart contract.

- La prima chiamata è rappresentata da una sola transazione. In questa transazione l'indirizzo dello Stateless Smart Contract viene inserito nella variabile globale "AlgoLaw". Questo è il modo in cui il contratto stateless viene collegato alla logica stateful. La logica TEAL garantisce che una volta inserito nel contratto l'indirizzo nessuno può più modificarlo.
- La seconda chiamata rappresenta la richiesta del NFT e quindi del titolo attraverso le donazioni.

Le transazioni da effettuare sono 4 e vengono raggruppate dall'Atomic Transfer firmate e inviate insieme nella rete. La logica TEAL verifica se le transazioni raggruppate sono del tipo che ci si aspetta e nell'ordine corretto. La prima è una chiamata all'applicazione (Application Call) In questa transazione viene passato come argomento una stringa "Chair". In questo modo la logica dello Stateful Smart Contract comprende se si tratta di una transazione per la richiesta dell'Nft. La seconda è una transazione di tipo *pay* verso il reward pool. La terza è una altra transazione di tipo *pay* verso il creator per il pagamento dei diritti di autore. Infine la quarta è una transazione di tipo *transfer* per il trasferimento dell'NFT dall'attuale proprietario al nuovo. La logica teal dello smart contract verifica se durante la seconda chiamata il global round attuale sia superiore di quello precedentemente memorizzato. Aggiorna quindi il valore globale Round con il Round attuale. Poi verifica che la donazione al reward pool sia maggiore dell'ultima effettuata e aggiorna la variabile globale al nuovo importo. Infine viene verificato che la seconda donazione, quella al creator, sia il 10 per cento della prima.

Clear Program

Il Clear Program approverà semplicemente qualsiasi "Clear Call" da parte degli utenti per rimuovere l'applicazione dal saldo dei loro conti . Visto che non vengono implementate variabili locali nell' Approval Program questo non viene utilizzato.

6.5.2 Creazione NFT

Una volta implementato il contratto stateful e distribuito nella rete viene creato l'NFT. La creazione e la distribuzione avviene attraverso Algorand Standard Asset [19]. L'asset in questione presenta una sola unità e 0 decimals, questo perchè un NFT è indivisibile ed unico. Inoltre viene inserito nel campo *url* un puntatore ad un file Json. Questo file contiene il titolo e la descrizione dell'opera che si vuole digitalizzare e un altro puntatore ad un immagine. Le proprietà immutabili dell'asset non vengono modificate durante sua la creazione. Difatti tutti e quattro gli indirizzi sono impostati sul creator dell'NFT. L'asset viene impostato su "freeze". Questo significa che non potrà essere scambiato con nessun altro nella rete. Solo l'indirizzo di clawback potrà revocarlo e quindi trasferirlo da un account ad un altro.

6.5.3 Stateless Smart contract

Una volta creato e distribuito l'NFT si passa allo stateless smartcontract. La componente statless ha la funzionalità di approvare le transazioni di spesa durante la richiesta dell'NFT. La logica dello stateless controlla che le transazioni raggruppate

siano 4 e che siano nell'ordine corretto. La logica controlla che l'id dello stateful smartcontract chiamato nella prima transazione coincida con il nostro Smart Contract Stateful. In questo modo viene collegata la logica stateful a quella stateless. Inoltre la logica controlla anche che il primo pagamento venga effettuato all'indirizzo del reward pool, il secondo al creator e che nella quarta transazione l'id dell'NFT che dovrà essere trasferito sia quello creato poco prima per questa applicazione. Una volta compilato, lo Stateless Smart Contract svolge una particolare funzionalità; funge da clawback address per l' NFT. [20] Infatti vengono modificate le proprietà mutabili dell'asset. Il Manger Address, il Freeze Address e il Reserve Address vengono lasciati vuoti. Questo significa che nessun altro potrà più modificare le proprietà mutabili dell'asset. Il Clawback Address invece, viene aggiornato con l'indirizzo dello smartcontract stateless. Questo permette allo smartcontract di controllare e approvare il trasferimento dell'asset. Sarà lo stateless smart contract a firmare con la Logic sig la transazione di revoca dell'NFT da un account ad un altro.

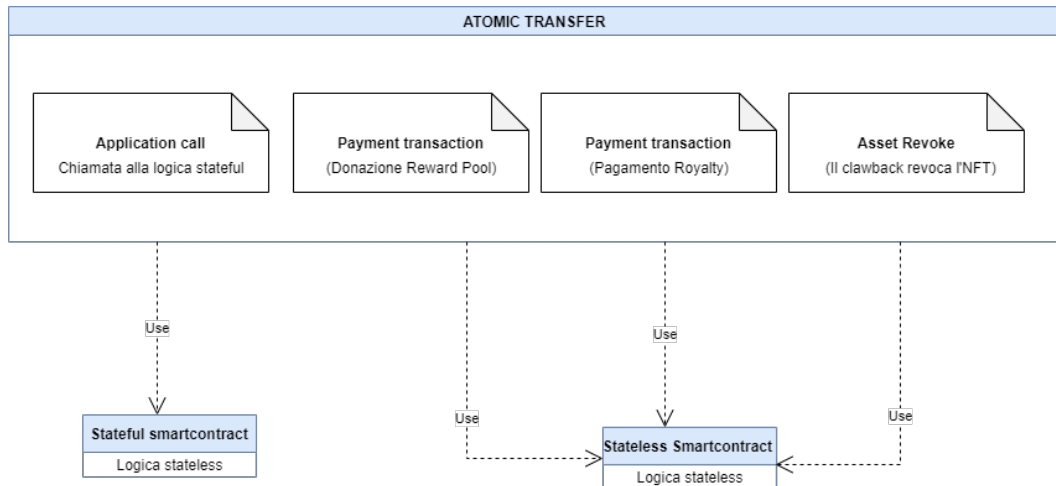


Figura 6.2. La figura illustra il ruolo dell'atomic transfer che raggruppa le varie transazioni. L'Application call viene controllata dalla logica stateful mentre le due transazioni di pagamento e la transazione di trasferimento vengono controllate dalla logica stateless.

6.6 Implementazione

L'applicazione stateful, l'NFT e lo Smart Contract Stateless vengono implementati e distribuiti grazie al plug-in Algodea dell'editor IntelliJ. Utilizzando questo plug-in, gli sviluppatori possono creare e testare contratti intelligenti sia senza stato che con stato direttamente dal loro IDE IntelliJ. Alcune delle altre funzionalità chiave supportate da questo plug-in sono il supporto dell'editor per i file TEAL e PyTeal, la gestione ASA, i trasferimenti atomici, la gestione di account di prova. L'obiettivo di questo plug-in è aumentare la produttività degli sviluppatori semplificando varie interazioni con la blockchain di Algorand. [22] Per la configurazione del nodo si utilizza un servizio chiamato *Pure stake*. Questo permette di collegarsi immediatamente ad un nodo Algorand senza eseguirlo dal proprio device.

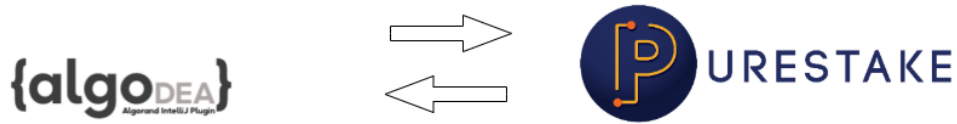


Figura 6.3. La figura illustra le tecnologie utilizzate per distribuire e chiamare la logica realizzata.

Capitolo 7

Conclusione e sviluppi futuri

Il lavoro svolto inizia con la definizione di un Non-Fungible-Token con il fine di averne una più chiara comprensione. Una parte è poi dedicata ad un'analisi più approfondita degli NFT e della loro creazione e gestione ad opera degli smart contracts.

In seguito sono stati studiati ed analizzati vari tipi di smart contract relativi alle due blockchain analizzate: Ethereum e Algorand.

Dopodiché sono stati approfonditi vari metodi per creare e gestire NFT su diversi sistemi blockchain. Il fine è stato quello di mostrare quale tra questi metodi fosse adatto all'obiettivo di realizzare una logica personalizzata per la gestione di NFT. A causa dell'alta scalabilità, del suo throughput elevato e di un alto livello di sicurezza e decentralizzazione, la blockchain Algorand si è rivelata la miglior soluzione per la realizzazione della nostra logica.

Lo studio approfondito di Atomic Transfer, degli Algorand Standard Assets, dei smart contract e del linguaggio TEAL ha reso possibile l'implementazione di una logica personalizzata per la gestione e lo scambio di un NFT. Questa logica, una volta distribuita nella rete, consente agli utenti reclamare un NFT semplicemente donando alcuni ALGO e pagando i diritti d'autore al creator. Se le donazioni vanno a buon fine la logica revoca l'asset dal attuale al nuovo proprietario e il nome del nuovo proprietario nella blockchain.

Il proseguo di questa tesi è volto all'implementazione di una dApp per la realizzazione di uno dei casi d'uso esposti. Realizzare applicazioni decentralizzate di questo genere ha come obiettivo portare più utenti, attirati dal mondo degli NFT, ad approdare alla blockchain di Algorand. Difatti lo scopo di questa logica è di essere implementata in un numero maggiore di contesti distinti.

Appendice A

Componente stateful

A.1 clear state

```
#pragma version 2
int 1
```

A.2 Approval program

```
pragma version 2
txn ApplicationID
int 0
==
bnz algoapp_creation
txn OnCompletion
int NoOp
==
bnz algocall
int 0
return
algoapp_creation:
byte "DepartmentChief"
byte "Mei"
app_global_put
byte "ChairDonation"
int 0
app_global_put
byte "Round"
int 0
app_global_put
byte "Royalty"
int 10
app_global_put
int 1
```

```

return
b end_contract
algocall:
global GroupSize
int 1
==
bnz algo_law
global GroupSize
int 4
==
bnz algonft_claims
int 0
return
b end_contract
algo_law:
int 0
byte "AlgoLaw"
app_global_get_ex
store 0
store 1
load 0
bnz promulgate_law_failure
byte "AlgoLaw"
txna ApplicationArgs 0
app_global_put
b promulgate_law
promulgate_law_failure:
int 0
return
b end_contract
promulgate_law:
int 1
return
algonft_claims:
gtxn 0 TypeEnum
int appl
==
gtxn 0 NumAppArgs
int 2
==
&&
gtxn 1 TypeEnum
int pay
==
&&
gtxn 2 TypeEnum
int pay

```

```
==
&&
gtxn 3 TypeEnum
int axfer
==
&&
gtxn 3 Sender
byte "AlgoLaw"
app_global_get
==
&&
bnz claimsnft
int 0
return
b end_contract
claimsnft:
gtxna 0 ApplicationArgs 0
byte "Chair"
==
byte "Round"
app_gloabal_get
int 1000
+
store 2
global round
load 2
>
&&
bnz claim_nft
int 0
return
b end_contract
claim_nft:
gtxn 1 Amount
byte "ChairDonation"
app_global_get
>
gtxn 1 Amount
byte "Royalty"
app_global_get
/
store 3
load 3
gtxn 2 Amount
==
&&
bnz nft_owner
```

```
int 0
return
b end_contract
nft_owner:
byte "DepartmentChief"
gtxna 0 ApplicationArgs 1
app_global_put
byte "ChairDonation"
gtxn 1 Amount
app_global_put
byte "Round"
global Round
app_global_put
int 1
return
end_contract:
```

Appendice A

Componente stateless

```
#pragma version
global GroupSize
int 4
==
gtxn 0 TypeEnum
int appl
==
gtxn 0 ApplicationID
int APP_ID
==
&&
&&
gtxn 1 TypeEnum
int pay
==
gtxn 1 Sender
gtxn 0 Sender
==
&&
gtxn 1 Receiver
addr 7377777777777777777777777777777777777777777777777777777UFEJ2CI
==
&&
&&
gtxn 2 TypeEnum
int pay
==
gtxn 1 Sender
gtxn 2 Sender
==
&&
gtxn 2 Reciver
addr CREATOR_ADDRESS
```

```
==
&&
&&
gtxn 3 TypeEnum
int axfer
==
gtxn 3 AssetReceiver
gtxn 1 Sender
==
&&
gtxn 3 XferAsset
int ID_ASSET
==
&&
gtxn 3 AssetAmount
int 1
==
&&
gtxn 3 Fee
int 1000
<=
&&
gtxn 3 AssetCloseTo
global ZeroAdress
==
&&
gtxn 3 RekeyTo
global ZeroAddress
==
&&
&&
```


Bibliografia

- [1] Anatomy of smart contracts. Available from: <https://ethereum.org/en/developers/docs/smart-contracts/anatomy/>.
- [2] Atomic transfers. Available from: https://developer.algorand.org/docs/features/atomic_transfers/.
- [3] Cryptopunks. Available from: <https://www.larvalabs.com/cryptopunks>.
- [4] Eip-721: Non-fungible token standard. Available from: <https://eips.ethereum.org/EIPS/eip-721>.
- [5] Introduction to smart contracts. Available from: <https://ethereum.org/en/developers/docs/smart-contracts/>.
- [6] Smart contract. Available from: https://en.wikipedia.org/wiki/Smart_contract.
- [7] The smart contract language. Available from: <https://developer.algorand.org/docs/features/asc1/teal/>.
- [8] Standard token erc-20. Available from: <https://ethereum.org/it/developers/docs/standards/tokens/erc-20/>.
- [9] Transaction reference. Available from: <https://developer.algorand.org/docs/reference/transactions/>.
- [10] Un'opera "digitale" è stata venduta all'asta per 69,3 milioni di dollari. Available from: <https://www.agi.it/cultura/news/2021-03-12/opera-digitale-beeple-venduta-asta-69-milioni-dollari-11736373/>.
- [11] Mode of use (2020). Available from: <https://developer.algorand.org/docs/features/asc1/stateless/modes/>.
- [12] Overview (2020). Available from: <https://developer.algorand.org/docs/features/asc1/stateful/>.
- [13] Smart contract overview (2020). Available from: https://developer.algorand.org/docs/features/asc1/?from_query=state#stateless-smart-contracts.

- [14] Cryptokitties: Collectible and breedable cats empowered by blockchain technology (2021). Available from: https://drive.google.com/file/d/1soo-eAaJHzhw_XhFGMJp3VNcQoM43byS/view.
- [15] Gli nft, spiegati (2021). Available from: <https://www.ilpost.it/2021/03/19/nft-non-fungible-token/>.
- [16] Non-fungible tokens (nft) (2021). Available from: <https://ethereum.org/en/nft/>.
- [17] Sia e rappresenta i diritti degli autori con asset digitali: Creati piÙ di 4.000.000 di nft sull'infrastruttura blockchain di algorand (2021). Available from: <https://www.siae.it/it/iniziative-e-news/siae-rappresenta-i-diritti-degli-autori-con-asset-digitali-creati-pi%C3%B9-di-4000000>.
- [18] AQUARO, D. Smart contract: cosa sono e come funzionano le clausole su blockchain. Available from: <https://www.ilsole24ore.com/art/smart-contract-cosa-sono-e-come-funzionano-clausole-blockchain-ACsDo2P>.
- [19] BASSI, C. Create and manage a non-fungible asset from the command line using goal. Available from: <https://developer.algorand.org/tutorials/create-and-manage-non-fungible-asset-command-line-using-goal/>.
- [20] BASSI, C. Algorealm, a nft royalty game (2021). Available from: <https://developer.algorand.org/solutions/algorealm-nft-royalty-game/>.
- [21] ERMISINO, M. Nft e musica, perché sono fatti l'uno per l'altra (2021). Available from: <https://www.wired.it/play/musica/2021/07/03/nft-musica-funzionano/>.
- [22] RANJAN, S. Making development easier with algodea intellij plugin (2021). Available from: <https://developer.algorand.org/articles/making-development-easier-algodea-intellij-plugin/>.
- [23] WEATHERSBY, J. Linking algorand stateful and stateless smart contracts (2020). Available from: <https://developer.algorand.org/articles/linking-algorand-stateful-and-stateless-smart-contracts/>.
- [24] WEATHERSBY, J. Building nfts on algorand (2021). Available from: <https://developer.algorand.org/articles/building-nfts-on-algorand/>.