

Laborator 5

ERC-721

Ce este ERC-721?

Un standard care descrie modul în care token-urile non-fungible sunt construite utilizând blockchain-uri compatibile EVM (mașină virtuală ethereum - EVM).

Ce este un token *non-fungible*?

non-fungible = nefungibil

fungibil = *Care poate fi înlocuit cu altul de același fel, de aceeași calitate și aceeași cantitate, în cazul când formează obiectul unei obligații.*¹

Unde se utilizează token-urile non-fungible sau NFT (Non-Fungible Tokens)?

- Artă digitală sau artă fizică
- Jocuri
- Imobiliare
- Finanțe
- Licențe software
- Tichete la concerte sau activități sportive

ERC-721 definește funcții care sunt conforme cu ERC-20.

Se recomandă parcurgerea următoarei surse a standardului: <https://eips.ethereum.org/EIPS/eip-721>, astfel implementarea și modul de implementare este mult mai ușor.

După cum am văzut în ERC20, următoarele funcții și proprietăți sunt necesare pentru a înțelege mecanismele de bază ale ERC-721, funcții și proprietăți precum:

- `name` – utilizat pentru a identifica token-ul după nume, astfel încât alte contracte și aplicații pot să-l identifice.
- `symbol` – folosit pentru a identifica un ID sau simbol pentru token.
- `totalSupply` – funcția este definită pentru definirea numărului total de token-uri din blockchain
- `balanceOf` – returnează numărul de NFT-uri deținute de către o adresă.

Funcții de ownership (proprietate)

- `ownerOf` – funcția returnează adresa proprietarului unui token. Fiecare token ERC-721 este unic și nefungibil, sunt reprezentați în blockchain prin ID. Alți utilizatori, contracte, aplicații pot să utilizeze ID-ul pentru a determina proprietarul token-ului.

¹ <https://dexonline.ro/definitie/fungibil>

- `approve` – funcția acordă sau aprobă o altă entitate permisiunea de a transfera token-uri ca fiind din partea proprietarului.
- `takeOwnership` – este o funcție opțională care se comportă ca o funcție de extragere (`withdraw`) întrucât o entitate externă poate să invoce funcția pentru a lua tokenii din contul altui utilizator. Prin urmare, funcția `takeOwnership` poate fi utilizată când un utilizator a fost aprobat să dețină o anumită cantitate de token-uri și dorește să extragă alte token-uri din balanța utilizatorului.
- `transfer` – un alt tip de funcție de transfer care permite transferul token-ului proprietarului către alt utilizator.
- `tokenOfOwnerByIndex` – este o funcție opțională dar și recomandată în același timp. Fiecare proprietar poate să dețină mai mult de un NFT în același timp. Identificatorul unic identifică fiecare NFT, și eventual, poate să devină foarte dificil să monitorizeze toate ID-urile. Așadar, contractul stochează aceste ID-uri într-un vector și funcția `tokenOfOwnerByIndex` ne permite să obținem aceste informații din vector.

Funcții de metadata

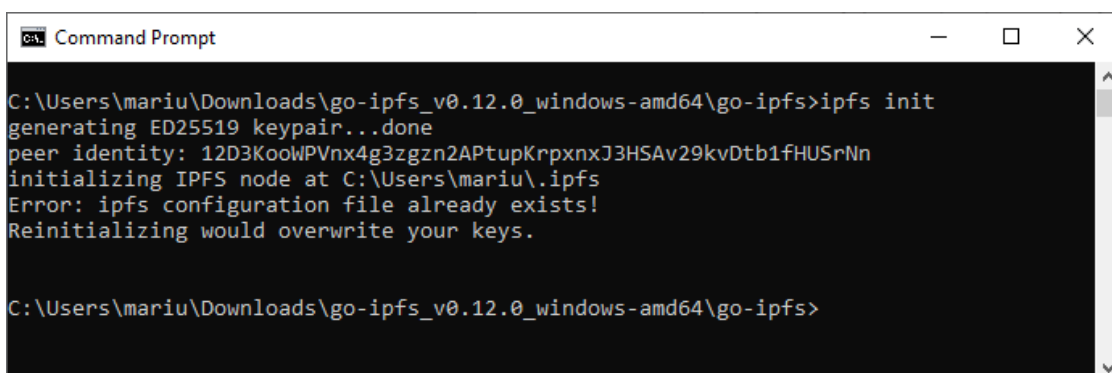
- `tokenMetadata` – o caracteristică opțională care este reprezentată de o interfață care ne permite să vizualizăm metadatale token-urilor sau un link la datele sale.

Evenimente

- `Transfer` – acest eveniment este lansat când proprietarul token-ului se schimbă de la un individ la altul. Emite informația asupra contului care transferă token-ul, ce cont a primit token-ul, și care token (în funcție de ID) a fost transferat.
- `Approve` – acest eveniment este lansat când un utilizator aprobă ca alt utilizator să dețină dreptul de proprietate al token-ului. Evenimentul este lansat când funcția de aprobare este invocată. Returnează informația asupra căruia contului curent care deține token-ul, care cont este aprobat pentru a obține dreptul de proprietate al token-ului, și ce token (în funcție de ID) este aprobat pentru a avea drept de proprietate pentru transfer.

Aplicație de laborator

1. Urmați Anexa 4 pentru instalarea IPFS
2. După instalare deschideți CMD/Terminal
3. Executați comanda `ipfs init` – aceasta creează un repository IPFS



```
C:\Users\mariu\Downloads\go-ipfs_v0.12.0_windows-amd64\go-ipfs>ipfs init
generating ED25519 keypair...done
peer identity: 12D3KooWPVnx4g3zgzn2APtupKrpXnxJ3HSAv29kvDtb1fHUSrNn
initializing IPFS node at C:\Users\mariu\.ipfs
Error: ipfs configuration file already exists!
Reinitializing would overwrite your keys.

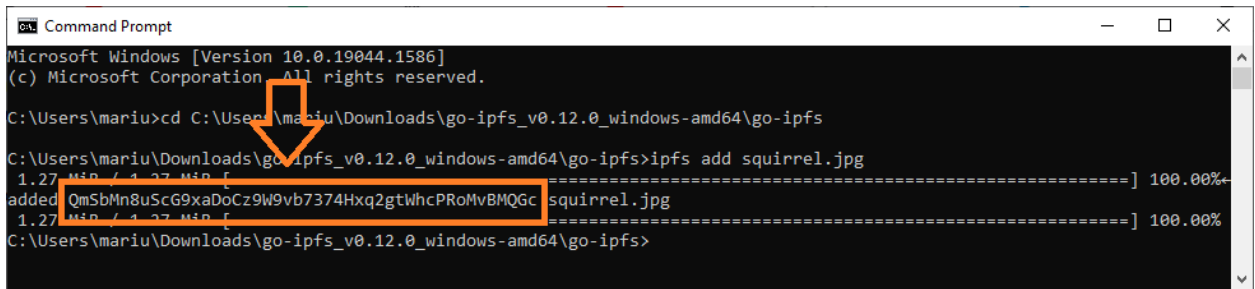
C:\Users\mariu\Downloads\go-ipfs_v0.12.0_windows-amd64\go-ipfs>
```

4. Rulați comanda ipfs daemon

```
Command Prompt - ipfs daemon

C:\Users\mariu\Downloads\go-ipfs_v0.12.0_windows-amd64\go-ipfs>ipfs daemon
Initializing daemon...
go-ipfs version: 0.12.0
Repo version: 12
System version: amd64/windows
Golang version: go1.16.12
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/127.0.0.1/udp/4001/quic
Swarm listening on /ip4/169.254.163.115/tcp/4001
Swarm listening on /ip4/169.254.163.115/udp/4001/quic
Swarm listening on /ip4/169.254.50.126/tcp/4001
Swarm listening on /ip4/169.254.50.126/udp/4001/quic
Swarm listening on /ip4/169.254.55.218/tcp/4001
Swarm listening on /ip4/169.254.55.218/udp/4001/quic
Swarm listening on /ip4/172.29.128.1/tcp/4001
Swarm listening on /ip4/172.29.128.1/udp/4001/quic
Swarm listening on /ip4/192.168.100.2/tcp/4001
Swarm listening on /ip4/192.168.100.2/udp/4001/quic
Swarm listening on /ip4/192.168.100.3/tcp/4001
Swarm listening on /ip4/192.168.100.3/udp/4001/quic
Swarm listening on /ip4/192.168.139.1/tcp/4001
Swarm listening on /ip4/192.168.139.1/udp/4001/quic
Swarm listening on /ip4/192.168.45.1/tcp/4001
Swarm listening on /ip4/192.168.45.1/udp/4001/quic
Swarm listening on /ip4/192.168.56.1/tcp/4001
Swarm listening on /ip4/192.168.56.1/udp/4001/quic
Swarm listening on /ip6/2a02:2f0a:7218:2a00:159e:4348:d833:2064/tcp/4001
Swarm listening on /ip6/2a02:2f0a:7218:2a00:159e:4348:d833:2064/udp/4001/quic
Swarm listening on /ip6/2a02:2f0a:7218:2a00:6d6a:bedc:6428:3af8/tcp/4001
Swarm listening on /ip6/2a02:2f0a:7218:2a00:6d6a:bedc:6428:3af8/udp/4001/quic
Swarm listening on /ip6/2a02:2f0a:7218:2a00:d1b7:e79a:61f1:1513/tcp/4001
Swarm listening on /ip6/2a02:2f0a:7218:2a00:d1b7:e79a:61f1:1513/udp/4001/quic
Swarm listening on /ip6/2a02:2f0a:7218:2a00:f060:b877:6b05:af33/tcp/4001
Swarm listening on /ip6/2a02:2f0a:7218:2a00:f060:b877:6b05:af33/udp/4001/quic
Swarm listening on /ip6:::1/tcp/4001
Swarm listening on /ip6:::1/udp/4001/quic
Swarm listening on /p2p-circuit
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/127.0.0.1/udp/4001/quic
Swarm announcing /ip4/188.26.47.44/udp/4001/quic
Swarm announcing /ip4/192.168.100.2/tcp/4001
Swarm announcing /ip4/192.168.100.2/udp/4001/quic
Swarm announcing /ip6/2a02:2f0a:7218:2a00:f060:b877:6b05:af33/tcp/4001
Swarm announcing /ip6/2a02:2f0a:7218:2a00:f060:b877:6b05:af33/udp/4001/quic
Swarm announcing /ip6:::1/tcp/4001
Swarm announcing /ip6:::1/udp/4001/quic
API server listening on /ip4/127.0.0.1/tcp/5001
WebUI: http://127.0.0.1:5001/webui
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

5. Deschideți un alt terminal și rulați comanda `ipfs add squirrel.jpg`. Veți avea nevoie de hash-ul marcat.



```
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mariu>cd C:\Users\mariu\Downloads\go-ipfs_v0.12.0_windows-amd64\go-ipfs

C:\Users\mariu\Downloads\go-ipfs_v0.12.0_windows-amd64\go-ipfs>ipfs add squirrel.jpg
1.27 MiB / 1.27 MiB [-----] 100.00%
added QmSbMn8uScG9xaDoCz9W9vb7374Hxq2gtWhcPRoMvBMQGc squirrel.jpg
1.27 MiB / 1.27 MiB [-----] 100.00%
C:\Users\mariu\Downloads\go-ipfs_v0.12.0_windows-amd64\go-ipfs>
```

Hash: `QmSbMn8uScG9xaDoCz9W9vb7374Hxq2gtWhcPRoMvBMQGc`

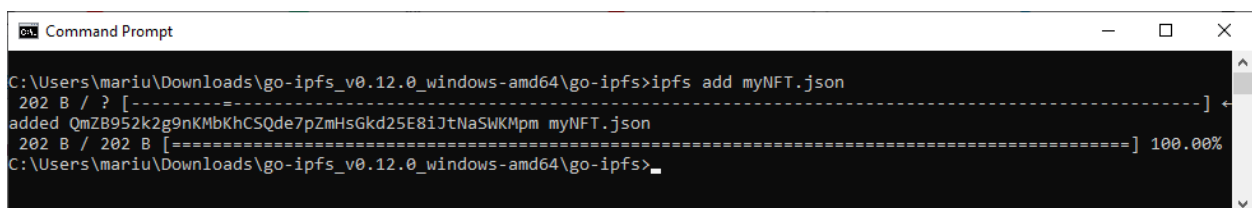
6. Copiați hash-ul respectiv și adăugați hash-ul la link astfel încât să obțineți un link asemănător cu cel de jos:

<https://ipfs.io/ipfs/QmSbMn8uScG9xaDoCz9W9vb7374Hxq2gtWhcPRoMvBMQGc>

7. Creați următorul fișier json și salvați fișierul cu numele `myNFT.json`

```
{
  "name": "Exemplu NFT - Veverita",
  "description": "Importanta unei veverite in procesul de impadurire",
  "image":
  "https://ipfs.io/ipfs/QmSbMn8uScG9xaDoCz9W9vb7374Hxq2gtWhcPRoMvBMQGc",
}
```

8. Acum în CMD rulați comanda `ipfs add myNFT.json`



```
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mariu\Downloads\go-ipfs_v0.12.0_windows-amd64\go-ipfs>ipfs add myNFT.json
202 B / ? [-----] 100.00%
added QmZB952k2g9nKmbKhCSQde7pZmHsGkd25E8iJtNaSWKMpm myNFT.json
202 B / 202 B [=====] 100.00%
C:\Users\mariu\Downloads\go-ipfs_v0.12.0_windows-amd64\go-ipfs>
```

9. Creați propriul token prin crearea unui fișier `tokenSquirrel.sol` cu următorul conținut:

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity 0.8.0;
3
4 import "https://github.com/0xcert/ethereum-erc721/src/contracts/tokens/nf-
5 token-metadata.sol";
6 import "https://github.com/0xcert/ethereum-
7 erc721/src/contracts/ownership/ownable.sol";
8
9 contract newNFT is NFTTokenMetadata, Ownable {
10
```

```
11 constructor() {
12     nftName = "NFT Squirrel";
13     nftSymbol = "SQUI";
14 }
15
16 function mint(address _to, uint256 _tokenId, string calldata _uri)
17 external onlyOwner {
18     super._mint(_to, _tokenId);
19     super._setTokenUri(_tokenId, _uri);
20 }
21
22 }
```

10. Rulați noul token în remix.ethereum.org

11. Selectați la Environment modul Injected Web3

12. Confirmați dacă reușiți să vă obțineți ETH. La momentul respectiv am rămas în pană de ETH și sunt în așteptare de ETH de la ropsten.

