Laborator 4

ERC20

Orice contract care respectă *standardul ERC20*[1] este un token ERC20.

Token-urile ERC20 furnizează următoarele funcționalități pentru transferul token-urilor și permiterea altor entități să transfere token-uri în numele proprietarului token-ului.

O interfață conform cu standardul are următoare structură (este o structură generală care poate să aibe diferite ajustări în funcție de necesități).

```
 1 // SPDX-License-Identifier: MIT
 2 pragma solidity ^0.8.10;
 3
 4 interface IERC20 {
 5     function totalSupply() external view returns (uint);
 6
 7     function balanceOf(address account) external view returns (uint);
 8
 9     function transfer(address recipient, uint amount) external returns (bool);
10
11     function allowance(address owner, address spender) external view returns
12 (uint);
13
14     function approve(address spender, uint amount) external returns (bool);
15
16     function transferFrom(
17         address sender,
18         address recipient,
19         uint amount
20     ) external returns (bool);
21
22     event Transfer(address indexed from, address indexed to, uint value);
23     event Approval(address indexed owner, address indexed spender, uint value);
24 }
```

Linia 5 – funcția *totalSupply()* – returnează cantitatea (amount-ul) de token-uri

Linia 7 – funcția *balanceOf(address account)* – returnează cantitatea de token-uri deținută de cont (*account*)

Linia 9 – funcția *transfer(address recipient, uint amount)* – mută cantitatea de token-uri (declarată în variabila *amount*) din contul *recipient.* Returnează o valoarea de adevăr dacă operația a fost realizată cu succes sau nu.

Linia 11 – funcția *allowance(address owner, address spender) external view returns (uint)* – returnează diferența de token-uri pe care *spender*-ul are posibilitatea să le cheltuiască din partea

[1] Standardul ERC20 - https://eips.ethereum.org/EIPS/eip-20

proprietarului (*owner*) prin intermediul *{transferForm}*. Valoarea se schimbă când *{approve}* sau *{transferForm}* sunt invocate.

Linia 14 – funcția *approve(address spender, uint amount) external returns (bool)* – setează cantitatea (amount) ca permisiune (allowance) a celui care care cheltuie (spender) pentru token-urile inițiatorului (caller-ului). O valoare de adevăr este returnată indicând dacă operația a avut loc cu succes sau nu. Un eveniment *{Approval}* este realizat.

Linia 16 – funcția *transferForm(address sender, address recipient, uint256 amount) external returns (bool)* – mută cantitatea (amount-ul) de la expeditor *(sender)* la recipient *(recipient)* folosind mecanismul implementat în Linia 11 prin intermediul funcției *allowance.* Cantitatea este dedusă *(amount)* din valoarea permisă (allowance). Returnează o valoare de adevăr care să indice dacă operația a fost realizată cu succes sau nu. Un eveniment {Transfer} este emis.

Linia 22 – evenimentul *event Transfer(address indexed from, address indexed to, uint value)* – este lansat atunci când valoarea token-urilor sunt mutate de la un cont (*from*) la altul (*to*). Valoarea poate să fie și 0.

Linia 23 – evenimentul *event Approval(address indexed owner, address indexed spender, uint value)* – este lansat atunci când valoarea permisă a spender-ului pentru un proprietar de token-uri este inițializat de un apel pentru aprobare prin *{approve}.* Valoarea (value) este noua valoare permisă.

Aplicație de laborator

Considerăm următorul exemplu de contract pentru token utilizând ERC20 de la adresa https://solidity-by-example.org/app/erc20/.

1. Deschideți în Visual Studio Code folderul *BLab4.*
2. Vizualizați fișierele `IERC20.sol` și `erc20_smartcontract.sol`
3. În terminalul din Visual Studio Code, rulați comanda `remixd -s .`
4. Accesați remix.ethereum.org în browser și compilați fișierul erc20_smartcontract.sol
5. Tot în remix.ethereum.org compilați și fișierul `erc20_token.sol`
6. Tot în remix.ethereum.org compilați și fișierul `erc20_schimbtoken.sol`
7. Evaluați funcțiile și observați comportamentul acestora.