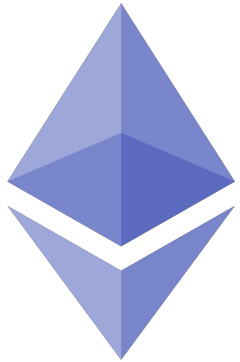
Desarrollo Blockchain Ethereum con Solidity

Módulo 4 - EIPs y estándares

EIPs y estándares

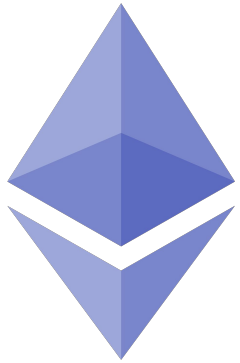
Introducción al standard

- Los **EIPS** (Ethereum Improvement Proposal), son básicamente propuestas de mejora a incluir en futuras versiones*.
- Dentro de los EIPS se encuentran todos los **ERC** *(Ethereum Request for Comments)***.
- El EIP número 20, que se llamó **ERC20,** proponía un estándar de funcionamiento para los Tokens.
- La aceptación supuso una mejora en el uso, ya que desde entonces los tokens son compatibles con este estándar y eso permite que los wallets puedan soportar los nuevos tokens por defecto.



Diferencias entre ERC20 y ERC721

- Es importante entender que si bien existen otros estándares, **ERC20** y **ERC721** son los más utilizados.
- ERC20 representa un token fungible.
- ERC721 representa un token no fungible.
- Cada ERC tiene su especificacion definida.
- Un token ERC20 **NO** puede ser un token ERC721.
- Ambos son tradeables dentro de la Ethereum Blockchain (y fuera*).
- Existen extensiones para ambos estándares.



Diferencias entre ERC20 y ERC721

- Cuando alguien dice que tiene un token "**ERC-20**" solo significa que ese contrato de token responde a una serie común (predefinida) de métodos.
- Significa que el token puede ser **transferido**, **consultado**, **aprobado**, etc.
- **No** significa que el token tenga un valor ($$$)
- **No** significa que el token haga algo*
- Es posible ver un **totalSupply****
- **Está totalmente pensado para ser Dinero.**

Diferencias entre ERC20 y ERC721

La interfaz a implementar para un token de tipo **ERC20**

```
pragma solidity ^0.4.18;
/**
 * @title ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/20
 * @jds 2018
 */
contract ERC20 {
  function totalSupply() public view returns (uint256);

  function balanceOf(address who) public view returns (uint256);
  function transfer(address to, uint256 value) public returns (bool);

  function allowance(address owner, address spender) public view returns (uint256);
  function transferFrom(address from, address to, uint256 value) public returns (bool);
  function approve(address spender, uint256 value) public returns (bool);

  event Transfer(address indexed from, address indexed to, uint256 value);
  event Approval(address indexed owner, address indexed spender, uint256 value);
}
```

Diferencias entre ERC20 y ERC721

- Para entender la utilidad de un token **ERC-721** basta con pensar en objetos coleccionables.
- Figuritas de un álbum, cartas especiales, cualquier objeto distinto.
- La definición del ERC721 lo categoriza como “**non-fungible**”.
- Cada token del estándar es **único** e **irrepetible.**
- Un token de tipo ERC721 **NO puede ser dividido** ya que funciona como una unidad*.
- El caso de éxito más conocido sin dudas fue CryptoKitties.

CryptoKitties

Diferencias entre ERC20 y ERC721

La interfaz a implementar para un token de tipo **ERC721**

```
pragma solidity ^0.4.18;
/**
 * @title ERC721 Non-Fungible Token Standard basic interface
 * @dev see https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md
 * JdS 2018
 */
contract ERC721Basic {
  event Transfer(address indexed _from, address indexed _to, uint256 _tokenId);
  event Approval(address indexed _owner, address indexed _approved, uint256 _tokenId);
  event ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved);

  function balanceOf(address _owner) public view returns (uint256 _balance);
  function ownerOf(uint256 _tokenId) public view returns (address _owner);
  function exists(uint256 _tokenId) public view returns (bool _exists);

  function approve(address _to, uint256 _tokenId) public;
  function getApproved(uint256 _tokenId) public view returns (address _operator);

  function setApprovalForAll(address _operator, bool _approved) public;
  function isApprovedForAll(address _owner, address _operator) public view returns (bool);

  function transferFrom(address _from, address _to, uint256 _tokenId) public;
  function safeTransferFrom(address _from, address _to, uint256 _tokenId) public;
  function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes _data) public;
}
```

¡Muchas gracias!

¡Sigamos trabajando!