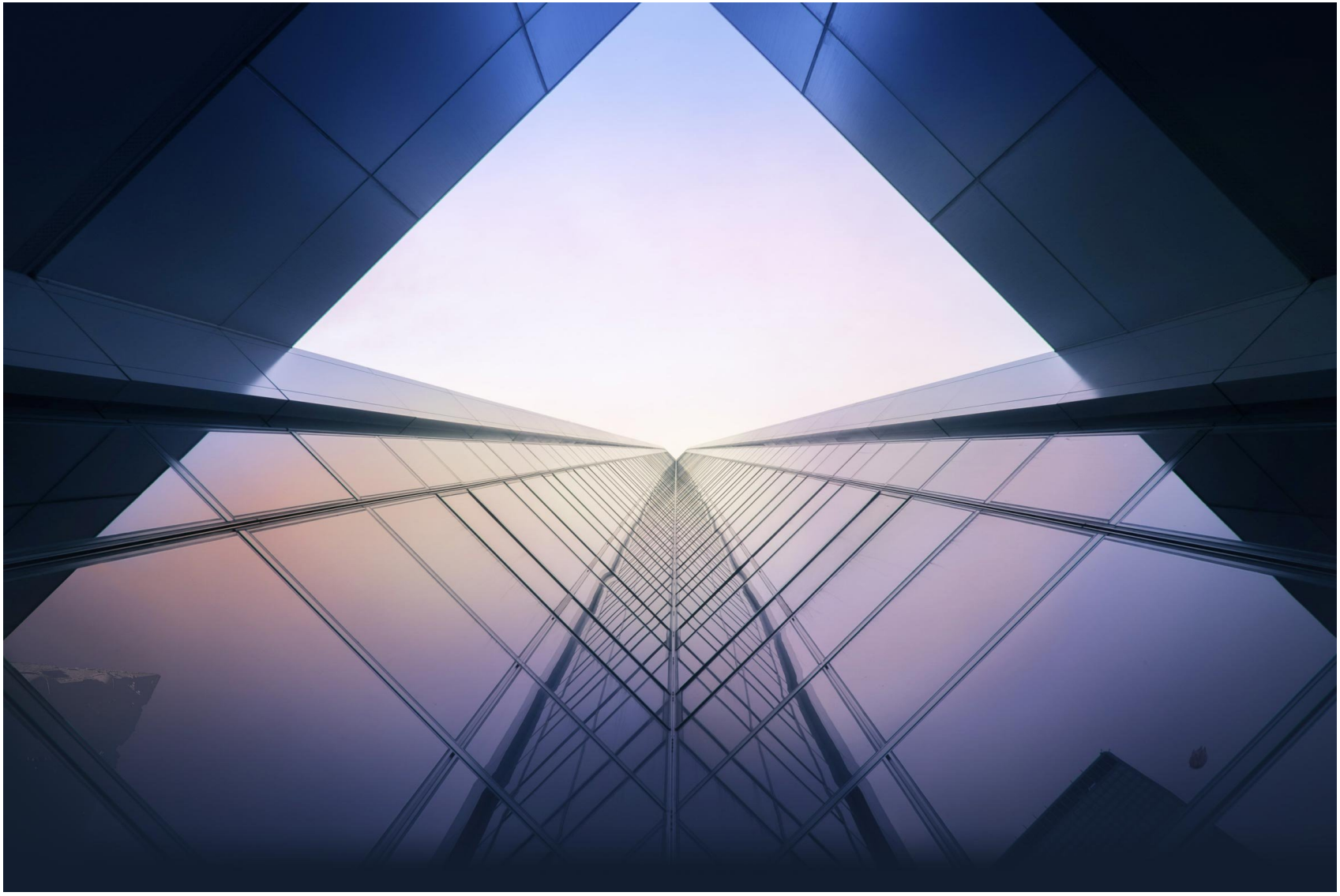
Contratos Inteligentes y *Tokens* en Ethereum

Miguel Angel Astor Romero

Universidad Católica Andrés Bello
10 de abril de 2018

Agenda

1. Introducción
2. Contratos Inteligentes
3. Tokens en Ethereum
4. Conclusiones



Ethereum

- Ethereum es una plataforma de cómputo distribuido basada en Blockchain.
- Centrada en una criptomoneda llamada ether (ETH).

Origenes de Ethereum

Vitalik Buterin

- Propuesto en 2013 por V. Buterin.
- Especificación inicial en el Ethereum white-paper.
- Especificación de la EVM en el Ethereum yellow-paper.

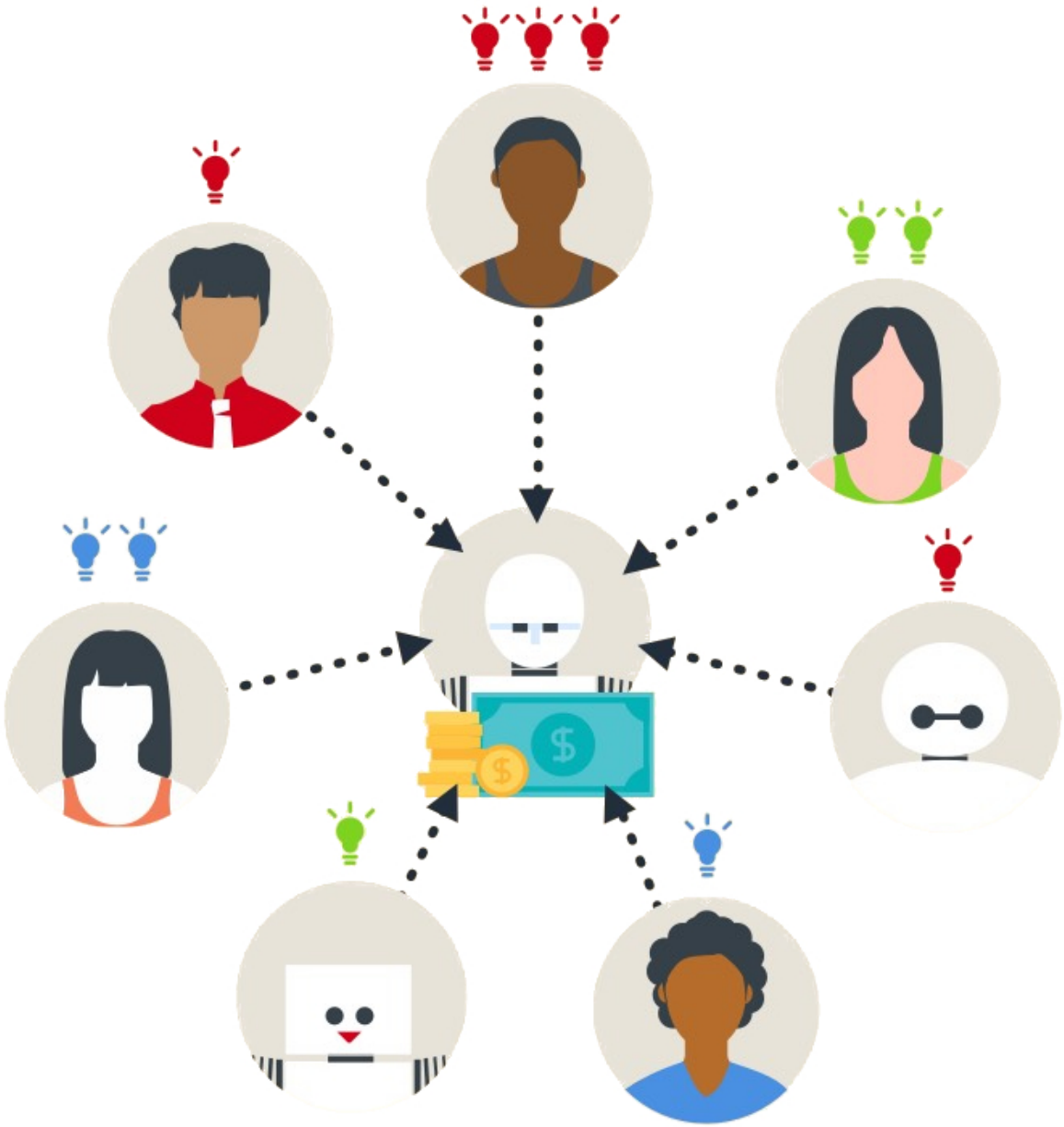
Roadmap de Ethereum

- El desarrollo de Ethereum sigue las siguientes etapas:

Olympic	Primera versión pública (beta). Mayo de 2015.
Frontier	Segunda versión pública, experimental. Julio de 2015.
Homestead	Primera versión estable. Marzo de 2016.
Metropolis	Segunda versión estable. Publicada en dos etapas.
Byzantium	Metropolis, parte 1. Octubre de 2017.
Constantinople	Metropolis, parte 2. Versión futura.
Serenity	Versión futura.

Características Técnicas

- Algoritmo de consenso `Ethash` tipo *Proof-of-Work* (PoW).
  - Resistente a minado ASIC.
  - Basado en el algoritmo de *hashing* Keccak (SHA-3).
  - Se espera cambiar por *Proof-of-Stake* (PoS) en la versión *Serenity*.
- No se paga comisión por las transacciones.
- Las transacciones se cobran según la cantidad y tipo de operaciones que ejecutan (gas).
- Los bloques se generan aprox. cada 14 segundos.
- No hay límite a la cantidad de ETH, de momento.



Antecedentes: Lógica de Negocios en Sistemas de E-Commerce

El término “contrato inteligente” fue inventado en 1996 por Nick Szabo.

Nick Szabo

Antecedentes: Bitcoin

Bitcoin permite programar transacciones con `Script`.

Propiedades de Script

- Similar a `Forth`.
- No es Turing-completo.
- Limitado por razones de seguridad.
- Ejecutado en una máquina virtual.

Contratos Inteligentes en Ethereum

Definición

Son programas que permiten embeber lógica de negocios arbitraria en las transacciones de un sistema basado en Blockchain.

Propiedades

- Modelo de cómputo turing-completo basado en una máquina de pila.
- Ejecución descentralizada.
- Estado persistente.

Ethereum Como Plataforma de Cómputo Descentralizado

A diferencia de Bitcoin, el objetivo de Ethereum es ser una plataforma de cómputo distribuido primero y un sistema financiero después.

- Los contratos inteligentes son la base de las aplicaciones descentralizadas (dApp).
- Las dApp son el modelo de cómputo de Ethereum:
  - Modelo de red Peer-2-Peer (P2P).
  - Estado persistente verificado mediante la Blockchain.
  - Cambio de estado basado en transacciones.
- Las DAPP se ejecutan de forma asíncrona y concurrente en múltiples nodos de la red.

Operaciones sobre contratos inteligentes

Ethereum permite a los clientes cuatro operaciones sobre los contratos desplegados en la red.

Despliegue

Enviar un contrato a la red.

Ejecución

Ejecutar una función del contrato.

Consulta

Examinar un contrato con su API público constante.

Destrucción

Desactivar el contrato.

IMPORTANTE

La operación de destrucción tiene serias implicaciones de uso y seguridad.

Programación de Contratos Inteligentes

Hay dos formas de programar contratos inteligentes.

Frameworks

Truffle simplifica considerablemente el proceso.

Manualmente

- Escribir código.
- Compilar.
- Desplegar a prueba.
- Depurar.
- Desplegar a Ethereum.

Lenguajes

- Serpent (similar a Python; obsoleto).
- Solidity (similar a Javascript).
- Low-Level Lisp (similar a Scheme y Common Lisp).
- Viper (similar a Python; experimental).
- Mutan (similar a Go; obsoleto).

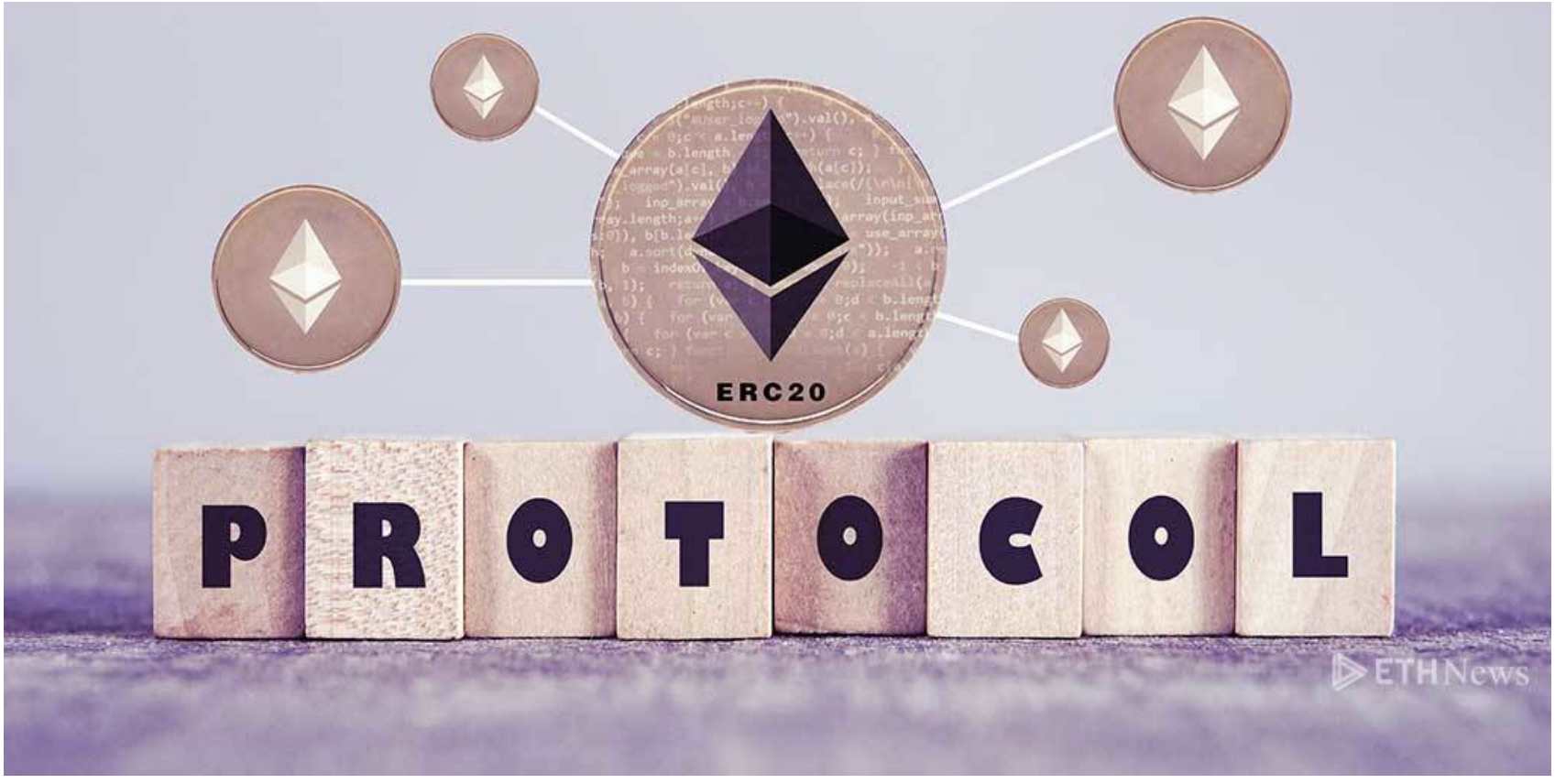
Hello, Solidity!

```
contract Greeter {
    address owner;
    string greeting;

    function Greeter(string _greeting) public {
        owner = msg.sender;
        greeting = _greeting;
    }

    function kill() {
        if (msg.sender == owner) selfdestruct(owner);
    }

    function greet() constant returns (string) {
        return greeting;
    }
}
```

El Framework Truffle

http://truffleframework.com/

- Integración con Node.js y npm.
- Gestión automática de artefactos.
- *Framework* de pruebas unitarias.
- Despliegue a redes privadas, de prueba o la red pública Ethereum.
- Consola de depuración basada en Javascript.


ERC20
PROTOCOL
ETHNews

Fundamentos de los Tokens de Ethereum

Un *token* en Ethereum es un contrato inteligente que representa un bien digital.

Propiedades

- Siguen una interfaz estandar (pe. ERC-20).
- Acuñables o completamente creados desde el inicio.
- Pueden transferirse entre cuentas Ethereum.
- Pueden ser fungibles[a] o no.

[a] Los *tokens* son indistinguible entre si.

Que no es un token

- Una nueva cryptomoneda.

Ethereum Improvement Proposals (EIP)

Las interfaces de los *tokens* están estandarizadas en EIP's.

- El desarrollo de Ethereum es guiado por la comunidad, en un modelo similar al de Bitcoin.
- Las EIP son sometidas a un proceso de revisión por pares en un model similar al de la IETF.
- Hay cuatro clases de EIP (capas):
  - *Consensus*
  - *Networking*
  - *API/RPC*
  - *Application*

Tokens ERC-20

Estandar de funcionalidades básicas para *tokens*.

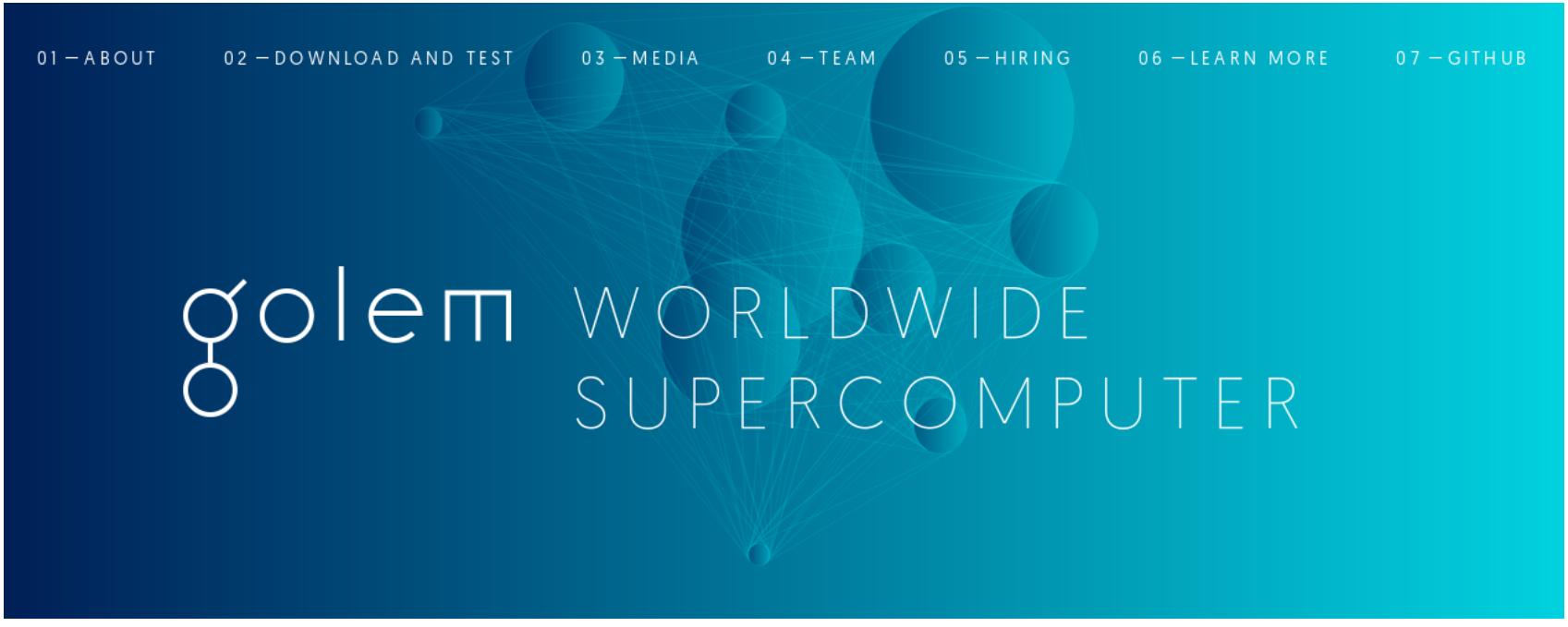
Interfaz

- name()
- symbol()
- decimals()
- totalSupply()
- balanceOf()
- transfer()
- transferFrom()
- approve()
- allowance()

Propiedades

- Fungibles.
- Divididles.
- Delegables.
- Públicos.

Ejemplo: https://golem.network/



Tokens no fungibles ERC-721

Los *tokens* ERC-20 son fungibles y divisibles. Esto trae limitaciones para representar ciertos tipos de bienes digitales.

Tokens no fungibles

Sirven para representar "títulos de propiedad" digitales y bienes similares.

Interfaz

Deben implementar los estándares ERC-720 y ERC-165

Propiedades

- No fungibles.
- Indivisibles.
- Delegables.
- Públicos.

Ejemplo: https://www.cryptokitties.co/

What is CryptoKitties?

CryptoKitties is a game centered around breedable, collectible, and oh-so-adorable creatures we call CryptoKitties! Each cat is one-of-a-kind and 100% owned by you; it cannot be replicated, taken away, or destroyed.

Programación de Tokens

Existen bibliotecas y frameworks para programación de contratos inteligentes que incluyen implementación de *tokens*.

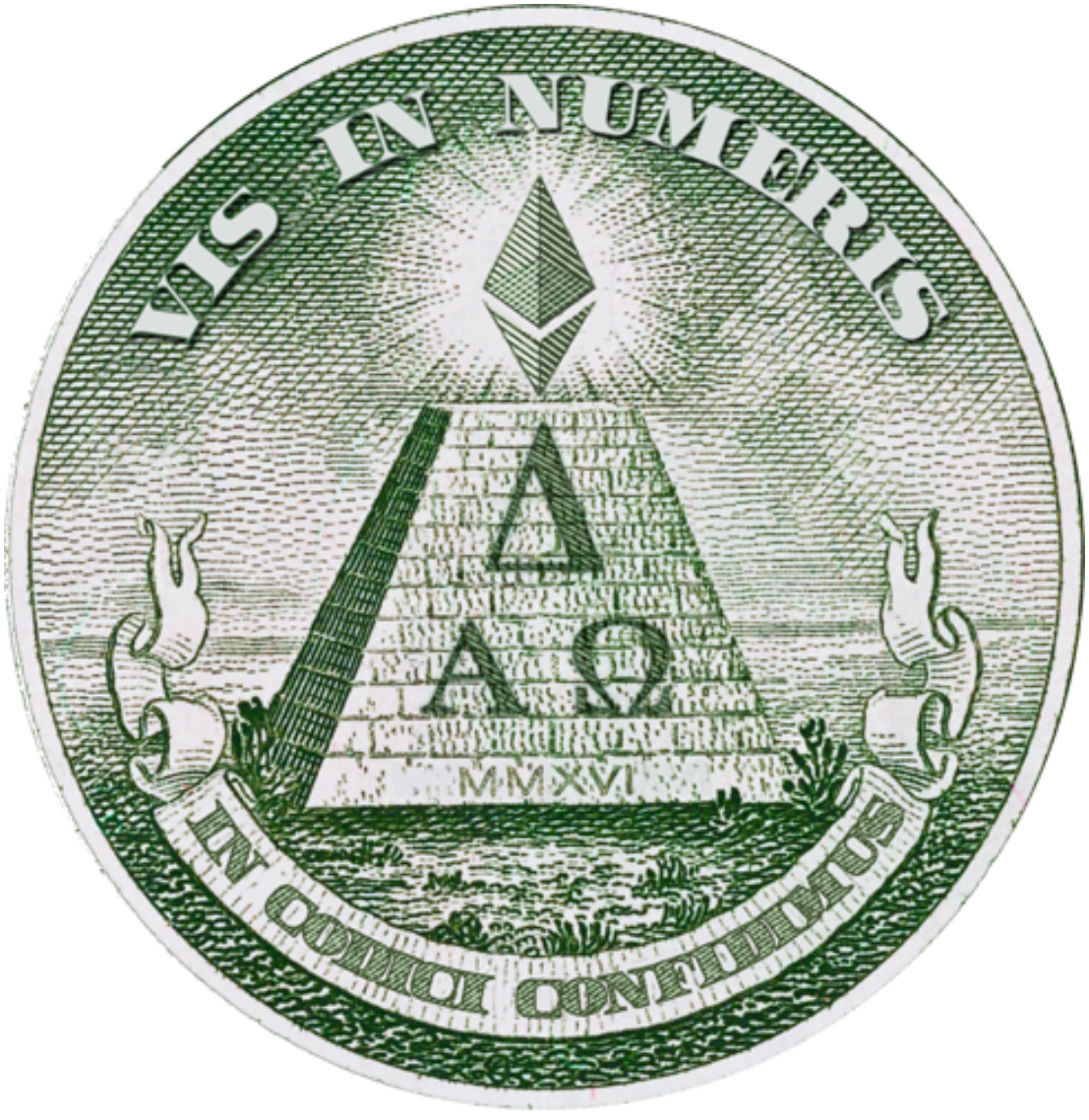
https://openzeppelin.org/

bash

```
[~]$ npm install zeppelin-solidity
Installing [============] 100%
> zeppelin-solidity@1.2.0 install
> scripts/install.sh
[~]$ vim ExampleToken.sol
[~]$ truffle console
> var myToken = ExampleToken.deployed();
> myToken.totalSupply()
100000000000000000000
> myToken.transfer(...)
true
```

vim ExampleToken.sol

```
pragma solidity ^0.4.11;
import "zeppelin-solidity/contracts/token/StandardToken.sol";

contract ExampleToken is StandardToken {
 string public name = "ExampleToken";
 string public symbol = "EGT";
 uint public decimals = 18;
 uint public INITIAL_SUPPLY = 10000 * (10 ** decimals);

 function ExampleToken() {
  totalSupply = INITIAL_SUPPLY;
  balances[msg.sender] = INITIAL_SUPPLY;
 }
}
```


VIS IN NUMERIS
MMXVI
IN COMMON CONFIDENCE

Conclusiones

1. Ethereum es un sistema de computo distribuido basado en dApp's y asociado a una criptomoneda.
2. Los contratos inteligentes permiten representar e intercambiar bienes digitales con lógica de negocios arbitraria.
3. Otras criptomonedas están adoptando el modelo de contratos inteligentes de Ethereum (pe. `https://www.rsk.co/` para Bitcoin).

Referencias

1. N. Szabo, Smart Contracts: Building Blocks for Digital Markets.
2. `https://github.com/ethereum/wiki/wiki`
3. `http://eips.ethereum.org`
4. V. Buterin, A Next-Generation Smart Contract and Decentralized Application Platform. Ethereum *white-paper*, 2013.
5. G. Wood, ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER, Ethereum *yellow-paper*, 2014.

Contactos

Prof. Miguel A. Astor

- miguel.astor@ciens.ucv.ve
- miguel.a.astor@ucv.ve

¿Donde descargar estas láminas?

- `https://github.com/miky-kr5/Presentations`

¿Preguntas?