

A dark blue background featuring a complex, glowing network of interconnected dots and lines, resembling a molecular or neural network.

Abril/2019

Blockchain 101

Introducción a Solidity con ejemplos

VR3

Oscar Chavez
CTO @ VR3
oscar@vr3.io



Oscar Chávez

oscar@vr3.io | [@oschvr](https://twitter.com/oschvr)

Systems & Engineering

Front/Backend Services

Systems & Cloud Admin

Networks & Security

Distributed Systems

Blockchain & Smart Contracts



Blockchain es un sistema de transacciones digitales que consiste en una base de datos descentralizada.

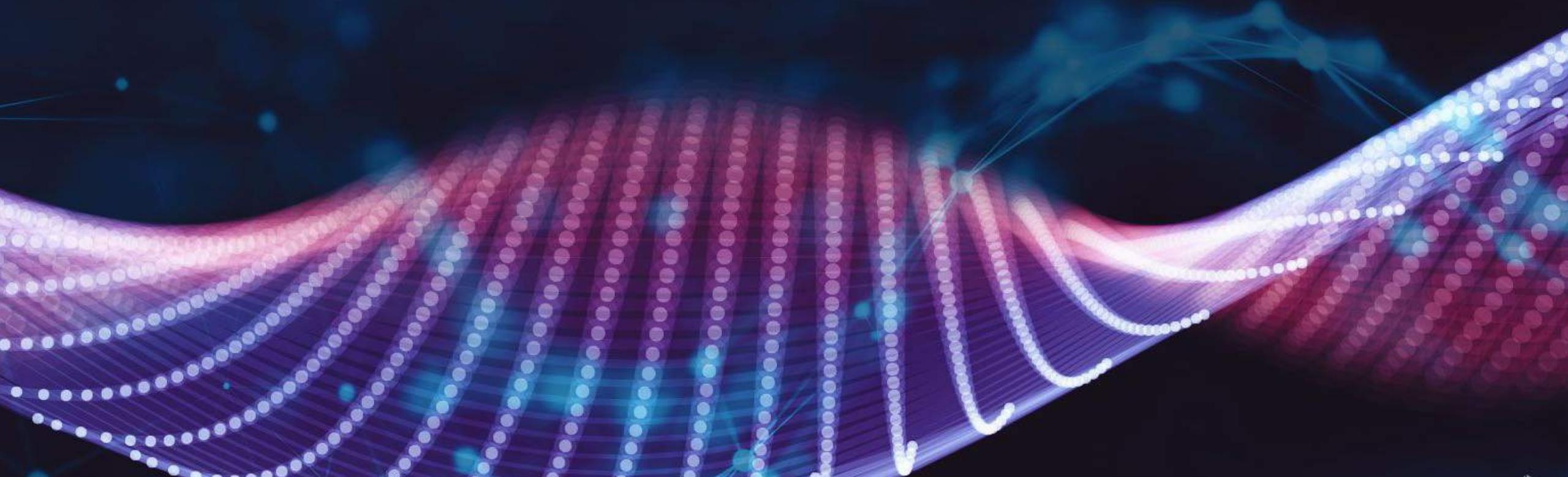


1. ¿Qué es Ethereum?
2. Smart Contracts (Solidity)
3. ¿Por qué usar Blockchain?
4. Conclusión

- 1. ¿Qué es Ethereum?
- 2. Smart Contracts (Solidity)
- 3. ¿Por qué usar Blockchain?
- 4. Conclusión



ETHEREUM

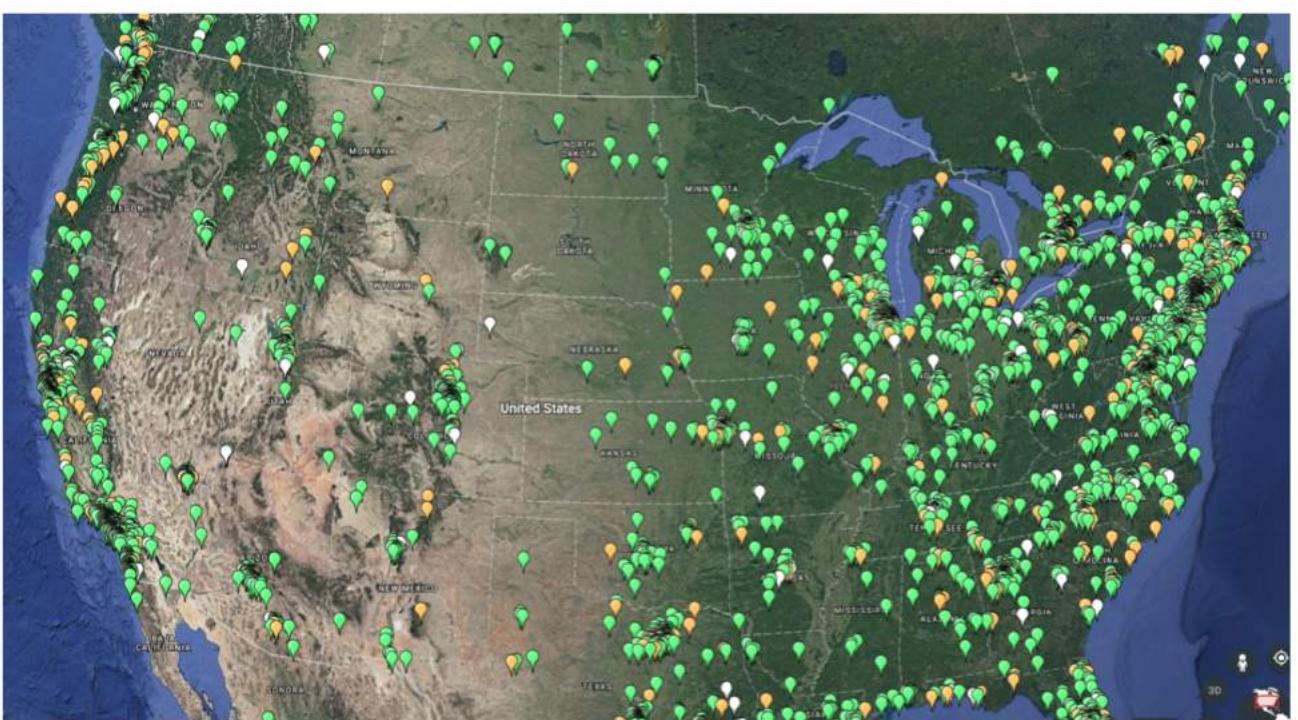
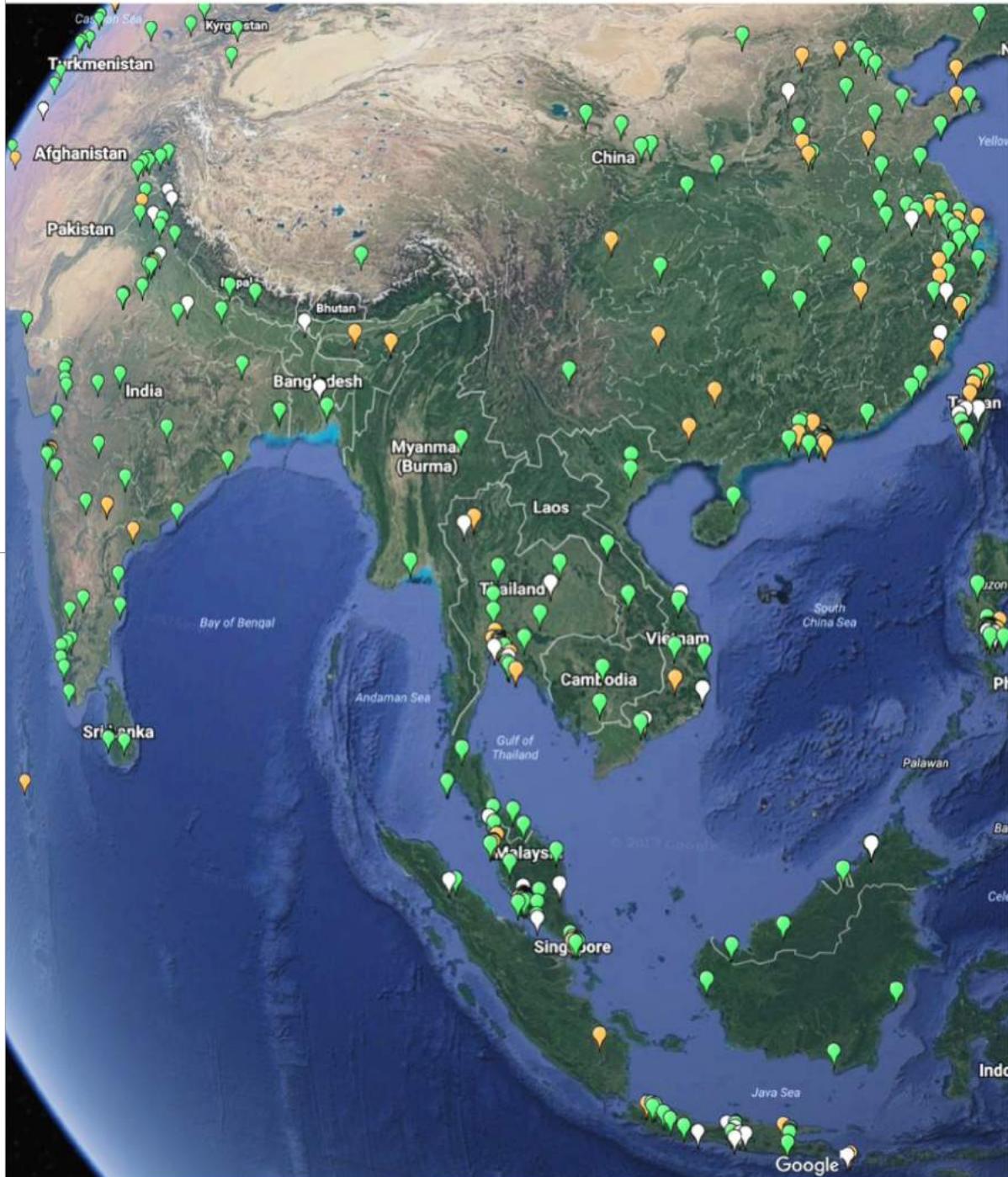
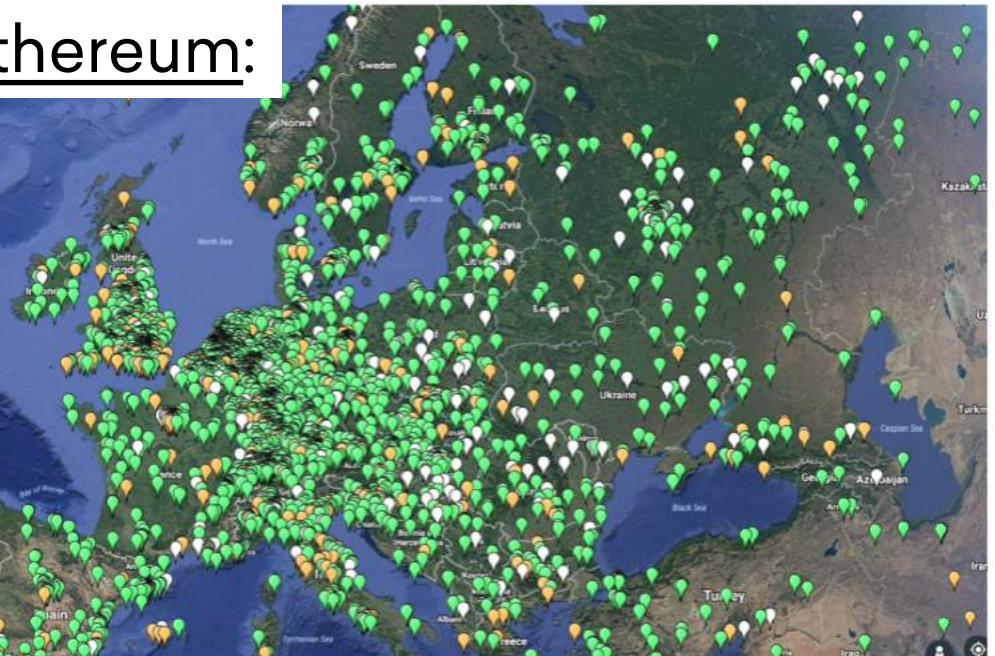


- Es una estructura de datos decentralizada (Cadena de bloques)
- Usa el **Ether (ETH)** (Criptomoneda)
 - > Divisa nativa
 - > Base de sistema de valor y transferencia.
- *Trustless*: No hay necesidad de un tercero.
- Bitcoin 2.0 (?)

Ethereum es:

- Una plataforma **descentralizada** que ejecuta **contratos inteligentes**.
- Una maquina que **depende de transacciones** para moverse entre estados.
- Una blockchain **basada en cuentas**.

Red de Ethereum:



Descentralizada: No hay punto de control/falla.
Resistente a censura. Inmutable

Contratos Inteligentes: Programación sofisticada
que permite cómputo arbitrario.

Basada en Transacciones: El estado de toda la red cambia cada nueva **Transacción**.

Basada en Cuentas: El estado esta hecho por las **Cuentas** de la red. Cada **Cuenta** tiene:

- Una **Dirección** (`0x916ce67283eb295D2f70F1D944436d0A222083B1`)
- Un balance en **Ether** (`12,345.6789 ETH`)
- (Opcionalmente) Código de un **Contrato y Almacenamiento**

Transacciones:

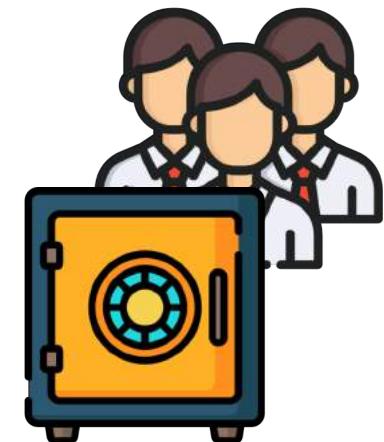
- Transfieren valor e información entre **Cuentas**.

Cuentas:

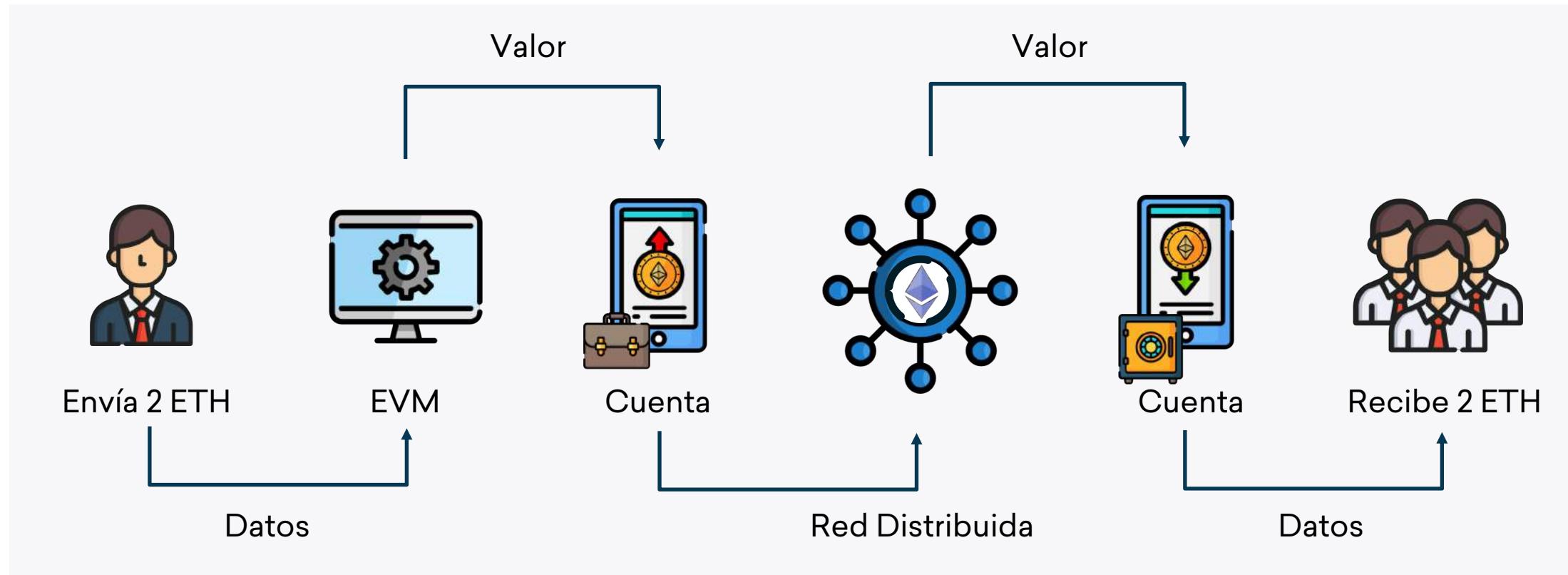
- Cuentas Externas
- Cuentas de Contratos

Cuentas Externas:

- Propiedad de alguna entidad externa
- Identificadas por una **Dirección**
- Mantienen un balance en **Ether (ETH)**
- Pueden enviar **Transacciones**:
 - > Transferir **ETH** a otras **Cuentas Externas**
 - > Llamar a **Contratos**

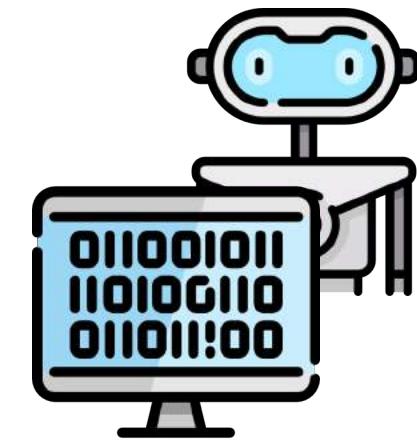


Proceso de Transferencia de Cuenta a Cuenta:



Cuentas de Contratos:

- Identificadas por una **Dirección**
- Mantienen un balance en **Ether (ETH)**
- Tiene asociado código de un **Contrato**
- La ejecución del código del **Contrato** es detonado por **Transacciones** de:
 - > **Cuentas Externas**
 - > Mensajes de otros **Contratos**



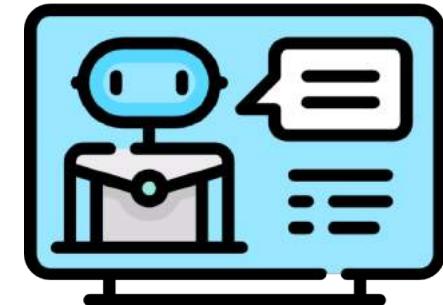
1. ¿Qué es Ethereum?
2. Smart Contracts (Solidity)
3. ¿Por qué usar Blockchain?
4. Conclusión

Un **Contrato Inteligente** es una pieza de software que guarda reglas y términos de un contrato específico.

Lleva a cabo el acuerdo **entre partes** y luego **ejecuta** los términos una vez que los **objetivos** se cumplen



- En el caso de Ethereum, un **Contrato Inteligente** es una **Cuenta** que tiene código(instrucciones) asociado.
- “Agentes autónomos” viviendo en la Red de Ethereum.



- Diseñados lógicamente (*if-this-then-that*)
- Código programable (**Solidity**)
- Transparencia y apertura.
- **Transacción == Ejecución automática.**
- **Irrevocables e irreversible.**



Solidity

- **Solidity** es un lenguaje de alto nivel, orientado a objetos, diseñado para implementar contratos inteligentes
- Influencia en C++, Python, JavaScript y diseñado para trabajar sobre la **Ethereum Virtual Machine (EVM)**
- Lenguaje tipado, soporta herencia, librerías y otros tipos definidos por el usuario

```
pragma solidity ^0.4.24;

contract NombreContrato {

    ...
    <declaración de variables>
    ...
    <mapeos>
    ...
    <constructor>
    ...
    <funciones>
    ...
    <modificadores>
}
```

HolaMundo.sol

```
pragma solidity ^0.4.24;

contract HelloWorld {

    string saySomething;

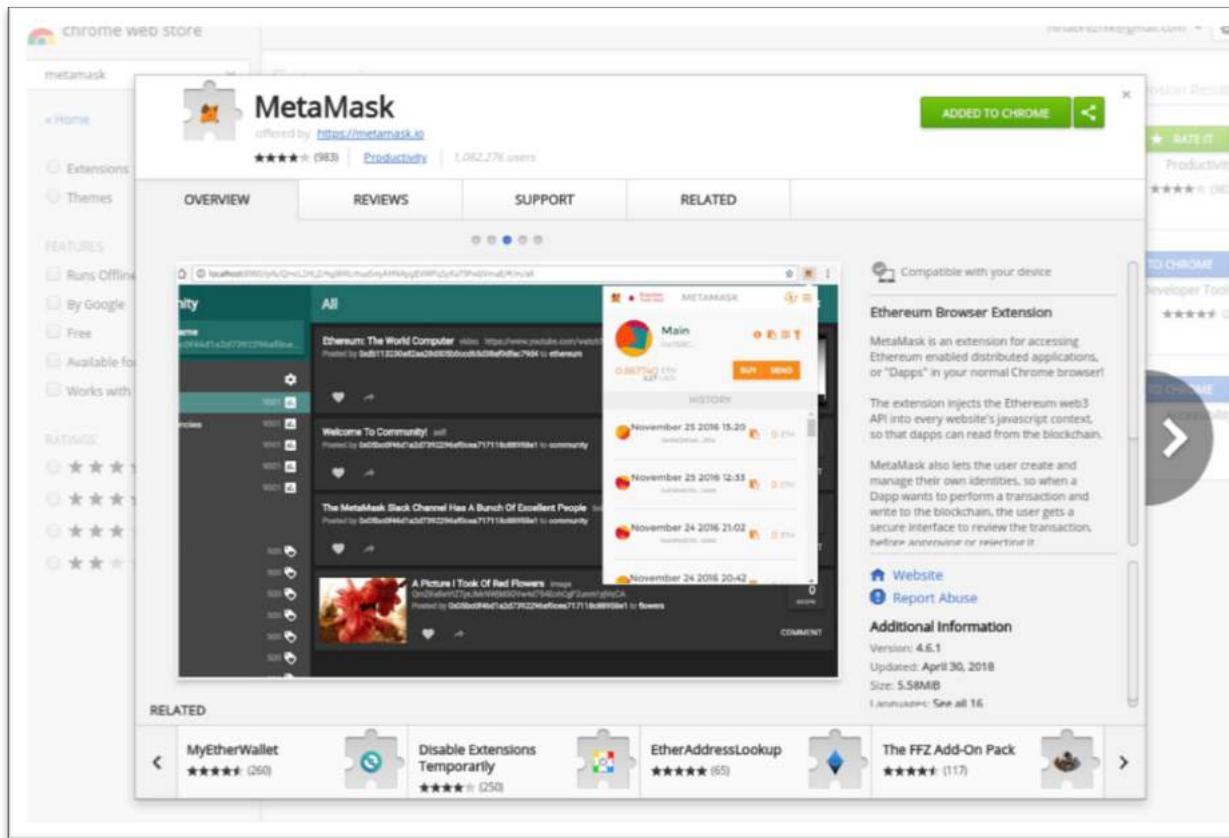
    constructor() public {
        saySomething = "Hola FLISOL!";
    }

    function speak() constant public returns(string isSaying) {
        return saySomething;
    }

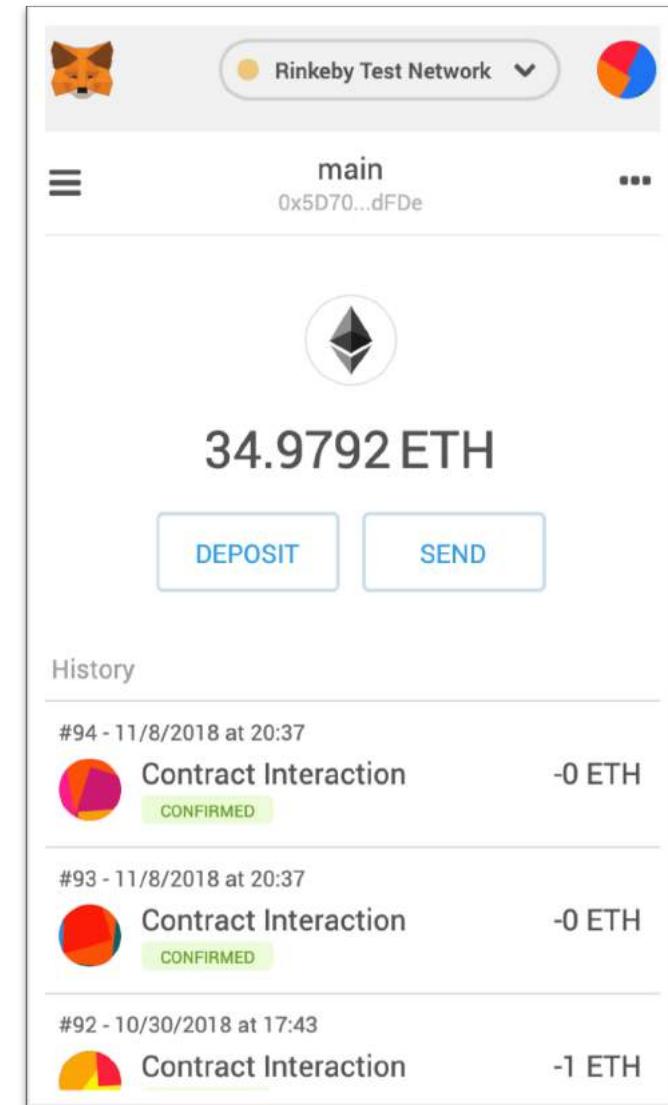
    function sayOtherThing(string somethingElse) public returns
        (bool success) {
        saySomething = somethingElse;
        return true;
    }
}
```

Instalar Metamask

chrome.google.com/webstore



- Metamask es el puente entre el Blockchain de Ethereum y el navegador.
- Extension para Chrome y Firefox que funciona como cartera para ETH
- Funciona para desarrollo y acceso a redes de prueba (Ropsten, Rinkeby)
- faucet.rinkeby.io



Remix: Solidity IDE

remix.ethereum.org

The screenshot shows the Remix Solidity IDE interface. On the left, there's a code editor with a file named "browser/HolaMundo.sol". The code defines a contract named "HelloWorld" with two functions: "saySomething" and "sayOtherThing". The "saySomething" function returns a string. The "sayOtherThing" function takes a string parameter and returns a boolean value. On the right side, there are tabs for "Environment", "Run", "Analysis", "Testing", "Debugger", "Settings", and "Help". Under "Environment", it shows "Injected Web3" and "Rinkeby (4)" accounts, gas limit set to 3000000, and a value of 0 wei. Below that is a section for "HelloWorld" with a "Deploy" button and an "At Address" input field. A "Transactions recorded:" section indicates no contracts have been deployed yet. At the bottom, there's a terminal window with the text: "Welcome to Remix v0.7.6 - You can use this terminal for: • Checking transactions details and start debugging. • Running JavaScript scripts. The following libraries are accessible: • web3 version 1.0.0 • ethers.js • evmnow • compilers - contains currently loaded compiler • Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run from a JavaScript script. • Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions. creation of HelloWorld pending...".



The screenshot shows the Truffle UI interface with the following configuration:

- Environment:** Injected Web3, Rinkeby (4)
- Account:** 0x5d7...7dfde (34.974002714 ether)
- Gas limit:** 3000000
- Value:** 0 wei

The contract being interacted with is **HelloWorld**. There is a **Deploy** button and an **At Address** input field containing `0xcef2d246ac4fac144e6803ce180f999b80cc54`.

The **Transactions recorded:** section shows 0 transactions.

The **Deployed Contracts** section lists the deployed **HelloWorld** contract at address `0xcef2d246ac4fac144e6803ce180f999b80cc54`. It includes a function call `sayOtherThing(string somethingElse)` with value `string somethingElse` and a function call `speak()` which returns `O: string: isSaying Hola Toluca`.

HelloWorld.sol

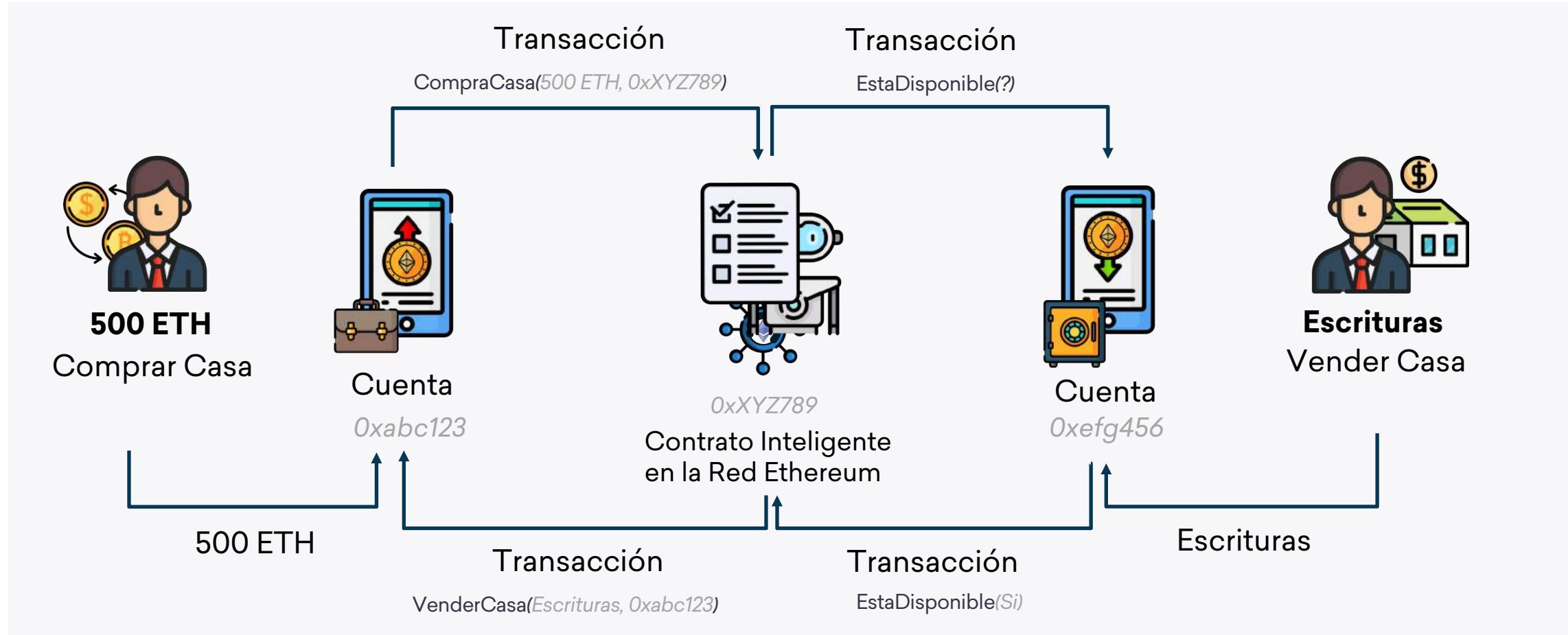
<https://rinkeby.etherscan.io/address/0xf961740ad17bf1a9a6eef7ac6d6ceafa19c033de>

0xf961740ad17bf1a9a6eef7ac6d6ceafa19c033de



Proceso de Ejecución de Contrato

Ejemplo: Comprar una casa en Ethereum



Safe Remote Purchase

<https://solidity.readthedocs.io/en/v0.4.0/solidity-by-example.html - safe-remote-purchase>



1. Vendedor deposita 2, precio de producto es 1.

Balance de contrato es ahora 2 💰 💰

2. Comprador hace oferta, paga 2: 1 es el precio del producto, 1 es un depósito.

Balance del contrato ahora es 4 💰 💰 💰 💰

3. Vendedor envía producto 🎁

Balance del contrato sigue en 4 💰 💰 💰 💰

4. Comprador confirma recibo, recibe su depósito de 1 de regreso.

Balance ahora es 3 💰 💰 💰

5. Contrato envía 2 del depósito del vendedor, 1 del precio del producto ya enviado 💸.

Balance ahora es 0

Purchase.sol

Variables

Valor

Vendedor

Comprador

Estado

Constructor

*El vendedor
invoca el
contrato*

*El valor a
bloquear debe
ser par (la mitad
de **costo * 2**)*

```
pragma solidity ^0.4.22;

contract Purchase {
    uint public value;
    address public seller;
    address public buyer;
    enum State { Created, Locked, Inactive }
    State public state;

    // Ensure that `msg.value` is an even number.
    // Division will truncate if it is an odd
    // number.
    // Check via multiplication that it wasn't an
    // odd number.
    constructor() public payable {
        seller = msg.sender;
        value = msg.value / 2;
        require((2 * value) == msg.value, "Value
            has to be even.");
    }
}
```

Purchase.sol

Modificadores

Middlewares
de validación.

```
modifier condition(bool _condition) {
    require(_condition);
    _;
}

modifier onlyBuyer() {
    require(
        msg.sender = buyer,
        "Only buyer can call this."
    );
    _;
}

modifier onlySeller() {
    require(
        msg.sender = seller,
        "Only seller can call this."
    );
    _;
}

modifier inState(State _state) {
    require(
        state = _state,
        "Invalid state."
    );
    _;
}
```

Purchase.sol

Funciones

Abortar:

Solo la puede llamar el vendedor, y solo si el estado de la venta está en “Creado”

El balance regresa al vendedor.

```
.... /// Abort the purchase and reclaim the ether.  
.... /// Can only be called by the seller before  
.... /// the contract is locked.  
function abort()  
.... public  
.... onlySeller  
.... inState(State.Created)  
{  
.... emit Aborted();  
.... state = State.Inactive;  
.... seller.transfer(address(this).balance);  
.... }
```

Purchase.sol

Funciones

Confirmar Compra:

*El comprador envía el **costo * 2** en Ether que será bloqueado para confirmar y bloquear la venta*

```
.... /// Confirm the purchase as buyer.  
.... /// Transaction has to include `2 * value`  
.... ether.  
.... /// The ether will be locked until  
.... confirmReceived  
.... /// is called.  
function confirmPurchase()  
.... public  
.... inState(State.Created)  
.... condition(msg.value == (2 * value))  
.... payable  
{  
.... emit PurchaseConfirmed();  
.... buyer = msg.sender;  
.... state = State.Locked;  
}
```

Purchase.sol

Funciones

Confirmar Recepcion:

*El comprador
confirma
recepción,*

*El excedente
se regresa al
comprador*

*El monto se
transfiere al
vendedor.*

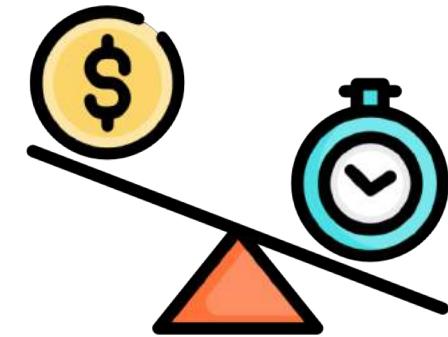
```
.... // Confirm that you (the buyer) received the
.... item.
.... // This will release the locked ether.
function confirmReceived()
.... public
.... onlyBuyer
.... inState(State.Locked)
{
.... emit ItemReceived();
.... // It is important to change the state
.... first because
.... // otherwise, the contracts called using
.... `send` below
.... // can call in again here.
.... state = State.Inactive;

.... // NOTE: This actually allows both the
.... buyer and the seller to
.... // block the refund - the withdraw pattern
.... should be used.

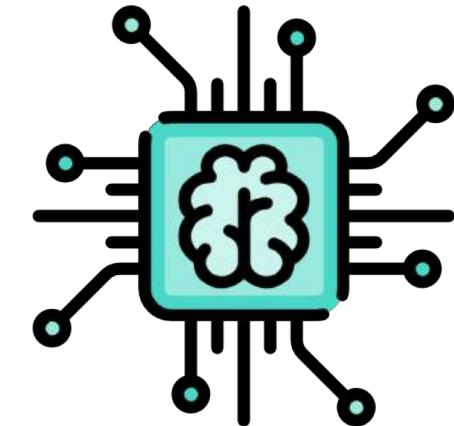
.... buyer.transfer(value);
.... seller.transfer(address(this).balance);
}
```

1. ¿Qué es Ethereum?
2. Smart Contracts (Solidity)
3. ¿Por qué usar Blockchain?
4. Conclusión

- Transacciones frecuentes entre partes
- Tareas manuales o repetitivas.
- Una **Blockchain** actúa como una base de datos compartida y la única fuente de verdad.



- Los **Contratos Inteligentes** automatizan aprobaciones, cálculos y otras actividades que tienden al desvío y el error.



1. ¿Qué es Ethereum?
2. Smart Contracts
3. ¿Por qué usar Blockchain?
4. Conclusión

La tecnología del **Blockchain** de **Ethereum** ofrece la habilidad de crear aplicaciones descentralizadas que generen nuevos negocios y operaciones sin intermediarios y que son seguras y flexibles.

github.com/oschvr/blockchain101



A dark blue background featuring a complex, glowing network graph composed of numerous small, light blue dots connected by thin, translucent lines, creating a sense of depth and connectivity.

Abril/2019

VR3

Oscar Chavez
CTO @ VR3
oscar@vr3.io
[@oschvr](https://twitter.com/oschvr)