Desarrollo Blockchain Ethereum con Solidity

Módulo 1 - Introducción a Solidity

Contratos

Contrato

Un *contrato* no es más que un acuerdo entre dos o más partes, un entorno donde se define lo que se puede hacer, cómo se puede hacer, qué pasa si algo no se hace.

Son reglas de juego que permite, a todas las partes que lo aceptan, entender en qué va a consistir la interacción que van a realizar.

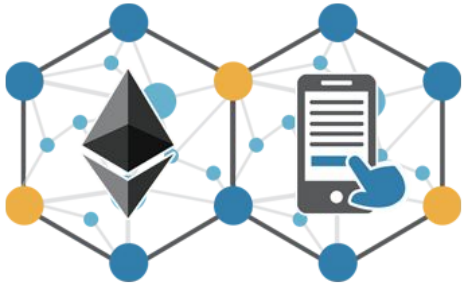
Se encuentran distribuidos en la Ethereum Blockchain a bajo nivel, se trata de una serie de algoritmos, hashes, lógica y varios miles de líneas de código.

Son capaces de ejecutarse por sí mismos de manera autónoma **sin intermediarios.**

Contrato

Un Smart Contract podría tener infinitas razones de uso, como ser:

- **Mercados distribuidos** que permitieran implementar contratos P2P y trading en los mercados con CriptoMonedas (evitando el FIAT).
- **Propiedades** como automóviles, teléfonos, casas o elementos no físicos controlados a través de la blockchain (Smart Property)*
- **Automatización de herencias** estableciendo la asignación de los activos tras el fallecimiento. En cuanto llegase el fallecimiento, el contrato entraría en vigor y se ejecutaría repartiendo en este caso los fondos a la dirección establecida en el contrato.
- **Seguros** partes de accidente, pagos de la compañía para reparaciones, reducción del fraude en accidentes.



Contrato

- ¿Piensan que todos los contratos de la vida real aplican a smart contracts?
- ¿Qué otros escenarios conocen?
- ¿Se animan a pensar algún caso de uso innovador donde se podría aplicar un contrato?

Introducción a Solidity

Introducción a Solidity

- Es popularmente conocido como el lenguaje de programación con el que se programan los contratos inteligentes en Ethereum.
- Es un lenguaje tipado que hace el proceso de verificar y cumplir las restricciones en tiempo de compilación en lugar de en tiempo de ejecución.
- Es un lenguaje Turing Complete.
- Es un lenguaje orientado a contratos, similar a lo que es la orientación a objetos.
- Está diseñado para correr específicamente sobre la EVM.
- Cuenta con un IDE online llamada *Remix.*

Introducción a Solidity

Este es un ejemplo de un contrato sumamente sencillo hecho en solidity versión 0.7.0

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

/**
 * @title Storage
 * @dev Store & retrieve value in a variable
 */
contract Storage {

    uint256 number;

    /**
     * @dev Store value in variable
     * @param num value to store
     */
    function store(uint256 num) public {
        number = num;
    }

    /**
     * @dev Return value
     * @return value of 'number'
     */
    function retrieve() public view returns (uint256){
        return number;
    }
}
```

Introducción a Solidity

Del contrato anterior, cada línea tiene su explicación.

- **Pragma** solidity permite indicar la versión de Solidity a utilizar.
- **Contract** es una palabra reservada que indica el inicio de la definición de un contrato.
- **Public** es un modificador de acceso que le da visibilidad fuera del contrato.
- **Function** es una palabra reservada para indicar el inicio de definición de una función.
- **Returns** indica que la función retornará algún valor.

La asignación en Solidity es de derecha a izquierda.

Versiones y retrocompatibilidad

- La versión actual (2021) de Solidity es la **v0.8.4 ***
- Es posible interactuar con contratos de versiones anteriores.
- No es recomendable combinar más de dos cambios en el número principal de versión**.
- Cuando ocurre un cambio mayor de versión, es posible que muchos contratos dejen de compilar debido a cambios.
- Todos estos cambios siempre son notificados públicamente y a todo se le puede hacer seguimiento en GitHub.

¡Muchas gracias!

¡Sigamos trabajando!