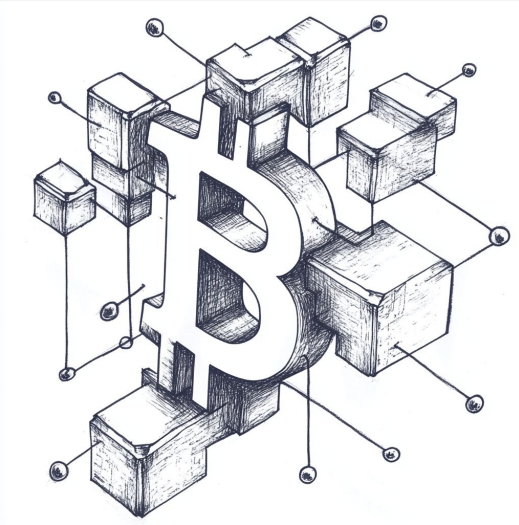
Solidity



Alejandro Narancio
ale.narancio@gmail.com
@anarancio

Tipos de datos

- bool
- uint, uin8, uint16, … uint256
- Address
- mapping(key => value)
- arrays
- structs
- enums
- string

Visibilidad

- public
- private
- internal
- external

Definición de un contrato

```
pragma solidity >=0.6.12 <0.9.0

contract <NOMBRE_CONTRATO> <HERENCIA> {

        <Estado del contrato>

        <constructor>

        <funciones>

}
```

Definición de una interface

```
pragma solidity >=0.6.12 <0.9.0

interface <NOMBRE_INTERFACE> {

        <funciones>

}
```

Constructor

```
constructor(<PARAMETROS>) {
        <LOGICA>
}
```

Mappings

mapping(key => value) <NAME>

Definición de una función

function NOMBRE_FUNCION(lista de parametros) visibilidad modificadores returns(tipo del valor salida de la función)

Definición de un Struct

```
struct <NAME> {
        <type> <name>;
        <type> <name>;
}
```

Definición de un Enum

enum <NAME> { <VAL1>, <VAL2>, <VAL3> }

Definición de un modifier

```
modifier <NOMBRE>() { }

modifier onlyOwner() {
        require(msg.sender == owner, "Not the contract owner");
        _;  // The underscore represents where the modified function's code is executed
    }
```

Modifiers especificos

PURE
VIEW
PAYABLE

Definición de un evento

event <NOMBRE> (<PARAMETROS>)

Para emitir: emit <NOMBRE>

Ejemplos:

```
event ContractInitialized();
event Deposit(address indexed from, uint amount, uint timestamp);
```

Ejercicio

Control de Acceso:

El contrato debe tener un propietario (owner) que se establece en el momento del despliegue
Solo el propietario puede realizar ciertas acciones críticas
Implementar un sistema de whitelist para controlar quién puede donar

Gestión de Campaña:

Establecer una meta de recaudación y un plazo en el constructor
Permitir que solo las direcciones en la whitelist puedan donar
Finalizar la campaña cuando se alcance la meta o venza el plazo

Manejo de Fondos:

Recibir ETH a través de una función específica y mediante transferencias directas
Registrar las donaciones de cada dirección
Permitir al propietario retirar los fondos solo si se alcanza la meta
Implementar un mecanismo de reembolso para los donantes si la campaña fracasa

Eventos y Estado:

Emitir eventos para todas las acciones importantes
Proporcionar funciones para consultar el estado actual de la campaña
Mantener un registro de todos los donantes

Ejercicio 2

Control de Acceso:
Sistema con rol de ADMIN (puede crear propuestas y gestionar votantes) - Pueden ser muchas addresses
Sistema con rol de VOTER (puede emitir votos) - Pueden ser muchas addresses
Funciones para registrar y eliminar votantes
El unico que puede agregar/quitar ADMIN es el owner del contrato

Gestión de Propuestas:
createProposal(string memory description, uint votingDuration): Crear una nueva propuesta
endProposal(uint proposalId): Finalizar una propuesta manualmente (solo ADMIN)
getProposalStatus(uint proposalId): Obtener el estado actual de una propuesta

Sistema de Votación:
vote(uint proposalId, bool inFavor): Votar a favor o en contra de una propuesta
getVoteCount(uint proposalId): Obtener el recuento de votos de una propuesta
hasVoted(address voter, uint proposalId): Verificar si un votante ya ha votado
getVoterInfo(address voter): Obtener información sobre un votante

Estados y Eventos:
Enum ProposalStatus con estados: Pending, Active, Approved, Rejected, Canceled
Eventos para todas las acciones que modifiquen estado agregando el cambio en los parametros

Restricciones y Seguridad:
Los votantes solo pueden votar una vez por propuesta
Solo se puede votar en propuestas activas