

PRÁCTICA 1 - BC

EJERCICIOS 1-5

María Andrea Ugarte Valencia
[UDC]

ÍNDICE

| | |
|-------------------|---|
| EJERCICIO 1 | 2 |
|-------------------|---|

| | |
|-------------------|---|
| EJERCICIO 2 | 7 |
|-------------------|---|

| | |
|-------------------|---|
| EJERCICIO 3 | 8 |
|-------------------|---|

| | |
|-------------------|----|
| EJERCICIO 4 | 10 |
|-------------------|----|

| | |
|-------------------|----|
| EJERCICIO 5 | 11 |
|-------------------|----|

EJERCICIO 1

En este ejercicio se creará una fábrica donde se fabrican o ensamblan diferentes productos inteligentes. Para ello, deberemos realizar los siguientes subapartados:

E1.1

Para comenzar a crear nuestros productos inteligentes, crearemos un contrato base llamado FabricaContract.

- Asegúrate de que nuestro contrato utilice la versión Solidity > 0.8.0.

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.10;
contract FabricaContract {
```

E1.2

Nuestro ID de producto estará determinado por un número de 16 dígitos. Distintas partes del identificador se corresponderán con diferentes atributos del producto (por ejemplo, los dos primeros dígitos pueden corresponder al tipo de producto).

- Declaremos un uint denominado idDigits igual a 16.

```
uint idDigits = 16;
```

E1.3

Vamos a crear algunos productos. Los productos tienen múltiples atributos, por lo que este es un caso de uso perfecto para un struct.

- Crearemos una struct denominada Producto. Tendrá dos propiedades: nombre (string) e identificación (uint).

```
struct Producto {
    uint identificacion;
    string name;
}
```

E1.4

Queremos almacenar una colección de productos inteligentes en nuestra aplicación. Como queremos mostrar todos nuestros productos inteligentes a otras aplicaciones, queremos que sea público.

- Crearemos un array público de structs de Producto y le asignaremos el nombre productos.

```
Producto[] public productos;
```

E1.5

Programemos una función para crear productos inteligentes.

- Crearemos una función privada denominada crearProducto. Debería tomar dos parámetros: _nombre (string) e _id (uint). Utilizamos el primer argumento por valor. No debe olvidarse la convención de nomenclatura.
- Completamos el cuerpo de la función para que cree un nuevo Producto y lo agregue al array productos. El nombre y la identificación del nuevo Producto deben provenir de los argumentos de la función

```
function _crearProducto(string memory _nombre, uint _id) private {  
    productos.push(Producto(_id, _nombre));  
}
```

E1.6

Queremos crear también una función que genere un número de identificación (ID) aleatorio a partir de un string.

- Crearemos una función privada llamada _generarIdAleatorio. Tomará un parámetro llamado _str (string) y devolverá un uint. Confirmaremos la ubicación de los datos del parámetro _str en memoria. Esta función verá algunas de las variables de nuestro contrato pero no las modificará, así se pondrá como view. Respecto al cuerpo de la función _generarIdAleatorio:
- La primera línea de código debe tomar el hash keccak256 de abi.encodePacked(_str) para generar un hexadecimal pseudoaleatorio, typecast como uint y finalmente almacenar el resultado en un uint llamado rand. Queremos que nuestro ID tenga solo 16 dígitos, por tanto, la variable rand debe devolver el valor anterior módulo (%) idModulus. (uint con valor de 10 elevado a idDigits).

```
uint idModulus = 10^idDigits;
```

```
function _generarIdAleatorio(string memory _str) private view returns (uint) {  
    uint rand = uint(keccak256(abi.encodePacked(_str))) % idModulus;  
    return rand;  
}
```

E1.7

También crearemos una función pública que tome como entrada (el nombre del producto) y use el nombre para crear un producto con un ID aleatorio:

- Crearemos una función pública denominada `crearProductoAleatorio`. Tomará un parámetro llamado `_nombre` (un string con la ubicación de los datos configurada en memoria).
 - Debe ejecutar la función `_generarIdAleatorio` con `_nombre` y almacenarla en un uint llamado `randId`.
 - Debe ejecutar la función `_crearProducto` y pasarle `_nombre` y `_randID`

```
function crearProductoAleatorio(string memory _nombre) public {  
  
    uint randId = _generarIdAleatorio(_nombre);  
    _crearProducto(_nombre,randId);  
    Propiedad(randId);  
  
}
```

E1.8

Queremos que un evento informe a nuestro front-end cada vez que se crea un nuevo producto, para que la aplicación pueda mostrarlo.

- Declararemos un evento llamado `NuevoProducto`. Debe pasar `ArrayProductoid` (un uint), `nombre` (un string) y un `id` (un uint).

```
event NuevoProducto(uint ArrayProductoid, string nombre, uint id);
```

- Modificaremos la función `_crearProducto` para iniciar el evento `NuevoProducto` después de agregar el nuevo `Producto` a nuestro array de productos.

```
function _crearProducto(string memory _nombre, uint _id) private {  
  
    productos.push(Producto(_id, _nombre));  
    emit NuevoProducto(productos.length, _nombre, _id);  
  
}
```

E1.9

Para almacenar la propiedad del producto, usaremos dos mappings: uno que realiza un seguimiento de la dirección que posee un producto y otra que realiza un seguimiento de cuántos productos tiene un propietario.

- Crearemos un mapping público denominado productoAPropietario. La clave será un uint y el valor una dirección.

```
mapping (uint => address) public productoAPropietario;
```

- Crearemos otro mapping llamado propietarioProductos, donde la clave es una dirección y el valor es un uint

```
mapping (address => uint) propietarioProductos;
```

E1.10

Crearemos una nueva función Propiedad para asignar la propiedad del producto a quien llamó la función. Añadiremos los parámetros que sean necesarios.

- Actualizaremos el mapping productoAPropietario para almacenar msg.sender bajo ese productId.
- Aumentamos propietarioProductos para msg.sender

```
function Propiedad(uint _productId) private {  
    productoAPropietario[_productId] = msg.sender;  
    propietarioProductos[msg.sender] += 1;  
}
```

E1.11

Crearemos una nueva función llamada getProductosPorPropietario que devuelva los productos de un propietario específico. La función debe tener un argumento, una dirección llamada _propietario. Será una función externa (solo las funciones externas pueden llamarla) y de vista (informa al compilador de que la función no modificará las variables de estado). La función debería devolver un uint[] (un array de uint) como ubicación de datos en la memoria. Dentro del cuerpo de la función:

- Declaremos un uint llamado contador de valor 0. Usaremos esta variable para realizar un seguimiento del índice en nuestro array de resultados.

- Declaremos una variable memory uint[] llamada resultado, un nuevo array uint[]. La longitud del array debe ser los productos que posee un _propietario, que podemos buscar en nuestro mapping con: propietarioProductos [_propietario].
- Escribiremos un bucle for que recorra todos los productos, compare su propietario para ver si tenemos una coincidencia y lo envíe al array de resultados antes de devolverlo.

```
function getProductosPorPropietario(address _propietario) view external returns (uint[] memory) {
    uint contador = 0;
    uint longitud = propietarioProductos[_propietario];
    uint[] memory resultado = new uint[](longitud);

    for (uint x = 0; x < productos.length; x++){
        uint id_producto = productos[x].identificacion;
        if(productoAPropietario[id_producto] == _propietario){
            resultado[contador] = id_producto;
            contador+=1;
        }
    }

    return resultado;
}
```

A continuación, proporciono algunas pruebas del ejercicio.

Creando un producto:

crearProducto...

tuberia

▼

```
{
  "from": "0x032ACfAFb4Ffa8acb556e4b7535Ee542bc152726",
  "topic": "0x0638820bbfa7f1ab0bfb9054649b6198da60bd10654cbfe5c50d26b1bec65878",
  "event": "NuevoProducto",
  "args": {
    "0": "1",
    "1": "tuberia",
    "2": "22",
    "ArrayProductoId": "1",
    "nombre": "tuberia",
    "id": "22"
  }
}
```

Obteniendo los productos de un propietario:


getProductosP...

0x78731D3Ca6b7E34aC0F8;

```
decoded output {
  "0": "uint256[]: 22,19,18"
}
```

EJERCICIO 2

En este ejercicio debemos modificar el código dado para que cualquier token pueda ser comprado con Ether, siempre que el propietario todavía tenga suficientes tokens. 1 token debería costar 5 Ether después de la nueva implementación. Si la cantidad de Ether enviada no es suficiente o el propietario no tiene suficientes tokens, se debe enviar un mensaje de error. Mostraremos la cantidad de Ether que hay en el contrato inteligente.


```
function comprarToken(uint256 _amount) public payable{  infinite gas
    require(msg.value >= _amount * 5, "Ether insuficiente para comprar tokens");
    require(users[owner].tokens >= _amount, "Tokens insuficientes disponibles");



    uint256 cost = _amount * 5;
    users[owner].tokens -= _amount * 5;
    users[msg.sender].tokens += _amount;

    payable(owner).transfer(cost);
}

function getCantidadEther() public view returns (uint256) {  339 gas
    return address(this).balance;
}
```

Compramos un token:

ACCOUNT 


0x617...5E7f2 (99.99999999)  

GAS LIMIT

3000000


VALUE

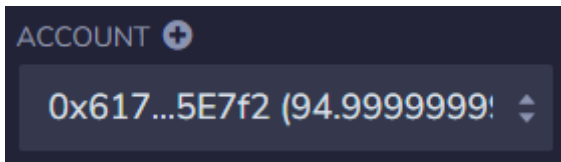
5

Ether 

Balance: 4.99999999999999995 ETH

comprarToken

1 



```
decoded input {
  "uint256 _amount": "1"
}
```

Si intentamos comprar un token sin ether suficiente:

```
The transaction has been reverted to the initial state.
Reason provided by the contract: "Ether insuficiente para comprar tokens".
Debug the transaction to get more information.
```

Si intentamos comprar tokens sin que el owner cuente con ellos:

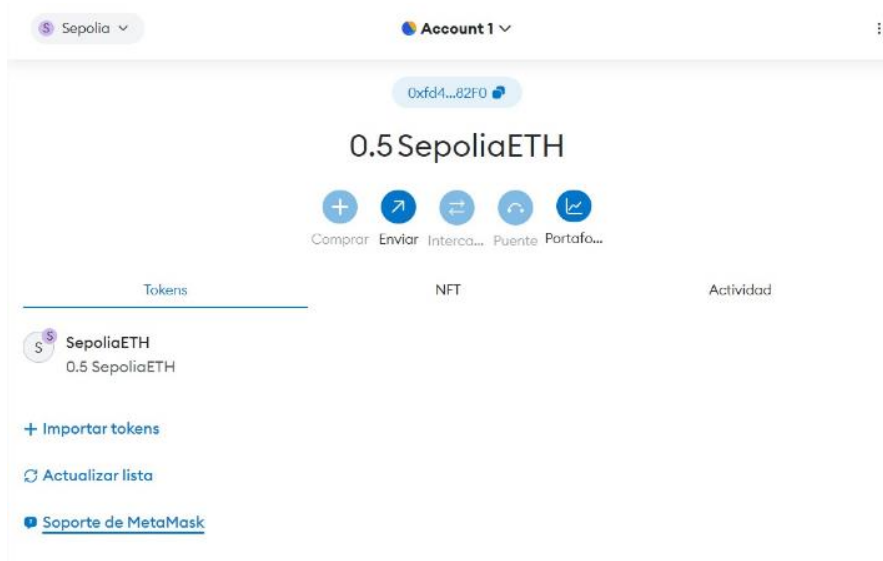
```
The transaction has been reverted to the initial state.
Reason provided by the contract: "Tokens insuficientes disponibles".
Debug the transaction to get more information.
```

EJERCICIO 3

En este ejercicio deberemos familiarizarnos con Metamask.

E3.2

Obtener ETH de prueba de Sepolia.



E3.3


Enviar algo de Sepolia ETH a tus compañeros de clase.

Enviar

OxdB268237a88f77821bcb999Ae9a95C028D4c13b1

X

Activo:

 **SepoliaETH**
Saldo: 0.49330209 SepoliaETH

Importe:

Máx.

0.001 SepoliaETH

No hay tasa de conversión disponible

Gas (estimada)

0.00509671 SepoliaETH

Probablemente en < 30 segundos

Tarifa máxima: 0.00686953 SepoliaETH

Cancelar

Siguiente

Sepolia

Account 1

⋮

Oxfd4...82F0

0.4933 SepoliaETH

+

↗

↻

↶

📈

Comprar Enviar Interca... Puente Portafol...

Tokens

NFT

Actividad

Oct 17, 2023

↗

Enviar

Confirmado

-0.001 SepoliaETH

-0.001 SepoliaETH

🔗 Soporte de MetaMask

E3.4

Realizar un seguimiento de las transacciones en Sepolia Blockchain Explorer:
<https://sepolia.etherscan.io/>

The screenshot shows the Etherscan Sepolia Blockchain Explorer interface. At the top, the Etherscan logo and navigation links (Home, Blockchain, Tokens, NFTs, Misc) are visible. The main header displays the address `0xfd420d507D03C0CE8C17b057eBA5AA5E943f82F0`. Below this, there are three tabs: Overview, More Info, and Multi Chain. The Overview tab is active, showing the ETH balance as `0.493302089750164 ETH`. The More Info tab shows the last and first transactions sent, both from 2 minutes ago. The Multi Chain tab shows the multichain addresses as N/A. Below the tabs, there are two buttons: Transactions and Token Transfers (ERC-20). The Transactions button is active, showing a list of transactions. The list shows the latest 2 transactions from a total of 2 transactions. The first transaction is a transfer of 0.001 ETH from `0xa0790e5c6f842051d...` to `0xd82682...8D4c13b1` with a transaction fee of 0.00569791. The second transaction is a transfer of 0.5 ETH from `0x7Ed746...F7FB271f` to `0xfd420d...943f82F0` with a transaction fee of 0.01302964.

| Transaction Hash | Method | Block | Age | From | To | Value | Txn Fee |
|-------------------------------------|----------|---------|------------|----------------------------------|----------------------------------|-----------|------------|
| <code>0xa0790e5c6f842051d...</code> | Transfer | 4507683 | 2 mins ago | <code>0xfd420d...943f82F0</code> | <code>0xd82682...8D4c13b1</code> | 0.001 ETH | 0.00569791 |
| <code>0x132f1874b021364a...</code> | Transfer | 4507672 | 6 mins ago | <code>0x7Ed746...F7FB271f</code> | <code>0xfd420d...943f82F0</code> | 0.5 ETH | 0.01302964 |

EJERCICIO 4

En este ejercicio se pide completar la lección Solidity: Beginner to Intermediate Smart Contracts. El código se ha incluido en Github.

The screenshot shows a progress bar with six lessons listed on the left and their completion status on the right. Each lesson has a blue progress bar and a dropdown arrow. The lessons and their completion status are:

- Creando la Fábrica de Zombies: 100% completed
- Los Zombis Atacan A Sus Victimias: 100% completed
- Conceptos Avanzados de Solidity: 100% completed
- Sistema de Batalla Zombie: 107% completed
- ERC721 & Crypto-Coleccionables: 100% completed
- App Front-ends & Web3.js: 100% completed


EJERCICIO 5


Aquí se nos pide escoger y probar uno de los contratos de OpenZeppelin. Yo he escogido Box.sol:

```
// contracts/Box.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

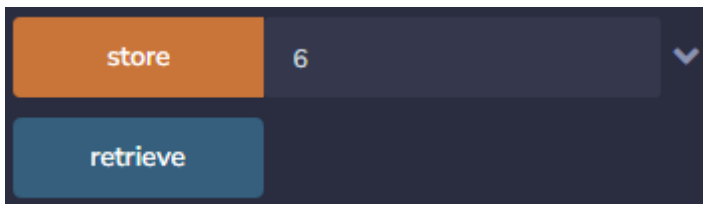
contract Box {
    uint256 private _value;

    // Emitted when the stored value changes
    event ValueChanged(uint256 value);

    // Stores a new value in the contract
    function store(uint256 value) public {  infinite gas
        _value = value;
        emit ValueChanged(value);
    }

    // Reads the last stored value
    function retrieve() public view returns (uint256) {  2415 gas
        return _value;
    }
}
```

Este código guarda un valor en el contrato con store y devuelve el último valor guardado con retrieve:



The interface shows a dark-themed UI. At the top, there is an orange button labeled 'store'. To its right is a dark grey dropdown menu currently displaying the number '6'. Below the 'store' button is a blue button labeled 'retrieve'.

```
{
  "from": "0xEaAB3C6dbC000c7cBA784923FE142b771F39624",
  "topic": "0x93fe6d397c74fdf1402a8b72e47b68512f0510d7b98a4bc4cbdf6ac7108b3c59",
  "event": "ValueChanged",
  "args": {
    "0": "6",
    "value": "6"
  }
}
```