Guía: Smart Contracts

Introducción a Blockchain - Primer Semestre 2025

Preliminares

Esta guía práctica tiene como objetivo desarrollar habilidades básicas en relación a la comprensión y desarrollo de contratos inteligentes. La dificultad de los ejercicios es incremental, siempre es bueno buscar en la bibliografía y animarse a investigar en internet.

Conceptos básicos

Ejercicio 1. ¿Cuál es el valor de la variable pública `myPublicVar` y de la variable privada `myPrivateVar` después de deployar este contrato y llamar a las funciones correspondientes?

```
contract VariableExercise {
    uint public myPublicVar = 10;
    uint private myPrivateVar = 20;

    function getMyPublicVar() public view returns (uint) {
        return myPublicVar;
    }

    function getMyPrivateVar() public view returns (uint) {
        return myPrivateVar;
    }
}
```

Ejercicio 2. ¿Cuál(es) de las siguientes afirmaciones es/son correcta(s)?

```
contract VariableExercise {
    uint internal myInternalVar = 30;
    uint external myExternalVar = 40;

    function getMyInternalVar() public view returns (uint) {
        return myInternalVar;
    }
}
```

a. La variable `myInternalVar` sólo puede ser accedida desde contratos herederos.
b. La variable `myExternalVar` puede ser accedida desde fuera del contrato mediante una transacción externa.

c. La función `getMyInternalVar` es innecesaria ya que las variables internas son accesibles directamente desde cualquier contrato.

Ejercicio 3. Crear un contrato llamado `Bank` que tenga una variable pública llamada `balance` de tipo `uint` y una función pública llamada `deposit` que tome un parámetro `amount` de tipo `uint` y aumente el valor de balance por el monto proporcionado. Asegurarse de que `balance` comience con un valor predeterminado de 0.

Ejercicio 4. Ampliar el contrato `Bank` incluyendo una variable privada llamada `owner` de tipo `address`, que almacenará la dirección del propietario del contrato. Agregar una función pública llamada `getOwner` que devuelva la dirección del propietario. Solo el propietario del contrato debe poder llamar a esta función.

Importante: a partir del ejercicio 5, se comienzan a ejercitar conceptos de desarrollo de smart contracts que forman parte del segundo módulo de la materia. Es decir, este contenido **no será evaluado en el primer parcial.**

Ejercicio 5. Desarrollar un contrato llamado `Votacion` que tenga una variable interna llamada `votos` de tipo `mapping(address => uint)`, que almacenará el número de votos para cada dirección. También incluir una función externa llamada `votar` que tome un parámetro `candidato` de tipo `address` y permita a cualquier cuenta externa emitir un voto para el candidato especificado. La función debe aumentar el recuento de votos para el candidato en 1.

Ejercicio 6. Crear un contrato llamado `Loteria` que permita a varias cuentas externas participar en una lotería. Cada cuenta debe enviar una cantidad de `ether` (vamos a usar `ETH` de `Sepolia`) para comprar boletos y participar en el sorteo. Debe tener:

- Una variable pública llamada `ticketPrice` que represente el precio de cada boleto en `wei` (unidad más pequeña de ether).
- Una variable pública llamada `totalTickets` que represente la cantidad total de boletos disponibles para la lotería.
- Un mapeo llamado `ticketsBought` que asocie cada cuenta con el número de boletos comprados.
- Una función pública llamada `buyTickets(uint numberOfTickets)` que permita a las cuentas externas comprar una cantidad específica de boletos. Asegúrate de que el número de boletos comprados no exceda el límite de boletos disponibles. Además, debe actualizarse el mapeo `ticketsBought` con la cantidad de boletos comprados por cada cuenta.
- Una función pública llamada `getMyTickets()` que devuelva el número de boletos comprados por la cuenta que llama a la función.

- Una función pública llamada `getTotalTicketsSold()` que devuelva la cantidad total de boletos vendidos hasta el momento.
- Una función pública llamada `drawWinner()` que seleccione aleatoriamente una cuenta participante como ganadora y transfiera el monto acumulado de la lotería al ganador. Asegúrate de que solo el propietario del contrato pueda llamar a esta función.
- Una función pública llamada `getContractBalance()` que devuelva el saldo actual del contrato en ether.

Ejercicio 7. Imaginate que sos un artista. Queres compartir tus creaciones con el mundo, pero te tenes que asegurar de que tu trabajo sea auténtico, protegido y reconocido adecuadamente en el mercado. Sin embargo, te preocupa que la propiedad y la autenticidad de tus obras puedan ser fácilmente cuestionadas.

Para resolver este desafío, se te presenta la idea de utilizar blockchain y contratos inteligentes para registrar y transferir la propiedad de tus obras de arte. La idea es crear un contrato inteligente en la tesnet `Sepolia` que permita el registro y la gestión transparente de tus creaciones, asegurando que tus clientes, compradores y seguidores tengan la certeza de que están adquiriendo obras auténticas y respaldadas por vos como el artista original.

Tu objetivo como artista es pensar en cómo diseñar este contrato inteligente para lograr los siguientes aspectos clave:

1. Registro de obras de arte: debe permitirte registrar cada una de tus obras de arte digital con detalles específicos, como el nombre de la obra, descripción, fecha de creación y cualquier otro metadato relevante.
2. Autenticidad y propiedad: Asegurar que cada obra de arte tenga un propietario inicial, que, en este caso, vas a ser vos, y que esta información sea inmutable.
3. Transferencia de propiedad: deberías poder transferir la propiedad de una obra de arte a un comprador o coleccionista interesado. Solo vos, como propietario actual, podrás iniciar esta transferencia.
4. Transparencia y verificabilidad: Garantizar que la historia de propiedad de cada obra de arte sea transparente y verificable por cualquier usuario en la cadena de bloques.
5. Simplicidad de uso: Asegurarte de que el contrato sea fácil de usar tanto para vos, como artista, como para tus clientes y compradores, de modo que puedan interactuar con él de manera sencilla.

Ante este desafío, debes pensar en cómo abordar cada uno de estos aspectos para crear un contrato inteligente que proteja tus obras de arte y brinde confianza y autenticidad a tus clientes.

Ejercicio 8. Explicar con detalle la funcionalidad de los siguientes contratos.

a. FollowNFT.sol: es un contrato de Lens Protocol, un protocolo que genera interfaces para construir redes sociales en web3.
b. Market.sol: es un contrato de Exactly Finance, un protocolo de lending descentralizado.
c. Spot.sol: es un contrato de MakerDAO, es la plataforma a través de la cual cualquier persona, en cualquier lugar, puede generar la moneda Dai contra activos criptográficos colaterales.

Bibliografía

Palladino, S. (2019). Ethereum for Web Developers: Learn to Build Web Applications on top of the Ethereum Blockchain (1st ed.). Apress.