Contrato de Subasta:

En el Trabajo Practico del modulo 2 se solicitaba realizar un Smart Contract de Una subasta cuyas características y consideraciones fueron indicadas en el Campus, motivo por el cual se expondrán los pasos realizados para concretar la tarea asignada.

ESTRUCTURA DEL CONTRATO: VISIÓN GENERAL

El contrato gestiona una subasta en Ethereum, permitiendo:

* Ofertar con ETH.
* Reembolsar a no-ganadores.
* Determinar automáticamente al mejor postor.
* En caso que un postor no haya ganado por oferta superior, al volver a ofertar se tiene en cuanta su primer oferta agregadas a las subsiguientes.
* Aplicar comisiones.
* Extender la subasta si hay ofertas tardías.
* Gestionar finalización de la subasta
* Determinar quién puede realizar Acciones, como por ejemplo finalizar la subasta

Variables Globales:

* address public owner; Propietario de la subasta (el que la crea).
* uint public deadline; Tiempo límite de la subasta.
* uint public commissionPercent = 2; Comisión fija del 2%.
* uint public timeExtension = 10 minutes; Si alguien oferta en los últimos 10 minutos, se extiende 10 más.

Estructura de una Oferta:

* struct Bid { address bidder; uint amount;}; Representa una oferta individual: quién ofertó y cuánto.

Estructuras de Almacenamiento:

* Bid[] public bidHistory; Lista con todas las ofertas válidas registradas..
* Bid public highestBid; Guarda la oferta más alta acumulada.
* mapping(address => uint) public accumulatedBids; Guarda cuánto ha ofertado cada usuario en total.
* mapping(address => uint[]) public userBidHistory; Guarda un array con todas las ofertas por usuario.
* mapping(address => uint) public refunds; Montos que pueden retirarse por reembolsos parciales.

Modificadores:

* modifier onlyBeforeEnd() { require(block.timestamp < deadline, "The auction has ended.") \_; }; Solo aceptar ofertas antes del deadline.
* modifier onlyOwner() { require(msg.sender == owner, "Only the owner can perform this action."); \_;}; Solo el propietario puede finalizar la subasta.

Constructor:

* constructor(uint \_durationSeconds) { owner = msg.sender; deadline = block.timestamp + durationSeconds;}; Define el owner y calcula la fecha límite.

Funciones:

function placeBid():

* Valida que el monto enviado sea mayor que 0.
* Suma la oferta al total acumulado del usuario.
* Actualiza su historial.
* Comprueba si el total supera la mejor oferta actual en al menos 5%.
* Actualiza la mejor oferta y el historial global.
* Extiende el deadline si quedan menos de 10 minutos.

function endAuction():

* Solo el owner puede llamarla.
* Verifica que ya pasó el deadline.
* Aplica una comisión del 2%.
* Transfiere el resto al owner.
* Emite el evento AuctionEnded.

function withdrawDeposit():

* Si no eres el ganador, puedes retirar tu oferta acumulada luego de la finalización.

function getWinner(): Ganador actual.

function getBidHistory(): Historial de ofertas.

function timeRemaining(): Tiempo restante.

function bidsOf(address user): Ofertas individuales de un usuario.

Eventos:

* event NewBid(address indexed bidder, uint amount);
* event AuctionEnded(address winner, uint winningAmount);

Emiten mensajes para ser capturados por dApps o interfaces web (frontend).