Obligatorio - 2025

Su equipo es contratado para diseñar una solución basada en blockchain que sirva como prueba de concepto funcional para un sistema de gobernanza basado en DAO (Decentralized Autonomous Organization) utilizando tokens ERC-20.

Requerimientos funcionales y no funcionales:

- Cada equipo debe desarrollar los componentes de frontend, backend, smartcontracts y documentación necesarias para el proyecto.
- El proyecto debe ser EVM compliant
- Cada grupo puede elegir el ambiente de desarrollo que desee y le sea más cómodo
- Los tokens ERC20 representarán el poder de voto que poseen los participantes dentro de la DAO
- El owner (dueño) de la DAO debe ser una multisig que podrá hacer las siguientes acciones dentro del contrato:
    - Mintear nuevos tokens ERC20 utilizados para el poder de voto (inicialmente habrán 0 tokens)
    - Cambiar parámetros del funcionamiento de la DAO
    - Transferir ownership de la DAO
- La DAO debe tener asignada una multisig de “pánico”, la misma debe ser configurada por el owner de la DAO.
- Sin una wallet de pánico configurada la DAO no debe poder operar.
- Debe haber una función llamada “pánico” que no recibe parámetros y solo puede ser activada por el owner de la DAO. Esta función al ser activada debe suspender cualquier operación de la DAO excepto la función “tranquilidad”
- La función “tranquilidad” debe poder ser invocada sólo por la multisig de “pánico” y re-establece la operativa de la DAO.
- Para poder votar un usuario debe dejar un cierto número de tokens en staking, la cantidad es un parámetro de la DAO
- Para poder hacer propuestas un usuario debe dejar un cierto número de tokens en staking, la cantidad es un parámetro de la DAO
- Los tokens en staking para votar no se consideran para hacer propuestas y viceversa. Por ende si un usuario desea poder votar y hacer propuestas debe realizar 2 stakings
- dApp:
    - Los usuarios deben poder comprar tokens a cambio de ETH (el valor de cada token es un parámetro de la DAO)
    - Se debe poder ver el balance de tokens que posee el usuario
    - Se deben ver todas las propuestas realizadas en la DAO, junto con un detalle de las mismas (información de la propuesta, votos, quienes votaron y cómo votaron)

    - El usuario debe poder ver el staking que posee (tanto para votos como para propuestas)
    - El usuario debe poder quitar del staking sus tokens
    - Crear propuestas, cada propuesta tiene: título y descripción
    - Votar: A favor o En contra
    - Se debe poder conectar con al menos dos wallets: Metamask obligatorio y una segunda a elección.
    - Las propuestas deben poder filtrarse por estado: ACTIVAS, RECHAZADAS, ACEPTADAS
- Los tokens en staking se deben bloquear por un mínimo de tiempo, el tiempo es un parámetro de la DAO
- Se deben poder crear propuestas y votarlas a favor o en contra.
- Cada propuesta puede estar activa X cantidad de días (definido como parámetro de la DAO), luego de esa cantidad de días no se puede votar más y se toma una decisión.
- La decisión se toma basado en la mayor cantidad de votos recibidos basado en el poder de voto.
- Poder de voto: dependiendo de cuantos tokens se encuentren en el staking será el poder de voto que tenga el usuario. Habrá un parámetro de la DAO que determine cuántos tokens es considerado un poder de voto de 1. Ejemplo:
    - Usuario tiene 10.000 tokens en staking para votar
    - La DAO considera que 1.000 tokens es un VP de 1
    - El usuario tiene un poder de voto de 10
- Debe haber test coverage del 100% de los smart contracts (excepto para contratos de prueba o test en caso de que los hubiere, pero en estos casos debe ser especificado cuales son en la documentación y porque fueron incluidos en el código base)
- Se debe tener en cuenta los casos de borde de cada funcionalidad
- Se debe prevenir los ataques comunes (al menos los dados en clase)

Requerimientos específicos por grupos

A su vez cada grupo debe realizar algunos de los siguientes conjuntos de funcionalidades. Al menos debe haber un grupo que implemente cada conjunto en la clase (es decir no pueden quedar conjuntos sin implementar).

Conjunto A

- Cambiar el poder de voto para que sea cuadrático, es decir que el poder de voto escala como la raíz cuadrada de sus tokens
- Permitir la delegación de voto para una propuesta específica (es decir que un usuario pueda delegar su voto a otra cuenta que no sea parte de la DAO)

Conjunto B

- Permitir una DAO que permita cambiar la estrategía de voto. Por ejemplo en lugar de simple mayoría de votantes puede ser permitir también mayoría de todo los permitidos para votar, sociocracia, etc.

Conjunto C

- Crear un tipo específico de propuesta que sirva para manejar el treasury (ETH en el balance de la DAO).
- La propuesta es una propuesta normal que recibe obligatoriamente dos campos extras:
    - Address a donde enviar los ETH
    - Cantidad de WEIs a enviar
- Si la propuesta se aprueba la DAO debe hacer la transferencia automáticamente

Se pide

- Documento que describa:
    - La arquitectura completa del sistema y justificación de las decisiones tomadas en la misma
    - Diagrama que muestra los contratos en que se encapsuló cada responsabilidad y las responsabilidades que se le asignaron, marcando que requerimiento se cumple en cada una.
    - Aspectos asumidos
    - Desafíos encontrados (si hubo alguno)
    - Resultado de pruebas unitarias
    - Detalle de los flujos principales.
- Documento explicando cómo correr el sistema (backend, frontend, smart contracts, tests, etc.). El profesor correrá en el siguiente ambiente:
    - Ubuntu 24.04
    - Ganache para simular una red Ethereum
    - Visual Studio Code para visualizar el proyecto
    - Hardhat
    - Node
    - Se debe proveer una guía paso a paso cómo levantar cada aspecto del sistema. En caso de tener que instalar alguna librería y/o servidor no especificado anteriormente se debe proveer una guía de como instalarlo.
    - Se debe proveer una guía sobre cómo correr los tests y el test coverage (se validará la completitud del test coverage y si no se cumple con el requerimiento será motivo de no continuar probando el obligatorio).

    - Recordar que se debe explicar cómo probar cada componente (dApp, contratos, backend, etc)
- Video con demo que muestre todas las funcionalidades del sistema, en el caso de las interacciones desde dapp con smart contracts se debe mostrar las transacciones y estados del contrato para validar que realmente se realizó la acción en los smart contracts.
- La versión final de los contratos deben ser publicados en la red de testnet de Ethereum o Polygon
- Los contratos de la versión final deben ser verificados ya sea en Etherscan (testnet) o Polygonscan (testnet) dependiendo de en qué red se decidió hacer el deploy. Tener en cuenta que cada grupo es responsable de adquirir con tiempo los tokens de cada red necesarios para pruebas y deploys.

Información importante

- **Lectura del obligatorio:** 3/4/2025
- **Entrega del obligatorio:** 26/6/2025
- **Defensa del obligatorio:** A definir
- **Cantidad de participantes por grupo:** 3 estudiantes
- **Puntaje mínimo:** 1
- **Puntaje máximo:** 60

El trabajo es individual de cada grupo. Les recordamos que copiar, cometer plagio o recibir ayuda no autorizada de terceros en la realización de trabajos académicos es considerado una falta grave según el Art. 51 del Reglamento Estudiantil (http://www.ort.edu.uy/varios/pdf/documento001.pdf).

La no presentación en la defensa implica la pérdida de la totalidad de los puntos.