Desafio Tecnico

Fecha: **07/24**
Autor: **Martin Escudero**

Resumen

La finalidad de este desafío es evaluar cómo se desempeñan los candidatos frente a una tarea propuesta. No es 100% necesario que el código sea funcional, pero ¡si lo logran, mejor!

Descripción General

La arquitectura propuesta para la Dapp se divide en tres componentes principales: la interfaz de usuario (frontend), el contrato inteligente (smart contract) y la red blockchain (testnet). A continuación, se detalla cada uno de estos componentes y su interacción.

Componentes de la Arquitectura

1. **Interfaz de Usuario (Frontend)**
    - **Tecnologías**: React.js o cualquier otro framework de JavaScript, HTML, CSS.
    - **Descripción**: La interfaz de usuario contendrá los campos de entrada para interactuar con el smart contract, como ingresar datos y recuperar datos de la blockchain.
    - **Funcionalidades**:
        - Campo input para ingresar un dato en la blockchain.
        - Botón para enviar el dato al smart contract.
        - Campo para mostrar el dato recuperado de la blockchain.
        - Conexión con la wallet Metamask para firmar las transacciones.
2. **Contrato Inteligente (Smart Contract)**
    - **Tecnologías**: Solidity.
    - **Descripción**: El smart contract se encargará de almacenar y recuperar datos en la blockchain.
    - **Funcionalidades**:
        - Función para almacenar un dato.
        - Función para recuperar un dato.
        - Eventos para notificar a la interfaz de usuario sobre cambios en el estado del contrato.
3. **Red Blockchain (Testnet)**
    - **Redes Sugeridas**: Sepolia, Zksync Sepolia (Ideal).
    - **Descripción**: La red blockchain se utilizará para desplegar el smart contract y para realizar las transacciones necesarias.
    - **Proveedores de Nodos**:
        - QuickNode

        - Infura
        - Alchemy
    - **Faucets**
        - https://cloud.google.com/application/web3/faucet/ethereum/sepolia
        - https://learnweb3.io/faucets/zksync_sepolia/

Requisitos Detallados

1. **Interfaz de Usuario (Frontend)**
    - Crear una página web simple utilizando React.js.
    - Implementar un campo input para ingresar datos.
    - Implementar un botón para enviar los datos al smart contract.
    - Implementar una sección para mostrar los datos recuperados del smart contract.
    - Conectar la aplicación a Metamask para firmar transacciones.
2. **Contrato Inteligente (Smart Contract)**
    - Escribir el smart contract en Solidity.
    - Implementar funciones para almacenar y recuperar datos.
    - Desplegar el smart contract en una testnet (Sepolia o Zksync Sepolia).
3. **Integración y Despliegue**
    - Utilizar Hardhat, Truffle o Remix para compilar y desplegar el smart contract.
    - Conectar la interfaz de usuario con el smart contract utilizando Web3.js o Ethers.js.
    - Documentar el proceso de instalación y despliegue en el repositorio.

Entregables

1. **Arquitectura / Diagrama (Obligatorio)**: Gráfico simple donde se visualice la solución a nivel macro.
2. **Descripción General (Obligatorio)**: Detallar los puntos necesarios para el entendimiento de la solución.
3. **Tecnologías Involucradas (Obligatorio)**: Detallar las tecnologías utilizadas.
4. **Repositorio (Obligatorio)**: Indicar el link al repositorio y su branch/tag correspondiente.
5. **Implementación (Obligatorio)**: Pasos para instalar/desplegar la solución y información de administración del mismo.

6. **Step by Step (Opcional)**: Si aplica, dejar el paso a paso para reproducirlo, comandos ejecutados, prints de pasos en consolas, etc.
7. **Anexo (Opcional)**: Incluir cualquier información relacionada con la tarea.

Glosario de Términos Blockchain

Blockchain: Una base de datos distribuida que mantiene un registro continuo y creciente de transacciones ordenadas en bloques.

Dapp (Aplicación Descentralizada): Aplicaciones que funcionan en una red P2P en lugar de en un solo servidor, y que utilizan smart contracts para gestionar la lógica de la aplicación.

Smart Contract (Contrato Inteligente): Programa que se ejecuta en la blockchain y que se activa cuando se cumplen ciertas condiciones.

Testnet: Una red de prueba donde se pueden realizar experimentos sin costo, utilizando tokens que no tienen valor real.

Ethereum: Una plataforma blockchain descentralizada que permite la creación de smart contracts y Dapps.

Sepolia: Una red de prueba (testnet) compatible con Ethereum.

ZkSync: Una solución de escalado para Ethereum que utiliza pruebas de conocimiento cero para procesar transacciones de manera eficiente y económica.

Metamask: Una extensión de navegador que permite a los usuarios interactuar con Dapps y smart contracts en la blockchain de Ethereum.

Typescript: Un lenguaje de programación que es un superconjunto de JavaScript y agrega tipos estáticos.

Solidity: Un lenguaje de programación diseñado para desarrollar smart contracts en la blockchain de Ethereum.

Web3.js: Una colección de bibliotecas que permiten interactuar con una blockchain Ethereum local o remota.

Ethers.js: Una biblioteca para interactuar con la blockchain Ethereum, que ofrece funcionalidades similares a Web3.js.

Hardhat: Un entorno de desarrollo para la creación, prueba y despliegue de smart contracts en la blockchain de Ethereum.

Truffle: Un entorno de desarrollo, marco de pruebas y canal de activos para la creación de smart contracts en la blockchain de Ethereum.

Remix: Un entorno de desarrollo web basado en navegador para escribir, compilar y desplegar smart contracts en Ethereum.