En este examen, los estudiantes aplicarán los conocimientos fundamentales adquiridos en el Módulo 2 desarrollando, desplegando y documentando su primer proyecto completo de contrato inteligente.

Por qué esto es importante

Este examen es un paso crítico en tu viaje como desarrollador Web3. No se trata solo de escribir Solidity, sino de escribir contratos inteligentes seguros, mantenibles y bien documentados. Aplicarás las mejores prácticas en desarrollo, patrones de seguridad que protegen protocolos del mundo real, y aprenderás a presentar tu trabajo de manera profesional a través de GitHub.

Esta tarea práctica tiende un puente entre la teoría y la aplicación, preparándote para contribuciones y colaboraciones en el ecosistema de Ethereum.

Objetivos del Examen

- Aplicar los conceptos centrales de Solidity aprendidos en clase.
- Seguir patrones de seguridad.
- Usar comentarios y una estructura limpia para mejorar la legibilidad y mantenibilidad del contrato.
- Desplegar un contrato inteligente completamente funcional en una testnet.
- Crear un repositorio de GitHub que documente y muestre tu proyecto.

Descripción de la Tarea y Requisitos

Tu tarea es recrear el contrato inteligente **KipuBank** con toda la funcionalidad y documentación, como se describe a continuación.

Características de KipuBank:

- Los usuarios pueden depositar tokens nativos (ETH) en una bóveda personal.
- Los usuarios pueden retirar fondos de su bóveda, pero solo hasta un límite fijo por transacción, representado por una variable inmutable. El contrato aplica un límite máximo de depósito global (`bankCap`), establecido durante el despliegue.
- Las interacciones internas y externas deben seguir las mejores prácticas de seguridad y usar sentencias `revert` con errores personalizados claros si no se cumplen las condiciones.
- Se deben emitir eventos tanto en los depósitos como en las retiradas exitosas.
- El contrato debe llevar un registro del número de depósitos y retiradas.
- El contrato debe tener al menos una función `external`, una `private` y una `view`.

Prácticas de Seguridad a Seguir:

- Usar errores personalizados en lugar de cadenas de texto en `require`.
- Respetar el patrón checks-effects-interactions y las convenciones de nomenclatura.

Ing. Jesús Parra
parra.jesus@gmail.com

- Usar modificadores donde sea apropiado para validar la lógica.
- Manejar las transferencias de tokens nativos de forma segura.
- Mantener las variables de estado limpias, legibles y bien comentadas.
- Añadir comentarios NatSpec para cada función, error y variable de estado.
- Aplicar las convenciones de nomenclatura adecuadas.

Entregables

Envía lo siguiente a través de la plataforma:

- **URL del Repositorio de GitHub**
  - Un repositorio público llamado `kipu-bank` que contenga:
    - Tu código del contrato inteligente dentro de una carpeta `/contracts`. Debe tener los siguientes componentes:
      - Variables `Immutable` o `Constant`
      - Variables de almacenamiento (`Storage`)
      - `Mapping`
      - Eventos (`events`)
      - Errores Personalizados (`Custom Errors`)
      - Constructor (`Constructor`)
      - Modificador (`Modifier`)
      - Función `External Payable`
      - Función `Private`
      - Función `External View`
    - Un `README.md` bien estructurado que incluya:
      - Una descripción de lo que hace el contrato.
      - Instrucciones de despliegue.
      - Cómo interactuar con el contrato.
- **Dirección del Contrato Desplegado**
  - En una testnet con el código fuente verificado en un explorador de bloques.

Este proyecto pasará a formar parte de tu creciente portafolio público y demostrará tu capacidad para entregar soluciones seguras y funcionales en Web3.

Ing. Jesús Parra
parra.jesus@gmail.com