GUÍA DEL ASESOR BLOCKCHAIN

LÍNEA BLOCKCHAIN		
II. NÚCLEO ESPECÍFICO	**e. Desafíos económicos de la blockchain**	
Sesión 13	**Estrategias de implementación blockchain**	**Tiempo: 3 hora**

Con antelación verificar si se cuenta con:

Aula virtual	**X**	**Accesos**	**X**	**Presentación**	**X**	**Recursos**		**Cámara y micrófono**	**X**

- Explorador Web (chrome, mozilla o brave)
- Complemento MetaMask (disponible en exploradores chrome, mozilla y brave)

Objetivos

- Conocer los componentes de un contrato inteligente
- Conectarse a un contrato inteligente de una red de prueba
- Diferenciar las arquitecturas de implementación de Blockchain

Introducción al tema

En esta sesión se expondrá el concepto de aplicaciones descentralizadas partiendo de las bases y diseños de contratos inteligentes, para profundizar posteriormente en su integración con interfaces de usuario y otros sistemas complementarios.

Contratos Inteligentes:
Los contratos inteligentes son lógicas definidas y comunicadas a los nodos de una Blockchain, a cada contrato se le asigna un hash que lo hace único, de modo que todas las funciones públicas del contrato sean accesibles por los participantes de la red y sus reglas respetadas por los mineros mediante algoritmos criptográficos.

Cualquier persona puede definir sus propias reglas y crear nuevos contratos, por este motivo sus correctas funcionalidades y dinámicas dependen completamente de su creador; se podría decir que es inteligente porque hace respetar las reglas impuestas por su creador, pero la inteligencia realmente es diseñada por una persona o grupos de personas y se codifica en un lenguaje de programación que pueden interpretar los equipos que validan los datos.

Componentes:

- **Condiciones Iniciales**: Al momento de desplegar un contrato (enviarlo a la red y hacerlo accesible por los demás participantes) se ejecuta en primera instancia una función para determinar las condiciones iniciales. En esta sección se definen valores importantes para la dinámica que se quiera lograr como límites de suministro en monedas y otros parámetros. En el diseño del contrato se requiere definir con claridad los valores iniciales para posteriormente definir en las funciones las posibilidades de modificar esos valores.

- **Variables:** En todo contrato se deben declarar las variables donde se almacenará información, como un contador de procesos o variables internas para el funcionamiento de las lógicas.
- **Estructuras de Datos:** En muchos casos de uso se tiene información con estructuras más complejas como por ejemplo una persona puede tener un nombre, apellido, cédula y dirección. Es importante definir los paquetes o estructuras de datos que se van a manejar en el proceso.
- **Listas:** Una vez definidas las estructuras de datos se pueden crear listas o arreglos de datos para registrar información usando una estructura predefinida. Por ejemplo, si tenemos la estructura persona, podemos hacer listados de personas como estudiantes y hacer otra lista con la misma estructura "persona" para docentes, cada lista puede registrar datos con la estructura "persona" pero los almacena en lugares de memoria diferentes.
- **Mapeos:** Una vez definidas las variables, las estructuras de datos o listados podemos crear relaciones simples entre ellos, por ejemplo relacionar el index de una persona en un listado de personas directamente con su clave pública, para tener una relación entre ambas estructuras de datos (dirección Pública y estructura persona).
- **Funciones:** Las funciones permiten crear hilos de procesos con información de entrada y devolver datos a la salida
    - **Consulta**: Las funciones de consulta pueden realizarse sin costo desde cualquiera de los nodos que tenga una copia de la Blockchain disponible.
    - **Registro**: Las funciones de registro implican almacenamiento o modificación a los datos almacenados en el contrato. Estas funciones implican un esfuerzo para la red en cuanto a procesamiento y por lo tanto consumen costos de comisión.
- **Visibilidad y Seguridad:** Las variables, listados, mapeos y funciones pueden parametrizarse para que sean públicos (accesible por cualquiera en la red) o privados (accesible por funciones internas del contrato). La seguridad en cuanto al acceso a los datos se gestiona principalmente definiendo los datos como privados y concediendo el acceso a ellos mediante funciones públicas que tienen lógicas especiales para detectar la dirección que consulta y otorgar sólo la información privada perteneciente a esa dirección según su nivel de acceso.

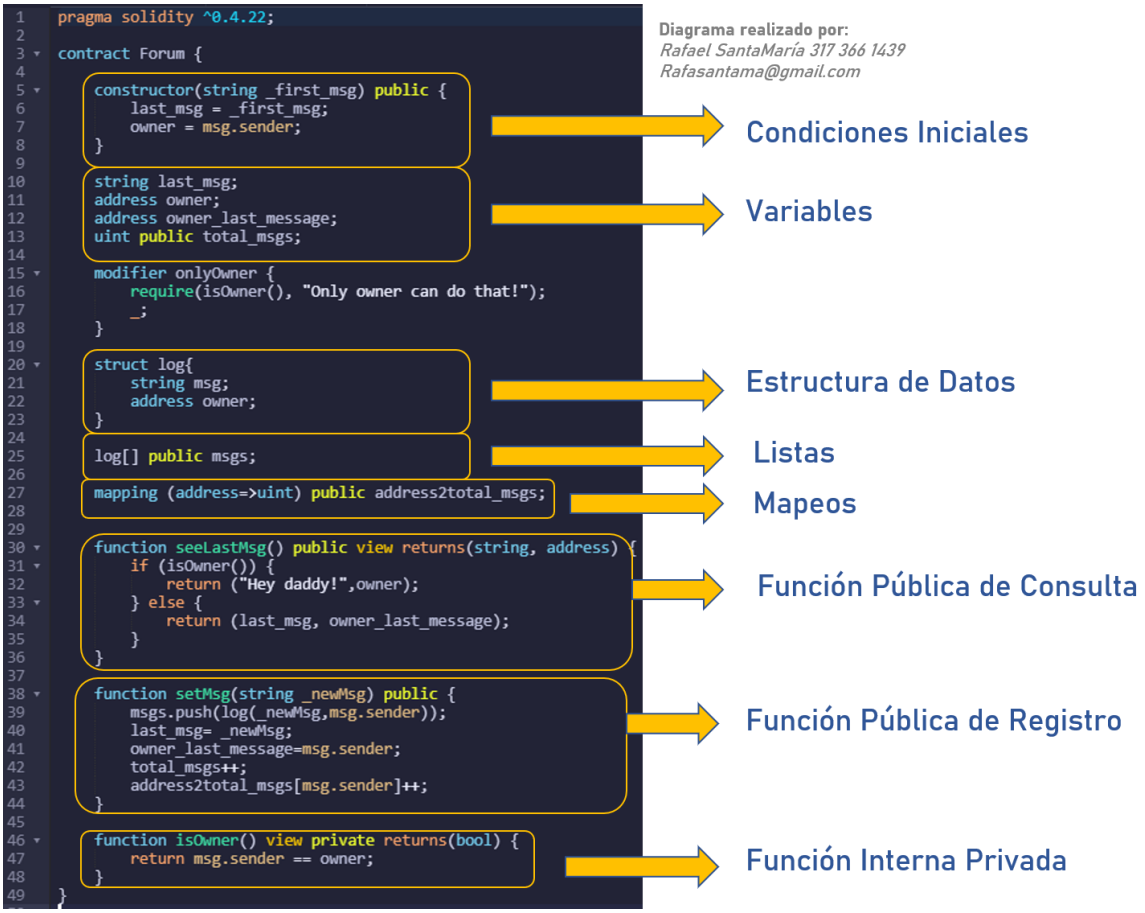
Ejemplo:

Supongamos que se quiere crear un foro inmutable, es decir un lugar donde cualquier persona pueda escribir un mensaje y ese mensaje quede registrado junto con clave pública como el emisor del mismo. Tendremos una función para consultar el último mensaje y también estarán todos los mensajes públicos en un listado. A la función de último mensaje le programaremos una lógica especial para reconocer al dueño del contrato y darle un saludo especial (así se prueba que el contrato efectivamente reconoce la clave pública de quien interactúa con él y reconoce a su creador o administrador).

Componentes del Contrato: Forum	
Condiciones Iniciales	• Primer mensaje • Dueño del contrato
Variables	• Último mensaje • Dueño del último mensaje • Total de mensajes

Estructuras de Datos	• log
	  ○ mensaje
	  ○ dirección del dueño

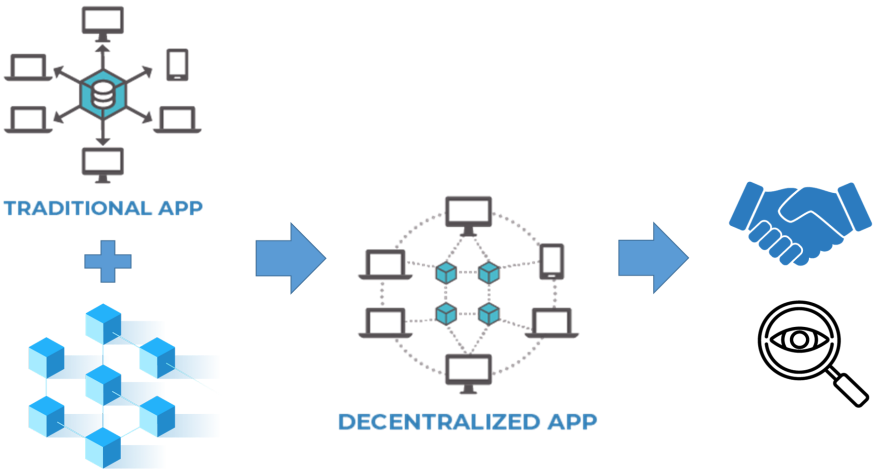
Listas	Listado de logs - Público
Mapeos	Dirección hacia total de mensajes registrados
Consulta	Ver último mensaje
Registro	Registrar Nuevo Mensaje



En este Taller también se darán a conocer nuevas herramientas para el despliegue de contratos inteligentes en redes de prueba para prototipado y su acceso o consulta mediante herramientas IDE. Las redes de prueba permiten desplegar contratos para verificar su interacción desde diferentes billeteras, validando su interacción. También permiten realizar consultas y registros al mismo contrato en una red provisional en la cual podemos conseguir saldo para pagar las comisiones de manera gratuita mediante el uso de faucets.

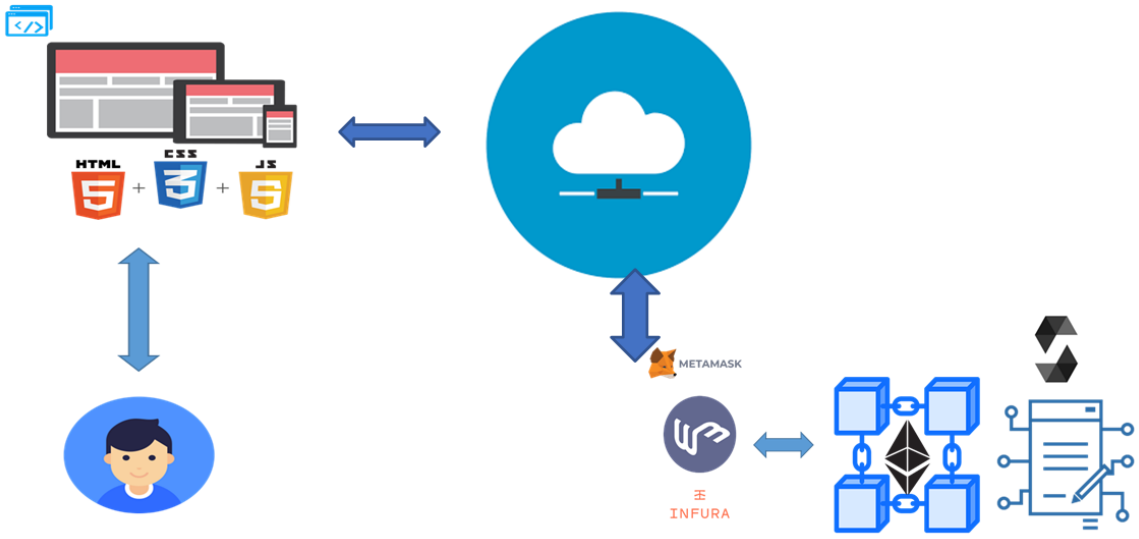
Finalmente, en esta sesión se hará énfasis en las diferentes formas de integrar los contratos inteligentes con interfaces de usuario, bases de datos y demás sistemas para crear aplicaciones descentralizadas, mejor conocidas como D-Apps. Al agregar

contratos inteligentes a una aplicación se convierte en un D-App, lo cual le agrega un valor adicional de confianza y transparencia, según el diseño del contrato.



Dependiendo de las funcionalidades que requiera la D-App, será necesario usar alguno de los siguientes esquemas:

D-App Solo Blockchain: Algunas D-App utilizan una interfaz gráfica de usuario conectada únicamente a un contrato inteligente.

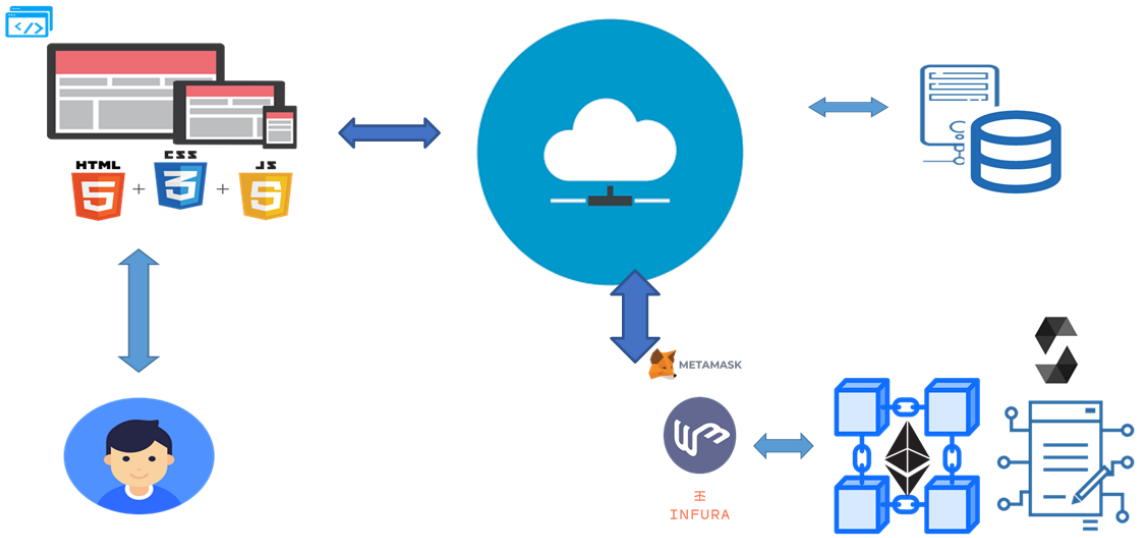


Por lo general al utilizar esta opción es necesario que el usuario final gestione su propia clave privada a través de algún proveedor como Metamask u otro mecanismo de gestión de sus llaves privadas, para poder usarlas en la gestión de firmas en las funciones del contrato. En este tipo de aplicaciones toda la información debe almacenarse en el contrato inteligente por ende en la Blockchain lo cual consume mayores costos en comisiones; por lo tanto es necesario depurar muy bien qué tipo de datos se van a almacenar. Por otro lado este tipo de aplicaciones no utiliza un sistema de registro y logeo de usuarios convencional, sino que se vale de las direcciones públicas y privadas para reconocer los diferentes actores que interactúan con ella.

Algunos de los ejemplos más comunes de este tipo de D-Apps los encontramos en los ecosistemas De-Fi tal como Uniswap en Ethereum o PancakeSwap en Binance Smart Chain, en ambos casos es necesario usar Metamask o algún proveedor tercero para interactuar con la D-App conectada a la red respectiva. Por otro existen otras de Dapps

Full Blockchain, que se valen de otros mecanismos como por ejemplo un archivo que almacena cada usuario el cual contiene su firma. (Ejemplo: Demo Edubits versión demo de D-App de Diplomas en red de prueba rinkeby sin usar Metamask o tercero).

D-App Blockchain + DB: En la mayoría de los casos las D-Apps requieren de otro tipo de sistemas para poder ofrecer una experiencia completa a sus usuarios. En este caso la Blockchain se utiliza para almacenar toda la información crítica y gestionar los permisos de acceso a la misma y la base de datos para el resto de información.



Las bases de datos y sistemas complementarios se utilizan apoyados de la capa de seguridad Blockchain para almacenamiento de archivos, streaming, gestión de sesiones y demás funcionalidades auxiliares. En algunos casos las claves privadas se encuentran asociadas a los usuarios y se encuentran también almacenadas en a base de datos de la D-App, sin embargo esto disminuye la seguridad ya que a pesar de tener una Blockchain el administrador de la base de datos puede tener acceso a las claves privadas de los usuarios perjudicando la D-App respecto a su centralización.

Método: Propuesta pedagógica

La sesión se realizará de forma sincrónica a distancia a través de la plataforma TEAMS.

- Primera parte: charla magistral de un invitado especial para todos los participantes de la línea blockchain, se realizará con apoyo audiovisual y una ronda de preguntas para resolver las dudas de los participantes presentes.
  Duración: 1 hora.

- Segunda parte: los participantes se dirigen a sus respectivos grupos con su asesor asignado. El asesor contextualizará y profundizará el tema de la sesión, y responderá a las preguntas de los participantes que tengan relación con la conferencia. Se realizarán las actividades previstas con acompañamiento del asesor blockchain, utilizando demos para reforzar conocimientos a través de la práctica.
  Duración: 2 horas.

Habilidades y competencias desarrolladas

- Diseño Conceptual de Contratos Inteligentes
- Despliegue y Comunicación con contratos inteligentes en Redes de Prueba
- Análisis de requerimientos y selección de arquitecturas para D-Apps

Recursos

- Explorador Web (Chrome, Mozilla o Brave)
- Billetera: Metamask
- Remix: Herramienta de Desarrollo en Línea
- Faucet de Rinkeby
- Contrato Inteligente de Prueba

Instrucciones

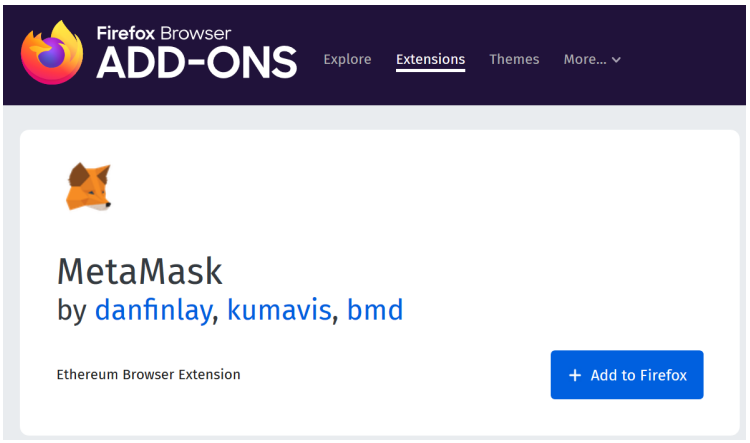
Actividad 1: Definir los componentes del contrato inteligente

Utilizando los conceptos disponibles en la introducción identificar los componentes claves del contrato inteligente para su caso de uso en la siguiente tablas:

Componentes del Contrato:	
Condiciones Iniciales	
Variables	
Estructuras de Datos	
Listas	
Mapeos	
Consulta	
Registro	

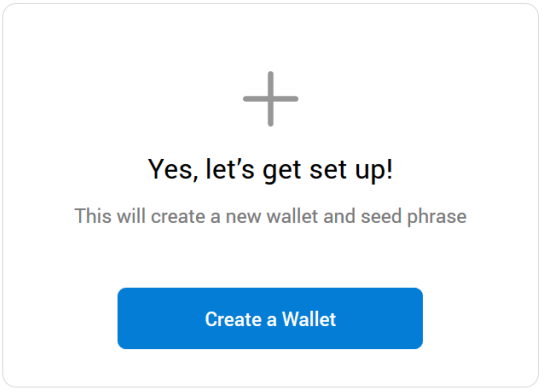
Actividad 2: Contrato Inteligente en Red de Prueba

1. Instalar Metamask

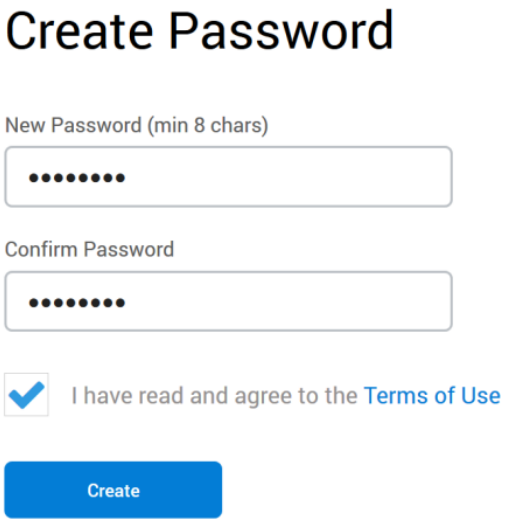


2. Crear y Cargar Billetera

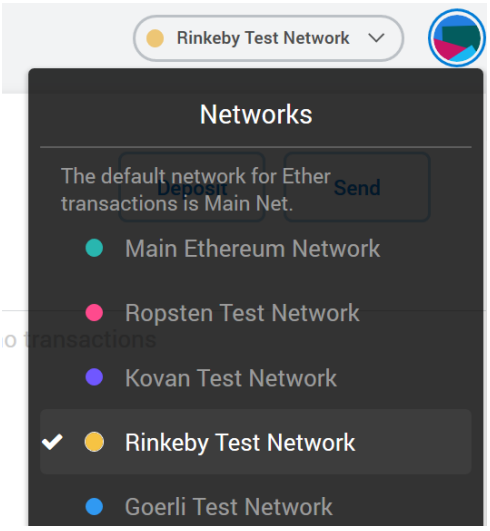
2.1 Crear: Una vez instalado el complemento en el explorado al abrirlo por primera vez tendremos la opción de crear una nueva billetera.



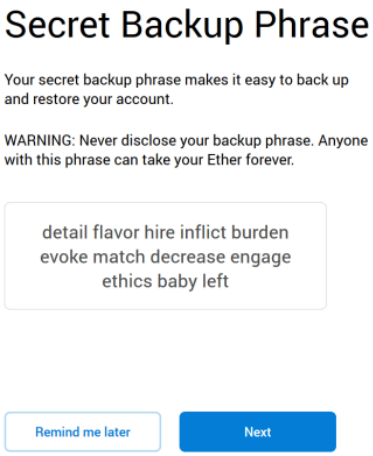
2.2 Definir Contraseña: Para asegurar el acceso a su cuenta debe asignar una contraseña a la misma.



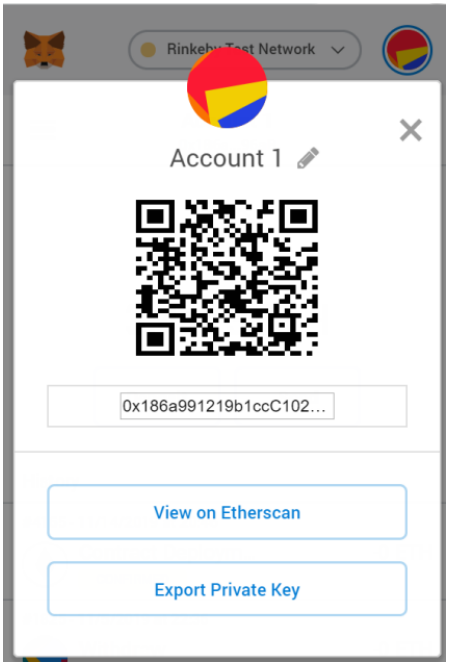
2.3 Conectarse a la red de prueba: Ahora vamos a conectar el explorador Blockchain a la Red de Prueba Rinkeby.



2.4 Guardar Respaldo de Billetera: En caso de pérdida de la contraseña o el equipo donde se encuentra almacenada la billetera, es posible recuperarla usando 12 palabras secretas de respaldo (debe protegerlas de acceso no autorizado ya que el proceso aplico igual para redes principales).

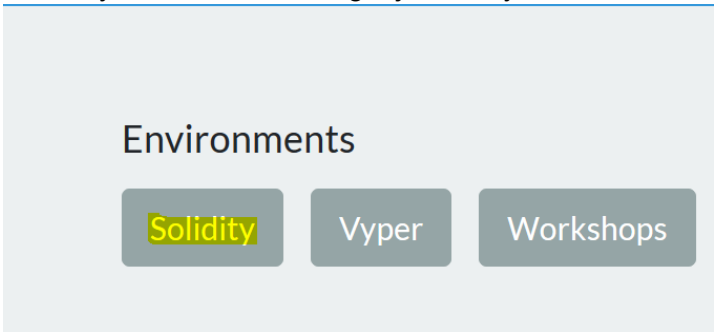


2.5 Copiar Dirección Pública: Para poder recibir saldo en su cuenta es necesario consultar primero su dirección pública.

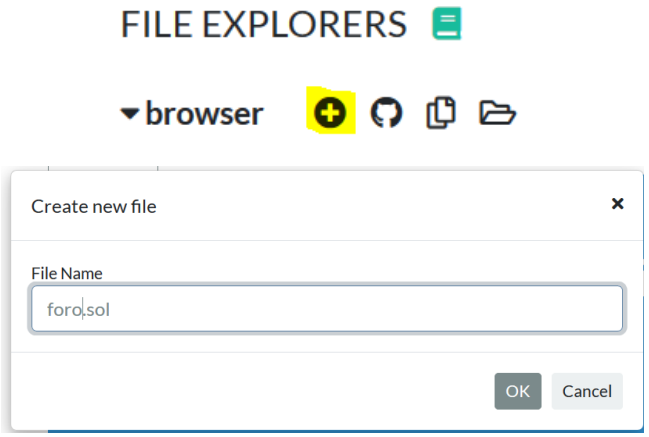


2.6 Recargar Billetera en Faucet: Para reclamar saldo en la red de prueba y pagar las comisiones necesarias en las funciones de registro podemos utilizar la herramienta de faucet enviando un tweet con la siguiente estructura y la dirección donde queremos solicitar el saldo. El url del tweet se ingresa en el portal Faucet de Rinkeby y luego el faucet libera el saldo a la dirección solicitada.

3. Copiar y compilar contrato en Remix
   3.1 Ingresar a remix y seleccionar el lenguaje solidity:



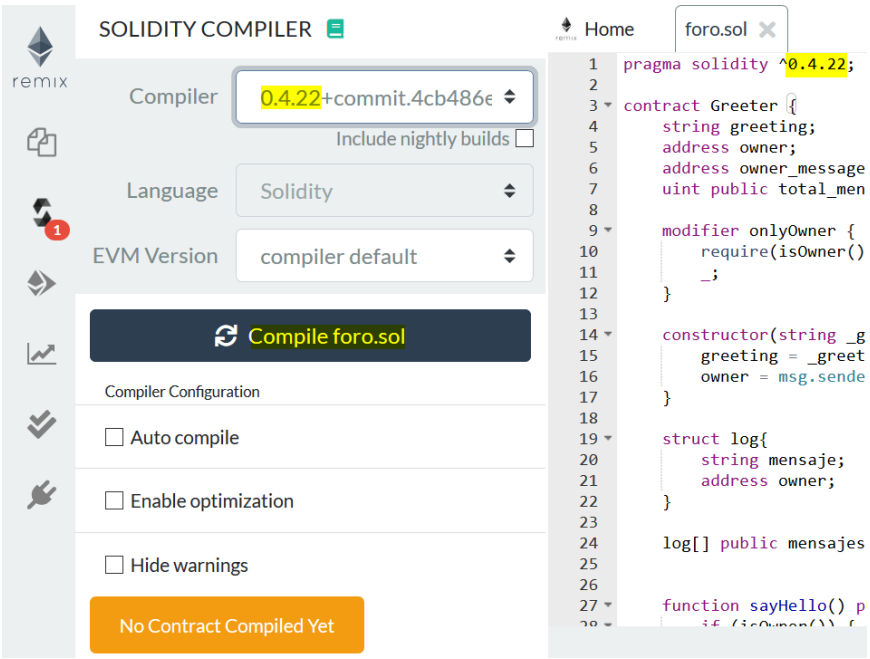
3.2 Ingrese al editor de archivos seleccionando la primera opción del menú lateral derecho (símbolo de dos hojas). En el explorador de archivos crear un nuevo archivo:



Pegar dentro del nuevo archivo el código disponible en el siguiente enlace:

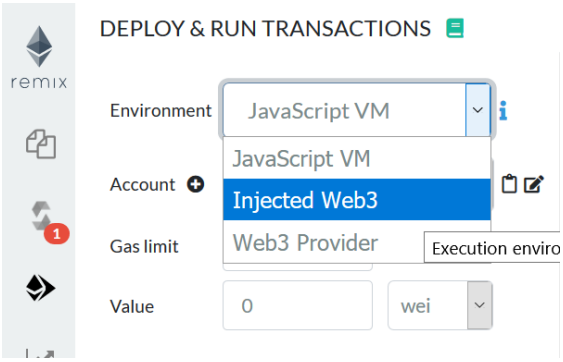
https://drive.google.com/file/d/1bSrnb18Kh4dl3P4GbppuC-izEkqlWCri/view

3.3 Compilar el código: Este paso permite a la herramienta de pruebas reconocer las funciones disponibles en el contrato para interactuar con ellas y revisar errores en el código. Ingrese a la segunda opción del menú lateral derecho (símbolo de solidity). Con el archivo abierto seleccione la opción Compilar, si copio correctamente el código debe compilar sin problemas (verifique que la versión del compilador concuerde con la versión del código, en este caso 0.4.22)

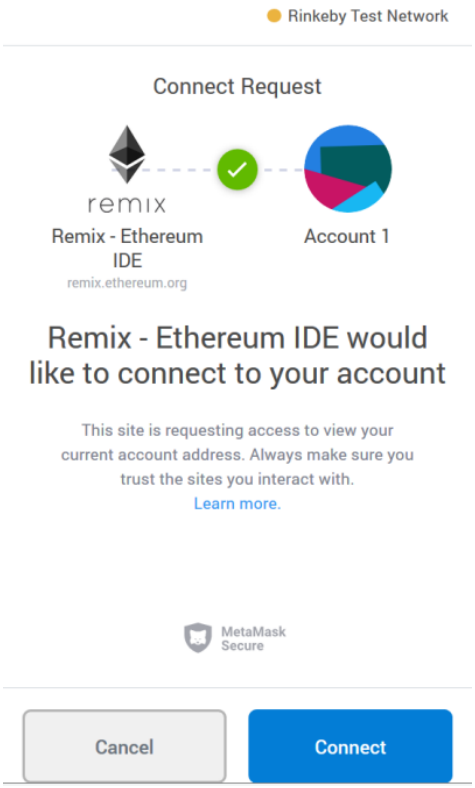


4. Conectar Remix y Metamask: Luego de haber realizado la compilación ingrese a la tercera opción del menú lateral derecho (símbolo de Ethereum). El contrato inteligente compilado ya se encuentra desplegado en la Blockchain de Rinkeby, para conectarse a él siga los siguientes pasos.

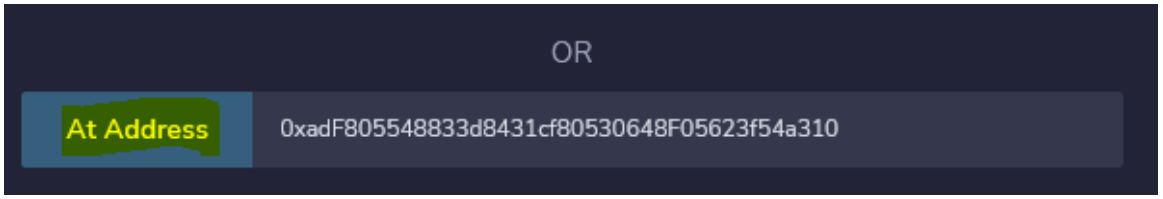
   4.1 Conectarse al ambiente Rinkeby: En la opción “Enviroment” y con el Metamask activado en su explorador, seleccione la opción “Injected Web3”.



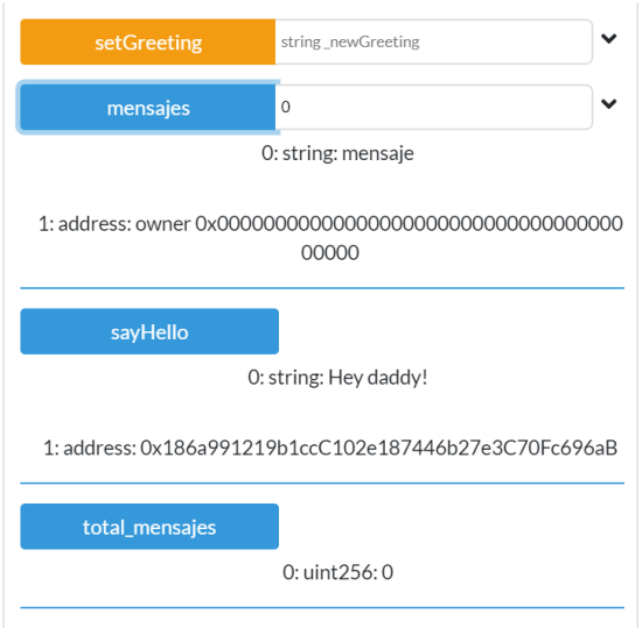
4.2 Autorizar Wallet: Desde Metamask es necesario autorizar a la wallet la conexión con remix. En el complemento de Metamask aparecerá un ventana emergente o notificación para dar autorización a la solicitud de conexión:



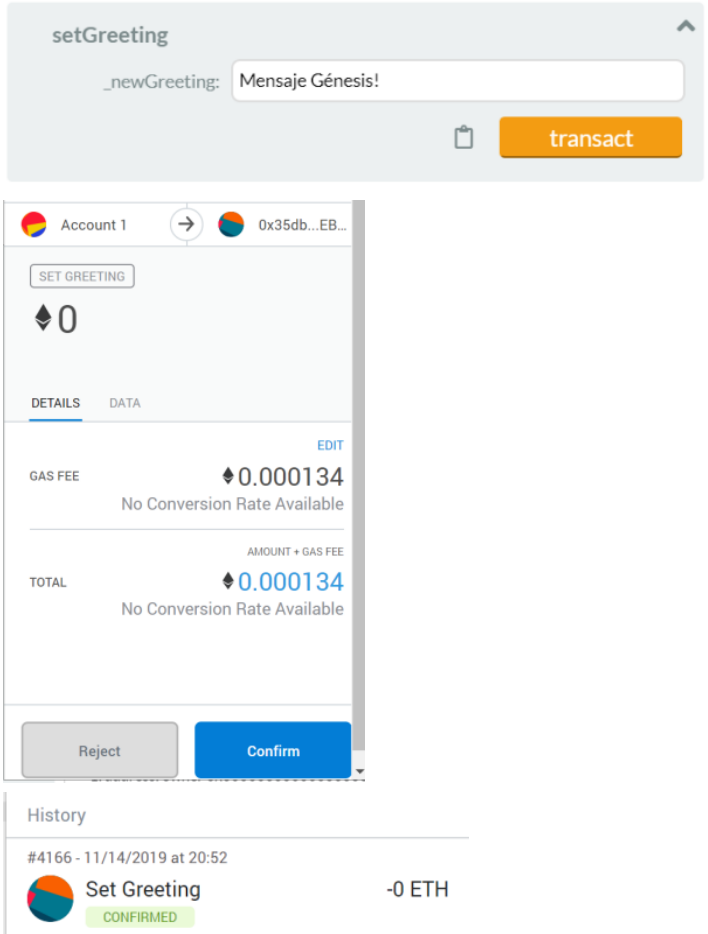
5. Conectarse al contrato
   5.1 Una vez compilado el contrato y conectados con la red de prueba utilizar la dirección pública del contrato y presionar el botón azul que dice “at address” para comunicarse con el contrato y obtener una interfaz de prueba con las funciones de registro y consulta.
   Dirección: 0xadF805548833d8431cf80530648F05623f54a310



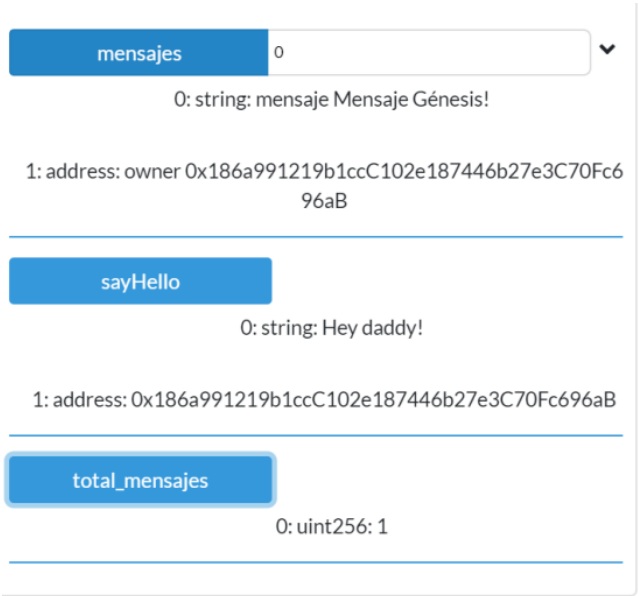
   5.2 Funciones:



5.3: Registrar:



5.4 Consultar información:



Actividad 3: Selección de Arquitectura

Utilizando la información disponible en la siguiente tabla:

Arquitectura	Ventajas	Desventajas
Full Blockchain	• La seguridad es completamente descentralizada. • Genera mayor confianza a usuarios experimentados en criptomonedas.	• Representa mayores barreras de adopción para usuarios no experimentados con criptomonedas. • Reduce la cantidad de funcionalidades disponibles para la experiencia del usuario
Blockchain + DB	• La aplicación permite generar una experiencia de usuario más completa. • Es mucho más fácil de utilizar y adoptar por los usuarios no experimentados con criptomonedas.	• La seguridad de las claves privadas puede tender a centralizarse • Genera desconfianza en los usuarios experimentados con criptomonedas.

Responder a los siguientes criterios para seleccionar su arquitectura:

¿Su público objetivo son usuarios experimentados en criptomonedas y Blockchain?
Si___ No__
¿Requiere almacenar archivos o algún volumen alto de datos?
Si___ No__

Respuesta	Recomendación
Si , Si	**Blockchain + DB** **NOTA:** Utilizar billetera de un tercero como Metamask para transacciones evitando guardar en la DB las claves privadas de los usuarios (mayor confianza a usuarios cripto).
Si , No	**Blockchain Full**
No , Si	**Blockchain + DB**
No , No	**Blockchain Full** **NOTA:** ● Utilizar algún mecanismo para gestión de claves privadas como archivos que custodian los usuarios. ● Gestionar los saldos de la moneda nativa para evitar a los usuarios la gestión de consumo de gas

BALANCE DEL DÍA

Al final de la sesión los participantes habrán aprendido a identificar los elementos y variables a tener en cuenta antes de formular un contrato inteligente.
También aprenderán a diferenciar y saber seleccionar las arquitecturas más adaptadas a las aplicaciones descentralizadas.