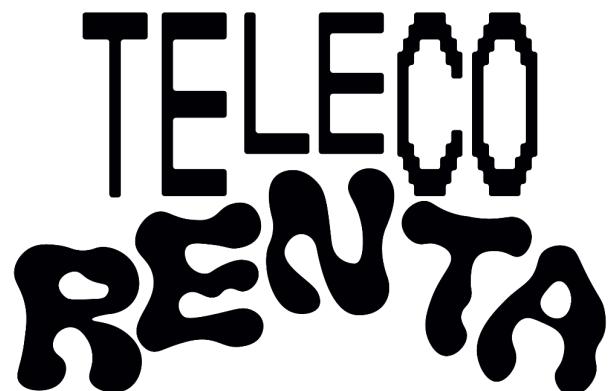


# TALLER DE BLOCKCHAIN

---



PLAN DE  
PROMOCIÓN DE LOS ESTUDIOS  
DE TELECOMUNICACIÓN



Financiado por  
la Unión Europea  
NextGenerationEU



## **Introducción a la Actividad**

---

¡Bienvenidos al mundo de la tecnología blockchain! En esta actividad, exploraremos dos aspectos clave: las funciones hash y los contratos inteligentes en Ethereum. Desde los fundamentos hasta la práctica, descubriremos cómo estas herramientas están transformando la manera en que manejamos datos y transacciones digitales. ¡Comencemos!

## ACTIVIDAD 1. Funciones de hash

---

¡Explora el fascinante mundo de la blockchain! En esta emocionante travesía, nos sumergimos en el corazón de la seguridad y la integridad de la cadena de bloques.

Blockchain no es más que una lista digital de registros de datos en constante crecimiento. Esta lista se compone de muchos bloques que no son más que conjuntos de transacciones que están ordenados en orden cronológico, vinculados entre sí y protegidos por pruebas criptográficas. Las transacciones de la blockchain ocurren dentro de una red P2P de computadoras, proporcionando un medio seguro. Cada bloque contiene tres cosas: *la información, su hash y el hash del bloque anterior*.

Pero, ¿Qué es un hash? Un hash es un número, que ha sido generado por una función de hash, único e irrepetible de identificación. Al tener el hash del bloque anterior queda conectado a su sucesor, creando la famosa blockchain. Es importante remarcar que si se cambia la información de un bloque cambia su hash, por tanto, dejará de encajar y la cadena quedará invalidada.

Para entender este concepto mejor, imagínate que estás enviando un mensaje secreto a través de correo postal y decides proteger el contenido del sobre utilizando un *sello especial*. Cada vez que envías un mensaje, colocas un sello especial en el sobre. Este sello contiene información única y condensada de todo el contenido del mensaje. Si alguien intenta abrir el sobre o modificar su contenido, el sello se rompe o cambia drásticamente.

En blockchain, cada bloque de transacciones es como el sobre que contiene información importante. La función de hash es como el sello especial que garantiza la integridad del contenido. Si alguien intenta manipular las transacciones dentro del bloque, el hash cambiará, indicando que algo no está bien.

Por otra parte tenemos un concepto muy interesante que es el de **Proof of Work** (PoW o Prueba de Trabajo) el cual es el protocolo de consenso más conocido y antiguo que consiste en que las partes de una red realicen con éxito un trabajo computacionalmente costoso para acceder a los recursos de dicha red. Este mecanismo se utiliza para evitar la repetición de transacciones y asegurar la integridad de la cadena de bloques. Si alguien quiere generar una transacción falsa es necesario que éste tenga una infraestructura computacionalmente superior a la existente en la propia red, cosa que es bastante improbable.

¿Conocéis la identidad del creador de Bitcoin ? Su nombre es Satoshi Nakamoto que dejó de trabajar en el proyecto Bitcoin en el 2011 y posteriormente desapareció de la vida pública. Su identidad es un enigma

En un sistema PoW, los participantes de la red son conocidos como “**mineros**”, los cuales básicamente compiten entre sí para resolver problemas matemáticos complejos. Una vez resueltos, el bloque de transacciones se agrega a la cadena de bloques. Cabe destacar que estos problemas matemáticos son difíciles de resolver, pero la solución es fácil de verificar. Este proceso de resolución de problemas se conoce como “**minería**”.

Un ejemplo de minería puede ser el caso de **Bitcoin**, en el cual los mineros intentan encontrar un valor (llamado nonce) que cuando se agrega al encabezado del bloque y se pasa a través de una función de hash, produzca un resultado que cumpla con ciertos requisitos (por ejemplo, comenzar con un número específico de ceros). Esta solución es verificada por los demás nodos de la red. El nodo minero que gana ese juego recibe una recompensa en bitcoins. Probablemente el concepto de Blockchain ha ido obteniendo más popularidad en los últimos años debido al “boom” de las criptomonedas. Una criptomoneda es una forma de dinero digital que utiliza criptografía para garantizar la seguridad de las transacciones y operan dentro de la red blockchain. A diferencia de las monedas tradicionales controladas por gobiernos o entidades financieras, las criptomonedas operan en una red descentralizada de nodos. Esto significa que no hay una entidad central que controle o regule la moneda. Además, algunas criptomonedas como Ethereum, permiten la creación de contratos inteligentes y tokens. Los tokens son activos digitales que representan un valor específico y pueden utilizarse para diversas aplicaciones dentro de la plataforma blockchain.

A continuación veremos dos sitios web que simulan el comportamiento de blockchain para entender mejor los conocimientos previamente explicados.

Para visualizar de manera efectiva cómo funcionan las funciones de hash, puedes visitar <https://blockchaindemo.io/>, donde puedes crear fácilmente un nuevo bloque y agregarlo a la red de blockchain.

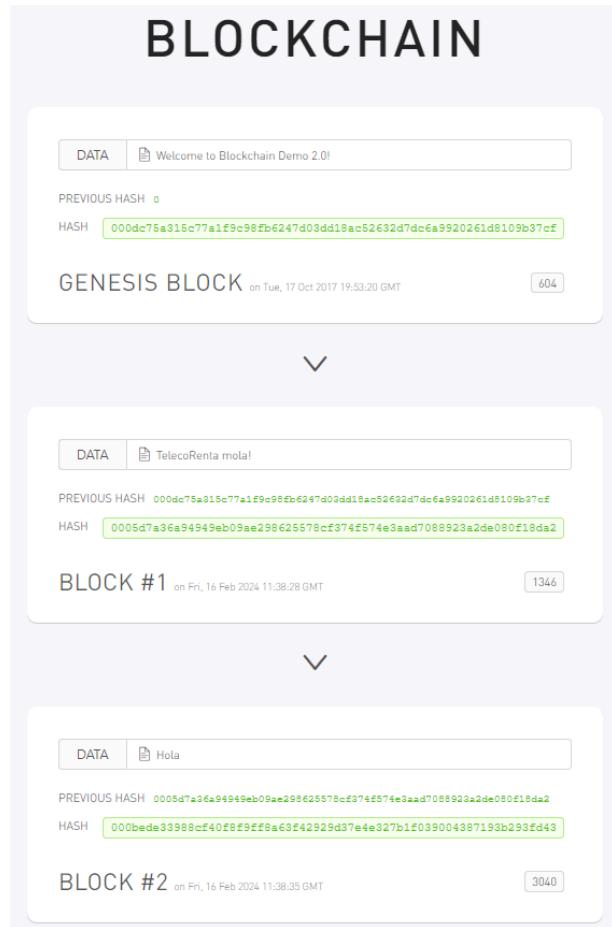
Una vez dentro, encontrarás un menú desplegable donde podrás ingresar un mensaje para el siguiente bloque. Por defecto, verás que el primer bloque fue creado en 2017, así que construiremos nuestra cadena de bloques a partir de este.

En la siguiente imagen se pueden ver los distintos campos que se necesitan para formar un bloque:

- **DATA**: Información que se quedará grabada en el bloque.
- **PREVIOUS HASH**: Hash del bloque anterior.
- **HASH**: Hash del propio bloque.
- **NONCE**: Valor que se utiliza para generar el hash.
- **TIMESTAMP**: Fecha a la que se ha creado el bloque.

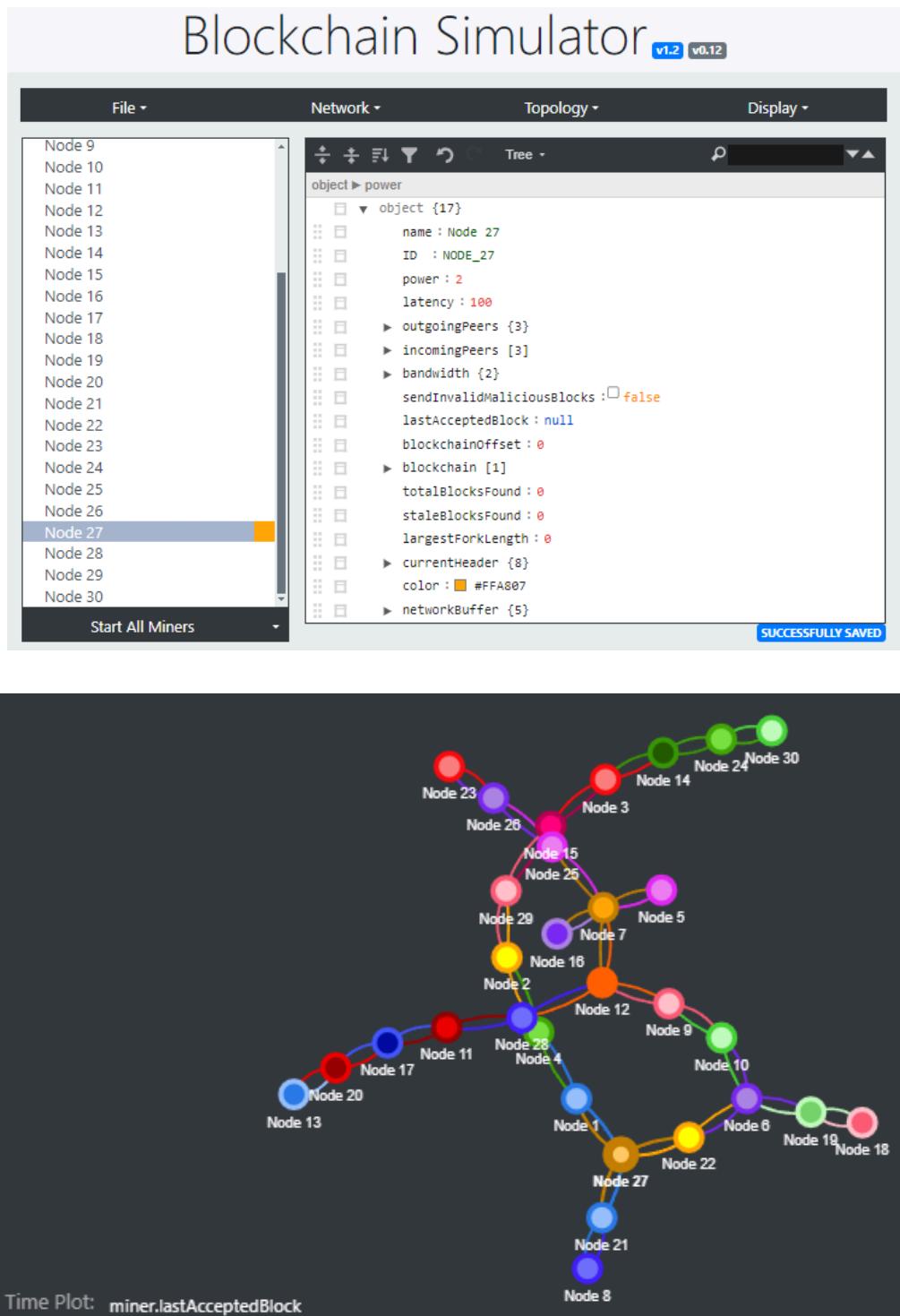
The screenshot shows a blockchain simulation interface. At the top, there is a button labeled "DATA" and a link "nuevo bloque". Below it, the first block is displayed with the following details:  
PREVIOUS HASH: 00058e57504de02e6a0952e8794eb7483437c2c671fc74715b0b7c4f469a507  
HASH: 00072639d63eeb10c8a1f70291f94e72291b90eb86c52678813cb71d95d18e7b  
BLOCK #2 on Fri, 16 Feb 2024 10:11:01 GMT  
A small number "1725" is shown in a box to the right of the timestamp.  
  
Below this, the second block is shown with the following details:  
DATA  
HASH  
+ ADD NEW BLOCK

Después de crear el segundo bloque, notarás que junto al mensaje se ha generado un segundo hash basado en la información introducida. A partir de este punto, cada bloque contendrá su información, el hash del bloque anterior y su propio hash, asegurando la continuidad de la cadena.



En este otro sitio web, <https://simewu.com/blockchain-simulator/>, encontramos un simulador de blockchain que nos permite observar su comportamiento y funcionamiento.

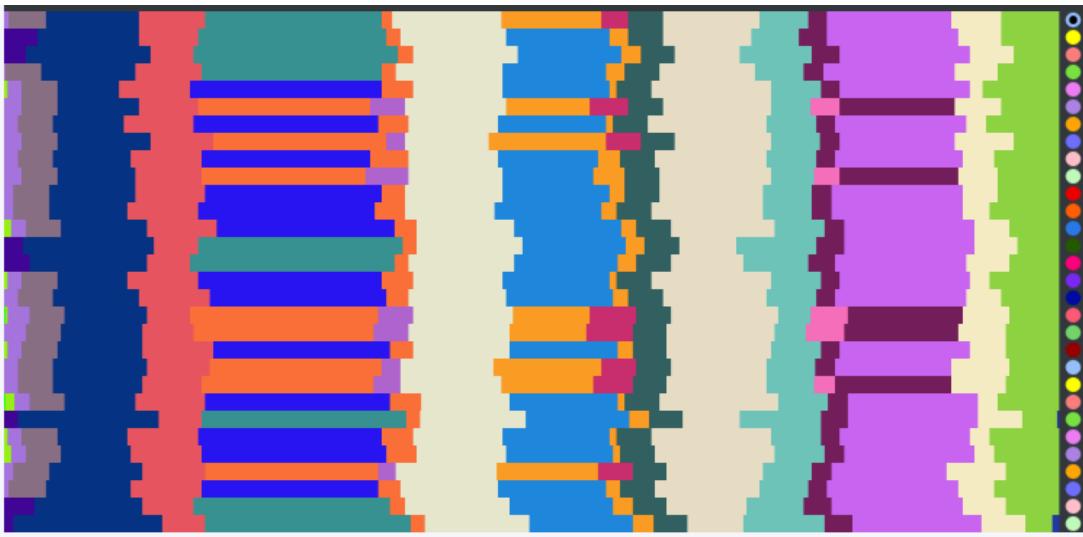
En esta plataforma, podemos modificar algunos valores de los bloques para experimentar con ellos. Sin embargo, lo más interesante ocurre al presionar el botón 'Start All Miners', momento en el cual podemos observar cómo se comportan los bloques una vez están activos.



Una vez que los bloques están activos, se genera el siguiente gráfico que muestra, a través de colores, cómo los nodos mineros verifican y validan las transacciones. Este proceso es esencial para la seguridad y la integridad de la cadena de bloques.

En la siguiente imagen, podemos notar casos donde una transacción, representada por el color rosa en el centro de la pantalla, ha sido validada solo por 8 bloques, lo que podría indicar una posible falta de consenso y, por lo tanto, se considera inválida. Por otro lado, observamos

transacciones representadas por el color beige, que han sido validadas por todos los bloques, lo que garantiza su autenticidad y validez en la cadena.



La actividad te permite comprender los conceptos clave de blockchain, como la estructura de bloques encadenados y la Prueba de Trabajo (PoW), utilizando dos sitios web:

- En <https://blockchaindemo.io/>, puedes visualizar de manera efectiva cómo funcionan las funciones de hash en la blockchain. Te permitirá crear fácilmente un nuevo bloque y agregarlo a la red de blockchain. Puedes seguir la demostración detallada que ofrece la página web.
- Por otro lado, en <https://simewu.com/blockchain-simulator/>, puedes experimentar con un simulador de blockchain que te permitirá observar el comportamiento y funcionamiento de la cadena de bloques. Puedes modificar algunos valores de los bloques para experimentar con ellos y, al presionar el botón 'Start All Miners', podrás observar cómo se comportan los bloques una vez están activos. Además, podrás ver cómo los nodos mineros verifican y validan las transacciones, lo que es esencial para la seguridad y la integridad de la cadena de bloques.

## ACTIVIDAD 2. Hello World Smart Contract

---

¡Bienvenidos a una emocionante aventura en el mundo de los contratos inteligentes en Ethereum! En esta actividad, vas a dar tus primeros pasos en el desarrollo de blockchain. Aprenderás cómo crear y desplegar tu primer contrato inteligente: un sencillo "Hello World" en la red Ethereum. Esta es tu oportunidad para sumergirte en el fascinante universo de la programación en blockchain y ver cómo tus ideas cobran vida en este innovador espacio digital.

¿Qué vamos a hacer?

- Desarrollar un Contrato Inteligente: Utilizarás Solidity, el lenguaje de programación de Ethereum, para escribir tu propio contrato inteligente.
- Interactuar con la Red Ethereum: Aprenderás a utilizar Alchemy, una poderosa herramienta que simplifica la forma de conectar y operar con la red Ethereum.
- Experimentar el Proceso de Despliegue: Compilarás y desplegarás tu contrato inteligente en una red de prueba de Ethereum, dándote una visión práctica de cómo funcionan estas tecnologías en el mundo real.

### Primera parte: Preparación del entorno

En esta primera parte nos centraremos en la creación de cuentas en diferentes páginas con el objetivo de crear un contrato inteligente y ver su propósito en las redes blockchain.

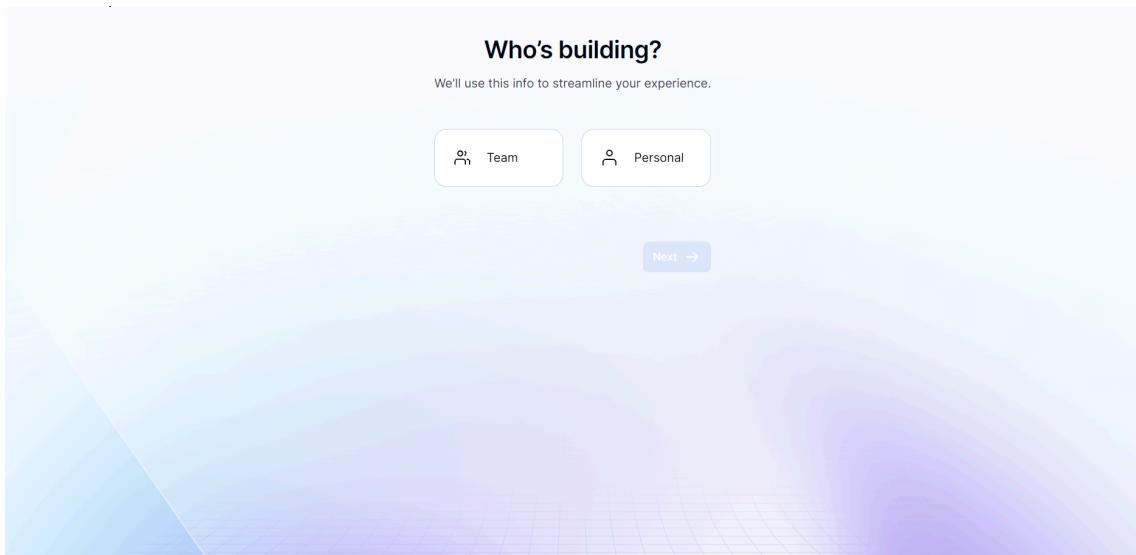
#### Paso 1: Conectarse a la red ethereum

Hay muchas formas de realizar solicitudes a la cadena de Ethereum. Para simplificar, utilizaremos una cuenta gratuita en Alchemy, una plataforma para desarrolladores de blockchain y API que nos permite comunicarnos con la cadena de Ethereum sin tener que ejecutar nuestros propios nodos. La plataforma también cuenta con herramientas de desarrollo para monitoreo y análisis que aprovecharemos en este tutorial para entender lo que está sucediendo en el despliegue de nuestro contrato inteligente. Si aún no tienes una cuenta en Alchemy, puedes registrarte gratuitamente aquí:

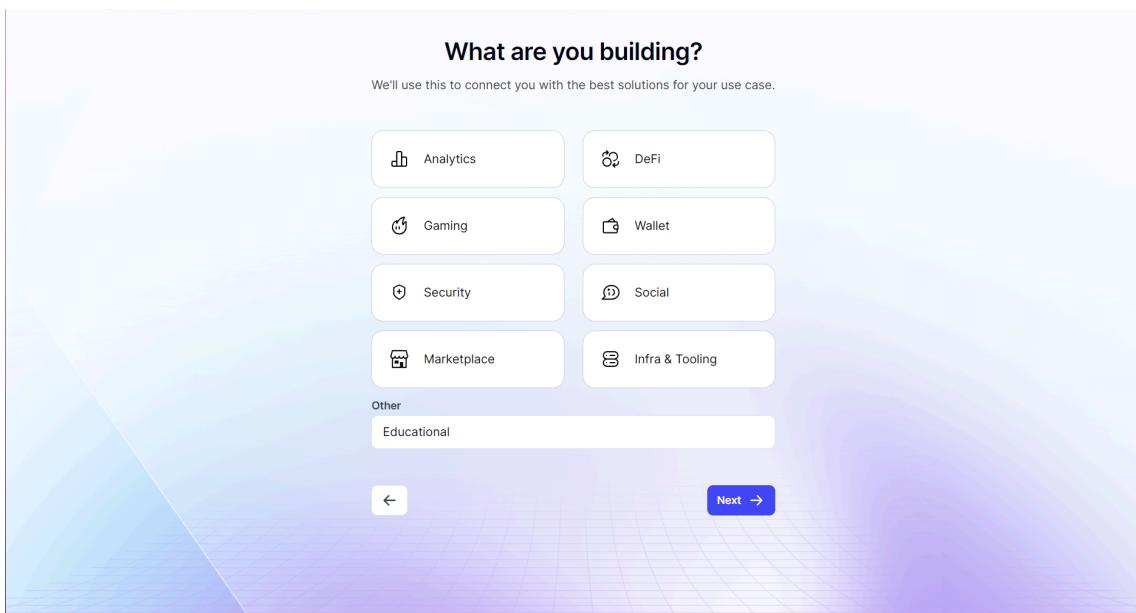
<https://auth.alchemy.com/?redirectTo=https%3A%2F%2Fdashboard.alchemy.com%2Fsignup>.

Sigue los pasos que verás a continuación para crear la cuenta correctamente:

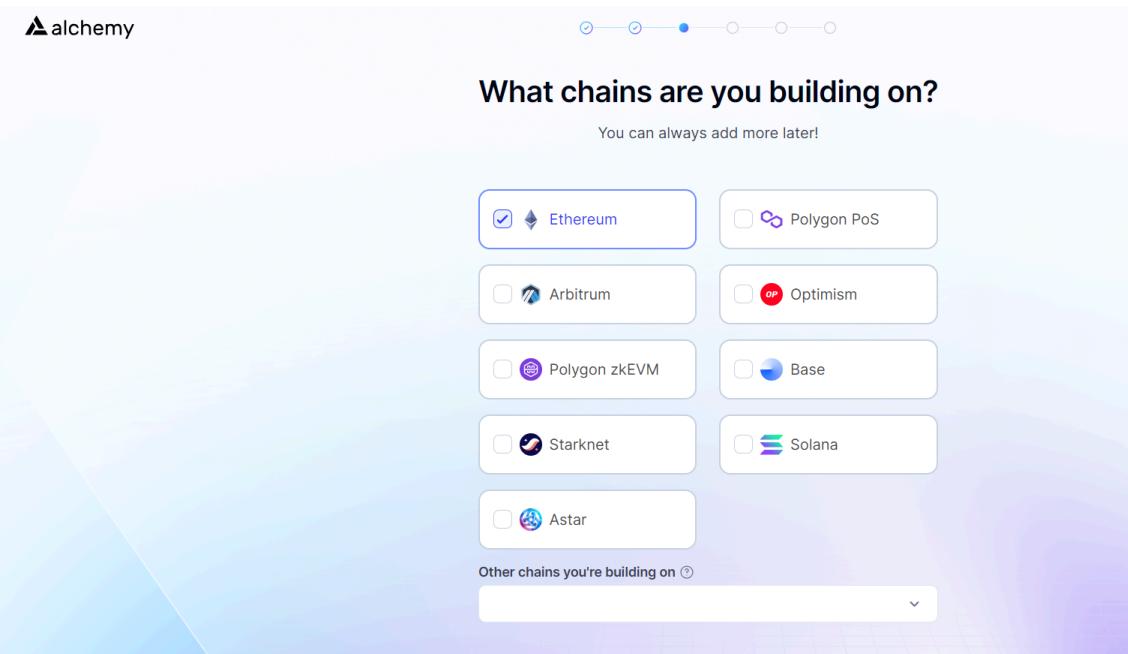
1. Crea una cuenta con google o introduciendo tus credenciales y en la pantalla de "Who's building?" selecciona "Personal".



2. En la pantalla de “What are you building?” escribe “Educational” en “Other”.



3. En la pantalla de “What chains are you building on?” selecciona “Ethereum”.



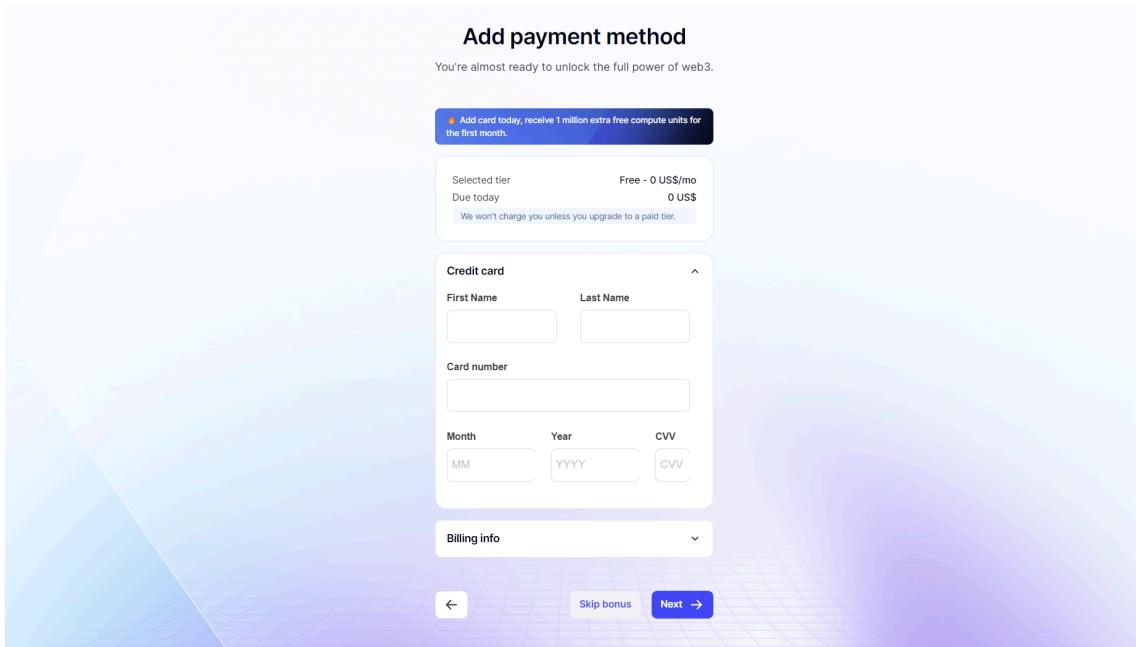
4. Selecciona el plan gratuito y en la siguiente pestaña sin introducir datos selecciona “Skip bonus”.

Select your plan

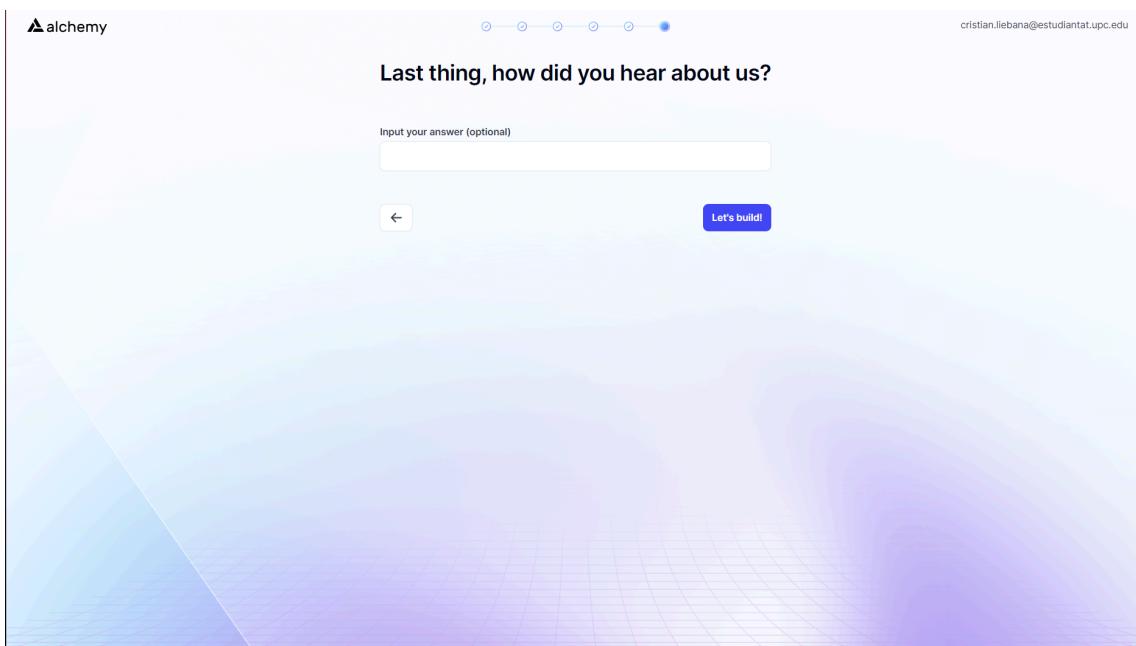
Not sure which plan is best for you? [Compare plans](#)

Free	Growth	Scale	Enterprise	Custom
0 US\$ /mo	49 US\$ /mo	199 US\$ /mo		
The largest & most powerful free tier in web3	For teams that need more capacity to grow	Enterprise discounts, no salesperson required	Volume pricing, unmatched support, custom SLAs	
300 million CU per month 5 apps	400 million CU per month 15 apps	1.5 billion CU per month 30 apps	Custom CU Unlimited	
Powering teams like 	Powering teams like 	Powering teams like 	Powering teams like 	

←      Next →



5. Finalmente oprime el botón de “Let’s build” y tu cuenta se habrá registrado correctamente.



### Paso 2: Crea tu aplicación (y clave API)

Una vez que hayas creado una cuenta en Alchemy, puedes generar una clave API creando una aplicación. Esto nos permitirá hacer solicitudes a la red de pruebas.

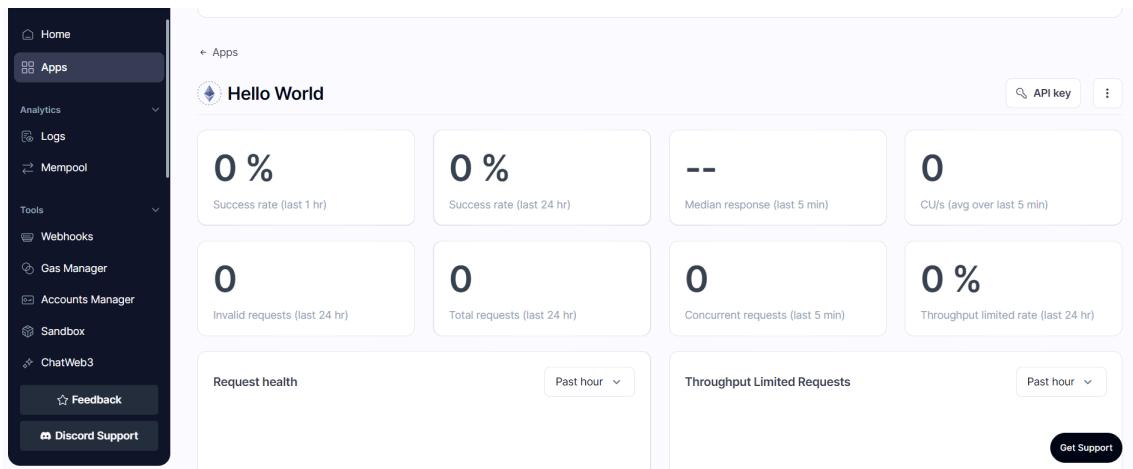
1. Navega a la Página "Create Application": En tu panel de control de Alchemy, ves a la pestaña de "Apps" en la barra de navegación lateral izquierda. Verás que ya tienes creada una app, pero vamos a crear una nueva con la red "Ethereum Sepolia", para ello haz clic en "Create new app".

The screenshot shows the Alum's Ethereum App dashboard. On the left, there is a dark sidebar with various navigation options: Home, Apps (selected), Analytics, Logs, Mempool, Tools, Webhooks, Gas Manager, Accounts Manager, Sandbox, ChatWeb3, Feedback, and Discord Support. At the bottom of the sidebar are two buttons: 'Get Support' and a blue 'Create new app' button. The main area is titled 'Apps' and contains a table with one row. The table columns are: App Name, Network, Requests (24h), Failed requests (24h), Created on, and Actions. The single row shows 'Alum's Ethereum App' under 'App Name', 'Ethereum Mainnet' under 'Network', '0' under 'Requests (24h)', '0' under 'Failed requests (24h)', '18/3/2024' under 'Created on', and 'API Key' and 'App Details' under 'Actions'.

2. Nombra tu Aplicación: Ponle el nombre de "Hello World" a tu aplicación, ofrece una breve descripción, selecciona "Ethereum" para el Chain y elige "Ethereum Sepolia" como tu red.

The screenshot shows the 'Create new app' modal window. It has fields for 'Chain' (set to Ethereum) and 'Network' (set to Ethereum Sepolia). The 'Name' field contains 'Hello World'. The 'Description' field contains the text 'Activity 2 TelecoRenta'. At the bottom are 'Cancel' and 'Create app' buttons.

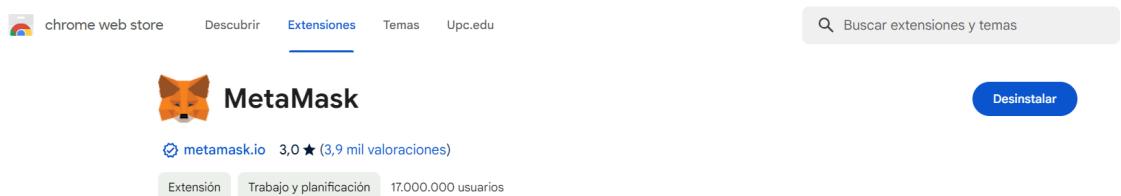
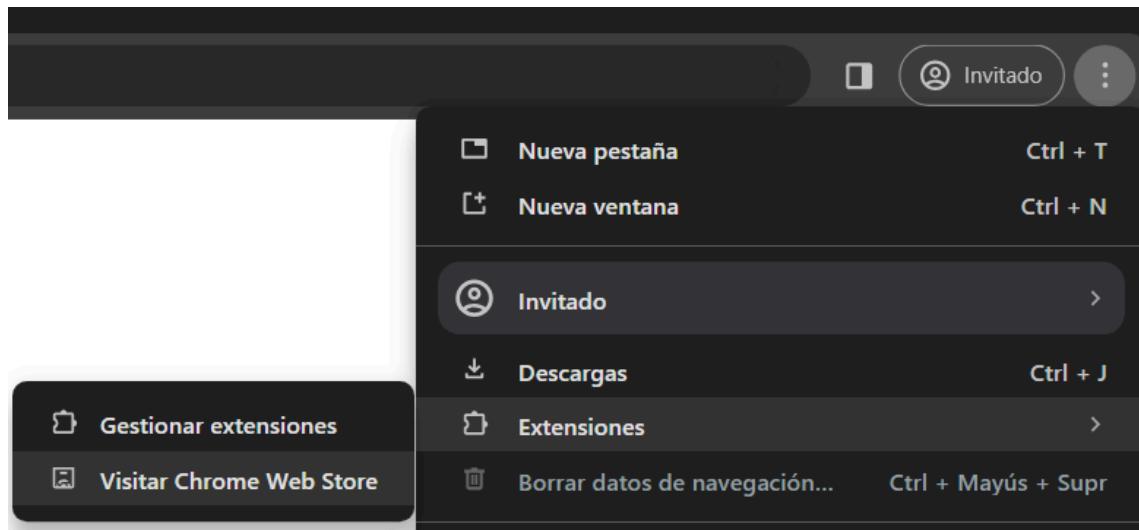
3. Haz clic en "Create app": ¡Y eso es todo! Tu aplicación debería aparecer en la tabla a continuación.



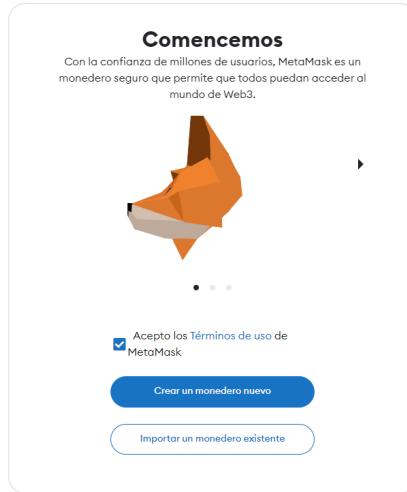
## Paso 2: Configura tu cuenta de Ethereum

Para enviar y recibir transacciones en Ethereum, necesitarás una cuenta de Ethereum. En este tutorial, utilizaremos Metamask, una cartera virtual en tu navegador que ayuda a gestionar tu dirección de cuenta de Ethereum.

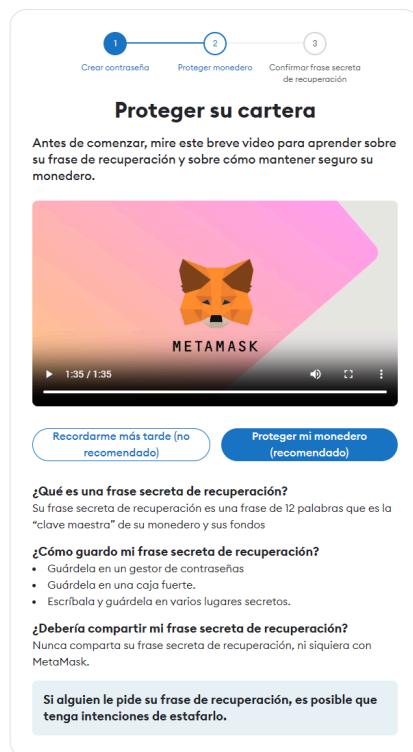
Para empezar, instalaremos MetaMask (situado en las extensiones del navegador Chrome). Para instalar una extensión en google chrome navega a la pestaña “Visitar Chrome Web Store” en el menú de opciones del navegador. Una vez dentro buscaremos MetaMask y la añadiremos a nuestras extensiones.



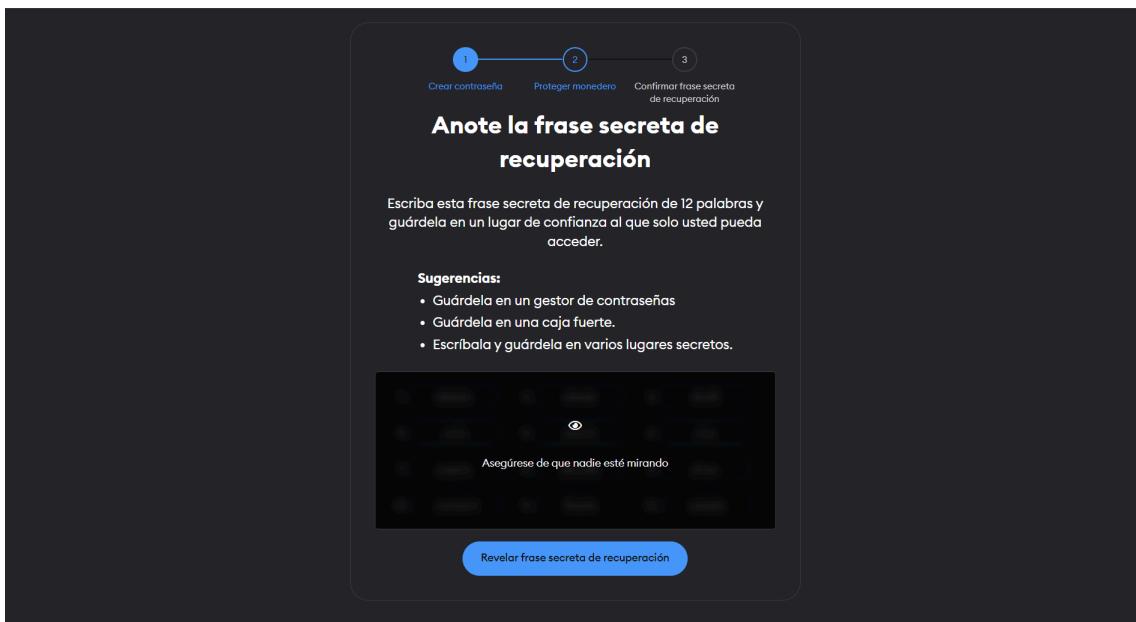
Una vez instalada, automáticamente se abrirá una cuenta, aquí le daremos a “Crear un monedero nuevo”:



Crea una contraseña segura y en el siguiente paso mira el video y selecciona “Protege mi monedero (recomendado)”:

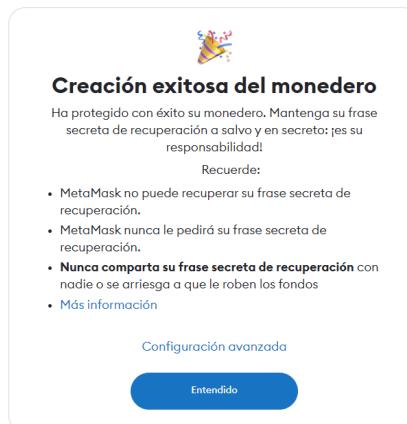


En la siguiente pestaña obtendrás tu frase secreta de recuperación. Selecciona el botón de “Revelar frase secreta de recuperación” y apúntate las 12 palabras junto a su número. Este paso es crucial porque en la siguiente pantalla se requerirá de llenar algunas palabras para confirmar la frase secreta de recuperación:



 METAMASK

Español ▾



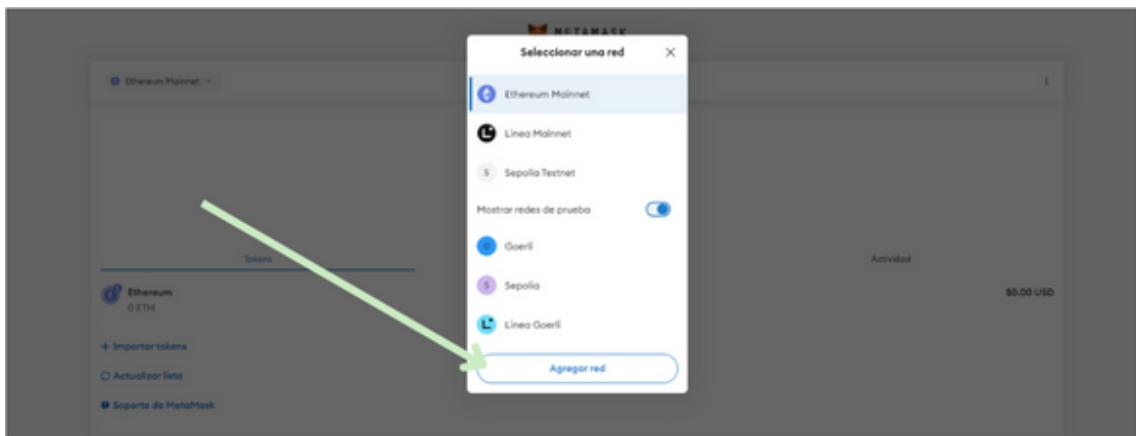
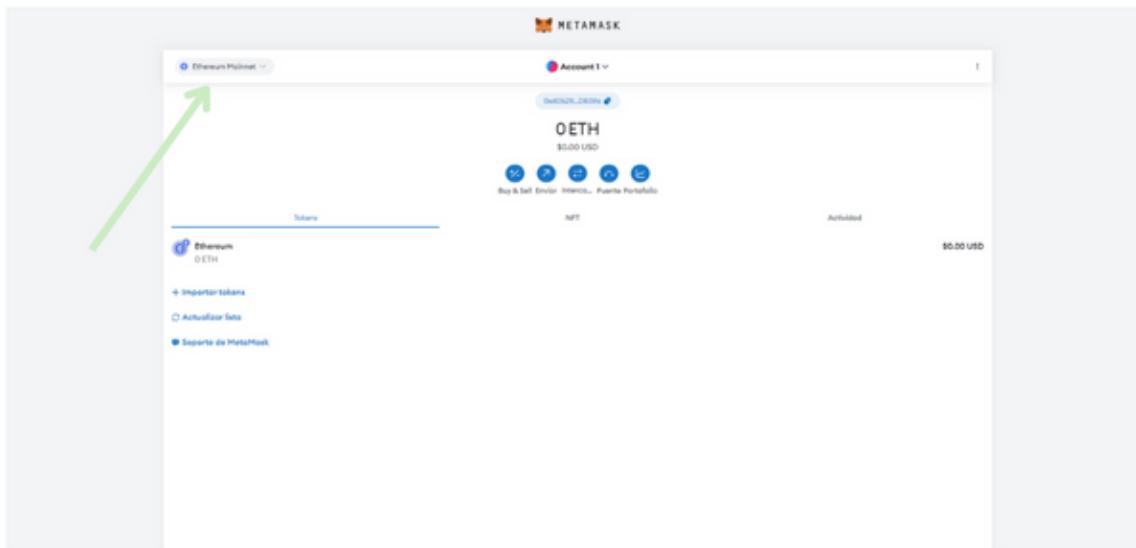
Síganos en Twitter 

Una vez que tengas una cuenta, necesitarás cambiar a la red de Ethereum Sepolia.

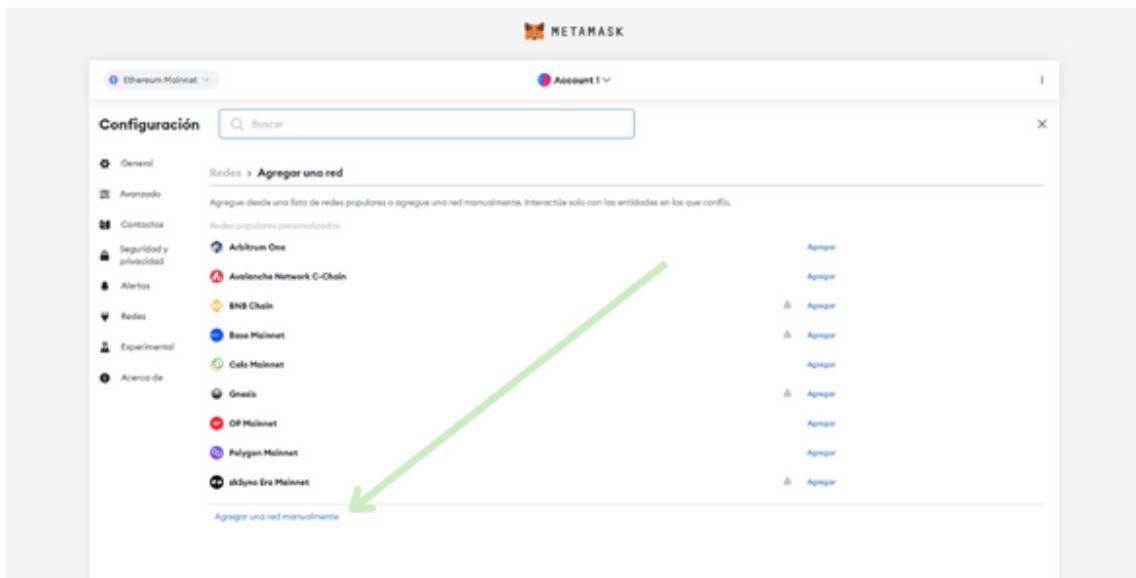
¿Sabías qué? La red Ethereum Sepolia es una red de pruebas para la blockchain principal de Ethereum. Las redes de prueba, son como versiones de práctica de una blockchain real. Los desarrolladores las utilizan para probar nuevas funciones y contratos inteligentes antes de implementarlas en la red principal de Ethereum.

Sigue los pasos a continuación para cambiar:

1. Configura la red: En la esquina superior izquierda, haz clic en el menú desplegable de la red y selecciona "Agregar red".



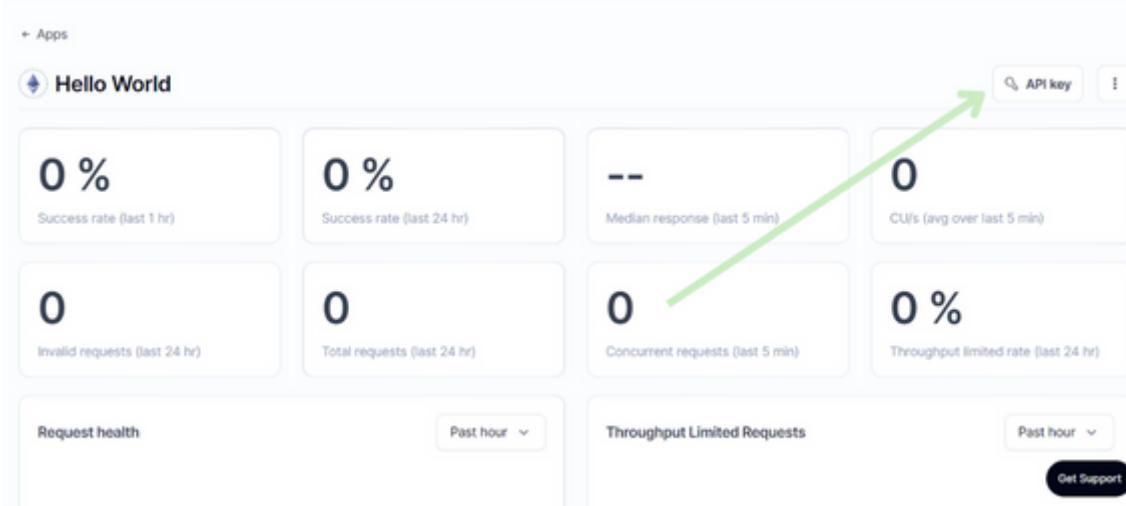
2. Seguidamente, en el menú de “Configuración” le daremos a “Agregar una red manualmente”.



3. Ingresa los Detalles de Sepolia: En los campos en blanco, ingresa los detalles mencionados en la sección de Testnet de Sepolia:

- Nombre de la red:** Sepolia Testnet
- Nueva dirección URL de RPC:**  
<https://eth-sepolia.g.alchemy.com/v2/your-alchemy-api-key>

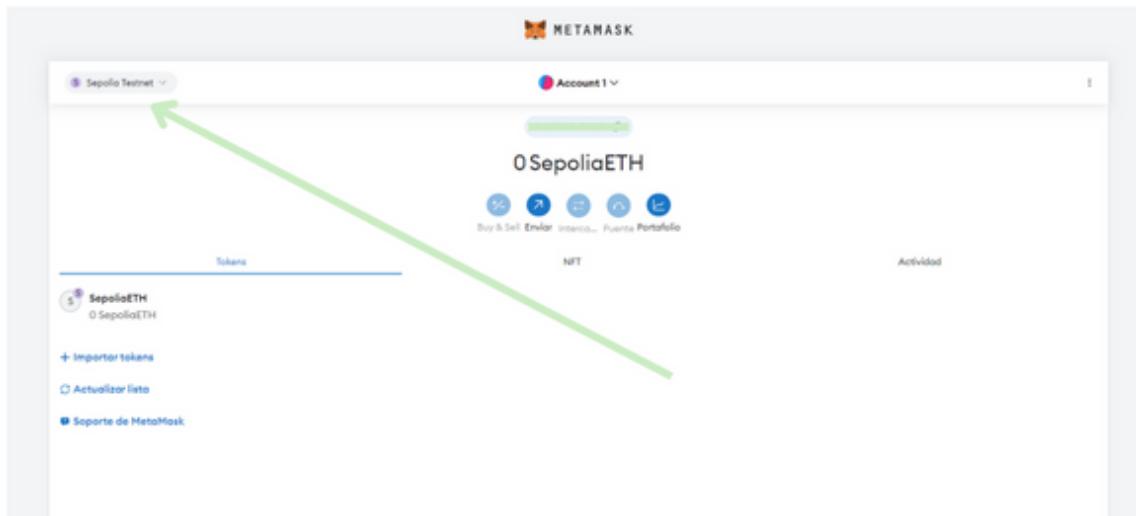
Dónde *your-alchemy-api-key* es la api key de la aplicación que hemos creado anteriormente en alchemy. Para facilitar el proceso de obtener nuestra dirección URL de RPC podéis volver a la página de Alchemy y dentro de la app creada, copiaremos la dirección HTTPS que está dentro de la API key :



- Identificador de cadena:** 11155111
- Símbolo de moneda:** ETH
- Dirección URL del explorador de bloques:** <https://sepolia.etherscan.io>

The screenshot shows a modal window titled "Redes > Agregar una red > Agregar una red manualmente". The form contains the following fields with placeholder text: "Nombre de la red" (Network name), "Nueva dirección URL de RPC" (New RPC URL), "Identificador de cadena" (Chain ID), "Símbolo de moneda" (Currency symbol), and "Dirección URL del explorador de bloques (Opcional)" (Optional Block explorer URL). A warning message in a yellow box states: "Un proveedor de red malintencionado puede mentir sobre el estado de la cadena de bloques y registrar su actividad de red. Agregue solo redes personalizadas de confianza." (A malicious network provider may谎言 about the state of the blockchain and register its activity. Add only trusted custom networks.) At the bottom are "Cancelar" (Cancel) and "Guardar" (Save) buttons.

Luego, haz clic en "Guardar" y estarás listo.

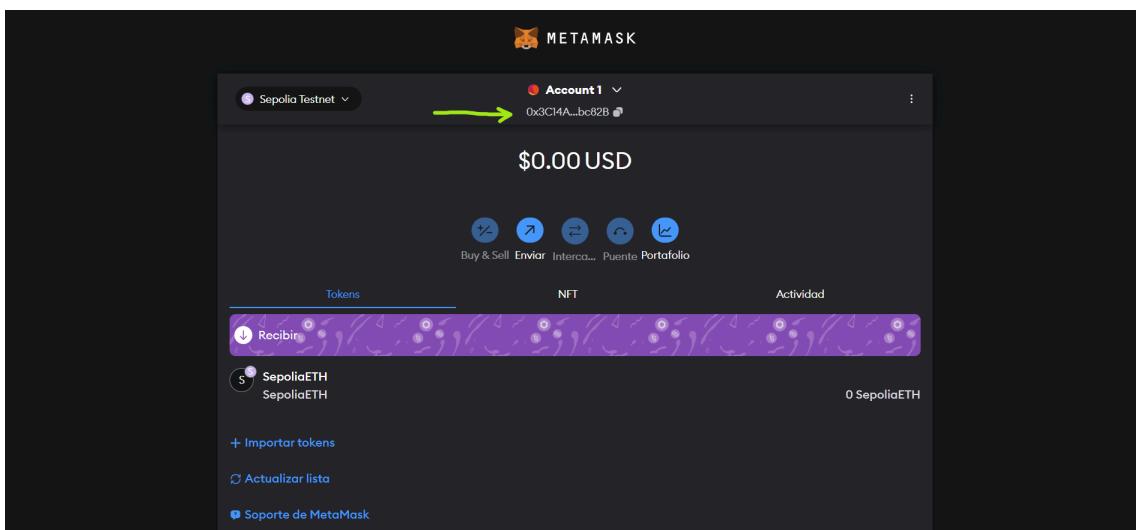


## Segunda parte: Creación de un contrato inteligente

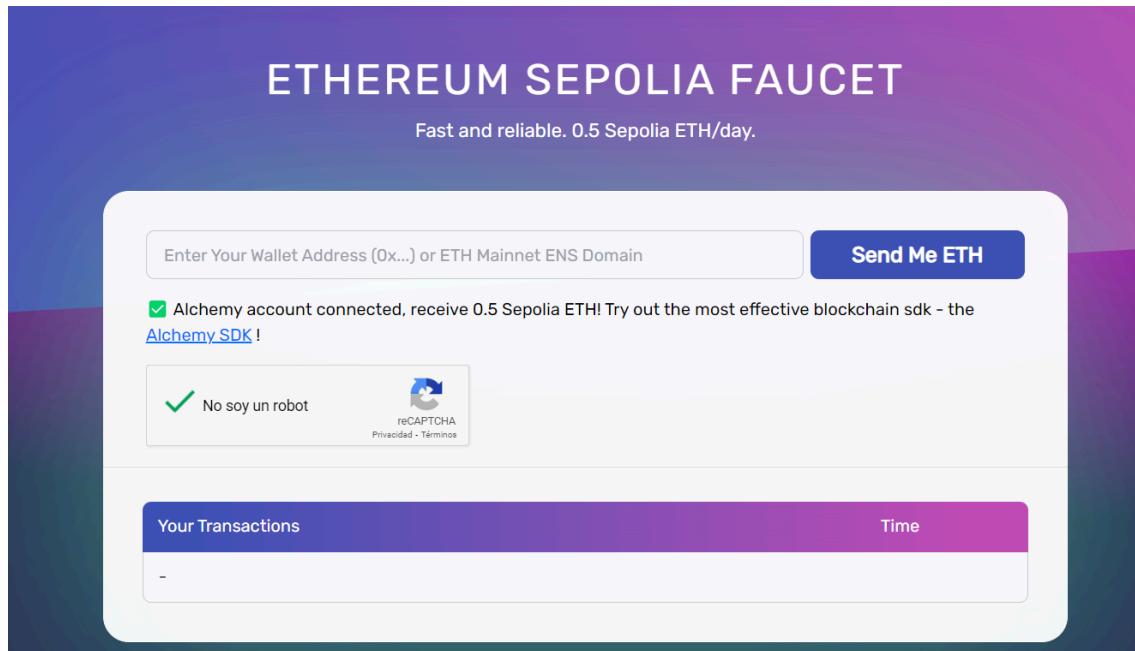
Después de haber realizado toda la configuración inicial, ya estamos preparados para crear un contrato inteligente. Un contrato inteligente es un programa informático que se ejecuta automáticamente cuando se cumplen ciertas condiciones predefinidas en una red blockchain. Estos contratos se utilizan para automatizar y hacer cumplir acuerdos digitales entre dos o más partes sin necesidad de intermediarios.

### Paso 4. Agregando Ether desde Sepolia Faucet:

Para poder realizar este contrato inteligente necesitamos realizar una transacción, pero para poder llevar a cabo esa transacción necesitaremos un mecanismo para poder transferir Ethereum falso a la red de pruebas . Cabe recalcar que este Ethereum falso no tiene ningún valor monetario porque es una moneda virtual que simula al Ethereum real con el único fin de realizar pruebas. Para obtener Ethereum, puedes visitar Sepolia Faucet (<https://sepoliafaucet.com/>) e ingresar la dirección de tu cuenta de Sepolia, que se encuentra en la página principal de MetaMask:



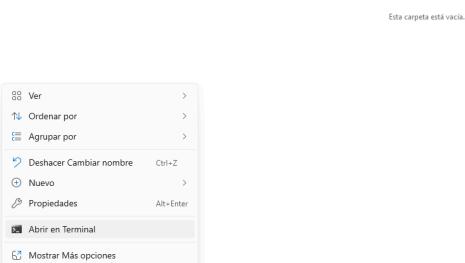
Después en Sepolia Faucet inicia sesión con la cuenta de alchemy ("Please signup or login") y debería quedarte algo así:



Seguidamente introduce la dirección de Sepolia y luego haz clic en "Send Me Eth". Puede tardar un poco (30 minutos) en recibir tu Ethereum falso debido al tráfico de la red. Deberías ver el Eth en tu cuenta de Metamask poco después.

#### Paso 5. Inicializar el proyecto

Primero, crea una carpeta en el escritorio que se llame telecorenta. Seguidamente, haz click derecho dentro de la carpeta y selecciona la opción abrir en terminal.



Después en el terminal escribe los siguientes comandos:

```
mkdir hello-world-smart-contract
```

```
cd hello-world-smart-contract
```

Una vez que estés dentro de la carpeta del proyecto, podemos inicializar el proyecto usando npm init :

```
npm init
```

NPM (Node Package Manager) es un administrador de paquetes para el ecosistema de JavaScript (específicamente para Node.js) el cual permite descargar y utilizar fácilmente bibliotecas de código abierto y frameworks entre otras herramientas.

Seguidamente te pedirá responder una serie de preguntas, las cuales puedes usar las respuestas predeterminadas. Aquí tienes un ejemplo:

```
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (hello-world-smart-contract)
version: (1.0.0)
description: hello world smart contract
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\User\OneDrive\Escritorio\Projects\hello-world-smart-contract\package.json:

{
  "name": "hello-world-smart-contract",
  "version": "1.0.0",
  "description": "hello world smart contract",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes)
```

#### Paso 6. Descarga y configura HardHat

Hardhat es un entorno de desarrollo potente diseñado para el desarrollo de software en Ethereum. Te permite compilar, desplegar, probar y depurar tus contratos inteligentes y Apps localmente antes de desplegarlos en la cadena en vivo. Esta herramienta es especialmente útil para desarrolladores que buscan construir y probar sus proyectos antes de lanzarlos en vivo.

Dentro de nuestro proyecto *hello-world-smart-contract*, ejecuta:

```
npm install --save-dev hardhat
```

```
npx hardhat
```

Después de ejecutar el comando anterior, aparecerá un mensaje de bienvenida, dándote diferentes opciones para elegir. Selecciona la opción "Create an empty hardhat.config.js", esto generará un archivo *hardhat.config.js*. Utilizaremos este archivo que será necesario para los próximos pasos en nuestro proyecto.

```

888 888          888 888          888
888 888          888 888          888
888 888          888 888          888
888888888888 8888b. 888d888 .d88888 88888b. 8888b. 8888888
888 888 "88b 888P" d88" 888 888 "88b      "88b 888
888 888 .d888888 888 888 888 888 .d888888 888
888 888 888 888 888 Y88b 888 888 888 888 888 888 Y88b.
888 888 "Y888888 888      "Y888888 888 888 "Y888888  "Y888

Welcome to Hardhat v2.19.4

? What do you want to do? ...
  Create a JavaScript project
  Create a TypeScript project
  Create a TypeScript project (with Viem)
> Create an empty hardhat.config.js
  Quit

```

### Paso 7. Agregar carpetas al proyecto

Para mantener nuestro proyecto organizado, crearemos dos nuevas carpetas. Navega a la raíz de tu proyecto en la línea de comandos y escribe:

`mkdir contracts`

`mkdir scripts`

- contracts/ es donde guardaremos nuestro archivo de código del contrato inteligente "Hello World".
- scripts/ es donde guardaremos los scripts para desplegar e interactuar con nuestro contrato.

### Paso 8. Escribir nuestro contrato

Quizás te estés preguntando, ¿cuándo vamos a escribir el código del contrato? Bueno, aquí estamos, en el paso 8.

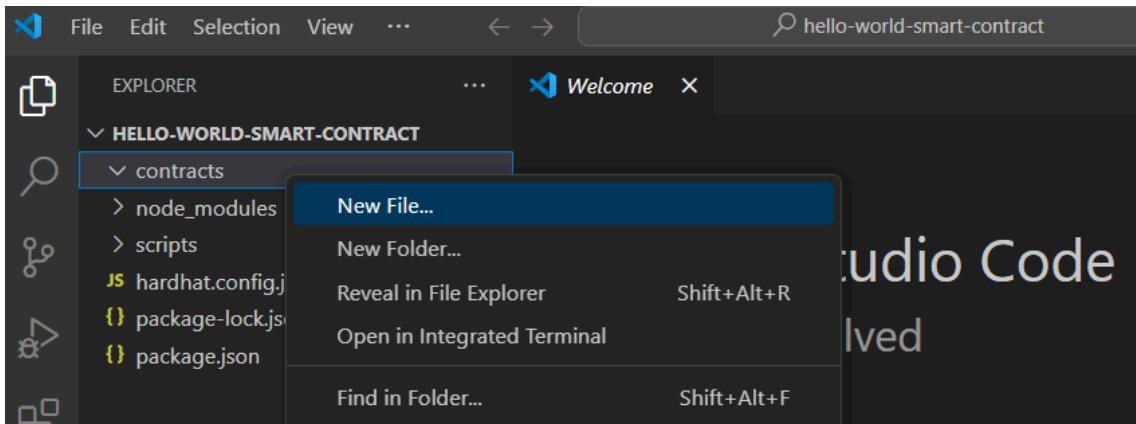
Abre el proyecto *hello-world-smart-contract* en VSCode que previamente te habrá instalado tu instructor (Se puede descargar aquí <https://code.visualstudio.com/download>). Los contratos inteligentes se escriben en un lenguaje llamado Solidity, que es lo que usaremos para escribir nuestro contrato inteligente *HelloWorld.sol*.

Solidity es un lenguaje de programación utilizado para escribir contratos inteligentes en plataformas blockchain, como Ethereum. Está diseñado para ser seguro y permite definir reglas y lógica para la ejecución automatizada de acuerdos digitales.

Para instalar Solidity, escribe el siguiente comando:

`npm install -g solc`

Seguidamente, navega a la carpeta "contracts" y crea un nuevo archivo llamado *HelloWorld.sol*



A continuación se muestra un ejemplo de contrato inteligente "Hello World" de la Fundación Ethereum que utilizaremos para este tutorial. Copia y pega el contenido a continuación en tu archivo HelloWorld.sol, y asegúrate de leer los comentarios para entender lo que hace este contrato:

```
// Especifica la versión de Solidity, utilizando versionado semántico.
pragma solidity ^0.7.0;

// Define un contrato llamado `HelloWorld`.
// Un contrato es una colección de funciones y datos (su estado). Una vez desplegado, un
// contrato reside en una dirección específica en la blockchain de Ethereum.
contract HelloWorld {
    // Declara una variable de estado `message` de tipo `string`.
    // Las variables de estado son variables cuyos valores se almacenan permanentemente en
    // el almacenamiento del contrato. La palabra clave `public` hace que las variables sean
    // accesibles desde fuera de un contrato y crea una función que otros contratos o clientes
    // pueden llamar para acceder al valor.
    string public message;

    // Similar a muchos lenguajes orientados a objetos basados en clases, un constructor es
    // una función especial que solo se ejecuta en la creación del contrato. Los constructores se
    // utilizan para inicializar los datos del contrato.
    constructor(string memory initMessage) {
        // Acepta un argumento de cadena `initMessage` y establece el valor en la variable de
        // almacenamiento
        // `message` del contrato.
        message = initMessage;
    }

    // Una función pública que acepta un argumento de cadena y actualiza la variable de
    // almacenamiento `message`.
    function update(string memory newMessage) public {
        message = newMessage;
    }
}
```

### Paso 9. Conectar MetaMask y Alchemy a tu proyecto

Hemos creado una cartera en MetaMask, una cuenta en Alchemy y escrito nuestro contrato inteligente; ahora es el momento de conectar los tres.

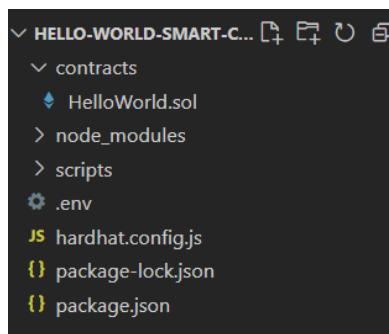
Cada transacción enviada desde tu cartera virtual requiere una firma utilizando tu clave privada única. Para proporcionar a nuestro programa este permiso, podemos almacenar de forma segura nuestra clave privada (y la clave API de Alchemy) en un archivo de entorno.

Primero, instala el paquete dotenv en el directorio de tu proyecto:

```
npm install dotenv --save
```

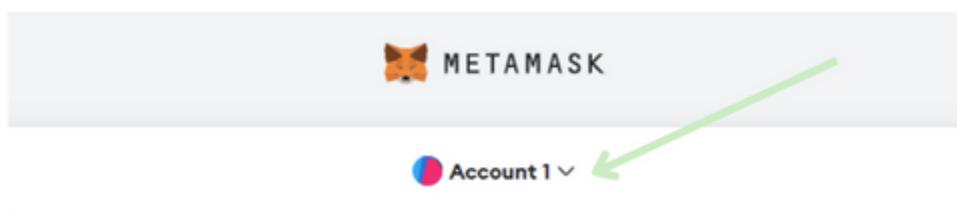
El archivo ".env" es utilizado para almacenar variables de entorno en proyectos de software, proporcionando un método para configurar y mantener valores sensibles, como claves de API o datos de conexión a la base de datos, de manera segura y separada del código fuente.

Luego, crea un archivo .env en el directorio raíz de nuestro proyecto y agrega tu clave privada de MetaMask y la URL de la API HTTP de Alchemy.

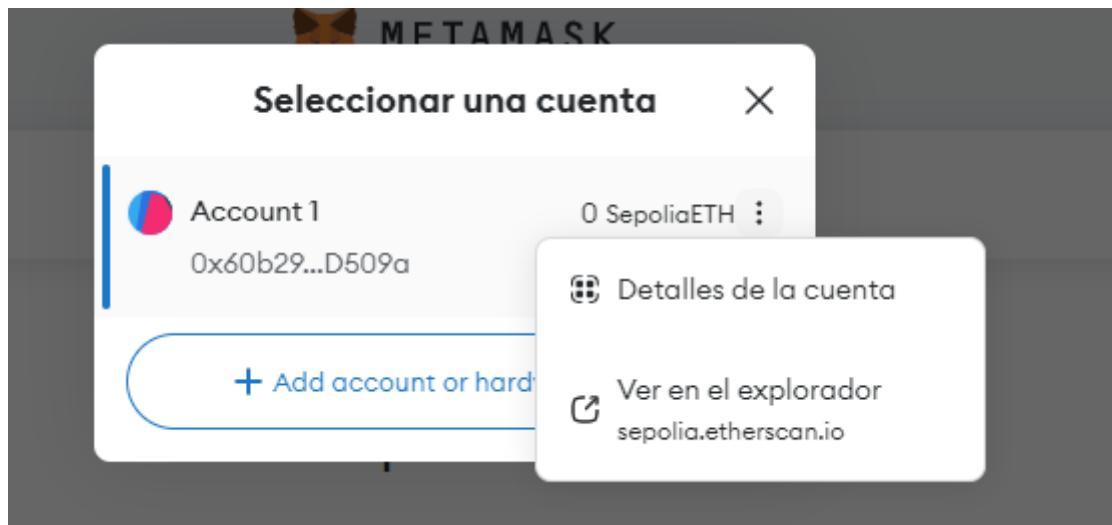


- Exporta tu clave privada de MetaMask:

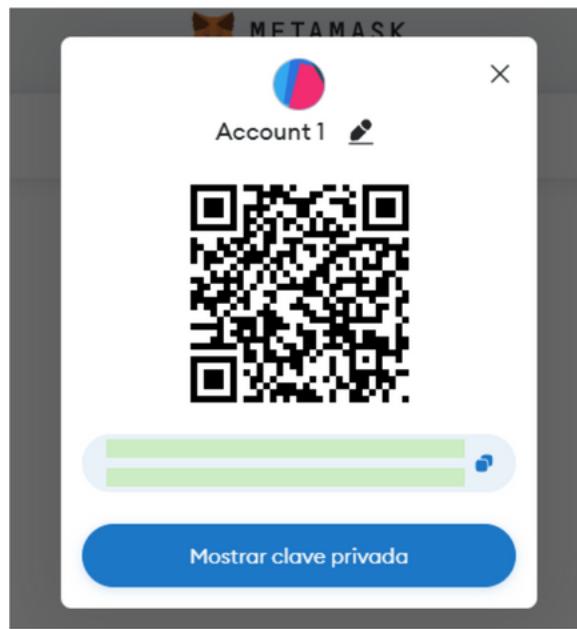
- Haz clic en el selector de cuentas en la parte superior de tu pantalla.



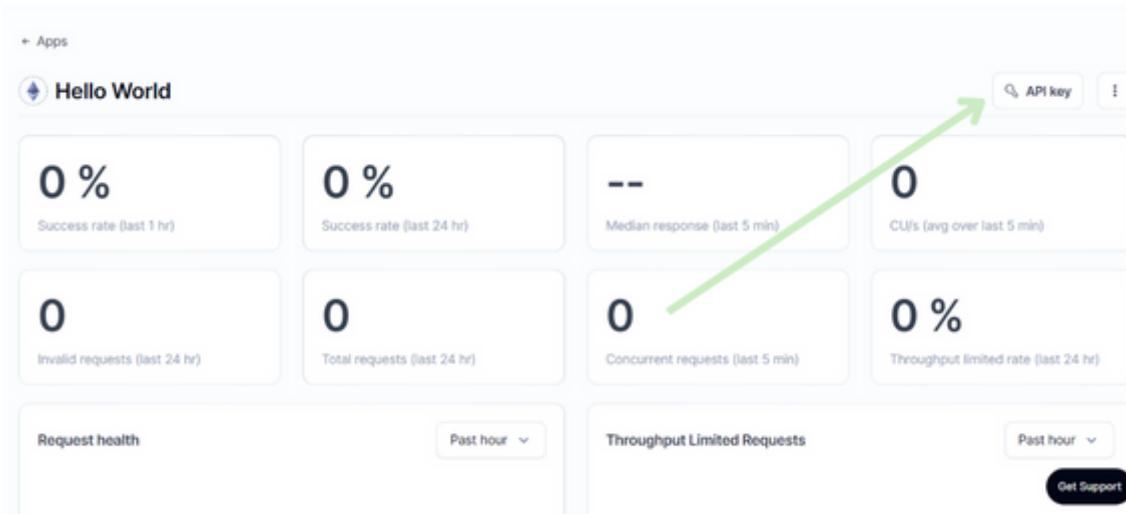
- Haz clic en los tres puntos verticales junto a la cuenta que deseas exportar.



- En la página de 'Detalles de la cuenta', haz clic en 'Mostrar clave privada'.



- Ingresa la contraseña de tu cartera y haz clic en 'Confirmar'.
- Haz clic y mantén presionado en 'Mantener para revelar Clave Privada' para mostrar tu clave privada.
- Haz clic para copiar la clave privada al portapapeles. Asegúrate de guardarla en un lugar seguro.
- Haz clic en 'Hecho' para cerrar la pantalla.
- Obtén la URL de la API HTTP de Alchemy:



- Copia la URL de la API de Alchemy.

Tu archivo .env debería verse así:

```
API_URL = "https://eth-goerli.alchemyapi.io/v2/tu-clave-api"
```

```
PRIVATE_KEY = "tu-clave-privada-metamask"
```

#### Paso 10. Instalar ethers.js

Ethers.js es una biblioteca que facilita interactuar y realizar solicitudes a Ethereum envolviendo los métodos estándar JSON-RPC con métodos más amigables para el usuario.

Hardhat facilita mucho la integración de plugins para herramientas adicionales y funcionalidad extendida. Vamos a aprovechar el plugin de Ethers para el despliegue de contratos (Ethers.js tiene métodos de despliegue de contratos muy limpios y eficientes).

En el directorio de tu proyecto, escribe:

```
npm install --save-dev @nomiclabs/hardhat-ethers "ethers@^5.0.0"
```

#### Paso 11. Actualizar hardhat.config.js

Hemos añadido varias dependencias y plugins hasta ahora, ahora necesitamos actualizar `hardhat.config.js` para que nuestro proyecto sepa acerca de todos ellos.

Actualiza tu `hardhat.config.js` para que se vea así:

```
require('dotenv').config();
require("@nomiclabs/hardhat-ethers");
const { API_URL, PRIVATE_KEY } = process.env;

/**
 * @type import('hardhat/config').HardhatUserConfig
 */
module.exports = {
  solidity: "0.7.3",
  defaultNetwork: "goerli",
```

```
networks: {  
    hardhat: {},  
    goerli: {  
        url: API_URL,  
        accounts: [`0x${PRIVATE_KEY}`]  
    }  
},  
}
```

### Paso 12. Compilar nuestro contrato

Para asegurarnos de que todo funciona hasta ahora, vamos a compilar nuestro contrato. La tarea de compilación es una de las tareas integradas en Hardhat.

Desde la línea de comandos, ejecuta:

```
npx hardhat compile
```

Puede que recibas una advertencia sobre el identificador de licencia SPDX no proporcionado en el archivo fuente, pero no te preocupes por eso, ¡esperemos que todo lo demás se vea bien! Deberías ver algo así:

```
Downloading compiler 0.7.3  
contracts/Helloworld.sol: Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a SPDX license identifier.  
Compiled 1 Solidity file successfully (evm target: istanbul).
```

### Paso 13. Escribir nuestro script de despliegue

Ahora que nuestro contrato está escrito y nuestro archivo de configuración está listo, es hora de escribir nuestro script de despliegue del contrato.

Navega a la carpeta scripts/ y crea un nuevo archivo llamado *deploy.js*, agregando el siguiente contenido:

```
async function main() {  
    const HelloWorld = await ethers.getContractFactory("HelloWorld");  
    // Inicia el despliegue, devolviendo una promesa que se resuelve en un objeto contrato  
    const hello_world = await HelloWorld.deploy("¡Hola Mundo!");  
    console.log("Contrato desplegado en la dirección:", hello_world.address);  
}  
main()  
.then(() => process.exit(0))  
.catch(error => {  
    console.error(error);  
    process.exit(1);  
});
```

```
const HelloWorld = await ethers.getContractFactory("HelloWorld");
```

Una ContractFactory en ethers.js es una abstracción utilizada para desplegar nuevos contratos inteligentes, por lo que aquí HelloWorld es una fábrica de instancias de nuestro contrato hello world. Cuando se usa el plugin hardhat-ethers, las instancias de ContractFactory y Contract están conectadas al primer firmante por defecto.

```
const hello_world = await HelloWorld.deploy();
```

Llamar a deploy() en una ContractFactory iniciará el despliegue y devolverá una Promesa que se resuelve en un Contrato. Este es el objeto que tiene un método para cada una de las funciones de nuestro contrato inteligente.

#### Paso 14. Desplegar nuestro contrato

¡Finalmente estamos listos para desplegar nuestro contrato inteligente! Navega a la línea de comandos y ejecuta:

```
npx hardhat run scripts/deploy.js --network goerli
```

Luego deberías ver algo como:

*Contract deployed to address: 0x6cd7d44516a20882cEa2DE9f205bF401c0d23570*

Si por alguna razón ves el siguiente error:

```
Error: insufficient funds for intrinsic transaction cost [ See: https://links.ethers.org/v5-errors-INSUFFICIENT_FUNDS ]
```

Significa que aún no has recibido los ETH que has pedido.

Si vamos al etherscan de Sepolia(<https://etherscan.io/>) y buscamos nuestra dirección de contrato, deberíamos poder ver que se ha desplegado con éxito. La transacción se verá algo así:

The screenshot shows the Etherscan Sepolia Testnet Explorer interface. At the top, it says "Sepolia Testnet". Below that is the Etherscan logo and navigation links for Home, Blockchain, Tokens, NFTs, and Misc. On the left, there's a sidebar with "Sepolia Testnet Explorer", "All Filters", and a search bar containing "0x3C14A8Bb99a2...". Under "Latest Blocks", there are three blocks listed: 5511009 (16 secs ago), 5511008 (28 secs ago), and 5511007 (Fee Recipient 0x9B984D5a...). On the right, there's a "List Transactions" section showing three transactions: 0x24c47375c9... (From 0xd1bd27c9... To 0xA95bc30...), 0xfe548c23090... (From 0xB1814F... To 0x305ea0...), and 0x47598c9fa4... (From 0x0cfe5a19... To 0x1970aaC06). Each transaction includes its hash, timestamp, and fees.

La dirección "To" coincidirá con la dirección de tu cuenta de MetaMask:

The screenshot shows the Etherscan interface for a Sepolia Testnet address. The address is 0x3C14A8Bb99a246318ae21b5aAEA7BAAb59Aebc82B. The transaction details are as follows:

- Overview:** ETH BALANCE: 0.5 ETH
- More Info:** LAST TXN SENT: No transactions sent; FIRST TXN SENT: No transactions sent
- Multichain Info:** N/A
- Transactions:** Latest 1 from a total of 1 transactions
 

Transaction Hash	Method	Block	Age	From	To	Value	Txn Fee
0x3e65e654678f14696d91...	Transfer	5510786	1 hr ago	0x3C352eA3...6eCC21917	0x3C14A8Bb...59Aebc82B	0.5 ETH	0.00846654

Y si queremos, se puede entrar dentro de la transacción para ver más detalles:

[ This is a Sepolia Testnet transaction only ]

Transaction Details:

- Transaction Hash: 0x3e65e654678f14696d91... [REDACTED]
- Status: Success
- Block: 5510786 [336 Block Confirmations]
- Timestamp: 1 hr ago (Mar-18-2024 10:58:36 AM +UTC)
- From: 0x3C352eA32DFBb757CC0... [REDACTED]
- To: 0x3C14A8Bb99a246318ae21b5aAEA7BAAb59Aebc82B [REDACTED]
- Value: 0.5 ETH (\$0.00)
- Transaction Fee: 0.008466541534752 ETH (\$0.00)
- Gas Price: 403.168644512 Gwei (0.000000403168644512 ETH)

¡Felicitaciones! Acabas de desplegar un contrato inteligente en la cadena de Ethereum 🎉

Para sintetizar toda esta actividad, nuestro objetivo ha sido llevar a cabo la creación de un contrato inteligente que tiene la utilidad de ver y confirmar que se ha realizado una transacción dentro de una red Ethereum. Este contrato inteligente proporciona varias ventajas.

Primero, usar contratos inteligentes en Ethereum nos permite ver todas las acciones que ocurren en ellos de forma clara y segura. Esto nos hace sentir más seguros de que todo está sucediendo como debería.

Segundo, descubrimos que al eliminar intermediarios, como personas que supervisan los acuerdos, podemos hacer que todos los procesos sean más simples y rápidos. No necesitamos esperar a que alguien apruebe algo, el contrato inteligente lo hace automáticamente.

Y hasta aquí llega la práctica de blockchain. ¡Esperemos que os haya gustado!

Esta actividad te introduce en el mundo de los contratos inteligentes en Ethereum, proporcionándote las bases necesarias para comprender y trabajar con esta tecnología. A través de una serie de pasos prácticos, aprenderás a desarrollar y desplegar tu primer contrato inteligente utilizando Solidity, el lenguaje de programación de Ethereum, así como a interactuar con la red Ethereum utilizando Alchemy. Este proceso te brindará una comprensión práctica de cómo funcionan los contratos inteligentes en un entorno de blockchain, desde la configuración inicial de cuentas en Alchemy y MetaMask hasta la

compilación y despliegue del contrato. En resumen, esta actividad te ofrece la oportunidad de adentrarte en el mundo del desarrollo en blockchain y de aplicar tus conocimientos en un contexto práctico y realista.