



ENTREGA FINAL

AMPLIACIÓN SOBRE BLOCKCHAIN

Sistemas Industriales

Sistema de encuestas utilizando la tecnología Blockchain y Ethereum

Judith Vilella Cantos

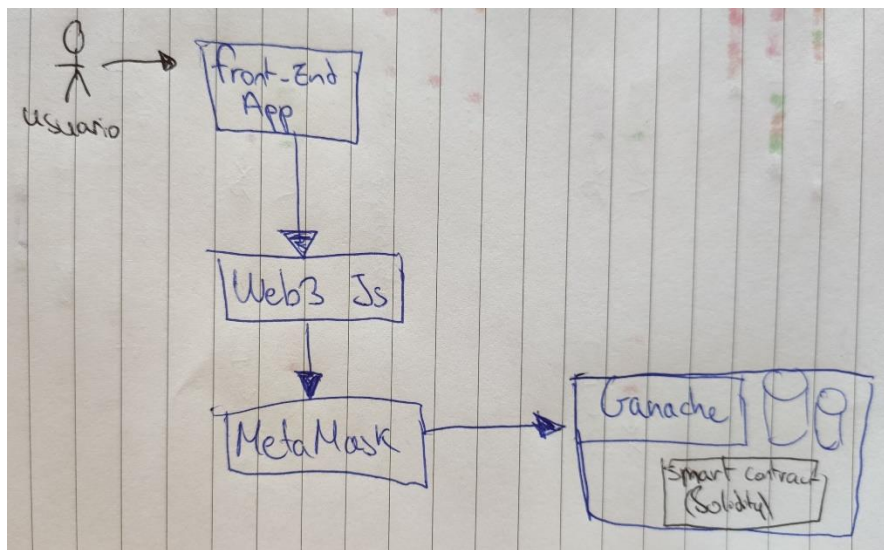
Contenido

Introducción	2
Front (JavaScript y Angular)	2
Smart Contracts (Remix IDE)	4
MetaMask	9
Ganache.....	10
Conectar Blockchain y el Front (Web3).....	15
Conclusiones	20

Introducción

Para la ampliación de una de las dos partes de la asignatura Sistemas Industriales, he decidido ampliar la parte de Blockchain que expuse en el primer entregable. Para esta ampliación proporcionaré un caso de uso real de blockchain, mediante un proyecto desarrollado desde cero por mí. Este proyecto se trata de un sistema de encuestas que utiliza la tecnología blockchain (añadir preguntas y contabilizar votos mediante el sistema de cadena de bloques), y muestra todo al usuario mediante un front hecho con JavaScript y Angular. Para la implementación de esta aplicación mediante el uso de la tecnología Blockchain, se utilizarán Smart contracts para llevar un registro de cualquier actividad relacionada con las encuestas y se usará la red pública de blockchain más conocida para este propósito: Ethereum.

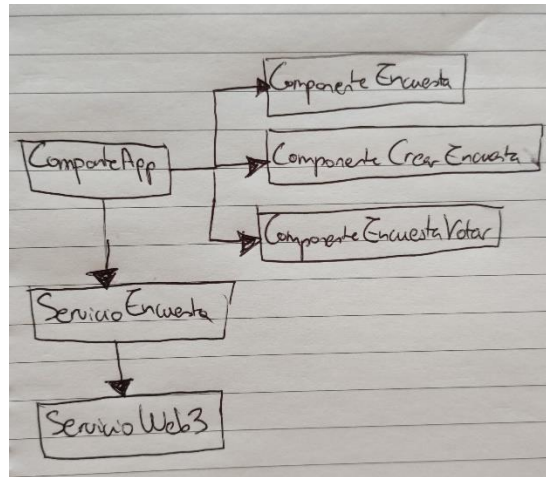
A continuación, adjunto un pequeño esquema que muestra la estructura del proyecto.



Como se puede observar en la imagen, el usuario podrá interactuar con una aplicación web. Esta a su vez conectará con Web3, un módulo de JavaScript que permite crear y administrar conexiones a un internet descentralizado y altamente relacionado con las transacciones de criptomonedas. Este último módulo se conectará a MetaMask, una extensión del navegador que permite conectarse a una red Blockchain (que, como se ve en el esquema, en el caso de este proyecto sería una pequeña red blockchain local desplegada con Ganache) y administrar las transacciones de criptomonedas que se realizan en cualquier página Web. Por último, notar que, dentro de la red local de Ganache, se encontrará nuestro Smart contract, que contiene toda la información relacionada con qué tipo de transacciones se pueden realizar y cómo hacerlo. Cada componente se explicará y utilizará en profundidad en cada uno de los siguientes apartados.

Front (JavaScript y Angular)

El Front-End dispone, a su vez, de su propio esquema:



Como se puede observar en el esquema, el front-end dispondrá de distintos componentes: Por un lado, tendrá los componentes “Encuesta”, “Crear encuesta” y “Votar Encuesta”, que se ocuparán, no solo del aspecto que tendrá en la aplicación cada componente, sino de crear eventos para notificar cualquier cambio relevante en la aplicación (por ejemplo, si una encuesta se ha creado), y de interpretar situaciones que afecten a qué se le muestra al usuario (por ejemplo, si una encuesta no ha sido votada aún, el usuario tiene la opción de hacerlo). Por otro lado, tenemos los servicios, que son los que se encargan de la lógica de la aplicación. Por un lado, está el servicio Encuesta, que se encargará de, dados una serie de datos, operar con la información disponible de las distintas encuestas (añadir un voto, crear una nueva...). Para ello deberá comunicarse con el servicio Web3. Este último servicio será el encargado de realizar toda comunicación con el blockchain y el Smart contract, y se explicará su funcionamiento e implementación en el penúltimo apartado de esta memoria.

En el puerto 4200 de nuestro equipo se estará ejecutando la aplicación front-end de nuestro proyecto, la cual tendrá el siguiente aspecto.



Cuando una encuesta aparezca en la barra lateral a la izquierda como “Votada”, podremos acceder a su gráfica mostrando los votos sobre las distintas opciones a modo de barras verticales. En cuanto a las encuestas que no aparezcan como votadas, si clickamos sobre ellas nos dará la opción de votar como se muestra a continuación.

Sistema de encuestas utilizando la tecnología Blockchain y Ethereum

The screenshot shows a web browser window with the address bar displaying 'localhost:4200/#'. The page title is 'Sistema de encuestas con Blockchain'. The main heading is 'Encuestas' with the subtitle 'Desarrollado con tecnología Blockchain'. In the top right corner, there is a button labeled 'Crear encuesta'. The main content area contains two poll questions. The first question is '¿Cuál es tu mascota favorita?' with a text input field containing '13' and a green 'Votada' button. The second question is '¿Cuál es tu día de la semana favorito?' with radio button options for 'Jueves', 'Viernes', 'Sábado', and 'Domingo', and a grey 'Enviar voto' button.

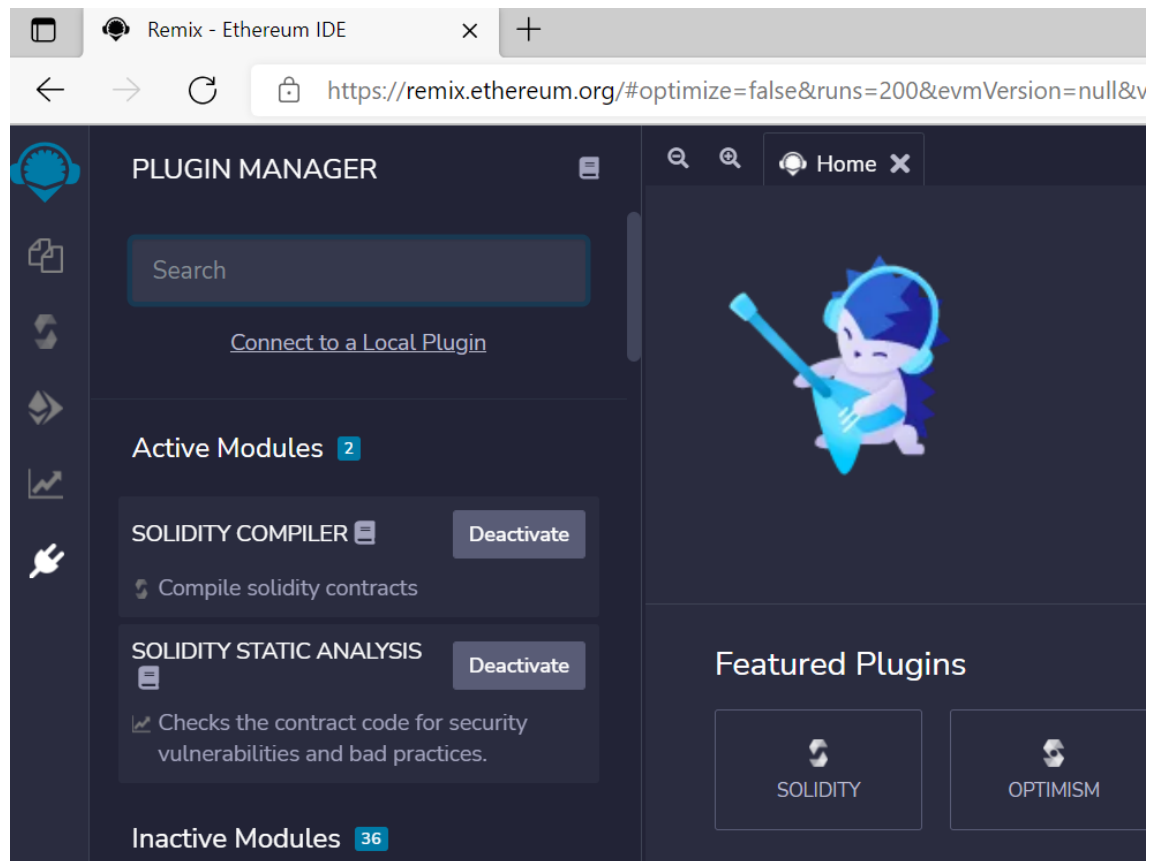
Además, si queremos crear una nueva encuesta bastaría con pulsar el botón de la esquina superior derecha para poder comenzar el proceso.

This screenshot shows the 'Crear encuesta' form. The page layout is identical to the previous one, but the main content area is for creating a new poll. It features a 'Pregunta:' label followed by a text input field containing 'Pregunta de la encuesta'. Below this, there is an 'Opciones:' label followed by four stacked text input fields containing 'Primera opción', 'Segunda opción', 'Tercera opción', and 'Cuarta opción'. A grey 'Publicar encuesta' button is positioned below these options. The poll questions from the previous screenshot are visible in the bottom left corner.

Como vemos en la anterior captura, una vez hacemos click en “Crear encuesta” nos deja introducir la información con respecto a esa encuesta (la pregunta en cuestión y las distintas opciones de respuesta disponibles), y para publicarla habría que pulsar en el botón inferior (“Publicar encuesta”).

Smart Contracts (Remix IDE)

Ahora, pasamos a los apartados puramente de uso de Blockchain del proyecto. Para trabajar con smart contracts (los cuales se utilizarán para registrar cualquier actividad relacionada con las encuestas haciendo uso de la tecnología Blockchain, como he mencionado en la introducción de este trabajo), utilizaremos el lenguaje de programación Solidity. Este lenguaje de programación de alto nivel utiliza una sintaxis muy similar a JavaScript. Está orientado a contratos y, en específico, está enfocado a la máquina virtual de Ethereum (EVM). Para trabajar con esta tecnología he utilizado el IDE Online Remix, ideado para trabajar con Ethereum, la cual es la red Blockchain pública más popular, como expliqué en el primer entregable.



La anterior captura muestra el IDE mencionado con los módulos que he utilizado para crear y trabajar con los mencionados smart contracts (el compilador de Solidity y el analizador de código). Dentro del workspace de este IDE, he creado el archivo para el Smart contract de esta aplicación: encuesta.sol. A continuación, explicaré en qué consiste el código Solidity de este Smart contract paso a paso.

```
1 // SPDX-License-Identifier: GPL-3.0-or-later
2 pragma solidity ^0.8.7;
3
4 contract EncuestaContract{
5     struct Encuesta {
6         uint256 id;
7         string pregunta;
8         uint64[] votos;
9         string[] opciones;
10    }
11
12    struct Votante{
13        address id;
14        uint256[] votadas;
15        mapping(uint256 => bool) MapVotadas;
16    }
17
18    Encuesta[] private encuestas;
19    mapping(address => Votante) private votantes;
```

Primero, se definen dos estructuras: una almacenará los datos de una encuesta (su identificador, un string con la pregunta, un array de enteros con el total de votos para cada opción, y un array de strings para almacenar las distintas opciones) y otra para los datos de un votante (un identificador en formato “address” (dirección), un array de enteros con los identificadores de las encuestas que ha votado y una función map que, dado un identificador de tipo entero, tendrá una equivalencia de tipo booleano que representará si ha votado la encuesta dada o no). El total de encuestas se guardará en un array de encuestas y, los votantes, se obtendrán mediante otra función de mapeo que, dada una dirección, se accederá a la información del votante cuyo identificador sea el mismo a dicha dirección.

```
21     event EncuestaCreada(uint256 _encuestaID);
22
23     function crearEncuesta(string memory _pregunta, string[] memory _opciones) public {
24         require(bytes(_pregunta).length > 0, "Error: Pregunta requerida");
25         require(_opciones.length > 1, "Error: Opciones requeridas");
26         uint256 encuestaId = encuestas.length;
27         Encuesta memory nuevaEncuesta = Encuesta({
28             id: encuestaId,
29             pregunta: _pregunta,
30             opciones: _opciones,
31             votos: new uint64[](_opciones.length)
32         });
33         encuestas.push(nuevaEncuesta);
34         emit EncuestaCreada(encuestaId);
35     }
```

Ahora, empezando con las funciones: se dispone de una función para añadir una nueva encuesta a partir de la pregunta y las distintas opciones (que se pasan por parámetro). Se crea la encuesta tomando dicha pregunta con las correspondientes opciones, y el identificador será el número total de encuestas guardadas (si no hay ninguna será 0, si hay dos será dos, etc). Los votos se inicializan a un array vacío y se añade esta nueva encuesta al array de encuestas. Por último, se crea un evento llamado “EncuestaCreada” que podrá ser capturado por la parte front de la aplicación.

```
37     function getEncuesta(uint256 _encuestaID) external view returns(uint256, string memory, uint
38         require(_encuestaID < encuestas.length, "Identificador de encuesta incorrecto");
39         return (
40             encuestas[_encuestaID].id,
41             encuestas[_encuestaID].pregunta,
42             encuestas[_encuestaID].votos,
43             encuestas[_encuestaID].opciones
44         );
45     }
```

Se dispone de una función que, dado el número de identificación de una encuesta, devuelve toda la información de la misma.

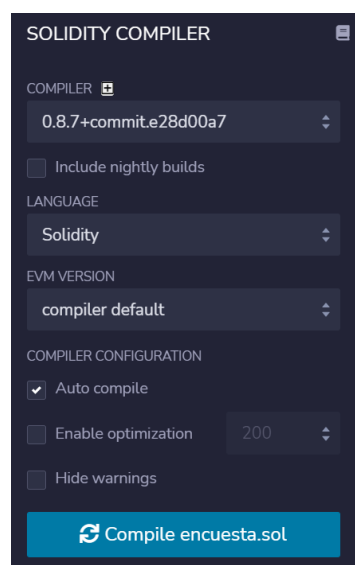
```
47     function votar(uint256 _encuestaID, uint64 _voto) external {
48         require(_encuestaID < encuestas.length, "Identificador de encuesta incorrecto");
49         require(_voto < encuestas[_encuestaID].opciones.length);
50         require(votantes[msg.sender].MapVotadas[_encuestaID] == false, "Ya has votado esta encuesta");
51
52         encuestas[_encuestaID].votos[_voto] += 1;
53         votantes[msg.sender].votadas.push(_encuestaID);
54         votantes[msg.sender].MapVotadas[_encuestaID] = true;
55     }
56
57     function getVotante(address _id) external view returns(address, uint256[] memory) {
58         return (
59             _id,
60             votantes[_id].votadas
61         );
62     }
```

Además, se disponen de dos funciones llamadas votar y getVotante. La primera se encarga de, dado el número identificador de la encuesta y el número de la opción que se ha votado, haciendo las comprobaciones iniciales pertinentes (si dicho votante, del cual obtenemos su dirección mediante la directiva msg.sender, todavía no ha votado esa encuesta, o si son válidas tanto la opción votada como la encuesta indicada). Si todo es correcto, se le añade un voto a dicha opción de la encuesta dada, y se registra dicha encuesta como votada dentro de la información del votante que invoca a esta función (nótese que, como son funciones de tipo external, solo pueden ser accedidas fuera del código). La función getVotante simplemente devuelve la información de un votante dada su dirección que lo identifica.

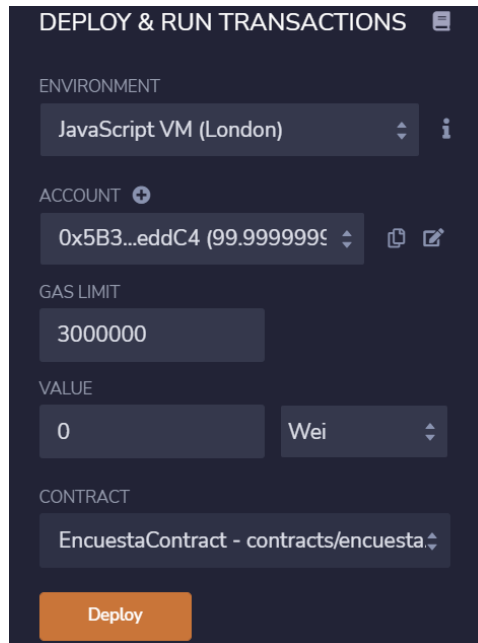
```
64     function getTotalEncuestas() external view returns(uint256) {
65         return encuestas.length;
66     }
67 }
```

Y, por último, se dispone de una función que simplemente devuelve cuántas encuestas hay guardadas en el momento de la invocación a dicho procedimiento.

Una vez terminado el código del Smart contract, procedemos a compilarlo con la segunda opción del menú del borde izquierdo del IDE.



Una vez compilado y todo en orden, procedemos a comprobar el funcionamiento de este Smart contract haciendo “Deploy” del mismo. Para ello, simplemente vamos a la tercera opción del borde izquierdo en el IDE.



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT
JavaScript VM (London)

ACCOUNT
0x5B3...eddC4 (99.99999999)

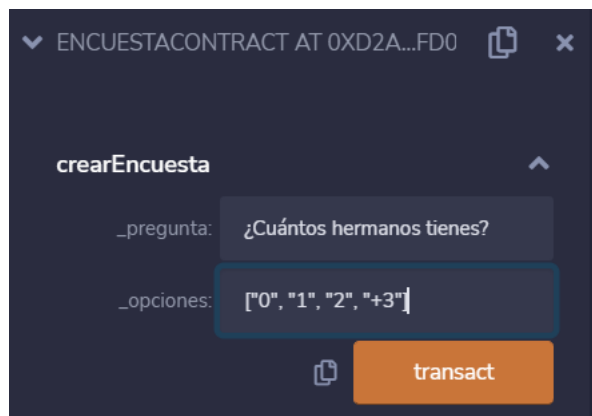
GAS LIMIT
3000000

VALUE
0 Wei

CONTRACT
EncuestaContract - contracts/encuesta

Deploy

Para ello, simplemente elegimos el entorno en el que queremos hacer el deploy (por defecto y para hacerlo de manera local, he elegido la máquina virtual de JavaScript) e indicamos el contrato que vamos a desplegar (para este proyecto, es EncuestaContract, como mostraba el código anterior).



ENCUESTACONTRACT AT 0XD2A...FD0

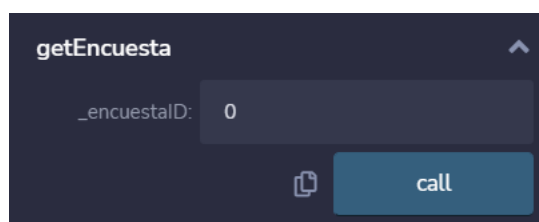
crearEncuesta

_pregunta: ¿Cuántos hermanos tienes?

_opciones: ["0", "1", "2", "+3"]

transact

Una vez desplegado, observamos que nos muestra los diferentes procedimientos del contrato y botones para ejecutarlos. Le paso los parámetros y, primeramente, decido invocar al procedimiento “crearEncuesta” con esa pregunta y opciones.



getEncuesta

_encuestaID: 0

call

El procedimiento “crearEncuesta” no nos da ningún error así que, para comprobar el funcionamiento, invoco a “getEncuesta” para obtener los datos de la encuesta que acabo de crear, si todo ha ido bien.

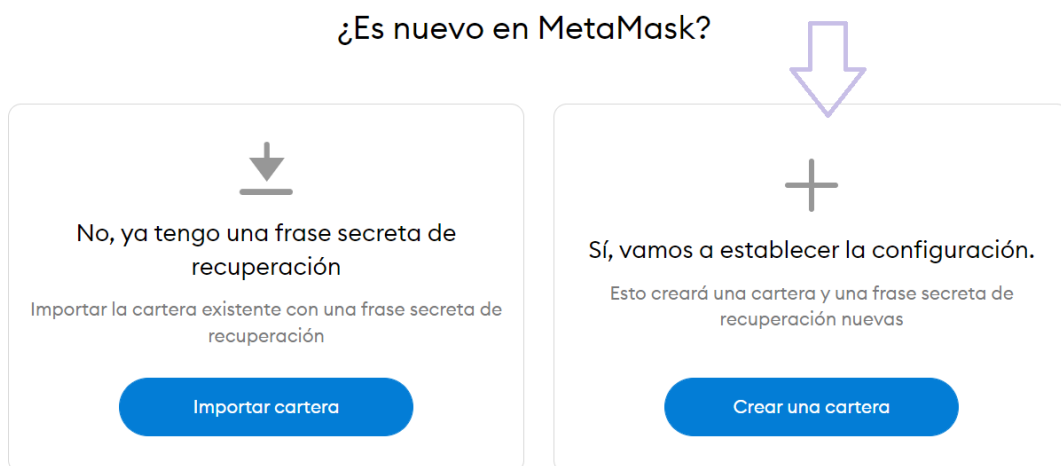
```
decoded input      {
                    {
                        "uint256 _encuestaID": "0"
                    }
                }

decoded output      {
                    {
                        "0": "uint256: 0",
                        "1": "string: ¿Cuántos hermanos tienes?",
                        "2": "uint64[]: 0,0,0,0",
                        "3": "string[]: 0,1,2,+3"
                    }
                }
```

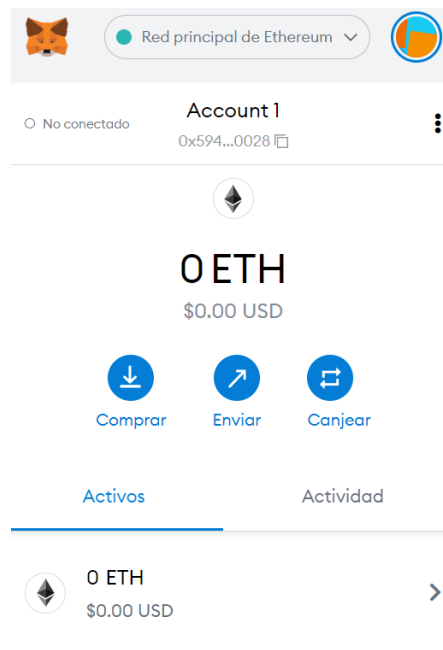
La consola me muestra esta información tras pulsar el botón “call”, por tanto, podemos afirmar que el Smart contract hace su trabajo.

MetaMask

Para desarrollar este proyecto he utilizado el plugin del navegador Google Chrome MetaMask. Este plugin se utiliza para interactuar con la plataforma blockchain Ethereum y, por tanto, para este proyecto es realmente útil. Esta extensión sirve como una especie de “Wallet” o monedero virtual para Ethereum, y se suele utilizar para la compra o venta de tokens. MetaMask no es un monedero de cualquier criptomoneda, es un monedero de Ether (abreviado ETH), es decir, la moneda virtual de Ethereum. Ahora, en cuanto al uso de MetaMask: una vez añadida la extensión en el navegador, teniendo en cuenta el funcionamiento y propósito de dicha extensión, hay que crear una cartera virtual.



Una vez creada la cartera o “wallet”, en esta extensión que acabamos de añadir tenemos al alcance de un click nuestro monedero de Ether.

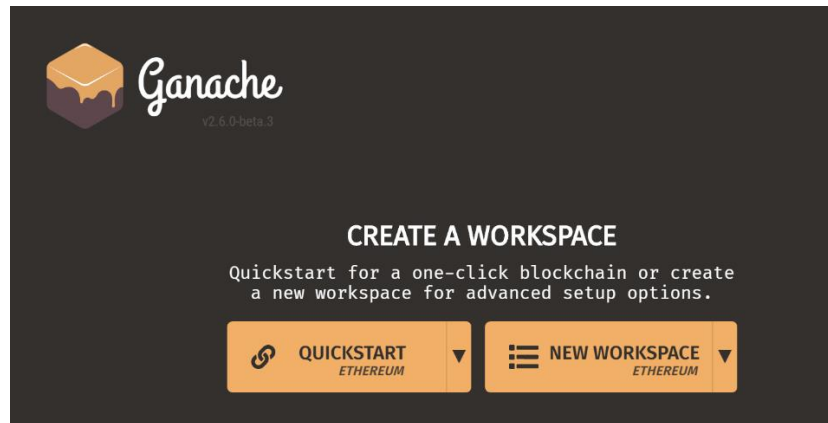


Ganache

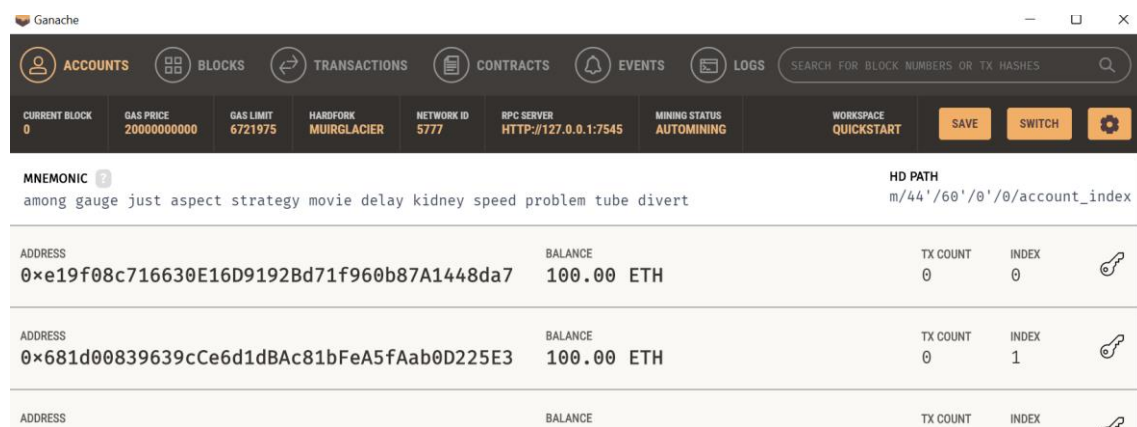
El último apartado independiente de la parte de desarrollo con blockchain y Ethereum del proyecto es instalar la interfaz gráfica de Ganache. Para ello, basta con hacerlo desde su página oficial.



Antes de utilizarlo cabría explicar qué es Ganache. El software Ganache es, simplemente, una red de pruebas local para trabajar con Ethereum. Es decir, una pequeña red blockchain privada que opera sobre Ethereum. Una vez descargado e instalado el programa, este es su aspecto inicial.

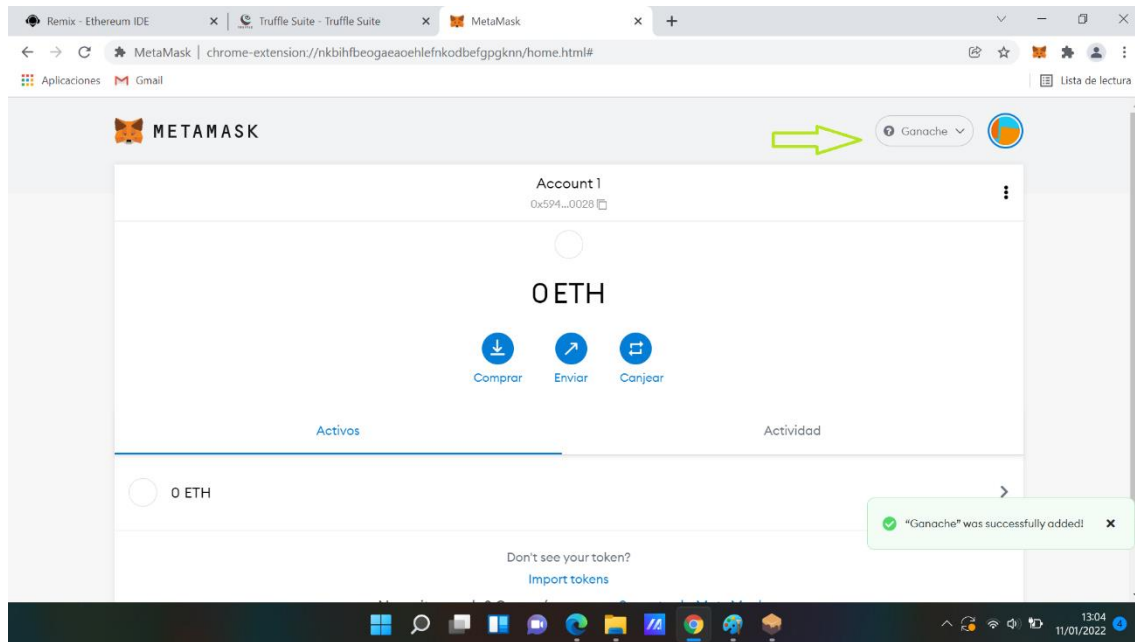


Para el propósito de este proyecto, basta con seleccionar “Quickstart” y tendremos nuestra blockchain local corriendo con funcionalidades más que suficientes.

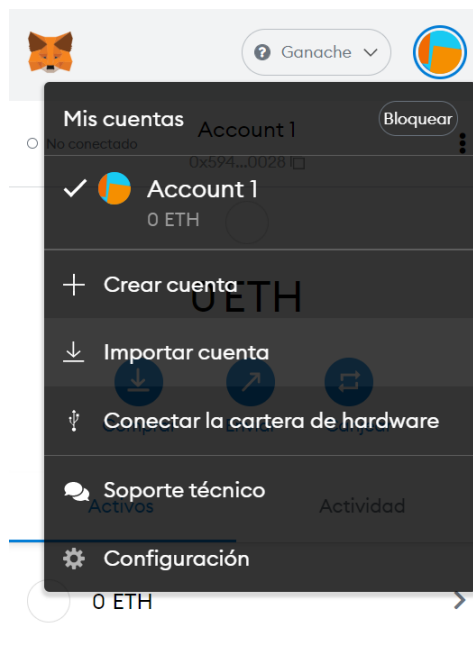


Una vez abierto el programa, en la barra superior podemos ver que está operativo en la dirección localhost de la máquina, concretamente en el puerto 7545. Además, tenemos a nuestro alcance una serie de cuentas con fondos de 100 ETH, cada una con su correspondiente clave privada. La funcionalidad de estas últimas se mostrará más adelante. Una vez esté Ganache en funcionamiento, podemos conectarnos desde MetaMask a esta red, teniendo en cuenta la dirección y el puerto en la que esta se está ejecutando, como muestra la anterior captura.

Sistema de encuestas utilizando la tecnología Blockchain y Ethereum

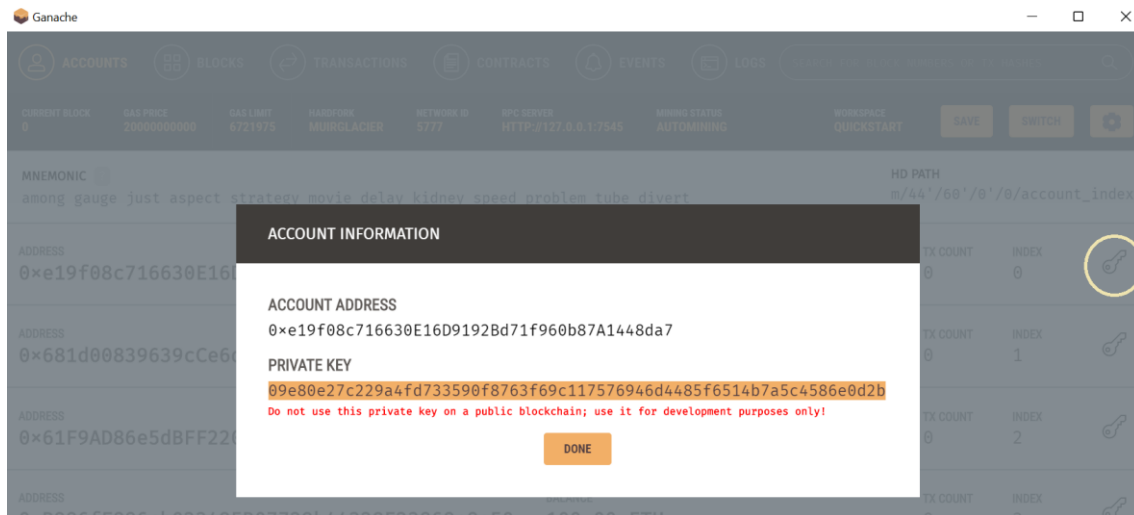


Como muestra esta captura, ahora estamos conectados a Ganache a través de localhost y el puerto 7545. Pero seguimos sin tener ETH en nuestro monedero. ¿Cómo conseguimos Ether? Fácil, importando la clave privada de una de las direcciones que tenemos en la interfaz de Ganache (en este caso, la primera) como una nueva cuenta de MetaMask.

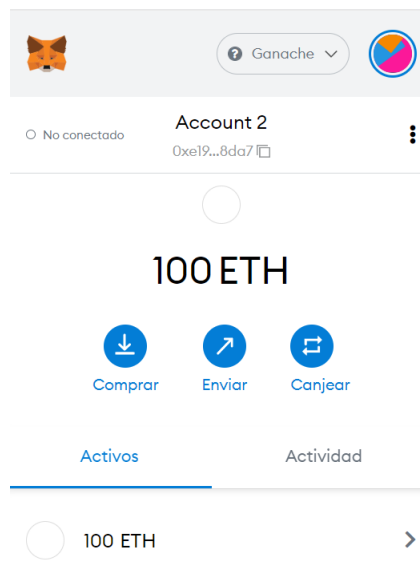


En esta última captura seleccionaríamos “Importar cuenta” y nos llevaría al siguiente paso (introducir la clave privada de la cuenta a importar).

Sistema de encuestas utilizando la tecnología Blockchain y Ethereum

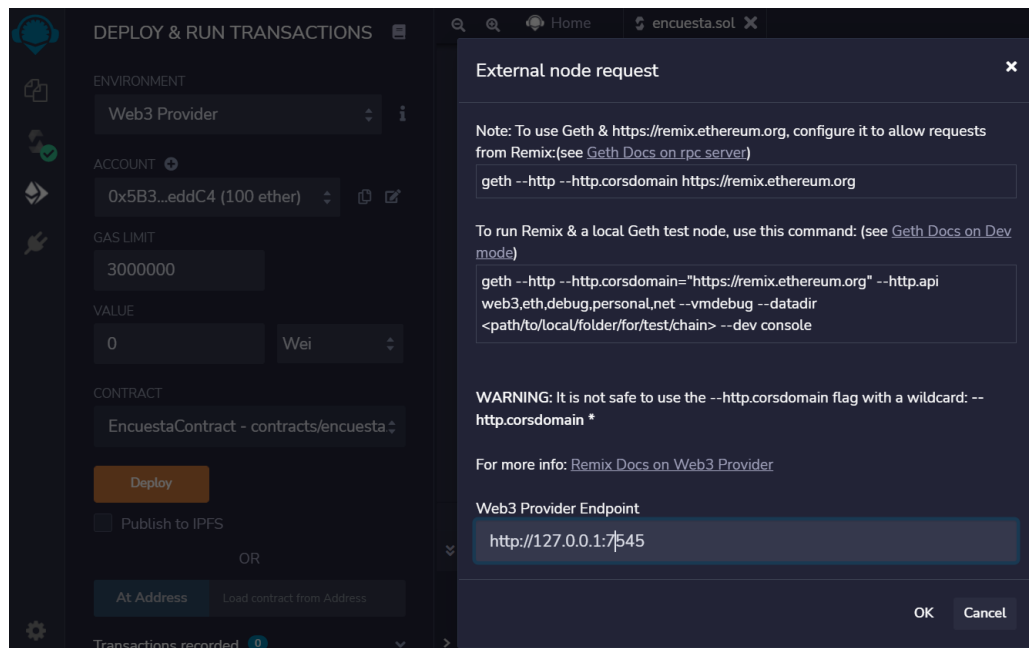


Esta es la clave que he utilizado de Ganache.

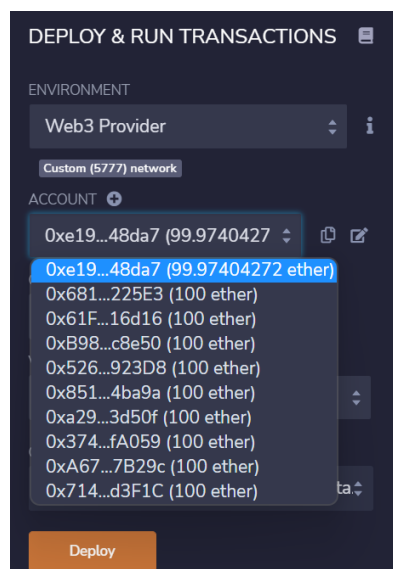


Tras importar la cuenta, ya tenemos fondos en el monedero de MetaMask y aparece como Account (cuenta) 2. Después de todo este proceso, podemos pasar a la parte más importante: ¿cómo despliego el Smart contract que he creado con Remix para mi aplicación (encuesta.sol) ya no solo a nivel local, sino en mi propia red blockchain privada? De una forma muy sencilla: en el IDE Remix, en el apartado de deploy, cambiaremos la opción de Environment (entorno) que estaba puesta como JavaScript VM a Web3 Provider, y ahí pondremos la URL donde se está ejecutando Ganache (como he indicado antes, en este caso será localhost:7545).

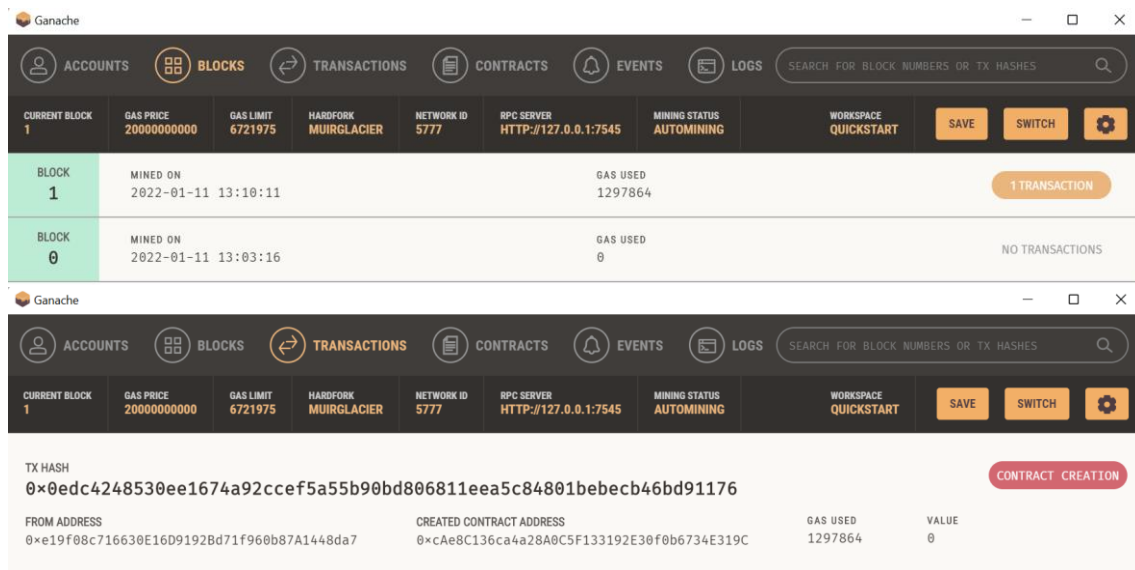
Sistema de encuestas utilizando la tecnología Blockchain y Ethereum



Al pulsar OK, vemos que todo está correcto, ya que nos muestran las diferentes direcciones y la network ID que aparece en nuestra página de inicio de la aplicación de Ganache.



Ahora, veamos si no hay ningún problema si desplegamos este Smart contract en nuestra red blockchain privada.

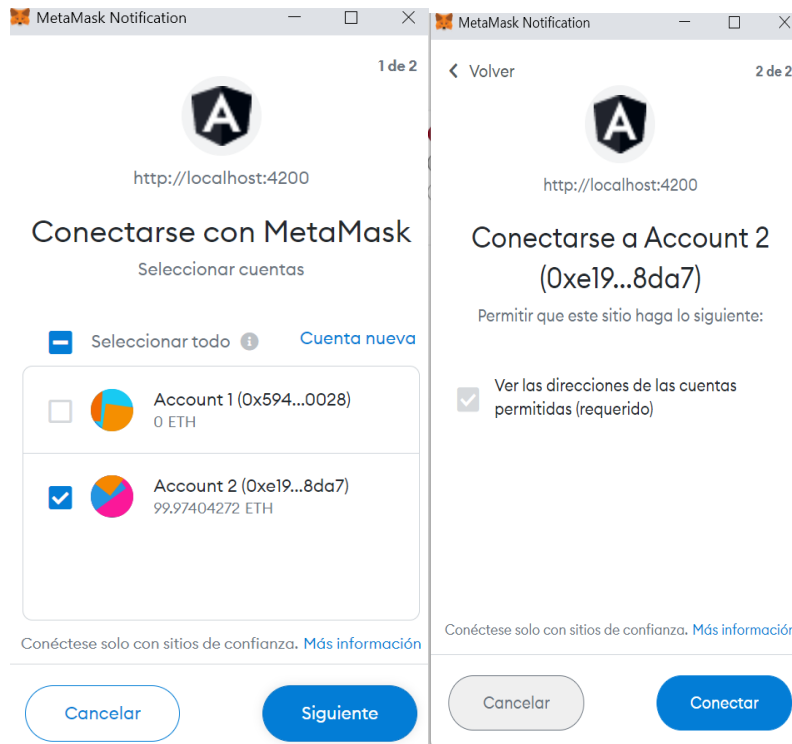


Tras pulsar “Deploy” en Remix IDE, podemos ver en Ganache (tanto en el apartado de Blocks como el de Transactions, como muestran las anteriores capturas) que el despliegue se ha realizado correctamente y además queda constancia de ello: como podemos ver en la última captura, en la transacción podemos ver a la derecha en una etiqueta de color rojo que la transacción ha sido de tipo “Contract creation” (creación de contrato).

Conectar Blockchain y el Front (Web3)

En el apartado anterior he mencionado brevemente que, para desplegar el Smart contract que utilizo en mi aplicación me he conectado a mi red blockchain local de Ganache utilizando para el despliegue, desde el IDE Remix, un entorno de tipo Web3 Provider. Pero, ¿qué es Web3? Web3 es conocido en los últimos años como una variación del conocido World Wide Web (www), en la que se incorpora una descentralización total basándose en la tecnología de la cadena de bloques. Para añadir Web3 al proyecto basta con añadir el nodo web3 y crear un servicio llamado Web3, donde se accederá al Smart contract a través de su ABI (al compilar el contrato en Remix IDE, abajo aparece la opción de copiar el ABI al portapapeles). Yo lo he añadido al archivo contratoABI.json, que se encuentra en una carpeta llamada “blockchain” junto al servicio. El ABI solo cambia si añadimos cambios al Smart contract, pues este archivo contiene toda la información del contrato en formato JSON (las funciones que tiene, los atributos, los tipos que requiere cada variable o parámetro...). Desde el servicio, todo lo que hay que hacer para conectarse a la blockchain local es, a través del navegador, acceder a la cuenta de Metamask que está activa y comunicándose con Ganache. Y, una vez hecho eso, simplemente se añade lógica que permita comunicarse con las funciones del contrato usando typescript, como se usó para desarrollar código en los diferentes componentes en el apartado del front. Ahora, probemos el funcionamiento. Para ello, primero voy a crear una encuesta y luego a votarla, para ver que ocurre satisfactoriamente y que, además, se registran las correspondientes transacciones en la cadena de bloques.

Sistema de encuestas utilizando la tecnología Blockchain y Ethereum



Para empezar, como muestran las anteriores capturas, Metamask nos pide (desde nuestra aplicación web localhost:4200) que autoricemos que esta página acceda a la dirección de nuestra cuenta. Lo permitimos, para poder realizar las transacciones que se desean.

Encuestas
Desarrollado con tecnología Blockchain

Pregunta: ¿Cuántos hermanos tienes?

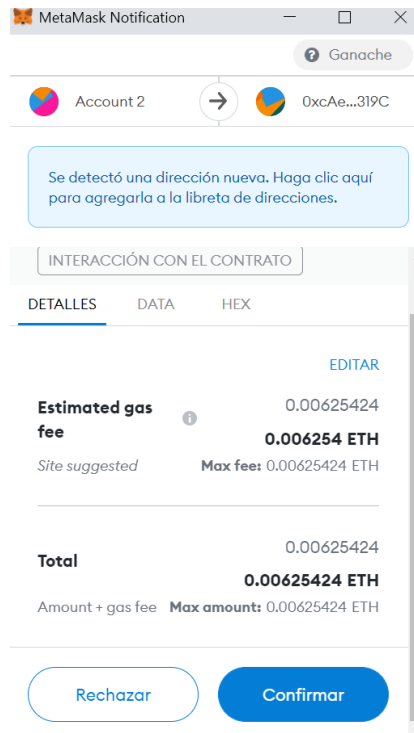
Opciones:

- 0
- 1
- 2
- +3

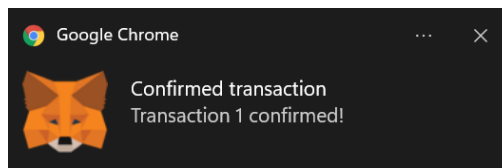
Publicar encuesta

Ahora, relleno los campos como muestra la anterior pantalla para publicar esta nueva encuesta en la web.

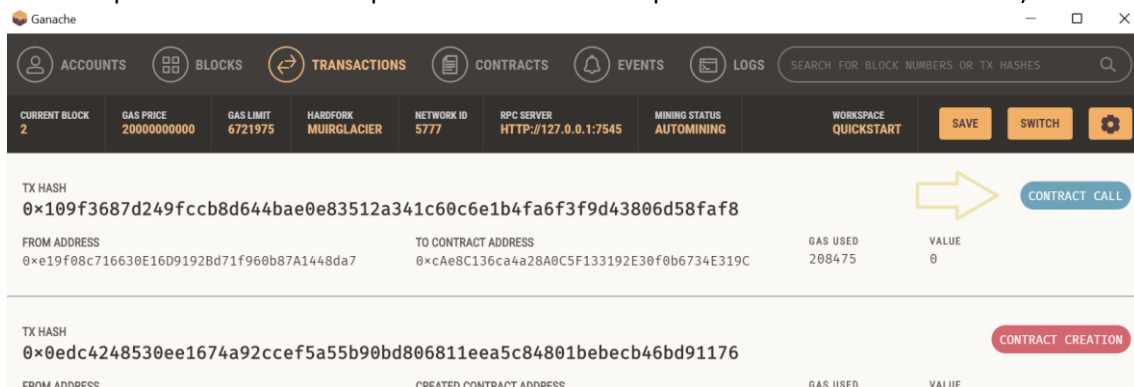
Sistema de encuestas utilizando la tecnología Blockchain y Ethereum



Justo en el momento en el que hago click en “Crear encuesta”, Metamask hace un resumen de la transacción que busco realizar (cuanto ETH requerirá). Confirmamos la transacción y saldrá la siguiente notificación:



Y ahora, si nos vamos a Ganache, veremos que la transacción de crear una nueva encuesta se ha registrado en la cadena de bloques (apartado “Transacciones” de la aplicación) y que aparece dicha transacción como “llamada al contrato” (pues se ha llamado correctamente al procedimiento correspondiente del contrato para crear una nueva encuesta).



Encuestas

Desarrollado con tecnología Blockchain

¿Cuántos hermanos tienes?

0

¿Cuántos hermanos tienes?

☐ 0

☐ 1

☐ 2

☐ +3

Enviar voto

Ahora, la página web se habrá recargado automáticamente (ya que en el contrato se emite un evento al crear un contrato, que he recogido en el front y lo toma como indicativo para refrescar la página). Una vez creada la nueva encuesta, voy a probar a votarla y ver si todo sucede satisfactoriamente.

Blockchain

Encuestas

Desarrollado con tecnología Blockchain

¿Cuántos hermanos tienes?

☐ 0

☒ 1

☐ 2

☐ +3

Enviar voto

Account 2 → 0xcAe...319C

Se detectó una dirección nueva. Haga clic aquí para agregarla a la libreta de direcciones.

INTERACCIÓN CON EL CONTRATO

DETALLES DATA HEX

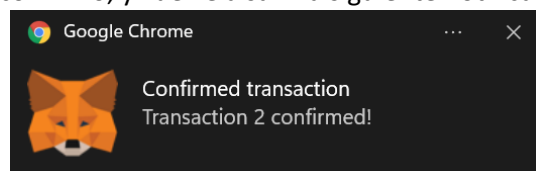
[EDITAR](#)

Estimated gas fee 0.0027666
0.002767 ETH
Site suggested **Max fee: 0.0027666 ETH**

Total 0.0027666
0.0027666 ETH
Amount + gas fee **Max amount: 0.0027666 ETH**

[Rechazar](#) [Confirmar](#)

De nuevo, Metamask me indica cuánto ETH requiere la transacción y me pide que la confirme. La confirmo, y vuelve a salir la siguiente notificación:

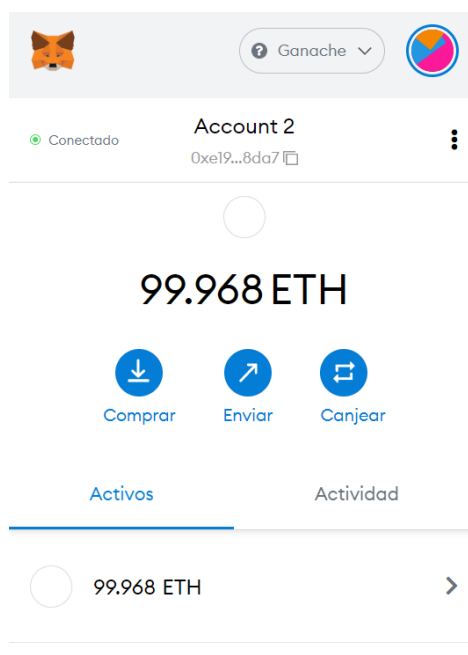




Al recargar la página, ahora la encuesta aparece como “Votada” y, además, se genera el consiguiente gráfico del total de votos. Además, en el apartado “Transacciones” de Ganache aparece la nueva transacción como llamada al contrato. Se añadió satisfactoriamente a la cadena de bloques.

CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE	SAVE	SWITCH	⚙️
3	20000000000	6721975	MUIRGLACIER	5777	HTTP://127.0.0.1:7545	AUTOMINING	QUICKSTART			
TX HASH										
0x22c9a88dbb3b0cde922333a812efd877a3df4fff4fb42b05e48e00fa88c65c										
FROM ADDRESS										
0xe19f08c716630E16D9192Bd71f960b87A1448da7										
TO CONTRACT ADDRESS										
0xcAe8C136ca4a28A0C5F133192E30f0b6734E319C										
GAS USED										
92220										
VALUE										
0										
TX HASH										
0x109f3687d249fccb8d644bae0e83512a341c60c6e1b4fa6f3f9d43806d58faf8										
FROM ADDRESS										
0xe19f08c716630E16D9192Bd71f960b87A1448da7										
TO CONTRACT ADDRESS										
0xcAe8C136ca4a28A0C5F133192E30f0b6734E319C										
GAS USED										
208475										
VALUE										
0										
TX HASH										
0x0edc4248530ee1674a92cce5a55b90bd806811eea5c84801bebecb46bd91176										
CONTRACT CREATION										

Por último, si vamos a nuestra cuenta de MetaMask, podemos ver cómo el balance ha sido actualizado tras realizar esas dos transacciones, pues cada una consume un poco de ETH para realizarse.



Conclusiones

Con todos estos componentes, se obtiene una aplicación web que es capaz de conectarse a Ethereum a través de un blockchain local y capaz de realizar transacciones. Además, se puede observar que el front es completamente independiente del diseño del Smart contract. El código completo del proyecto (tanto de todo el front como el Smart contract en sí, que se encuentra en la carpeta smart_contract como encuesta.sol) está disponible en este repositorio que he subido a GitHub: [JudithV/blockchain-proyecto-main \(github.com\)](https://github.com/JudithV/blockchain-proyecto-main). De todos modos, adjunto a este entregable se encuentra el contrato programado con Solidity, ya que es la base fundamental del proyecto.

Como conclusión, en este proyecto se muestra que, a día de hoy, trabajar con Blockchain e iniciarse en el mundo de las criptomonedas es algo sencillo, aunque se requieran de diversos “componentes”. Se podría afirmar que es algo que cualquier persona puede hacer.