



13-5-2022

Memoria Trabajo CBD

Visc-Labs – Tecnología Blockchain



Javier Vilariño Mayo
Javier Martínez Fernández

Índice

Tecnología.....	2
Blockchain y Smart Contracts	2
Tecnología aplicada a nuestro proyecto	3
Cómo nos hemos enfrentado	3
Problemas encontrados	4
Aplicación creada	6
Catálogo de requisitos.....	7
Objetivos	7
Requisitos generales	7
Requisitos funcionales de información	7
Requisitos funcionales de conducta	8
Requisitos no funcionales de seguridad.....	8
Requisitos no funcionales de usabilidad	8
Gestión del código fuente.....	9
Manual de instalación.....	11
Manual de usuario	18

Tecnología

Blockchain y Smart Contracts

La tecnología escogida para almacenar los datos de la aplicación ha sido la tecnología blockchain. Una blockchain es un sistema basado en una arquitectura descentralizada, donde en lugar de depender de unos nodos centralizados que pertenecen a alguna compañía, la red depende de todos los nodos participantes. Estos nodos son los encargados de gestionar la propia red consiguiendo de esta forma que en caso de que algún nodo falle el sistema siga funcionando correctamente. Por tanto, se trata de una red autogestionada en la que todos los nodos participan de igual a igual, también conocida como *red peer to peer*.

La forma de almacenar información en esta red es a través de una cadena de bloques, donde cada bloque almacena transacciones realizadas.

Para poder añadir un nuevo bloque a la cadena, es necesario que este sea verificado por la red para evitar problemas relacionados con la seguridad. Cada nodo validador, también conocido como minero, trata de validar o “minar” nuevos bloques antes de ser incluidos en la blockchain. Para poder validarlos se requiere de un coste computacional que consiste en tratar de resolver un problema criptográfico basado en el algoritmo sha-256. El primer nodo que consigue resolver este problema podrá agregar el bloque validado al registro de la cadena de bloques, y compartirá esto con el resto de los miembros de la red.

En caso de que otro nodo hubiese conseguido minar otro bloque distinto al mismo tiempo, tendríamos dos cadenas de bloques distintas simultáneamente. Ambas seguirían operando por separado hasta que una de estas fuese más larga que la otra, donde todos los nodos se quedarían con el registro más largo.

Esta forma de validarlos es conocida como Proof of Work y actualmente es la más conocida y usada. Hoy en día están apareciendo nuevas formas de validarlos que no requieran de tanto coste computacional.

Con todo esto conseguimos una red segura donde cada bloque contiene su hash (resultado del problema criptográfico) y el hash del bloque anterior. Por tanto, si alguien tratase de modificar un bloque anterior necesitaría volver a minar todos los siguientes de nuevo, acción que resulta imposible debido a la política mencionada anteriormente donde siempre se escoge el registro más largo.

Para poder aprovechar toda esta tecnología en el ámbito de las aplicaciones, existen redes como la de Ethereum, que permiten la ejecución de contratos inteligentes. Un contrato inteligente es un trozo de código que está en ejecución en la blockchain y que no se puede

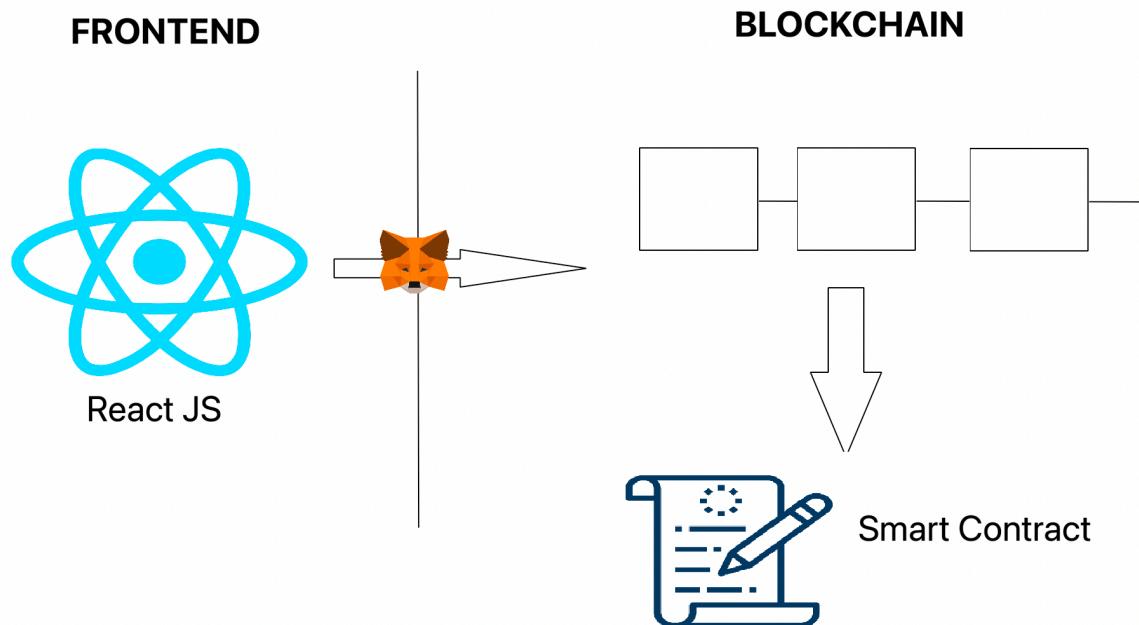
modificar, es decir, los bloques al igual que contienen información de las transacciones pueden contener código.

Con esto conseguimos que los datos que se almacenen en estos smart contracts sean inmutables, accesibles desde cualquier lugar del mundo, seguros y sin que dependan de una entidad que pueda caer o tomar decisiones que repercutan a los usuarios. Por ejemplo: En una red social como Twitter, si esta plataforma decide cerrarse, todos los datos se quedan con ellos, mientras que, en una aplicación descentralizada, los datos pertenecen a todos los usuarios de la red y no a una única entidad.

Tecnología aplicada a nuestro proyecto

En nuestro caso, la aplicación está estructurada de la siguiente forma:

Contamos con un frontend maquetado con el framework React JS que se conecta a la blockchain a través de una cartera de Metamask. Una vez conectados, podemos acceder al contrato inteligente creado y a todos sus métodos como si de una API se tratase.



Cómo nos hemos enfrentado

Para enfrentarnos a esta nueva tecnología, hemos tenido que buscar bastante información en la red, ya que teníamos nociones muy básicas pero que no eran suficientes para llevar con éxito el desarrollo del proyecto.

Para empezar, buscamos a través de diversas fuentes audiovisuales los conceptos de blockchain, Smart Contract, Web 3.0 y Bitcoin, consiguiendo con esto obtener ciertos conocimientos acerca de la tecnología.

Una vez tuvimos los conceptos claros, empezamos a buscar información acerca de la estructura general que suele tener un proyecto basado en la creación de una aplicación descentralizada (DApp) que se conecta a la blockchain. Para esto encontramos distintos tutoriales que mostraban nociones de cómo crear contratos inteligentes básicos y como conectarlos con nuestro código mediante JavaScript. La gran mayoría de estos hace uso de la red de Ethereum por tanto nos decidimos a usar esta para el desarrollo, ya que el resto de las redes como Rose, Solana, etc... encontramos que no estaban tan extendidas todavía.

Por tanto, procedimos a buscar más información relacionada con el lenguaje de programación Solidity, en el cual se desarrollan los Smart Contracts de la red Ethereum.

Una vez consideramos tener los conocimientos suficientes, comenzamos con el desarrollo de la aplicación.

Problemas encontrados

Al comenzar el desarrollo, nos encontramos con bastantes problemas, sobre todo a la hora de instalar el entorno necesario.

El primer problema fue con el framework truffle, ya que al instalarlo en el Sistema Operativo Windows 10 daba bastantes errores. Para solucionar esto fue necesario instalar de forma distinta esta herramienta, así como la instalación de otras librerías npm. En concreto la solución a esto fue la ejecución de los siguientes comandos:

- npm install --global windows-build-tools
- npm install -g truffle --unsafe-perm

El Segundo problema encontrado fue a la hora de conectar Metamask con la blockchain local, ya que al ser nuevos en esta tecnología no sabíamos exactamente qué parámetros debíamos modificar. Finalmente se solventó con la configuración de los puertos, ya que en Metamask se esperaba el Puerto 8545, y la blockchain local estaba desplegada en el 7545.

El tercer problema estuvo relacionado con el desarrollo del contrato inteligente, ya que al ser el primero que desarrollamos aparte de los tutoriales nos encontramos con problemas relacionados con la sintaxis de Solidity.

Por último, hemos encontrado un problema relacionado con la tecnología blockchain, y se debe a que a pesar de la inmensa cantidad de aspectos positivos que ofrece, el costo de realizar transacciones es muy elevado y son bastante lentas, al menos sobre la red de Ethereum, que es la más popular a día de hoy y sobre la que se crean la mayoría de las

aplicaciones descentralizadas. Actualmente, están apareciendo nuevos proyectos que proponen solucionar los problemas mencionados, pero todavía ninguno de estos se ha asentado totalmente en el sector proporcionando la estabilidad que se requiere.

Aplicación creada

Debido al auge de la tecnología blockchain en los últimos tiempos y el interés que esta genera, la propuesta que hemos escogido para el trabajo de la asignatura Complementos de Bases de Datos se trata del desarrollo una aplicación descentralizada (DApp), de forma que el almacenamiento de datos se realice sobre una blockchain local a través de contratos inteligentes.

Respecto a la temática, hemos llevado a cabo la implementación de un sistema, llamado Visc-Labs, que permita tener acceso a los principales datos clínicos de personas físicas, así como la edición o eliminación de estos.

Con nuestra propuesta pretendemos solucionar el problema de centralización de datos médicos que existen a día de hoy, donde no en cualquier parte del mundo la información clínica de un paciente es accesible de forma sencilla. Además, en caso de que un dato concreto de un paciente esté almacenado en una única entidad médica, existe la exposición a que esta entidad tenga cualquier tipo de problema y esa información se pierda, de aquí la necesidad de la descentralización de datos. También se pretende aportar la seguridad y transparencia que proporciona la tecnología blockchain.

Por tanto, el sistema en caso de ser desplegado en una blockchain privada de una organización obtendríamos todas las ventajas de seguridad y transparencia. Pero donde realmente encontramos su mayor potencial es al desplegarla sobre una red pública, además de lo anterior se conseguiría que los datos se encontrasen accesibles globalmente de forma totalmente descentralizada.

La estructura del proyecto se basa en un contrato inteligente, programado en el lenguaje Solidity, desplegado sobre una red Ethereum local y su interacción a través de un frontend desarrollado en ReactJS.

Catálogo de requisitos

Objetivos

OBJ-0001	Recopilar información médica de pacientes
Descripción	El sistema deberá ofrecer un panel de control en el cual se puedan gestionar de manera completa los datos de los pacientes.

OBJ-0002	Control de datos
Descripción	El sistema deberá ofrecer seguridad y control remoto de los datos almacenados.

Requisitos generales

RQG-0001	Gestión de pacientes
Descripción	El sistema deberá permitir la visualización de un panel de control, en el cual se puedan ver los datos completos de los pacientes, editarlos, borrarlos y añadir nuevos datos.

RQG-0002	Almacenamiento y seguridad de datos
Descripción	El sistema deberá permitir el almacenamiento de los datos en una red remota, la cual ofrezca altas funcionalidades de seguridad y a la cual se pueda acceder desde cualquier parte física.

Requisitos funcionales de información

RQI-0001	Información sobre pacientes
Descripción	El sistema deberá almacenar la siguiente información sobre los pacientes: <ul style="list-style-type: none">- Nombre completo- Edad- Teléfono- Dirección- DNI- Alergias- Grupo sanguíneo- Email- Género

Requisitos funcionales de conducta

RQC-0001	Listado de pacientes
Descripción	Al conectarse a la red del sistema, este deberá proporcionar el listado con todos los pacientes creados junto con las distintas acciones disponibles.
RQC-0002	Creación de pacientes
Descripción	Al conectarse a la red del sistema y pulsar para añadir un nuevo paciente, este deberá ser creado correctamente y aparecer en la lista.
RQC-0003	Información completa de pacientes
Descripción	Al conectarse a la red del sistema y pulsar sobre el botón de expansión de un paciente, el sistema deberá mostrar toda la información completa del paciente.
RQC-0004	Eliminación de pacientes
Descripción	Al conectarse a la red del sistema y pulsar sobre el botón de borrado de un paciente, este deberá ser borrado correctamente y la lista deberá ser actualizada.
RQC-0005	Edición de pacientes
Descripción	Al conectarse a la red del sistema y pulsar sobre el botón de edición de un paciente, el sistema deberá permitir la edición de los datos y actualizar la lista en caso de que los datos sean actualizados.

Requisitos no funcionales de seguridad

RNS-0001	Control de transacciones
Descripción	El sistema deberá permitir de algún modo controlar las transacciones realizadas en la red, haciendo transparentes todas estas transacciones para los usuarios del sistema.

Requisitos no funcionales de usabilidad

RNU-0001	Acceso a datos
Descripción	El sistema deberá permitir acceder a los datos almacenados en la red desde cualquier parte física.

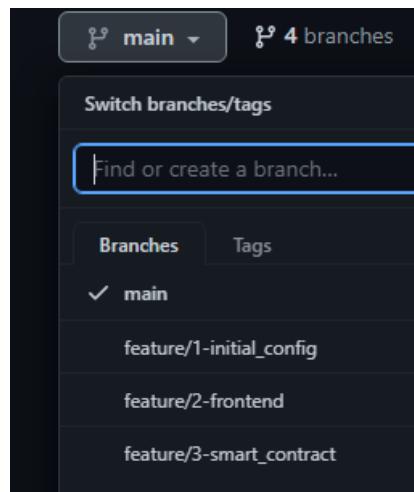
Gestión del código fuente

Para la gestión del código fuente hemos usado GitHub. El enlace al repositorio es el siguiente: <https://github.com/javimv17/Visc-Labs>.

Para gestionar las tareas las hemos dividido en front-end y back-end, siendo estas creadas en el proyecto de GitHub correspondiente. Todos los requisitos referentes a la visualización y estética del sistema han sido realizados en la tarea de front-end y todos los requisitos referentes a la base de datos y la lógica de negocio han sido realizados en la tarea de back-end.

The screenshot shows a GitHub project board for the 'Visc-Labs' repository. The board is divided into four columns: 'To do', 'In progress', 'In review', and 'Done'. The 'Done' column contains three cards, each with a checkmark icon and a title: 'Configuración inicial' (enhancement), 'Frontend' (enhancement), and 'Creación smart contract' (enhancement). A search bar at the top right is labeled 'Filter cards'.

Por cada una de las tareas se ha creado una rama con la nomenclatura “feature/{número de tarea}-{nombre}”.



En el caso de ser necesario, se han realizado “merges” entre las distintas ramas y en el momento en que todos los requisitos se han completado, probado y se han comprobado las buenas prácticas del código, se han realizado las “pulls requests” correspondientes a la rama “main”.

Para la realización de los distintos commits, hemos seguido las buenas prácticas para los títulos y las descripciones de estos.

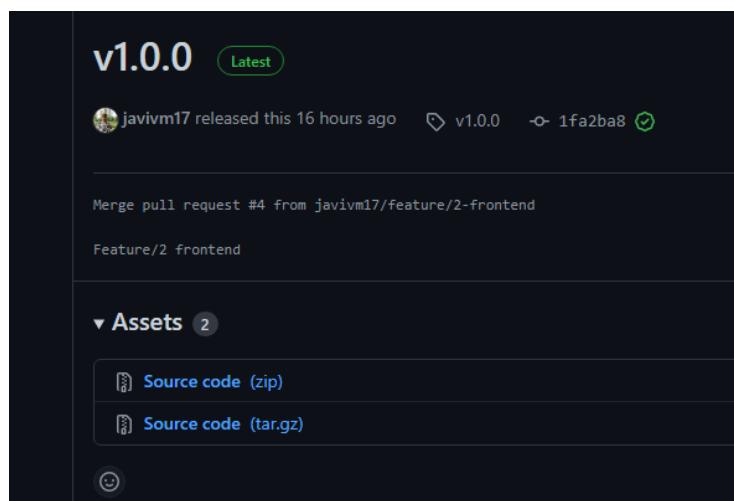
Para el control de versiones, vamos a llevar a cabo un versionado semántico de forma que cada versión del proyecto se definirá de la forma: x.y.z.

Siendo X, la versión mayor la cual se incrementará en caso de un cambio importante en el software que provoque un cambio drástico en la funcionalidad.

Siendo Y, la versión menor que no rompe compatibilidades y añade alguna funcionalidad. Puede incluir pequeñas soluciones a errores.

Siendo Z, el número que representa el arreglo a un nuevo bug o error.

Al dar como terminado el proyecto, hemos creado la release con versión v1.0.0, siguiendo el versionado comentado anteriormente.

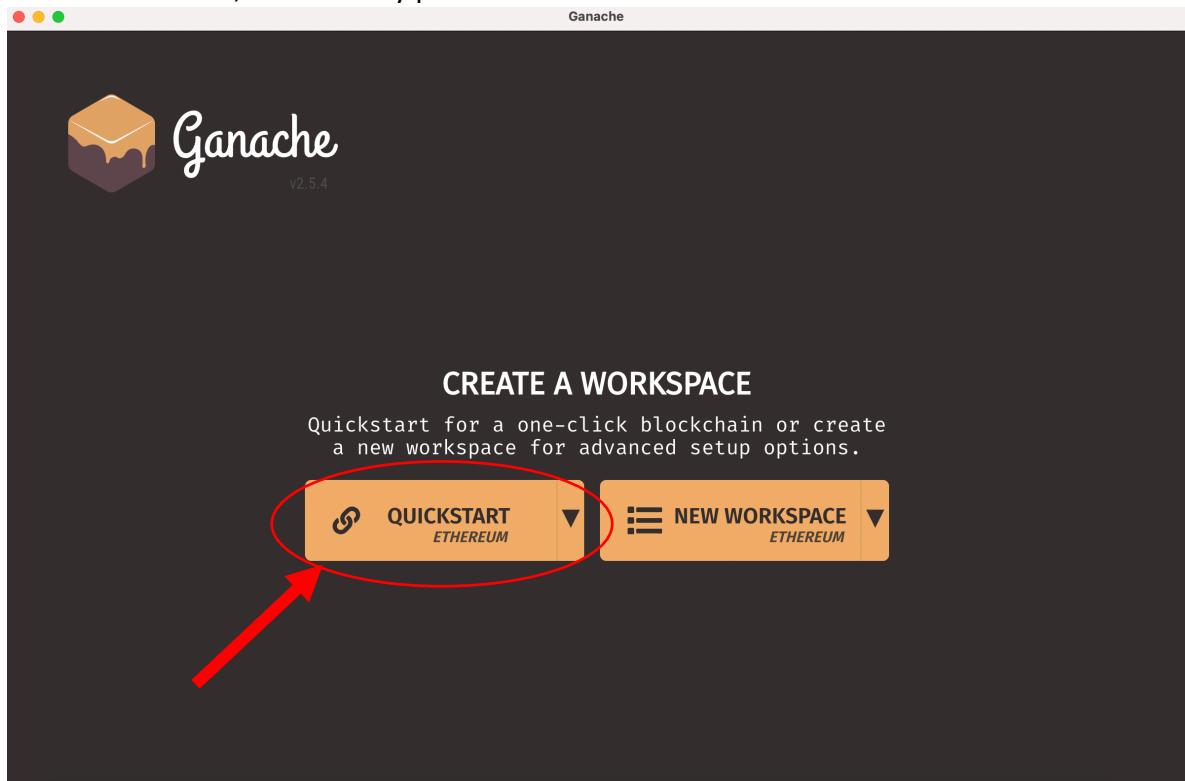


Manual de instalación

Para poder reproducir la aplicación creada es necesario realizar los siguientes pasos:

1. Se debe instalar Node en su versión v16.13.0
<https://nodejs.org/ko/blog/release/v16.13.0/>
2. Se debe instalar npm en su versión 8.1.0
<https://www.npmjs.com/package/npm/v/8.1.0>
3. Se debe instalar Ganache a través del siguiente enlace, ya que este permite correr una blockchain local.
<https://trufflesuite.com/ganache/>.

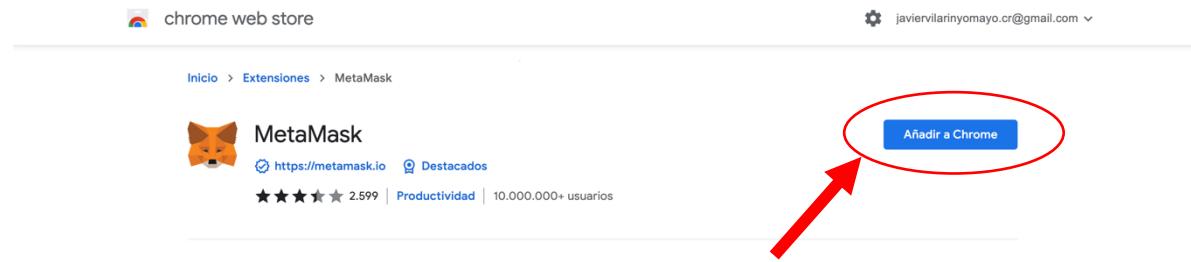
Una vez instalado, lo abrimos y pulsamos en “QUICKSTART”



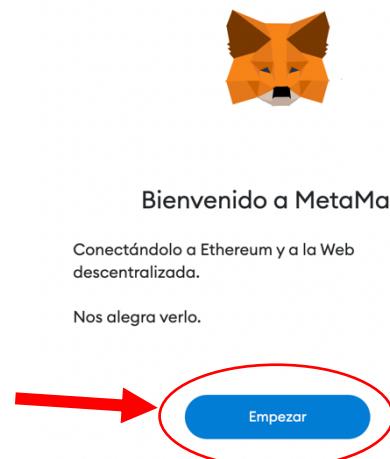
4. A continuación, se debe instalar el conjunto de herramientas truffle de forma global con el siguiente comando: `npm install -g truffle`.

Nota: Es posible que este paso de problemas en el sistema operativo Windows 10/11.

5. En el navegador Chrome se debe instalar, a través de la chrome web store, la extensión Metamask, que es la que nos permitirá conectarnos e identificarnos en una blockchain.



Una vez instalada, se nos abrirá una nueva ventana dando la bienvenida a la aplicación, dónde pulsamos en “Empezar”.



En el siguiente paso, se nos pregunta si ya tenemos una cuenta, en cuyo caso bastaría con importar la frase de recuperación, o en caso contrario pulsamos sobre “Crear una cartera”



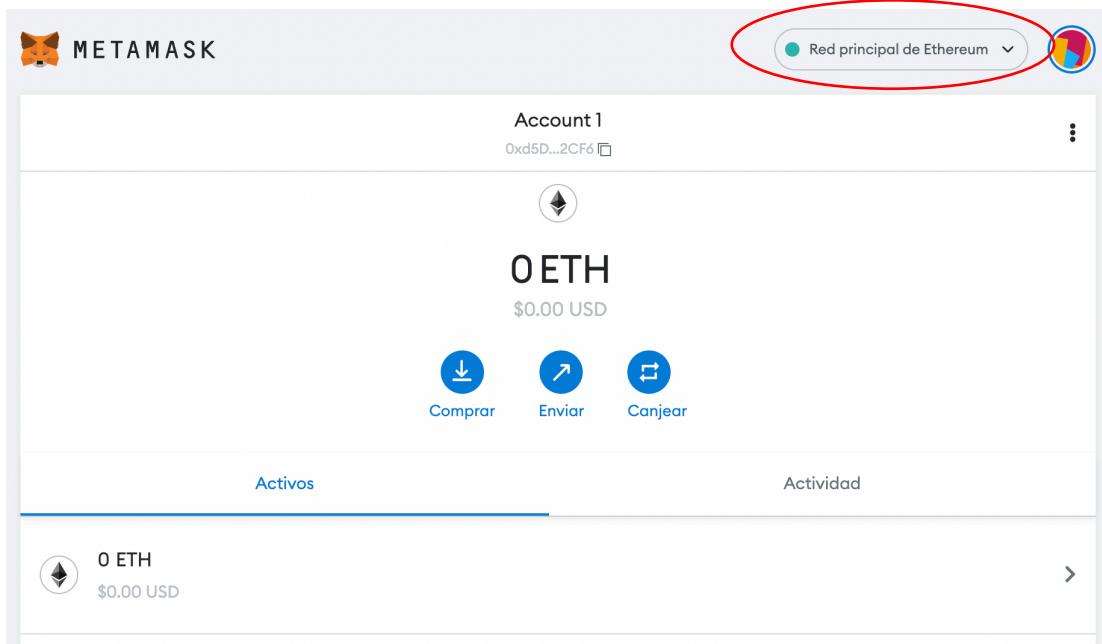
Posteriormente, se nos pregunta si se desea contribuir en la mejora de Metamask mediante la recopilación de nuestros datos. En este caso se puede pulsar la opción que se deseé.

A continuación, nos requerirá de crear una nueva contraseña para nuestra cuenta. Una vez creada, el siguiente paso es muy importante, ya que se obtendrá la frase de recuperación. Esta es fundamental en caso de que olvidemos la contraseña o que queramos importar la cuenta en otro dispositivo.



Pulsamos donde nos indica la imagen, y nos aparecerán un conjunto de palabras ordenadas que debemos copiar en algún sitio seguro. Posteriormente, el sistema nos preguntará por esta frase para confirmar que la hemos copiado correctamente.

Una vez creada la cuenta nos debe aparecer la siguiente ventana, en la cual se nos indica en la esquina superior izquierda que estamos usando la red principal de Ethereum.



Para poder lanzar la aplicación debemos tener configurada una blockchain de pruebas en lugar de la red de Ethereum, ya que es sobre la cual se desplegará el contrato inteligente. Para esto, pulsamos en “Red principal de Ethereum”, y posteriormente en “agregar red”.



Redes

[Show/hide test networks](#)

[Ignorar](#)

✓ ● Red principal de Ethereum

[Añadir red](#)



A continuación, insertamos y guardamos los siguientes valores:

Nombre de la red

localhost 7545

Nueva dirección URL de RPC

http://localhost:7545

Identificador de cadena ⓘ

1337

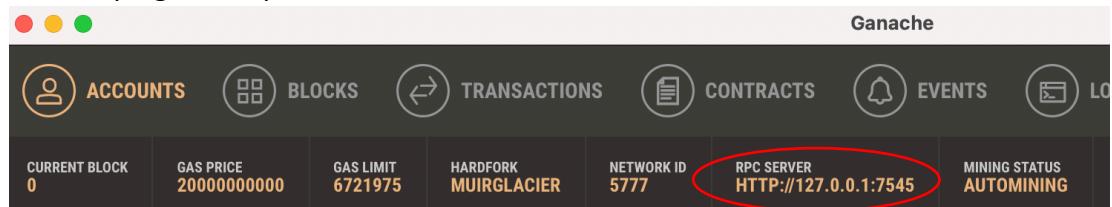
Símbolo de moneda

ETH

Dirección URL del explorador de bloques

(Optional)

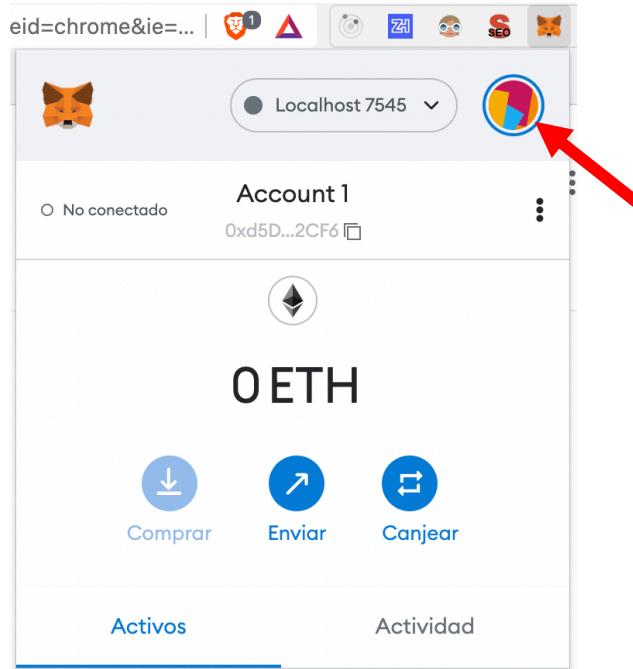
Nota: Es importante el puerto que indicamos en la dirección, ya que este debe ser sobre el que Ganache despliega la blockchain. Para comprobar sobre qué puerto está desplegada, lo podemos ver desde la interfaz de Ganache.



- Para interactuar posteriormente con el contrato inteligente será necesario tener fondos en nuestra cartera Metamask, y para esto podemos importar una de las cuentas de prueba que nos provee Ganache. Para esto, primero debemos obtener la clave privada de una cuenta para posteriormente importarla.

The screenshot shows the Metamask interface. At the top, there are icons for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below this is a search bar labeled 'SEARCH FOR BLOCK NUMBERS OR TX HASHES'. Underneath are status indicators: CURRENT BLOCK (0), GAS PRICE (20000000000), GAS LIMIT (6721975), HARDFORK (MUIRGLACIER), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:7545), MINING STATUS (AUTOMINING), WORKSPACE (QUICKSTART), and buttons for SAVE, SWITCH, and SETTINGS. A mnemonic phrase 'strong south alter exercise rather sentence element rival confirm village letter story' is displayed above the accounts. The accounts section shows two entries: one with address '0xD7dcA690991f4Ec860bDd64E01C9Aa7F84d91462' and balance '100.00 ETH', and another with address '0x315f95ac21C5b1Fa7c2eCb9cccD4c774b0012705' and balance '100.00 ETH'. Each account has a 'key' icon (a padlock with a key) to its right, which is circled in red with a large red arrow pointing to it.

Copiamos la “PRIVATE KEY” que se ha generado. Después, abriremos la extensión de Metamask, donde pulsaremos sobre nuestra cuenta.



A continuación, pulsaremos sobre “Importar cuenta”, dónde introduciremos la clave privada que copiamos anteriormente.

Si todo ha salido bien, debemos poder ver que nuestros fondos han pasado de 0 ETH a 100 ETH.

7. Una vez realizados los pasos anteriores, clonamos el repositorio de la aplicación.
<https://github.com/javim17/Visc-Labs.git>
8. Lanzamos el siguiente comando dentro del directorio del proyecto, que desplegará los contratos inteligentes a la blockchain local: truffle deploy
9. Dentro de la carpeta “client”, lanzamos npm install, para instalar todas las dependencias npm.
10. Por último, lanzamos el comando npm start para abrir el frontend.

A screenshot of a web browser window titled "Visc-Labs" at "localhost:3000". The page displays a table header for "Pacientes" with columns: DNI, Nombre, Teléfono, Edad, Género, and Grupo sanguíneo. Below the header, a message says "No results found". At the bottom, there are navigation links: <<, <, >, >>, and "(1 of 0)". A "Conectar" button is located in the top left corner, and a "+" button is in the top right corner.

Para poder visualizar los datos almacenados, debemos conectar nuestra cuenta de Metamask a través del botón “conectar”. A continuación, se nos abrirá una ventana de Metamask y escogemos la cuenta que deseamos conectar.

Si todo se ha realizado correctamente, debemos poder visualizar lo siguiente:

A screenshot of the same web application after connecting Metamask. The message "Se han registrado un total de 2 paciente/s" is displayed. The patient list now shows two entries:

DNI	Nombre	Teléfono	Edad	Género	Grupo sanguíneo	Actions
77939030B	Javier Vilarino	638824971	21	M	0+	
77939030A	Manuel Luis	638824971	43	M	0+	

The page includes navigation links: <<, <, >, >>, and "(1 of 1)". The "Conectar" button is now replaced by the account address "0xfc3e...8bc6b".

Manual de usuario

Una vez conectados correctamente al sistema, podremos ver una lista con todos los pacientes registrados. La información mostrada en la lista no es toda la información completa de los pacientes. En esta lista podremos filtrar y ordenar por columnas y navegar a través de la paginación de la tabla.

Pacientes							Se han registrado un total de 2 paciente/s		
Ordenación			Filtrado						
DNI ↑↓	Nombre ↑↓	Teléfono ↑↓	Edad ↑↓	Género ↑↓	Grupo sanguíneo ↑↓				
77939030B	Javier Vilarino	638824971	21	M	0+				
77939030A	Manuel Luis	638824971	43	M	0+				

<< < < 1 > >> (1 of 1)

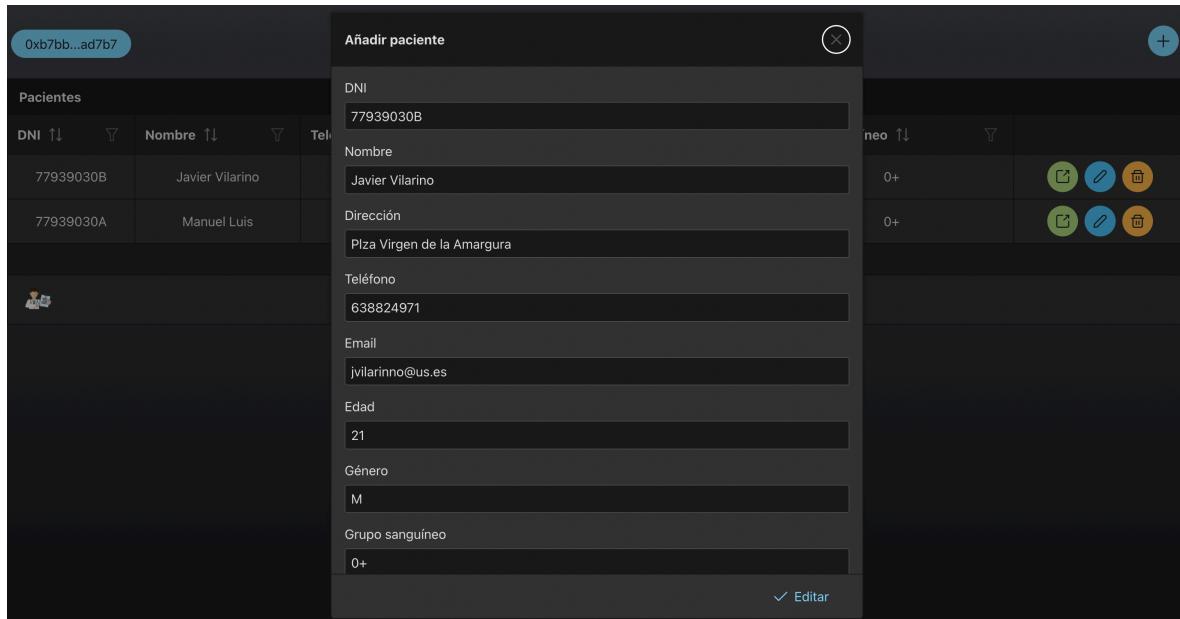
Página

Al pulsar sobre el botón de expansión de un paciente (botón verde), visualizaremos la información completa del paciente correspondiente.

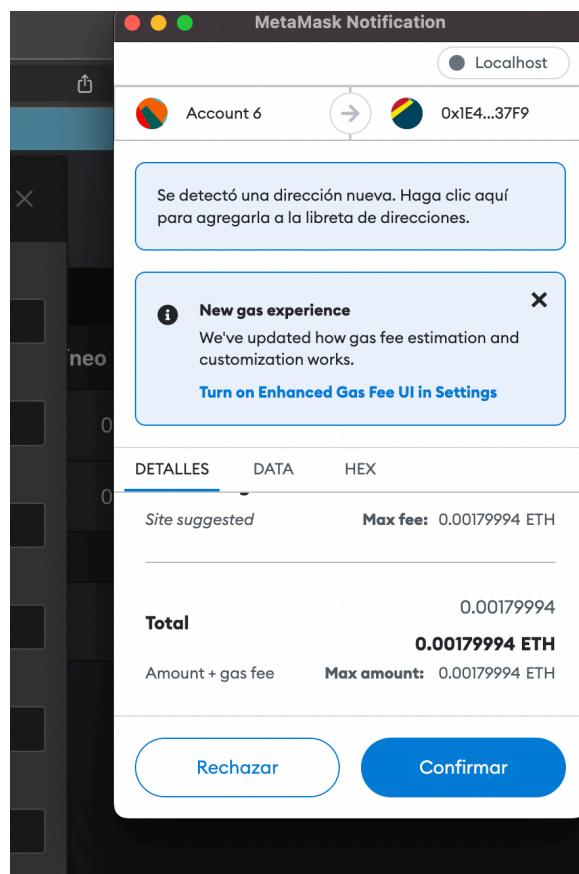
Pacientes		Detalles del paciente						
DNI ↑↓	Nombre ↑↓	Teléfono ↑↓	Edad ↑↓	Género ↑↓	Grupo sanguíneo ↑↓			
77939030B	Javier Vilarino				0+			
77939030A	Manuel Luis				0+			

DNI: 77939030B
Nombre: Javier Vilarino
Dirección: Plaza Virgen de la Amargura
Teléfono: 638824971
Email: jvilarinno@us.es
Edad: 21
Género: M
Grupo sanguíneo: 0+
Alergias: El paciente tiene alergia a los acaros

Al pulsar sobre el botón de edición de un paciente (botón azul), visualizaremos un formulario en el que podemos editar los datos del paciente correspondiente.



Si cambiamos algunos datos y pulsamos en “Editar”, se mostrará una vista en la que podremos confirmar o rechazar la transacción.



Si confirmamos, se realizará la transacción y los datos serán actualizados automáticamente.

The screenshot shows a table titled "Pacientes" with columns: DNI, Nombre, Teléfono, Edad, Género, and Grupo sanguíneo. There are two rows of data: one for Javier Vilarino Mayo (DNI 77939030B) and one for Manuel Luis (DNI 77939030A). Each row has three action buttons: a green edit icon, a blue delete icon, and an orange trash icon. A green success message box at the top right says "Edición exitosa" and "Paciente existente ha sido editado". The status bar at the bottom left shows "0xb2bb...a22e2".

Al pulsar sobre el botón de borrado de un paciente (botón naranja), se mostrará una vista en la que podremos confirmar o rechazar la transacción al igual que antes. Al confirmar los datos serán actualizados automáticamente.

The screenshot shows the same patient table as before, but the row for Manuel Luis is highlighted with a red background. The status bar at the bottom left shows "0xb2bb...a22e2". This indicates that the delete button for Manuel Luis has been pressed, and a confirmation dialog is now displayed.

Al pulsar sobre el botón de añadir un paciente (botón azul en la esquina superior derecha), visualizaremos un formulario en el que podemos añadir los datos del paciente.

Añadir paciente

DNI	
Nombre	
Dirección	
Teléfono	
Email	
Edad	
Género	
Grupo sanguíneo	
<input type="button" value="✓ Crear"/>	

Si rellenamos los datos y pulsamos en “Crear”, al igual que en las ocasiones anteriores, se mostrará una vista en la que podremos confirmar o rechazar la transacción. Al confirmar los datos serán añadidos automáticamente.

✓ Creación exitosa
Nuevo paciente añadido

DNI	Nombre	Teléfono	Edad	Género	Grupo sanguíneo
77939030A	Manuel Luis	638824971	43	M	0+
42545466T	Javi Martínez	635096785	21	M	A+