Expositor de obras de arte sobre blockchain

Albert Campaña Soler
Guillermo Mollá Munilla
David Vives Conesa
Enginyeria Informàtica
Grup 10_1
31 de mayo del 2021

Resumen

Este proyecto se basa en el desarrollo de una dapp (aplicación descentralizada) que pretende ser un museo online, donde la gente pueda disfrutar del arte y los autores puedan estar seguros de que nadie les va a robar su obra gracias a los NFTs.
Esta dapp basa su back-end en la blockchain de Ethereum gracias a un Smart Contract escrito en Solidity y almacena el contenido multimedia en IPFS, un servicio descentralizado.
Para la web se ha usado react y la librería de web3.js para permitir la interacción con el contrato inteligente.
El otro pilar de este proyecto es el desarrollo y uso de una infraestructura donde poder desarrollar dapps cómodamente, creando una figura de DevOps de la que no dependan los desarrolladores para poder avanzar con su trabajo. Para este punto hemos configurado varios servidores con Docker swarm para asegurarnos de que los servicios están siempre funcionando, GlusterFS para conseguir que la memòria persista entre los nodos, hemos dockerizado los servicios y hemos usado GitLab como plataforma para desplegar todos nuestros servicios..

Índice de contenidos

1. Introducción 1
2. Justificación del Proyecto y contexto actual 2
   2.1 Problemática 2
   2.2 Contexto del proyecto 2
   2.3 Punto de partida y aportación del Proyecto 3
   2.4 Blockchain 4
   2.5 Smart Contracts 5
   2.6 ¿Qué es un NFT? 5
3. Objetivos del Proyecto 7
   3.1 Creación del front-end 7
   3.2 Creación del back-end 7
   3.3 Creación de una infraestructura profesional 7
4. Arquitectura 8
5. Tecnologías utilizadas 10
   5.1 React 10
   5.2 Web3.js 10
   5.3 Truffle 10
   5.4 Ganache 11
   5.5 Metamask 11
   5.6 IPFS 11
   5.7 Node.js 12
   5.8 Docker 12
   5.9 GitLab 12
   5.10 GlusterFS 12
6. Método de trabajo 13
7. Implementación 15
   7.1 Front-end 15
   7.2 Back-end 17
   7.3 Infraestructura 19
8. Testing 22
   8.1 Configuración 22
   8.2 Funcionalidades 23
9. Deploy de versiones 27

10. Futuro del proyecto 29
11. Conclusiones 30
12. Bibliografía 31
13. Anexos 32

Índice de figuras

Figure 1: Esquema arquitectura del proyecto 8
Figure 2: React 10
Figure 3: Truffle Logo 10
Figure 4: Ganache Logo 11
Figure 5: Metamask Logo 11
Figure 6: IPFS Logo 11
Figure 7: NodeJS Logo 12
Figure 8: Docker Logo 12
Figure 9: GitLab Logo 12
Figure 10: Gluster FS Logo 12
Figure 11: Working Schedule Albert & David 14
Figure 12: Working Schedule Guillermo 14
Figure 13: Front-end 15
Figure 14: Instalación de docker 19
Figure 15: Montaje glusterfs 19
Figure 16: Creación de discos 20
Figure 17: Código servicios de docker 20
Figure 18: Docker Code 20
Figure 19: GitLab CI 21
Figure 20: Cartera MetaMask 22
Figure 21: Funcionamiento de la cartera MetaMask 23
Figure 22: Pantalla principal del proyecto 24
Figure 23: NFT en propiedad + opción del cambio de nombre y posibilidad de añadir nuevo NFT 25
Figure 24: Función para subir un NFT 25
Figure 25: Transacción en MetaMask 26
Figure 26: Operación de compra en MetaMask 26
Figure 27: Añadir entradas en GitLab 27
Figure 28: Interfaz de GitLab 27
Figure 29: Deploy desde GitLab 27
Figure 30: Código resultante en GitLab 28

1. Introducción

Los NFT son herramientas muy útiles para presentar y demostrar la autenticidad de una obra de arte. Hoy en día este mundo ha quedado muy pervertido por especuladores, ya que todos los portales están enfocados a la compra y venta de NFT como si se tratara de un mercado de segunda mano.

Nosotros vemos más allá y creemos que son una herramienta que debería servir para promocionar y dar a conocer nuevos artistas. Hemos creado un portal web que actúa de museo donde los autores publican sus obras agrupadas por colecciones. Dando prioridad a las obras de artistas, ofreciendo la opción a cualquier usuario de internet a comprar o vender las obras que más le gusten.

2. Justificación del Proyecto y contexto actual

2.1 Problemática

Actualmente es muy difícil que un artista pueda vivir de su arte por varios motivos, el primero y probablemente más importante es la dificultad para darse a conocer. Si eres un pintor sin ingresos no puedes pagar una galería para que exponga tus obras y aunque te gastes todos tus ingresos para hacerlo su arte sólo llega a las personas que tengan la suerte de asistir a la exposición. En el caso de la música es parecido ya que alomejor el músico/cantante puede hacer un concierto en algún bar o pequeño local y ver si a la gente le gusta, ya que aunque todo el mundo puede subir su música a internet, en aplicaciones como youtube o spotify, necesitas miles de reproducciones para ganar muy poco dinero y si no eres famoso nadie busca tus canciones.

Al problema de darse a conocer se le suma que el artista no se lleva la mayoría del dinero que genera su arte, ya que existen contratos abusivos por parte de discográficas y hay injustas comisiones por parte de aplicaciones de streaming que cada vez cobran más pero pagan lo mismo a los artistas.

Otro problema es el gran número de sitios donde la gente consume arte, ya que un artista está casi obligado a publicar sus obras en todas las plataformas y tiendas físicas posibles si quiere llegar a un número suficiente de público como para ganarse la vida.

2.2 Contexto del proyecto

Actualmente estamos en medio de una pandemia donde todo el mundo se ha tenido que confinar en casa y que el ocio ha sido obligado a cerrar por culpa de la COVID, y esto ha afectado sobretodo a los artistas que no han podido exponer sus obras en galerías, que no han podido dar conciertos, que no han podido mostrar su arte al mundo.

Por este motivo la idea de publicar arte en forma de NFT ha sido toda una tendencia durante estos últimos meses. Para muchos artistas publicar su arte online ha sido la única opción y haciéndolo en forma de NFT pueden estar seguros de que reciben el dinero que les corresponde y que nadie copia su obra ni la pública en otras webs apropiándose.

Nosotros nos centramos en pintores, fotógrafos, ilustristas y cualquier otro artista que haga arte visual. Hemos visto que la tendencia en este grupo ha sido crear sus nuevas colecciones en formato NFT y venderlas en aplicaciones online como opensea.io, rarible.com o superrare.co. Todas estas aplicaciones son mercados enfocados a la compraventa, donde se presentan los NFT por separado y la información de la obra, colección y el artista no están en la primera página, ya que son webs pensadas para ser como amazon o ebay. Es por esto que hay artistas que han creado su propio portal para vender sus obras en formato NFT.

2.3 Punto de partida y aportación del Proyecto

Nosotros creemos que hacer un museo online donde los artistas puedan crear y exponer sus obras, que la gente pueda verlas y disfrutarlas solucionando varios de los problemas que hemos mencionado en este documento.

Primero permite que la gente pueda ir al museo sin salir de casa, lo que significa que ni una pandemia mundial puede privar a la gente del arte y que todo el mundo puede acceder a la misma web, permitiendo que los artistas lleguen a mucha más gente, de la que los conocería exponiendo sus obras en una galería. También permite a los artistas subir sus obras en formato NFT sin necesidad de saber programar, con todas las ventajas que tiene publicar sus obras en este formato. Pero la cosa más importante con la que nos diferenciamos de esos mercados online que actualmente se están usando és que damos importancia al autor de la obra, no lo vemos únicamente como un bien con el que especular, ya que en los otros portales las obras se exponen por separado y una vez vendidas el nuevo propietario puede decidir quitarlo de ese portal. Nuestra web siempre mostrará el trabajo del autor junto a su colección incluso después de ser vendida, y si el nuevo propietario no quiere venderlo de nuevo no tiene porque hacerlo, pero no estará privando a todo el mundo de ese arte.

La otra gran aportación de este proyecto es comprobar la importancia de la figura DevOps en el desarrollo de aplicaciones basadas en blockchain y también la utilidad de una infraestructura común donde los desarrolladores puedan trabajar todos a la vez con una tecnología de naturaleza descentralizada.

2.4 Blockchain

Blockchain es una estructura de datos que permite crear un libro de contabilidad digital de datos y compartirlo entre una red de partes independientes. Las cadenas de bloques utilizan la criptografía para permitir que cada participante en una red determinada administre el libro de forma segura sin la necesidad de una autoridad central para hacer cumplir las reglas. La eliminación de la autoridad central de la estructura de la base de datos es uno de los aspectos más importantes y poderosos de esta tecnología. Para conseguirlo, todos los nodos de la red tienen una copia del libro.

Las páginas de este libro se conocen como bloques y están unidas las unas con las otras mediante hashes criptográficos lo que le de esta forma de cadena, por esto se conoce como cadena de bloques o blockchain. Al estar unidas mediante hashes, es fácil detectar cuando un bloque ha sido modificado, e invalida todos los bloques siguientes, al tener todos los nodos una copia del libro es fácilmente recuperable y extremadamente complicado de manipular ya que tendrías que modificar todos las copias a la vez.

Las cadenas de bloques crean registros e historiales permanentes de transacciones, pero nada es realmente permanente. La permanencia del registro se basa en la permanencia de la red. Es decir que a mayor tamaño de la red más segura es la blockchain, por eso la mayoría tienen una criptomoneda asociada, que se usa como incentivo para que haya el mayor número posible de nodos participando en la red.

Cuando los datos se registran en una cadena de bloques, es extremadamente difícil cambiarlos o eliminarlos. Cuando alguien quiere agregar un registro a una cadena de bloques, también llamado transacción o entrada, los usuarios de la red que tienen control de validación verifican la transacción propuesta.

Las características de esta tecnología hacen que consiga generar confianza en los datos digitales. Cuando la información se ha escrito en una base de datos de blockchain, es casi imposible eliminarla o cambiarla. Esta capacidad nunca ha existido antes. Cuando los datos son permanentes y confiables en un formato digital, puede realizar transacciones comerciales en línea de formas que, en el pasado, solo eran posibles sin conexión.

Todo lo que se ha mantenido analógico, incluidos los derechos de propiedad y la identidad, ahora se puede crear y mantener en línea. Los procesos comerciales y

bancarios lentos, como transferencias de dinero y liquidación de fondos, ahora se pueden realizar casi instantáneamente. Las implicaciones para los registros digitales seguros son enormes para la economía global.

2.5 Smart Contracts

Un contrato inteligente es un software autónomo que puede tomar decisiones financieras. En términos simples es un contrato escrito que se ha traducido a un código y se ha construido como declaraciones complejas del tipo "if-else". El contrato puede autoverificar que se han cumplido las condiciones para ejecutar el contrato. Lo hace extrayendo datos confiables de fuentes externas. Los contratos inteligentes también pueden ejecutarse automáticamente mediante la liberación de datos de pago u otros tipos de datos.

La tecnología Blockchain permite que los contratos inteligentes existan porque es una tecnología que consigue que estos sean seguros e inmutables sin la necesidad de una autoridad confiable. Podemos estar seguros que un contrato se ejecutará como fue escrito. No se necesita ninguna aplicación externa. La cadena de bloques actúa como intermediaria y ejecutora.

2.6 ¿Qué es un NFT?

Un token no fungible o NFT es un tipo especial de token criptográfico que representa algo único. Esto contrasta con las criptomonedas como el bitcoin, que son fungibles por naturaleza. Las cuatro principales características de los NFTs es que son únicos, indivisibles, transferibles y con la capacidad de probar su escasez.

Al estar funcionando sobre una blockchain permite demostrar la autoría de este, reseguir todas las transacciones que se han hecho con este, es decir ver todos sus propietarios y cuanto ha pagada cada uno por el NFT, y lo que es probablemente más importante es la posibilidad de vincularlos a contratos inteligentes, con los que el creador de la obra imprimida sobre el NFT puede definir una comisión que llevarse cada vez que alguien compre su obra, y tambíen definir un precio para cada posible uso. Por ejemplo si un NFT que representa una canción va a ser usado para simple reproducción puede ser gratuito, si va a ser la canción de un anuncio costará un precio, para una película otra y para un tono de llamada otro. Lo que asegura los royalties del artista.

A nivel de programación un NFT o ERC 721 es una clase con las funciones y propiedades necesarias para que se puedan hacer todas las cosas que hemos comentado anteriormente de forma segura. Para crear tu NFT solo tienes que crear un contrato inteligente partiendo de esta clase (puedes usar la implementación de OpenZeppelin que es una empresa que proporciona productos de seguridad para crear, automatizar y operar aplicaciones descentralizadas. https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/token/ERC721), indicar los parametros exactos que tendra el NFT y añadir la logica que quieras seguir para los royalties.

Los NFT no sirven únicamente para el mundo del arte, al tratarse de un token no fungible con todas las características que le da la tecnología blockchain de seguridad y privacidad, encaja en contextos como el registro de propiedad donde cada propiedad sería un NFT, quedaría registrado todo lo que pasa a cada propiedad, se podrían intercambiar de forma segura y seria un registro público, también podría implementarse un sistema de historiales mèdicos mundial y público gracias al cual investigadores podrían hacer interesantes estudios y un usuario tendría un único historial para todos los centros mèdicos del mundo, no uno para cada centro, facilitando y mejorando el trabajo de los médicos y todo esto sin comprometer la privacidad. Otro campos en los que se está implementando esta tecnología es en la educación para acabar con la falsificación de títulos y en la identificación ya que permite crear un sistema mucho mejor que el actual basado en pasaportes y dni.

3. Objetivos del Proyecto

Con la idea en mente de hacer un museo online de NFT los objetivos de nuestro proyecto son:

3.1 Creación del front-end

Queremos crear la página web que se comunique con un contrato inteligente y que muestre las obras que hay subidas en ese contrato, ordenadas por artistas y colecciones.
El front-end debe permitir crear NFTs, comprarlos, venderlos y tener una sección donde un usuario pueda ver los NFTs que ha comprado.

3.2 Creación del back-end

Queremos crear un smart contract que será la base de datos de nuestra web, que permitirá la creación de NFTs y transferirlos entre cuentas de forma segura.

3.3 Creación de una infraestructura profesional

Todo el desarrollo de la nuestra plataforma queremos hacerlo en entorno parecido al de una empresa real y hacer que esta plataforma sea resistente a fallos y fácilmente escalable. Para cómo de importante es la figura de un DevOps en el desarrollo de una aplicación descentralizada.

4. Arquitectura

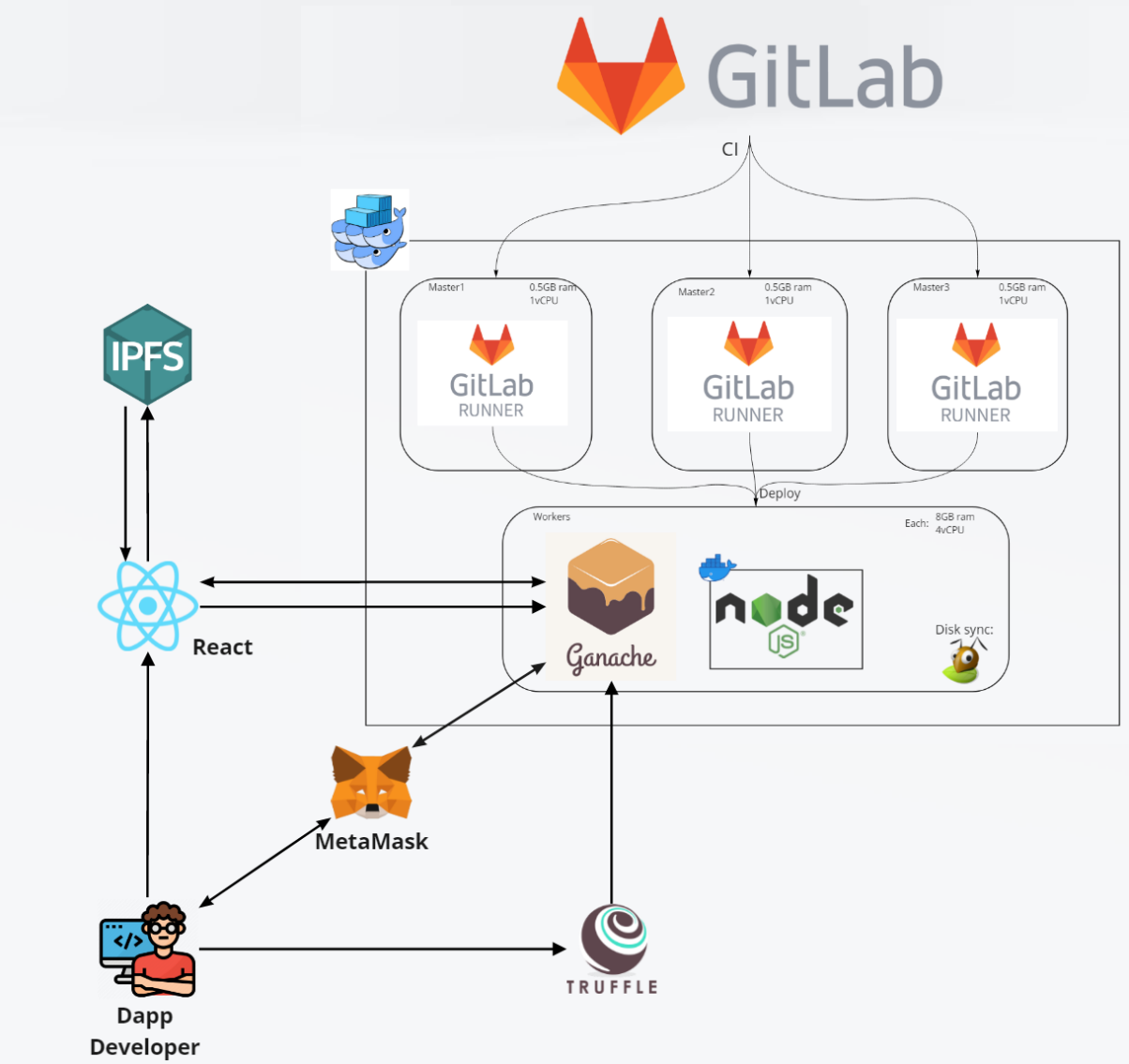


Figure 1: Esquema arquitectura del proyecto

El desarrollador compila el smart contract usando truffle y lo migra a la blockchain de Ganache.

Gracias a metamask el usuario de la aplicación puede conectarse con la blockchain mediante Web3 y también le permite confirmar sus transacciones.

El usuario interactúa con la web hecha en react que sube el contenido multimedia a ipfs, recibe el hash que lo identifica, y lo guarda en Ganache al crear un NFT. A la vez que la web carga y muestra los NFT que hay guardados en la blockchain.

Gracias a Gitlab podemos guardar nuestro código de manera fácil y cómoda. Además usando sus gitlab-runners podemos desplegar el código en cualquier plataforma, en nuestro caso, una de Docker Swarm.

Docker Swarm nos permite mantener nuestros servicios dockerizados en diferentes máquinas manteniendo un estado de HA constante, ya que si un servidor o servicio se muere, se levanta automáticamente en otra maquina. A causa de esta última posibilidad usamos GlusterFS para tener pequeñas porciones de disco sincronizadas, manteniendo el estado de los datos en toda la plataforma

5. Tecnologías utilizadas

Para cumplir los objetivos de este proyecto hemos empezamos buscando cómo podríamos implementar cada punto y hemos decidido usar las tecnologías que mencionamos en este apartado.

5.1 React

Es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Nos ha servido para crear nuestra página web. Hemos decidido usar esta librería por la facilidad que ofrece al usar la API javascript de web3.

Figure 2: React

5.2 Web3.js

Es una API que le permite interactuar con un nodo ethereum local o remoto, necesaria para conectarse al smart contract.

5.3 Truffle

Truffle es un conjunto de herramientas que permite a los desarrolladores crear aplicaciones sostenibles y profesionales en cualquier blockchain utilizando la Máquina Virtual Ethereum (EVM). EVM es el entorno que permite a los desarrolladores crear contratos inteligentes y aplicaciones inteligentes que la Blockchain de Ethereum puede entender.

Figure 3: Truffle Logo

5.4 Ganache

Los smarts contracts en Ethereum son programas que se ejecutan en el contexto de transacciones dentro de la blockchain de Ethereum. Hacer dichas acciones en la red real, supone un coste económico elevado. Para ello, existe Ethereum Ganache, que forma parte de Truffle. Ganache permite recrear la blockchain en un entorno local para testear los smarts contracts, esto permite evitar los costes económicos de trabajar en la blockchain real, y permite un desarrollo seguro de los contratos.

Figure 4: Ganache Logo

5.5 Metamask

Esta API permite que los sitios web soliciten las cuentas de Ethereum de los usuarios, lean datos de las blockchains a las que el usuario está conectado y permitir que el usuario firme mensajes y transacciones.

Tiene una extensión para navegadores que convierte el navegador web2.0 del usuario en un navegador web3.0 (le da la capacidad de interactuar con una blockchain). También es una librería javascript que se integra junto a la librería Web3.js.

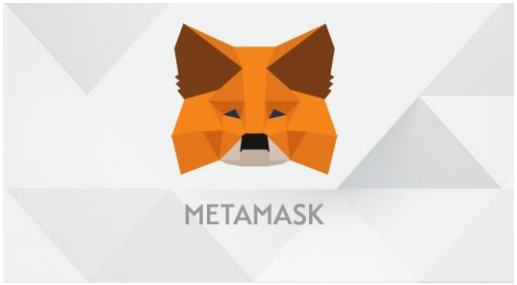


Figure 5: Metamask Logo

5.6 IPFS

IPFS es un sistema distribuido para almacenar y acceder a archivos, sitios web, aplicaciones y datos.

Figure 6: IPFS Logo

5.7 Node.js

Node.js es un entorno de ejecución de JavaScript back-end, multiplataforma y de código abierto que se ejecuta en el motor V8 y ejecuta código JavaScript fuera de un navegador web.

Figure 7: NodeJS Logo

5.8 Docker

Docker es un proyecto de código abierto capaz de automatizar el despliegue de aplicaciones dentro de contenedores de software, lo que nos proporciona una capa adicional de abstracción y automatización en el nivel de virtualización del sistema operativo.

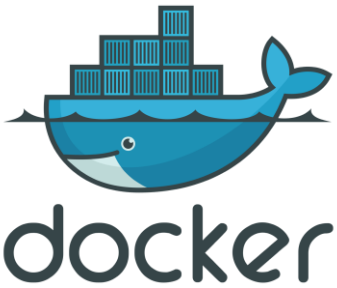


Figure 8: Docker Logo

5.9 GitLab

Gitlab es un repositorio de código que además permite gestión de pipelines de deploy y CI/CD.

Figure 9: GitLab Logo

5.10 GlusterFS

GlusterFS es un software que permite replicación de datos entre servidores para que todos tengan un acceso con velocidad local, pero añadiendo una capa de sincronización

Figure 10: Gluster FS Logo

6. Método de trabajo

Este trabajo lo hemos dividido en 3 bloques, el front-end, back-end y la infraestructura de desarrollo. Al inicio nuestra intención era dividir estos bloques entre los 3 componentes del grupo haciendo que Albert desarrollara la web, David hiciera el Smart Contract y Guillermo hiciera de DevOps montando y proporcionando la infraestructura.

Durante la documentación encontramos un canal de youtube (Dapp University) que hacía proyectos muy sencillos, que seguimos para aprender lo básico para poder empezar con nuestro proyecto, de hecho empezamos clonando un proyecto suyo (https://github.com/dappuniversity/nft), ya que nos proporcionaba una estructura de código bàsica y el archivo de configuración de node.js necesario para poder empezar a programar sin tener que configurarlo nosotros mismos de cero.

Una vez finalizada la etapa de documentación, empezamos a programar, mientras Guillermo montaba la infraestructura Albert usó node.js para crear un servidor local para ir avanzando en el desarrollo de la web. David usó Ganache en local para poder ir testeando las funciones del Smart Contract.

Para probar las funcionalidades de la web tuvimos que empezar a conectar front-end y back-end antes de que la infraestructura estuviera funcionando, lo que suponía un gran trabajo extra porque teníamos que ejecutar ganache cada vez, en el mismo ordenador que ejecutamos la aplicación de node.js y configurar Metamask con las cuentas de la nueva ejecución del ganache.

Es por esto que fue un acierto que una vez la infraestructura estuvo funcionando lo primero que hizo Guillermo fue correr un Ganache dockerizado. Lo que permitió agilizar en gran medida el desarrollo de la aplicación.

Aquí podemos ver un diagrama de Gantt del desarrollo de las distintas tareas realizadas:

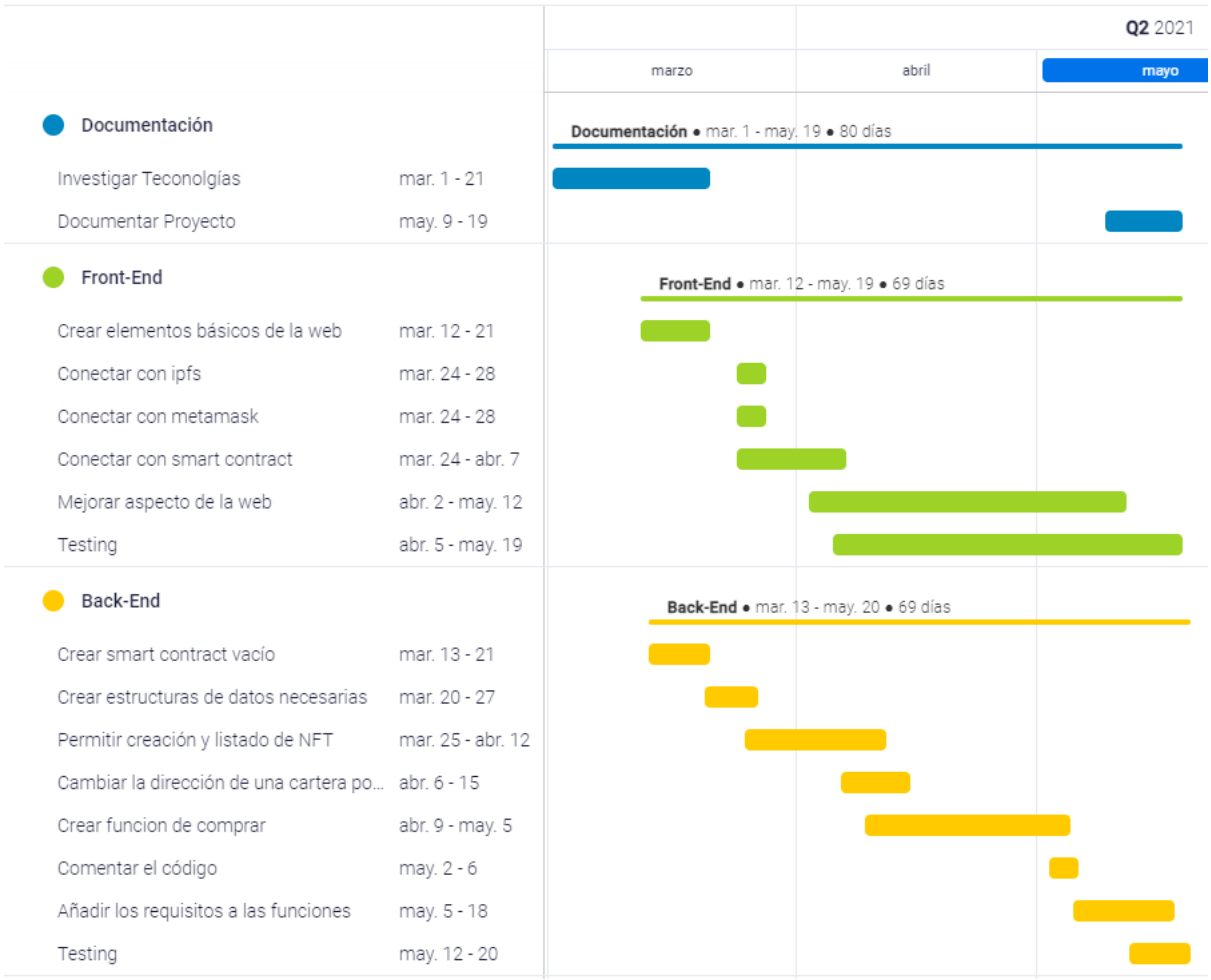


Figure 11: Working Schedule Albert & David

Figure 12: Working Schedule Guillermo

7. Implementación

Para poder ejecutar el front-end en local es necesario tener Node.js instalado en la màquina que se quiera ejecutar, clonar el código del proyecto, abrir una consola en el directorio del proyecto y ejecutar “npm install” esto instalará todas las dependencias necesarias especificadas en el archivo “package.json”.

Ahora ejecutando “npm run start” empezaremos un servidor local que nos permitirá ver el estado de la web en la dirección http://localhost:3000/.

Si se quiere probar el Smart Contract es necesario instalar Truffle i Ganache, ejecutarlo, tener la extensión de Metamask o otra cartera y configurarlo con las cuentas de Ganache. Ahora ejecutando en la cmd “truffle migrate --reset” se compilara y publicará el contrato en Ganache. El último paso es conectar el front-end con Ganache, para ello debemos modificar el archivo truffle-config.js y en development poner los campos host a “127.0.0.1” y port a 7545.

7.1 Front-end

El código del front-end se encuentra en el directorio /src/components, allí podemos ver los archivos css que contiene el principio la paleta de colores usada en la web, ordenada de más claro a más oscuro en orden descendente.

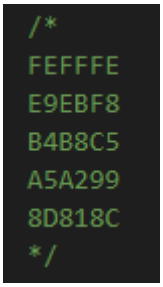


Figure 13: Front-end

También vemos que hay un archivo para cada componente:

Navbar.js hace referencia a la barra superior que contiene un título y la dirección de la cartera del usuario.

Museum.js hace referencia al componente principal donde se muestran todas las obras.

Owned.js hace referencia al componente donde se muestran los NFT de los cuales es propietario el usuario.

FormName.js hace referencia al formulario que permite ponerte/actualizar el nombre del artista.

FormUpload.js hace referencia al formulario para crear un NFT.

Estos componentes se basan en la función render() y son ficheros muy parecidos a un fichero HTML indicando las variables javascript entre { }.

El fichero Main.js contiene la estructura de la página web, se montan los distintos componentes en los div necesarios para darle la forma deseada. En este fichero se aprecia muy bien como se pasan variables entre los distintos ficheros, se hace usando el objeto props que contiene todo elemento de react, al declarar un componente puedes llenar este objeto con todos los elementos que quieras y después dentro del componente los usas llamándolos con “this.props.objeto”.

```
<Museum
        account={this.props.account}
        museum={this.props.museum}
        buyNFT={this.props.buyNFT} />
```

En este código de ejemplo estamos llenando el objeto props del componente Museum con tres elementos que contiene el objeto props del componente Main.
El fichero App.js contiene toda la lógica de la web y en la función render() simplemente llama el componente Main y le pasa todas las funciones y datos que este necesita llenandole el objeto props como se puede ver en el anexo. Este fichero se encarga de la conexión con IPFS y el contrato. Podemos ver como variable global el cliente IPFS que se establece con las siguientes lineas de código:

```
const ipfsClient = require('ipfs-http-client')
const ipfs = ipfsClient({ host: 'ipfs.infura.io', port: 5001,
                          protocol: 'https' })
```

Si instalamos un nodo IPFS en un servidor propio y queremos conectarnos a él tendríamos que poner en host la dirección ip de nuestro servidor y en port el puerto de ipfs.
Dentro de la clase App lo primero que se observa es la función **`componentWillMount()`**, esta es una función de react que se llama antes de renderizar el componente, por esto en ella se llaman dos funciones, **`loadWeb3()`** encargada de asegurarse que nuestro navegador tiene Metamask instalado o otra cartera para poder usar la web.
La otra función que se llama antes de renderizar el componente es **`loadBlockchainData()`** encargada de mirar a que blockchain está conectada la cartera del usuario, mirar que esté el Smart Contract en esa blockchain, lee su abi,

que es el contrato compilado con todas sus funciones, y también lee la dirección de la cartera del usuario para identificarlo.
Una vez se ha establecido la conexión con el contrato se empieza a leer los datos almacenados en el contrato, se ordenan y se guardan en el objeto state, objeto que usa react para representar el estado de esa sesión de la web.
Tenemos la función `captureFile = event =>` encargada de leer y poner en el formato necesario el archivo multimedia para poderlo subir a IPFS cuando se decide crear el NFT.
Se pueden observar también las funciones encargadas de llamar los distintos métodos del Smart Contract, todas siguen la misma estructura, llaman la función con los parámetros adecuados y se esperan a recibir la respuesta. Una vez el contrato responde la función actualiza la variable state de la página para que el usuario pueda ver los resultados sin recargar la página.

7.2 Back-end

El struct addres2name se usa únicamente para poder transformar la dirección de una cartera (en hexadecimal) a un nombre (string). Por lo tanto, contiene dos campos. El primero, una dirección de una cartera. El segundo, un string que hace referencia al nombre del propietario (autor) de la cartera.

```
struct addres2name{
    address userAddress;
    string username;
  }
```

El struct Image contiene todos los atributos necesarios para formar un NFT. Este struct se usa para almacenar la información de los NFT. Por ejemplo, en la función uploadImage, se crea un nuevo struct Image con los parámetros pasados por la función. Más adelante en funciones como la de buy, se reutiliza el struct para recuperar su precio y owner.

```
struct Image {
  uint id;                    //id of the nft
```

```
    string title;              //title of the nft
    uint price;                //price of the nft
    string hash;               //hash from ipfs
    string description;        //nft description
    string collection;         //which collection the nft bellongs
    address payable author;    //address of the creator
    address payable owner;     //address of the actual owner
  }
```

Este evento se usa cuando se crea un nuevo NFT utilizando la función uploadImage(). Emite la información que corresponde al NFT que se acaba de crear.

```
event TokenCreated(
    uint id,
    string title,
    uint price,
    string hash,
    string description,
    string collection,
    address payable author,
    address payable owner
  );
```

Este evento se usa cuando se añade un nuevo nombre utilizando la función updateUsername(). Emite la dirección de la cartera del autor y el nombre que se ha asociado a dicha cartera.

```
event UsernameAddeed(
    address author,
    string name
  );
```

Este evento se usa cuando se compra un NFT mediante la función de buy(). Emite la dirección de la cartera del vendedor, la dirección de la cartera del comprador, y finalmente el precio de dicho NFT.

```
event NftBought(
   address _seller,
   address _buyer,
   uint256 _price
 );
```

```
function updateUsername(string memory _name) public {}
```

7.3 Infraestructura

Empezaremos con la “marca” de servidores escogida. Los servidores escogidos son los de IONOS, a causa de su coste barato y su buenas prestaciones. Para un comienzo estilo start-up estan muy bien.

Después de conseguir tales servidores hemos tenido que instalar docker swarm en todas las máquinas e iniciar un cluster. (despues del --token irian el token y la IP del servidor)

```
apt-get install docker-ce docker-ce-cli containerd.io
docker swarm join --token
```

Figure 14: Instalación de docker

Al tener todo esto montado, tuvimos que pasar a montar glusterfs en los servidores workers.

```
sudo apt-get install software-properties-common -y
sudo add-apt-repository ppa:gluster/glusterfs-3.12
sudo apt-get update
sudo apt install glusterfs-server -y
sudo systemctl start glusterd
sudo systemctl enable glusterd
gluster pool list
```

Figure 15: Montaje glusterfs

Después de esto deberíamos crear los discos que creamos necesarios para nuestros servicios. Este sería un ejemplo:

```
gluster volume create granafa worker01:/gluster/grafana force
gluster volume start grafana
```

Figure 16: Creación de discos

Ahora ya tenemos la infraestructura, necesitaremos una manera de guardar y deployear nuestro codigo. En este caso usaremos gitlab.com que con crearnos una cuenta gratuita ya nos permite usar todas las funcionalidades que necesitamos.
Ahora necesitaremos que gitlab pueda enviar ordenes de deploy a nuestro servidor, asi que montaremos un gitlab runner privado en la plataforma.

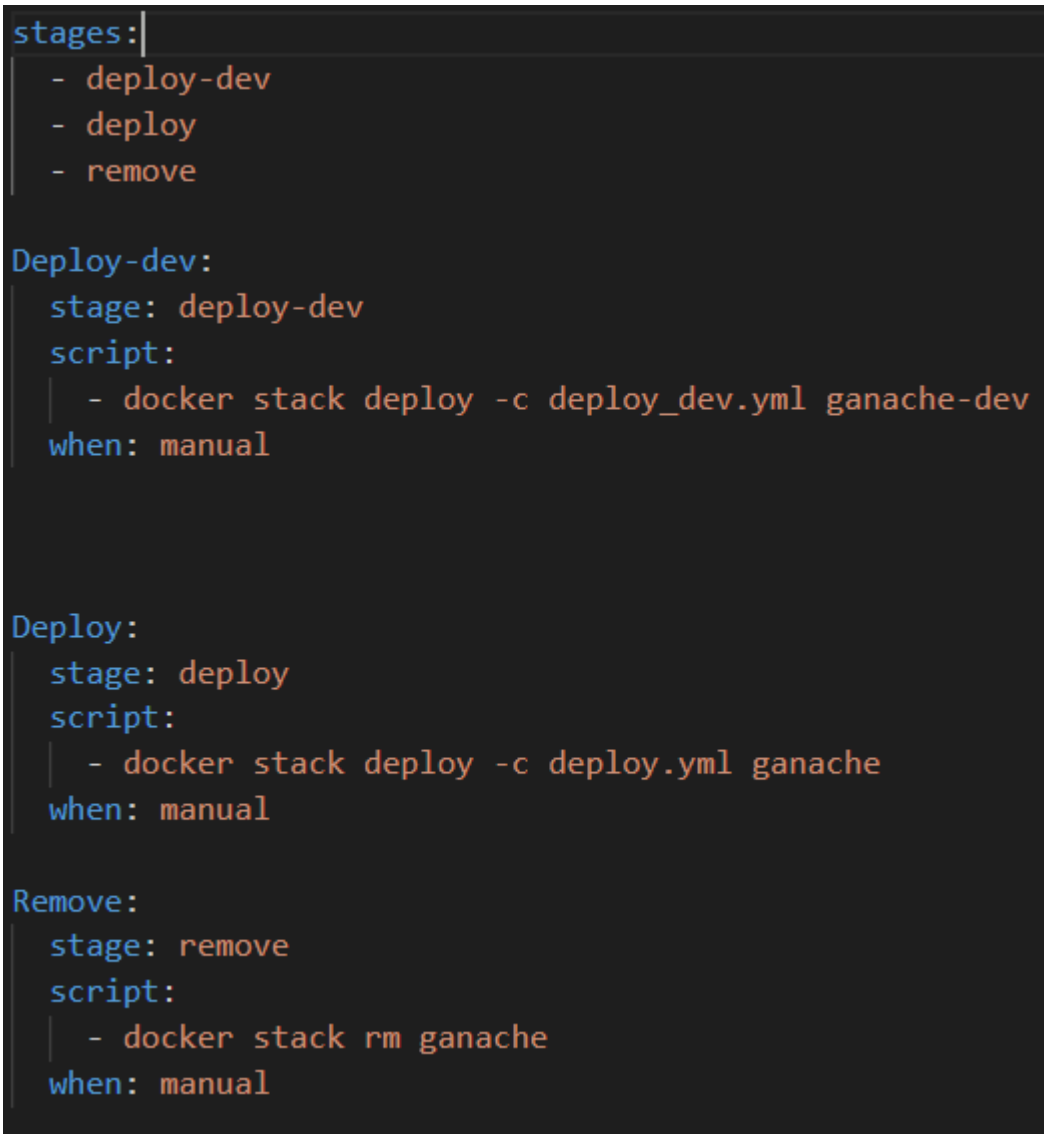
Una vez tengamos todo listo, podemos pasar a montar nuestros servicios en dockers. Para cada servicio necesitaremos (adjunto un ejemplo para cada uno):

- Docker compose:
  -
```
version: "3.7"
services:
  ganache-cli:
    image: trufflesuite/ganache-cli:latest
    ports:
      - target: 8546
        published: 8546
        protocol: tcp
    deploy:
      placement:
        constraints:
          - node.role == worker
```

Figure 17: Código servicios de docker

- Docker file en caso de querer modificar una imagen oficial
  -
```
FROM influxdb:1.7.9
COPY ./entrypoint.sh /entrypoint.sh
COPY ./init-influxdb.sh /init-influxdb.sh
RUN chmod +x /entrypoint.sh
RUN chmod +x /init-influxdb.sh
ENTRYPOINT ["/entrypoint.sh"]
CMD ["influxd"]
```

Figure 18: Docker Code

- Gitlab CI
  - 

```
stages:
  - deploy-dev
  - deploy
  - remove

Deploy-dev:
  stage: deploy-dev
  script:
    - docker stack deploy -c deploy_dev.yml ganache-dev
  when: manual



Deploy:
  stage: deploy
  script:
    - docker stack deploy -c deploy.yml ganache
  when: manual

Remove:
  stage: remove
  script:
    - docker stack rm ganache
  when: manual
```

Figure 19: GitLab CI

Una vez tengamos todo esto, simplemente hemos de subirlo a gitlab y en el apartado de CI/CD podremos desplegarlo

8. Testing

Para comprobar las diferentes funcionalidades de nuestra aplicación hemos hecho lo siguiente:

8.1 Configuración

Para poder usar esta aplicación es indispensable que el navegador pueda comunicarse con la blockchain, nosotros recomendamos el plugin para navegadores de Metamask (https://metamask.io/) con el cual convertimos nuestro navegador usual en un navegador Web3.0 que es capaz de conectarse a blockchains.
Este manual del usuario está basado en Metamask, pero si se usa otra cartera funcionaria de forma parecida.

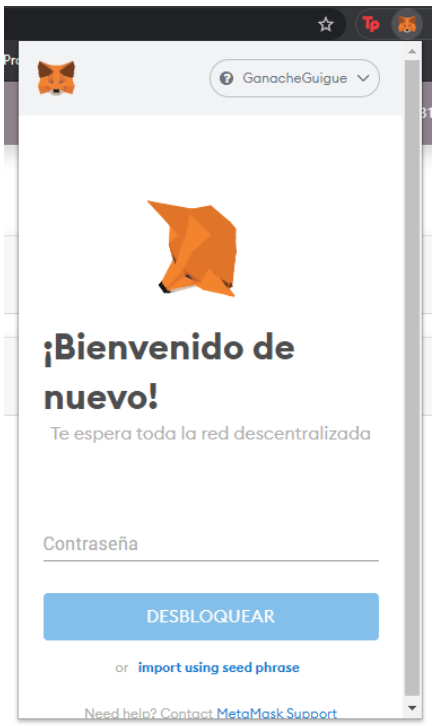


Figure 20: Cartera MetaMask

Una vez entremos a la web lo primero que veremos es una notificación de Metamask pidiéndonos que nos identifiquemos, si no lo hemos hecho antes tendremos que seguir los pasos que nos indica hasta crear una nueva cartera o importar una ya existente.

Una vez identificados en Metamask es importante asegurarse de que estamos conectados. Abriendo la extensión nos sale un desplegable como el que podemos ver en las figuras, indicando el estado de la conexión. Si vemos que no estamos conectados hacemos clic sobre “No conectado” y aparecerá un nuevo desplegable con todas nuestras carteras y la opción de conectarnos. Hacemos clic en “Conectar” y cerrando este desplegable se vuelve a mostrar el anterior pero ahora vemos que si estamos conectados.
Una vez completados estos pasos ya estamos preparados para empezar a interactuar con la web. Es importante saber que no necesitas dinero para poder usarla, simplemente una cartera.

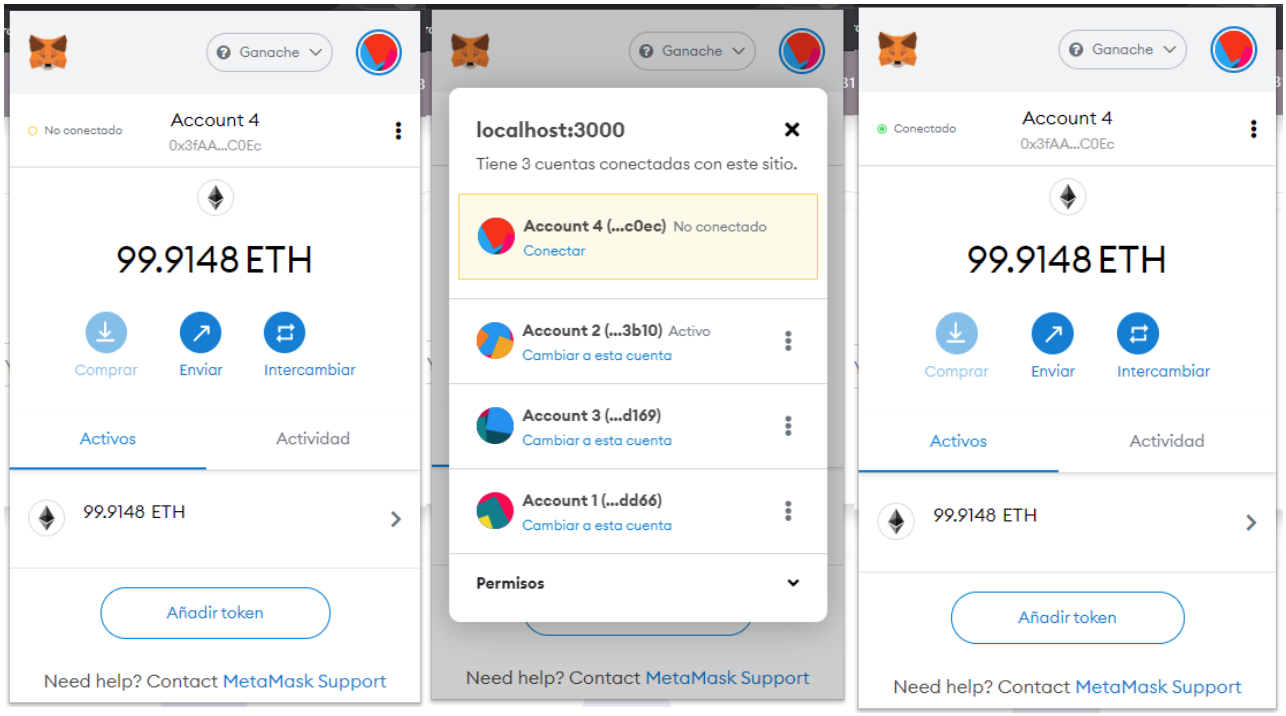


Figure 21: Funcionamiento de la cartera MetaMask

8.2 Funcionalidades

Esta web cuenta con una sola página con varios apartados. Al conectarnos podemos ver que la aplicación tiene una barra en la parte superior indicando la cartera del usuario, pero lo que llama la atención es el primer y principal componente que es el museo, donde puedes ver todas las obras expuestas ordenadas en sus colecciones y cada colección dentro del artista que la ha publicado. En la figura adjunta se puede apreciar el artista David, con sus colecciones HypeAF y Supercars y los NFT que componen la primera colección.

Podemos ver que si un NFT está a la venta aparece su precio, y sino es propiedad del usuario al lado un botón para comprarlo.

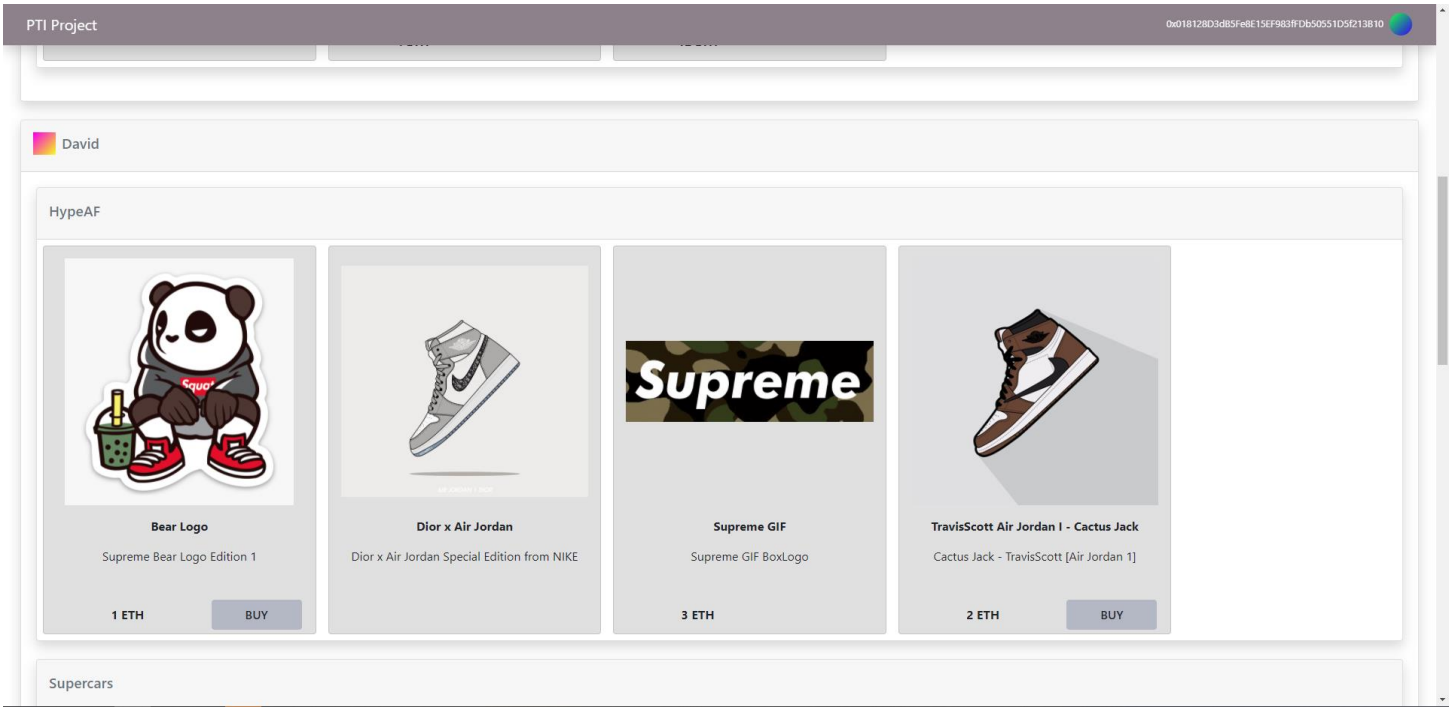


Figure 22: Pantalla principal del proyecto

Si bajamos al final de la página web podemos ver los otros componentes. Un apartado bajo la etiqueta “Owned NFTs” donde podemos ver los NFT que son nuestros, con todos sus campos y un campo input donde podemos fijar el precio de venta si estamos interesados en vender y el botón para confirmar este precio. No querer vender un NFT significa que su precio es 0, cuando esta condición se cumple no aparece la opción de comprarlo en el museo.

A la derecha podemos ver dos formularios, el primero sirve para asociar un nombre de artista a la cuenta con la que vamos a subir los NFT, en la figura adjunta podemos ver que el usuario tiene el nombre puesto de “Campa”.

El segundo formulario es el que permite publicar un NFT, se debe escoger el contenido multimedia que se le quiere asociar, si todo va bien el botón se pondrá de color verde y aparecerá el nombre del fichero seleccionado.También se debe indicar

los atributos de este NFT: título, colección, descripción y precio. Cabe recordar que poner el precio a 0 es lo mismo que no ponerlo a la venta.

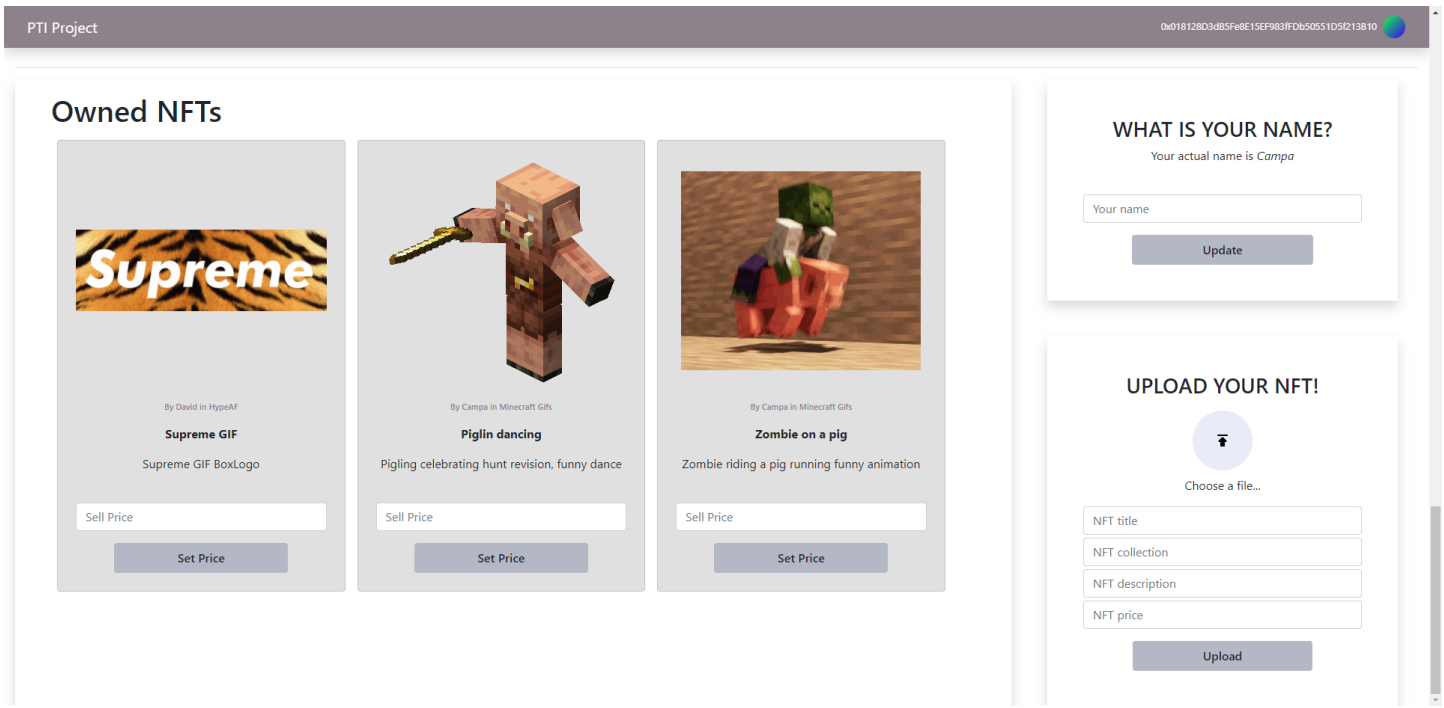


Figure 23: NFT en propiedad + opción del cambio de nombre y posibilidad de añadir nuevo NFT

Otro punto importante es que cualquier acción que se haga se debe confirmar, por la seguridad de la tecnología blockchain, para que nadie pueda operar con tu número de cartera sin tu consentimiento. Es por esta razón que cada vez que subamos un NFT, actualizemos su precio, lo compramos o nos cambiemos el nombre aparecerá un desplegable de Metamask preguntando si quieres confirmar la transacción y dando información de esta, como el dinero que

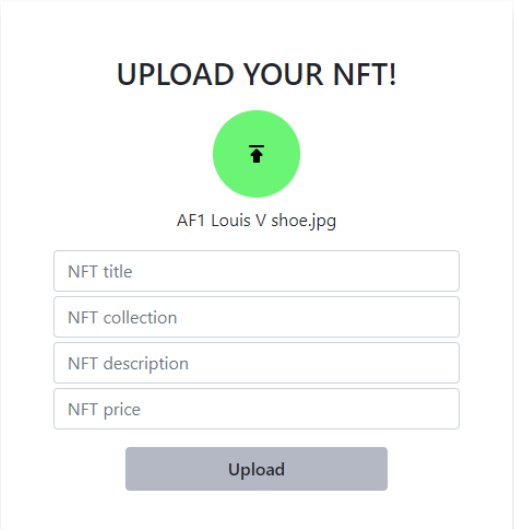


Figure 24: Función para subir un NFT

estamos enviando y el precio que estamos dispuestos a pagar para que un minero confirme nuestra transacción (Gas).

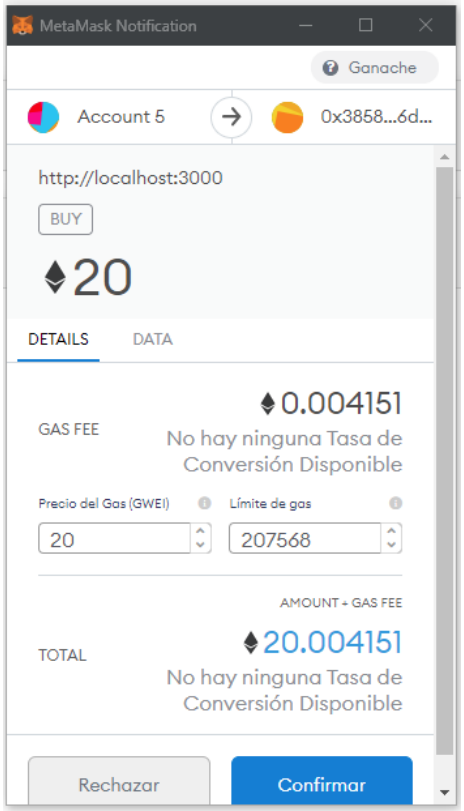


Figure 26: Operación de compra en MetaMask

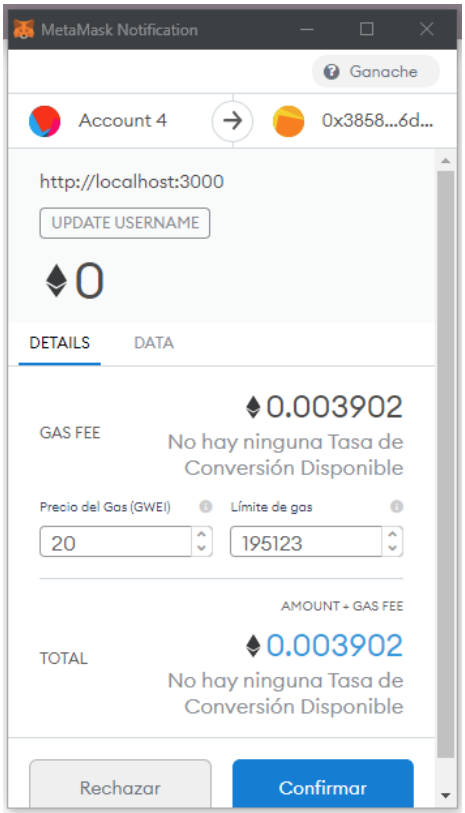


Figure 25: Transacción en MetaMask

9. Deploy de versiones

Para desplegar nuevos cambios en el proyecto hay que seguir estos pasos:

Primero hacer los cambios en el código y hacer push:

```
git add .; git commit -m "New feature"; git push
```

Figure 27: Añadir entradas en GitLab

.

Después hay que ir al repositorio del código e irte al apartado de CI/CD:

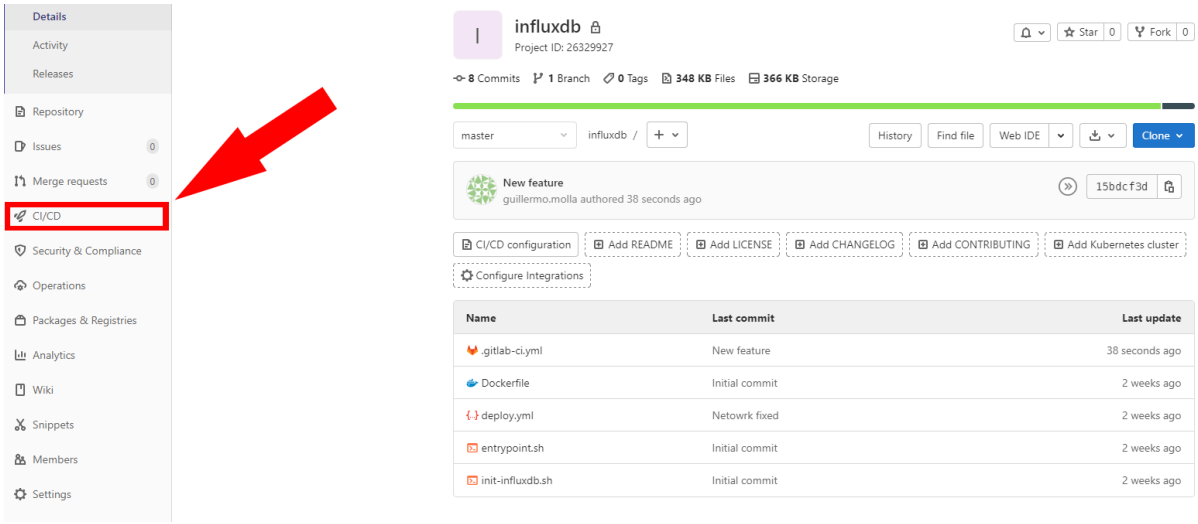


Figure 28: Interfaz de GitLab

Al tener un archivo ".gitlab-ci.yml" Gitlab creara una pipeline automaticamente, simplemente hay que ir al "Job" que queramos ejecutar y darle al play:

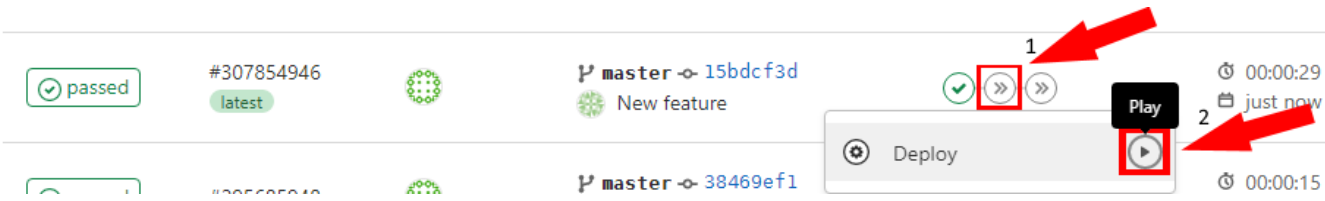


Figure 29: Deploy desde GitLab

Si entramos dentro del job dandole en su nombre, podríamos ver su ejecución:

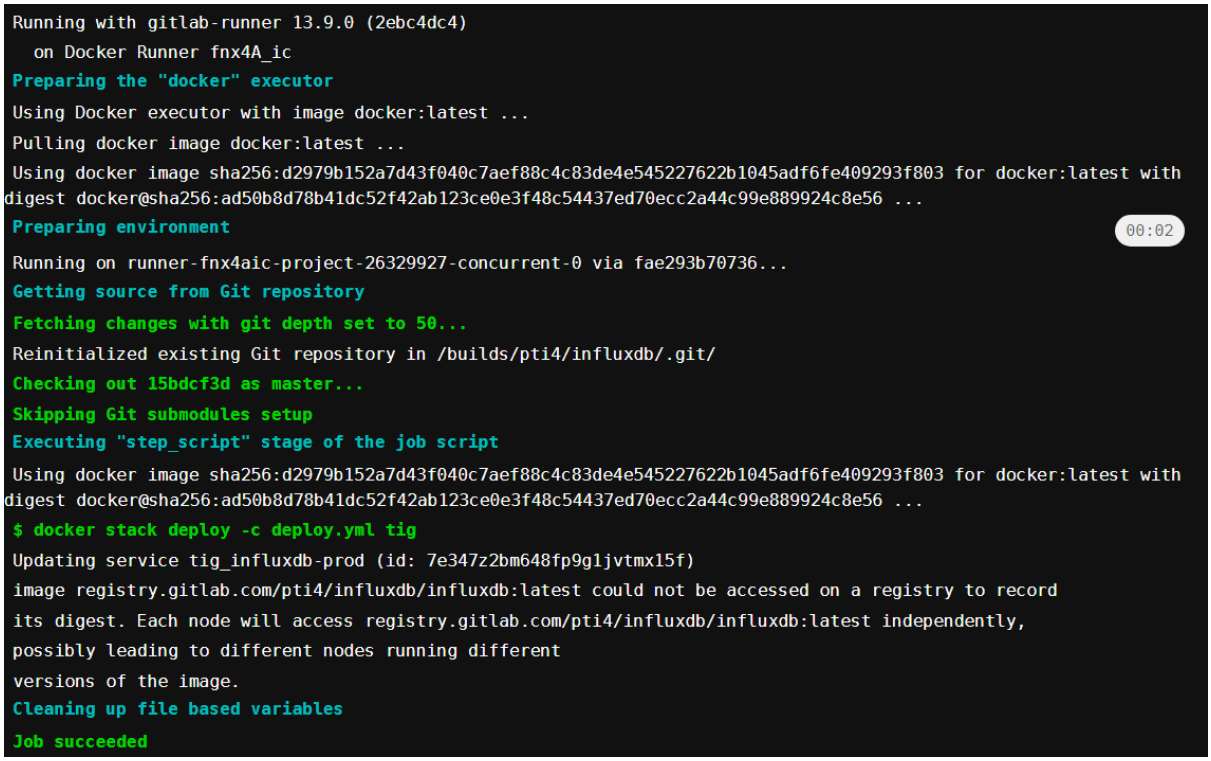


Figure 30: Código resultante en GitLab

Al recibir un “Job succeded” podemos asegurar que el job ha acabado su ejecución correctamente

10. Futuro del proyecto

Este proyecto no ha alcanzado su máximo nivel de desarollo, si hubiera más tiempo para trabajar en él, nos gustaría crear una nueva base de datos para mejorar la experiència de usuario, ya que el backend hecho en su totalidad en blockchain es lento. La web podría leer esta base de datos, subir los NFT al Smart Contract y crear un proceso en el servidor que vaya validando la nueva base de datos ya que esta sí sería hackeable.

Otro punto sería dockerizar la web y lanzarla desde gitlab con la plantilla de CI que ya tenemos preparada, pero no somos capaces de dockerizarla por falta de recursos en el servidor donde hemos montado la plataforma.

Una vez finalizado todo este proyecto, el último paso sería deployear el smart contract a la red de Etherum para ofrecer todas las funcionalidades que hemos desarrollado al público. Los artistas de distintas partes del mundo podrán empezar a exponer sus obras en nuestro portal, a la vez que los visitantes tendrán opción a disfrutar del arte y en algunos casos la posibilidad de comprarlo si el autor lo permite.

11. Conclusiones

Al finalizar este proyecto podemos asegurar que el conjunto de tecnologías escogido para desarrollar tanto la web como el Smart Contract han sido un completo acierto, por la gran documentación que tienen que facilita enormemente el desarollo, la gran comunidad que hay desarrollando aplicaciones descentralizadas y que nos ha ayudado siempre que lo hemos necesitado y sobretodo por lo bien que funcionan las unas con las otras.

Este Smart Contract se basa en la creación de NFT usando la implementación de openzeppelin, queremos destacar una vez más el gran futuro de este token que nosotros usamos para asegurar la autoría y el traspaso seguro entre comprador y vendedor de obras de arte, pero que tiene grandes aplicaciones que van desde nuevos sistemas de identificación hasta registros médicos únicos para cada paciente y disponibles por todos los centros sin comprometer la privacidad de los usuarios.

Con nuestra aplicación hemos conseguido que artistas sin el menor conocimiento de informática puedan crear sus NFT y compartir sus obras de forma segura por internet haciendo el arte más accesible a la gente y facilitando a los artistas que si arte llegue a más gente.

Hemos visto que Docker Swarm nos da una capa de HA (High Availability) bastante potente ya que en servidores “baremetal” montar un servicio en HA solido es muy complicado y en este caso fue casi automático.

Uno de los objetivos de este trabajo era comprobar la importancia de la figura del DevOps en el desarrollo de una aplicación descentralizada, y rápidamente vimos que es igual de importante que en cualquier otro proyecto porque sigue teniendo la capacidad de proporcionar comodidades a los desarrolladores.

.

12. Bibliografía

https://coolors.co/

https://docs.docker.com/

https://www.expansion.com/juridico/actualidad-tendencias/2020/01/16/5e1f57d3e5fdeaac348b4611.html

https://elingeniero.net/2021/03/15/que-es-un-nft-y-como-ganar-dinero/

https://www.forbes.com/advisor/investing/nft-non-fungible-token/

https://github.com/dappuniversity

https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/token/ERC721

https://docs.gitlab.com/runner/install/

https://ipfs.io/

https://industriamusical.es/por-que-los-artistas-ganan-tan-poco-por-streaming/

https://metamask.io/

https://nodejs.org/es/

https://openzeppelin.com/

https://opensea.io/

https://promocionmusical.es/streaming-musica-reparto-ingresos-generados/

https://rarible.com/

https://es.reactjs.org/

https://superrare.co/

https://www.theverge.com/22310188/nft-explainer-what-is-blockchain-crypto-art-faq

https://thenewstack.io/tutorial-create-a-docker-swarm-with-persistent-storage-using-glusterfs/

https://web3js.readthedocs.io/en/v1.3.4/

https://www.youtube.com/channel/UCY0xL8V6NzzFcwzHCgB8orQ

13. Anexos

Función render() en App.js

```
    render() {
        return (
            <div>
                <Navbar account={this.state.account} />
                { this.state.loading
                { this.state.loading
                    ? <div id="loader" className="loading-text text-
center"><p>Loading...</p></div>
                    : <Main
                        account={this.state.account}
                        museum={this.state.museum}
                        ownedNft={this.state.ownedNft}
                        nft={this.state.nft}
                        name={this.state.name}
                        fileName={this.state.fileName}
                        captureFile={this.captureFile}
                        uploadImage={this.uploadImage}
                        updateName={this.updateName}
                        buyNFT={this.buyNFT}
                        updatePrice={this.updatePrice}
                    />
                }
            </div>
        );
    }
```