UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Second Hand Chain
Documentación Técnica

Presentado por Áxel Rubio González
en Universidad de Burgos — 4 de julio de 2023
Tutor: Jesús Manuel Maudes Raedo
Co-tutor: Sandra Rodríguez Arribas
Tutor de Empresa: Julia Zuara Jiménez

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este capítulo se va a exponer cómo se ha planeado el proyecto y cómo se ha llevado a cabo dentro de cada uno de los apartados.

A.2. Planificación temporal

La gestión de este proyecto se ha realizado empleando la metodología *Scrum*. Por ello, se celebraba con los tutores el *sprint review*, una reunión para revisar todo el trabajo realizado y resolver posibles dudas, esta era realizada con una frecuencia de dos semanas, coincidiendo con el final e inicio de *sprint* .

Para la clasificación de *issues* se ha empleado un sistema que los divide por las siguientes etiquetas para poder facilitar mejor su identificación:

1. *Backend*: Empleada para *issues* relacionadas con el *backend*, predomina durante la primera mitad del proyecto, fase en la que se estaban creando los contratos digitales.

2. *Blockchain*: Se ha usado para *issues* relacionadas con la tecnología *Blockchain*, predomina durante la primera mitad del proyecto, suele ir asociada a la de *Backend*.

3. *Bug*: Al final del proyecto se pudieron detectar varios pequeños errores en el *Frontend*. Para dar más visibilidad a esas tareas se creó esta etiqueta.

4. *Frontend*: Durante la segunda mitad del proyecto el desarrollo se centra en el *Frontend* por lo que es cuando comienza a tomar protagonismo.
5. *Management*: Etiqueta relacionada con actividades de la gestión de tareas.
6. *Research*: Para tareas que incluyen investigación para conocer cómo realizar dicha tarea.
7. *Software engineering*: Para las actividades realizadas durante las fases de diseño inicial de la aplicación.

Sprint 1: Inicio del proyecto

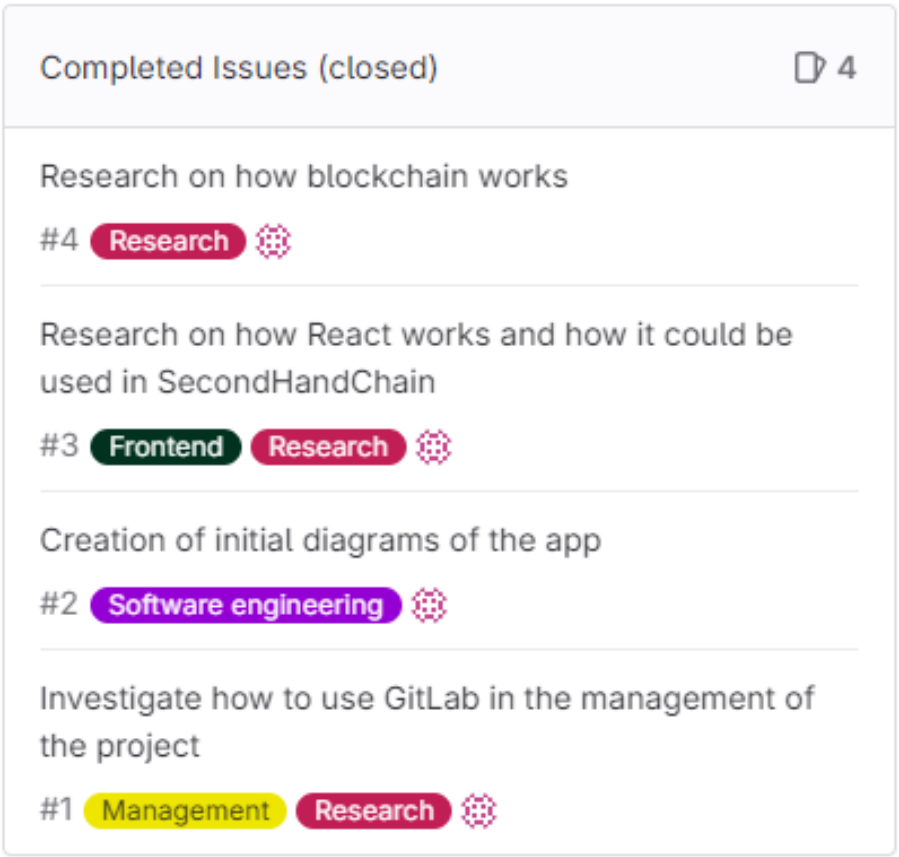

Figura A.1: Tareas finalizadas en el Sprint 1

En este primer sprint transcurrió entre el 24-02-2023 y el 10-03-2023 se realizo una primera toma de contacto con el proyecto, consistió en investigar

el funcionamiento de *Blockchain*. Además, se comenzó a aprender a usar *GitLab* para realizar la gestión de tareas durante el proyecto. Finalmente, se realizaron unos primeros diagramas UML a partir de los requisitos ideados en la reunión incial del proyecto. Podemos ver todas estas tareas con sus etiquetas en la figura A.1.

Sprint 2: Configuraciones iniciales y primera iteración del *smart contract*

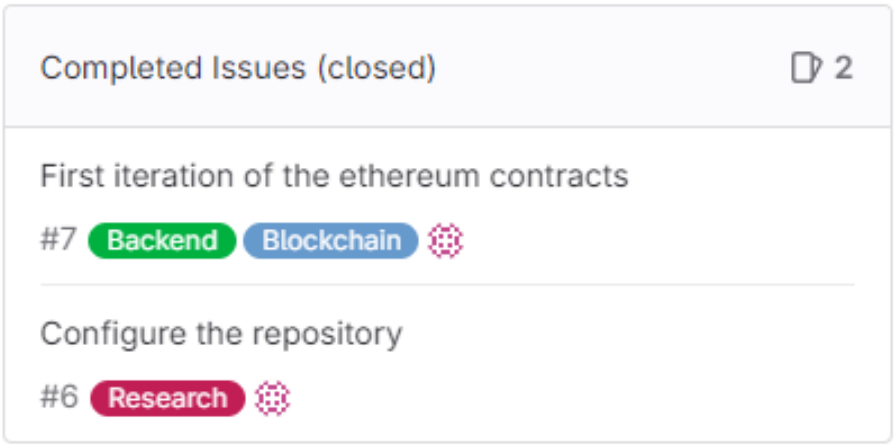

Figura A.2: Tareas finalizadas en el Sprint 2

En este segundo *sprint*, el cual transcurrió entre el 11-03-2023 y el 24-03-2023, se comenzó a configurar el repositorio de GitLab, al cual solo se podía acceder mediante clave SSH, por ello tuve que configurar una y después resolver unos cuantos problemas que me daban las cuentas de *git*.

A partir de entonces, ya con el repositorio configurado, comencé a formarme en *Solidity*, un lenguaje que debido a las características de *Blockchain* tiene una serie de particularidades que deben ser bien comprendidas para empezar a emplearlo. Después de realizar esta tarea, se creó un primer *smart contract* que permitía el registro de teléfonos y definía inicialmente cómo se iban a almacenar estos datos.

Sprint 3: Adaptación del *smart contract* al estándar *ERC721*

Transcurrido entre el 25-03-2023 y el 14-04-2023 se continuó con el desarrollo del *smart contract*, se implementaron nuevos *getters*, proceso que

es distinto al empleado en otros lenguajes ya que se debe guardar en en local el *array* para estos métodos *view* (sin coste al solo consultar información) y por ello debes almacenar el número de objetos que vas a añadir a esa lista que debe conocer de antemano el tamaño que va a tener al instanciarla, no es dinámica como los *arrays* en *storage*.

Figura A.3: Tareas finalizadas en el Sprint 3

Además, aunque de esto no se pudo hacer *commit* al detectar un error que se tuvo que preguntar a los tutores, se comenzó con la implementación de la interfaz *IERC721*, el estándar para gestionar traspasos de *NFTs*.

Sprint 4: Continuación de la adaptación al estándar *ERC721*

Entre el 15-04-2023 y el 21-04-2023, este sprint fue solo de una semana (el anterior de tres) por problemas de horarios por la semana santa, se resolvió el error descrito anteriormente que se debía a un problema con las versiones del compilador de *Solidity*.

Figura A.4: Tareas finalizadas en el Sprint 4

Además, se finalizó del desarrollo de las características básicas necesarias a implementar en los *smart contract*. Para la implementación del estándar *ERC721* se crearon otros métodos que no se iban a usar en Second Hand Chain, principalmente los correspondientes a dar permisos a terceros para transferir el token, para dejar la puerta abierta en un futuro a su uso por otras aplicaciones.

Hasta esta fase, se había usado la consola de *Truffle* y *Ganache* para comprobar el correcto funcionamiento de estos *smart contract*, es decir, funcionaban correctamente n el entorno de desarrollo local.

Sprint 5: Inicio del desarrollo del *frontend*

En este sprint, entre el 22-04-2023 y el 5-05-2023, se desarrolla todo lo relacionado con el despliegue del *backend* en *Sepolia* y el inicio del *frontend*.

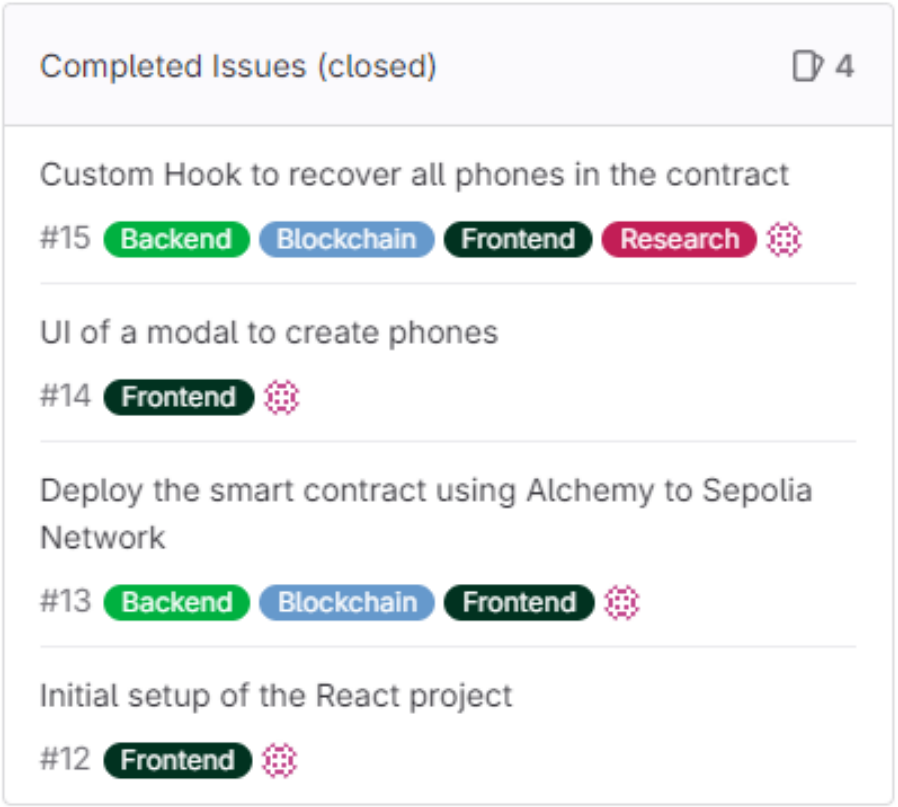

Figura A.5: Tareas finalizadas en el Sprint 5

Se toma la decisión de migrar en este momento el contrato a la *testnet* para facilitar el desarrollo y evitar posibles problemas de integración si el despliegue se produce con el *frontend* ya terminado.

En consecuencia, se inicia el proyecto de *React*, en el cual se empieza implementando un *custom hook* para obtener los móviles y se empieza con el diseño de un modal para crear teléfonos.

Sprint 6: Continuamos con el *frontend*

En el sprint que transcurre entre el 6-05-2023 y el 26-05-2023 se continúa con el desarrollo del *frontend*, tras el cual se va a permitir que los usuarios puedan consultar los teléfonos a la venta y los teléfonos propios, siempre y cuando se tenga *wallet* y teléfonos registrados.

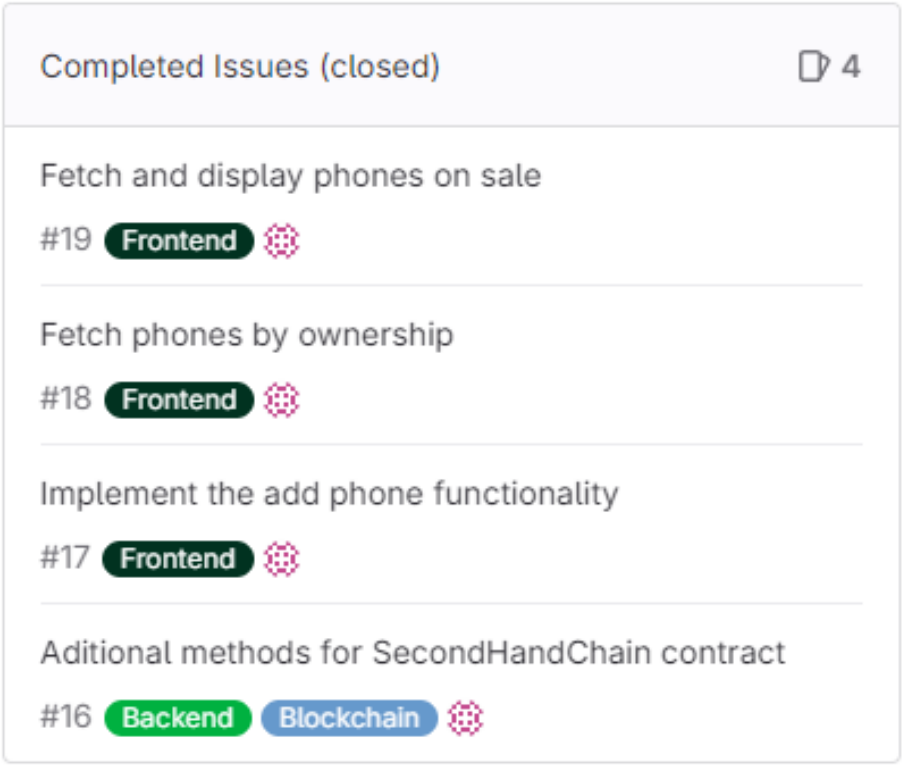

Figura A.6: Tareas finalizadas en el Sprint 6

El otro tipo de usuario, el que no dispone de *wallet*, podrá consultar los teléfonos a la venta aunque no podrá ni crear ni comprar.

Sprint 7: Llegamos al producto mínimo viable

En este sprint se alcanzó el producto mínimo viable, transcurrió ente el 27-05-2023 y el 9-06-2023. Se implementaron todas las funciones necesarias para cumplir todos los requisitos funcionales de la aplicación. Se implementó la compra venta de teléfonos, la posibilidad de actualizar el precio de los productos que están ya a la venta y retirar del mercado. Además se implementó un nuevo modal que permite a los usuarios controlar todas estas funciones anteriores por teléfono y consultar el historial de precios y ventas previas.

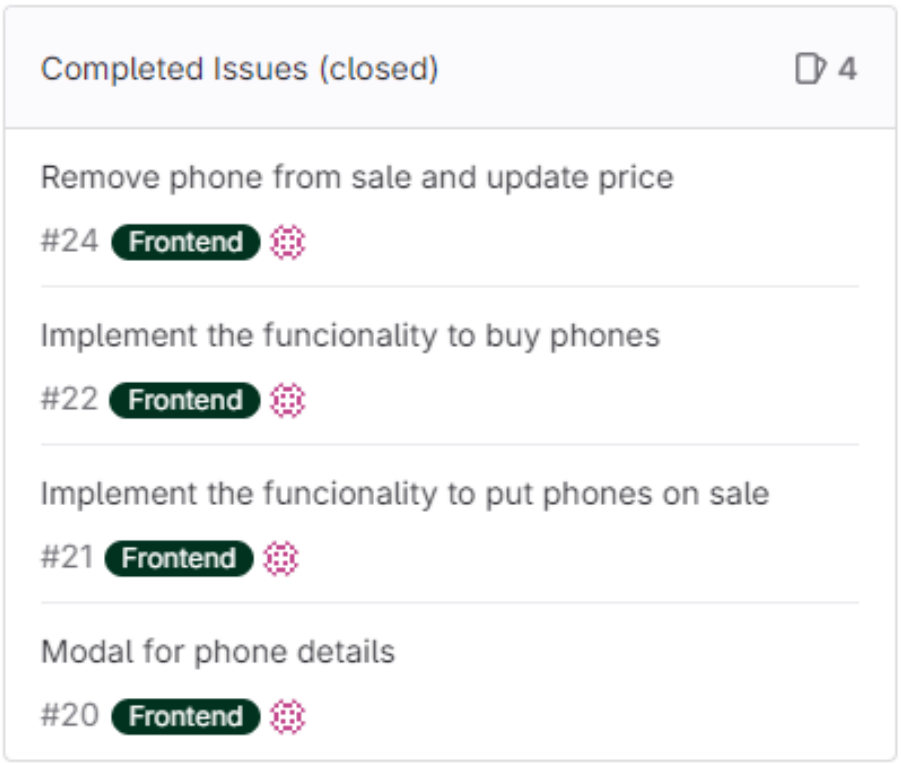

Figura A.7: Tareas finalizadas en el Sprint 7

En consecuencia, se produce el primer despliegue de la aplicación en *Vercel* para comprobar su correcto funcionamiento una vez ya desplegada.

Sprint 8: Grandes Mejoras a nivel de *UX* y nuevas características

En este sprint final transcurrido entre el 10-06-2023 y el 23-06-2023, se decide añadir a la aplicación la posibilidad de registrar imágenes de los teléfonos, ese añadido se realiza respetando las ventajas de *Blockchain*. Por ello este sprint está focalizado principalmente en esta característica que requiere de la adaptación del contrato para poder registrar el CID y en la adaptación del *frontend* para esta característica.

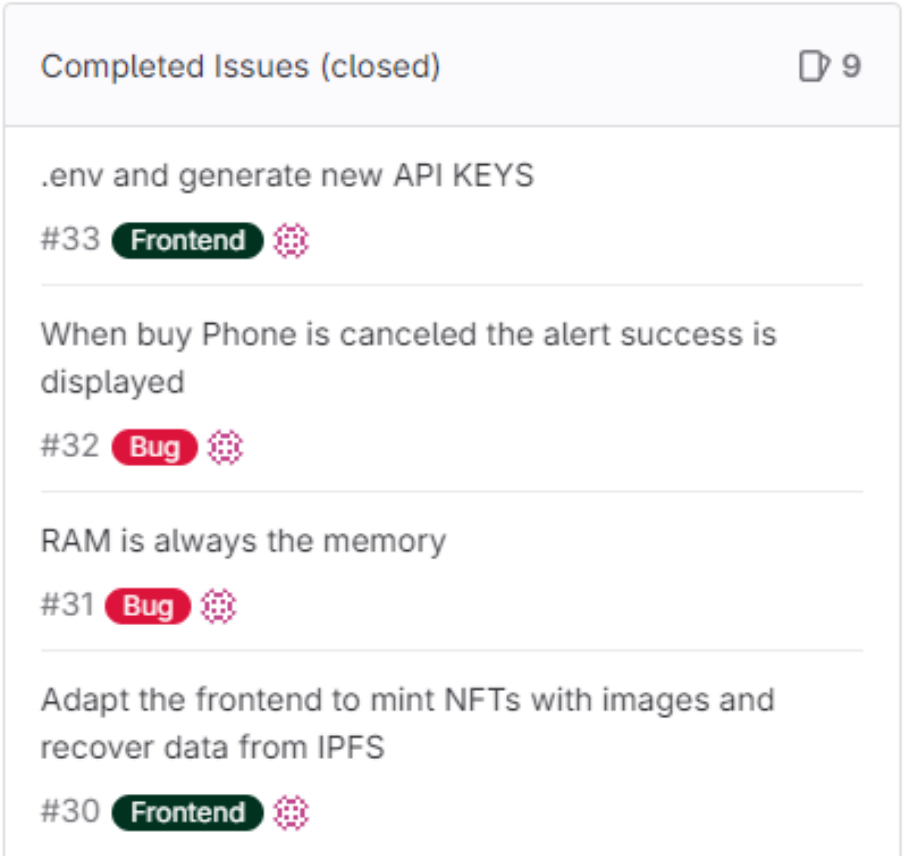

Figura A.8: Tareas finalizadas en el Sprint 8 (Parte 1)

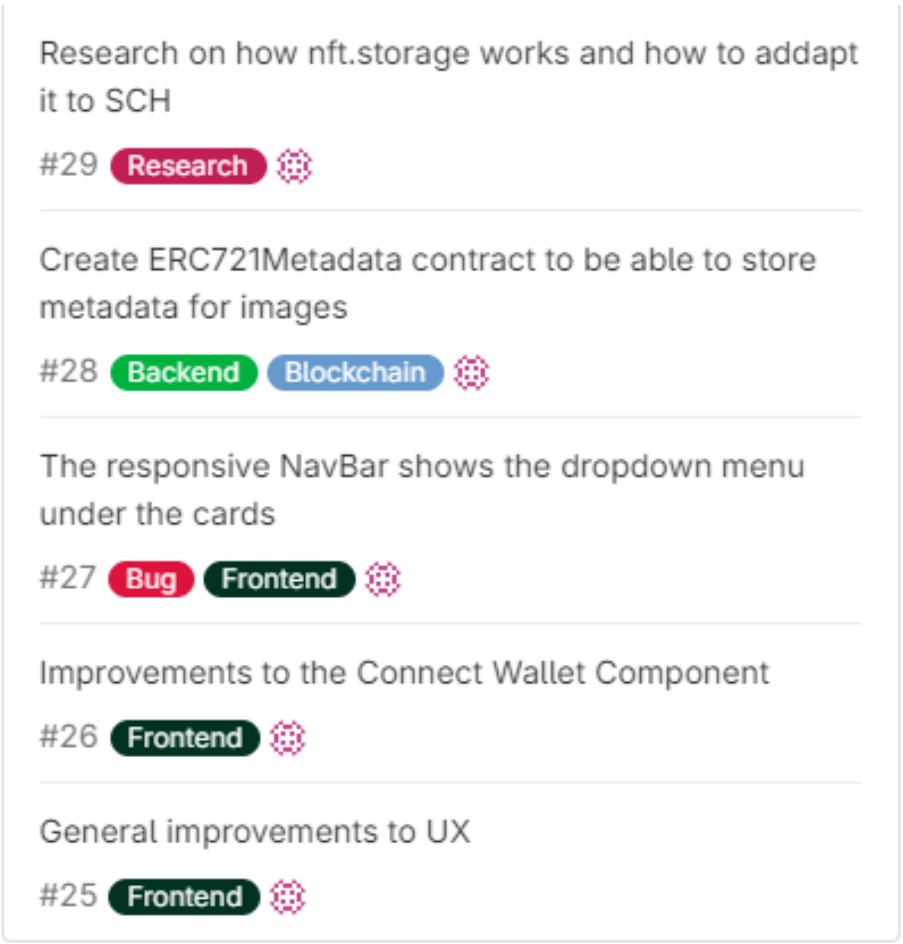

Figura A.9: Tareas finalizadas en el Sprint 8 (Parte 2)

Se mejora el funcionamiento del botón de la parte superior derecha que indica el estado de conexión de la aplicación con la *wallet*, haciéndolo mucho más comprensible para el usuario, de esta forma ahora el usuario conoce en cualquier momento si tiene *Metamask* instalado, si lo tiene pero debe iniciar sesión o si ya está conectado.

Surgieron una serie de *bugs*, principalmente estéticos por actualizar *DaisyUi* a una nueva versión que había salido en Junio que tuvieron que ser resueltos y otros fallos menores.

Finalmente, se realizaron mejoras para garantizar la seguridad de la aplicación, se creó un fichero *.env* para garantizar la seguridad de las distintas *API KEYS*, además estas claves fueron regeneradas. La aplicación se volvió a desplegar y se creo un repositorio paralelo al original en *GitHub* para poder hacer público el proyecto.

A.3. Estudio de viabilidad

En esta sección se va a estudiar la viabilidad tanto económica como legal del proyecto, para ello, se va a hacer un repaso de las tecnologías empleadas para ver que costes conlleva y las licencias empleadas.

Viabilidad económica

En este apartado se estudiarán los costes económicos del proyecto y posibles formas de monetizar el desarrollo.

Costes del proyecto

En primer lugar vamos a considerar los costes del proyecto relacionados con el personal:

- En total se han empleado alrededor de 250 horas para realizar el proyecto, entre realización de las tareas en sí e investigación, ello da una media de 14 horas a la semana.
- Las principales tecnologías empleadas han sido *Solidity* y *React*, sus sueldos son respectivamente de 70.368 USD[1] y 69.480 USD, para estimar un sueldo para nuestro desarrollador vamos a calcular la media de ambas para estimar un sueldo medio que quiera aceptar nuestro

[1] United States Dolar

desarrollador, serán 69.924$. Estos sueldos han sido extraídos de la encuesta anual de *Stack Overflow*[2]. El sueldo bruto anual en euros será de 64.053 euros a la tasa de cambio actual por la que 1 USD son 0,92 euros. Esto nos daría como resultado unos 35 euros por hora brutos a 1820 horas de trabajo anuales (40 semanales).

$$14\frac{horas}{1semana} \times 35\frac{euros}{1hora} \times 4\frac{semanas}{1mes} = 1960euros\ \ al\ \ mes$$

- A continuación, debemos pagar los impuestos para calcular el coste total del trabajador, datos extraídos de [1]:
  - FOGASA: 0,2 %
  - Desempleo: 5,5 %
  - Formación: 0,6 %
  - Contingencias: 23,6 %

$$\frac{1960\frac{euros}{1mes}}{1-(0{,}002+0{,}055+0{,}006+0{,}236)} = 2796\ \ al\ \ mes$$

Además, se deben añadir los costes del equipo informático y su software:

- Un ordenador de sobremesa con un coste total de unos 2200 euros después diversas mejoras, aunque el original es de hace 7 años, actualmente tiene las siguientes características:
  - Procesador: AMD Ryzen 5 5600X 6 núcleos.
  - Gráfica: Nvidia GTX 1070.
  - RAM: 16GB DDR4.
  - SSD: 750GB.
  - HD: 2Tb.

$$\frac{2200\ \ euros}{7\ \ años} = 314\ \ euros\ \ al\ \ año$$

$$314euros\frac{4\ \ meses}{12\ \ meses} = 104euros\ \ en\ \ total\ \ por\ \ hardware$$

- De software solo se ha pagado Windows 10, hace 7 años también:

$$\frac{130\ euros}{7\ años} = 19\ euros\ al\ año$$

$$19euros\frac{4\ meses}{12\ meses} = 6{,}2euros\ en\ total\ por\ software$$

En consecuencia, podemos concluir que los costes totales del proyecto en estos cuatro meses han sido de:

- Personal: 11184 euros.
- Hardware: 104 euros.
- Software: 6,2 euros.
- Otros (electricidad y conexión a Internet): 130 euros.

En total suma unos 11424 euros.

Costes del proyecto

Actualmente este proyecto al no haber sido desplegado en la red principal y además al ser distribuido con licencia MIT no va a recaudar dinero. No obstante, si lo deseáramos, dispondríamos de las siguientes alternativas:

- Cobrar una tasa por transacción a nivel de contratos. El contrato enviaría esta tasa a la *wallet* definida. Además, los usuarios podrían consultar el funcionamiento esta tasa a nivel de código.
- De forma similar a la anterior, cobrar una tasa por usar el *frontend*.
- Proponer el proyecto como *open source* y financiarlo mediante una campaña de *crowdfunding*.

Viabilidad legal

Cómo se puede ver en la tabla A.1 todas las licencias permiten su uso para proyectos del tipo de Second Hand Chain de forma completamente libre.

La única particularidad sería la de *Metamask*, con cuya compañía se necesitaría llegar a un acuerdo para emplear su software en caso de que se usase para fines comerciales o se superase los 10000 usuarios mensuales, cifras que no vamos a alcanzar en ningún caso.

Herramienta	**Licencia**
React	MIT
Tailwinds css	MIT
Daisy UI	MIT
Solidity	GNU
GIT	GNU
NFT.Storage	MIT y Apache License 2.0
Truffle Suite	MIT
Web3.js	GNU
Vite	MIT
TypeScript	Apache License 2.0
Metamask	Non-Commercial Use*
Node.js	MIT
Ethereum	GNU

Tabla A.1: Licencias empleadas en el proyecto

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apéndice se van a listar los requisitos que satisface la aplicación que forma parte de este proyecto. Aunque los requisitos se definieron en la fase de diseño de la aplicación este listado ha sido ampliado según se desarrollaba la aplicación.

B.2. Objetivos generales

A continuación se adjunta un listado de los objetivos principales de la aplicación:

- Crear un *marketplace* que permita a los usuarios vender y comprar *NFT*.
- Desplegar los contratos en la *Blockchain Ethereum* o una de sus *testnets*.
- Aportar a los usuarios información que por sus características de inmutabilidad respaldada por *Ethereum* aporte valor a sus productos.
- Acercar las tecnologías *Blockchain* a un mayor numero de usuarios.
- Crear un aplicación web que aporte una experiencia lo más accesible posible a estas tecnologías *Blockchain*.

B.3. Catalogo de requisitos

En esta sección se incluye un listado con todos los requisitos que debe cumplir la aplicación:

Requisitos Funcionales

- **RF1-Gestión de *wallet***: El usuario deberá poder iniciar sesión en la aplicación desde la extensión de *Metamask* de su navegador, si tiene la *wallet* instalada le aparecerá directamente para iniciar sesión, si no, se mostrarán los correspondientes mensajes de aviso.

- **RF2-Gestión de usuario sin *wallet***: Un usuario que no cuente con *wallet* instalada podrá usar la aplicación para consultar teléfonos a la venta y características. Además de la *landing page* en la cual podrá consultar cómo añadir esta extensión si lo deseara.

- **RF3-Creación de teléfonos**: Se debe poder registrar los teléfonos en *Blockchain*, para ello el usuario deberá introducir todos sus datos además de una imagen. No se podrá registrar un teléfono con un IMEI ya registrado.

- **RF4-Gestión de teléfonos**: Los usuarios registrados deben poder ver los teléfonos que les pertenecen, además, podrán acceder al mercado de teléfonos en venta.

- **RF5-Gestión de ventas**: Los usuarios deben ser capaces de gestionar el estado en que se encuentren sus teléfonos, si se venden o no.

    - **RF5.1-Puesta a la venta**: Los usuarios pueden poner a la venta los teléfonos previamente registrados.
    - **RF5.2-Cambio de precio**: Dado un teléfono que está en venta su propietario podrá cambiar su precio de venta.
    - **RF5.2-Retirada de la venta**: Dado un teléfono que está en venta su propietario podrá retirarlo de la venta.

- **RF6-Seguimiento de transacciones por *Etherscan***: Se requiere que por cada por cada operación que implique realizar una transacción en *Blockchain*, el usuario, tras firmar dicha transacción, dispondrá de un enlace que le permitirá seguir su estado a través de *Etherscan*.

- **RF7-Compra de teléfonos**: Los usuarios deben poder comprar los teléfonos, para realizar esta operación aportarán el *Ether* correspondiente a la transacción , siempre y cuando este teléfono no les pertenezca ya.

Requisitos No Funcionales

- **RNF1-Seguridad**: Se requiere que la aplicación ofrezca un entorno seguro para el usuario y no suponga ningún riesgo para sus datos el uso de ella.
- **RNF2-Disponibilidad**: Se requiere que la aplicación esté disponible constantemente excepto en sus periodos de mantenimiento.
- **RNF3-Usabilidad**: Se requiere que la aplicación ofrezca la mejor experiencia posible al usuario. Además, es necesaria la correcta visualización de la aplicación desde teléfonos móviles, por lo tanto, los elementos en pantalla que sean necesarios se adaptarán dinámicamente para estas pantallas.
- **RNF4-Confiabilidad**: Se requiere que la aplicación cumpla todos los requisitos que se establecen en estos anexos.
- **RNF5-Eficiencia**: Se requiere que la aplicación haga un uso eficiente de los recursos de la máquina que lo vaya a ejecutar.
- **RNF6-Privacidad**: Se requiere que la aplicación haga un uso responsable de los datos de los usuarios, en este caso de su clave privada, dato que no debe ser conocido por nadie excepto el propio usuario.
- **RNF7-Mantenibilidad**: La aplicación debe aplicar las mejores prácticas posibles para facilitar el mantenimiento durante todo el ciclo de vida de la aplicación.

B.4. Especificación de requisitos

Diagrama de casos de uso

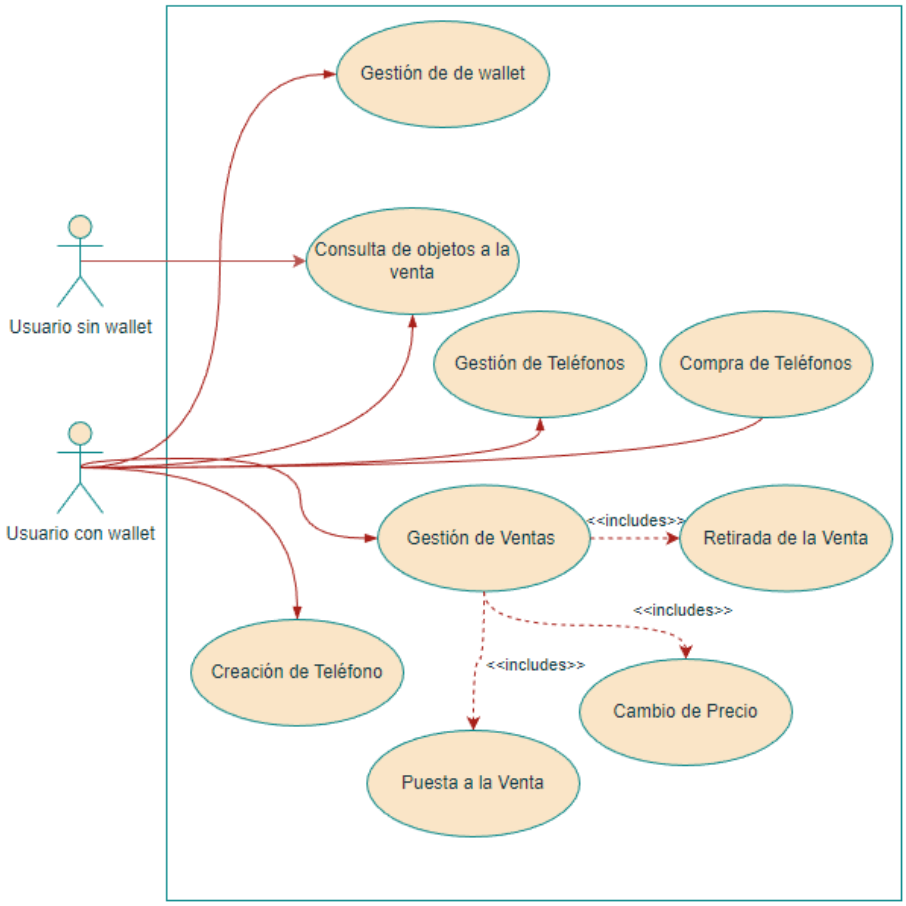

Figura B.1: Diagrama de casos de uso

CU-1	**Inicio de sesión en *Metamask***
Versión	1.0
Autor	Áxel Rubio González
Requisitos asociados	RF-1
Descripción	En este caso de uso el usuario iniciará sesión en *Metamask*
Precondición	El usuario no ha iniciado sesión en *Metamask* y tiene la extensión instalada
Acciones	1. El usuario accede a la web de Second Hand Chain. 2. Le aparece una ventana emergente para iniciar sesión en *Metamask*. 3. El usuario introduce su contraseña. 4. El usuario selecciona la wallet con la que quiere operar.
Postcondición	Que *Metamask* mantenga esa sesión iniciada
Excepciones	*Metamask* no se encuentra instalado o se ha cerrado la ventana emergente
Importancia	Alta

Tabla B.1: CU-1 Inicio de sesión en *Metamask*.

CU-2	**Usuario sin *Metamask* accede a los teléfonos en venta**
Versión	1.0
Autor	Áxel Rubio González
Requisitos asociados	RF2
Descripción	En este caso de uso el usuario sin *Metamask* consultará los teléfonos a la venta.
Precondición	El usuario no ha iniciado sesión en *Metamask* o no tiene la extensión instalada.
Acciones	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en teléfonos en venta. 3. Si está en pantalla pequeña clic en el *dropdown* y después en en teléfonos en venta. 4. El usuario puede visualizar los teléfonos a la venta. 5. El usuario accede a un teléfono y no podrá comprarlo, le aparece un mensaje diciendo que inicie sesión justamente en el botón.
Postcondición	Ninguna
Excepciones	Ninguna
Importancia	Alta

Tabla B.2: CU-2 Usuario sin *Metamask* accede a los teléfonos en venta.

CU-3	**Creación de teléfonos**
Versión	1.0
Autor	Áxel Rubio González
Requisitos asociados	RF3 y RF6
Descripción	En este caso de uso el usuario con *Metamask* registrará un teléfono
Precondición	El usuario ha iniciado sesión en *Metamask*
Acciones	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en crear teléfono. 3. Rellena los datos necesarios y selecciona un archivo. 4. Hace clic en crear teléfono. 5. La extensión de *Metamask* le solicita firmar la transacción, lo cual acepta. 6. Le aparece un mensaje para seguir la transacción en *Etherscan*.
Postcondición	El teléfono se registra correctamente en *Blockchain*
Excepciones	No se ha iniciado sesión, no hay fondos o no se han rellenado los datos
Importancia	Alta

Tabla B.3: CU-3 Creación de teléfonos.

CU-4	**Gestión de teléfonos**
Versión	1.0
Autor	Áxel Rubio González
Requisitos asociados	RF4
Descripción	En este caso de uso el usuario con *Metamask* consultará los datos de todos los teléfonos
Precondición	El usuario ha iniciado sesión en *Metamask*
Acciones	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en sus teléfonos. 3. Puede consultar los datos del teléfono. 4. Hace clic en teléfonos en venta. 5. Puede acceder a todos los teléfonos que se están vendiendo. 6. Consulta las características de cualquiera de ellos.
Postcondición	Ninguna
Excepciones	Ninguna
Importancia	Alta

Tabla B.4: CU-4 Gestión de teléfonos.

CU-5	**Puesta a la venta**
Versión	1.0
Autor	Áxel Rubio González
Requisitos asociados	RF51 y RF6
Descripción	En este caso de uso el usuario con *Metamask* podrá poner un teléfono a la venta
Precondición	El usuario ha iniciado sesión en *Metamask* y tiene teléfonos
Acciones	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en sus teléfonos. 3. Selecciona uno. 4. Define un precio de venta y hace clic en el botón. 5. La extensión de *Metamask* le solicita firmar la transacción, lo cual acepta. 6. Le aparece un mensaje para seguir la transacción en *Etherscan*.
Postcondición	La puesta a la venta se registra correctamente en *Blockchain*
Excepciones	No se ha iniciado sesión, no hay fondos o no se han rellenado los datos
Importancia	Alta

Tabla B.5: CU-5 Puesta a la venta de teléfono.

CU-6	**Cambio de Precio**
Versión	1.0
Autor	Áxel Rubio González
Requisitos asociados	RF52 y RF6
Descripción	En este caso de uso el usuario con *Metamask* podrá poner un teléfono a la venta
Precondición	El usuario ha iniciado sesión en *Metamask* y tiene teléfonos en venta
Acciones	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en sus teléfonos. 3. Selecciona uno en venta. 4. Define un precio de venta y hace clic en el botón. 5. La extensión de *Metamask* le solicita firmar la transacción, lo cual acepta. 6. Le aparece un mensaje para seguir la transacción en *Etherscan*.
Postcondición	El cambio de precio se registra correctamente en *Blockchain*
Excepciones	No se ha iniciado sesión, no hay fondos o no se han rellenado los datos
Importancia	Alta

Tabla B.6: CU-6 Cambio de Precio.

CU-7	**Retirar de la venta**
Versión	1.0
Autor	Áxel Rubio González
Requisitos asociados	RF53 y RF6
Descripción	En este caso de uso el usuario con *Metamask* podrá poner un teléfono a la venta
Precondición	El usuario ha iniciado sesión en *Metamask* y tiene teléfonos en venta
Acciones	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en sus teléfonos. 3. Selecciona uno en venta. 4. Se hace click en retirar. 5. La extensión de *Metamask* le solicita firmar la transacción, lo cual acepta. 6. Le aparece un mensaje para seguir la transacción en *Etherscan*.
Postcondición	La retirada de la venta se registra correctamente en *Blockchain*
Excepciones	No se ha iniciado sesión, no hay fondos o no se han rellenado los datos
Importancia	Alta

Tabla B.7: CU-7 Retirar de la venta.

CU-8	**Compra**
Versión	1.0
Autor	Áxel Rubio González
Requisitos asociados	RF7 y RF6
Descripción	En este caso de uso el usuario con *Metamask* comprará un teléfono
Precondición	El usuario ha iniciado sesión en *Metamask* y tiene fondos
Acciones	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en teléfonos en venta. 3. Selecciona uno en venta que no le pertenezca. 4. Se hace click en comprar. 5. La extensión de *Metamask* le solicita firmar la transacción, lo cual acepta. 6. Le aparece un mensaje para seguir la transacción en *Etherscan*.
Postcondición	La compra se registra correctamente en *Blockchain*
Excepciones	No se ha iniciado sesión o no hay fondos
Importancia	Alta

Tabla B.8: CU-8 Compra.

Apéndice C

Especificación de diseño

C.1. Introducción

En este apéndice se va a comentar cómo se ha llevado a cabo el diseño de la aplicación, por ello comentaré cómo se han ido llevando a cabo las distintas partes del diseño.

C.2. Diseño de datos

En primer lugar, el diseño de datos va a ser algo diferente a la forma de hacerlo en *Blockchain*. Por un lado, a bajo nivel el diseño de datos de *Blockchain* se ha explicado en el capítulo de la memoria relativo a conceptos teóricos.

Por ello, en esta sección me voy a centrar en el diseño de datos dentro de los *smart contract* que forman Second Hand Chain que se puede consultar en la figura C.1:

```
struct Phone {
    uint256 id;
    string model;
    string brand;
    string colour;
    uint16 ram;
    uint32 mem;
    address[] owners;
    uint[] saleTime;
    uint[] salePrice;
    uint price;
    string url;
}

Phone[] public phones;
uint256 numberOfPhonesOnSale;

mapping(uint => address) phoneOwner;
mapping(address => uint) phonesPerOwner;
mapping(string => uint) imeiPhoneId;
mapping(string => bool) isImeiRegistered;
mapping(uint => bool) isPhoneOnSale;
mapping(uint => address) phoneApproval;
mapping(address => address) approvalForAll;
```

Figura C.1: Conjunto de datos con los que operan los contratos

- Podemos ver que se compone de un *struct* que representa un teléfono.
- Cada teléfono registrado se va a ir guardando en una lista de teléfonos.
- Guardaremos un contador para el número de teléfonos en venta.
- En un mapa registraremos a quién pertenece cada teléfono.

- Además, guardaremos el número de teléfonos por dueño, este mapa es necesario por el patrón de diseño *memory array building*, explicado en la memoria.
- En otro mapa se lleva el recuento de los IMEI ya registrados.
- Existe otro mapa para registrar si un teléfono está o no a la venta.
- Por otro lado, los dos últimos mapas sirven respectivamente para der permisos a un tercero para transferir un token y para transferir todos los de una cuenta. Estos métodos, son necesarios para poder implementar los métodos encargados de gestionar los permisos de los *NFT* de la interfaz *IERC721*.

C.3. Diseño procedimental

A continuación se van a mostrar los distintos diagramas de secuencia correspondientes a Second Hand Chain:

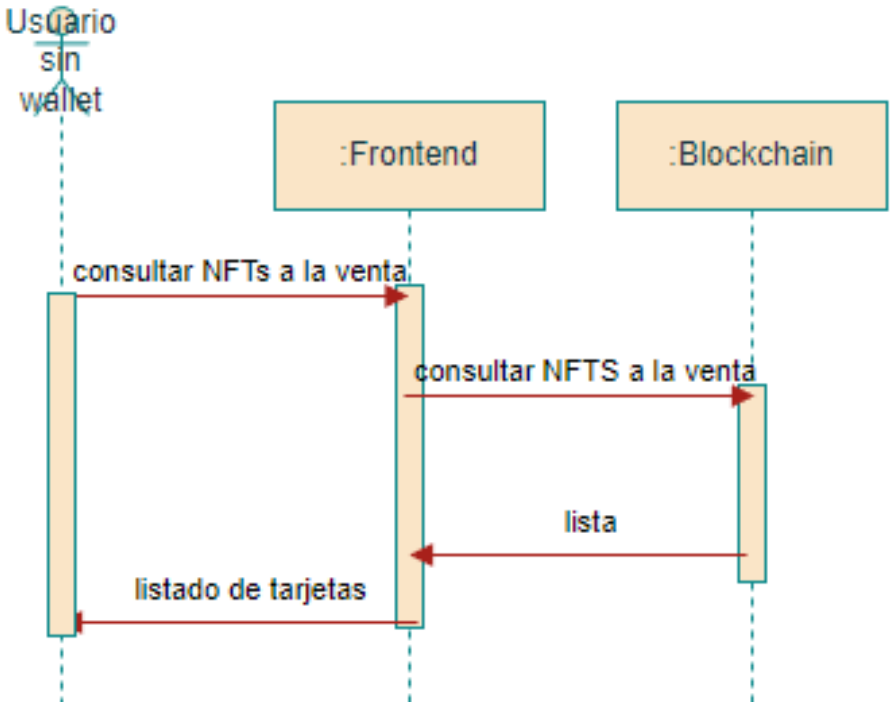

Figura C.2: Diagrama de secuencia de la consulta de teléfonos por parte de un usuario sin *wallet*

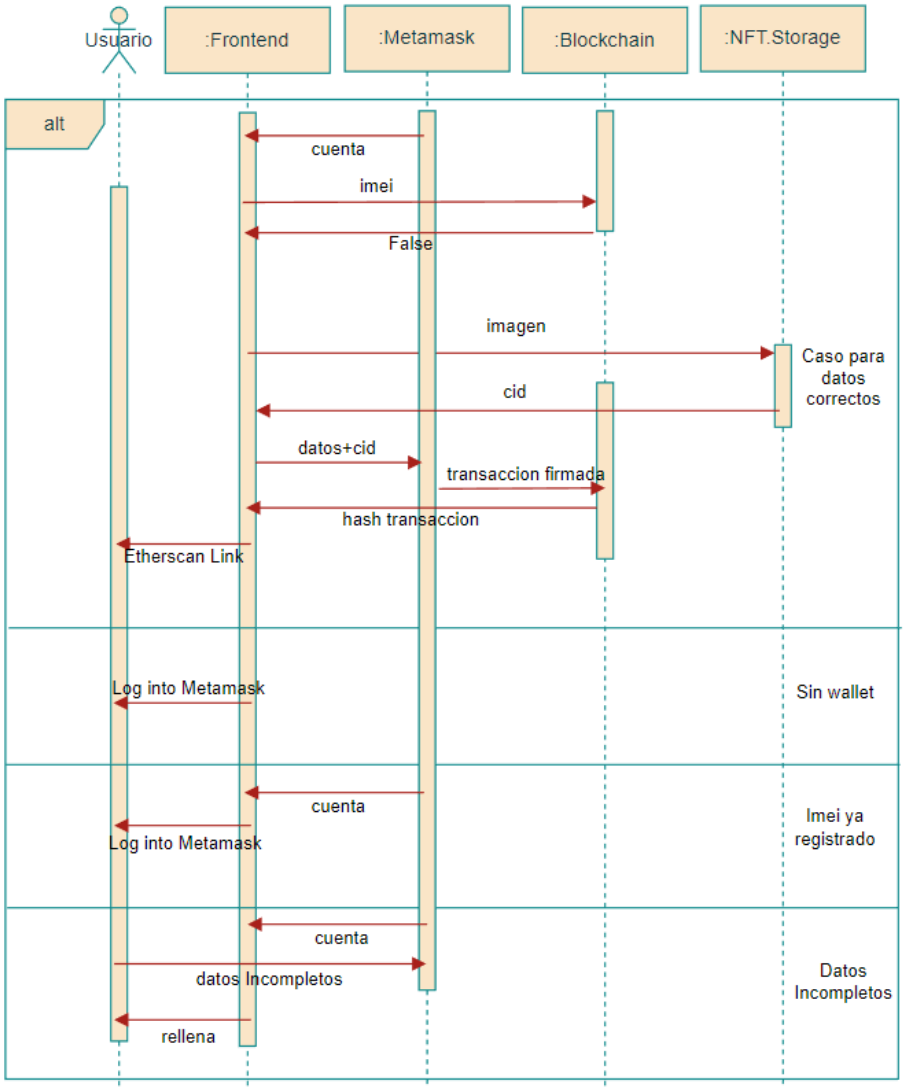

Figura C.3: Diagrama de secuencia de la creación de un teléfono

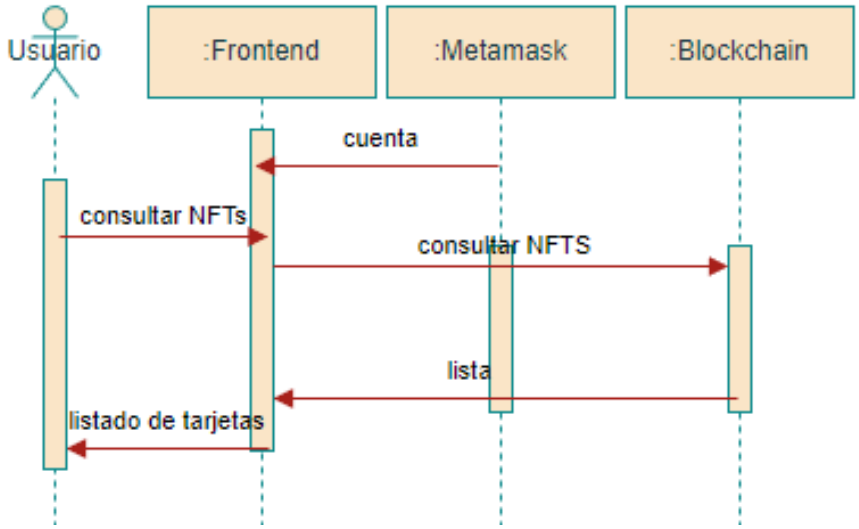

Figura C.4: Diagrama de secuencia sobre la consulta de teléfonos

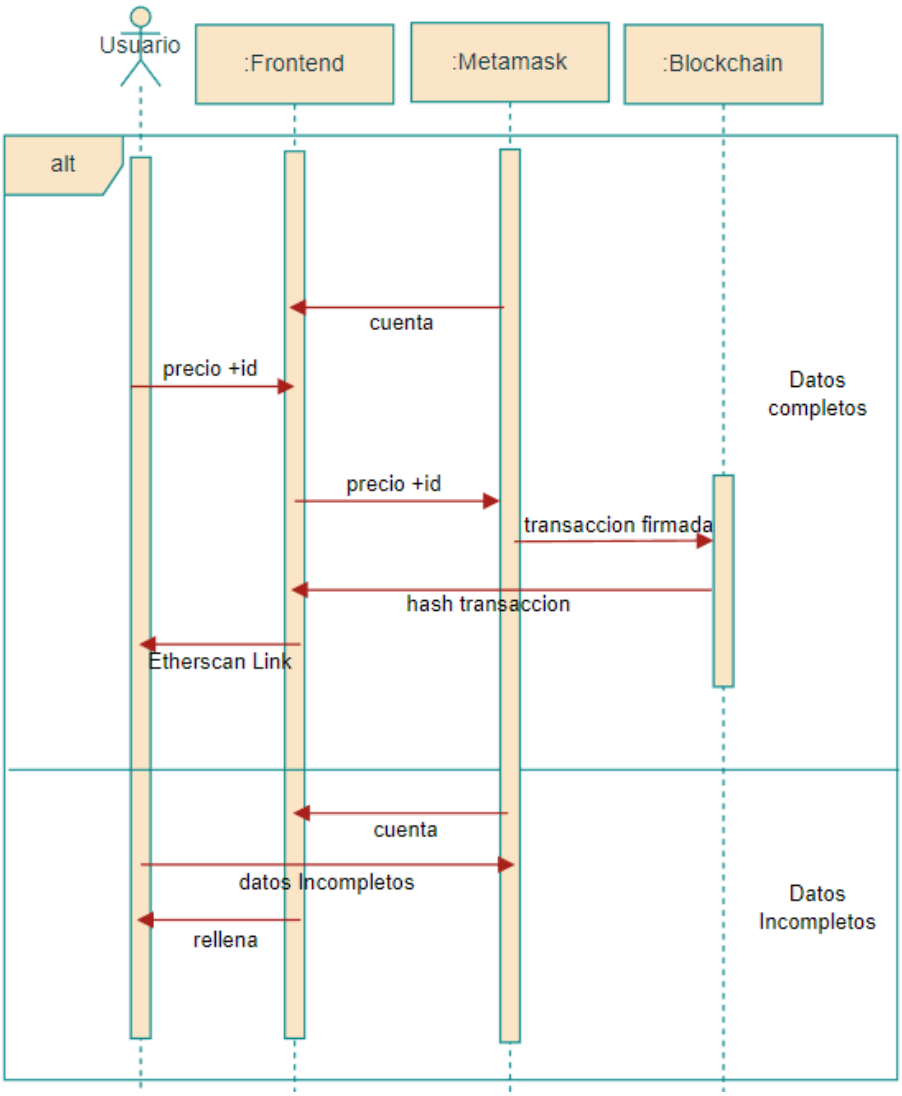

Figura C.5: Diagrama de secuencia de puesta a la venta

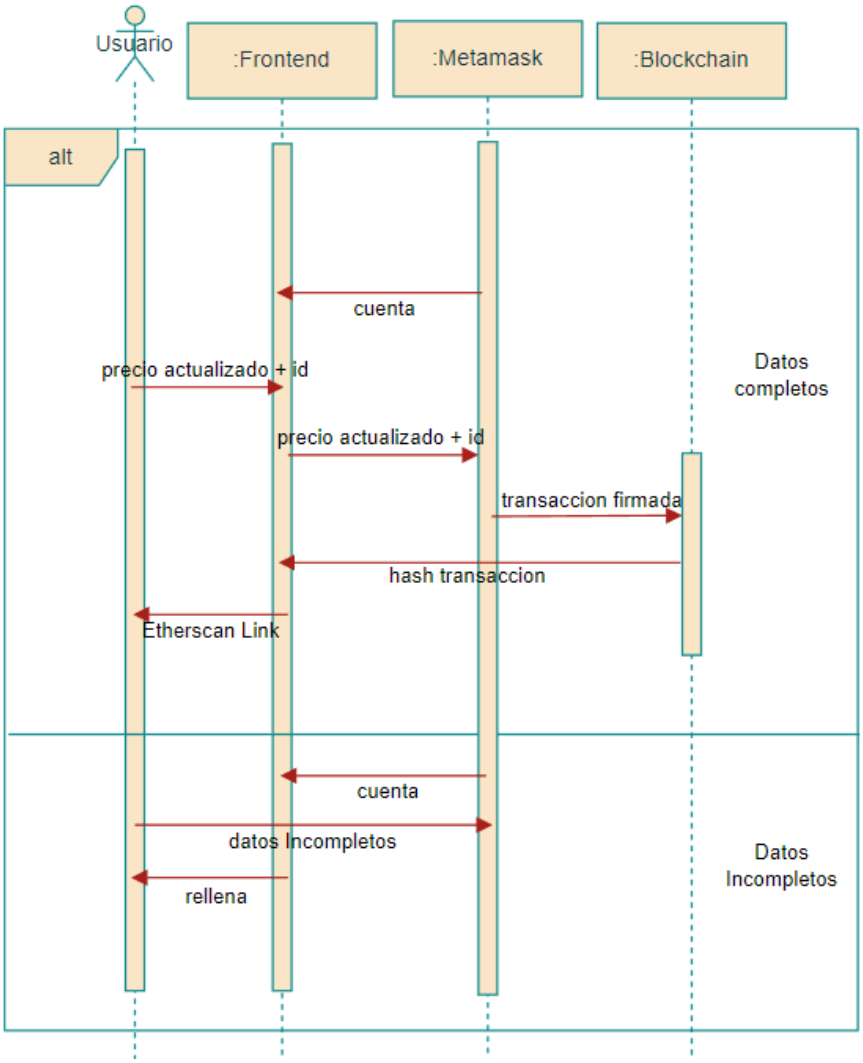

Figura C.6: Diagrama de secuencia de cambio de precio de un teléfono a la venta

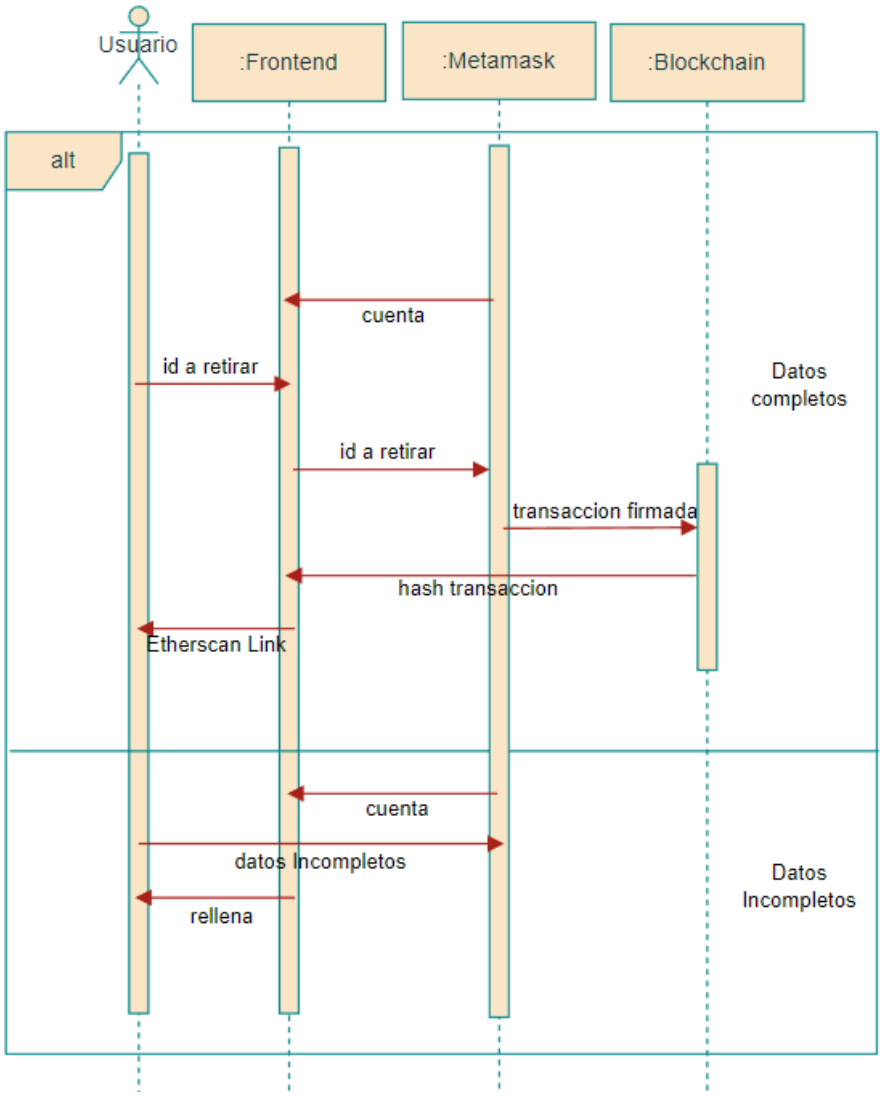

Figura C.7: Diagrama de secuencia de la retirada de la venta de un teléfono

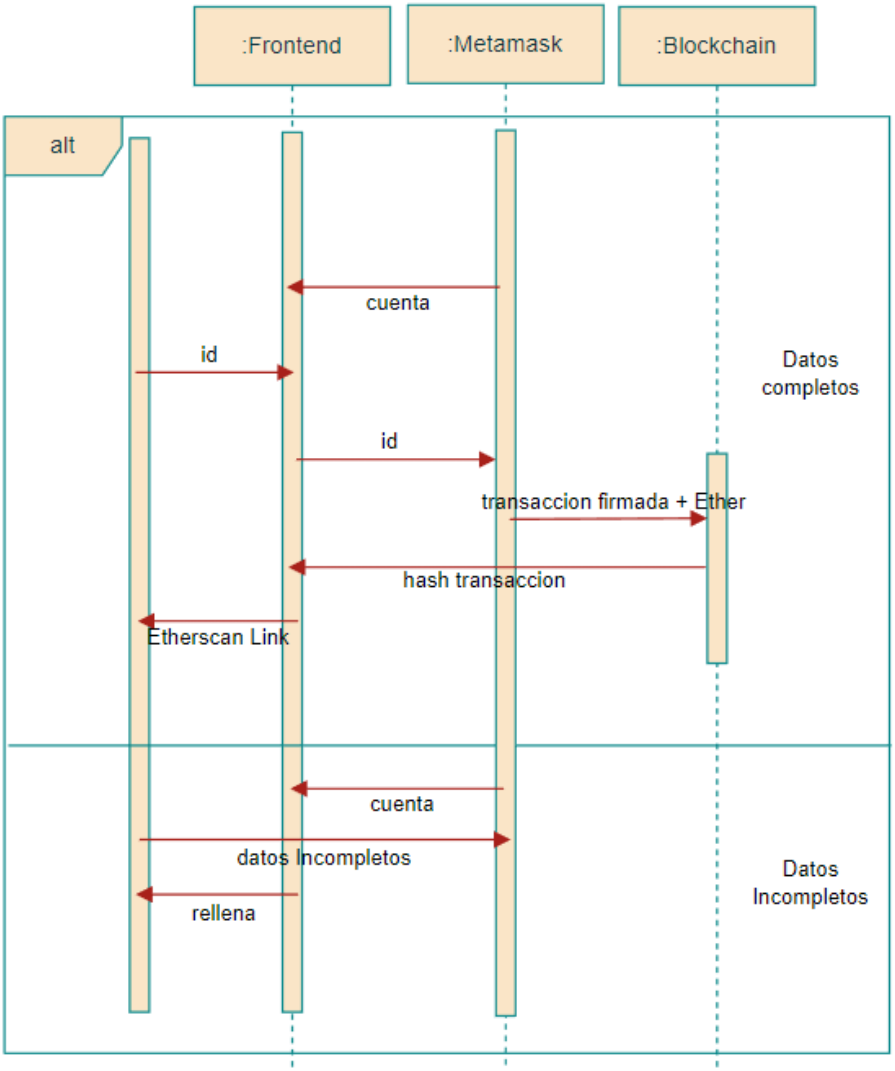

Figura C.8: Diagrama de secuencia de la compra de un teléfono

C.4. Diseño arquitectónico

En la figura C.9 podemos ver el diagrama de diseño arquitectónico del conjunto de la aplicación:

- Los usuarios (tanto de ordenador como de dispositivo móvil) pueden contar con *Metamask* o directamente con la aplicación.

- *Metamask* recibirá solicitudes del *frontend* para firmar transacciones que devolverá a este y serán enviadas. Además un usuario puede iniciar sesión para poder consultar los datos asociados a su cuenta.
- *NFT.Storage* recibirá imágenes que almacenará, a cambio devolverá al *frontend* el CID de esa imagen.
- *Alchemy*, nuestro proveedor de nodos recibirá peticiones de el *frontend*, a las cuales responderá según tenga esa información. Además ejecutará las transacciones firmadas que reciba del *frontend*.
- El *frontend* es la piedra angular de nuestra aplicación, el punto en común entre todos estos servicios y el encargado de su organización.

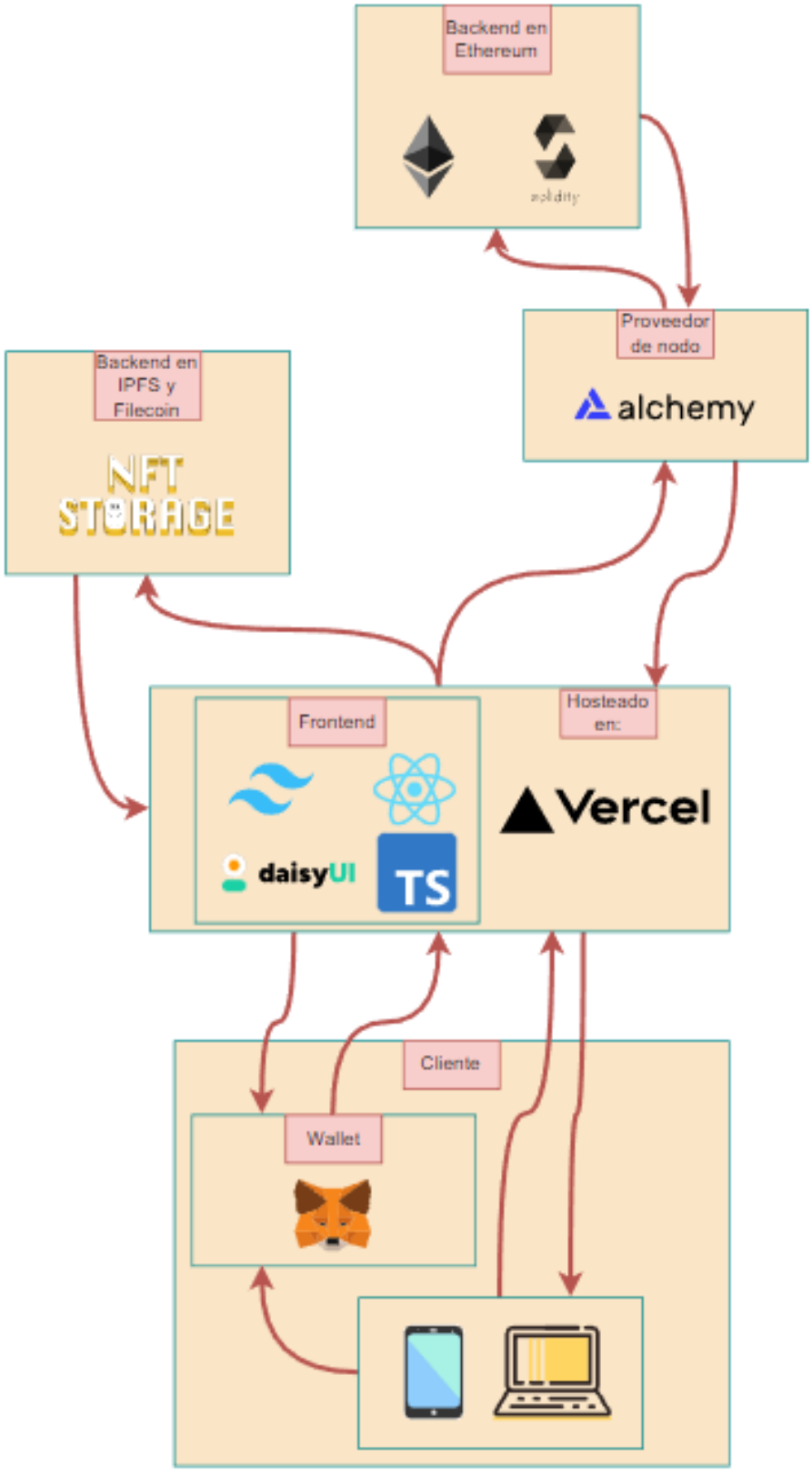

Figura C.9: Diagrama de diseño arquitectónico de la aplicación

Apéndice D

Documentación técnica de programación

D.1. Introducción

En primer lugar me gustaría destacar que esta aplicación está completamente desplegada y se puede acceder a ella desde https://second-hand-chain.vercel.app/ donde se pueden verificar todas sus funcionalidades.

Por ello, para utilizar esta aplicación no es necesario instalar nada (excepto la extensión de *Metamask* en tu navegador).

En este apéndice se va a dar toda la información necesaria al programador para poder configurar la aplicación y sus entornos de desarrollo.

D.2. Estructura de directorios

- *documentation*: Documentación.
- *Models UML*: Carpeta con los diagramas de la aplicación.
- *Truffle*: Contratos Inteligentes.
  - *Build/contracts*: ABI[1] de los contratos.
  - *contracts*: código de los contratos.
  - *test*: carpeta creada por defecto para posibles tests .

[1]Interfaz Binaria de Aplicación

- *frontend/second-hand-chain*: Provecto de *React TypeScript*
    - *dist*: Ficheros y videos en el proyecto compilado.
    - *node_modules*: node y librerias.
    - *src*: código.
        - *componentes*: Componentes de *React*.
        - *hooks*: *Custom hooks* empleados.
        - *services*: Servicios a llamar desde los componentes.

D.3. Manual del programador

A continuación se explica el software que es necesario instalar para ejecutar la aplicación:

Instalar *Node.js*

Es necesario instalar la versión v16.14.0 de *Node.js*. Es importante que sea esta versión ya que la mayoría de este proyecto depende de este *software*.

Por otro lado cuenta con la ventaja de ser una versión LTS[2] para la cual se va a ofrecer soporte durante más tiempo de para otras.

Podemos descargar este software desde su web oficial: `https://nodejs.dev/en/`.

Instalar *Truffle*

Este entorno de desarrollo para contratos inteligentes nos permitirá crear proyectos de *Truffle* e interactuar con ellos.

Para ello emplearemos el comando: `npm install -g truffle`.

Además instalaremos *Ganache* con: `npm i ganache-cli`

Esta versión de *Ganache* es solamente para consola aunque a mí personalmente me ha parecido más cómoda que la que cuenta con interfaz.

[2]Long term service

D.4. Compilación, instalación y ejecución del proyecto

Obtener claves para *Alchemy* y *NFT.Storage*

Para poder ejecutar el proyecto una vez desplegado en *Blockchain* vamos a necesitar las siguientes *API KEYS*:

- *Alchemy*: Podemos obtenerla mediante el registro en su web y creando un proyecto, sección en la que nos aparecerá la clave y el *endpoint*. Enlace: https: //docs.alchemy.com/docs/alchemy-quickstart-guide#1key-create-an-alchemy-key.
- *Nft.Storage*:Deberemos obtener otra clave para este servicio, para lo cual también es necesario el registro. Enlace: https://nft.storage/docs/.

Configurar el *.env*

En los directorios con el *frontend* podemos comprobar que tenemos un fichero llamado *.env.example*, este fichero contiene el nombre exacto de las variables a ser igualadas con las *API KEYS* obtenidas en la sección anterior. Para ello:

- Copiar y pegar dicho fichero en el mismo directorio.
- La copia renombrarla a *.env*.
- Ahora registra las *API KEYS* en ese fichero.
- Si tienes un despliegue de los contratos copiar también la dirección del contrato a su variable.

Instalación de librerías de *Javascript* en los proyectos de *Truffle* y *React*

Para realizar este paso debemos entrar en el directorio raíz de cada uno de los proyectos y ejecutar: `npm install`.

Con este comando instalamos todas las librerías de golpe con sus versiones ya configuradas, ya que están definidas en los ficheros de dependencias[3].

[3]package.json

A continuación, para ejecutar el proyecto de *React* deberemos ejecutar en consola: `npm run dev`.

Tras ello nos aparecerá un enlace en la consola con la dirección local en la que se está ejecutando. No hay que hacer nada más ya que a cada cambio en este proyecto se recompila automáticamente añadiendo solamente el código que ha cambiado gracias a *Vite*.

Desplegar el contrato en *Sepolia*

- Descargar el fichero *truffle-config.js* de uno de los *commits* anteriores a añadirlo a *.gitIgnore*.
- Deberemos configurar el fichero *truffle-config.js* siguiendo el formato en la figura D.1.
- Introducir nuestra clave privada de una *wallet* con suficiente *Ether* donde se indica en la figura D.1.
- Introducir la *API KEY* de *Alchemy*.
- Ejecutar: `truffle migrate -network sepolia`.
- En la salida se esta última ejecución aparece la dirección del contrato, pegarla en su campo correspondiente del fichero *.env*.

truffle > migrations > 01-secondHandChainDeployment.js > ...

You, 1 second ago | 1 author (You)

```
1  var SecondHandChain = artifacts.require("SecondHandChain");
2  var ERC721 = artifacts.require("ERC721");
3  var ERC721Metadata =  artifacts.require("ERC721Metadata");
4
5  module.exports = async function(deployer, network, accounts) {
6    await deployer.deploy(SecondHandChain);
7    await deployer.link(SecondHandChain, ERC721);
8    await deployer.deploy(ERC721);
9    await deployer.link(ERC721, ERC721Metadata);
10   await deployer.deploy(ERC721Metadata);
11 }
```

Figura D.1: Configuración de *truffle-config.js* para despliegue en *Sepolia*

D.5. Pruebas del sistema

Se ha realizado una serie de pruebas manuales a lo largo del desarrollo de la aplicación para verificar que se cumplen en todo momento los requisitos establecidos.

Se han diseñado estas pruebas para conseguir el mayor porcentaje posible de recubrimiento de código. A continuación se listan cada uno de los casos de prueba testados:

CP-1	**Inicio de Sesión en *Metamask* correcto**
Autor	Áxel Rubio González
Descripción	En este caso de prueba el usuario iniciará sesión en *Metamask* correctamente
Pasos	1. El usuario accede a la web de Second Hand Chain. 2. Le aparece una ventana para iniciar sesión en *Metamask*. 3. Introduce su contraseña. 4. Puede consultar sus teléfonos en propiedad.
Estado	Correcto

Tabla D.1: CP-1 Inicio de Sesión en *Metamask* correcto.

CP-2	**Inicio de Sesión en *Metamask* pero cierra la ventana antes**
Autor	Áxel Rubio González
Descripción	En este caso de prueba el usuario accederá a la web, cerrará la ventana de *Metamask* y accederá desde la extensión del navegador.
Pasos	1. El usuario accede a la web de Second Hand Chain. 2. Le aparece una ventana para iniciar sesión en *Metamask*. 3. Cierra la Ventana. 4. Le aparece un mensaje diciéndole que acceda desde la extensión del navegador en el botón derecho de la *NavBar*. 5. Accede a la extensión del navegador de *Metamask*. 6. Introduce su contraseña. 7. Su cuenta aparece en el botón. 8. Puede consultar sus teléfonos en propiedad.
Estado	Correcto

Tabla D.2: CP-2 Inicio de Sesión en *Metamask* pero cierra la ventana antes.

CP-3	**Usuarios sin *Metamask***
Autor	Áxel Rubio González
Descripción	En este caso de prueba el usuario accederá a la web, sin la extensión instalada, podrá consultar datos pero no operar.
Pasos	1. El usuario accede a la web de Second Hand Chain. 2. Le aparece un mensaje diciéndole que instale *Metamask*. 3. Accede a consultar los teléfonos en venta. 4. Selecciona uno. 5. Le aparece un mensaje en el botón de compra diciendole que inicie sesión en *Metamask*.
Estado	Correcto

Tabla D.3: CP-3 Usuarios sin *Metamask*.

CP-4	**Creación correcta de teléfono**
Autor	Áxel Rubio González
Descripción	En este caso de prueba el usuario, con la cuenta de *Metamask* ya configurada registrará correctamente un teléfono.
Pasos	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en crear teléfono. 3. Introduce todos los datos correctamente, incluyendo la imagen que selecciona de su dispositivo. 4. Hace clic en crear. 5. Firma correctamente la transacción en *Metamask*. 6. Le aparece un mensaje con un enlace a Etherscan. 7. Accede satisfactoriamente a *Etherscan* para seguir el estado de su transacción.
Estado	Correcto

Tabla D.4: CP-4 Creación correcta de teléfono.

CP-5	**Creación incorrecta de teléfono por datos incompletos**
Autor	Áxel Rubio González
Descripción	En este caso de prueba el usuario, con la cuenta de *Metamask* ya configurada intentará registrar un teléfono sin haber introducido todos los datos necesarios.
Pasos	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en crear teléfono. 3. Introduce todos los datos excepto uno que no completa. 4. Hace clic en crear. 5. Le aparece un mensaje avisando de que tiene datos incompletos.
Estado	Correcto

Tabla D.5: CP-5 Creación incorrecta de teléfono por datos incompletos.

CP-6	**Creación incorrecta de teléfono por IMEI ya registrado**
Autor	Áxel Rubio González
Descripción	En este caso de prueba el usuario, con la cuenta de *Metamask* ya configurada, intentará registrar un teléfono previamente registrado.
Pasos	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en crear teléfono. 3. Introduce todos los datos, pero el IMEI pertenece a otro teléfono ya registrado. 4. Hace clic en crear. 5. Le aparece un mensaje avisando de que ese IMEI ya existe en el contrato.
Estado	Correcto

Tabla D.6: CP-6 Creación incorrecta de teléfono por IMEI ya registrado.

CP-7	**Puesta a la venta correcta de teléfono**
Autor	Áxel Rubio González
Descripción	En este caso de prueba, con la cuenta de *Metamask* ya configurada, el usuario pondrá a la venta un teléfono de su propiedad correctamente.
Pasos	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en teléfonos en propiedad. 3. Selecciona uno. 4. Introduce el precio. 5. Hace clic en poner a la venta. 6. Firma correctamente la transacción en *Metamask*. 7. Le aparece un mensaje con un enlace a *Etherscan*. 8. Accede satisfactoriamente a *Etherscan* para seguir el estado de su transacción. 9. Espera a que la transacción se confirme. 10. Accede a teléfonos en venta y puede ver el teléfono a la venta.
Estado	Correcto

Tabla D.7: CP-7 Puesta a la venta correcta de teléfono.

CP-8	**Retirada de la venta correcta de teléfono**
Autor	Áxel Rubio González
Descripción	En este caso de prueba el usuario, con la cuenta de *Metamask* ya configurada, retirará un teléfono de la venta.
Pasos	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en teléfonos en propiedad. 3. Selecciona uno que este a la venta. 4. Hace clic en retirar de la venta. 5. Firma correctamente la transacción en *Metamask*. 6. Le aparece un mensaje con un enlace a *Etherscan*. 7. Accede satisfactoriamente a *Etherscan* para seguir el estado de su transacción. 8. Espera a que la transacción se confirme. 9. Accede a teléfonos en venta y puede ver el teléfono no se puede encontrar. 10. Accede a teléfonos en propiedad 11. Selecciona ese teléfono. 12. No aparece la opción de retirar de la venta.
Estado	Correcto

Tabla D.8: CP-8 Retirada de la venta correcta de teléfono.

CP-9	**Cambio de precio de teléfono a la venta**
Autor	Áxel Rubio González
Descripción	En este caso de prueba el usuario, con la cuenta de *Metamask* ya configurada, cambiará el precio de un teléfono la venta.
Pasos	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en teléfonos en propiedad. 3. Selecciona uno que esté a la venta. 4. Escribe un nuevo precio. 5. Hace clic en cambiar precio. 6. Firma correctamente la transacción en *Metamask*. 7. Le aparece un mensaje con un enlace a *Etherscan*. 8. Accede satisfactoriamente a *Etherscan* para seguir el estado de su transacción. 9. Espera a que la transacción se confirme. 10. Accede a teléfonos en venta y puede ver el teléfono con su nuevo precio.
Estado	Correcto

Tabla D.9: CP-9 Cambio de precio de teléfono.

CP-10	**Compra de Teléfono**
Autor	Áxel Rubio González
Descripción	En este caso de prueba el usuario, con la cuenta de *Metamask* ya configurada, comprará un teléfono.
Pasos	1. El usuario accede a la web de Second Hand Chain. 2. Hace clic en teléfonos a la venta. 3. Selecciona uno que esté a la venta. 4. Hace clic en comprar. 5. Firma correctamente la transacción en *Metamask*. 6. Le aparece un mensaje con un enlace a *Etherscan*. 7. Accede satisfactoriamente a *Etherscan* para seguir el estado de su transacción. 8. Espera a que la transacción se confirme. 9. Accede a teléfonos en propiedad y puede ver el teléfono recién adquirido.
Estado	Correcto

Tabla D.10: CP-10 Compra de Teléfono.

Apéndice E

Documentación de usuario

E.1. Introducción

En este anexo se explica todo lo que necesita el usuario para utilizar la aplicación, y se explica el funcionamiento de la misma, aunque también cuente como soporte con los vídeos de la aplicación.

E.2. Requisitos de usuarios

A continuación, se exponen los requisitos necesarios para poder utilizar la aplicación:

- Tener instalada la extensión de navegador *Metamask*.
- Es preferible usar *Brave* como navegador aunque al estar basado en *Chromium* no hay problemas en emplear cualquier otro que emplee esa base de código abierto.
- Para usuarios de dispositivos móviles tener instalada la aplicación *Metamask*.

E.3. Instalación

Los usuarios deberán añadir la extensión *Metamask* a su navegador, en la web desplegada hay un vídeo con el proceso de instalación, también se encuentra en el repositorio y en los USB entregados, en los ficheros públicos de la aplicación web.

Por otro lado, para poder realizar transacciones los usuarios necesitan *Ether* de la red *Sepolia*, pueden obtenerlos registrándose en *Alchemy* y recurriendo a su *Faucet* (como se muestra en el segundo vídeo en la página web) o los pueden pedir al autor por correo aportando la dirección de su *wallet*.

E.4. Manual del usuario

Finalmente en esta sección vamos a detallar el funcionamiento de la aplicación Second Hand Chain para que los usuarios puedan comprender su funcionamiento detalladamente:

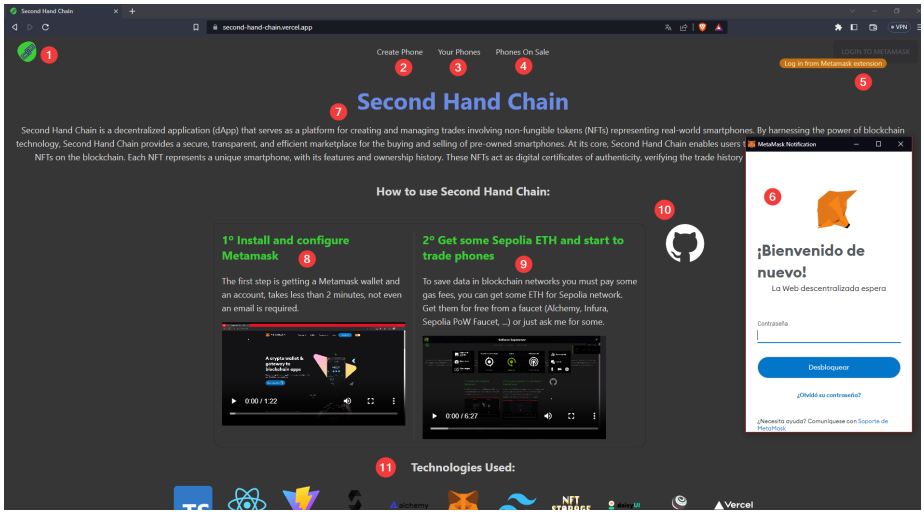

Figura E.1: Vista inicial de la aplicación.

En la figura E.1 podemos destacar los siguientes apartados:

1. Botón que nos lleva a la vista actual de presentación de la aplicación.
2. Botón que abre el modal para registrar teléfonos.
3. Botón para acceder a los teléfonos que posee el usuario.
4. Botón para acceder a los teléfonos en venta.
5. Botón que indica el estado de conexión con *Metamask*.
6. Ventana para iniciar sesión en *Metamask*, aparece si acabas de abrir tu navegador y tienes *Metamask* instalado, permite iniciar sesión, proceso necesario para poder firmar transacciones.

7. Descripción de la aplicación.
8. Vídeo para mostrar como añadir *Metamask*.
9. Vídeo para mostrar como añadir *Ether* a *Metamask* gracias al *faucet* de *Alchemy* y demostración del funcionamiento de la aplicación.
10. Botón para acceder al repositorio de *GitHub*.
11. Listado de tecnologías empleadas en el proyecto.

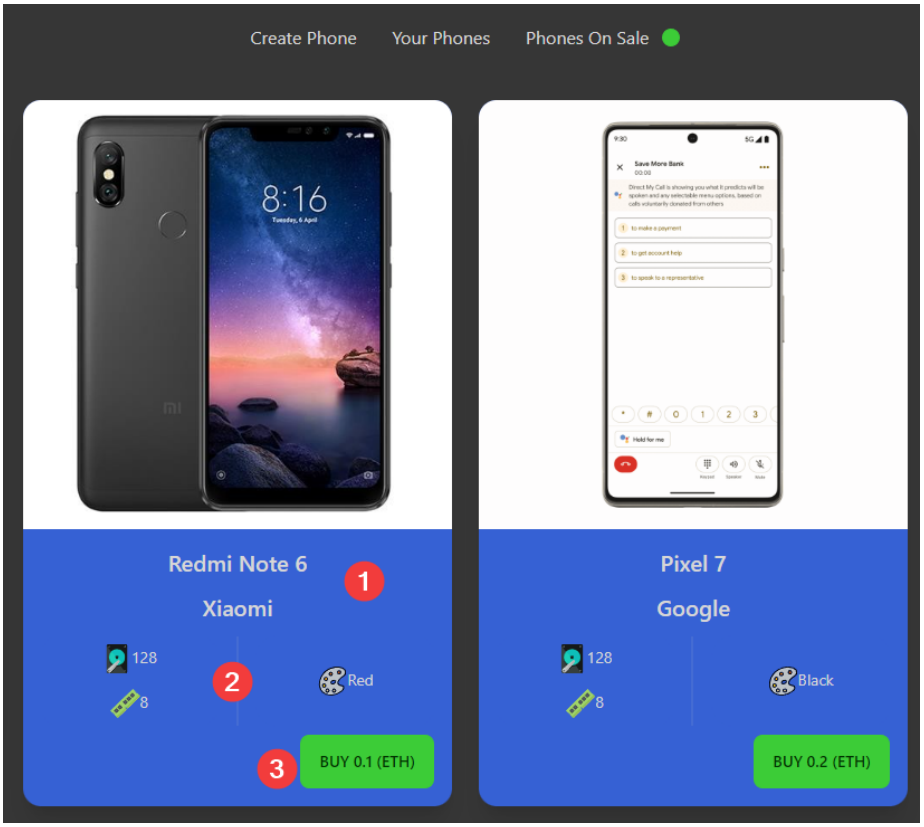

Figura E.2: Vista de tarjetas de teléfonos.

En la figura E.2 podemos ver un par de tarjetas de teléfonos:

1. Modelo y marca del teléfono.
2. Características del teléfono.

3. Botón de compra, permite acceder a los detalles, en el caso de los teléfonos en propiedad aparecerá simplemente la palabra detalles.

4. Características del teléfono.

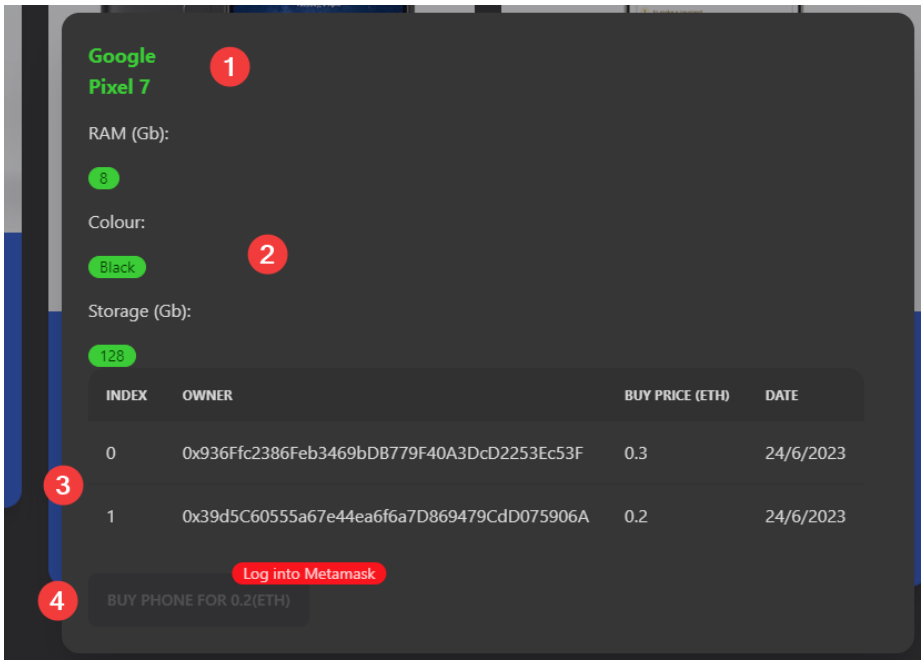

Figura E.3: Vista de detalles de teléfonos.

En la figura E.3 podemos ver un modal con los detalles del teléfono:

1. Modelo y marca del teléfono.

2. Características del teléfono.

3. Historial de acciones de compraventa, siendo la de índice cero la del registro del teléfono.

4. Botón de compra, puede aparecer deshabilitado si no iniciamos sesión, habilitado para poderlo comprar o no aparecer, en el caso de acceder desde teléfonos en propiedad.

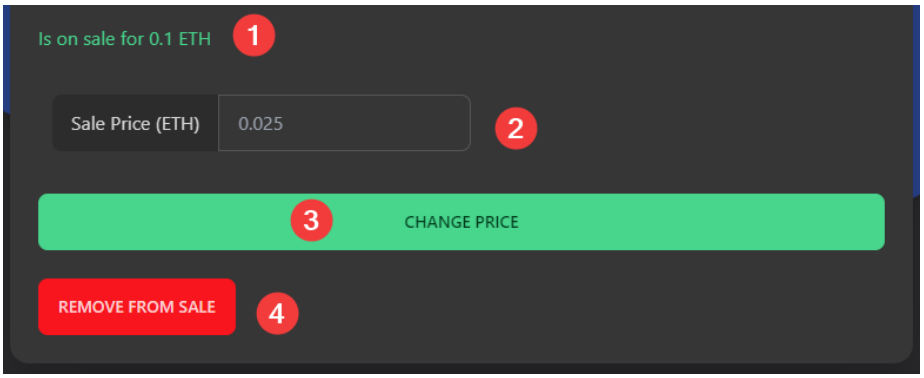

Figura E.4: Vista de detalles de teléfonos en propiedad, parte para configurar el estado del teléfono.

En la figura E.4 podemos ver un segmento del modal con las opciones a configurar por un propietario para poner a la venta el teléfono:

1. Si el teléfono está a la venta o no y su precio, en caso de estarlo.
2. Campo para introducir el precio por el teléfono.
3. Botón de cambiar de precio si está a la venta o de poner en venta si no lo estaba.
4. Botón de retirar de la venta, solo aparece si se vende el teléfono.

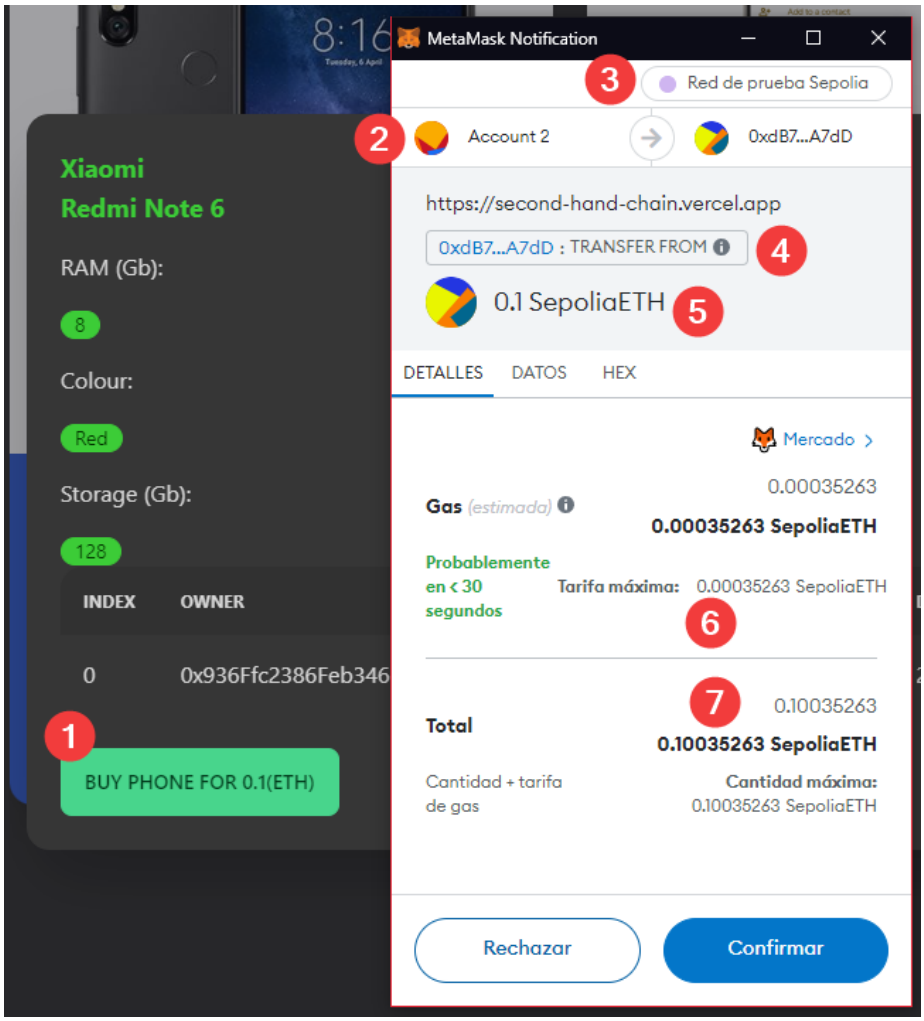

Figura E.5: Vista de transacción de compra de teléfono.

En la figura E.5 podemos ver el proceso para comprar un teléfono:

1. Se hace clic en el botón de compra por el cual nos aparece un modal de *Metamask* solicitando que firmemos la transacción.
2. Podemos ver la cuenta con la que vamos a firmar la transacción.
3. Estamos en la *testnet Sepolia*.
4. El método que vamos a ejecutar, en este caso es detectado al pertenecer a *ERC721*.
5. Valor de *Ether* que vamos a aportar.

6. *Gas fees.*

7. *Valor total de Ether que se va a emplear.*

Finalmente confirmaríamos la transacción.

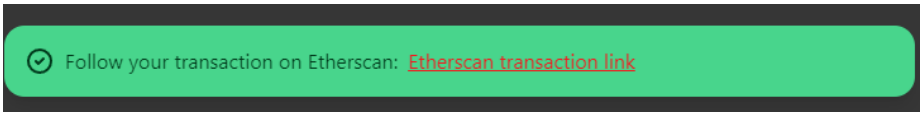

Figura E.6: Vista de transacción de compra de teléfono.

En la figura E.6 podemos ver el mensaje con un enlace para abrir la transacción realizada en *Etherscan*.

Figura E.7: Vista de la transacción de compra realizada en *Etherscan*.

La figura E.7 nos muestra los detalles de la transacción previamente realizada, en *Etherscan*, vamos a ver sus detalles:

1. Nos recuerda que la transacción pertenece a la *testnet Sepolia*.

2. El *hash* de la transacción.

3. Estamos en la *testnet Sepolia*.

4. Momento en el que se minó el bloque al que pertenece nuestra transacción.
5. Dirección origen, la del comprador.
6. Dirección con la que ha interactuado la anterior, la de nuestro contrato inteligente.
7. Nos detecta que es un token en cuyo contrato inteligente se implementa la interfaz *IERC721*
8. Identificador del token transferido.
9. Descripción del contrato.
10. Antiguo propietario del *NFT*.
11. Nuevo propietario del *NFT*.
12. Valor en *Ether* transferido.

Bibliografía

[1] SeguridadSocial. Seguridad Social: CotizacixF3;n / RecaudacixF3;n de Trabajadores — seg-social.es. https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537#36538. [Accessed 02-Jul-2023].

[2] StackOverflow. Stack Overflow Developer Survey 2022 — survey.stackoverflow.co. https://survey.stackoverflow.co/2022#top-paying-technologies-programming-scripting-and-markup-languages. [Accessed 02-Jul-2023].