Tecnologías de Registro Distribuido y Blockchain
Máster Inter-Universitario en Ciberseguridad (MUNICS)
Universidade da Coruña (UDC) y Universidade de Vigo (UVigo)
Curso 2024-2025

Memoria: Traballo tutelado

Axel Valladares Pazó, Pedro Blanco Casal

1 Introducción

i. Motivación:
La creciente popularidad de los videojuegos y la economía basada en activos digitales, como skins, ha generado la necesidad de crear mercados confiables y transparentes para su gestión. Muchos mercados tradicionales de skins carecen de transparencia, tienen altas tasas de fraude o no permiten una monetización justa para los usuarios. Este proyecto busca resolver esas limitaciones utilizando una red blockchain y redes de almacenamiento descentralizadas como IPFS. Al utilizar Ethereum para gestionar transacciones y IPFS para almacenar imágenes de las skins, se ofrece a los usuarios una solución transparente, confiable y resistente a la censura. Esto garantiza la autenticidad de las skins y su disponibilidad a nivel global.

ii. Definición del escenario:
El proyecto se centra en un mercado digital para skins de videojuegos, donde los usuarios pueden **comprarlas**, **alquilarlas** o pujar por ellas en una **subasta** de forma descentralizada. Utilizamos Ethereum como plataforma base, ya que permite un manejo seguro y verificable de los activos digitales. Además, de puede consultar y visualizar las imágenes de las skins almacenadas de forma segura en la red IPFS, lo que asegura que los archivos no puedan ser modificados ni eliminados por terceros.

iii. Objetivos:

- **Proveer un mercado seguro y descentralizado**: Crear una plataforma donde los usuarios puedan comerciar skins de videojuego con total confianza, respaldada por Ethereum e IPFS.
- **Integrar IPFS para almacenamiento descentralizado**: Asegurar que las imágenes de las skins estén disponibles en todo momento y sean inmutables, protegiendo tanto a vendedores como a compradores de manipulaciones.
- **Automatizar transacciones mediante contratos inteligentes**: Garantizar que las compras, alquileres y subastas se ejecuten automáticamente siguiendo reglas predefinidas, sin necesidad de intermediarios. Estos contratos inteligentes estarán desarrollados en Solidity.
- **Garantizar la transparencia y trazabilidad**: Permitir a los usuarios verificar en cualquier momento la propiedad, historial de transacciones y estado actual de cada skin.

- **Poner a disposición una aplicación web**: Para que tanto administradores como clientes de la tienda tengan un portal e interfaz web al que acceder para realizar con mayor comodidad y facilidad las transacciones que deseen.
- **Fomentar la adopción de tecnologías blockchain**: Demostrar cómo Ethereum e IPFS pueden combinarse para solucionar problemas del mundo real en mercados digitales.

2 Estudio del estado del arte

i. Análisis de soluciones parecidas
Hay muchas soluciones populares, como *Steam market* [1] o *cs.money* [2], la mayoría centralizadas, ya que dependen de uno o pocos servidores, carecen de transparencia total ya que están dirigidas y administradas por una entidad central.

Otros proyectos basados en redes blockchain ofrecen artes con otras finalidades, como *OpenSea* [3], vendiendo estos productos como *NFTs (Non-Fungible Token)*. Y otros también ofrecen skins, como *skinflow.gg*, que al igual que nuestra propuesta ofrecen skins a cambio de criptomonedas.

ii. Comparativa con la solución propuesta A diferencia de los anteriores ejemplos, nuestro proyecto ofrece una solución descentralizada y transparente basada en Ethereum. Lo cuál dará más seguridad a los clientes o usuarios finales a la hora de elegir esta solución.

Además se ofrece el alquiler de skins por días, lo cuál no es muy común debido a que se debe confiar en los usuarios finales.

3 Funcionalidades

La aplicación implementa la venta, alquiler y puja de skins de carácter genérico para lo que el propietario del Smart Contract quiera especificar.
El proyecto se basa en dos apartados:

- Smart Contract: Implementa toda la lógica de las funciones que se utilizan en el "frontend", permitiendo al propietario del mismo controlar la funcionalidad de crear skins y administrar los archivos desde un nodo IPFS, ponerlas a la venta o alquiler y la posibilidad de montar una subasta con ellas. Dicho Smart Contract cuenta con un mecanismo de seguridad que permite que algunas de las funcionalidades solo puedan ser ejecutadas por el propiertario del contrato.

- Aplicación react: Permite aplicar la lógica que presenta el Smart Contract a un entorno web amigable al usuario, implementando las distintas funcionalidades que recoge el contrato y manejando las solicitudes de los usuarios de la página. Además, permite subir archivos a un nodo IPFS cuando el propiertario del contrato quiera crear una skin.

4 Ciberseguridad

El Smart Contract cuenta con un manejo de usuarios de forma que el usuarios que despliegue el contrato podrá acceder a ciertas funcionalidades que solo él podrá ejecutar

(Ver Sección 8).

Además cada funcionalidad del Smart Contract comprueba constantemente los estados de las skins. Para impedir, por ejemplo, que un usuario pueda alquilar o pujar por una skin ya vendida a otro usuario, o impedir que el propietario del contrato monte una subasta con una skin que esté en alquiler o vendida.

Todo esto permite confiar con mayor facilidad en el Smart Contract ya que podrán ver ellos mismos como el estado de sus skins se comprueba en cada transacción impidiendo robos de skins.

5 Metodología (opcional)

La metodología utilizada en este proyecto se basó en un diseño inicial de funcionalidad, donde se dictaminó qué queríamos que incluyese nuestra aplicación.
Una vez realizado el diseño, se implementó el Smart Contract con toda la funcionalidad completa en conjunto con todos los miembros del desarrollo. Además, fue probada mediante Remix IDE, para corroborar que todas las funcionalidades se hacían correctamente e intentando buscar vulnerabilidades y fallos en la lógica del contrato.
Una vez probada toda la funcionalidad del Smart Contract, se desplegó mediante una wallet de Metamask y se implementó dentro de una aplicación React. Dicha aplicación se diseñó con la idea de hacer un entorno amigable a los usuarios y de fácil acceso, permitiendo poder interactuar con toda la funcionalidad del Smart Contract de forma sencilla y rápida. Cada acción que realice un usuario irá asociado con una transación directa con su wallet de Metamask, permitiendo de esta forma un método de pago sencillo e intuitivo.

6 Análisis

i. Actores
   Los dos tipos de actores que se pueden identificar en nuestra solución serán:

   - Administrador o propietario: aquel que despliegue el contrato inteligente, que dispondrá de las funcionalidades de crear las Skins, montar las subastas por estas y la obtención de los ingresos adquiridos por el contrato inteligente a la hora de cobrar tanto pujas como ventas y alquileres.
   - Cliente o usuario común: aquel que accederá a la DApp con la finalidad de comprar y alquilar skins ofrecidas o pujar por ellas en las subastas para obtenerlas.

ii. Requisitos
   En cuanto a los requisitos funcionales es necesario que los usuarios se puedan autenticar y tengan una correcta autorización a la hora de ejecutar las funcionalidades de la DApp. También será necesario comprobar el estado de las skins antes de cada transacción, las cuales deben ser en tiempo real.

Por otra parte, los requisitos no funcionales serían la capacidad de manejar múltiples usuarios simultáneamente sin comprometer el rendimiento. Que la aplicación sea segura y que las transacciones estén protegidas. Y todo ello ofreciendo una interfaz intuitiva, permitiendo que usuarios sin experiencia técnica puedan utilizarla.

Finalmente, los requisitos técnicos que se han destacados son: el uso de Ethereum como red base, y el uso de contratos inteligentes. Desarrollados en Solidity con optimizaciones para minimizar el consumo de gas, y con la finalidad de que sean compatibles con wallets como MetaMask.

iii. Justificación del uso de Ethereum
El uso de Smart Contracts es fundamental para el desarrollo de esta aplicación por multiples motivos. Estos son programas autoejecutables almacenados en una blockchain que permiten implementar reglas y lógicas de negocio de forma descentralizada, sin depender de uno o pocos servidores. Esto garantiza permite que operaciones como la compra, alquiler o subasta de skins se ejecuten automáticamente al cumplirse las condiciones establecidas. Por ejemplo, cuando un usuario paga el precio de una skin, el contrato transfiere automáticamente la propiedad al comprador. Además, estos contratos inteligentes proporcionan confianza, ya que son públicos y verificables por los usuario. Y son seguros, ya que una buena implementación protege el sistema contra manipulaciones y errores humanos.

Estos Smart Contracts deben estar soportados por una plataforma base robusta, es por ello que se ha elegido Ethereum. Esta blockchain es una de las más utilizadas para descentralizadas, ofreciendo un ecosistema rico en herramientas y recursos.

Ethereum destaca por su seguridad y transparencia ya que todas las transacciones son registradas en una blockchain pública, lo que garantiza dicha transparencia y evita manipulaciones. Además es compatible con una amplia variedad de Wallets, facilitando a los usuarios interactuar con la DApp sin necesidad de soluciones personalizadas.

Finalmente, comentar que Ethereum soporta estándares como ERC-721 para la gestión de tokens no fungibles (NFTs), lo que permite a las skins interoperar con otras plataformas en líneas futuras.

En conclusión, esta es la mejor solución encontrada que permite una mayor comodidad, seguridad y transparencia para que tanto usuarios como propietarios estén satisfechos.

7 Diseño

i. Casos de uso
Este contrato se pensó con la finalidad de que un usuario, propietario de la tienda ponga a la venta, alquiler o en subasta una colección de skins de su juego favorito sin la necesidad de un agente externo, ya que cuenta con todo lo necesario para realizar las distintas acciones de venta o alquiler sin la necesidad de transacciones bancarias, ya que funciona sobre la red de Ethereum.
Además, como el alquiler de skins se realiza mediante el Smart Contract, no es necesario que el propietario de la tienda esté pendiente de la devolución de las mismas, ya que se encarga el contrato de forma autónoma.

ii. Diagramas de secuencia

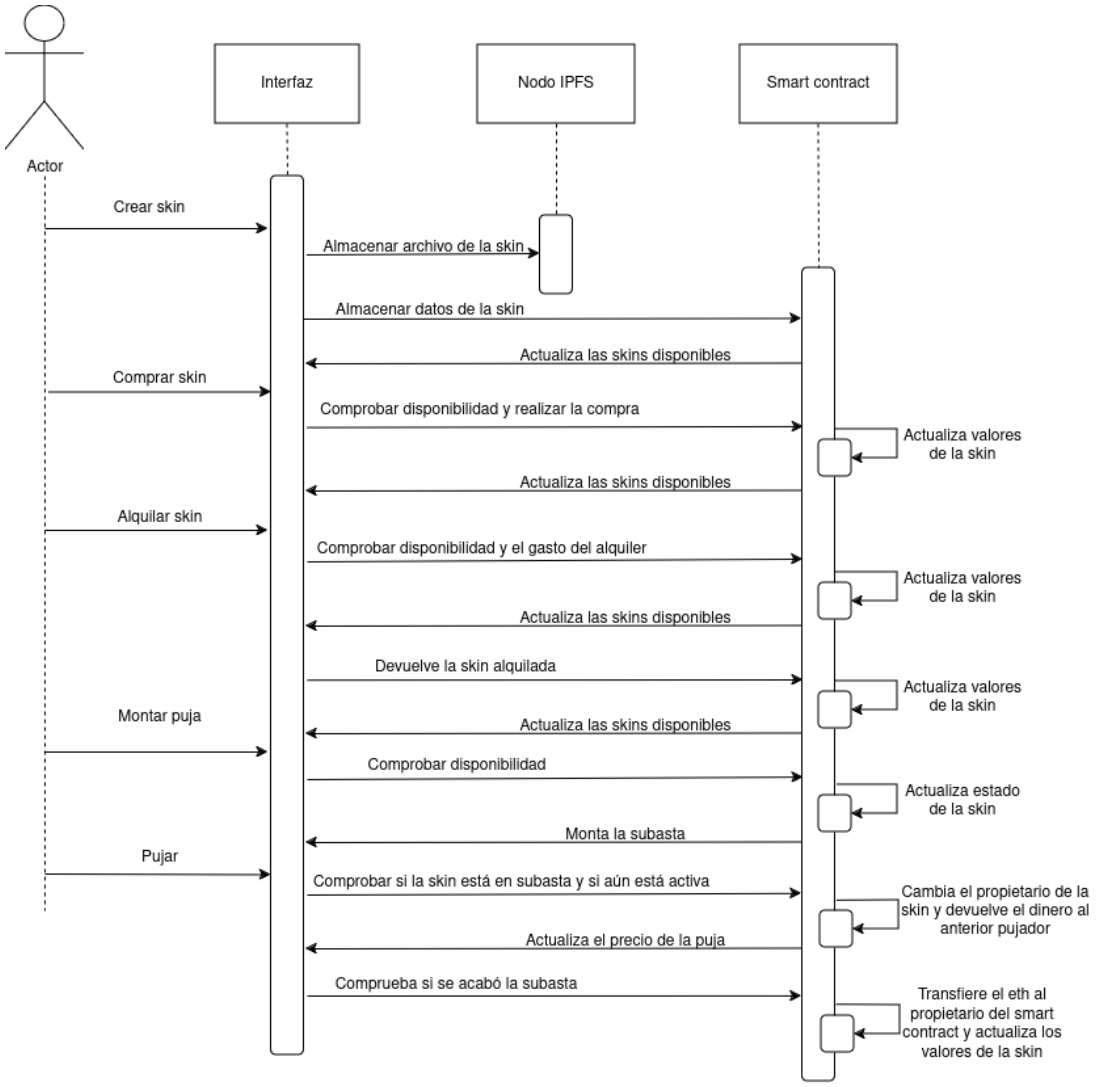

Figure 1: Diagrama de secuencia del la DApp.

iii. Arquitectura de comunicaciones

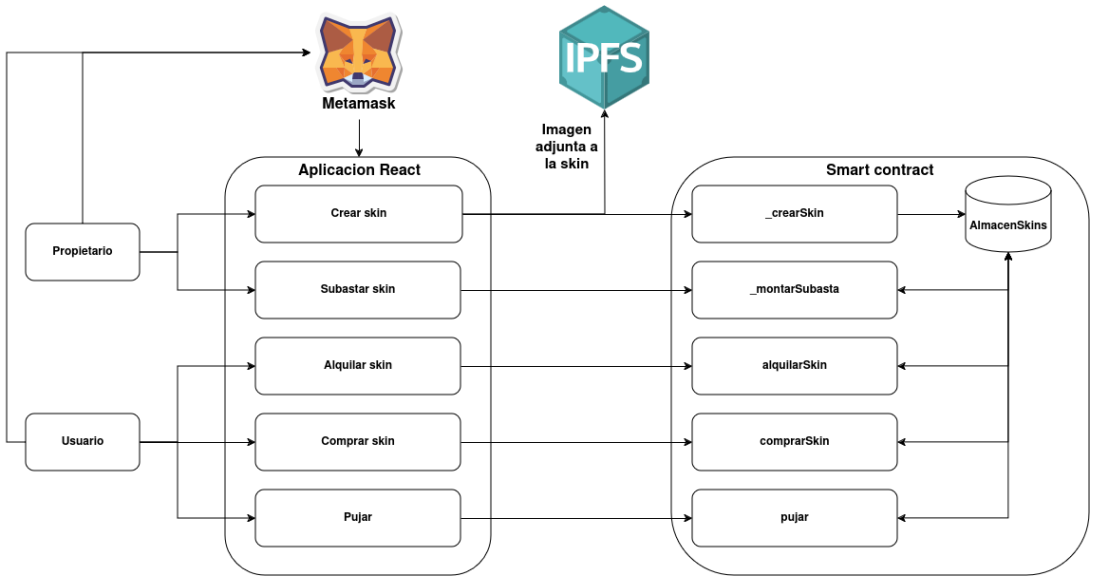

Figure 2: Arquitectura de comunicaciones de la DApp.

iv. Modelo de datos

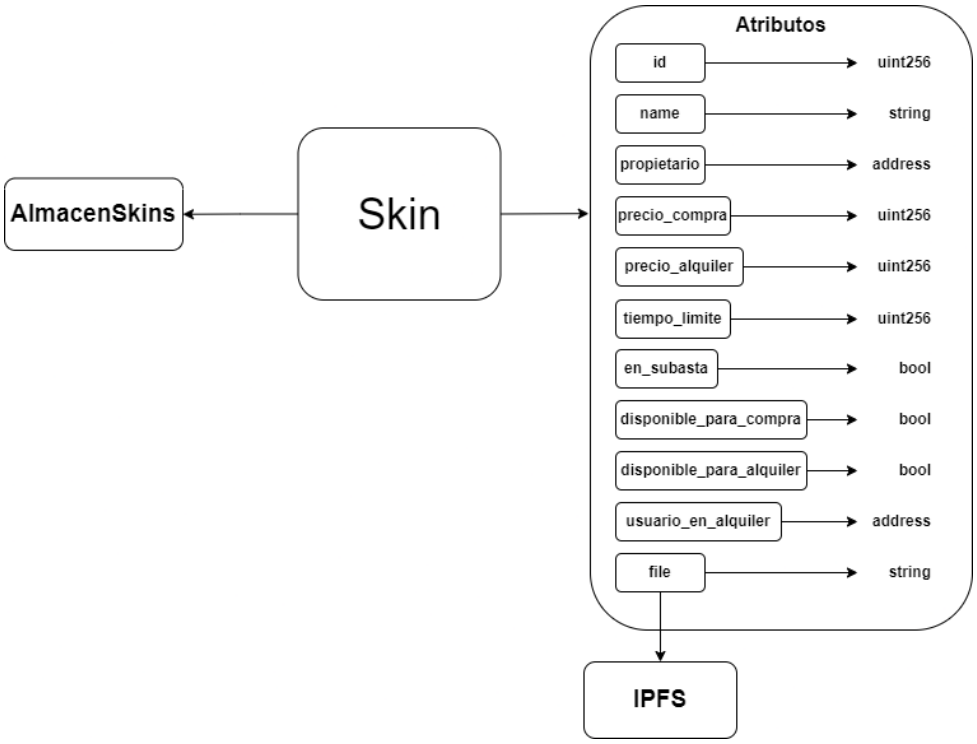

Figure 3: Modelo de datos de la DApp.

v. Tecnologías utilizadas Para este proyecto se han empleado las siguientes tecnologías:

- Solidity
- NodeJS
- Docker

8 Implementación

Se ha creado un Smart Contract que maneja la lógica de la aplicación que se ha implementado. Las funciones que implementa son las que siguen:

- _crearSkin: función encargada de crear skins por parte del propietario del Smart Contract.
  Atributos:
  - _name: nombre de la skin.
  - _file: el archivo asociado a la skin.
  - _precio_compra: precio de compra que le quiera poner el admin.
  - _precio_alquiler: precio del alquiler por días que le quiera poner el admin.
- comprarSkin: función que permite a un usuario comprar la skin que se indique mediante un identificador.
  Atributos:
  - _id: identificador de la skin que se quiera comprar.
- alquilerSkin: función que permite alquilar una skin durante el número de días que se indiquen.
  Atributos:
  - _id: identificador de la skins que se quiera alquilar.
  - _diasAlquiler: número de días que se quiera alquilar la skin.
- devolverSkin: función encargada de devolver una skin despúes de que haya vencido el tiempo de alquiler.
  Atributos:
  - _id: identificador de la skin que se vaya a devolver.
- _montarSubasta: función que permite al propietario del Smart Contract montar una subasta con una skin que esté disponible.
  Atributos:
  - _id: identificador de la skin con la que se vaya a crear la puja.
  - _tiempo: cantidad de tiempo en minutos que dura la subasta.
- pujar: función que permite pujar por una skin a los usuarios, siempre que su subasta haya sido montada por el propietario.
  Atributos:

  - _id: identificador de la skin por la que se vaya a pujar.

- _recogerSubastas: función que permite al propietario del Smart Contract recoger todos los ingresos de aquellas subastas cuyo tiempo limite haya vencido.

Como "frontend" se ha empleado una aplicación react, la cual implementa la funcionalidad del Smart Contract en una aplicación DApp, permitiendo una accesibilidad al usuario de forma cómoda y amigable a las distintas funcionalidades que nos proporciona el Smart Contract. El propietario del Smart Contract tendrá una vista de la DApp distinta al usuario común. Dicha vista permite crear skins y adjuntar un archivo que será enviado al IPFS para que quede alojado allí. Además, permite ver las skins que están disponibles y cuales fueron vendidas y a quién, y, de querer montar una puja con una skin que esté disponible, también podrá efectuar dicha funcionalidad.
En el caso de un usuario común, la funcionalidad que tendrá disponible es la de poder comprar o alquilar alguna skin que esté disponible para ello, y poder pujar por aquellas que el propietario haya puesto en subasta.

9 Demostración del funcionamiento del sistema

i. Despliegue
Para el correcto funcionamiento de la DApp, se ha desplegado un docker que aloja el nodo IPFS de forma local en el ordenador de prueba.

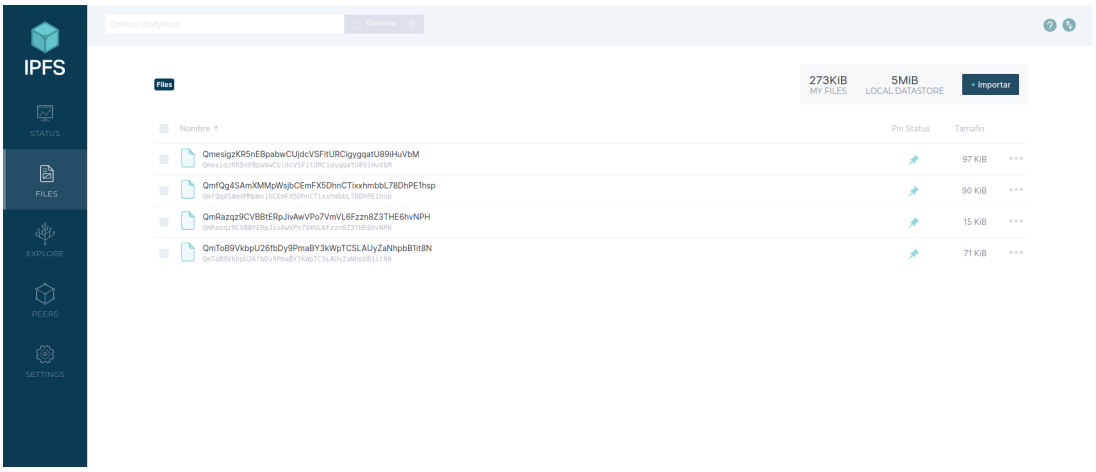

Figure 4: Nodo IPFS con los archivos correspondientes a las distintas skins.

Tras obtener Sepolia ETH se desplegó el Smart Contract y se recogió la dirección para hard codearlo en la aplicación React.

```
direccionTienda: "0x96E9C0C4aF933CD13D47cd41cc711e416b8e55F9",
```

Figure 5: Dirección del Smart Contract hard codeado en la aplicación React.

Tras esto, se desplegó la aplicación haciendo uso de NodeJS. Una vez desplegado, se conectará a la wallet de MetaMask del usuario que esté utilizando la aplicación y se

comprobará si es el usuario que desplegó el Smart Contract. De ser él, se mostrará la ventana de administrador de la web:

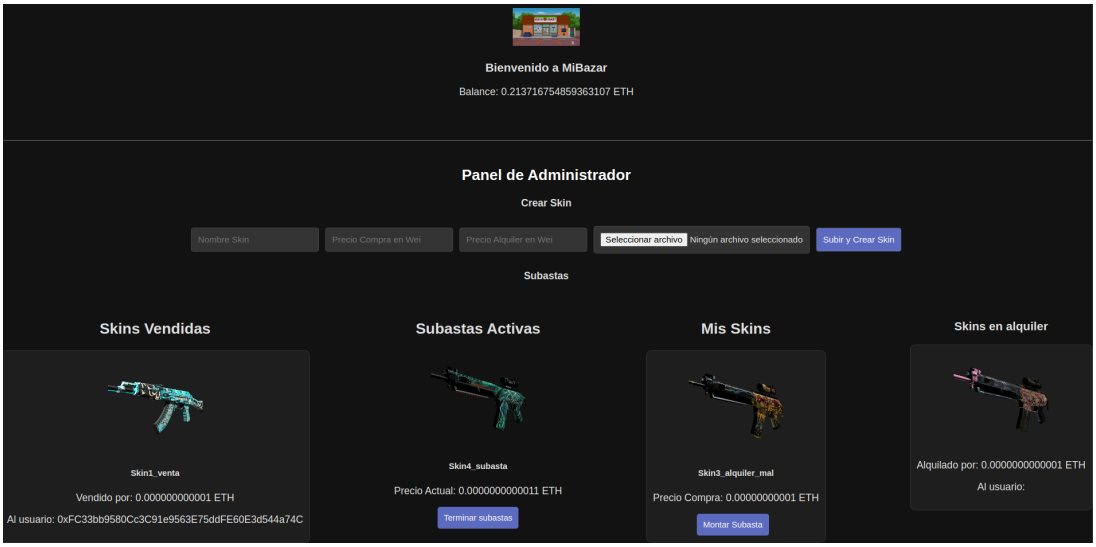

Figure 6: Ventana que se muestra en caso de ser el propietario (administrador) del Smart Contract.

Si se trata de un usuario normal, se mostrará la ventana de usuario, donde se permitirá comprar, alquilar y pujar por skins que estén disponibles en ese momento.

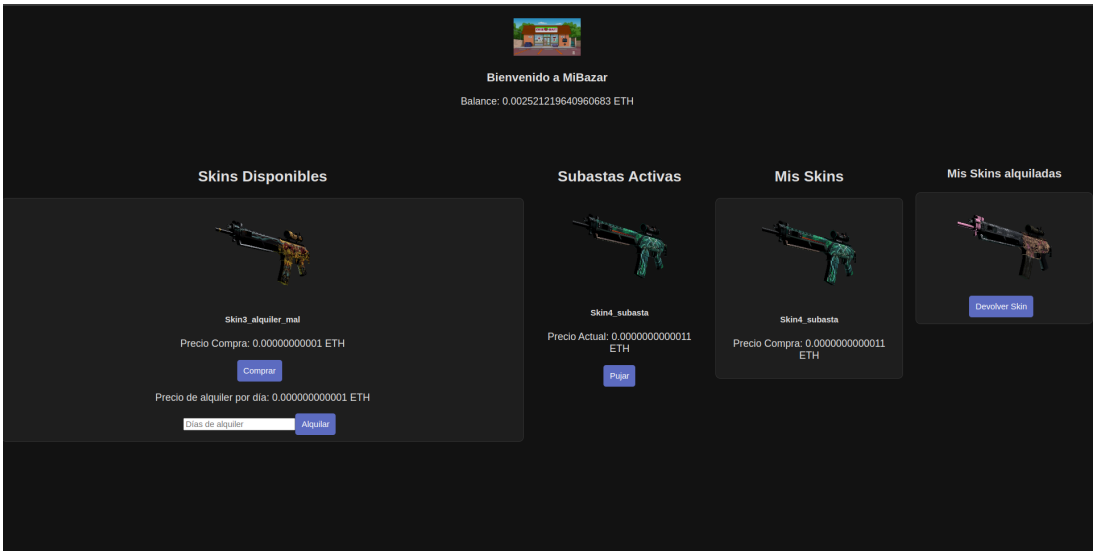

Figure 7: Ventana que se muestra en caso de ser un usuario normal.

Una vez un usuario adquiere una skin, se pagará el ETH al Smart Contract y este será el encargado de enviarlo a la wallet del propietario del contrato.

10 Análisis de los riesgos de seguridad (e.g., SRM)

Como riesgos de seguridad que presenta la aplicación, si un atacante se hiciese con la cuenta del propietario del contrato, podría actuar para su beneficio propio. Esto no es un

riesgo de nuestra propia DApp, si no que es más un riesgo de la seguridad de la wallet que emplee el usuario.

11 Plan de pruebas

En un primer momento, se probó la funcionalidad de crear skins por parte del propietario y a comprarla, revisando que una skin que ya se vendió no se pueda volver a vender, ni alquilar, ni montar una subasta con ella. Además, el Smart Contract garantiza que el comprador tenga suficiente Ether antes de realizar la transacción y que al propietario le llegue el importe correctamente.
Ampliando al funcionalidad, se implementó el sistema de subastas donde los usuarios pueden pujar por una skin que el propietario dictamine. En este caso, se probó lo siguiente:

- Un usuario que puja podrá volver a pujar nuevamente.
- Se asigna correctamente la skin al usuario ganador de la puja tras terminar.
- Los usuarios que pujaron pero que no ganaron, se les devuelve correctamente el dinero.
- El propietario recibe el ETH del pujador ganador.

Para finalizar, se añadió la funcionalidad de alquilar skins. En este caso, las pruebas realizadas fueron las siguientes:

- Un usuario que alquila una skin se le asigna por número de días. Tras el vencimiento del alquiler, este deberá devolverla al propietario. En caso de que se exceda de tiempo se le cobrará el doble por cada día extra.
- Un usuario no puede alquilar una skin que ya está alquilada.
- El propietario no puede montar una subasta de una skin que esté alquilada.

12 Lean Canvas

12.1 Problema

Los mercados tradicionales de skins carecen de transparencia y tienen altas tasas de fraude. Además de falta de monetización justa para los usuarios. Y que la funcionalidad de alquiler de skins es poco común por falta de confianza entre usuarios.

12.2 Segmentos de clientes

- Jugadores interesados en comprar, alquilar o subastar skins.
- Desarrolladores y propietarios de skins que desean una plataforma segura para vender sus activos digitales.

12.3 Propuesta de valor

El proyecto se centra en ofrecer un mercado descentralizado y transparente que asegure la autenticidad de las skins y brinde confianza a los usuarios. Funcionalidades como el alquiler de skins y la automatización y transparencia total en las transacciones lo diferencian de soluciones tradicionales.

12.4 Solución

Plataforma que permita la compra, alquiler y subasta de skins gestionadas con contratos inteligentes. La integración con wallets como MetaMask para facilitar transacciones con Ethereum. Y el uso de IPFS para garantizar la inmutabilidad de los archivos de las skins.

12.5 Canales

Para llegar a los usuarios se incluye una DApp (aplicación descentralizada) con una interfaz intuitiva, además de la idea de promoción en comunidades de videojuegos y redes sociales relacionadas con el sector.

12.6 Ingresos

Tarifas de transacción en la compra, alquiler y subasta de skins. Además será el propietario que despliegue el Smart Contract y que cree las skins para diferentes videojuegos quien obtenga la ganancia de las ventas, alquileres y subastas de sus skins.

12.7 Costos

Estos incluyen las comisiones de gas en Ethereum, los gastos de desarrollo y mantenimiento de la DApp, y la gestión de un nodo IPFS para el almacenamiento de los datos.

12.8 Indicadores clave

Los indicadores clave de éxito son el número de usuarios activos, la cantidad de transacciones realizadas (compras, alquileres y subastas) y el volumen total de ETH movido a través de la plataforma.

12.9 Ventaja diferencial

Principalmente esta radica en su descentralización completa, su transparencia basada en blockchain y la implementación de funcionalidades únicas como el alquiler de skins, poco común en el mercado actual.

13 Aspectos legales

13.1 Propiedad Intelectual y Derecho de Uso

Las skins adquiridas por un usuario se modifican para que el propietario sea el nuevo usuario, permitiendo que una skin solo pueda tener un único propietario. Una vez adquirida una skin, este proceso se lleva a cabo y la skin queda a nombre del nuevo

usuario. El Smart Contract está disponible para que cualquier usuario de la blockchain que quiera desplegarlo e integrarlo en su aplicación.

13.2 Términos del Contrato Inteligente

Una vez el contrato es desplegado, no se puede alterar ninguno de sus valores y operaciones. En lo referente a las compras, alquiler y subastas, todo se gestiona a través de transacciones con la wallet Metamask.

13.3 Aspectos Técnicos

Los archivos pertenecientes a las distintas skins no se encuentran dentro de la blockchain, si no se que se alojan dentro del nodo IPFS. Se garantiza que solamente el usuario que despliegue el Smart Contract pueda acceder a las funcionalidades de administrador, como puede ser el caso de crear una skin, montar subastas o terminar subastas.

13.4 Regulación de Protección al Consumidor

Una vez un usuario puja y otro usuario puja por encima de la suya, el ETH que haya depositado el anterior pujador se le reembolsará, y lo podrá ver reflejado en su wallet personal.

13.5 Consideraciones Financieras

Cada vez que un usuario quiera realizar una acción sobre el contrato, deberá pagar el coste de dicha transacción.
El beneficiario de toda la venta/alquiler/subastas del Smart Contract es la wallet del usuario que lo haya desplegado.

14 Conclusiones

Como conclusión, se ha implementado un Smart Contract basado en un mercado de skins digitales para una aplicación React, donde los usuarios podrán adquirir distintas skins creadas por el propietario. Gracias al nodo IPFS podemos tener distribuido el fichero asociado a la skin para que cualquiera que quiera ver quien es el usuario que tenga asignado, pueda obtener rápidamente el fichero y comprobarlo.
El Smart Contract implementa una lógica que no permite realizar distintas acciones sobre una skin si esta está vendida, alquilada o en una subasta activa. Es decir, antes de cada transacción se comprueba el estado de la skin para evitar errores o comportamientos anómalos, como vender una skin en alquiler o subastar una skin ya vendida, por ejemplo.
De cara a la aplicación React, su implementación permite un acceso amigable y cercano a los usuarios, donde podrán adquirir las distintas skins de forma sencilla.
Es por ello que consideremos haber puesto en práctica multitud de conocimientos de las redes blockchain y los sistemas IPFS y la seguridad que ambos campos llevan implicitas. Hemos aprendido a desplegar contratos inteligentes con Ether de Sepolia usando una wallet de MetaMask, además de securizar estos contratos a prueba de ataques o errores humanos. Además, hemos aprendido a desarrollar una aplicación React para mostrar

las transacciones disponibles por el contrato inteligente de una forma intuitiva para el usuario.
Realmente, los Smart Contract tienen una gran utilidad ya que en el mundo actual muchas de las soluciones y servicios que se ofrecen a la población y sociedad son herramientas centralizadas y con intenciones ofuscadas, en las que es necesario confiar en una entidad central y su buena intención, lo cuál, como vemos más a diario es más difícil. Sin embargo esta solución, ya no tan novedosa, ofrece una descentralización y transparencia por la que cualquier usuario puede adentrarse en un entorno o emplear las herramientas confiando no en una entidad, si no, en toda una red de nodos que confirman el estado de todo objeto o atributo por medio de cadenas de bloques.

15 Líneas futuras

Como posibles mejoras, se podría implementar un oráculo que permita recoger los ingresos de subastas finalizadas, opción que valoramos para la elaboración de esta práctica pero que por tema de tiempo de no pudimos implementar.
Otra buena implementación futura podría ser que los usuarios pudiesen poner a la venta o en subasta sus skins adquiridas, generando un mercado donde los usuarios pueden ser partícipes y favorecer una compra/venta segura y fiable.

16 Lecciones aprendidas

Como hemos comentado en la Sección 14, hemos aprendido especialmente a implementar de forma segura contratos inteligentes, sistemas IPFS sencillos y DApps.
Además hemos adquirido un nuevo enfoque sobre la filosofía de las blockchain y los contratos inteligentes, viendo así su utilidad en el mundo actual en el que es difícil confiar en entidades y personas debido a la globalización y el desarrollo y madurez de Internet.

i. Incidencias
   Ha sido complejo adquirir Ether suficiente para todas las pruebas por lo que en cada una debíamos asegurarnos de que todo lo implementado estuviera correctamente antes de su despliegue.

   Además, debido a la cantidad de modificaciones que fueron necesarias para esta práctica sumado al poco margen de tiempo entre esta y la anterior entrega y teniendo en cuenta las demás asignaturas cursadas, han provocado que no haya sido posible ciertas implementaciones como los oráculos u otras ideas que se han tenido en mente.

17 Tiempo dedicado/esfuerzo

Pedro trabajó aproximadamente 30 horas.
Axel trabajó aproximadamente 30 horas.

18 Referencias

References

[1] Steam Market `https://steamcommunity.com/market/?l=spanish`

[2] CS Money `https://cs.money/es/`

[3] OpenSea `https://opensea.io/es`