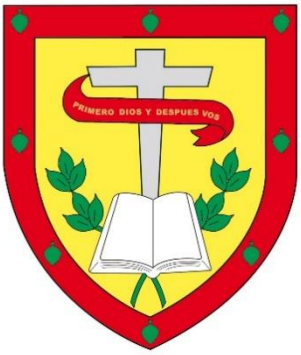


UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

UNIDAD ACADÉMICA DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN E INNOVACIÓN TECNOLOGICA

CARRERA DE SISTEMAS DE INFORMACIÓN

DISEÑO Y DESARROLLO DE UN PROTOTIPO DE UNA PLATAFORMA PARA LA GESTIÓN DE SMART CONTRACTS EN BLOCKCHAIN

TRABAJO DE TITULACIÓN O PROYECTO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS DE INFORMACIÓN

AUTOR: JOHN BRAYAN RIVERA PALAGUACHI

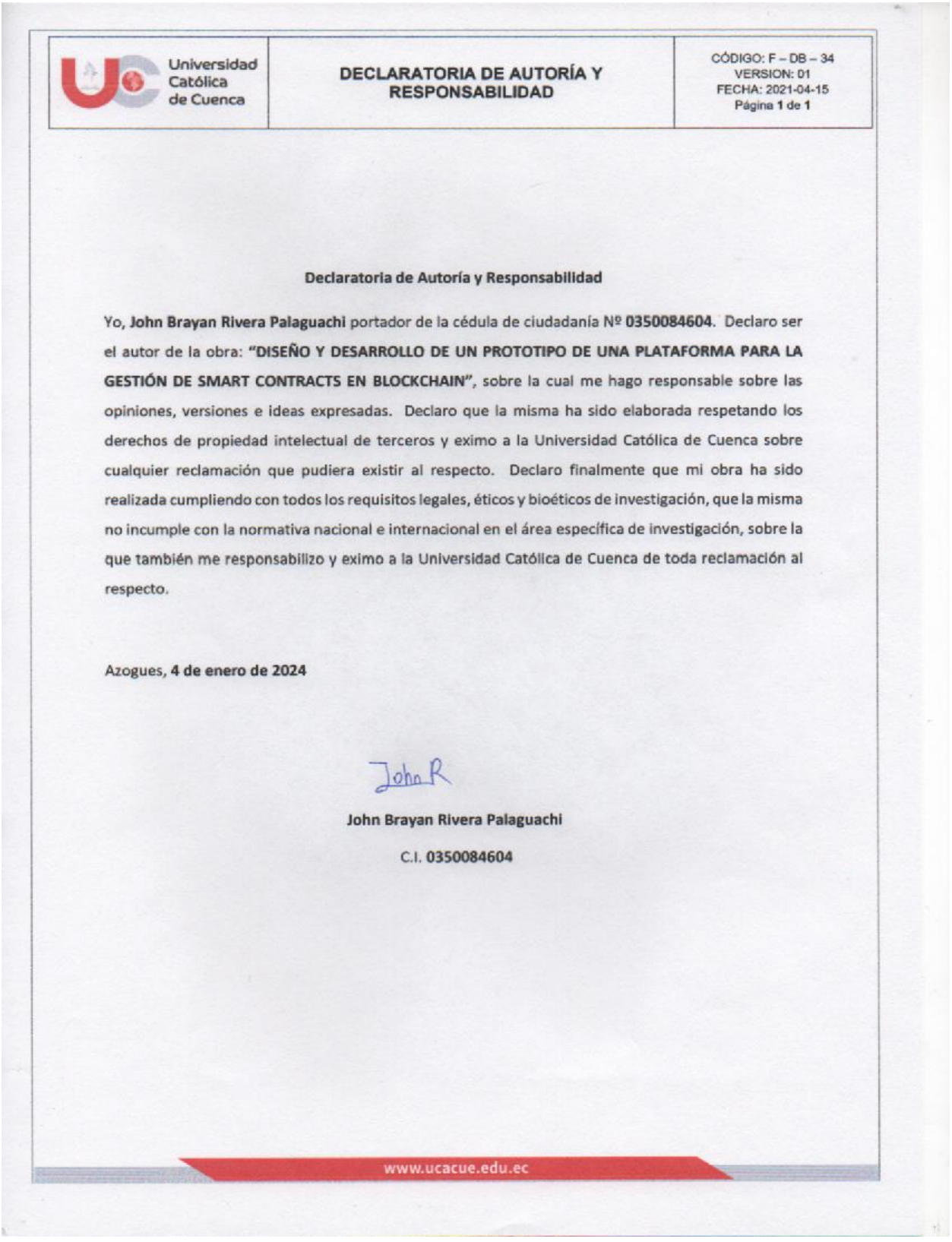
DIRECTOR: ING. ANDRÉS SEBASTIÁN QUEVEDO SACOTO, Mgs.

AZOGUES - ECUADOR

2024

DIOS, PATRIA, CULTURA Y DESARROLLO

DECLARATORIA DE AUTORIA Y RESPONSABILIDAD



Universidad Católica de Cuenca	DECLARATORIA DE AUTORÍA Y RESPONSABILIDAD	CÓDIGO: F – DB – 34 VERSION: 01 FECHA: 2021-04-15 Página 1 de 1

Declaratoria de Autoría y Responsabilidad

Yo, **John Brayan Rivera Palaguachi** portador de la cédula de ciudadanía Nº **0350084604**. Declaro ser el autor de la obra: **"DISEÑO Y DESARROLLO DE UN PROTOTIPO DE UNA PLATAFORMA PARA LA GESTIÓN DE SMART CONTRACTS EN BLOCKCHAIN"**, sobre la cual me hago responsable sobre las opiniones, versiones e ideas expresadas. Declaro que la misma ha sido elaborada respetando los derechos de propiedad intelectual de terceros y eximo a la Universidad Católica de Cuenca sobre cualquier reclamación que pudiera existir al respecto. Declaro finalmente que mi obra ha sido realizada cumpliendo con todos los requisitos legales, éticos y bioéticos de investigación, que la misma no incumple con la normativa nacional e internacional en el área específica de investigación, sobre la que también me responsabilizo y eximo a la Universidad Católica de Cuenca de toda reclamación al respecto.

Azogues, **4 de enero de 2024**

John R

John Brayan Rivera Palaguachi

C.I. **0350084604**

www.ucacue.edu.ec

CERTIFICACION DEL DIRECTOR DE TESIS

CERTIFICACIÓN DEL DIRECTOR DE TESIS

Andrés Sebastián Quevedo Sacoto

DOCENTE DE LA CARRERA DE INGENIERIA EN SISTEMAS DE INFORMACION

De mi consideración:

Certifico que el presente trabajo de titulación denominado: **"DISEÑO Y DESARROLLO DE UN PROTOTIPO DE UNA PLATAFORMA PARA LA GESTIÓN DE SMART CONTRACTS EN BLOCKCHAIN"**, realizado por: **John Brayan Rivera Palaguachi**, con documento de identidad: **0350084604,** previo a la obtención del título de **Ingeniero en Sistemas de Información** ha sido asesorado, orientado, revisado y supervisado durante su ejecución, bajo mi tutoría en todo el proceso, por lo que certifico que el presente documento, fue desarrollado siguiendo los parámetros del método científico, se sujeta a las normas éticas de investigación que exige la Universidad Católica de Cuenca, por lo que está expedito para su presentación y sustentación ante el respectivo tribunal.

Azogues, 4 de enero de 2024

Andrés Sebastián Quevedo Sacoto

C.I 0301826434

DIRECTOR

AGRADECIMIENTO

En primer lugar, deseo expresar mi más sincero agradecimiento a mis padres, a mis hermanos, a mi leal compañero felino, asi como a todos mis familiares, cuyo constante respaldo y apoyo han sido pilares fundamentales a lo largo de mi travesía universitaria.

En segundo término, deseo extender mi gratitud hacia mi tutor, el Ingeniero Sebastián Quevedo, cuya invaluable orientación y asistencia han sido cruciales en el desarrollo de mi trabajo de titulación. A pesar de las dificultades encontradas en el transcurso, su confianza inquebrantable en mi capacidad fue un impulso determinante para alcanzar el éxito.

Por último, pero menos importante, deseo expresar mi reconocimiento a mis compañeros de estudios y al cuerpo docente del programa de Ingeniería en Sistemas de Información. Compartir este camino de conocimiento junto a ellos ha sido enriquecedor y memorable, y su colaboración y enseñanzas han sido fundamental en mi formación académica.

DEDICATORIA

Dedico este trabajo de titulación a mis pilares fundamentales: mis padres, **Cristhian Rivera** y **Sandra Palaguachi**, wuienes han sido el faro y mi mayor apoyo a lo largo de esta travesía académica. Su constante aliento y sacrificio han sido motor que impulsa este logro.

A mis hermanos, **Cristhian Rivera** y **Marco Rivera**, quienes con su inmenso cariño y alegría han sido una fuente inagotable de motivación durante mis estudios universitarios.

A mi tía, **Maria Palaguachi,** quien ha sido mi fuente de inspiración y mi guía en la búsqueda de la excelencia académica. Su ejemplo y aliento han sido pilares que han sostenido mi camino hacia la superación personal.

A **Rayo**, mi gato fiel compañero felino, por su presencia reconfortante durante largas horas de estudio.

A mi **familia**, cuyo amor y comprensión han sido fundamentales para mi desarrollo académico y personal.

Y a mis **amigos**, por su invaluable compañía y por entender los sacrificios necesarios para alcanzar la meta.

Este trabajo de titulación es el fruto del esfuerzo conjunto de todos ustedes. Gracias por ser mi soporte y por inspirarme a alcanzar lo mejor de mi en este viaje hacia la graduación.

DISEÑO Y DESARROLLO DE UN PROTOTIPO DE UNA PLATAFORMA PARA LA GESTIÓN DE SMART CONTRACTS EN BLOCKCHAIN

John Brayan Rivera Palaguachi – Ing. Andrés Sebastián Quevedo Sacoto, Mgs. Universidad Católica de Cuenca – john.rivera.04@est.ucacue.edu.ec

RESUMEN

El presente trabajo se centra en el diseño y creación de un prototipo de plataforma que tiene como finalidad la gestión de Smart Contracts en la tecnología Blockchain, con el objetivo principal de simplificar el proceso de generación, ejecución y supervisión de Smart Contracts en un entorno descentralizado y seguro.

El prototipo se ha desarrollado tomando como base un Marketplace de vehículos, proporcionando una dimensión práctica y aplicada al trabajo.

Los Smart Contracts desempeñan un papel fundamental en esta plataforma, siendo desplegados en la Blockchain de Ethereum para automatizar transacciones relacionadas en la compra y venta de vehículos. El enfoque no se limita únicamente a la implementación técnica de los Smart Contracts, sino que también incluye la creación de una DApp con una interfaz de usuario intuitiva, facilitando la interacción de los usuarios con la plataforma

Además, se consideran aspectos fundamentales de seguridad y privacidad para garantizar la integridad de las transacciones y la confidencialidad de los datos de los usuarios.

El trabajo se destaca por su enfoque principal en el diseño y desarrollo de una plataforma especializada en la gestión de Smart Contracts en la tecnología Blockchain, con especial atención a su aplicación en un Marketplace de vehículos.

El propósito final de este proyecto es demostrar la viabilidad y el potencial de esta tecnología Blockchain en un entorno practico y funcional, contribuyendo al avance y comprensión de la aplicación de Smart Contracts en casos específicos, como el sector de vehículos.

Palabras Claves: Blockchain, Smart Contracts, Marketplace, DApp, Vehículos

DESIGN AND DEVELOPMENT OF A PROTOTYPE OF A PLATFORM FOR THE MANAGEMENT OF SMART CONTRACTS IN BLOCKCHAIN

John Brayan Rivera Palaguachi - Andrés Sebastián Quevedo Sacoto, Eng. Mag. Catholic University of Cuenca - john.rivera.04@est.ucacue.edu.ec

ABSTRACT

This work focuses on designing and creating a prototype platform that aims to manage Smart Contracts in Blockchain technology, with the primary objective of simplifying the generation, execution, and monitoring of Smart Contracts in a decentralized and secure environment.

The prototype has been developed based on a vehicle Marketplace, providing a practical and applied dimension to the work.

Smart Contracts play a fundamental role in this platform, being deployed on the Ethereum Blockchain to automate related transactions in the buying and selling vehicles. The approach is not limited to the technical implementation of Smart Contracts but also includes the creation of a DApp with an intuitive user interface, facilitating users' interaction with the platform.

In addition, fundamental security and privacy aspects are considered to ensure the integrity of transactions and the confidentiality of user data.

This research mainly focuses on designing and developing a platform specialized in managing Smart Contracts in Blockchain technology, with particular attention to its application in a vehicle marketplace.

The final goal of this project is to demonstrate the viability and potential of this Blockchain technology in a practical and functional environment, contributing to the advancement and understanding of the application of Smart Contracts in specific cases, such as the vehicle sector.

Keywords: Blockchain, Smart Contracts, Marketplace, DApp, Vehicles

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

LISTA DE ANEXOS

CAPÍTULO 1

1. MARCO REFERENCIAL

1.1 Introducción

La gestión de Smart Contracts (Contratos Inteligentes) en Blockchain (Cadena de Bloques) es un campo de estudio fascinante y de creciente importancia en la era actual. A gran medida, dicha tecnología ha ganado prominencia en diversos sectores, los Smart Contracts se han convertido en un pilar fundamental para la automatización, la transparencia y la seguridad de las transacciones.

Simultáneamente, en el ámbito empresarial, lideres de la industria automotriz como el grupo alemán Wolkswagen, han establecido colaboraciones con empresas como Minespider para aprovechar las ventajas de la tecnología Blockchain en el abastecimiento de minerales y el seguimiento de su procedencia. Hyundai también, se encuentra inmerso en el desarrollo de una aplicación basada en tecnología Blockchain que facilitara la conexión entre teléfonos móviles y diversas funciones de los vehículos. Ford, por su parte, está enfocado en la protección de datos de transacciones y la información de los pasajeros en vehículos autónomos, mientras que Toyota, en alianza con Ludicity, se esfuerza por reducir fraudes en la compra de anuncios digitales, contribuyendo a aumentar la transparencia en los procesos publicitarios. [1]

En el marco de este proyecto, se profundizará el estudio de la gestión de Smart Contracts, analizando cómo esta tecnología emergente ha cambiado la gestión de activos, la ejecución de contratos en línea y los negocios. A través de un Marketplace (Mercado en Línea) dedicado a la compra y venta de vehículos, nuestra investigación se enfocará en comprender las bases teóricas de los Smart Contracts y la tecnología Blockchain, asi como su aplicación en el mercado

automotriz.

En contexto local, la mayoría de las personas han tenido que enfrentar el desafío de adquirir o vender un vehículo en algún momento, una experiencia que con frecuencia está marcada por la incertidumbre y el estrés. La proliferación de plataformas especializadas en forma de Marketplaces, donde los interesados pueden explorar una amplia variedad de opciones acorde a sus preferencias, ha modernizado la búsqueda de un vehículo adecuado. Sin embargo, después de inspeccionar el vehículo y acordar un precio, el comprador y el vendedor se enfrentan a la pregunta de cómo completar la transacción de manera segura y eficiente. [2]

Por lo que todo el mundo ha experimentado en algún momento el desafío de comprar o vender un vehículo, una experiencia que conlleva una sensación que genera duda y estrés. La búsqueda de un vehículo adecuado se moderniza mediante plataformas especializadas en forma de Marketplace donde los interesados pueden explorar una amplia gama de opciones según sus preferencias. Sin embargo, cuando el comprador y vendedor, después de inspeccionar el vehículo y acordar un precio, inevitablemente surge la pregunta de cómo completar la transacción. Este paso se ve empañado por desconfianza, que a menudo plantea un dilema para ambas partes. [2]

Para afrontar este desafío, las nuevas tecnologías están desempeñando un papel clave en la remodelación del proceso tradicional en la compra y venta de vehículos. La evolución hacia los mercados digitales ha brindado a los usuarios una amplia gama de oportunidades para explorar, comparar y seleccionar vehículos. Sin embargo, las cuestiones de seguridad y confiabilidad en el comercio siguen siendo importantes. Los compradores no están seguros de cuándo y cómo se realizarán los pagos, y los vendedores están preocupados por garantizar que los compradores puedan cumplir con sus obligaciones financieras. [2]

En este escenario, la tecnología Blockchain y los Smart Contracts demuestran ser soluciones prometedoras para abordar los desafíos en el mercado automotriz. La capacidad de la tecnología Blockchain para crear registros inmutables y transparentes, y la automatización precisa que permiten los Smart Contracts, podrían ser la clave para recuperar la confianza en este proceso. La Blockchain de Ethereum en particular se destaca por su versatilidad y sofisticación en la ejecución de Smart Contracts. [3]

En esta investigación se propone analizar la integración de la tecnología Blockchain de Ethereum[1] y los Smart Contracts en el contexto del mercado automotriz. A través de un enfoque que abarque aspectos técnicos, comerciales y regulatorios, comprender como estas tecnologías resuelven problemas arraigados en el proceso de compra y venta de vehículos y la ayuda a construir un mercado más eficiente y seguro.

1.2 Planteamiento del problema

En la actualidad, el proceso de compra y venta de vehículos enfrenta numerosos obstáculos y limitaciones incluida la falta de transparencia, los costos adicionales causados por intermediarios, los riesgos de fraude y la documentación inadecuada. Además, los sistemas de registros y transacciones convencionales en el mercado automotriz suelen ser ineficientes y propensos a errores.

La tecnología Blockchain, con su capacidad para garantizar la integridad de los datos y las transacciones mediante registros distribuidos e inmutables, junto con la utilización de Smart Contracts, que permiten la ejecución automática y transparente de acuerdos, ofrece una solución potencial para estos problemas. Sin embargo, a pesar del crecimiento y adopción de la tecnología

[1] Ethereum: Plataforma de código abierto utilizada para la ejecución de Smart Contracts

Blockchain en varios campos, su aplicación específica para la compra y venta de vehículos, respaldada por Smart Contracts en la plataforma Ethereum, aún no ha sido completamente explorada y desarrollada. Por lo tanto, la carencia de una solución efectiva y eficiente que combine tecnologías Blockchain y Smart Contracts en la plataforma Ethereum para gestionar transacciones de compra y venta de vehículos es evidente. La pregunta clave es, si esta combinación de tecnologías puede resolver los desafíos actuales del mercado automotriz, mejorando la transparencia, seguridad, eficiencia y reduciendo la necesidad de intermediarios.

Para facilitar y mejorar los procesos de compra y venta de vehículos, este proyecto se enfocará en diseñar y desarrollar un prototipo de una plataforma que utilice Smart Contracts sobre la Blockchain de Ethereum. Esto contribuirá a la búsqueda de soluciones innovadoras en este sector.

1.3 Justificación

La presente investigación se centra en hacer accesible y fácil de usar la tecnología de Smart Contracts en Blockchain, con el objetivo de que pueda ser adoptada de manera más amplia por empresas y organizaciones. Eso se traduce en una mayor eficiencia en los procesos empresariales, una mayor seguridad y privacidad en las transacciones, asi como una mayor confianza en la tecnología Blockchain. [4]

Además, el desarrollo de herramientas de gestión adecuadas para la gestión de Smart Contracts también puede reducir el riesgo de errores de programación y vulnerabilidades de seguridad, por lo que el proyecto se va a basar en el mercado automotriz debido a las dificultades que hay al momento de conseguir un vehículo. [5]

El mercado automotriz requiere soluciones innovadoras para abordar las limitaciones

actuales en términos de transparencia y seguridad de las transacciones. La tecnología Blockchain, y los Smart Contracts ofrecen un potencial prometedor para superar estos desafíos. Al proporcionar un registro inmutable y transparente de la información del vehículo, además de automatizar y garantizar la ejecución de acuerdos entre compradores y vendedores, puede aumentar la confianza y la eficiencia en el mercado automotriz. [5]

1.4 Objetivos

1.4.1 Objetivo General

Diseñar y desarrollar un prototipo de una plataforma de gestión, para la compra y venta de vehículos, mediante el uso de Smart Contracts sobre la Blockchain de Ethereum.

1.4.2 Objetivos Específicos

- Investigar y analizar la problemática propuesta, revisando investigaciones y proyectos previos relacionados con la tecnología Blockchain, con el propósito de demostrar de manera íntegra la aplicabilidad y relevancia de esta tecnología.
- Desarrollar un prototipo funcional de la plataforma, diseñada específicamente para la compra y venta de vehículos, con el objetivo de generar un software adaptado y orientado a satisfacer las necesidades específicas de los usuarios y del entorno del Marketplace de vehículos.
- Integrar e implementar la plataforma, a través de la Blockchain de Ethereum, asegurando la trazabilidad completa de cada vehículo, que permita garantizar la transparencia y confiabilidad de las transacciones.
- Evaluar y optimizar el rendimiento de la plataforma, a través de un informe final

del entregable, con el fin de determinar si el prototipo cumple con los requerimientos.

1.5 Alcance

El desarrollo del presente trabajo de investigación es crear un prototipo funcional de un Marketplace de vehículos utilizando tecnología Blockchain y Smart Contracts. Todo esto se llevará a cabo en un ambiente de prueba local en lugar de una implementación a gran escala. El prototipo se enfocará en demostrar cómo se pueden aplicar las tecnologías y cómo funciona la solución propuesta en un entorno particular. Por ello esta versión tiene ciertas restricciones, como la ausencia de un login, la conversión de criptomonedas a dólares y el despliegue en una Blockchain de entorno real.

Conceptos Relacionados

- **Blockchain:** Es una lista ordenada de bloques de transacciones con enlaces hacia atrás, es decir, los bloques posteriores se vinculan con los bloques anteriores de la cadena. Cada bloque se identifica mediante un hash SHA-256, el cual se ubica en la cabecera del bloque. Cada bloque hace referencia a un bloque anterior a través de un campo llamado "hash de bloque anterior". Es decir, cada bloque contiene un hash de su bloque padre dentro de su propia cabecera. El primer bloque de una Blockchain es llamado génesis y a partir de éste, se generan nuevos bloques incorporados a la cadena y su red es descentralizada; es decir no existe un nodo central que se encargue de manejar toda la información. [6]

- **Ethereum:** Es una plataforma de Blockchain con una computadora integrada lo cual es la base para crear aplicaciones y organizaciones de manera descentralizada.

Hay una sola computadora (EVM) cuyo estado está de acuerdo con todos en la red de Ethereum. Todos los que participan en la red Ethereum guardan una copia del estado de esta computadora. [7]

- **Smart Contract:** Es un programa que se ejecuta en una Blockchain de Ethereum. Es una colección de código y datos que reside en una dirección especifica en la Blockchain de Ethereum. Son un tipo de cuenta Ethereum lo que significa que tienen saldo y pueden ser objetivo de transacciones. [8]

1.6 Trabajos relacionados

En el año 2020, se publicó un artículo acerca de una arquitectura de microservicios para la plataforma de compra en línea de una plataforma denominada "¡ala orden!". Este enfoque arquitectónico surgió como respuesta a las circunstancias globales que surgieron a raíz de la pandemia del SARS-CoV-2, lo que genero un notorio auge en el sector del comercio electrónico. como resultado de esta estrategia, se implementó un modelo de microservicios que demostró ser altamente eficaz en lo que respecta al desarrollo, mantenimiento y despliegue de la plataforma. Además, este enfoque de diseño se presentó como una opción que proporciona agilidad, capacidad de escalabilidad y autonomía en el proceso de creación del software. [9]

En el año 2021, se realizó el desarrollo de un sistema de seguridad con Blockchain en una página web, lo que resulto como una solución que aprovecha las ventajas que proporciona esta tecnología, el uso de Blockchain y Smart Contracts permiten agregar una capa adicional de seguridad a una aplicación web y por ende brinda protección a la información y activos de individuos y organizaciones, lo cual se probó que se previno ataques comunes como XSS e inyección SQL, lo que con ayudade herramientas como Vooki y Owasap Zap, lo que resulto en

reducción del 100% en vulnerabilidades altas y un 80% en las vulnerabilidades medias y bajas. [10]

En el mismo año, se realizó el desarrollo de un modelo de sistematización en la gestión de derechos de autor aplicando Smart Contracts en Blockchain, lo que resulto que la tecnología Blockchain es factible para ser utilizada como base de aplicaciones basada en Smart Contracts ayudando a realizar la gestión de derechos de autor para artículos científicos. Su modelo puede gestionar los derechos de autor mediante el uso de Smart Contracts permitiendo el registro, validación, consulta y reconocimiento de un artículo científico. [11]

En el año 2022, se realizó el análisis y desarrollo de una aplicación de registro de permisos y ausentismos en Blockchain mediante Smart Contracts y usando herramientas compatibles para entornos Linux y Windows, investigando sobre la tecnología y aprendiendo más sobre el lenguaje de programación Solidity, lo que resulto demostrar la capacidad que puede tener la tecnología Blockchain junto con los Smart Contracts para brindar seguridad a los datos que llegan almacenarse en la testnet de Ethereum y a su vez la certeza que tienen los Smart Contracts. [12]

En el año 2023, se realizó el desarrollo de un prototipo de aplicación web para la gestión de historias geriátricas utilizando Smart Contracts en Blockchain, lo que resultó ser un gran aporte para el sector de la salud, garantizando la seguridad y privacidad de los datos de los pacientes. Además, los Smart Contracts, fueron desarrollados considerando la que información sea inmutable y para ello se validaron en 2 aspectos de seguridad como: perfiles de acceso y autentificación de usuario. [13]

En el mismo año, se publicó un artículo que se centró en la tecnología innovadora de Blockchain y su potencial contribución a la economía popular y solidaria, tal como se describe en

el artículo 283 en la Constitución de la Republica del Ecuador. Como resultado de esta investigación, se identificaron oportunidades para la aplicación de esta tecnología en diversas organizaciones que forman parte de la economía popular solidaria. En particular, se propuso una solución dirigida a las organizaciones cooperativas de producción de cacao, que involucra el registro de productores y la trazabilidad de la cadena de producción. [14]

En el análisis de estos estudios, se observa que las tecnologías que se utilizaran en el proyecto, como los Smart Contracts y la Blockchain, han sido implementadas de diversas maneras. Lo que se buscó en la mayoría de estos trabajos es su enfoque en la seguridad y privacidad de los datos de las personas, la trazabilidad de las transacciones, la programación mediante el lenguaje de Solidity, la conexión de los Smart Contracts con diferentes frontends y la configuración en la Blockchain.

A pesar de los desafíos inherentes a la novedad de estas tecnologías, las cuales aún no han alcanzado su pleno apogeo, lo que se traduce en una limitada disponibilidad de información y trabajos relacionados, y su continua evolución, la mayoría de las aplicaciones desarrolladas han logrado alcanzar sus objetivos. Estos avances no solo contribuyen a la innovación tecnológica, sino que también promueven el uso más amplio y efectivo de estas tecnologías.

CAPÍTULO 2

2. MARCO TEÓRICO

2.1 Marco Metodológico

En el desarrollo de software, es esencial emplear una metodología, que es un marco de trabajo utilizado para guiar el proceso de creación de aplicaciones. Varios métodos han surgido a lo largo del tiempo, como: Scrum, Kanban y Extreme Programming XP son algunos de los que han tenido un impacto en aspectos como el análisis de requerimientos y la entrega final del software. Como resultado, es necesario comparar las metodologías para determinar la mejor opción para el desarrollo del sistema.

Tabla I. Comparación de Metodologías Agiles

ASPECTO	SCRUM	KANBAN	XP
ROLES CLAVE	Product Owner, Scrum Master, Scrum Team	No especifica roles	Programadores, Cliente, coach XP
PLANIFICACIÓN	Basado en Sprints	Sin Sprints	Basado en iteraciones más cortas
PRIORIZACIÓN	Product Backlog	WIP	Historias de usuario y valores comerciales
REUNIONES REGULARES	Daily Scrum, Sprint Review y Sprint Retrospective	No requiere funciones regulares	Planificación de juego, reuniones diarias de programación en pareja
ENTREGA	Al final de cada Sprint	Entrega continua	Al final de cada iteración
FLEXIBILIDAD	Cambios permitidos después de un Sprint, pero no durante el mismo	Cambios permitidos en cualquier momento	Cambios permitidos en cualquier momento; se enfatiza la comunicación cercana con el cliente

Fuente: Autor

Después de analizar la comparativa, se opta por la metodología Scrum es la opción más adecuada para avanzar en el desarrollo del prototipo. Esto se debe a su capacidad para adaptarse a cambios, lo que la convierte en la mejor opción para la creación de prototipos. Además, ofrece la posibilidad de realizar revisiones regulares para adaptarse a los cambios necesarios.

2.2 Scrum

Scrum se configura como un marco que posibilita el desarrollo de software al facilitar a individuos, equipos y organizaciones, a través de un entorno colaborativo altamente funcional, flexible y adaptable al cambio, la creación de soluciones adaptables para problemas complejos. Lo logra mediante la entrega periódica y parcial del producto final. [15]

Scrum se sustenta en dos pilares fundamentales: el empirismo y la filosofía Lean. El empirismo sostiene que el conocimiento surge de la experiencia y de la toma de decisiones basada en observaciones. Mientras que el pensamiento Lean se concentra en la reducción del desperdicio y la focalización en aspectos esenciales. [15]

Para alcanzar sus objetivos, Scrum emplea un enfoque iterativo e incremental con el propósito de mejorar la previsibilidad y gestionar los riesgos de manera efectiva. Además, involucra a grupo de personas que, en conjunto, reúnen todas las habilidades y experiencias necesarias para llevar a cabo la tarea, y promueve la compartición o adquisición de habilidades según sea necesario. [15]

2.2.1 Ciclo de vida de Scrum

El ciclo de vida en Scrum se basa en la realización de iteraciones llamadas sprints. Estos son ciclos cortos y repetitivos con duración de 2 a 4 semanas fijas durante los cuales se desarrolla, prueba y entrega un incremento del producto. [15]

A continuación, se puede observar en la Figura 1, como se maneja un ciclo de vida de Scrum:

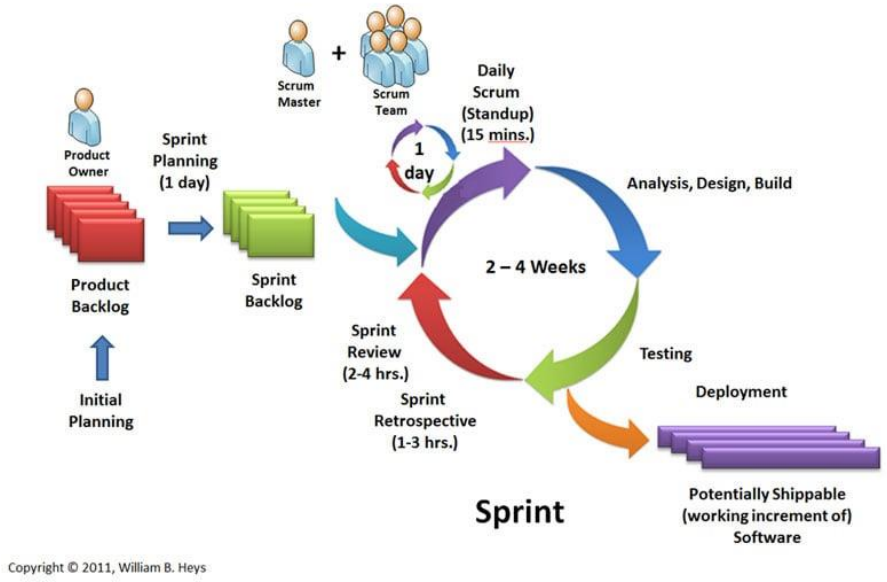


Fig 1. Ciclo de Vida de Scrum
Fuente: [16]

2.2.2 Scrum Team

Scrum fomenta un entorno que está formado por tres roles:

- **Product Owner:** Es el responsable de maximizar el valor del producto resultante del equipo Scrum y ordena el trabajo en un Product Backlog. [15]
- **Scrum Master:** Es el responsable de establecer Scrum, lo consigue ayudando a todos a comprender la teoría y la práctica de Scrum, también convierte una selección del trabajo en un Increment durante un Sprint. [15]
- **Desarrolladores:** Son las personas del equipo Scrum que se comprometen a crear cualquier aspecto de un Increment en cada sprint. [15]

2.2.3 Eventos de Scrum

Scrum fomenta cinco eventos:

- **Sprint:** Es un periodo de tiempo fijo, generalmente de 2 a 4 semanas en el que se desarrolla un incremento del producto. [15]
- **Planificación de Sprint:** El scrum team colabora para definir los elementos del backlog del producto que se trabajara en el sprint. [15]
- **Daily Scrum:** Son sesiones diarias de 15 minutos donde los desarrolladores sincronizan su trabajo y discuten los avances, obstáculos y la planificación para el siguiente día. [15]
- **Sprint Review:** Al finalizar un sprint, los desarrolladores muestran lo que han logrado y reciben comentarios de los stakeholders para ajustar prioridades. [15]
- **Sprint Retrospective:** Es una sesión en la que el scrum team reflexiona sobre el sprint pasado, identifica mejoras y planifica ajustes para el siguiente sprint. [15]

2.2.4 Artefactos de Scrum

Scrum fomenta tres artefactos:

- **Product Backlog:** Es una lista priorizada de elementos que describen las características, funciones, mejoras y correcciones que se desean para el producto. [15]
- **Sprint Backlog:** Los elementos seleccionados del backlog de producto que el equipo se compromete a completar durante el sprint actual. [15]
- **Increment:** El resultado tangible del sprint, que incluye todas las historias de usuario y tareas completadas durante el sprint. [15]

2.3 Blockchain

En el año 2008 se publicó un artículo titulado “Bitcoin: A Peer-to-Peer Electronic Cash System” por una persona o grupo de personas bajo el seudónimo de “Satoshi Nakamoto”. En el documento se presentó la idea de un sistema digital descentralizado llamado bitcoin, permitiendo que las personas puedan enviar su dinero entre ellas sin la necesidad de una institución financiera, que utilizaba una cadena de bloques (Blockchain) para registrar todas las transacciones realizadas por criptomonedas. El lanzamiento de bitcoin como software de código abierto ocurrió en enero de 2009, fue el primer uso de la tecnología Blockchain. Fue diseñada para mantener un registro de todas sus transacciones. [17]

Blockchain es una tecnología revolucionaria de registro distribuido, representada por una cadena de bloques en expansión constante. En esta cadena, cada bloque contiene un conjunto de datos o transacciones y se encuentra interconectado de manera secuencial. Estos bloques se almacenan en una red de computadoras descentralizadas que operan bajo el modelo P2P, y esta estructura se sustenta en principios criptográficos que aseguran la integridad y transparencia de la información registrada. [18]

En esencia, Blockchain proporciona una base de datos distribuida que se basa en una secuencia de bloques en constante crecimiento. Estos bloques, aunque públicos, mantienen un nivel de anonimato ya que los usuarios se identificación mediante direcciones seudónimas en lugar de revelar sus datos personales. Esta combinación de apertura y anonimato establece una sólida base de confianza para quienes hacen uso de esta tecnología. [18]

Como resultado Blockchain aporta solidez, seguridad y transparencia a la gestión de información y datos, una ventaja invaluable en un mundo profundamente globalizado y

digitalizado. [18]

Hoy en día, se clasifica como una de las Tecnologías de Registro Distribuidas (Distribuited Ledger Technologies, DLT), siendo predecesoras de esta.

Las aplicaciones de Blockchain principalmente se dieron en el sector financiero, los ejemplos más comunes es el de las criptomonedas. Pero su aplicabilidad no se detiene que se ha utilizado en diversas aplicaciones como: el uso de Smart Contracts, el seguimiento en cadenas de suministro, en la validez de certificados educativos, sistemas de votaciones, entre otras.

2.3.1 Estructura de un bloque

Un bloque en una Blockchain tiene una estructura especifica que incluye varios componentes clave. A continuación, como se describe en la Figura 2 que muestra la información que contiene cada bloque en la Blockchain:

- **Hash de bloque previo:** Es el valor que permite que los bloques se vinculen entre si formando una cadena. [18]
- **Timestamp:** Es la marca de tiempo que permite identificar cuando fue creado el bloque. [18]
- **Nonce:** Es el valor encontrado durante el proceso de minado. [18]
- **Hash árbol Merkle:** Es el valor hash que representa todas las transacciones incluidas en el bloque. Este valor se utiliza para verificar la integridad de las transacciones en el bloque. [18]
- **Información:** El bloque contiene la información en sí. [18]

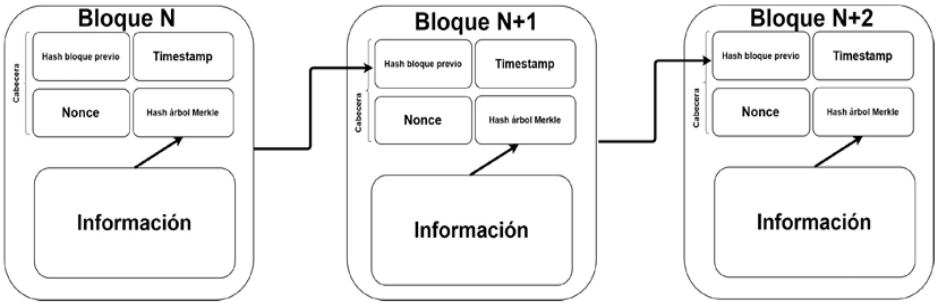


Fig 2. Estructura de un Bloque

Fuente: [18]

2.3.2 Tipos de Blockchain

Cuando hablamos de Blockchain debemos tener en cuenta que es una tecnología que existe en más de un tipo. Y la diferencia entre esos tipos se encuentra en su funcionalidad, las reglas para validar su información, sus protocolos de consenso y la flexibilidad de la administración. A continuación, como se muestra en la Figura 3 vamos a definir los tipos de Blockchain:

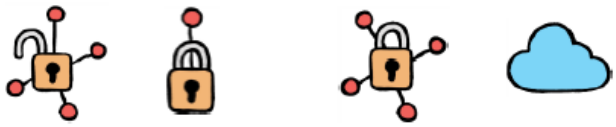
- **Blockchain Publicas:** Las redes de Blockchain Publicas son accesibles para cualquier individuo. Por lo general, estas redes se caracterizan por su transparencia, donde los usuarios mantienen el anonimato. Es estas redes, ningún participante ostenta un nivel superior de autoridad que los demás, por lo cual no existen administradores de la red. [19]

- **Blockchain Federadas:** Las redes de Blockchain Federadas son altamente solicitadas cuando se trata de desarrollar soluciones compartidas para gobiernos, empresas y asociaciones. En general, no están disponibles para el público en general. En cambio, un grupo de organizaciones, entidades o empresas colabora en la gestión de la red y se asegura de mantener registros sincronizados. [19]

- **Blockchain Privadas:** Las redes de Blockchain Privadas se caracterizan por estar

bajo el control exclusivo de una sola entidad, que asume la responsabilidad de mantener la cadena de bloques, otorgar permisos a los usuarios seleccionados, proponer transacciones y aprobar la incorporación de bloques. En esencia, son similares a las redes federadas, pero en este caso, solo una entidad tiene control y toma decisiones. [19]

- **Blockchain como Servicio:** Grandes empresas brindan soluciones basadas en Blockchain en entornos de nube. Estos servicios van más allá del simple almacenamiento de datos Blockchain; también ofrecen ventajas como un refuerzo en la seguridad, la eliminación de la necesidad de invertir en infraestructura de hardware y la posibilidad de un entorno de trabajo más intuitivo. [19]



	Públicos Bitcoin, Ethereum, Litecoin	**Privados** Hyperledger, Corda, Quorum	**Federados** Hyperledger, Corda, Quorum	***Blockchain as a Service*** IBM, Microsoft, Amazon
Cualquiera puede participar	✓	✗	✗	NA
Los participantes actúan, en general, como nodos	✓	✗	✗	NA
Transparencia	✓	≈	≈	NA
Hay un único administrador	✗	✓	✗	NA
Hay más de un administrador	✗	✗	✓	NA
No hay administradores	✓	✗	✗	NA
Ningún participante tiene más derechos que los demás	✓	✗	✗	NA
Se pueden implementar Smart *Contracts*	✓	✓	✓	NA
Existe recompensa por minado de bloques	≈	✗	✗	NA
Soluciona problema de falta de confianza	✓	✗	≈	NA
Seguridad basada en protocolos de consenso	✓	✗	≈	NA
Seguridad basada en funciones *hash*	✓	≈	≈	NA
Provee servicios en la nube	NA	NA	NA	✓

✓ Sí ✗ No ≈ A veces NA No Aplica

Fig 3. Tipos de Blockchain

Fuente: [19]

2.3.3 Criptografía de Hash

Es una técnica que transforma una cantidad variables de datos en una cadena fija de caracteres alfanuméricos, llamadas hash. Los algoritmos de hash toman una entrada y generan un valor hash único y representativo de datos. Cualquier modificación en los datos originales

producirá un hash completamente diferente. La criptografía de hash se utiliza para garantizar la integridad de los datos y seguridad en la autentificación. [20]

El algoritmo de hash se caracteriza por:

- Unidireccionales
- Determinista
- Fáciles de calcular
- Comprensible
- Funcionamiento tipo avalancha
- Resistencia débil y fuerte a colisiones
- Irreversibles

Uno de los algoritmos de hash que utiliza la tecnología Blockchain más utilizado es el SHA-256. Es parte de la familia de algoritmos SHA-2 y genera un hash de 256 bits de longitud. Es conocido por su robustez y su capacidad para producir hashes únicos para entradas diferentes. Se utiliza en Blockchain para crear los hashes de los bloques y garantizar la seguridad de la red. [21]

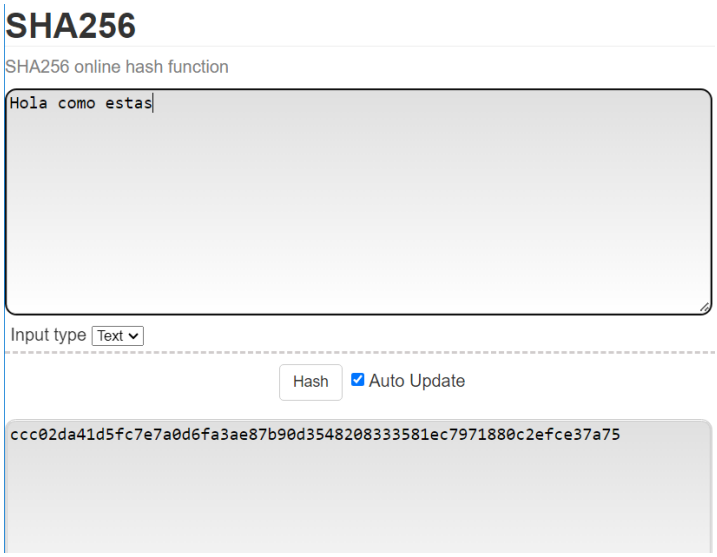


Fig 4. Criptografía de Hash

Fuente: Autor

2.3.4 Libro Mayor Inmutable

Es una parte fundamental de la tecnología Blockchain. Se trata de un registro público, distribuido y descentralizado que almacena de forma cronológica todas las transacciones realizadas en una red Blockchain especifica. La característica principal de este libro es su inmutabilidad, lo que significa que una vez que se registra una transacción en la Blockchain, no se puede alterar ni borrar de forma retroactiva. Por lo mismo cada transacción está vinculada a la anterior mediante un hash, lo que garantiza la integridad de los datos y la seguridad de la red.

Algunas de sus características del libro mayor inmutable son:

- **Descentralización:** El libro mayor se encuentra en múltiples nodos de la red Blockchain y cada nodo mantiene una copia idéntica del libro mayor. Por lo que no necesita de un intermediario central. [22]
- **Inmutabilidad:** La inmutabilidad del libro mayor se logra mediante la criptografía explicada en un punto anterior y la estructura de bloques encadenados.

Cada bloque contiene un conjunto de transacciones, y su contenido se asegura mediante funciones hash de criptografía. Cualquier intento de modificar un bloque requerirá cambiar todos los bloques posteriores. [22]

- **Transparencia:** El libro mayor es transparente y accesible públicamente. Cualquier persona puede ver todas las transacciones registradas en la cadena de bloques, lo que fomenta la confianza. [22]

- **Seguridad:** La seguridad del libro mayor se basa en la criptografía anteriormente explicada y en la participación de la red. Las transacciones se validan mediante consenso entre los nodos de la red. [22]

2.3.5 Red distribuida (P2P)

La red distribuida P2P (Peer-to-Peer) de Blockchain se refiere a la infraestructura de comunicación utilizada por una red Blockchain especifica. En lugar de depender de un servidor centralizado, las redes Blockchain se basan en una arquitectura descentralizada en la que los nodos se conectan directamente entre sí a través de una red P2P. cada nodo en la red es igual y tiene la misma capacidad de verificar y registra transacciones. [22]

2.3.6 Minado

El minado de Blockchain es el proceso mediante el cual los nodos de una red Blockchain compiten para resolver complejos problemas matemáticos y al hacerlos, verifican y registran nuevas transacciones den la cadena de bloques. A cambio de su trabajo, los mineros tienen la oportunidad de recibir recompensas en forma de criptomonedas y comisiones de transacción. El minado es esencial para mantener la seguridad y la integridad de la Blockchain y es un mecanismo

de conceso fundamental en muchas redes descentralizadas. [22]

En el minado los mineros recopilan u grupo de transacciones pendientes y compiten para resolver un problema criptográfico complejo. La primera computadora en encontrar la solución correcta tiene el derecho de agregar un nuevo bloque a la Blockchain. [22]

Por consecuente la dificultad de dicho problema criptográfico se ajusta automáticamente para que aproximadamente cada tiempo se mine un bloque dependiendo de la Blockchain especifica. Este ajuste garantiza que el proceso de minado sea competitivo y que no se creen nuevos bloques demasiado rápido ni demasiado lento.

2.3.7 Protocolo de consenso

El protocolo de consenso es un conjunto de reglas y procedimientos que rigen como los nodos en una red Blockchain alcanzan un acuerdo sobre la validez y el orden de las transacciones en el libro mayor. Los protocolos de consenso son fundamentales para garantizar que la red funcione de manera confiable, segura y descentralizada. Hay varios protocolos de consenso utilizados en Blockchain, como a continuación se muestran los más utilizados:

- **Proof of Work (PoW):** Es un protocolo de consenso utilizado en Blockchains como Bitcoin, Ethereum, entre otras. Su objetivo es asegurar la red y validar las transacciones de manera descentralizada. El proceso se basa en la resolución de problemas matemáticos complejos, que requieren un alto poder de procesamiento computacional. Su idea se basa en que, si un minero quiere agregar un bloque a la cadena de bloques, debe invertir una gran cantidad de recursos, como energía y equipo de minería. Por eso ha habido críticas sobre su eficiencia energéticas debido a la cantidad que se requiere para resolver dichos problemas matemáticos.

[23]

- **Proof of Stake (PoS):** Es un protocolo de consenso que se utiliza en Blockchains como Tezos, Cosmos, Binance, Cardano, entre otras. Para alcanzar acuerdos y validar transacciones por lo que su proceso se basa en la tenencia de criptomonedas y la participación en la red. Su idea se basa en el que los nodos validadores deben poseer y apostar una cantidad específica de la criptomoneda para ser elegibles para validar transacciones y agregar bloques. Los validadores son seleccionados de manera aleatoria en función de su participación y cuanto más grande es la participación de un validador, mayor será su probabilidad de ser seleccionado para validar transacciones. [23]

2.4 Ganache

Ganache es una Blockchain personal diseñada principalmente para el desarrollo de DApps (Aplicaciones Descentralizadas) en los entornos de Ethereum y Filecoin. Esta herramienta es ampliamente utilizada por desarrolladores Blockchain y Smart Contracts para la creación y prueba de DApps en un entorno de red local completamente controlado. Ganache brinda la capacidad de probar y ejecutar comandos, asi como de interactuar con cuentas que contienen unidades de prueba de Ether. Esto facilita la observación de estados de cuenta, direcciones, transacciones y saldos. Además, Ganache ofrece dos versiones: UI y CLI, lo que brinda flexibilidad en su uso. [24]

Fig 5. Logo de Ganache

Fuente: [25]

2.5 Ethereum

Ethereum es una plataforma Blockchain descentralizada con una capacidad especial que permite desarrollar programas denominados Smart Contracts. Estos contienen lógica de aplicación y hacen posible la implementación de lo que se denominan DApps. Dicha tecnología fue desarrollada por el programador ruso Vitalik Buterin y 2 compañeros. [26]

El propósito principal del proyecto es para descentralizar la web mediante el uso de mensajes dinámicos, publicación de contenido estático, transacciones confiables y una interfaz de usuario integrada y funcional, además la creación de una plataforma que permita y facilite a otros programadores el desarrollo y despliegue de DApps. [26]

Ethereum dispone de un lenguaje de programación denominado Solidity. Se trata de un lenguaje Turing completo, que tiene la capacidad para resolver problemas complejos. También dispone de su propia criptomoneda llamada Ether, por lo que se realiza pagos para asegurar que las aplicaciones sean de calidad o recompensar a los mineros por las operaciones en la red.

Fig 6. Logo de Ethereum

Fuente: [27]

2.5.1 El Ether y gas en Ethereum

El Ether es la criptomoneda de la Blockchain de Ethereum, es necesaria para pagar el despliegue de Smart Contracts en la red y la ejecución de transacciones. Y de los más importantes de Ethereum es el gas, medidas que sirve para dimensiona la carga computacional que se ejerce al

ejecutar cualquier transacción en la red. Para ejecutar ciertas transacciones o crear cuentas y Smart Contract en Ethereum es necesario disponer de gas. [26]

Cada transacción tiene asociado un límite de gas y un precio de gas que define las tasa por ejecutarlas en la red, estas comisiones compensan a los mineros por el coste computacional de ejecutar la transacción y añadir un nuevo bloque. Es posible que existan bucles infinitos en el código, de manda que la red podría quedar colapsada, pero esto se soluciona a través del gas que evita este tipo de situaciones, ya que el coste sería infinito para la cuenta asociada que ejecute el código. Por lo que existe una relación directa entra la complejidad de una transacción y su coste en gas. [26]

2.5.2 Smart Contracts

Los Smart Contracts se describen como un conjunto de promesas especificadas digitalmente por las partes implicadas que se cumplen automáticamente por la implementación de ciertas reglas o protocolos. Los Smart Contracts vienen a reemplazar a los contratos legales tradicionales que por la gestión humana son susceptibles de cometer errores o de corromperse por incentivos. Lo cual, un Smart Contract se ejecutar de manera autónoma, descentralizada y sin que se pueda influir en su comportamiento. [28]

Este tipo de contratos son diferentes dependiendo de que tokens se integren en las aplicaciones y por los cuales tenemos a los siguientes:

- **ERC-20**: Tiene la capacidad de crear, transferir y operar tokens creando un valor transferible en la red. Para crear este token se necesita un nombre, un símbolo, los decimales que tendrá y el balance total. [29]
- **ERC-777:** Se trata de una versión mejorada del ERC-20 que añade funcionales

como la capacidad de aceptación o rechazo de una cantidad de tokens. También permite que los operators tengan poder para transferir token en nombre del propietario de la cuenta. [29]

- **ERC-721:** Es el protocolo para crear tokens no fungibles (NFT). Este concepto se refiere a un activo que es único y no sustituible por otro. A menudo, se denominan tokens coleccionables, que se puede definir su valor en función de sus propiedades. [29]

2.5.3 DApp

Una DApp (Aplicación Descentralizada) es una aplicación informática que se ejecuta en una red descentraliza como la Blockchain de Ethereum, lo cual es posible interactuar con los Smart Contract para ejecutar lógica de negocio sin ningún tipo de intermediarios de manera descentralizada. La ventaja de estas aplicaciones es su nivel de seguridad, ya que todos los nodos de la red deben validar la información que se genera en el proceso. [30]

Las características más principales de una DApp son:

- La aplicación debe ser 100% código abierto.
- Su funcionamiento debe ser autónomo y sin intermediaros.
- La aplicación debe asegurar los intereses de todos los usuarios.
- Toda la información generada por la aplicación debe almacenarse en una plataforma descentralizada y abierta.

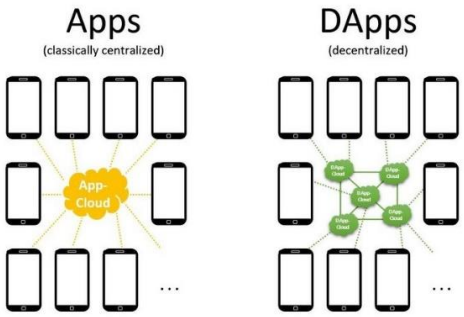


Fig 7. Estructura de una DApp

Fuente: [31]

2.5.4 La máquina virtual de Ethereum (EVM)

La máquina virtual de Ethereum está integrada en el software que se ejecuta en Ethereum, manteniendo una memoria transitoria ya que dicha maquina está formado por nodos de la red, los cuales ejecutan el consenso de toda la Blockchain, lo cual obtiene niveles extremos de tolerancia a fallas, en DApps escritas en Solidity. [32]

Se ejecutan al mismo tiempo todos los nodos llegando a será la información descentralizada y convirtiéndose en inmutable mediante el consenso, pero dicha ejecución no es gratuita ya que conlleva fastos a los mineros, por el gas necesario para ejecutar el proceso. [32]

A continuación, se muestra que requiere la EVM para poder proporcionar el entorno de ejecución:

- Estado de las cuentas
- Estado del mundo
- Estado de almacenamiento
- Información del tiempo de ejecución para tramitar las transacciones,
- Información correspondiente al bloque.

2.6 Wallet

La wallet (Billetera Digital) es una aplicaciones o dispositivo utilizado para almacenar, gestionar y realizar transacciones con criptomonedas y tokens digitales. Funciona de manera similar a una billetera física que almacena dinero en efectivo, pero en lugar de billetes o monedas, almacenas claves criptográficas que representan la propiedad y el acceso a activos digitales en una Blockchain. [33]

Las wallets almacena las claves privadas necesarias para acceder a las criptomonedas y tokens, llevan un registro detallado del historial de transacciones y ofrecen medidas de seguridad como autentificación de dos factores y tu propia clave privada compuesta de 12 palabras que sirven para el acceso a dicha wallet. [33]

2.6.1 Metamask

Metamask se destaca como una de las wallets de criptomonedas más renombradas y extensamente utilizadas en la actualidad. Su propósito es servir como la puerta de entrada al emocionante mundo de la Web3, las DeFi y los NFT. Esta plataforma ofrece una billetera digital integral que permite a los usuarios administrar, enviar y recibir criptomonedas, asi como interactuar sin esfuerzo con DApps en la red Ethereum y otras redes compatibles.

Además de su funcionalidad como wallet, opera como un valioso plugin de navegador específicamente diseñado para la red Ethereum. Permite a los usuarios almacenar Ether y, llevar a cabo transacciones hacia cualquier dirección de la red Ethereum gracias a esta herramienta. Además, ofreces múltiples conexiones que simplifican el acceso rápido tanto a la Blockchain de Ethereum como a diversas redes de prueba.

Esta funcionalidad no solo facilita las transacciones, sino que también agrega una capa

adicional de seguridad. Al permitir transacciones seguras y autenticadas, Metamask desemplea un papel crucial en la mitigación de riesgos, como posibles ataques y suplantación de identidad, en el entorno de desarrollo Blockchain. [34]

Fig 8. Logo de Metamask

Fuente: [35]

2.7 Editor de Código

Un editor de código es una aplicación de software diseñada para facilitar la creación y edición de archivos de código de programas. Se utiliza para escribir y modificar el código de las aplicaciones y proyectos de software.

Existen muchos editores de códigos disponibles, cada uno con sus propias características y ventajas. A continuación, una comparativa de algunos editores de código:

Tabla II. Comparación de Editores de Código

EDITOR DE CÓDIGO	DESARROLLADOR	LENGUA DE PROGRAMACIÓN	PERSONALIZACIÓN Y EXTENSIONES	INTEGRACIÓN CON HERRAMIENTAS	PLATAFORMAS SOPORTADAS
VISUAL STUDIO CODE	Microsoft	Amplia Variedad	Altamente personalizable con extensiones	Amplia integración con Git y Azure	Windows, macOs, Linux
ATOM	GitHub	Amplia Variedad	Altamente personalizable con paquetes	Integración con GitHub	Windows, macOs, Linux
INTELLIJ IDEA	JetBrains	Principalmente Java	Muy personalizable y extensible	Amplia integración con Java	Windows, macOs, Linux

Fuente: Autor

En base a la comparación vista en este proyecto se usará Visual Studio Code para el desarrollo del prototipo que se va a realizar.

2.7.1 Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente liviano pero potente y gratuito desarrollado por Microsoft, y se encuentra disponible de forma gratuita. Puede trabajar en cualquier plataforma existente (MacOs, Windows, Linux). Es altamente personalizable gracias a su sistema de extensiones, donde los usuarios pueden agregar y configurar dichas extensiones para adaptar el IDE a sus necesidades. Viene con soporte integrado para JavaScript, TypeScript y Node.js. Con sus extensiones VS Code facilitan la gestión de paquetes y entornos virtuales para lenguajes de programación como Java, Python, Solidity, PHP, etc. [36]

Fig 9. Logotipo de Visual Studio Code

Fuente: [37]

2.8 Frontend

El frontend se refiere a la parte de una aplicación o sitio web que los usuarios ven y en las que se puede interactuar. Es la interfaz de usuario que presenta información y funcionalidad de manera visual, por lo que se encarga de mostrar el contenido de una aplicación de manera organizada y amigable para el usuario.

Existen muchos, y cada uno tiene sus características. Por lo que, a continuación, una comparativa de algunos:

Tabla III. Comparativa de Frontend

FRONTEND	LENGUAJE PRINCIPAL	ARQUITECTURA	FLEXIBILIDAD	CURVA DE APRENDIZAJE	RENDIMIENTO
REACT	JavaScript	Virtual Dom	Alta	Moderada	Bueno
ANGULAR	TypeScript	MVC	Baja	Pronunciada	Bueno
VUE.JS	JavaScript	Virtual Dom	Moderada	Suave	Bueno

Fuente: Autor

En base a la comparación en este proyecto se usará React ya que permite desarrollar interfaces de usuario y también la podemos combinar con Bootstrap para que la interacción con el usuario sea de buen agrado.

2.8.1 React

React es una popular biblioteca de JavaScript utilizada en el desarrollo web y aplicaciones móvil para construir interfaces de usuario interactivas y de alta calidad. Desarrollada y mantenida por Meta, es una biblioteca de código abierto, basada en componentes, los cuales poseen lógica y controladores, permitiendo una creación y un desarrollo rápido. React se ha convertido en una herramienta esencial en el mundo del desarrollo web y es ampliamente adoptada por empresas y desarrolladores del mundo. [38]

Es crucial aclarar que React no es un framework de JavaScript, sino que asume el rol crucial de renderizar los componentes de la capa de vista en una aplicación. Esta característica lo distingue de frameworks como Angular y Vue. [38]

Fig 10. Logotipo de React

Fuente: [39]

2.8.2 Bootstrap

Bootstrap es framework CSS de código abierto que se utiliza para simplificar y acelerar el proceso de desarrollo web. Fue creado por Twitter y se ha convertido en uno de los marcos de diseño más populares y ampliamente utilizados en la industria del desarrollo web. Bootstrap se centra en la creación de sitios web receptivos, atractivos y funcionales, y ofrece una serie de componente predefinidos y herramientas que facilitan a la construcción de interfaces de usuario modernas y profesionales. [40]

Fig 11. Logotipo de Bootstrap

Fuente: [41]

2.9 Backend

Para el desarrollo del backend de la aplicación propuesta se decidió utilizar como lenguaje principal Solidity ya que permite la creación de Smart Contracts en su lenguaje de muy alto nivel, también utilizado JavaScript y Ethers.js para validar y facilitar la ejecución de los Smart Contracts con el frontend. Para la instalación de las librearías se utilizará node.js y para la compilación y despliegue de los Smart Contracts se usará HardHat.

2.9.1 Solidity

Solidity es un lenguaje de programación de alto nivel utilizado para escribir Smart Contracts en la plataforma Ethereum y otras Blockchains compatibles con Ethereum. Los Smart Contract son programas autónomos que se ejecutan en la Blockchain y pueden interactuar con activos digitales, realizar transacciones uy automatizar procesos de negocio. Es esencial para el

desarrollo de DApps en Ethereum y en otras Blockchains similares. Posee una sintaxis muy similar a JavaScript, dicho lenguaje utiliza como extensión .sol. [42]

La EVM admite la ejecución de código de Solidity por lo cual facilita a los desarrolladores en la creación y despliegue de Smart Contracts a la vez que permite tener acceso al estado y propiedad de Smart Contracts desplegados. [42]

Fig 12. Logotipo de Solidity

Fuente: [43]

2.9.2 JavaScript

JavaScript es un lenguaje de programación de alto nivel, interpretado y orientado a objetos ampliamente utilizado en el desarrollo web y en la creación de aplicaciones en el lado del cliente. [44]

Creado por Netscap y desarrollado por Brendan Eich, es un lenguaje usado para realizar validaciones o construir webs dinámicas, convirtiéndose en un lenguaje para el desarrollo de aplicaciones web y móviles mediante entornos de ejecución como Node.js. Es una de las tecnologías más importantes en creación de sitios web interactivos y dinámicos. [44]

2.9.3 Node.js

Node Js es un entorno de tiempo de ejecución de JavaScript de código abierto que permite a los desarrolladores ejecutar código en lado del servidor. Fue creado por Ryan Dahl y lanzado por primera vez en 2009. Y todo esto es posible mediante el motor V8 que fue incorporado al núcleo

de Node.js permitiendo la creación de aplicaciones escalables y de alto rendimiento. [45]

Se ha convertido en una tecnología ampliamente adoptada en el desarrollo web y en el desarrollo de aplicaciones de red, y es conocido por su capacidad para construir aplicaciones escalables y de alto rendimiento. [45]

Fig 13. Logotipo de Node.JS

Fuente: [46]

2.9.4 Ethers.js

Ethers.js es una biblioteca de JavaScript utilizada para interactuar con la red Ethereum y desarrollar DApps y Smart Contracts en Ethereum. Esta biblioteca facilita la comunicación con la Blockchain Ethereum y proporciona una serie de funciones y utilidades que simplifican el desarrollo en la plataforma. Por lo que proporciona una interfaz de programación de Aplicaciones en JavaScript que permite interactuar con la red Ethereum. Incluyendo la creación y firma de transacciones, la consulta de información de bloques y transacciones, y la interacción con Smart Contracts. Permite la conexión a diferentes redes Ethereum y facilita la transferencia de tokens ERC-20y la interacción de tokens ERC-721 en Ethereum. [47]

2.9.5 Hardhat

Hardhat es un marco de desarrollo de Ethereum diseñado para simplificar y agilizar el proceso de desarrollo de DApps y Smart Contracts en la plataforma Ethereum. Se ha convertido en una herramienta popular y ampliamente es utilizada por desarrolladores del mundo. Por lo que

Hardhat proporciona un entorno completo para escribir, compilar y probar Smart Contracts en Ethereum. También permite a los desarrolladores escribir contratos en Solidity y Typescript, y ofrece herramientas para compilar u desplegar contratos en diferentes redes Ethereum. También ofrece un marco de prueba que permite a los desarrolladores escribir pruebas automatizadas, lo cual ayuda para garantizar la seguridad y la funcionalidad de los contratos. [48]

Fig 14. Logotipo de Hardhat

Fuente: [48]

2.9.6 IPFS

IPFS o el sistema de archivos interplanetario, es una red distribuida para almacenar y compartir archivos. Fue creado por Juan Benet y es un proyecto de código abierto respaldado por una comunidad activa de desarrolladores. Utiliza un enfoque descentralizado, donde los archivos se almacenan en la red de las computadoras participante, en lugar de un servidor central. Esto permite compartir de manera eficiente y resistente de archivos, ya que los datos se distribuyen en muchos nodos en la red. También proporciona una forma segura de compartir contenido. [49]

Fig 15. Logotipo de IPFS

Fuente: [50]

CAPÍTULO 3

3. MARCO METODOLÓGICO

3.1 Desarrollo

Este capítulo describe el desarrollo de la DApp, para la interacción del usuario en la compra y venta de vehículos. Buscando resolver las problemáticas existentes en el mercado automotriz, este objetivo se enfoca en la resolución de los desafíos actuales presentes en este mercado.

Se desarrollarán las fases y procesos que va a tener la aplicación web cumpliendo con la metodología propuesta en el capítulo anterior basados en la metodología Scrum.

3.2 Análisis de requerimientos

Los requerimientos iniciales se establecieron mediante una reunión con el Ing. Andrés Sebastián Quevedo Miranda con el objetivo de analizar y recolectar información sobre los requerimientos para determinar que funcionalidades se deben aplicar en la DApp, al recolectar la información se enlista las funcionalidades que tendría el software:

Tabla IV. Requerimiento de la DApp

REQUERIMIENTO FUNCIONALES	REQUERIMIENTOS NO FUNCIONALES
Capacidad de crear un vehículo	Diseñar una interfaz de usuario amigable
Capacidad de revender de un vehículo	
Capacidad de ver vehículos comprados	Integrar un sistema de pagos utilizando Blockchain
Capacidad de ver vehículos creados	
Capacidad de ver trazabilidad de un vehículo	
Capacidad de gestionar usuarios	Diseñar un Marketplace que nos permita movimiento de productos seguros

Fuente: Autor

3.3 Implementación de la Metodología Scrum.

Antes de iniciar el desarrollo de la DApp, es imperativo contar con las historias de usuario. Esta tarea es fundamental debido a la aplicación de la metodología Scrum, que desempeña un papel crucial en el desarrollo del prototipo. Scrum nos brinda la capacidad de estructurar el desarrollo de software en etapas llamadas "sprints". Para lograr esto de manera efectiva, es esencial definir con claridad el Product Backlog, detallando el análisis de requerimientos que se ha llevado a cabo con anterioridad.

Cada historia de usuario tiene:

- Id de la historia
- Nombre de historia
- Estimación de historia
- Importancia de historia
- Comentario de historia

Tabla V. Product Backlog

ID	NOMBRE DE HISTORIA	ESTIMACIÓN	IMPORTANCIA	COMENTARIO
1	Crear un vehículo	4	5	Esta historia de usuario es fundamental para la funcionalidad básica del sistema
2	Revender un vehículo	4	5	La capacidad de revender vehículos es una función clave que permite a los usuarios vender vehículos que no desean poseer
3	Gestionar usuarios	4	5	Esta historia de usuario nos permitirá gestionar usuarios en un entorno local de Blockchain, utilizando Ganache. Esto permitirá simular interacciones de usuario y probar la funcionalidad.

4	Ver vehículos comprados	2	4	Esta historia de usuario permite acceder a la información de los vehículos que ha comprado un usuario
5	Ver vehículos creados	2	4	Esta historia de usuario permite poder ver los vehículos que ha creado un usuario
6	Ver trazabilidad de un vehículo	3	5	La trazabilidad de un vehículo es crucial para rastrear su historial de transacciones y cambios de propiedad
7	Diseñar una interfaz de usuario amigable	3	4	La interfaz de usuario amigables y fácil de usar es esencial para la experiencia del usuario
8	Diseñar un Marketplace que permita movimiento de productos seguros	3	4	Esta historia de usuario se trata sobre el diseño Marketplace de manera que garantice la seguridad en el movimiento de productos y transacciones.
9	Integrar un sistema de pagos utilizando Blockchain	3	4	La integración de un sistema de pagos con Blockchain es fundamental para permitir transacciones de criptomonedas

Fuente: Autor

3.4 Definición de los Sprints

Después de establecer el Product Backlog en donde se encuentra cada historia de usuario, donde agrupados pasan a formar un Sprint para determinar el tiempo que se empleara para cumplirlo.

Se determinará las horas en base de la estimación y se determinará el número de horas por el esfuerzo y el tiempo que requiera cada Sprint. Cada Sprint tendrá un rango de estimación de entre 1 a 4 semanas de esfuerzo y un rango de importancia comprendido entre 1 a 5 intervalos.

A continuación, en la tabla V, se pueden apreciar las historias de usuario correspondientes al Sprint 1.

Tabla VI. Sprint 1

SPRINT 1				
ID	**HISTORIA DE USUARIO**	**ESTIMACIÓN**	**IMPORTANCIA**	**COMENTARIO**
1	Crear un vehículo	4	5	Esta historia de usuario es fundamental para la funcionalidad básica del sistema
2	Revender un vehículo	4	5	La capacidad de revender vehículos es una función clave que permite a los usuarios vender vehículos que no desean poseer
3	Gestionar usuarios	4	5	Esta historia de usuario nos permitirá gestionar usuarios en un entorno local de Blockchain, utilizando Ganache. Esto permitirá simular interacciones de usuario y probar la funcionalidad.

Fuente: Autor

A continuación, en la tabla VI, se pueden apreciar las historias de usuario correspondientes al Sprint 2.

Tabla VII. Sprint 2

SPRINT 2				
ID	**HISTORIA DE USUARIO**	**ESTIMACIÓN**	**IMPORTANCIA**	**COMENTARIO**
4	Ver vehículos comprados	2	4	Esta historia de usuario permite acceder a la información de los vehículos que ha comprado un usuario
5	Ver vehículos creados	2	4	Esta historia de usuario permite poder ver los vehículos que ha creado un usuario

Fuente: Autor

A continuación, en la tabla VII, se pueden apreciar las historias de usuario correspondientes al Sprint 3.

Tabla VIII. Sprint 3

SPRINT 3				
ID	**HISTORIA DE USUARIO**	**ESTIMACIÓN**	**IMPORTANCIA**	**COMENTARIO**
6	Ver trazabilidad de un vehículo	3	5	La trazabilidad de un vehículo es crucial para rastrear su historial de transacciones y cambios de propiedad
7	Diseñar una interfaz	3	4	La interfaz de usuario amigables y fácil de

	de usuario amigable			usar es esencial para la experiencia del usuario

Fuente: Autor

A continuación, en la tabla VIII, se pueden apreciar las historias de usuario correspondientes al Sprint 4.

Tabla IX. Sprint 4

SPRINT 4				
ID	**HISTORIA DE USUARIO**	**ESTIMACIÓN**	**IMPORTANCIA**	**COMENTARIO**
8	Diseñar un Marketplace que permita movimiento de productos seguros	3	4	Esta historia de usuario se trata sobre el diseño Marketplace de manera que garantice la seguridad en el movimiento de productos y transacciones.
9	Integrar un sistema de pagos utilizando Blockchain	3	4	La integración de un sistema de pagos con Blockchain es fundamental para permitir transacciones de criptomonedas

Fuente: Autor

Una vez que hemos organizado las historias de usuario de manera que nos permitan planificar sprints con estimaciones de tiempo, hemos generado 4 sprints. Estos sprints se estructuran en función de la prioridad que tienen para el desarrollo del prototipo.

3.5 Planificación de los Sprints

Para el Desarrollo de cada sprint. Se establecen evaluaciones para validar el progreso, lo que permite obtener retroalimentación cuando sea necesario. En caso de ser necesario, se toman medidas de mejora que se aplicaran en las futuras entregas del proyecto.

Cada desarrollo de sprint se documenta a través del Taskboard, que proporciona una representación visual clara de las tareas en diferentes estados: finalizadas, en curso o pendientes por cada historia de usuario. Además, se utiliza el Burndown para supervisar la velocidad de desarrollo del proyecto, lo que facilita la identificación de tareas o historias de usuario que puedan estar requiriendo un tiempo excesivo de desarrollo.

Al finalizar cada sprint, se presenta la documentación de los entregables obtenidos en ese periodo. A continuación, se detalla la planificación para cada sprint del proyecto.

A continuación, en la tabla IX, se pueden ver las historias de usuario junto con las estimaciones de tiempo para el desarrollo del Sprint 1.

Tabla X. Planificación Sprint 1

SPRINT 1	
AUTOR: JOHN RIVERA	
FECHA DE INICIO	05/Junio/2023
FECHA DE FIN	03/Julio/2023
DURACIÓN	4 semanas
TAREAS POR DESARROLLAR	Crear un vehículo Revender un vehículo Gestionar usuarios

Fuente: Autor

A continuación, en la tabla X, se pueden ver las historias de usuario junto con las estimaciones de tiempo para el desarrollo del Sprint 2.

Tabla XI. Planificación Sprint 2

SPRINT 2	
AUTOR: JOHN RIVERA	
FECHA DE INICIO	03/Julio/2023
FECHA DE FIN	17/Julio/2023
DURACIÓN	2 semanas
TAREAS POR DESARROLLAR	Ver vehículos comprados Ver vehículos creados

Fuente: Autor

A continuación, en la tabla XI, se pueden ver las historias de usuario junto con las estimaciones de tiempo para el desarrollo del Sprint 3.

Tabla XII. Planificación Sprint 3

SPRINT 3
AUTOR: JOHN RIVERA

FECHA DE INICIO	17/Julio/2023
FECHA DE FIN	07/Agosto/2023
DURACIÓN	3 semanas
TAREAS POR DESARROLLAR	Ver trazabilidad de un vehículo Diseñar una interfaz de usuario amigable

Fuente: Autor

A continuación, en la tabla XII, se pueden ver las historias de usuario junto con las estimaciones de tiempo para el desarrollo del Sprint 4.

Tabla XIII. Planificación Sprint 4

SPRINT 4	
AUTOR: JOHN RIVERA	
FECHA DE INICIO	07/Agosto/2023
FECHA DE FIN	28/Agosto/2023
DURACIÓN	3 semanas
TAREAS POR DESARROLLAR	Diseñar un Marketplace que permita movimiento de productos seguros Integrar un sistema de pagos utilizando Blockchain

Fuente: Autor

3.5.1 TaskBoard inicial y BurnDown Chat inicial

La tabla XIII muestra el Taskboard inicial de desarrollo del proyecto, en el que se incluyen todas las historias de usuario junto con su estado inicial en cada uno de los sprints.

Tabla XIV. Taskboard Inicial

N. SPRINT	**INICIO: 05/JUNIO/2023 FIN: 28/AGOSTO/2023 HISTORIA DE USUARIO**	**AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE**	**EN CURSO**	**FINALIZADO**
SPRINT 1	Crear un vehículo	✓		
	Revender un vehículo	✓		
	Gestionar usuarios	✓		
SPRINT 2	Ver vehículos comprados	✓		
	Ver vehículos creados	✓		
SPRINT 3	Ver trazabilidad de un vehículo	✓		
	Diseñar una interfaz de usuario amigable	✓		
SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros	✓		

	Integrar un sistema de pagos utilizando Blockchain	✓		

Fuente: Autor

En la Figura 16, se exhibe el grafico de Burndown Inicial que representa el progreso de todas las historias de usuario.

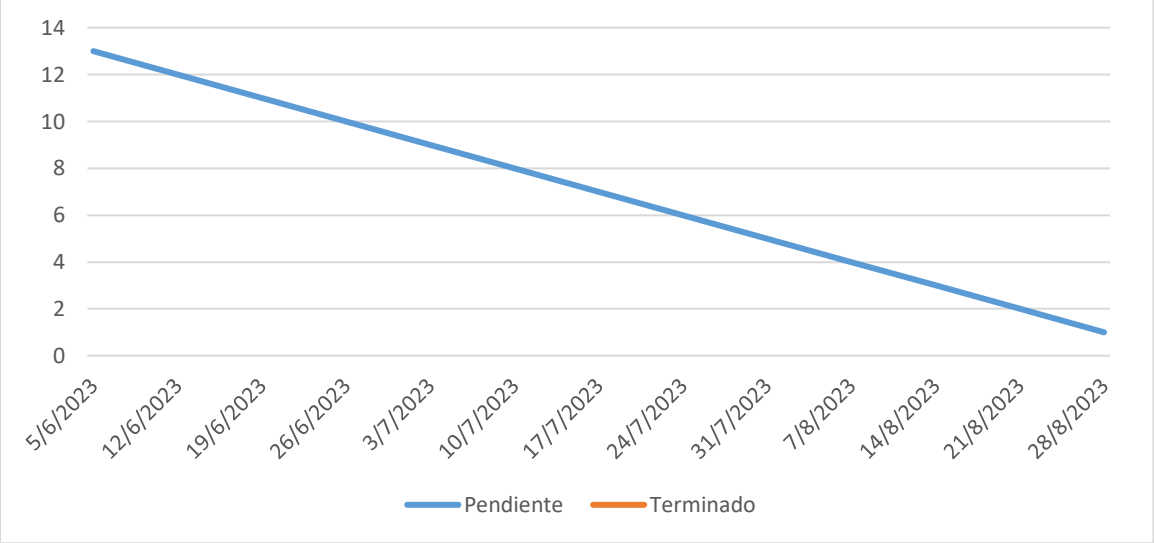


Fig 16. Burndown Inicial

Fuente: Autor

3.6 Desarrollo de la DApp

3.6.1 Sprint 1

Durante el Sprint 1, se llevarán a cabo las siguientes actividades:

- Crear un vehículo
- Revender un vehículo
- Gestionar usuarios

La tabla XIV presenta el Taskboard correspondiente al Sprint1, donde se observa que la historia de usuario “Crear un vehículo” está actualmente en curso.

Tabla XV. Taskboard del Sprint 1

N. SPRINT	INICIO: 05/JUNIO/2023 FIN: 03/JULIO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	EN CURSO	FINALIZADO
SPRINT 1	Crear un vehículo		✓	
	Revender un vehículo	✓		
	Gestionar usuarios	✓		
SPRINT 2	Ver vehículos comprados	✓		
	Ver vehículos creados	✓		
SPRINT 3	Ver trazabilidad de un vehículo	✓		
	Diseñar una interfaz de usuario amigable	✓		
SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros	✓		
	Integrar un sistema de pagos utilizando Blockchain	✓		

Fuente: Autor

La Figura 17 muestra el progreso alcanzado en el Sprint 1 a través del grafico Burndown.

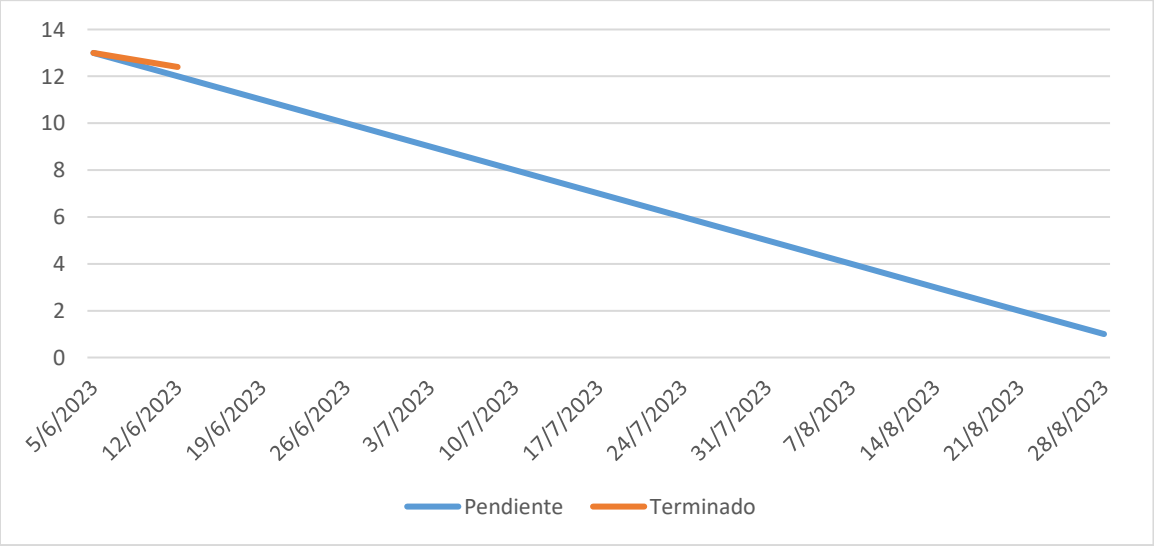


Fig 17. Burndown del Sprint 1

Fuente: Autor

3.6.1.1 Crear un vehículo

En la Figura 18 muestra la barra de navegación, al pulsar el módulo Crear lo lleva a el formulario para la creación de un vehículo.

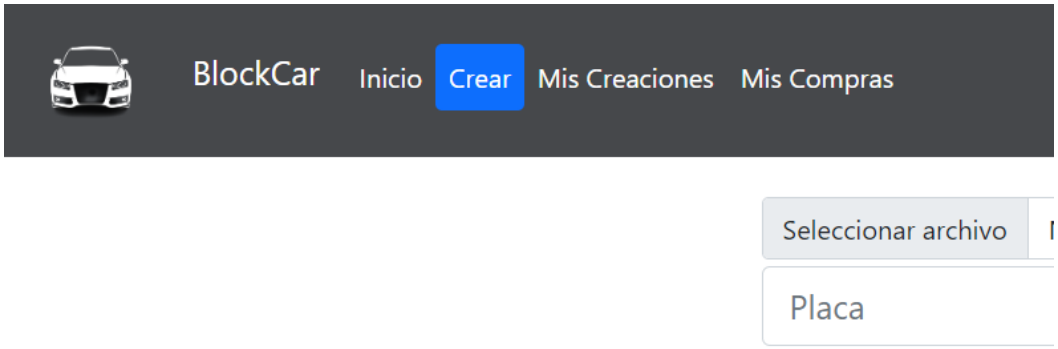


Fig 18. Interfaz del Marketplace

Fuente: Autor

En la Figura 19 se puede observar los datos necesarios para poder crear un vehículo.

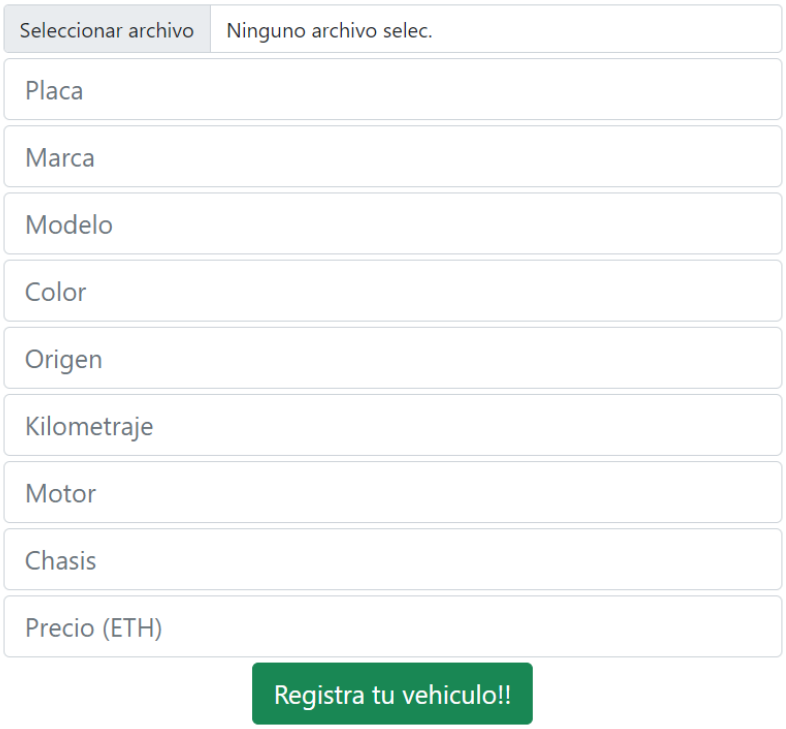


Fig 19. Formulario para crear vehículos

Fuente: Autor

En la Figura 20 muestra todos los vehículos que han sido ya creados.

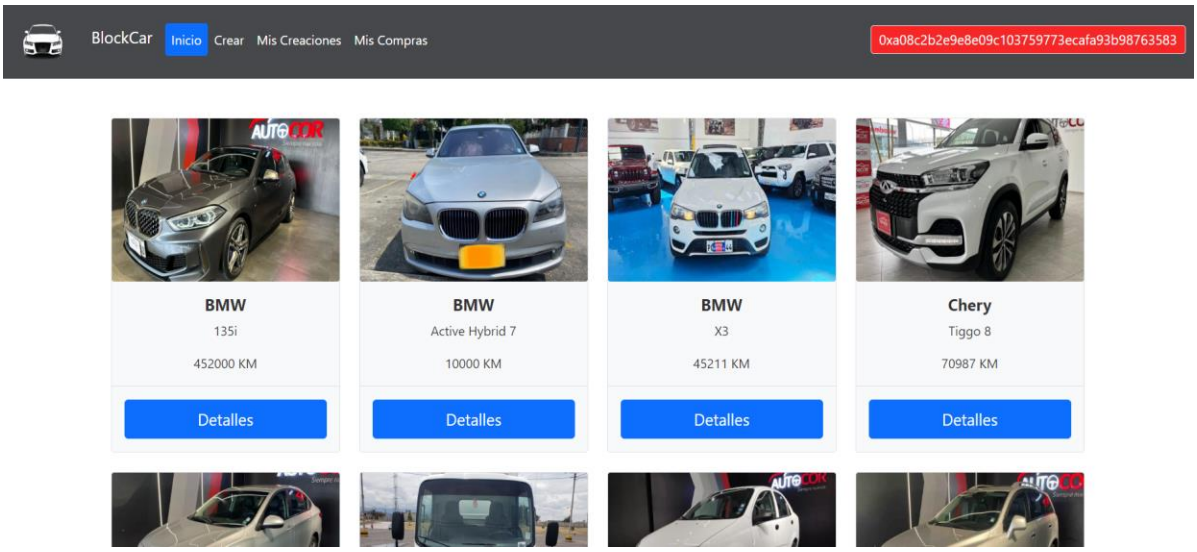


Fig 20. Vehículos creados

Fuente: Autor

La tabla XV exhibe el Taskboard correspondiente al Sprint 1, donde se indica la historia de usuario "Crear un vehículo" ha sido finalizada, mientras que la historia "Revender un vehículo" está actualmente en curso.

Tabla XVI. Avance de Taskboard del Sprint 1

N. SPRINT	INICIO: 05/JUNIO/2023 FIN: 03/JULIO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	EN CURSO	FINALIZADO
SPRINT 1	Crear un vehículo			✓
	Revender un vehículo		✓	
	Gestionar usuarios	✓		
SPRINT 2	Ver vehículos comprados	✓		
	Ver vehículos creados	✓		
SPRINT 3	Ver trazabilidad de un vehículo	✓		
	Diseñar una interfaz de usuario amigable	✓		
SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros	✓		
	Integrar un sistema de pagos utilizando Blockchain	✓		

Fuente: Autor

La Figura 21 presenta el progreso del Sprint 1, donde se puede observar que, dado que las actividades están en curso, no han tenido un efecto negativo en el Burndown de desarrollo. Se

mantienen los tiempos previstos para completar el proyecto.

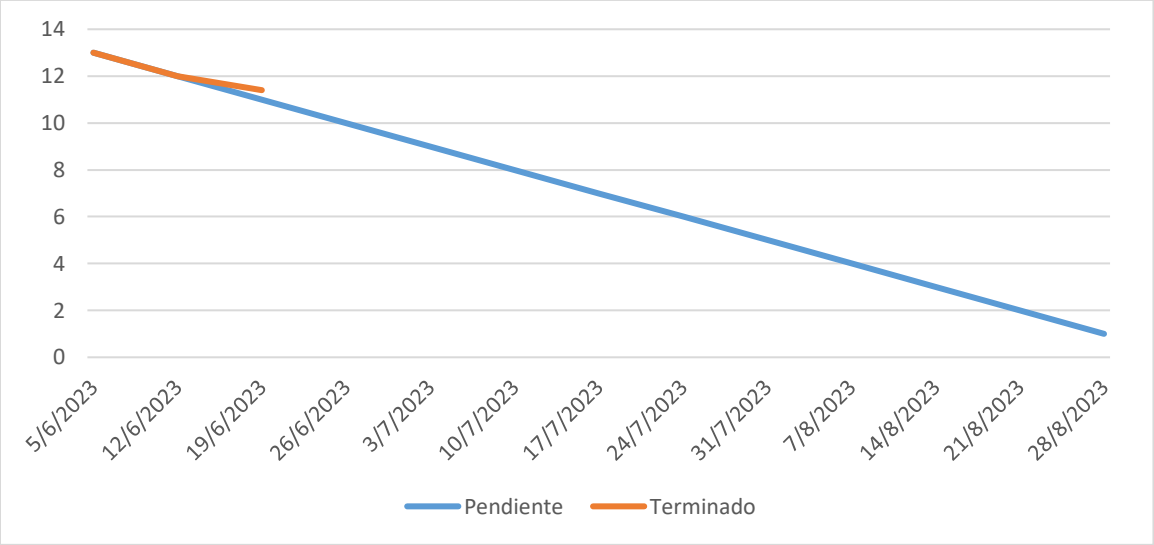


Fig 21. Avance de Burndown del Sprint 1

Fuente: Autor

3.6.1.2 Revender un vehículo

En la Figura 22 muestra la barra de navegación, al pulsar en Mis Compras, se puede observar las compras que ha hecho un usuario, permitiéndolo poder revender.

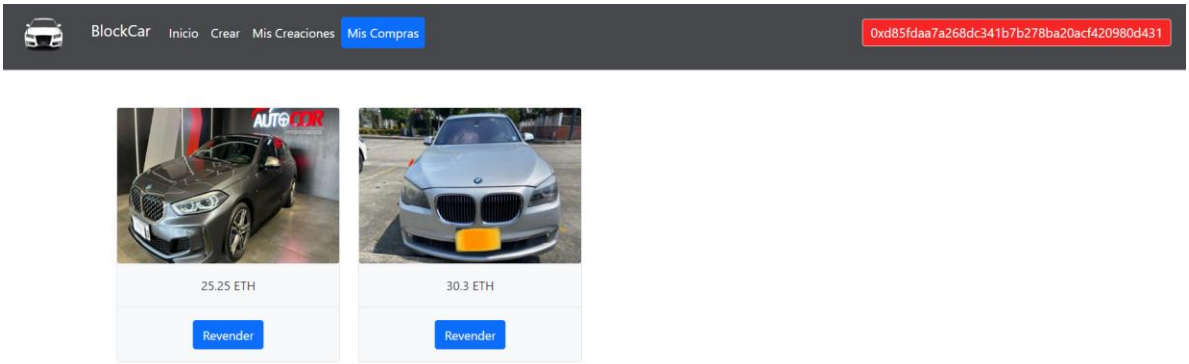


Fig 22. Interfaz de Mis Compras

Fuente: Autor

En la Figura 23 se puede observar un modal, en los que se encuentran los datos necesarios para revender un vehículo.

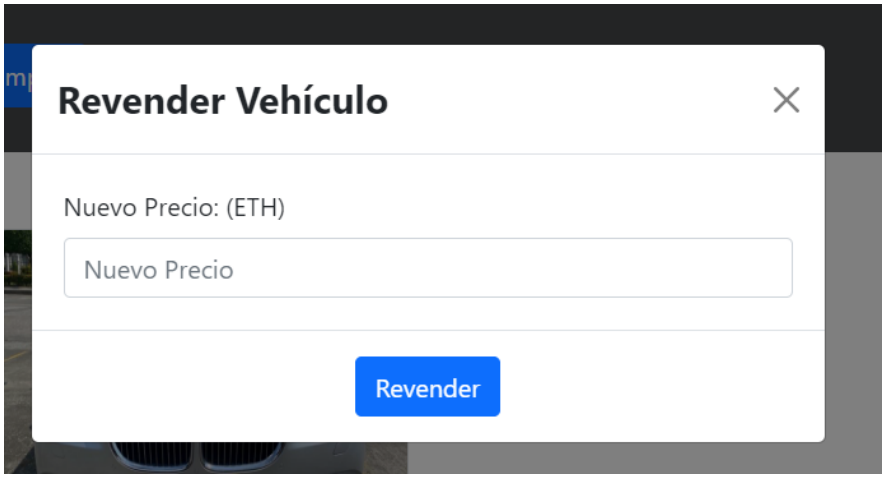


Fig 23. Modal de Revender un Vehículo

Fuente: Autor

La tabla XVI exhibe el Taskboard correspondiente al Sprint 1, donde se indica la historia de usuario "Revender un vehículo" ha sido finalizada, mientras que la historia "Gestionar usuario" está actualmente en curso.

Tabla XVII. Avance 2 de Taskboard del Sprint 1

N. SPRINT	INICIO: 05/JUNIO/2023 FIN: 03/JULIO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	EN CURSO	FINALIZADO
SPRINT 1	Crear un vehículo			✓
	Revender un vehículo			✓
	Gestionar usuarios		✓	
SPRINT 2	Ver vehículos comprados	✓		
	Ver vehículos creados	✓		
SPRINT 3	Ver trazabilidad de un vehículo	✓		
	Diseñar una interfaz de usuario amigable	✓		
SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros	✓		
	Integrar un sistema de pagos utilizando Blockchain	✓		

Fuente: Autor

La Figura 24 presenta el progreso del Sprint 1, donde se puede observar que, dado que las actividades están en curso, no han tenido un efecto negativo en el Burndown de desarrollo. Se mantienen los tiempos previstos para completar el proyecto.

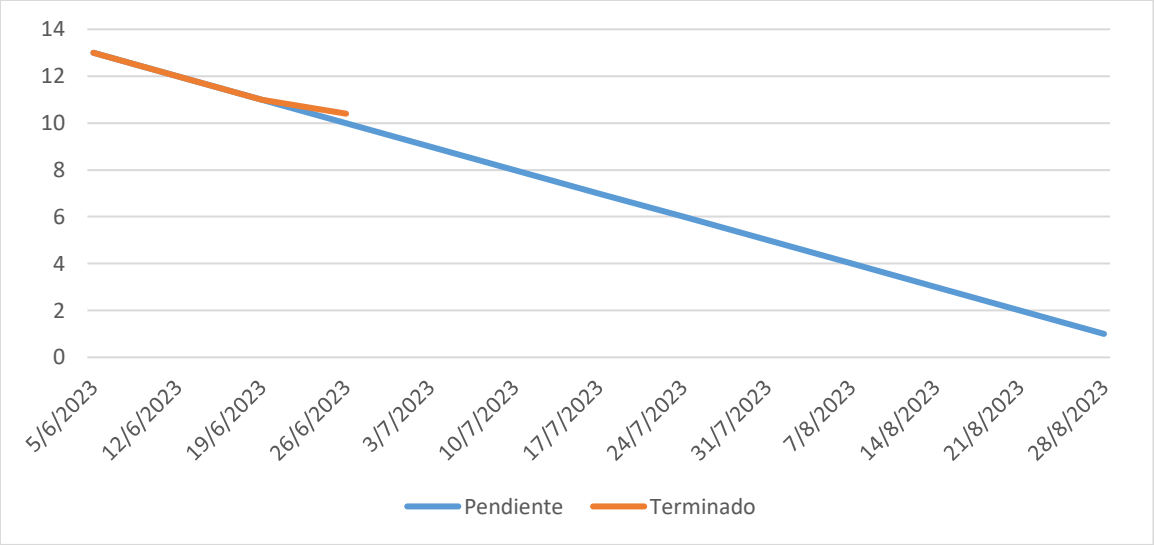


Fig 24. Avance 2 de Burndown del Sprint 1

Fuente: Autor

3.6.1.3 Gestionar usuario

En la Figura 25 muestra el despliegue de la DApp mediante Hardhat, lo cual se lo despliega en la red de Ganache, que es un entorno de pruebas de Blockchain, por lo cual se lo usara para la gestión de cuentas que el mismo entorno de facilita.

```
C:\Users\ASUS TUF\Desktop\block-car>npx hardhat run src/backend/scripts/deploy.js --network ganache
Desplegar contratos con la cuenta: 0x9374375B679C63929019Ee936a0Aa327A3a26F40
Balance de cuenta: 564831595125000000000

C:\Users\ASUS TUF\Desktop\block-car>
```

Fig 25. Despliegue de la DApp en Ganache

Fuente: Autor

En las Figura 26 y 27 se aprecia la aplicación de Ganache sobre el entorno y las direcciones que tienen cada una con un balance de Ethers distinto para el uso de la DApp.

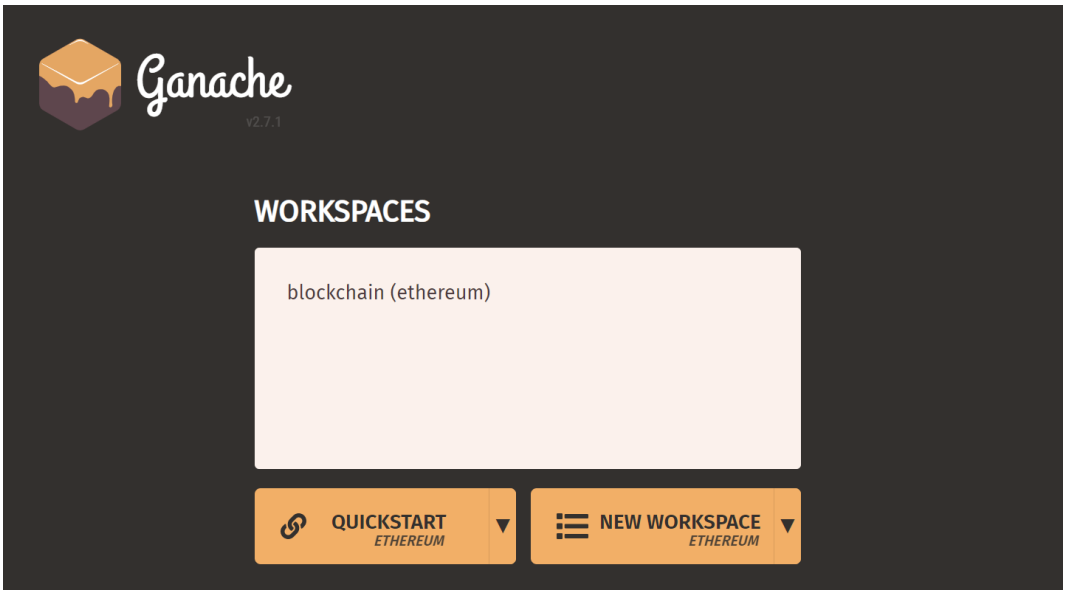


Fig 26. Workspaces de Ganache

Fuente: Autor

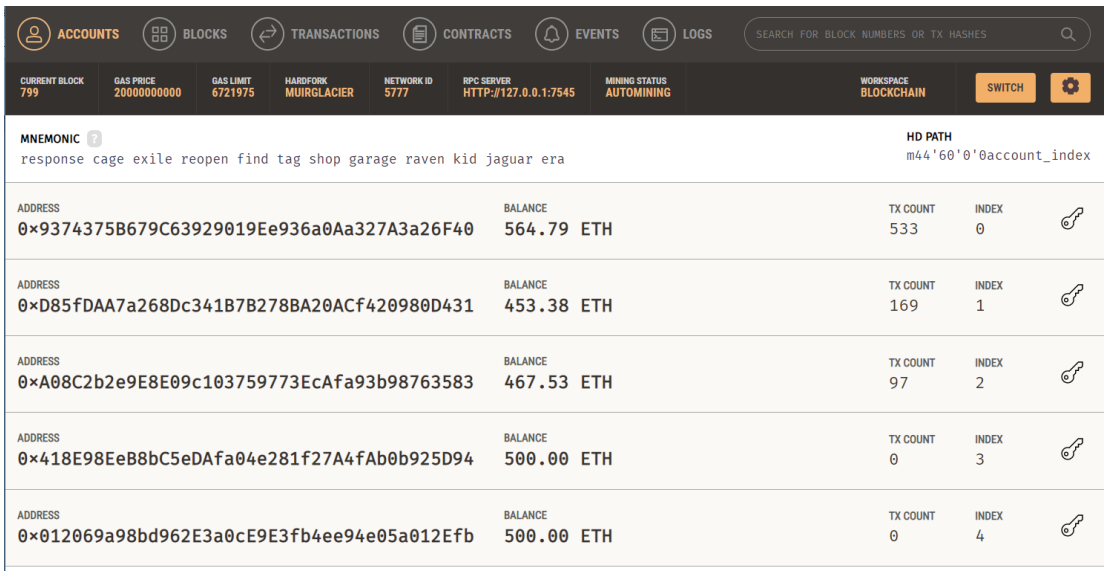


Fig 27. Interfaz del Workspace Blockchain

Fuente: Autor

En la Figuras 28 y 29 se aprecia que, para la interacción de las direcciones con la DApp, se utiliza la wallet de Metamask en donde se encuentra cargadas las cuentas que se tiene en Ganache.

Fig 28. Selección de cuenta en Metamask

Fuente: Autor

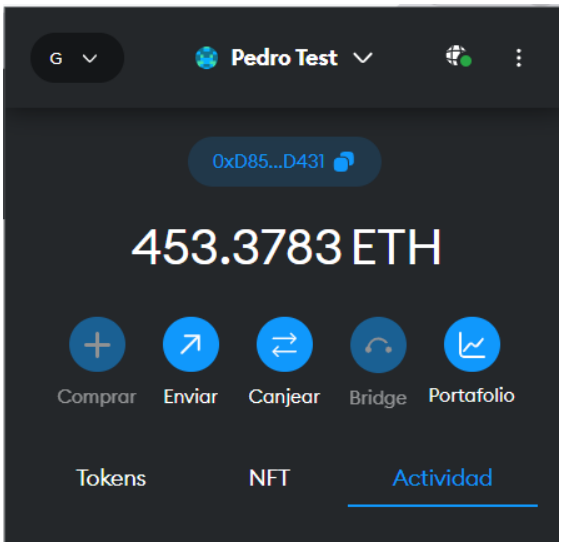


Fig 29. Cuenta Pedro Test

Fuente: Autor

En las Figuras 30, 31, 32, 33 y 34 muestra como seria la carga de las cuentas de Ganache a la wallet de Metamask.

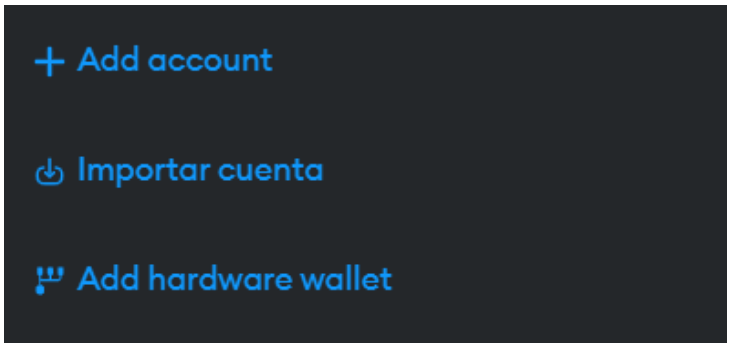


Fig 30. Interfaz para añadir cuentas

Fuente: Autor

Fig 31. Información de la llave privada de una cuenta de Ganache

Fuente: Autor

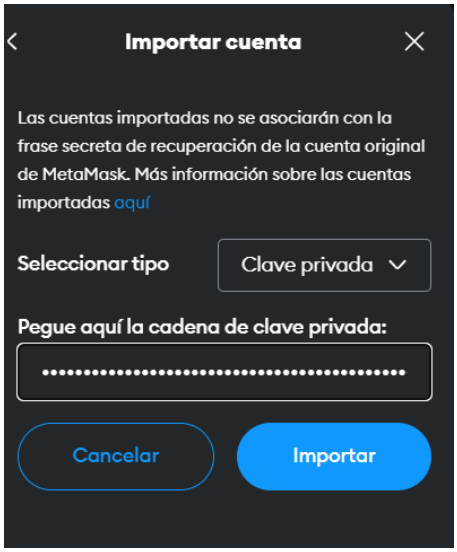


Fig 32. Importación de la llave privada de la cuenta de Ganache a Metamask

Fuente: Autor

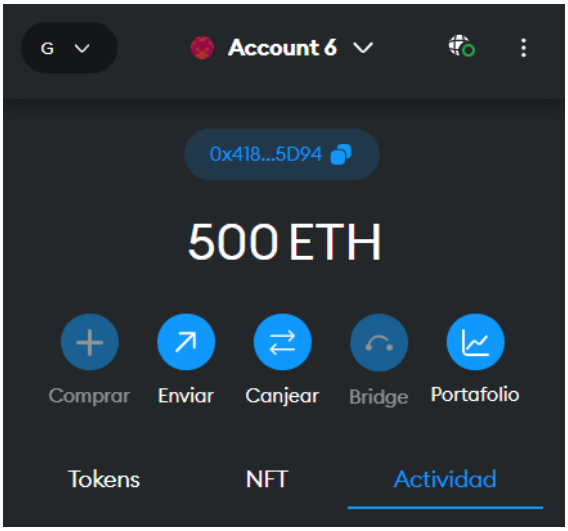


Fig 33. Cuenta de Ganache ya importada en Metamask

Fuente: Autor

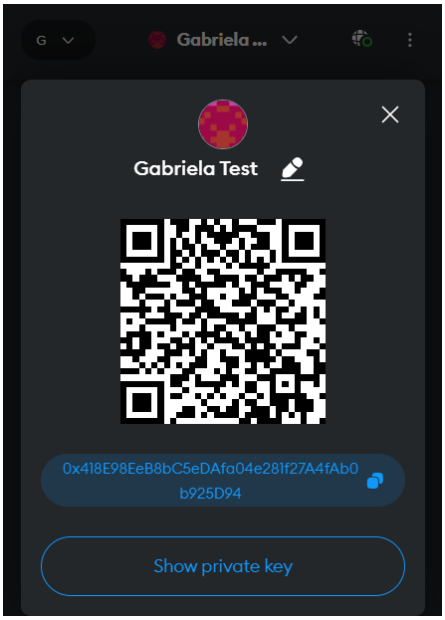


Fig 34. Cambio de nombre a la cuenta

Fuente: Autor

En las Figura 35 muestra la interfaz de la DApp sin una wallet conectada.

Fig 35. Interfaz de Inicio sin wallet conectada

Fuente: Autor

En la Figura 36 muestra que al momento de pulsar el botón conectar wallet sale el llamado a la extensión de Metamask y el login de Metamask.

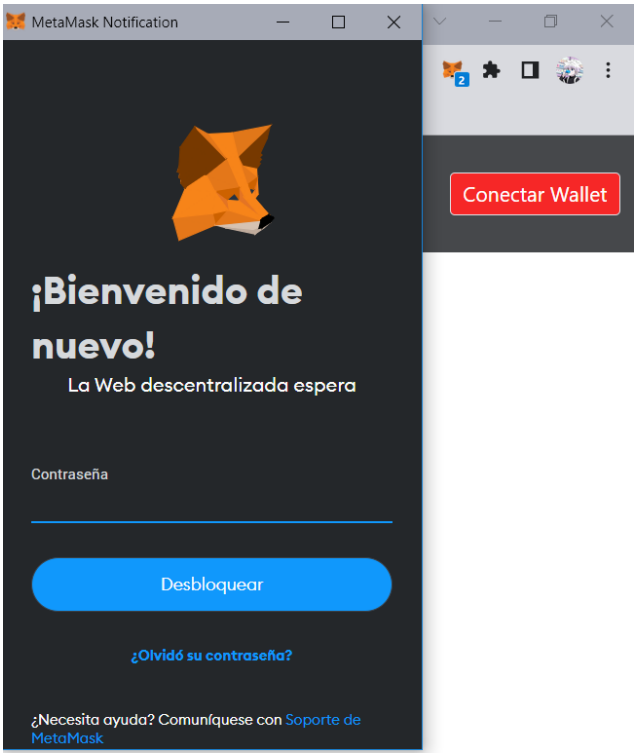


Fig 36. Conexión de la wallet

Fuente: Autor

En la Figura 37 muestra como es la interfaz de la DApp con una wallet conectada y usando una de las cuentas de Ganache.

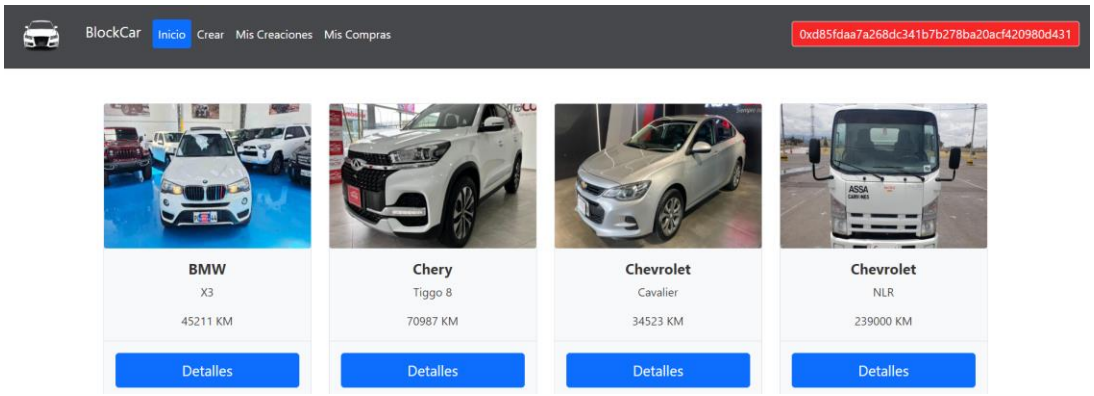


Fig 37. Interfaz de Inicio con wallet Conectada

Fuente: Autor

La tabla XVII presenta el Taskboard correspondiente al Sprint 1, donde se indica que la historia de usuario "Gestionar usuario" ha sido finalizada.

Tabla XVIII. Final de Taskboard del Sprint 1

N. SPRINT	INICIO: 05/JUNIO/2023 FIN: 03/JULIO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	EN CURSO	FINALIZADO
SPRINT 1	Crear un vehículo			✓
	Revender un vehículo			✓
	Gestionar usuario			✓
SPRINT 2	Ver vehículos comprados	✓		
	Ver vehículos creados	✓		
SPRINT 3	Ver trazabilidad de un vehículo	✓		
	Diseñar una interfaz de usuario amigable	✓		
SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros	✓		
	Integrar un sistema de pagos utilizando Blockchain	✓		

Fuente: Autor

La Figura 38 refleja el Sprint 1 y señala que la historia de usuario "Gestionar usuario" ha sido concluida.

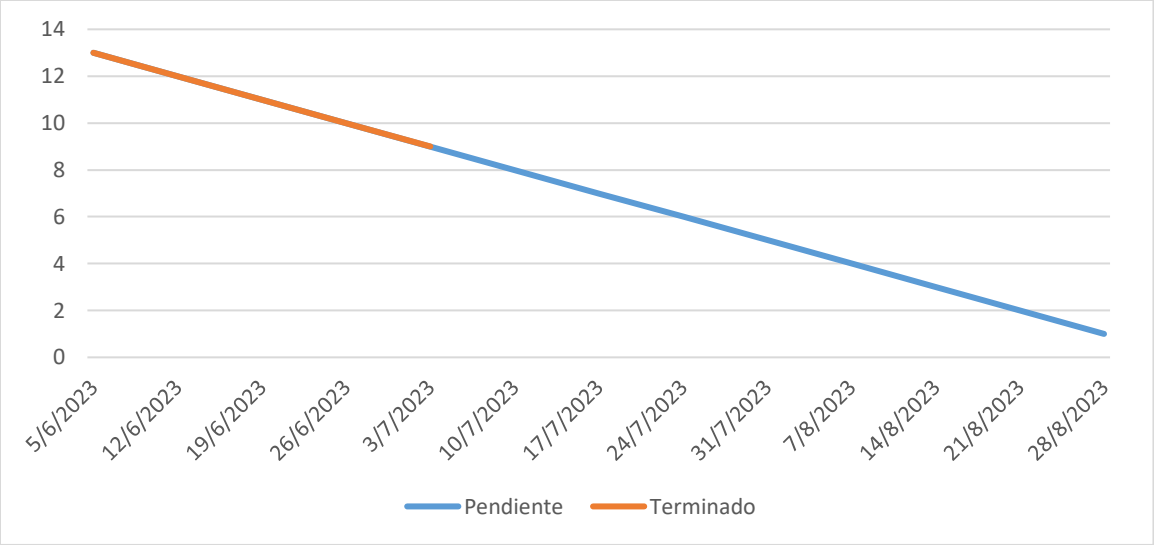


Fig 38. Final de Burndown del Sprint 1

Fuente: Autor

3.6.2 Sprint 2

Durante el Sprint 2, se llevarán a cabo las siguientes actividades:

- Ver vehículos comprados
- Ver vehículos creados

La tabla XVIII presenta el Taskboard correspondiente al Sprint 2, donde se observa que la historia de usuario "Ver vehículos comprados" está actualmente en curso.

Tabla XIX. Taskboard del Sprint 2

N. SPRINT	INICIO: 03/JULIO/2023 FIN: 17/JULIO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	EN CURSO	FINALIZADO
SPRINT 1	Crear un vehículo			✓
	Revender un vehículo			✓
	Gestionar usuario			✓
SPRINT 2	Ver vehículos comprados		✓	
	Ver vehículos creados	✓		
SPRINT 3	Ver trazabilidad de un vehículo	✓		
	Diseñar una interfaz de usuario amigable	✓		

SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros	✓		
	Integrar un sistema de pagos utilizando Blockchain	✓		

Fuente: Autor

La Figura 39 muestra el progreso alcanzado en el Sprint 2 a través del grafico Burndown.

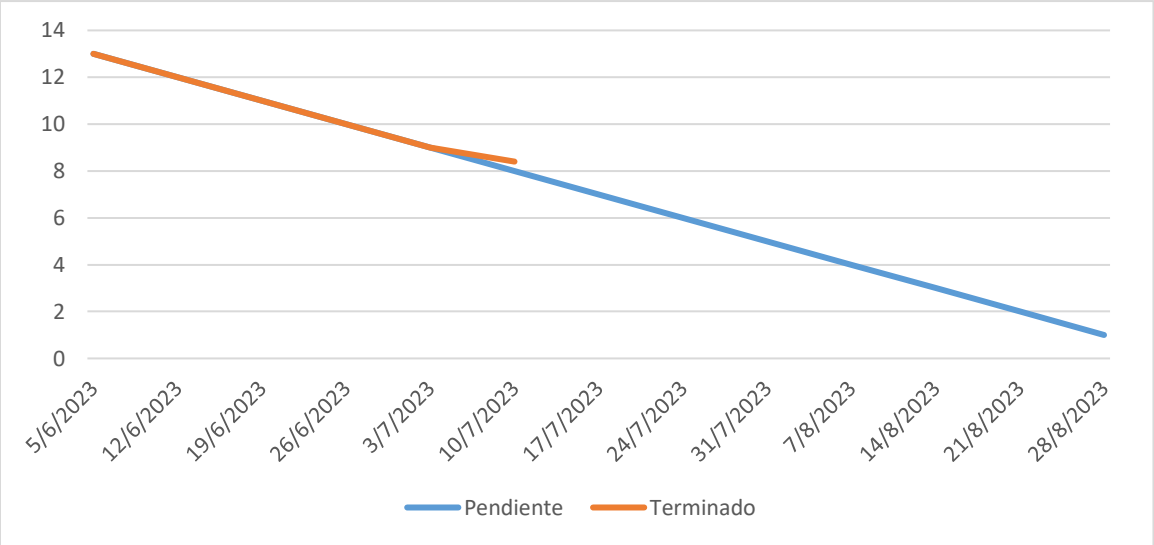
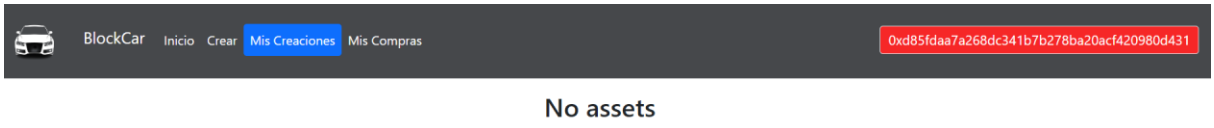


Fig 39. Burndown del Sprint 2

Fuente: Autor

3.6.2.1 Ver vehículos comprados

En la Figura 40 muestra el módulo de Mis Creaciones cuando un usuario no tenga vehículos creados.



No assets

Fig 40. Interfaz de Mis Creaciones sin vehículos creados

Fuente: Autor

En la Figura 41 muestra el módulo de Mis Creaciones cuando un usuario tenga vehículos

creados.

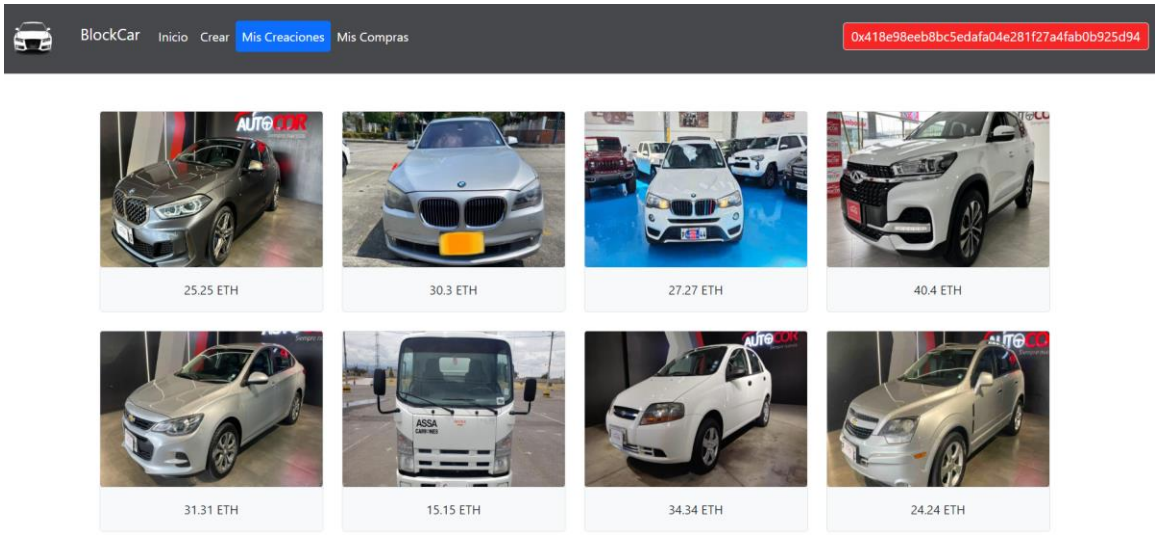


Fig 41. Interfaz de Mis Creaciones con vehículos creados

Fuente: Autor

La tabla XIX exhibe el Taskboard correspondiente al Sprint 2, donde se indica la historia de usuario "Ver vehículos comprados" ha sido finalizada, mientras que la historia "Ver vehículos creados" está actualmente en curso.

Tabla XX. Avance de Taskboard del Sprint 2

N. SPRINT	INICIO: 03/JULIO/2023 FIN: 17/JULIO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	EN CURSO	FINALIZADO
SPRINT 1	Crear un vehículo			✓
	Revender un vehículo			✓
	Gestionar usuario			✓
SPRINT 2	Ver vehículos comprados			✓
	Ver vehículos creados		✓	
SPRINT 3	Ver trazabilidad de un vehículo	✓		
	Diseñar una interfaz de usuario amigable	✓		
SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros	✓		
	Integrar un sistema de pagos utilizando Blockchain	✓		

Fuente: Autor

La Figura 42 presenta el progreso del Sprint 2, donde se puede observar que, dado que las

actividades están en curso, no han tenido un efecto negativo en el Burndown de desarrollo. Se mantienen los tiempos previstos para completar el proyecto.

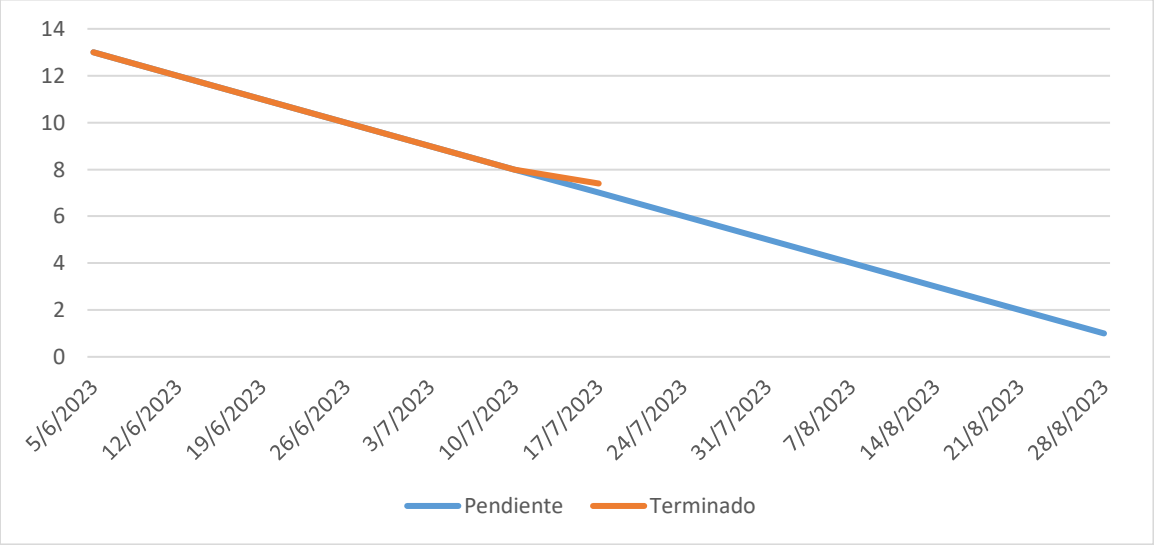


Fig 42. Avance de Burndown del Sprint 2

Fuente: Autor

3.6.2.2 Ver vehículos creados

En la Figura 43 muestra el módulo de Mis Compras cuando un usuario no tenga vehículos comprados.

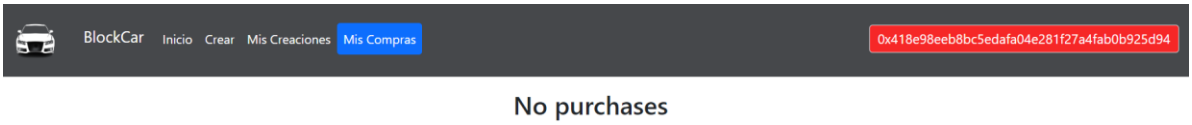


Fig 43. Interfaz de Mis Compras sin vehículos comprados

Fuente: Autor

En la Figura 44 muestra el módulo de Mis Compras cuando un usuario tenga vehículos comprados.

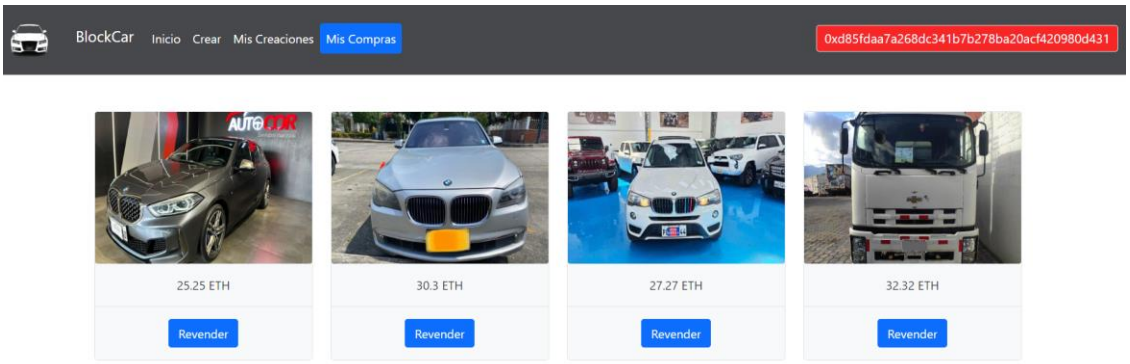


Fig 44. Interfaz de Mis Compras con vehículos comprados

Fuente: Autor

La tabla XX presenta el Taskboard correspondiente al Sprint 2, donde se indica que la historia de usuario "Ver vehículos creados" ha sido finalizada.

Tabla XXI. Final de Taskboard del Sprint 2

N. SPRINT	INICIO: 03/JULIO/2023 FIN: 17/JULIO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	EN CURSO	FINALIZADO
SPRINT 1	Crear un vehículo			✓
	Revender un vehículo			✓
	Gestionar usuario			✓
SPRINT 2	Ver vehículos comprados			✓
	Ver vehículos creados			✓
SPRINT 3	Ver trazabilidad de un vehículo	✓		
	Diseñar una interfaz de usuario amigable	✓		
SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros	✓		
	Integrar un sistema de pagos utilizando Blockchain	✓		

Fuente: Autor

La Figura 45 refleja el Sprint 2 y señala que la historia de usuario "Ver vehículos creados" ha sido concluida.

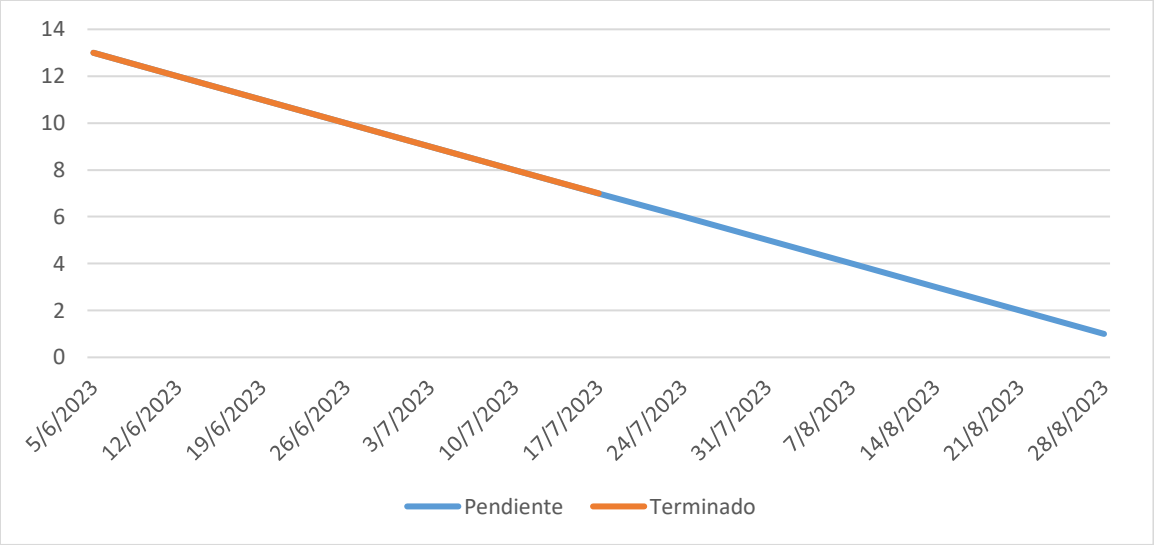


Fig 45. Final de Burndown del Sprint 2

Fuente: Autor

3.6.3 Sprint 3

Durante el Sprint 3, se llevarán a cabo las siguientes actividades:

- Ver trazabilidad de un vehículo
- Diseñar una interfaz de usuario amigable.

La tabla XXI presenta el Taskboard correspondiente al Sprint 3, donde se observa que la historia de usuario "Ver trazabilidad de un vehículo" está actualmente en curso.

Tabla XXII. Taskboard del Sprint 3

N. SPRINT	INICIO: 17/JULIO/2023 FIN: 07/AGOSTO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	EN CURSO	FINALIZADO
SPRINT 1	Crear un vehículo			✓
	Revender un vehículo			✓
	Gestionar usuario			✓
SPRINT 2	Ver vehículos comprados			✓
	Ver vehículos creados			✓
SPRINT 3	Ver trazabilidad de un vehículo		✓	
	Diseñar una interfaz de usuario amigable	✓		

SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros	✓		
	Integrar un sistema de pagos utilizando Blockchain	✓		

Fuente: Autor

La Figura 46 muestra el progreso alcanzado en el Sprint 3 a través del grafico Burndown.

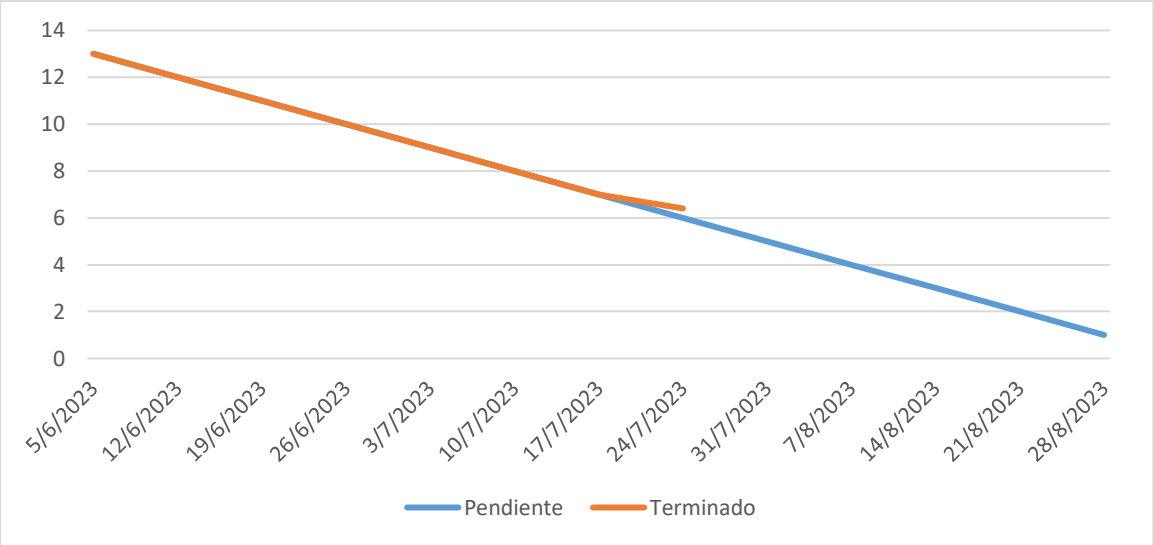


Fig 46. Burndown del Sprint 3

Fuente: Autor

3.6.3.1 Ver trazabilidad de un vehículo

En la Figura 47 muestra cuando un vehículo no tenga un historial de transacciones.

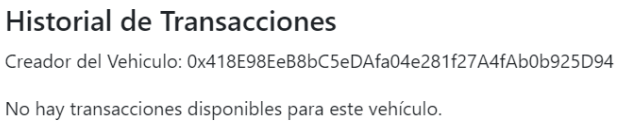


Fig 47. Interfaz del Historial de Transacciones sin Trazabilidad

Fuente: Autor

En la Figura 48 muestra cuando un vehículo tiene un historial de transacciones, se observa una tabla con los compradores que ha tenido el vehículo, a cuanto fue vendido y en qué fecha y hora fue comprado.

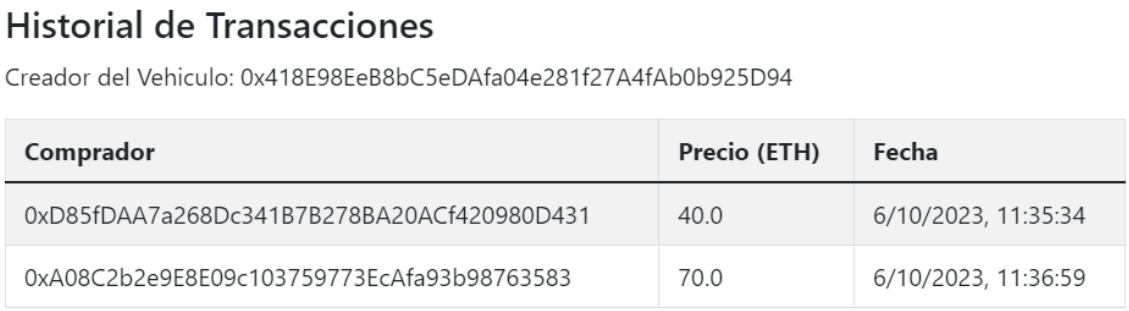


Fig 48. Interfaz del Historial de Transacciones con Trazabilidad

Fuente: Autor

La tabla XXII exhibe el Taskboard correspondiente al Sprint 3, donde se indica la historia de usuario "Ver trazabilidad de un vehículo" ha sido finalizada, mientras que la historia "Diseñar una interfaz de usuario amigable" está actualmente en curso.

Tabla XXIII. Avance de Taskboard del Sprint 3

N. SPRINT	INICIO: 17/JULIO/2023 FIN: 07/AGOSTO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	EN CURSO	FINALIZADO
SPRINT 1	Crear un vehículo			✓
	Revender un vehículo			✓
	Gestionar usuario			✓
SPRINT 2	Ver vehículos comprados			✓
	Ver vehículos creados			✓
SPRINT 3	Ver trazabilidad de un vehículo			✓
	Diseñar una interfaz de usuario amigable		✓	
SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros	✓		
	Integrar un sistema de pagos utilizando Blockchain	✓		

Fuente: Autor

La Figura 49 presenta el progreso del Sprint 3, donde se puede observar que, dado que las actividades están en curso, no han tenido un efecto negativo en el Burndown de desarrollo. Se mantienen los tiempos previstos para completar el proyecto.

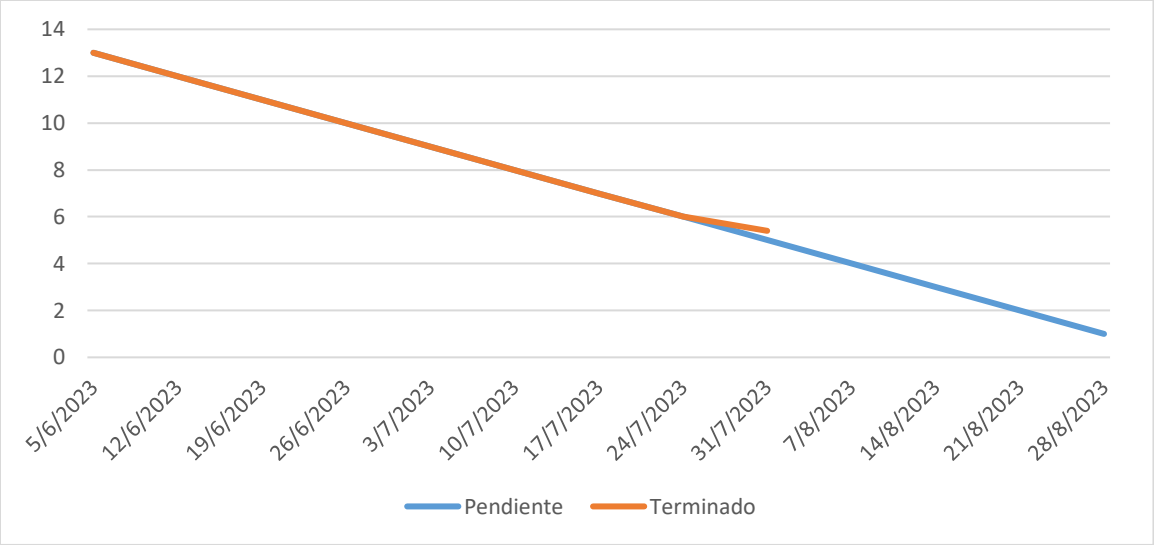


Fig 49. Avance de Burndown del Sprint 3

Fuente: Autor

3.6.3.2 Diseñar una interfaz de usuario amigable

En la Figura 50 muestra el diseño de la barra de navegación.

Fig 50. Diseño de la barra de navegación

Fuente: Autor

En la Figura 51 muestra el diseño de la interfaz de Inicio de la DApp.

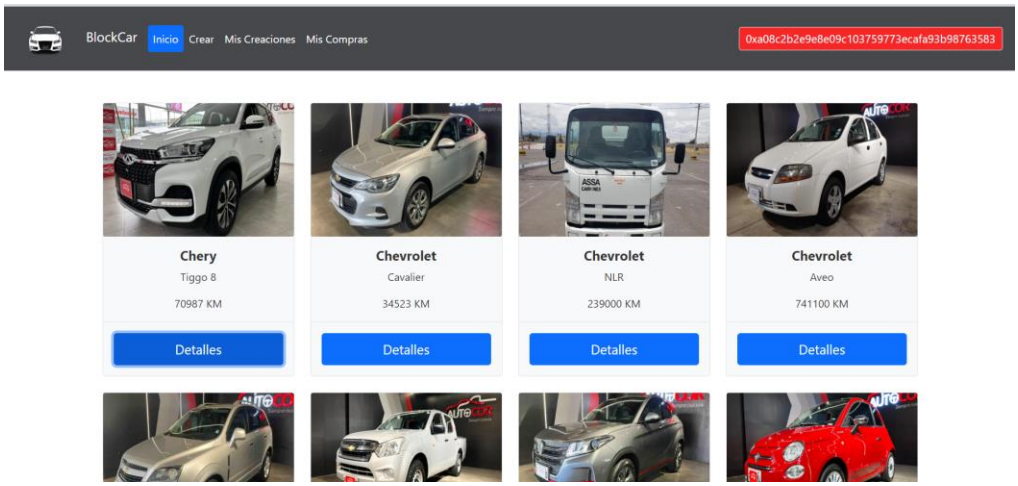


Fig 51. Diseño de Inicio

Fuente: Autor

En la Figura 52 muestra el diseño de la interfaz de Crear de la DApp.

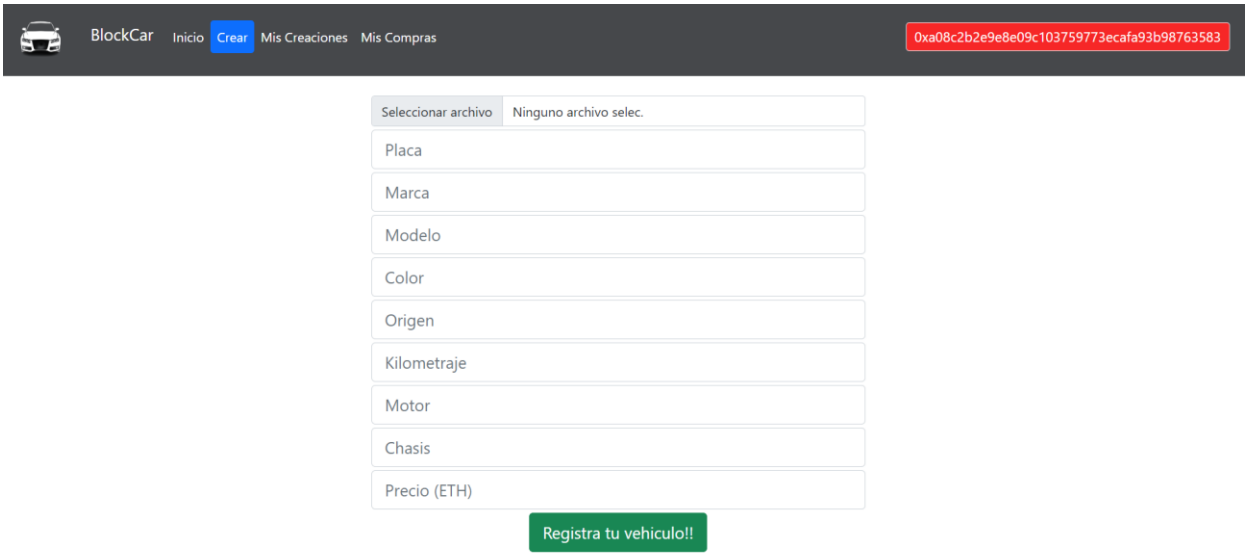


Fig 52. Diseño de Crear

Fuente: Autor

En la Figura 53 muestra el diseño de la interfaz de Mis Creaciones de la DApp.

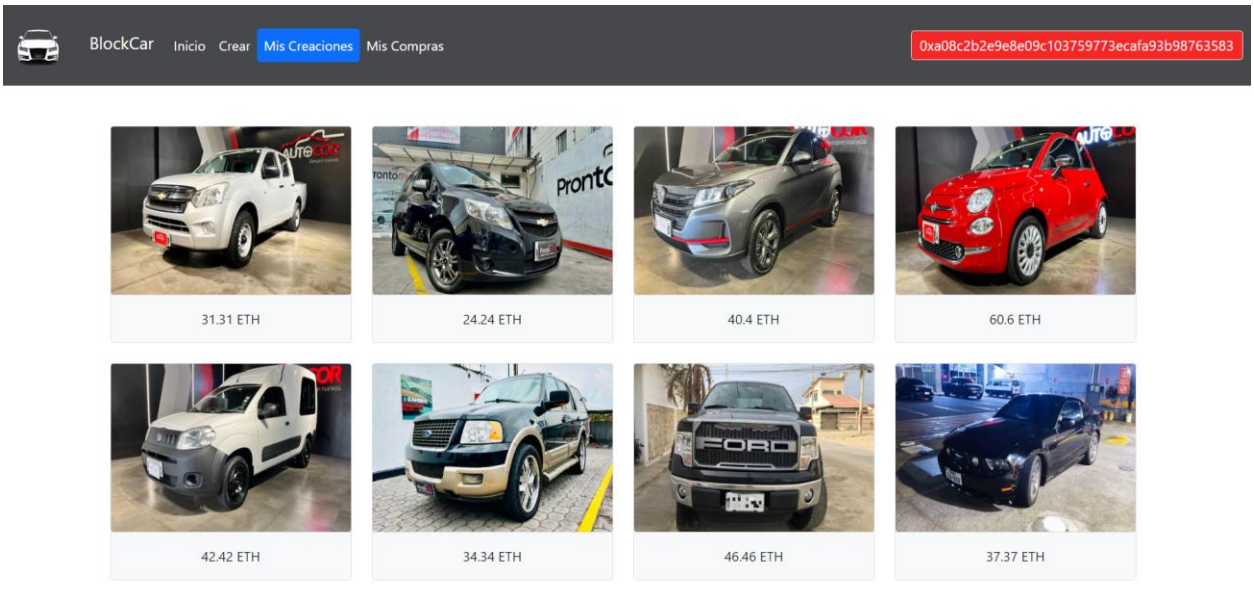


Fig 53. Diseño de Mis Creaciones

Fuente: Autor

En la Figura 54 muestra el diseño de la interfaz de Mis Compras de la DApp.

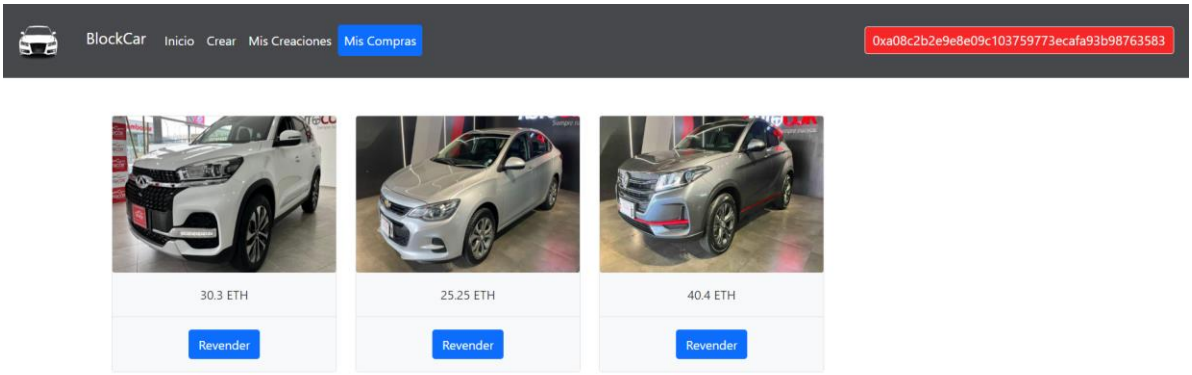


Fig 54. Diseño de Mis Compras

Fuente: Autor

En la Figura 55 muestra el diseño de los detalles de un vehículo.

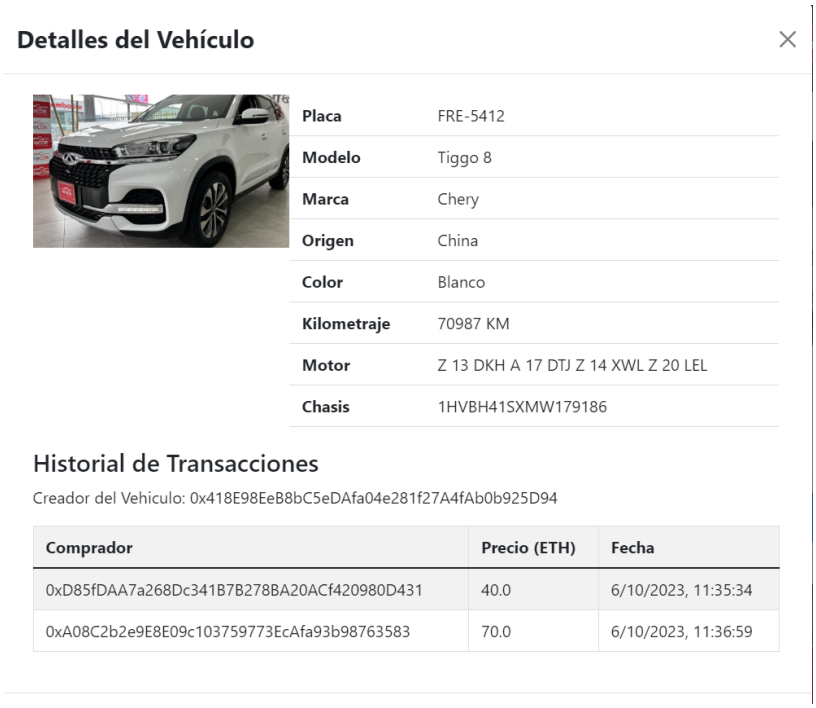


Fig 55. Diseño de detalles de un vehículo

Fuente: Autor

En las Figuras 56, 57 y 58 muestran las alertas de diseño que tiene la DApp cuando se hacen los procesos de crear, comprar y revender un vehículo.

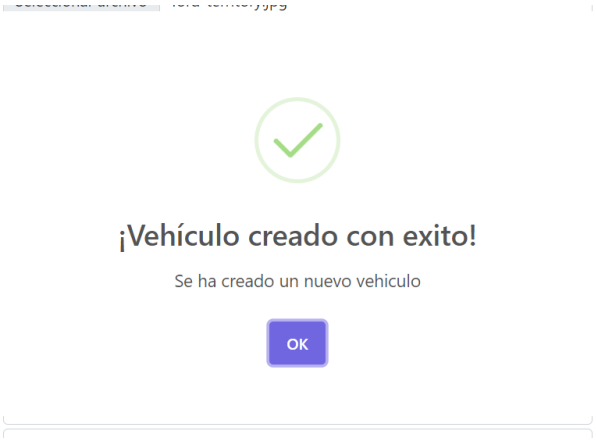


Fig 56. Alerta de crear un vehículo

Fuente: Autor

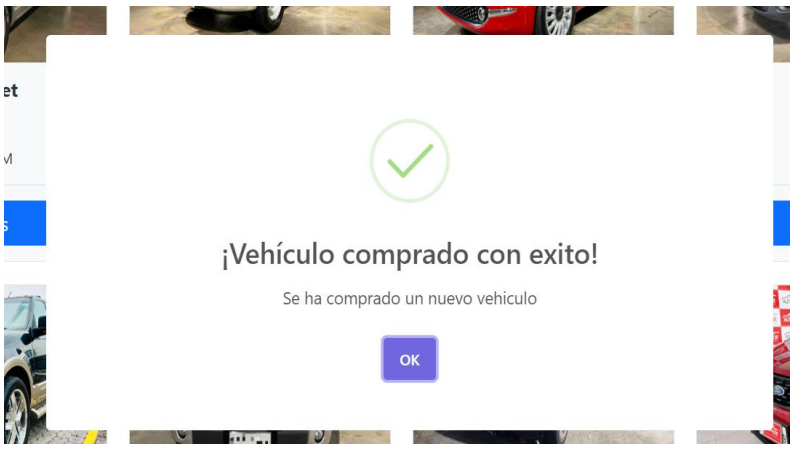


Fig 57. Alerta de comprar un vehículo

Fuente: Autor

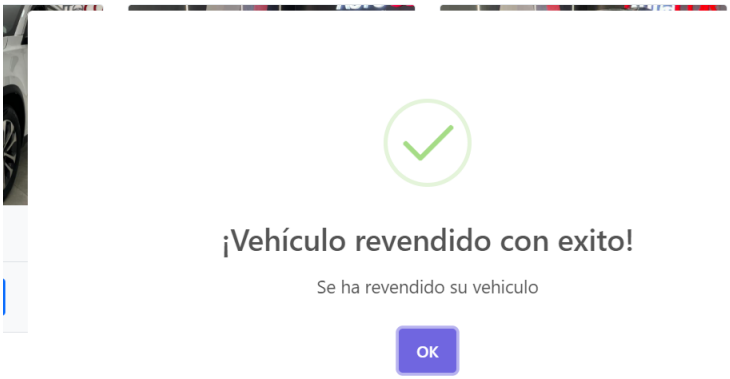


Fig 58. Alerta de revender un vehículo

Fuente: Autor

La tabla XIII presenta el Taskboard correspondiente al Sprint 3, donde se indica que la historia de usuario “Diseñar una interfaz de usuario amigable” ha sido finalizada.

Tabla XXIV.Final de Taskboard del Sprint 3

N. SPRINT	INICIO: 17/JULIO/2023 FIN: 07/AGOSTO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	EN CURSO	FINALIZADO
SPRINT 1	Crear un vehículo			✓
	Revender un vehículo			✓
	Gestionar usuario			✓
SPRINT 2	Ver vehículos comprados			✓
	Ver vehículos creados			✓
SPRINT 3	Ver trazabilidad de un vehículo			✓
	Diseñar una interfaz de usuario amigable			✓
SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros	✓		
	Integrar un sistema de pagos utilizando Blockchain	✓		

Fuente: Autor

La Figura 59 refleja el Sprint 3 y señala que la historia de usuario “Diseñar una interfaz de usuario amigable” ha sido concluida.

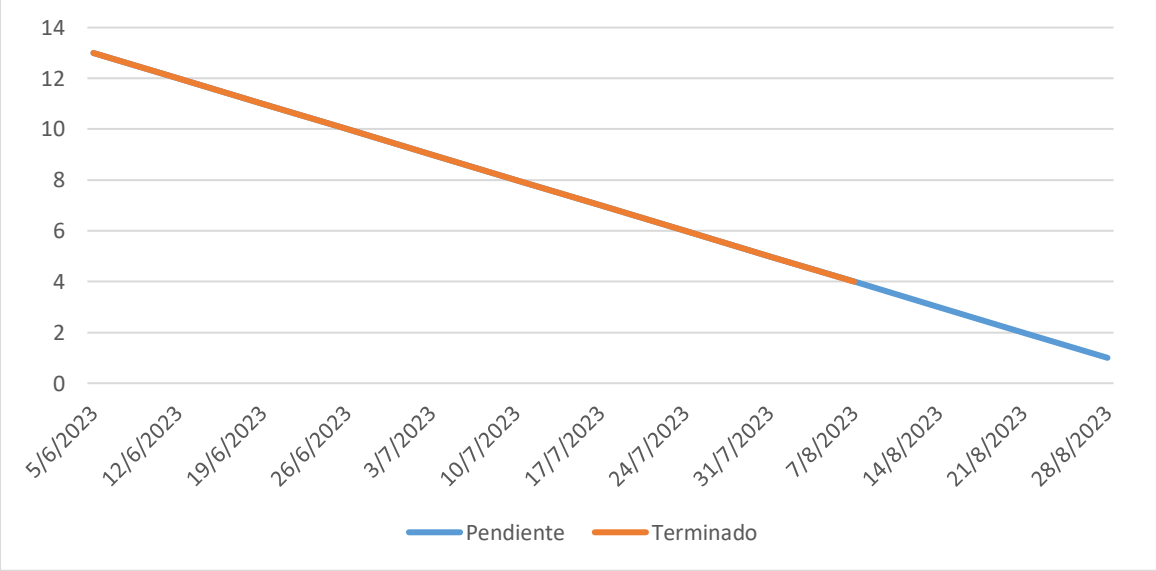


Fig 59. Final de Burndown del Sprint 3

Fuente: Autor

3.6.4 Sprint 4

Durante el Sprint 4, se llevarán a cabo las siguientes actividades:

- Diseñar un Marketplace que permita movimiento de productos seguros
- Integrar un sistema de pagos utilizando Blockchain.

La tabla XXIV presenta el Taskboard correspondiente al Sprint 4, donde se observa que la historia de usuario “Diseñar un Marketplace que permita movimiento de productos seguros” está actualmente en curso.

Tabla XXV. Taskboard del Sprint 4

N. SPRINT	INICIO: 05/JUNIO/2023 FIN: 28/AGOSTO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	EN CURSO	FINALIZADO
SPRINT 1	Crear un vehículo			✓
	Revender un vehículo			✓
	Gestionar usuario			✓
SPRINT 2	Ver vehículos comprados			✓
	Ver vehículos creados			✓
SPRINT 3	Ver trazabilidad de un vehículo			✓
	Diseñar una interfaz de usuario amigable			✓
SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros		✓	
	Integrar un sistema de pagos utilizando Blockchain	✓		

Fuente: Autor

La Figura 60 muestra el progreso alcanzado en el Sprint 4 a través del grafico Burndown.

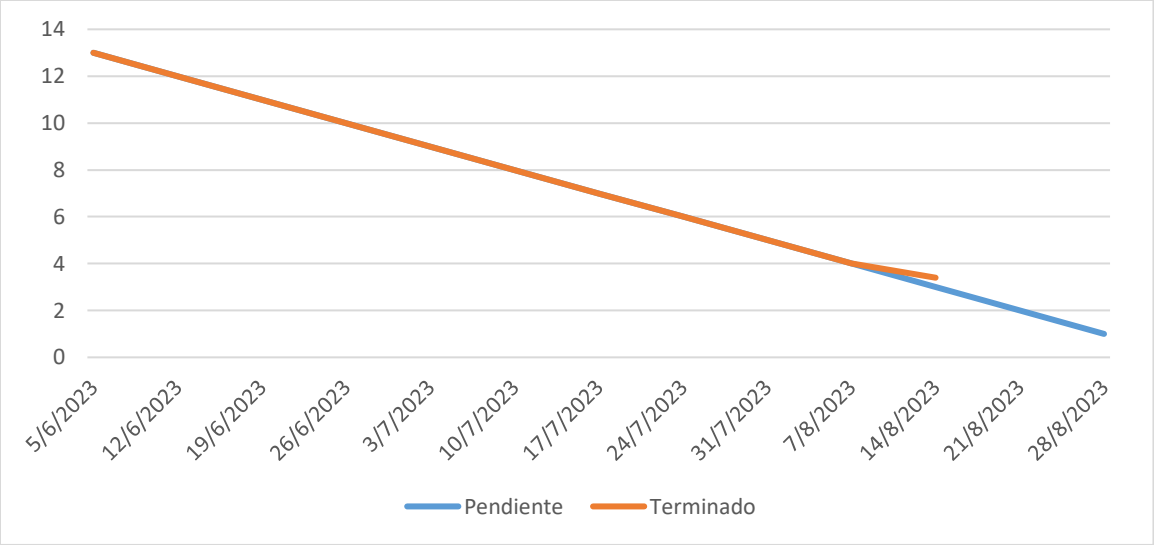


Fig 60. Burndown del Sprint 4

Fuente: Autor

3.6.4.1 Diseñar un Marketplace que permita movimiento de productos seguros

En la Figura 61 muestra la interfaz sobre los vehículos que se han creado anteriormente.

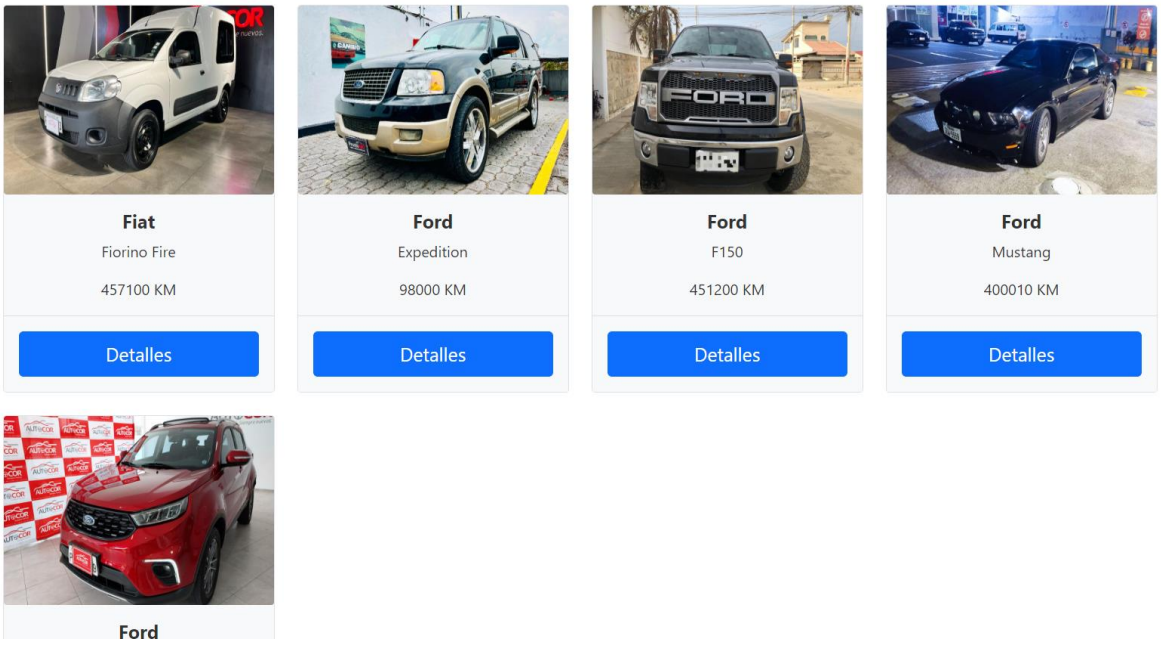


Fig 61. Interfaz vehículos

Fuente: Autor

En la Figura 62 muestra el formulario de Crear lleno para poder crear un vehículo en el Marketplace.

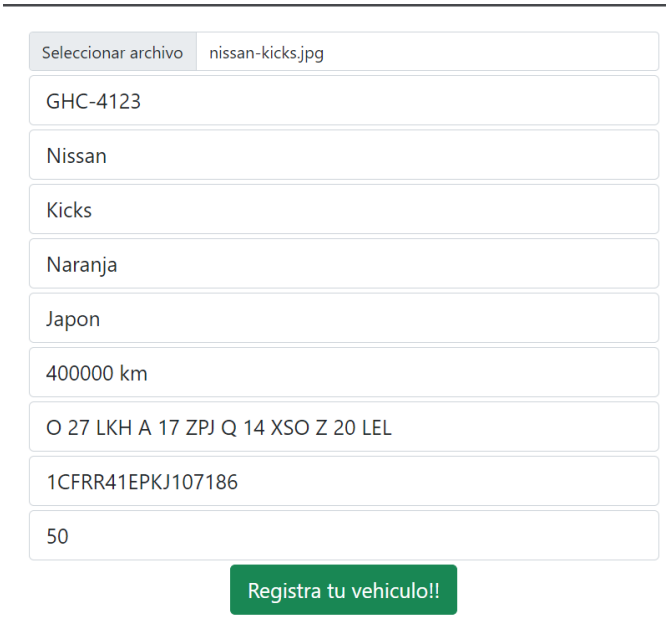


Fig 62. Formulario crear lleno

Fuente: Autor

En la Figura 63 muestra la interfaz actualizada con el nuevo vehículo que se creó para hacer su respectiva venta.

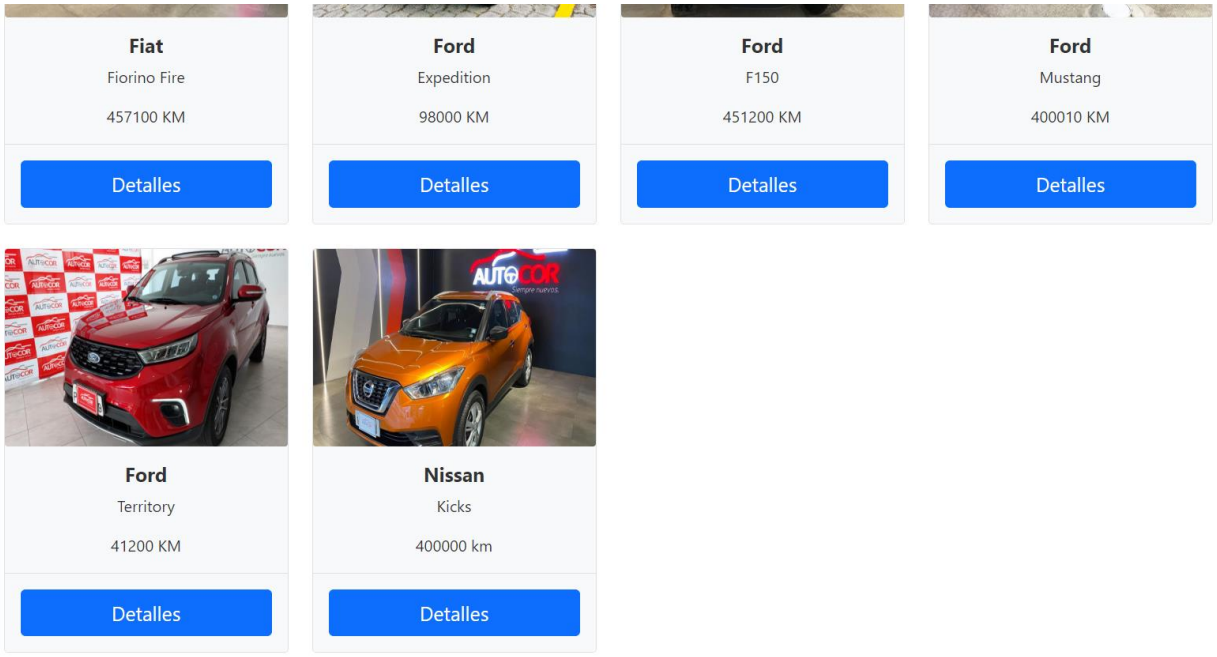


Fig 63. Interfaz vehículos actualizada

Fuente: Autor

En la Figura 64 muestra los detalles del vehículo y a cuento lo está vendiendo el usuario.

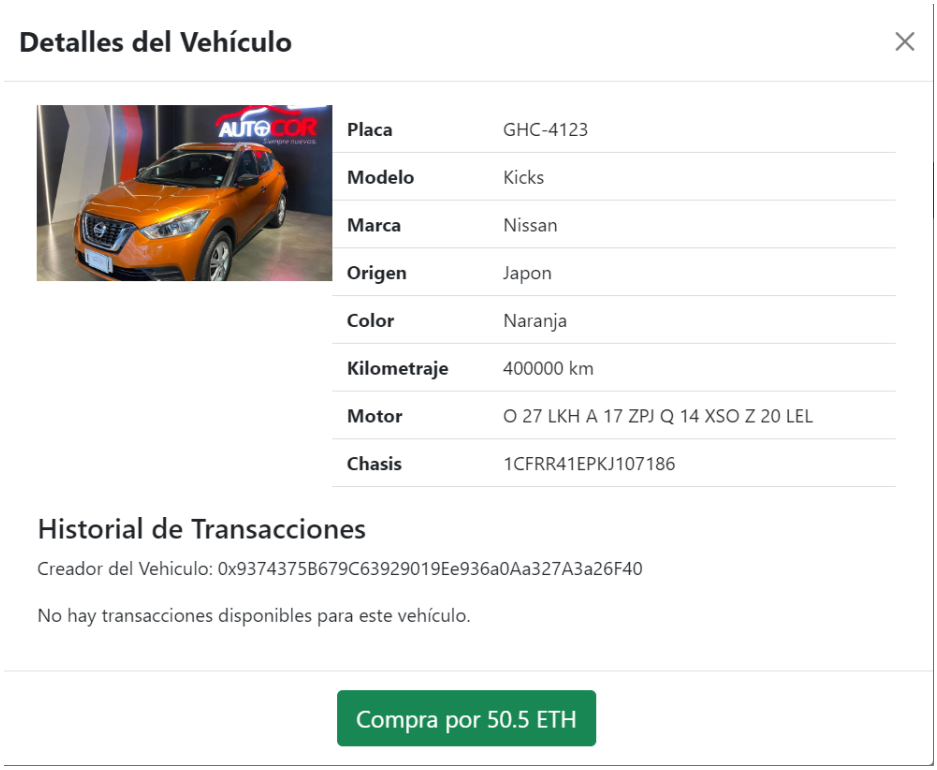


Fig 64. Información del vehículo.

Fuente: Autor

En la Figura 65 muestra que un usuario ha comprado el vehículo y ahora le pertenece.

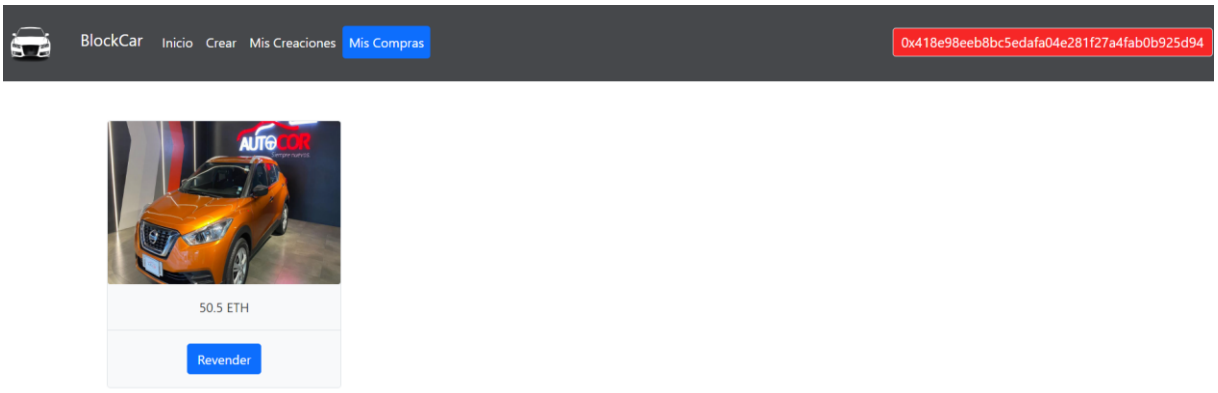


Fig 65. Interfaz de vehículo adquirido

Fuente: Autor

En la Figura 66 muestra que el vehículo comprado no está disponible a la venta en el Marketplace.

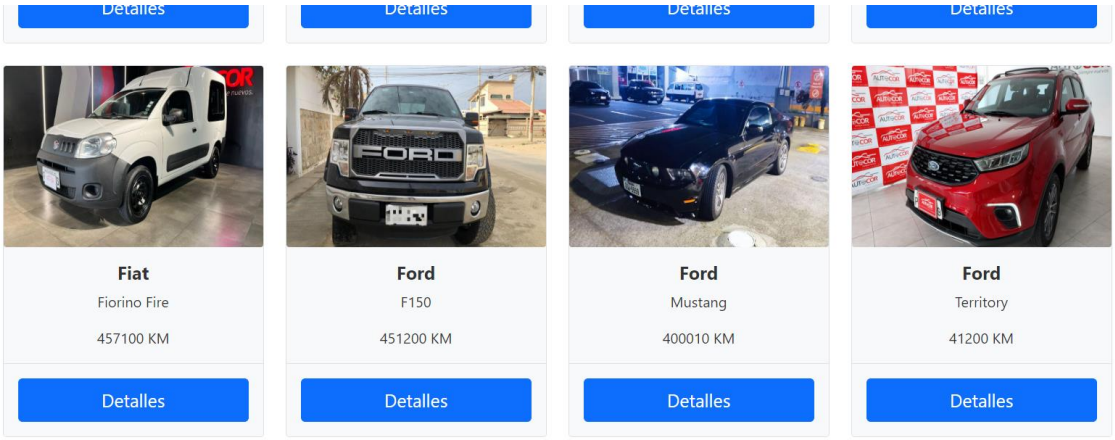


Fig 66. Interfaz vehículos actualizada 2

Fuente: Autor

En la Figura 67 muestra el formulario lleno en donde un usuario quiere revender su vehículo a un nuevo precio.

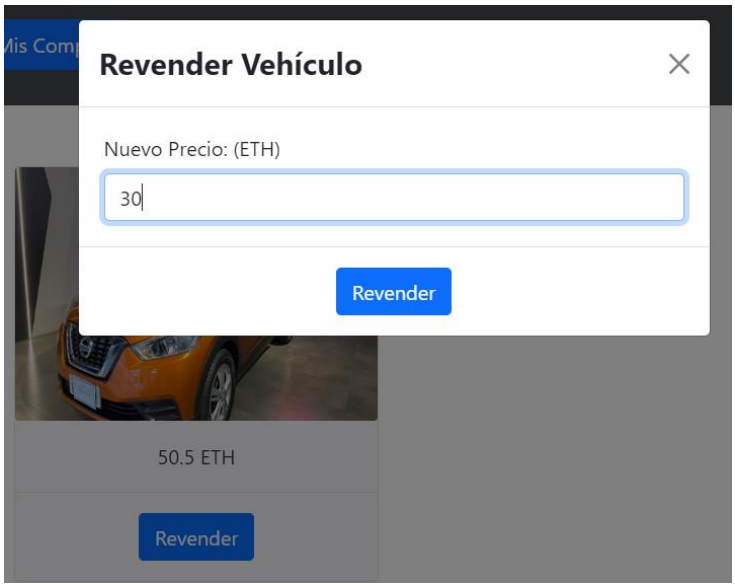


Fig 67. Formulario revender lleno

Fuente: Autor

En la Figura 68 muestra que el vehículo se puso de nuevo a la venta con un nuevo precio.

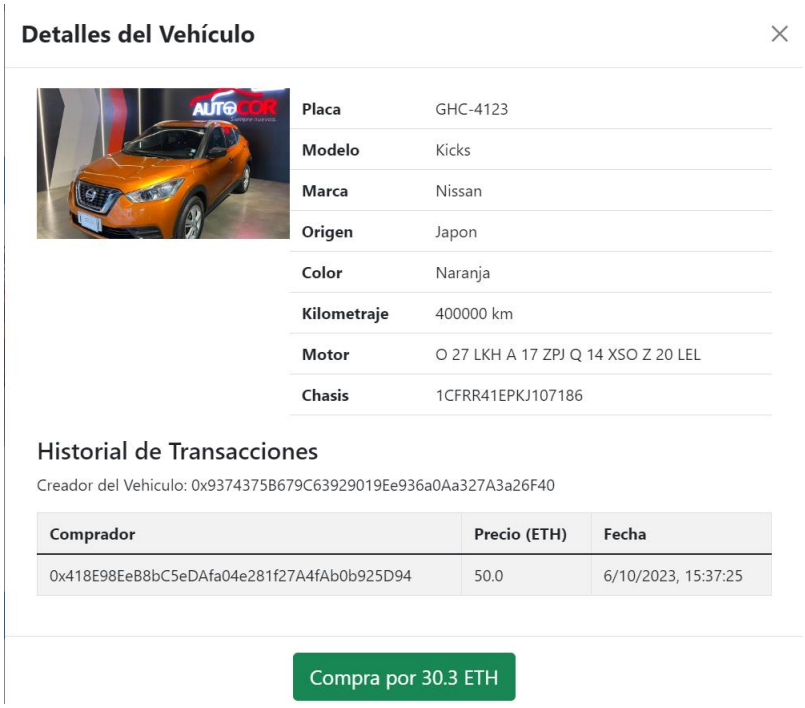


Fig 68. Información de vehículo actualizada

Fuente: Autor

La tabla XXV exhibe el Taskboard correspondiente al Sprint 4, donde se indica la historia de usuario “Diseñar un Marketplace que permita movimiento de productos seguros” ha sido finalizada, mientras que la historia “Integrar un sistema de pagos utilizando Blockchain” está actualmente en curso.

Tabla XXVI.Avance de Taskboard Sprint 4

N. SPRINT	INICIO: 05/JUNIO/2023 FIN: 28/AGOSTO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	ENCURSO	FINALIZADO
SPRINT 1	Crear un vehículo			✓
	Revender un vehículo			✓
	Gestionar usuario			✓
SPRINT 2	Ver vehículos comprados			✓
	Ver vehículos creados			✓
SPRINT 3	Ver trazabilidad de un vehículo			✓
	Diseñar una interfaz de usuario amigable			✓
SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros			✓

	Integrar un sistema de pagos utilizando Blockchain		✓	

Fuente: Autor

La Figura 69 presenta el progreso del Sprint 4, donde se puede observar que, dado que las actividades están en curso, no han tenido un efecto negativo en el Burndown de desarrollo. Se mantienen los tiempos previstos para completar el proyecto.

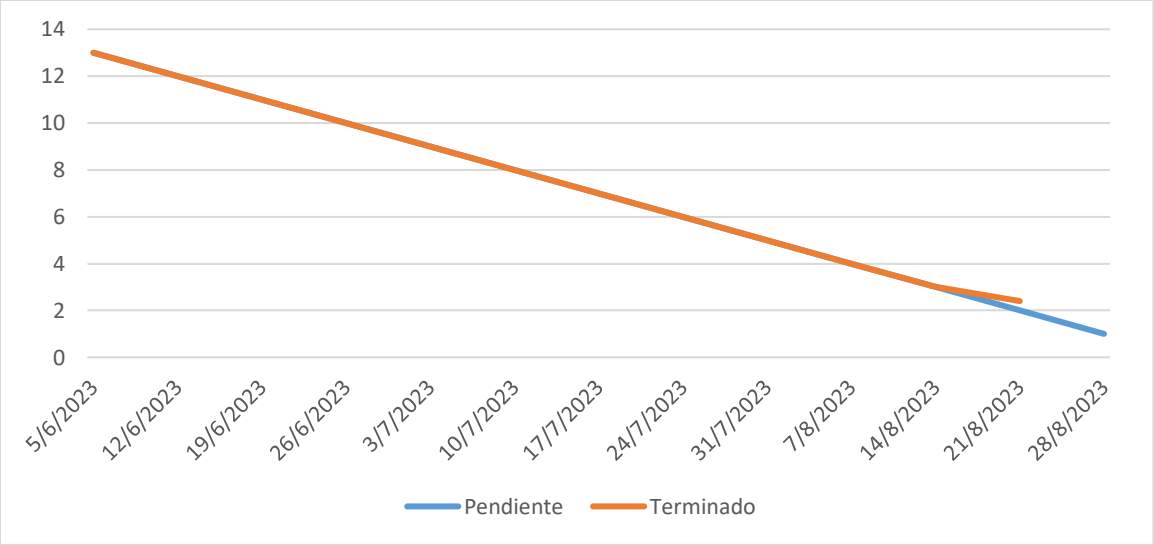


Fig 69. Avance de Burndown del Sprint 4

Fuente: Autor

3.6.4.2 Integrar un sistema de pagos utilizando Blockchain

En la Figura 70 muestra el valor que tiene el vehículo que un usuario desea comprar.

Fig 70. Precio de un vehículo

Fuente: Autor

En la Figura 71 muestra cómo se realiza el pago del vehículo que el usuario desea comprar.

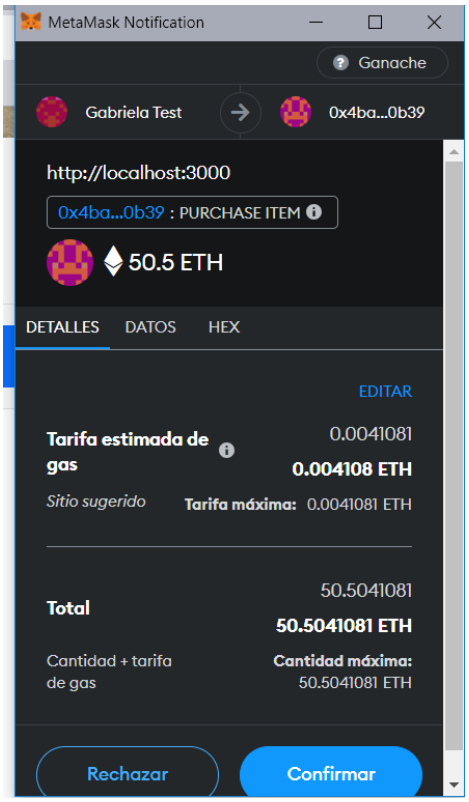


Fig 71. Pago mediante Metamask

Fuente: Autor

En la Figura 72 muestra la alerta de que un vehículo ha sido comprado.

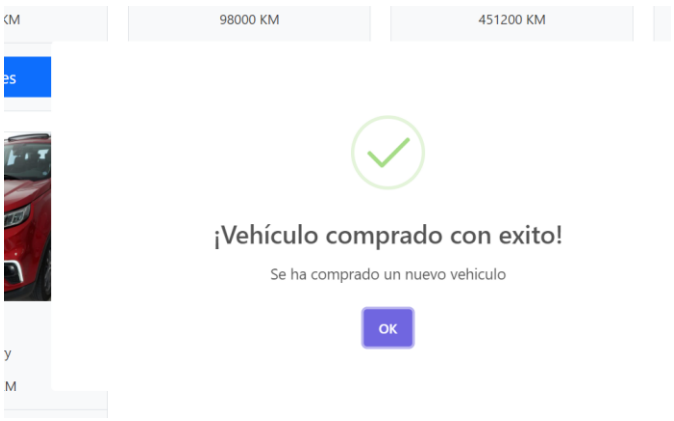


Fig 72. Alerta del vehículo comprado

Fuente: Autor

La tabla XXVI presenta el Taskboard correspondiente al Sprint 4, donde se indica que la historia de usuario “Integrar un sistema de pagos utilizando Blockchain” ha sido finalizada.

Tabla XXVII. Final de Taskboard del Sprint 4

N. SPRINT	INICIO: 05/JUNIO/2023 FIN: 28/AGOSTO/2023 HISTORIA DE USUARIO	AUTOR: JOHN RIVERA DESARROLLO DEL SISTEMA PENDIENTE	EN CURSO	FINALIZADO
SPRINT 1	Crear un vehículo			✓
	Revender un vehículo			✓
	Gestionar usuario			✓
SPRINT 2	Ver vehículos comprados			✓
	Ver vehículos creados			✓
SPRINT 3	Ver trazabilidad de un vehículo			✓
	Diseñar una interfaz de usuario amigable			✓
SPRINT 4	Diseñar un Marketplace que permita movimiento de productos seguros			✓
	Integrar un sistema de pagos utilizando Blockchain			✓

Fuente: Autor

La Figura 73 refleja el Sprint 4 y señala que la historia de usuario “Integrar un sistema de pagos utilizando Blockchain” ha sido concluida.

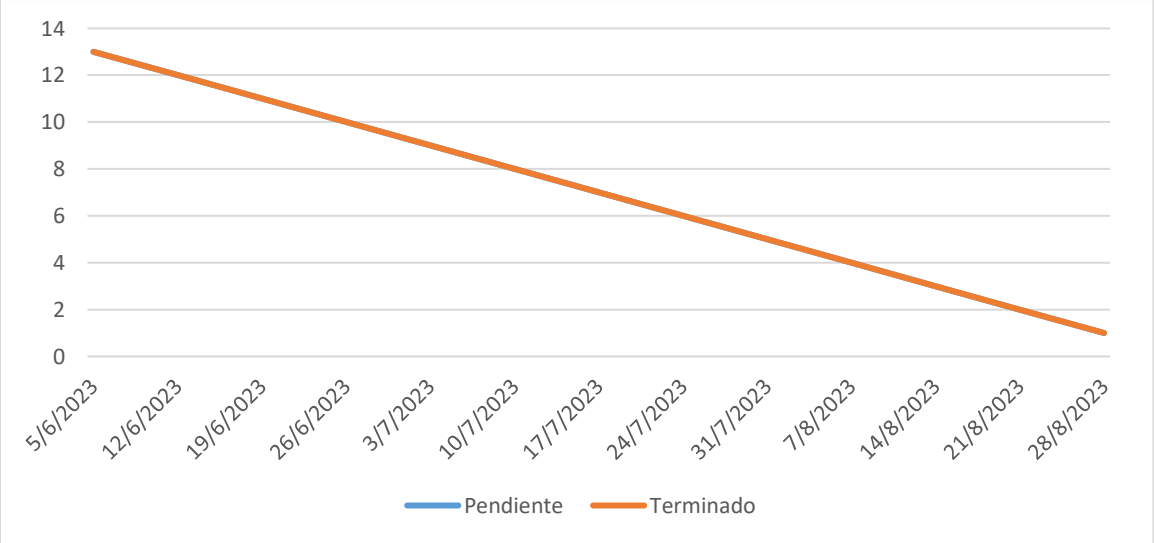


Fig 73. Final de Burndown del Sprint 4

Fuente: Autor

3.7 Cierre

Después de finalizar el ultimo sprint, se considera completo el desarrollo del prototipo. El desarrollo se ha realizado según lo previsto en la estimación de tiempo, por lo que no se obtuvo

retrasos y no tuvieron impacto en la entrega final del producto final. Esto se debe a que se planifico un 90% del tiempo en cada semana de desarrollo como un margen de trabajo optimo, mientras que el 10% restante se reservó para posibles retrasos que pueda surgir.

La tabla XXVII presenta el informe final del entregable, y tras la finalización de los sprints es posible verificar la funcionalidad del prototipo.

Tabla XXVIII. Informe Final del Entregable

HISTORIA DE USUARIO	RESPUESTA APLICADA	FECHA INICIO	FECHA FINAL	AVANCE (%)	OBSERVACIÓN
Crear un vehículo	Permite la creación	05/06/23	03/07/23	20%	Ninguna
Revender un vehículo	Permite revender	05/06/23	03/07/23	30%	Ninguna
Gestionar usuarios	Permite el acceso del sistema	05/06/23	03/07/23	40%	Ninguna
Ver vehículos comprados	Muestra el listado	03/06/23	17/07/23	50%	Necesita información cargada de los anteriores
Ver vehículos creados	Muestra el listado	03/06/23	17/07/23	60%	Necesita información cargada de los anteriores
Ver trazabilidad de un vehículo	Muestra las transacciones	17/07/23	07/08/23	70%	Necesita información cargada de los anteriores
Diseñar una interfaz de usuario amigable	Muestra una interfaz segura	17/07/23	07/08/23	80%	Ninguna
Diseñar un Marketplace que permita movimiento de productos seguros	Muestra una interfaz segura	07/08/23	28/08/23	90%	Ninguna
Integrar un sistema de pagos utilizando Blockchain	Permite pagos por criptomonedas	07/08/23	28/08/23	100%	Ninguna

Fuente: Autor

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La investigación detallada sobre la tecnología Blockchain y proyectos relacionados, proporcionaron una base sólida para comprender la complejidad de la problemática y su aplicación en plataformas. Con este análisis se pudo identificar casos de éxito y desafíos que aportaron en la toma de decisiones durante el desarrollo del prototipo.

- La metodología Scrum fue empleada en el desarrollo del prototipo, obteniendo un producto funcional. El software resultante no solo satisface las necesidades específicas de los usuarios, sino que también se adapta de manera eficiente al entorno del Marketplace de vehículos, demostrando ser una solución practica y orientada al usuario.

- La integración e implementación de la plataforma a través de la Blockchain de Ethereum ha sido un logro significativo. La trazabilidad completa de cada vehículo, asegurado la transparencia y confiabilidad de las transacciones, confirma la efectividad de la integración con esta tecnología de registro distribuido.

- La evaluación del rendimiento de la plataforma, respaldada por un informe final del entregable, ha generado datos de gran relevancia. Las optimizaciones implementadas han mejorado el prototipo, garantizando la conformidad con los requisitos establecidos y ofreciendo una experiencia de usuario eficiente y efectiva.

4.2 Recomendaciones

- Se sugiere desplegar la DApp en entornos Blockchain confiables como Rinkeby, Polygon, entre otros, para fortalecer la seguridad de los datos transaccionales y garantizar la integridad de la información.

- Implementar protocolos de seguridad avanzados para salvaguardar los datos de los usuarios, especialmente al permitir la creación de cuentas personales vinculadas a sus wallets, asegurando el almacenamiento seguro de información sensible.

- Mejorar la gestión de usuarios mediante la habilitación de la creación autónoma de cuentas en el Marketplace, lo que permitirá una mayor escalabilidad y flexibilidad en el crecimiento de la plataforma.

- Considerar soluciones de búsqueda avanzada y filtros para optimizar la experiencia del usuario al buscar vehículos, garantizando una escalabilidad sin comprometer la eficiencia.

- Integrar un sistema de conversión de criptomonedas a la monedas local utilizada, basado en datos en tiempo real, para ofrecer una experiencia transparente y conveniente a los usuarios.

- Establecer un mecanismo de recopilación y análisis de datos para identificar patrones de uso y preferencias de los usuarios, facilitando la toma de decisiones informadas para mejoras continuas en la plataforma.

REFERENCIAS BIBLIOGRÁFICAS

[1] «BLOCKCHAIN EN LA INDUSTRIA AUTOMOTRIZ». Accedido: 13 de octubre de 2023. [En línea]. Disponible en: https://es.linkedin.com/pulse/blockchain-en-la-industria-automotriz-oscar-gabriel-amerio

[2] J. Oddone, «Aplicación de Blockchain y smart contracts en la compra-venta de vehículos usados», *Rev. Blockchain E Intel. Artif.*, n.º 2, jul. 2021, doi: 10.22529/rbia.2021(2)11.

[3] A. Goel, «Blockchain Technology: Revolutionizing Tax Collection for Greater Efficiency and Transparency», GCCfintax. Accedido: 13 de octubre de 2023. [En línea]. Disponible en: https://www.gccfintax.com/articles/blockchain-technology-revolutionizing-tax-collection-for-greater-efficiency-and-transparency-4142.asp

[4] «Tecnología blockchain y su potencial en la mejora de la confianza empresa-cliente», Tecnología para los negocios. Accedido: 13 de octubre de 2023. [En línea]. Disponible en: https://ticnegocios.camaravalencia.com/servicios/tendencias/tecnologia-blockchain-y-su-potencial-en-la-mejora-de-la-confianza-empresa-cliente/

[5] M. Frąckiewicz, «Blockchain en la industria automotriz», TS2 SPACE. Accedido: 13 de octubre de 2023. [En línea]. Disponible en: https://ts2.space/es/blockchain-en-la-industria-automotriz/

[6] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. O'Reilly Media, Inc., 2014.

[7] «Intro to Ethereum», ethereum.org. Accedido: 21 de abril de 2023. [En línea]. Disponible en: https://ethereum.org

[8] «Introduction to smart contracts», ethereum.org. Accedido: 21 de abril de 2023. [En línea]. Disponible en: https://ethereum.org

[9] P. Quevedo-Avila, M. G. Zhindón-Mora, y A. S. Quevedo-Sacoto, «Arquitectura de microservicios para compras en línea: caso de uso "ala orden" Microservices architecture for online shopping: "to order" use case».

[10] C. A. Carrillo Villalva, «Diseño y aplicación de un sistema de seguridad descentralizado mediante la tecnología Blockchain para aplicaciones web.», ago. 2021, Accedido: 21 de abril de 2023. [En línea]. Disponible en: http://dspace.espoch.edu.ec/handle/123456789/14695

[11] L. F. Pantoja Yumbo y J. D. Quisingo Núñez, «Propuesta para un modelo de sistematización en la gestión de derechos de autor aplicando Smart Contracts sobre una plataforma basada en Blockchain, para publicaciones científicas de la UCE», bachelorThesis, Quito : UCE, 2021. Accedido: 21 de abril de 2023. [En línea]. Disponible en: http://www.dspace.uce.edu.ec/handle/25000/25067

[12] J. C. Cañar Uyaguari y R. V. Jara Jara, «Análisis y desarrollo de una aplicación de registro de permisos y ausentismos sobre una Blockchain mediante un Smart Contract

desplegado en una testnet de Ethereum», bachelorThesis, 2022. Accedido: 21 de abril de 2023. [En línea]. Disponible en: http://dspace.ups.edu.ec/handle/123456789/22142

[13] A. I. Males Anagumbla y W. G. Gualoto Alvaro, «Desarrollo de un prototipo de aplicación web para la gestión de historias geriátricas utilizando Smart Contracts basados en Blockchain.», bachelorThesis, Quito : UCE, 2023. Accedido: 21 de abril de 2023. [En línea]. Disponible en: http://www.dspace.uce.edu.ec/handle/25000/29598

[14] J. C. Ortega Castro *et al.*, «Educación digital, blockchain y su influencia sobre la economía popular y solidaria», *Conrado*, vol. 19, n.º 90, pp. 252-259, feb. 2023.

[15] «2020-Scrum-Guide-Spanish-Latin-South-American.pdf».

[16] «¿Cómo funciona la metodología Scrum? Qué es y sus 5 fases», Platzi. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://platzi.com/blog/metodologia-scrum-fases/

[17] «Blockchain-Framework-and-Guidance_WBFGS_res_spa_0922.pdf».

[18] C. Dolader Retamal, J. Bel Roig, y J. L. Muñoz Tapia, «La blockchain : fundamentos, aplicaciones y relación con otras tecnologías disruptivas.», *Econ. Ind.*, n.º 405, pp. 33-40, 2017.

[19] «Conoce los distintos tipos de blockchain», Abierto al Público. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://blogs.iadb.org/conocimiento-abierto/es/tipos-de-blockchain/

[20] P. Gutiérrez, «¿Qué son y para qué sirven los hash?: funciones de resumen y firmas digitales», Genbeta. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://www.genbeta.com/desarrollo/que-son-y-para-que-sirven-los-hash-funciones-de-resumen-y-firmas-digitales

[21] «¿Qué es SHA-256?» Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://academy.bit2me.com/sha256-algoritmo-bitcoin/

[22] I. Bashir, *Mastering Blockchain*. Packt Publishing Ltd, 2017.

[23] A. Hasselgren, K. Kralevska, D. Gligoroski, S. A. Pedersen, y A. Faxvaag, «Blockchain in healthcare and health sciences—A scoping review», *Int. J. Med. Inf.*, vol. 134, p. 104040, feb. 2020, doi: 10.1016/j.ijmedinf.2019.104040.

[24] «Ganache | Overview - Truffle Suite». Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://trufflesuite.com/docs/ganache/

[25] «What is Ethereum Ganache? - Mycryptopedia». Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://www.mycryptopedia.com/what-is-ethereum-ganache/

[26] «¿Qué es Ethereum?», ethereum.org. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://ethereum.org

[27] «Ethereum», *Wikipedia, la enciclopedia libre*. 30 de septiembre de 2023. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Ethereum&oldid=154186401

[28] A. Lipton y S. Levi, «An Introduction to Smart Contracts and Their Potential and Inherent Limitations», The Harvard Law School Forum on Corporate Governance. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://corpgov.law.harvard.edu/2018/05/26/an-introduction-to-smart-contracts-and-their-potential-and-inherent-limitations/

[29] A. M. Antonopoulos, «Mastering Ethereum».

[30] M. Yano, C. Dai, K. Masuda, y Y. Kishimoto, Eds., *Blockchain and Crypto Currency: Building a High Quality Marketplace for Crypto Data*. en Economics, Law, and Institutions in Asia Pacific. Singapore: Springer Singapore, 2020. doi: 10.1007/978-981-15-3376-1.

[31] «Aplicación vs. dApp – Criptopolita». Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://www.cryptopolitan.com/es/aplicacion-contra-dapp/

[32] «Ethereum Virtual Machine (EVM)», ethereum.org. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://ethereum.org

[33] «¿Qué es una wallet o monedero de criptomonedas?» Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://academy.bit2me.com/wallet-monederos-criptomonedas/

[34] D. / M. H. Phillips Daniel, «¿Qué es MetaMask? Cómo usar la mejor wallet de Ethereum», Decrypt. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://decrypt.co/es/resources/que-es-metamask-como-utilizar-la-mejor-y-mas-popular-wallet-de-ethereum/

[35] «MetaMask», *Wikipedia, la enciclopedia libre*. 16 de junio de 2023. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=MetaMask&oldid=151884676

[36] «Documentation for Visual Studio Code». Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://code.visualstudio.com/docs

[37] «Visual Studio Code», *Wikipedia, la enciclopedia libre*. 29 de septiembre de 2023. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Visual_Studio_Code&oldid=154161101

[38] D. A, «Qué es React: definición, características y funcionamiento», Tutoriales Hostinger. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://www.hostinger.es/tutoriales/que-es-react

[39] «React», *Wikipedia, la enciclopedia libre*. 29 de septiembre de 2023. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=React&oldid=154167376

[40] «Bootstrap: ¿qué es, para qué sirve y cómo instalarlo?», Rock Content - ES. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://rockcontent.com/es/blog/bootstrap/

[41] «Bootstrap (framework)», *Wikipedia, la enciclopedia libre*. 13 de septiembre de 2023. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Bootstrap_(framework)&oldid=153726078

[42] «Solidity — Solidity 0.8.21 documentation». Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://docs.soliditylang.org/en/v0.8.21/

[43] «Solidity», *Wikipedia, la enciclopedia libre*. 29 de septiembre de 2023. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Solidity&oldid=154178248

[44] «JavaScript | MDN». Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript

[45] «Acerca», Node.js. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://nodejs.org/es/about

[46] «Node.js», *Wikipedia, la enciclopedia libre*. 22 de septiembre de 2023. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Node.js&oldid=153931504

[47] «Documentation». Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://docs.ethers.org/v5/

[48] «Documentation | Ethereum development environment for professionals by Nomic Foundation». Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://hardhat.org

[49] «What is IPFS? | IPFS Docs». Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://docs.ipfs.tech/concepts/what-is-ipfs/

[50] «Sistema de archivos interplanetario», *Wikipedia, la enciclopedia libre*. 24 de noviembre de 2022. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Sistema_de_archivos_interplanetario&oldid=147525119

ANEXOS

Anexo 1. Contrato BlockCar 1

Fuente: Autor

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

import "@openzeppelin/contracts/token/ERC721/IERC721.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";

contract BlockCar is ReentrancyGuard {
    address payable public immutable feeAccount;
    uint public immutable feePercent;
    uint public itemCount;

    struct Item {
        uint itemId;
        IERC721 nft;
        uint tokenId;
        uint price;
        address payable seller;
        address owner;
        bool sold;
    }

    struct Transaction {
        uint itemId;
        address seller;
        address buyer;
        uint price;
        uint timestamp;
    }

    mapping(uint => Item) public items;
    mapping(uint256 => Transaction[]) private _Transactions;

    event Offered(
        uint itemId,
        address indexed nft,
        uint tokenId,
        uint price,
        address indexed seller
    );

    event Bought(
        uint itemId,
        address indexed nft,
        uint tokenId,
        uint price,
        address indexed seller,
        address indexed buyer
    );

    event Resold(
        uint itemId,
        address indexed nft,
        uint tokenId,
        uint price,
        address indexed seller,
        address indexed buyer
    );

    constructor(uint _feePercent) {
        feeAccount = payable(msg.sender);
        feePercent = _feePercent;
    }
```

Anexo 2. Contrato BlockCar 2

Fuente: Autor

```
    function makeItem(IERC721 _nft, uint _tokenId, uint _price) external nonReentrant {
        require(_price > 0, "Price must be greater than zero");
        itemCount++;
        _nft.transferFrom(msg.sender, address(this), _tokenId);
        items[itemCount] = Item(
            itemCount,
            _nft,
            _tokenId,
            _price,
            payable(msg.sender),
            address(0),
            false
        );
        emit Offered(itemCount, address(_nft), _tokenId, _price, msg.sender);
    }

    function purchaseItem(uint _itemId) external payable nonReentrant {
        uint _totalPrice = getTotalPrice(_itemId);
        Item storage item = items[_itemId];
        require(_itemId > 0 && _itemId <= itemCount);
        require(msg.value >= _totalPrice);
        require(!item.sold);
        require(item.seller != address(0), "Item does not exist");
        require(item.owner != msg.sender, "You already own this item");
        item.seller.transfer(item.price);
        uint feeAmount = (_totalPrice - item.price);
        feeAccount.transfer(feeAmount);
        items[_itemId].owner = msg.sender;
        items[_itemId].sold = true;
        Transaction memory transaction = Transaction(_itemId, item.seller, msg.sender,  item.price,
        block.timestamp);
       _Transactions[_itemId].push(transaction);
        emit Bought(
            _itemId,
            address(item.nft),
            item.tokenId,
            item.price,
            item.seller,
            msg.sender
        );
    }

    function resellItem(uint _itemId, uint _price) external nonReentrant {
        Item storage item = items[_itemId];
        require(_itemId > 0 && _itemId <= itemCount, "Item does not exist");
        require( item.owner == msg.sender, "Only the current owner can resell the item" );
        item.sold = false;
        item.price = _price;
        emit Resold(
            _itemId,
            address(item.nft),
            item.tokenId,
            _price,
            msg.sender,
            address(0)
        );
    }

    function getTotalPrice(uint _itemId) public view returns (uint) {
        return ((items[_itemId].price * (100 + feePercent)) / 100);
    }

    function getTransactions(uint256 _itemId) public view returns (Transaction[] memory) {
        return _Transactions[_itemId];
    }

}
```

Anexo 3. Contrato NFT

Fuente: Autor

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";

contract NFT is ERC721URIStorage {

    uint public tokenCount;

    constructor() ERC721("BlockCar NFT", "BKC"){}

    function mint(string memory _tokenURI) external returns (uint) {
        tokenCount++;
        _safeMint(msg.sender, tokenCount);
        _setTokenURI(tokenCount, _tokenURI);
        return tokenCount;
    }
}
```

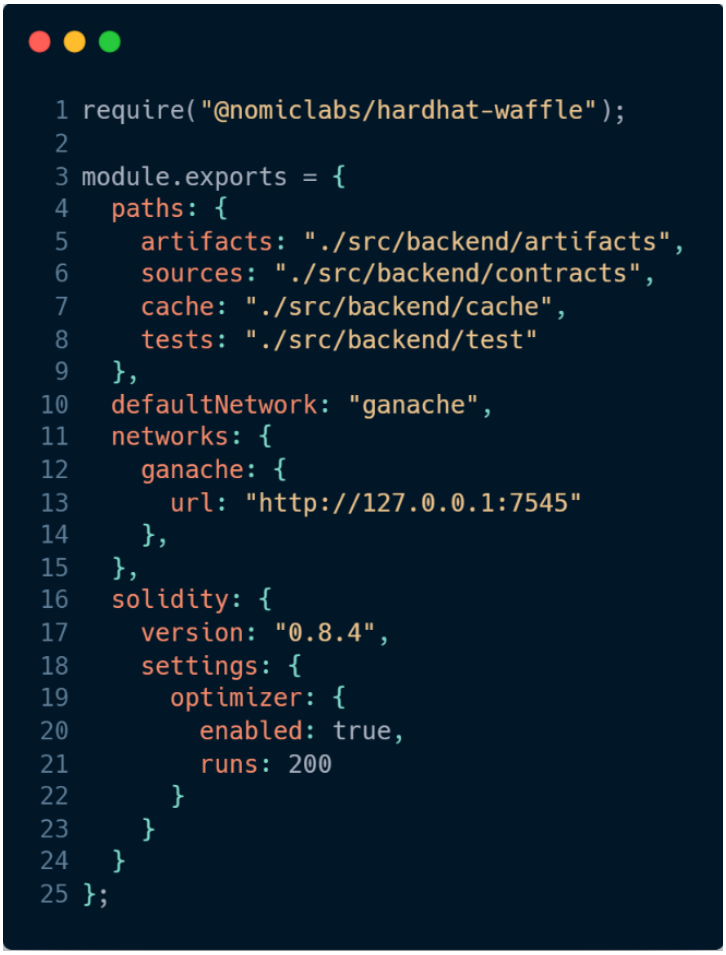
Anexo 4. Despliegue del Contrato

Fuente: Autor

```javascript
const hre = require("hardhat");
async function main() {
  const [deployer] = await hre.ethers.getSigners();

  console.log("Desplegar contratos con la cuenta:", deployer.address);
  console.log("Balance de cuenta:", (await
deployer.getBalance()).toString());
  const NFT = await hre.ethers.getContractFactory("NFT");
  const BlockCar = await hre.ethers.getContractFactory("BlockCar");
  const blockcar = await BlockCar.deploy(1);
  const nft = await NFT.deploy();

  saveFrontendFiles(blockcar, "BlockCar");
  saveFrontendFiles(nft, "NFT");
}

function saveFrontendFiles(contract, name) {
  const fs = require("fs");
  const contractsDir = __dirname + "/../../frontend/contractsData";

  if (!fs.existsSync(contractsDir)) {
    fs.mkdirSync(contractsDir);
  }

  fs.writeFileSync(
    contractsDir + `/${name}-address.json`,
    JSON.stringify({ address: contract.address }, undefined, 2)
  );

  const contractArtifact = hre.artifacts.readArtifactSync(name);

  fs.writeFileSync(
    contractsDir + `/${name}.json`,
    JSON.stringify(contractArtifact, null, 2)
  );
}

main()
  .then(() => process.exit(0))
  .catch(error => {
    console.error(error);
    process.exit(1);
});
```

Anexo 5. Configuración de Hardhat

Fuente: Autor



```
require("@nomiclabs/hardhat-waffle");

module.exports = {
  paths: {
    artifacts: "./src/backend/artifacts",
    sources: "./src/backend/contracts",
    cache: "./src/backend/cache",
    tests: "./src/backend/test"
  },
  defaultNetwork: "ganache",
  networks: {
    ganache: {
      url: "http://127.0.0.1:7545"
    },
  },
  solidity: {
    version: "0.8.4",
    settings: {
      optimizer: {
        enabled: true,
        runs: 200
      }
    }
  }
};
```

Anexo 6. Index.js

Fuente: Autor

```
import React from 'react';
import {createRoot} from 'react-dom/client';
import App from './frontend/components/App';
import 'bootstrap/dist/css/bootstrap.css'
import reportWebVitals from './reportWebVitals';
import './frontend/styles/styles.scss';

const root =
createRoot(document.getElementById('root'));

reportWebVitals();
```

Anexo 7. App.js 1

Fuente: Autor

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Navigation from "./Navbar";
import Home from "./Home";
import Create from "./Create";
import MyListedItems from "./MyListedItems";
import MyPurchases from "./MyPurchases";
import BlockCarAbi from "../contractsData/BlockCar.json";
import BlockCarAddress from "../contractsData/BlockCar-address.json";
import NFTAbi from "../contractsData/NFT.json";
import NFTAddress from "../contractsData/NFT-address.json";
import { useState } from "react";
import { ethers } from "ethers";
import { Spinner } from 'react-bootstrap';

function App() {
  const [account, setAccount] = useState(null);
  const [loading, setLoading] = useState(true);
  const [nft, setNFT] = useState({});
  const [blockcar, setBlockCar] = useState({});

  const web3Handler = async () => {
    const accounts = await window.ethereum.request({
      method: "eth_requestAccounts",
    });

    console.log(accounts);
    setAccount(accounts[0]);
    const provider = new ethers.providers.Web3Provider(window.ethereum);
    const signer = provider.getSigner();

    window.ethereum.on("chainChanged", (chainId) => {
      window.location.reload();
    });

    window.ethereum.on("accountsChanged", async function (accounts) {
      setAccount(accounts[0]);
      await web3Handler();
    });

    loadContracts(signer);
  };

  const loadContracts = async (signer) => {
    const blockcar = new ethers.Contract(BlockCarAddress.address,BlockCarAbi.abi, signer
);  setBlockCar(blockcar);
    const nft = new ethers.Contract(NFTAddress.address, NFTAbi.abi, signer);
    setNFT(nft);
    setLoading(false);
  };
```

Anexo 8. App.js 2

Fuente: Autor

```
return (
  <BrowserRouter>
    <div className="App">
      <>
        <Navigation web3Handler={web3Handler} account={account}/>
      </>
      <div>
        {loading ? (
          <div style={{
            display: 'flex',
            justifyContent: 'center',
            alignItems: 'center',
            minHeight: '80vh'
          }}>
            <Spinner animation='border' style={{ display: 'flex' }}/>
            <p className='mx-3 my-0'>Waiting for Metamask's connection...</p>
          </div>
        ) : (
      <Routes>
        <Route path="/" element={<Home blockcar={blockcar} nft={nft}/>}/>
        <Route path="/create" element={<Create blockcar={blockcar} nft={nft}/>}/>
        <Route path="/my-listed-items" element={<MyListedItems blockcar={blockcar} nft={nft}
        account={account}/>}/>
        <Route path="/my-purchases" element={<MyPurchases blockcar={blockcar} nft={nft} account=
        {account}/>}/>
      </Routes>
        )}
        </div>
    </div>
  </BrowserRouter>
);
}
export default App;
```

Anexo 9. Navbar,js

Fuente: Autor

```
import { Link } from "react-router-dom";
import { Navbar, Nav, Container, Button } from "react-bootstrap";
import car from "./car.png";

const Navigation = ({ web3Handler, account }) => {

  return (
    <Navbar expand="lg" className="navbar-custom ">
      <Container fluid>
        <Navbar.Brand>
          <img src={car} width="100" height="70" className="" alt="" />
            BlockCar
        </Navbar.Brand>
        <Navbar.Toggle aria-controls="responsive-navbar-nav" />
        <Navbar.Collapse  id="navbarScroll">
          <Nav variant="pills" className="me-auto my-6 my-lg-3" style={{ maxHeight: "150px" }}
           navbarScroll>
           <Nav.Item>
              <Nav.Link eventKey="link-1" style={{color: "#FFFFFF"}}  as={Link} to="/" >
                Inicio
              </Nav.Link>
              </Nav.Item>
              <Nav.Item>
              <Nav.Link eventKey="link-2" style={{color: "#FFFFFF"}} as={Link} to="/create">
                Crear
              </Nav.Link>
              </Nav.Item>
              <Nav.Item>
              <Nav.Link eventKey="link-3" style={{color: "#FFFFFF"}} as={Link} to="/my-listed-items">
                Mis Creaciones
              </Nav.Link>
              </Nav.Item>
              <Nav.Item>
              <Nav.Link eventKey="link-4" style={{color: "#FFFFFF"}} as={Link} to="/my-purchases">
                Mis Compras
              </Nav.Link>
              </Nav.Item>
          </Nav>
          <Nav>
            {account ? (
              <Nav.Link
                href={`https://etherscan.io/address/${account}`}
                target="_blank"
                rel="noopener noreferrer"
              >
                <Button variant="outline" >{account}</Button>
              </Nav.Link>
            ) : (
              <Button onClick={web3Handler} variant="outline"  >
                Conectar Wallet
              </Button>
            )}
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
};

export default Navigation;
```

Anexo 10. Home.js 1

Fuente: Autor

```
import { useState, useEffect } from "react";
import { ethers } from "ethers";
import {Row, Col, Card, Button, Modal, Image, Container, Table} from "react-bootstrap";
import Swal from "sweetalert2";

const Home = ({ blockcar, nft }) => {
  const [loading, setLoading] = useState(true);
  const [items, setItems] = useState([]);
  const [showModal, setShowModal] = useState(false);
  const [selectedItem, setSelectedItem] = useState(null);
  const [transactions, setTransactions] = useState([]);
  const loadMarketplaceItem = async () => {
    const itemCount = await blockcar.itemCount();
    let items = [];
    for (let i = 1; i <= itemCount; i++) {
      const item = await blockcar.items(i);
      if (!item.sold) {
        const uri = await nft.tokenURI(item.tokenId);
        const response = await fetch(uri);
        const metadata = await response.json();
        const totalPrice = await blockcar.getTotalPrice(item.itemId);
        items.push({
          totalPrice,
          itemId: item.itemId,
          seller: item.seller,
          owner: item.owner,
          placa: metadata.placa,
          modelo: metadata.modelo,
          marca: metadata.marca,
          origen: metadata.origen,
          color: metadata.color,
          kilometraje: metadata.kilometraje,
          motor: metadata.motor,
          chasis: metadata.chasis,
          image: metadata.image,
        });
      }
    }
    setLoading(false);
    setItems(items);
  };

  const buyMarketItem = async (item) => {
    try{
    await (await blockcar.purchaseItem(item.itemId, { value: item.totalPrice
})).wait();
    loadMarketplaceItem();
    Swal.fire({
      icon: "success",
      title: "¡Vehiculo comprado con exito!",
      width: 800,
      padding: "3em",
      text: `Se ha comprado un nuevo vehiculo`,
      backdrop: `
        left top
        no-repeat
      `,
    });
  }catch (error) {
    console.error('Error al realizar la compra', error);
  }
  };

  const handleShowDetails = async (item) => {
    setShowModal(true);
    setSelectedItem(item);
    try {
      const transactions = await blockcar.getTransactions(item.itemId);
      setTransactions(transactions);
    } catch (error) {
      console.error("Error al obtener transacciones:", error);
    }
  };

  useEffect(() => {
    loadMarketplaceItem();
  }, []);

  if (loading)
    return (
      <main style={{ padding: "1rem 0" }}>
        <h2>Loading...</h2>
      </main>
    );
```

Anexo 11. Home.js 2

Fuente: Autor

```
return (
   <div className="flex justify-center">
     {items.length > 0 ? (
       <div className="px-5 container">
         <Row xs={1} md={2} lg={4} className="g-4 py-5">
           {items.map((item, idx) => (
             <Col key={idx} className="overflow-hidden">
               <Card className="card-custom">
                 <Card.Img className="card-img" variant="top" src={item.image}/>
                 <Card.Body>
                   <Card.Title>{item.marca}</Card.Title>
                   <Card.Text>{item.modelo}</Card.Text>
                   <Card.Text>{item.kilometraje}</Card.Text>
                 </Card.Body>
                 <Card.Footer>
                   <div className="d-grid">
                     <Button variant="primary" onClick={() => handleShowDetails(item)} size="lg">
                       Detalles
                     </Button>
                   </div>
                 </Card.Footer>
               </Card>
             </Col>
           ))}
         </Row>
       </div>
     ) : (
       <main style={{ padding: "1rem 0" }}>
         <h2>No assets</h2>
       </main>
     )}
```

Anexo 12. Home.js 3

Fuente: Autor

```
{selectedItem && (
  <Modal show={showModal} onHide={() => setShowModal(false)} size="lg" aria-
   labelledby="contained-modal-title-vcenter"
   centered backdrop="static" className="modal-custom">
   <Modal.Header closeButton>
     <Modal.Title>Detalles del Vehículo</Modal.Title>
   </Modal.Header>
   <Modal.Body className="grid-example">
     <Container>
       <Row>
         <Col xs={6} md={4}>
           <Image src={selectedItem.image} />
         </Col>
         <Col xs={6} md={8}>
           <table striped bordered hover>
             <tbody>
               <tr>
                 <td>
                   <strong>Placa</strong>
                 </td>
                 <td>{selectedItem.placa}</td>
               </tr>
               <tr>
                 <td>
                   <strong>Modelo</strong>
                 </td>
                 <td>{selectedItem.modelo}</td>
               </tr>
               <tr>
                 <td>
                   <strong>Marca</strong>
                 </td>
                 <td>{selectedItem.marca}</td>
               </tr>
               <tr>
                 <td>
                   <strong>Origen</strong>
                 </td>
                 <td>{selectedItem.origen}</td>
               </tr>
               <tr>
                 <td>
                   <strong>Color</strong>
                 </td>
                 <td>{selectedItem.color}</td>
               </tr>
               <tr>
                 <td>
                   <strong>Kilometraje</strong>
                 </td>
                 <td>{selectedItem.kilometraje}</td>
               </tr>
               <tr>
                 <td>
                   <strong>Motor</strong>
                 </td>
                 <td>{selectedItem.motor}</td>
               </tr>
               <tr>
                 <td>
                   <strong>Chasis</strong>
                 </td>
                 <td>{selectedItem.chasis}</td>
               </tr>
             </tbody>
           </table>
         </Col>
```

Anexo 13. Home.js 4

Fuente: Autor

```
<Col xs={12} md={12}>
                <h4>Historial de Transacciones</h4>
                <p>Creador del Vehiculo: {selectedItem.seller}</p>
                {transactions.length === 0 ? (
                  <p>No hay transacciones disponibles para este vehículo.</p>
                ) : (
                  <Table striped bordered hover>
                    <thead>
                      <tr>
                        <th>Comprador</th>
                        <th>Precio (ETH)</th>
                        <th>Fecha</th>
                      </tr>
                    </thead>
                    <tbody>
                      {transactions.map((transaction, index) => (
                        <tr key={index}>
                          <td>{transaction.buyer}</td>
                          <td>
                            {ethers.utils.formatEther(transaction.price)}
                          </td>
                          <td>
                            {new Date(
                              transaction.timestamp * 1000
                            ).toLocaleString()}
                          </td>
                        </tr>
                      ))}
                    </tbody>
                  </Table>
                )}
              </Col>
            </Row>
          </Container>
        </Modal.Body>
        <Modal.Footer>
          <Button onClick={() => {buyMarketItem(selectedItem); setShowModal(false);}}
          variant="success" size="lg">
            Compra por {ethers.utils.formatEther(selectedItem.totalPrice)} ETH
          </Button>
        </Modal.Footer>
      </Modal>
    )}
  </div>
 );
};

export default Home;
```

Anexo 14. Create.js 1

Fuente: Autor

```
import React, { useState } from "react";
import { ethers } from "ethers";
import { Row, Form, Button } from "react-bootstrap";
import { create as ipfsHttpClient } from "ipfs-http-client";
import { Buffer } from "buffer";
import Swal from "sweetalert2";

const projectId = "2GN3gMoFd2bLjZjPTB07hYdxFWo";
const projectSecret = "234f3933139e7490ee9900f2f23e10d1";
const auth = "Basic " + Buffer.from(projectId + ":" + projectSecret).toString("base64");

const client = ipfsHttpClient({
  host: "ipfs.infura.io",
  port: 5001,
  protocol: "https",
  apiPath: "/api/v0",
  headers: {
    authorization: auth,
  },
});

const Create = ({ blockcar, nft }) => {
  const [image, setImage] = useState("");
  const [price, setPrice] = useState(null);
  const [placa, setPlaca] = useState("");
  const [marca, setMarca] = useState("");
  const [modelo, setModelo] = useState("");
  const [color, setColor] = useState("");
  const [motor, setMotor] = useState("");
  const [chasis, setChasis] = useState("");
  const [kilometraje, setKilo] = useState("");
  const [origen, setOrigen] = useState("");
  const uploadToIPFS = async (event) => {
    event.preventDefault();
    const file = event.target.files[0];
    if (typeof file !== "undefined") {
      try {
        const result = await client.add(file);
        console.log(result);
        setImage(
          `https://nftmarkeplacebrasil.infura-ipfs.io/ipfs/${result.path}`
        );
      } catch (error) {
        console.log("ipfs image upload error: ", error);
      }
    }
  };

  const createNFT = async () => {
    if (!image || !price || !placa || !marca || !modelo || !color || !motor || !chasis ||
    !kilometraje || !origen)
      return;
    try {
      const result = await client.add(JSON.stringify({ image, price, placa, marca, modelo, color,
      motor, chasis, kilometraje, origen})
      );
      mintThenList(result);
    } catch (error) {
      console.log("ipfs uri uload error: ", error);
    }
  };

  const mintThenList = async (result) => {
    const uri = `https://nftmarkeplacebrasil.infura-ipfs.io/ipfs/${result.path}`;
    await (await nft.mint(uri)).wait();
    const id = await nft.tokenCount();
    await await nft.setApprovalForAll(blockcar.address, true);
    const listingPrice = ethers.utils.parseEther(price.toString());
    await (await blockcar.makeItem(nft.address, id, listingPrice)).wait();
    Swal.fire({
        icon: "success",
        title: "¡Vehiculo creado con exito!",
        width: 800,
        padding: "3em",
        text: `Se ha creado un nuevo vehiculo`,
        backdrop: `
          left top
          no-repeat
        `,
      });
  };
```

Anexo 15. Create.js 2

Fuente: Autor

```
  return (
    <div className="container-fluid mt-4">
      <main role="main" className="col-lg-12 mx-auto" style={{ maxWidth: "600px" }}>
        <div className="mx-auto">
          <Row className="g-1">
            <Form.Control type="file" required name="file" onChange={uploadToIPFS}/>
            <Form.Control onChange={(e) => setPlaca(e.target.value)} value={placa} size="lg" required
            type="text" placeholder="Placa"/>
            <Form.Control onChange={(e) => setMarca(e.target.value)} value={marca} size="lg" required
            type="text" placeholder="Marca"/>
            <Form.Control onChange={(e) => setModelo(e.target.value)} value={modelo} size="lg"
            required type="text" placeholder="Modelo"/>
            <Form.Control onChange={(e) => setColor(e.target.value)} value={color} size="lg" required
            type="text" placeholder="Color"/>
            <Form.Control onChange={(e) => setOrigen(e.target.value)} value={origen} size="lg"
            required type="text" placeholder="Origen"/>
            <Form.Control onChange={(e) => setKilo(e.target.value)} value={kilometraje} size="lg"
            required type="text" placeholder="Kilometraje"/>
            <Form.Control onChange={(e) => setMotor(e.target.value)} value={motor} size="lg" required
            type="text" placeholder="Motor"/>
            <Form.Control onChange={(e) => setChasis(e.target.value)} value={chasis} size="lg"
            required type="text" placeholder="Chasis"/>
            <Form.Control onChange={(e) => setPrice(e.target.value)} value={price} size="lg" required
            type="number" placeholder="Precio (ETH)"/>
            <div className="g-grid px-0">
              <Button onClick={createNFT} variant="success" size="lg">
                Registra tu vehiculo!!
              </Button>
            </div>
          </Row>
        </div>
      </main>
    </div>
  );
};

export default Create;
```

Anexo 16. MyListedItems.js

Fuente: Autor

```
import { useState, useEffect } from "react";
import { ethers } from "ethers";
import { Row, Col, Card } from "react-bootstrap";

export default function MyListedItems({ blockcar, nft, account }) {
  const [loading, setLoading] = useState(true);
  const [listedItems, setListedItems] = useState([]);
  const loadListedItems = async () => {
    const itemCount = await blockcar.itemCount();
    let listedItems = [];
    for (let indx = 1; indx <= itemCount; indx++) {
      const i = await blockcar.items(indx);
      if (i.seller.toLowerCase() === account) {
        const uri = await nft.tokenURI(i.tokenId);
        const response = await fetch(uri);
        const metadata = await response.json();
        const totalPrice = await blockcar.getTotalPrice(i.itemId);
        let item = {
          totalPrice,
          price: i.price,
          itemId: i.itemId,
          name: metadata.name,
          description: metadata.description,
          image: metadata.image,
        };
        listedItems.push(item);
      }
    }
    setLoading(false);
    setListedItems(listedItems);
  };

  useEffect(() => {
    loadListedItems();
  }, []);
  if (loading)
    return (
      <main style={{ padding: "1rem 0" }}>
        <h2>Loading...</h2>
      </main>
    );

  return (
    <div className="flex justify-center">
      {listedItems.length > 0 ? (
        <div className="px-5 container">
          <Row xs={1} md={2} lg={4} className="g-4 py-5">
            {listedItems.map((item, idx) => (
              <Col key={idx} className="overflow-hidden">
                <Card className="card-custom">
                  <Card.Img className="card-img" variant="top" src={item.image} />
                  <Card.Footer>
                    {ethers.utils.formatEther(item.totalPrice)} ETH
                  </Card.Footer>
                </Card>
              </Col>
            ))}
          </Row>
        </div>
      ) : (
        <main style={{ padding: "1rem 0" }}>
          <h2> No assets </h2>
        </main>
      )}
    </div>
  );
}
```

Anexo 17. My Purchases.js 1

Fuente: Autor

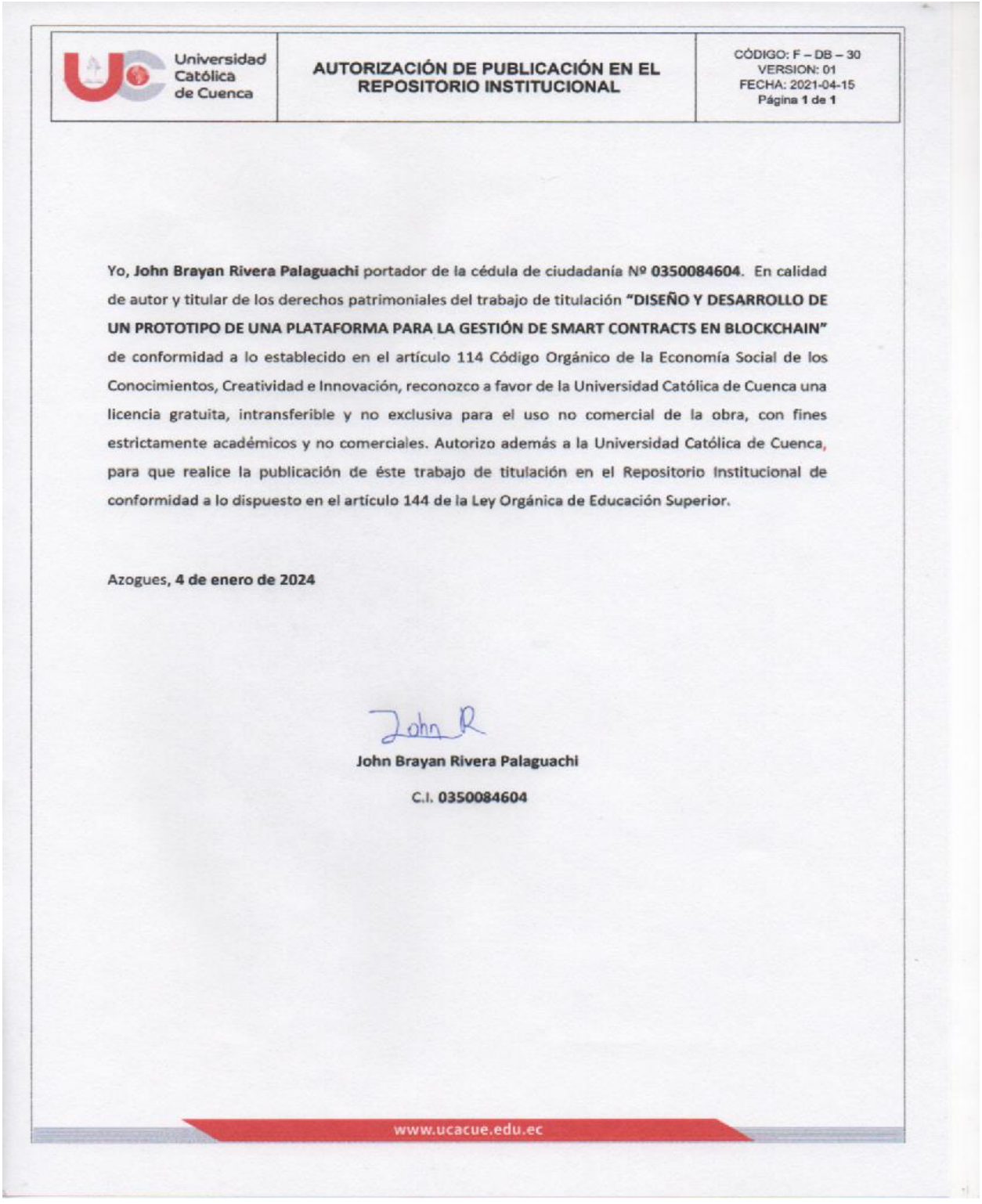
```jsx
import { useState, useEffect } from "react";
import { ethers } from "ethers";
import { Row, Col, Card, Button, Modal, Form } from "react-bootstrap";
import Swal from "sweetalert2";

export default function MyPurchases({ blockcar, nft, account }) {
  const [loading, setLoading] = useState(true);
  const [purchases, setPurchases] = useState([]);
  const [itemIdForResale, setItemIdForResale] = useState(null);
  const [newPrice, setNewPrice] = useState("");
  const [showModal, setShowModal] = useState(false);
  const loadPurchasedItems = async () => {
    const itemCount = await blockcar.itemCount();
    let purchases = [];
    for (let indx = 1; indx <= itemCount; indx++) {
      const i = await blockcar.items(indx);
      if (i.owner.toLowerCase() === account) {
        const uri = await nft.tokenURI(i.tokenId);
        const response = await fetch(uri);
        const metadata = await response.json();
        const totalPrice = await blockcar.getTotalPrice(i.itemId);
        let purchasedItem = {
          totalPrice,
          price: i.price,
          itemId: i.itemId,
          name: metadata.name,
          description: metadata.description,
          image: metadata.image,
        };
        purchases.push(purchasedItem);
      }
    }
    setLoading(false);
    setPurchases(purchases);
  };

  const resellItem = async () => {
    if (itemIdForResale && newPrice) {
      try {
        const item = await blockcar.items(itemIdForResale);
        if (item.sold) {
        }
        await await nft.setApprovalForAll(blockcar.address, true);
        const newPriceInWei = ethers.utils.parseEther(newPrice);
        await blockcar.resellItem(itemIdForResale, newPriceInWei);
        loadPurchasedItems();
        Swal.fire({
          icon: "success",
          title: "¡Vehiculo revendido con exito!",
          width: 800,
          padding: "3em",
          text: `Se ha revendido su vehiculo`,
          backdrop: `
            left top
            no-repeat
          `,
        });
        setShowModal(false);
      } catch (error) {
        console.error("Error al realizar la reventa", error);
      }
    }
  };

  useEffect(() => {
    loadPurchasedItems();
  }, []);
  if (loading)
    return (
      <main style={{ padding: "1rem 0" }}>
        <h2>Loading...</h2>
      </main>
    );
```

Anexo 18. MyPurchases.js 2

Fuente: Autor

```
return (
  <div className="flex justify-center">
    {purchases.length > 0 ? (
      <div className="px-5 container">
        <Row xs={1} md={2} lg={4} className="g-4 py-5">
          {purchases.map((item, idx) => (
            <Col key={idx} className="overflow-hidden">
              <Card className="card-custom">
                <Card.Img className="card-img" variant="top" src={item.image} />
                <Card.Body>
                  <Card.Text>
                    {ethers.utils.formatEther(item.totalPrice)} ETH
                  </Card.Text>
                </Card.Body>
                <Card.Footer>
                  <Button onClick={() => {setItemIdForResale(item.itemId); setShowModal(true);}}
                  variant="primary" className="lg">
                    Revender
                  </Button>
                </Card.Footer>
              </Card>
            </Col>
          ))}
        </Row>
      </div>
    ) : (
      <main style={{ padding: "1rem 0" }}>
        <h2> No purchases </h2>
      </main>
    )}
    <Modal show={showModal} onHide={() => setShowModal(false)} className="modal-custom">
      <Modal.Header closeButton>
        <Modal.Title>Resell NFT</Modal.Title>
      </Modal.Header>
      <Modal.Body>
        <Form.Group controlId="newPrice">
          <Form.Label>Nuevo Precio: (ETH)</Form.Label>
          <Form.Control type="text" placeholder="Nuevo Precio " value={newPrice} onChange={(e) =>
          setNewPrice(e.target.value)}/>
        </Form.Group>
      </Modal.Body>
      <Modal.Footer>
        <Button variant="primary" onClick={() => {setShowModal(false); resellItem();}}>
          Revender
        </Button>
      </Modal.Footer>
    </Modal>
  </div>
);
}
```

AUTORIZACION DE PUBLICACION EN EL REPOSITORIO INSTITUCIONAL



Universidad Católica de Cuenca	AUTORIZACIÓN DE PUBLICACIÓN EN EL REPOSITORIO INSTITUCIONAL	CÓDIGO: F – DB – 30 VERSION: 01 FECHA: 2021-04-15 Página 1 de 1

Yo, **John Brayan Rivera Palaguachi** portador de la cédula de ciudadanía Nº **0350084604**. En calidad de autor y titular de los derechos patrimoniales del trabajo de titulación **"DISEÑO Y DESARROLLO DE UN PROTOTIPO DE UNA PLATAFORMA PARA LA GESTIÓN DE SMART CONTRACTS EN BLOCKCHAIN"** de conformidad a lo establecido en el artículo 114 Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación, reconozco a favor de la Universidad Católica de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos y no comerciales. Autorizo además a la Universidad Católica de Cuenca, para que realice la publicación de éste trabajo de titulación en el Repositorio Institucional de conformidad a lo dispuesto en el artículo 144 de la Ley Orgánica de Educación Superior.

Azogues, **4 de enero de 2024**

John Brayan Rivera Palaguachi

C.I. 0350084604

www.ucacue.edu.ec