PLATAFORMA BLOCKCHAIN PARA REGISTROS DE MANTENIMIENTO DE VEHÍCULOS

BLOCKCHAIN-BASED PLATFORM FOR CAR MAINTENANCE RECORDS

TRABAJO FIN DE GRADO
CURSO 2022-2023

AUTOR
HAROON SAMMARAIE SALIH
JAVIER GARCÍA SUÁREZ

DIRECTOR
JESÚS CORREAS FERNÁNDEZ

GRADO EN INGENIERÍA SOFTWARE
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

AGRADECIMIENTOS

A nuestras familias por su apoyo y paciencia durante este camino. Han sido una fuente de fortaleza y estamos realmente agradecidos por su presencia en nuestras vidas.

A Jesús Correas por su valiosa orientación y apoyo durante el desarrollo de este proyecto.

RESUMEN

Este proyecto se centra en el desarrollo de una plataforma para gestionar la información de vehículos en general, incluyendo información de reparación y registros de incidentes, utilizando la tecnología Blockchain. La plataforma funciona mediante la validación de talleres y el almacenamiento de información clave en una base de datos descentralizada junto con datos on-chain para garantizar la transparencia y prevenir el fraude. Al aprovechar la seguridad y la transparencia de la tecnología Blockchain, la plataforma mejora la validez de la información de vehículos, asegurando que los datos no puedan ser manipulados o alterados. Esto proporciona un registro de vehículos confiable y completo para todas las partes involucradas, ya sean propietarios o empresas.

Además de sus características de seguridad, la plataforma también prioriza la experiencia del usuario a través de una interfaz de usuario (UI) intuitiva. La plataforma ha sido diseñada para ser fácil de usar y no requiere conocimientos previos de tecnología Blockchain, lo que la hace accesible para una amplia gama de usuarios.

La plataforma utiliza técnicas criptográficas para proteger los datos, lo que la hace difícil de comprometer. En general, esta plataforma representa un paso significativo hacia adelante en la búsqueda de una gestión más segura, transparente y eficiente de la información de vehículos. Proporciona una solución confiable y fácil de utilizar para propietarios y empresas de automóviles, asegurando que la información de vehículos se pueda administrar de una manera segura y transparente.

Palabras clave: Sistemas descentralizados, Blockchain, Smart Contracts, Seguridad de datos, Técnicas criptográficas, Transparencia, Fraude , Segura

ABSTRACT

This project focuses on the development of a platform to manage vehicle information in general, including repair information and incident records, using Blockchain technology. The platform works by validating workshops and storing key information in a decentralized database along with on-chain data to ensure transparency and prevent fraud. By leveraging the security and transparency of Blockchain technology, the platform improves the validity of vehicle information, ensuring that the data cannot be tampered with or altered. This provides a reliable and complete vehicle record for all parties involved, whether owners or businesses.

In addition to its security features, the platform also prioritizes the user experience through an intuitive user interface (UI). The platform has been designed to be easy to use and requires no prior knowledge of Blockchain technology, making it accessible to a wide range of users.

The platform uses cryptographic techniques to protect data, making it difficult to compromise. Overall, this platform represents a significant step forward in the search for a more secure, transparent and efficient management of vehicle information. It provides a reliable and easy-to-use solution for car owners and companies, ensuring that vehicle information can be managed in a secure and transparent manner.

Keywords: Platform, Vehicle information, Blockchain technology, Decentralized database, Transparency, Fraud, Secure

Índice

Capítulo 1 - Introducción **13**
1.1. Motivación 13
1.2. Objetivos 14
1.3. Plan de trabajo 15
1.3.1 Estructura del grupo de trabajo y roles de cada miembro 15
1.3.2 Planificación temporal del proyecto 16
1.3.3 Control de tiempo y reuniones del equipo 16
1.4. Estructura de la memoria 18
Capítulo 2 - Introduction **21**
2.1. Motivation 21
2.2. Objectives 21
2.3. Workplan 22
2.3.1 Team Structure and Roles of Each Member 23
2.3.2 Project Timeline 23
2.3.3 Time Control and Team Meetings 24
2.4. Memory structure 25
Capítulo 3 -Estado del arte **27**
3.1. Introducción a la tecnología blockchain 27
3.1.1. Blockchain 27
3.1.2. Tipos de blockchain 28
3.1.2.1. Pública 28
3.1.2.2. Privada 28
3.1.3. Características clave de la tecnología blockchain 29
3.1.3.1. Descentralización y consenso distribuido 29
3.1.3.2. Inmutabilidad de los datos 29
3.1.3.3. Transparencia y auditoría de las transacciones 30
3.1.3.4. Seguridad y criptografía en blockchain 30
3.1.4. Ethereum 31
3.1.5. Contratos inteligentes 31
3.1.5.1 Definición y características de los contratos inteligentes 31
3.1.5.1 Lenguajes de programación para la creación de contratos inteligentes 32
3.1.6. Solidity 33
3.1.7. Tokens 35
3.1.8. DApps 36
3.2. Tecnologías centralizadas en la gestión de información de vehículos 36
3.2.1. Libro electrónico de mantenimiento - DGT 37
3.2.1. Carfax 41
3.2.1. Autocheck 43
3.3. Tecnologías descentralizadas relevantes 44
3.3.1. Tecnología Blockchain para mantenimiento e ingeniería de aeronaves 45
Capítulo 4 - Tecnologías utilizadas **47**

4.1. Desarrollo Front-end 47
4.1.1. HTML 47
4.1.2. CSS 47
4.1.3. JavaScript 47
4.1.4. EJS 48
4.1.5. Bootstrap 48
4.2. Desarrollo Back-end 49
4.2.1. Node.js 49
4.2.1.1. Librerías utilizadas 49
4.2.2. Express 51
4.2.3. SQL 51
4.2.4. MySQL 52
4.2.5. XAMPP 52
4.2.6. Solidity 53
4.2.7. Remix 53
4.2.8. Geth 53
Capítulo 5 - Arquitectura y diseño del sistema 55
5.1. Arquitectura MVC 56
5.2. Datos almacenados On-chain y Off-chain 58
5.2.1. Estructura de datos On-chain 59
5.2.1.1. Variables 60
5.2.1.2. Funciones 61
5.2.1.3. Mofidicadores 62
5.2.2. Datos almacenados Off-chain 63
Capítulo 6 - Implementación de la plataforma 67
6.1. Estructura del código 67
6.1.1. Carpeta Config 68
6.1.2. Controllers 68
6.1.3. Ethereum 70
6.1.4. Routes 71
6.1.5. Views / Public 72
6.2. Configuración Inicial 72
6.2.1. Configuración de un nodo local de Ethereum y simulación de una red privada de blockchain 73
6.3. Conexión con la base de datos 76
6.4 Creación de nuestro Smart Contract 76
6.5 Implementación de la Interfaz de Usuario 76
6.6 Implementación de los flujos de trabajo 77
6.6.1 Flujo de trabajo del Administrador 80
6.6.2 Flujo de trabajo del Administrador de un taller 85
6.6.3 Flujo de trabajo para usuarios 88
Capítulo 7 - Conclusiones y trabajo futuro 101
7.1 Viabilidad del sistema en un contexto real 101
7.1.1 Introducción a la sociedad y colaboración con la DGT 101

7.1.2 Acuerdos con talleres autorizados e incorporación de fábricas, concesionarios y seguros 102
7.1.3 Posibles mejoras o ampliaciones de nuestra aplicación 104
7.2 Conclusiones 105
Capítulo 8 - Conclusions and Future Work 107
8.1 System viability in the real world 107
8.1.1 Introduction to society and collaboration with the DGT 107
8.1.2 Agreements with authorized workshops and incorporation of factories, dealers and insurance 108
8.1.3 Possible improvements or extensions of our application 110
8.2 Conclusions 111
Capítulo 9 - Contribuciones personales 113
9.1. Haroon Sammaraie Salih 113
9.2. Javier García Suárez 114
Capítulo 10 - Referencias 116

Índice de figuras

Figura 3.1 Pantalla de acceso a la autenticación de usuario en la DGT 39

Figura 3.2 Ejemplo de informe reducido 40

Figura 3.3 Ejemplo de informe completo 41

Figura 3.4 Ejemplo de informe Carfax 42

Figura 3.5 Ejemplo de informe AutoCheck 46

Figura 3.6 Diagrama de la aplicación propuesta a desarrollar 46

Figura 5.1 Arquitectura de la aplicación 55

Figura 5.2 Diagrama MVC 56

Figura 6.1 Paso 1 de instalación de GETH. 73

Figura 6.2 Paso 2 de instalación de GETH. 74

Figura 6.3 Paso 4 de instalación de GETH. 75

Figura 6.4 Vista de la sección de registro. 78

Figura 6.5 Vista de la sección de login. 79

Figura 6.6 Diagrama de flujo de trabajo de la creación de talleres 81

Figura 6.7 Vista de la sección de alta de usuarios 82

Figura 6.8 Vista de la sección de alta de talleres 82

Figura 6.9 Vista de la sección de alta de vehículos 83

Figura 6.10 Diagrama de flujo de trabajo de alta de servicio por parte de taller 85

Figura 6.11 Vista de panel de administración de taller 86

Figura 6.12 Diagrama de flujo de trabajo de los usuarios de la aplicación. 88

Figura 6.13 Pantalla de acceso al informe de vehículo. 89

Figura 6.14 Informe de vehículo generado por la aplicación. 91

Figura 6.15 Vista del dashboard de un usuario. 92

Figura 6.16 Vista del dashboard de un usuario - Sección de vehículos 93

Figura 6.17 Vista del dashboard de un usuario - Reparaciones de un vehículo. 94

Figura 6 18 Vista del dashboard de un usuario - Sección de autorizaciones 95

Figura 6 19 Vista del dashboard de un usuario - Sección de autorizaciones pendientes de aprobación. 96

Figura 6 20 Vista del dashboard de un usuario - Sección de vehículos en los que ya ha sido autorizado el usuario. 97

Figura 6 21 Vista del dashboard de un usuario - Sección para pedir autorización a un vehículo concreto y vista de los que ya han sido autorizados 98

Capítulo 1 - Introducción

1.1. Motivación

En la actualidad, la industria automotriz es una parte esencial de nuestra vida diaria, y dependemos de ella para desplazarnos y transportar bienes. Sin embargo, la gestión de la información de vehículos en general, incluyendo la información de reparación y de incidentes, sigue siendo un proceso complejo y opaco, lo que a menudo conduce a la desconfianza y la ineficiencia. Nuestra motivación para este proyecto es llevar transparencia y confiabilidad a la gestión de la información de vehículos.

Creemos que la tecnología Blockchain ofrece una solución única a este problema. Al aprovechar la seguridad y la transparencia de la tecnología Blockchain, podemos crear una plataforma descentralizada que garantice la integridad de la información de vehículos, evitando cualquier intento de manipulación o alteración. Esto puede mejorar significativamente la fiabilidad y eficiencia de la industria automotriz, beneficiando a todas las partes involucradas, desde los propietarios hasta las empresas.

Además, nuestra plataforma permite a los usuarios buscar información de vehículos a través de su matrícula, ofreciendo detalles técnicos, historial de propietarios, incidencias, reparaciones, kilometraje y detalles del propietario actual. Este nivel de transparencia y accesibilidad no tiene precedentes en la industria automotriz y creemos que puede tener un impacto significativo en la confianza y eficiencia del sector.

Reconocemos que no todo el mundo está familiarizado con la tecnología Blockchain, lo que a veces puede ser una barrera para su adopción. Por lo tanto, hemos diseñado nuestra plataforma para ser fácil de usar y accesible para una amplia gama de usuarios, independientemente de su conocimiento técnico.

Creemos que nuestro proyecto representa un paso significativo hacia adelante en la búsqueda de una gestión más segura, transparente y eficiente de la información de vehículos.

1.2. Objetivos

Dado el carácter innovador de nuestro proyecto, que combina la tecnología Blockchain con la gestión de información de vehículos, nuestros objetivos son dobles. El primer conjunto de objetivos se centra en los aspectos técnicos del proyecto, mientras que el segundo conjunto de objetivos se centra en sus implicaciones sociales.

Objetivos técnicos:

- Desarrollar una plataforma segura y transparente para la gestión de información de vehículos utilizando tecnología Blockchain.
- Diseñar e implementar workflows eficientes para los diferentes roles de usuario en la plataforma, incluyendo administradores, talleres y usuarios finales.
- Aprovechar las ventajas de la tecnología Blockchain, incluyendo la descentralización y la resistencia a la manipulación.
- Proporcionar una interfaz intuitiva y fácil de usar para una amplia gama de usuarios, independientemente de su conocimiento técnico.

Objetivos sociales:

- Mejorar la confianza y transparencia en la gestión de información de vehículos, lo que puede beneficiar a todas las partes involucradas, incluyendo propietarios, talleres y empresas.

- Promover una cultura de transparencia y confiabilidad en la gestión de información de vehículos, lo que puede mejorar la calidad del servicio en la industria automotriz.
- Ofrecer una plataforma segura y confiable para la gestión de información de vehículos, lo que puede ayudar a prevenir la manipulación de información y los intentos de fraude.
- Garantizar la privacidad y seguridad de la información de los usuarios, incluyendo información personal y de vehículos.

1.3. Plan de trabajo

En este capítulo, se describe la metodología de trabajo empleada en el proyecto, incluyendo la estructura del grupo de trabajo, los roles de cada miembro, la planificación temporal del proyecto y el control del tiempo y las reuniones de equipo.

1.3.1 Estructura del grupo de trabajo y roles de cada miembro

El equipo de trabajo está compuesto por dos miembros que colaboran estrechamente en todas las etapas del proyecto. Ambos miembros comparten responsabilidades y roles, lo que permite una distribución equitativa del trabajo y la adaptación a las necesidades del proyecto en cada momento. Algunos de los roles compartidos incluyen:

- Investigación y diseño: Ambos miembros realizan investigaciones para identificar las mejores prácticas y tecnologías en la gestión de información de vehículos y diseñar la solución basada en blockchain.
- Desarrollo de software: Cada miembro del equipo participa en el desarrollo y revisión del código del proyecto, utilizando herramientas como GitHub para gestionar el código y los commits.

- Gestión del proyecto y documentación: Los miembros del equipo colaboran en la planificación, seguimiento y documentación del proyecto, utilizando herramientas como Notion.so para gestionar las tareas, listas de cosas por hacer, prioridades, plazos, etc.

1.3.2 Planificación temporal del proyecto

La planificación temporal del proyecto se estructura en varias etapas, con objetivos y plazos específicos para cada una de ellas. A continuación, se presenta un esquema general de la planificación:

- **Fase de investigación y diseño:** En esta etapa, el equipo realiza investigaciones sobre la tecnología blockchain, la gestión de información de vehículos y las soluciones existentes en el mercado. Al finalizar la investigación, se diseña la arquitectura general de la plataforma.
- **Fase de desarrollo de software:** Durante esta etapa, el equipo implementa la solución diseñada, desarrollando y probando el código de la plataforma basada en blockchain. Se utilizan metodologías ágiles para garantizar la adaptabilidad y la calidad del proyecto.
- **Fase de pruebas y ajustes:** En esta etapa, el equipo realiza pruebas exhaustivas de la plataforma y realiza ajustes y mejoras según sea necesario.
- **Fase final:** Finalmente, la plataforma se implementa y entrega, junto con la documentación y los recursos de soporte necesarios.

1.3.3 Control de tiempo y reuniones del equipo

El control del tiempo y la comunicación entre los miembros del equipo son aspectos clave para garantizar el éxito del proyecto. Para ello, se utilizan las siguientes estrategias:

- **Reuniones regulares:** El equipo lleva a cabo reuniones semanales para revisar el progreso del proyecto, discutir problemas, establecer prioridades y asignar tareas. Estas reuniones también sirven como oportunidad para evaluar y ajustar la planificación temporal del proyecto según sea necesario.
- **Herramientas de gestión del tiempo:** El equipo utiliza herramientas como Notion.so para controlar el tiempo dedicado a cada tarea y garantizar el cumplimiento de los plazos establecidos.
- **Comunicación continua:** Los miembros del equipo mantienen una comunicación constante a través de canales de mensajería, lo que permite abordar rápidamente cualquier problema o pregunta que surja durante el desarrollo del proyecto. Además, se fomenta la colaboración y el intercambio de ideas para asegurar una visión compartida del proyecto y sus objetivos.
- **Reuniones de revisión y retrospectiva:** Al finalizar cada etapa del proyecto, el equipo realiza reuniones de revisión para evaluar el trabajo realizado y las lecciones aprendidas. Estas reuniones de retrospectiva permiten identificar áreas de mejora y ajustar la metodología de trabajo para las siguientes fases del proyecto.

1.4. Estructura de la memoria

Capítulo 1/2: Introducción

- Explicación del proyecto y su importancia en la industria automotriz
- Estructura del grupo de trabajo y roles de cada miembro.
- Planificación temporal del proyecto.
- Control de tiempo y reuniones del equipo.

Capítulo 3: Estado del arte

- Conceptos básicos de la tecnología Blockchain y su uso en la gestión de información de vehículos.
- Descripción de las tecnologías actuales en la gestión de información de vehículos.
- Identificación de las limitaciones y problemas actuales en la gestión de información de vehículos.
- Explicación de cómo nuestra plataforma mejorará la gestión de información de vehículos.

Capítulo 4: Tecnologías utilizadas en la plataforma

- Descripción de las tecnologías utilizadas en nuestra plataforma, incluyendo Blockchain y criptografía.

Capítulo 5: Arquitectura de la plataforma

- Diseño y arquitectura de la plataforma, incluyendo la estructura de la base de datos.

Capítulo 6: Implementación de la plataforma

- Detalles sobre la implementación de la plataforma, incluyendo los pasos para la integración de Blockchain y otras tecnologías.

Capítulo 7/8: Conclusiones

- Resumen de los resultados obtenidos durante el proyecto y su importancia en la gestión de información de vehículos.
- Discusión sobre posibles mejoras y futuras líneas de investigación.

Capítulo 9: Contribuciones personales

- Las tareas y responsabilidades de cada miembro.

Capítulo 10: Referencias

- Referencias utilizadas a lo largo del proyecto

Capítulo 2 - Introduction

2.1. Motivation

Currently, the automotive industry is an essential part of our daily lives, and we depend on it for transportation and the movement of goods. However, managing vehicle information in general, including repair and incident information, remains a complex and opaque process, often leading to distrust and inefficiency. Our motivation for this project is to bring transparency and reliability to vehicle information management.

We believe that Blockchain technology offers a unique solution to this problem. By leveraging the security and transparency of Blockchain technology, we can create a decentralized platform that ensures the integrity of vehicle information, preventing any attempts at manipulation or alteration. This can significantly improve the reliability and efficiency of the automotive industry, benefiting all parties involved, from owners to businesses.

Additionally, we recognize that not everyone is familiar with Blockchain technology, which can sometimes be a barrier to its adoption. Therefore, we have designed our platform to be easy to use and accessible to a wide range of users, regardless of their technical knowledge.

We believe that our project represents a significant step forward in the pursuit of more secure, transparent, and efficient management of vehicle information. It has the potential to revolutionize the automotive industry and bring benefits to all parties involved.

2.2. Objectives

Based on the innovative nature of our project, which combines Blockchain technology with vehicle information management, our objectives are twofold. The first set of objectives focuses on the technical aspects of the project, while the second set of objectives focuses on the social implications of the project.

Technical objectives:

- Develop a secure and transparent platform for vehicle information management using Blockchain technology.
- Leverage the advantages of Blockchain technology, including decentralization and resistance to manipulation.
- Provide an intuitive and user-friendly interface for a wide range of users, regardless of their technical knowledge.

Social objectives:

- Improve trust and transparency in vehicle information management, which can benefit all parties involved, including owners, workshops, and businesses.
- Promote a culture of transparency and reliability in vehicle information management, which can improve the quality of service in the automotive industry.
- Offer a secure and reliable platform for vehicle information management, which can help prevent information manipulation and fraud attempts.
- Ensure the privacy and security of users' information, including personal and vehicle information.

2.3. Workplan

This chapter outlines the work methodology employed in the project, including the team structure, the roles of each member, the project timeline, and time management and team meetings.

2.3.1 Team Structure and Roles of Each Member

The team comprises two members who closely collaborate at all stages of the project. Both members share responsibilities and roles, allowing for an equal distribution of work and adaptation to the project's needs at any given moment. Some of the shared roles include:

Research and Design: Both members carry out research to identify best practices and technologies in vehicle information management and design the blockchain-based solution.

Software Development: Each team member participates in the development and review of the project's code, using tools like GitHub to manage the code and commits.

Project Management and Documentation: The team members collaborate in planning, monitoring, and documenting the project, using tools like Notion.so to manage tasks, to-do lists, priorities, deadlines, etc.

2.3.2 Project Timeline

The project timeline is structured in various stages, with specific objectives and deadlines for each. The general layout of the plan is as follows:

Research and Design Phase: In this stage, the team conducts research on blockchain technology, vehicle information management, and existing solutions in

the market. At the end of the research, the general architecture of the platform is designed.

Software Development Phase: During this stage, the team implements the designed solution, developing and testing the code for the blockchain-based platform. Agile methodologies are used to ensure the project's adaptability and quality.

Testing and Adjustments Phase: In this stage, the team carries out comprehensive tests on the platform and makes adjustments and improvements as necessary.

Final Phase: Finally, the platform is implemented and delivered, along with the necessary documentation and support resources.

2.3.3 Time Control and Team Meetings

Time control and communication among team members are key aspects to ensure the project's success. For this, the following strategies are employed:

Regular Meetings: The team holds weekly meetings to review the project's progress, discuss issues, set priorities, and assign tasks. These meetings also serve as an opportunity to evaluate and adjust the project timeline as needed.

Time Management Tools: The team uses tools like Notion.so to control the time dedicated to each task and ensure adherence to set deadlines.

Continuous Communication: Team members maintain constant communication through messaging channels, which allows for quickly addressing any issues or questions that arise during the project's development. Additionally, collaboration and idea sharing are encouraged to ensure a shared vision of the project and its goals.

Review and Retrospective Meetings: At the end of each project stage, the team holds review meetings to evaluate the work done and the lessons learned. These retrospective meetings allow for identifying areas of improvement and adjusting the work methodology for the following project phases.

2.4. Memory structure

Chapter 1/2: Introduction

- Explanation of the project and its significance in the automotive industry
- Structure of the workgroup and roles of each member
- Project timeline planning
- Time management and team meetings

Chapter 3: State of the Art

- Basic concepts of Blockchain technology and its use in vehicle information management
- Description of current technologies in vehicle information management
- Identification of limitations and current issues in vehicle information management
- Explanation of how our platform will improve vehicle information management

Chapter 4: Technologies used in the platform

- Description of the technologies used in our platform, including Blockchain and cryptography

Chapter 5: Platform Architecture

- Design and architecture of the platform, including the database structure

Chapter 6: Platform Implementation

- Details about the platform's implementation, including steps for integrating Blockchain and other technologies

Chapter 7/8: Conclusions

- Summary of the results obtained during the project and their significance in vehicle information management
- Discussion about possible improvements and future lines of research

Chapter 9: Personal Contributions

- The tasks and responsibilities of each member

Chapter 10: References

- References used throughout the project

Capítulo 3 -Estado del arte

3.1. Introducción a la tecnología blockchain

3.1.1. Blockchain

Blockchain, o cadena de bloques, es una tecnología revolucionaria que permite el almacenamiento y la transferencia de información de manera descentralizada, segura y transparente. La información en una cadena de bloques se almacena en bloques de datos que están enlazados y asegurados mediante criptografía.

Una de las características clave de la tecnología blockchain es su naturaleza descentralizada. A diferencia de los sistemas de base de datos tradicionales que son controlados por una entidad central, una cadena de bloques se distribuye en una red de nodos, cada uno de los cuales tiene una copia completa de la cadena de bloques. Esto significa que no hay un punto único de fallo y la información en la cadena de bloques es resistente a la censura y a la manipulación.

La tecnología blockchain utiliza técnicas criptográficas para asegurar la información. Cada bloque en la cadena contiene un hash criptográfico del bloque anterior, un sello de tiempo y los datos almacenados en el bloque, organizados en forma de transacciones. Esto hace que la información en la cadena de bloques sea prácticamente inalterable, ya que cualquier intento de modificar la información en un bloque requeriría cambiar la información en todos los bloques siguientes en la cadena. Dependiendo del mecanismo de consenso utilizado por los nodos de la red de blockchain, la manipulación de los datos almacenados en la cadena de bloques es computacionalmente inviable, o infactible por la propia estructura del mecanismo de consenso.

Aunque la información en una cadena de bloques es segura, también es transparente. Cada transacción en la cadena de bloques es visible para todos los participantes de la red, lo que garantiza la trazabilidad y la transparencia de las transacciones. En el contexto de nuestra plataforma, esto significa que la información de los vehículos, incluyendo el historial de propietarios, las reparaciones y los incidentes, puede ser rastreada y verificada de manera transparente.

3.1.2. Tipos de blockchain

La tecnología blockchain se puede clasificar en diferentes tipos en función de su accesibilidad y permisividad. Los dos tipos principales son las blockchains públicas y las blockchains privadas *(1. Blockchain privada)*. A continuación, vamos a explorar cada uno de ellos:

3.1.2.1. Pública

Una blockchain pública, como su nombre lo indica, es accesible para cualquier persona que desee participar en ella. En este tipo de blockchain, cualquiera puede unirse a la red, validar transacciones y contribuir al proceso de consenso. No se requiere permiso para acceder a la blockchain pública y cualquier individuo puede ver todas las transacciones registradas en la cadena de bloques.

Un ejemplo popular de blockchain pública es la red de Bitcoin. Cualquier persona puede unirse a la red Bitcoin, realizar transacciones y participar en la minería para asegurar la red. La transparencia es una característica clave de las blockchains públicas, ya que todas las transacciones son visibles y verificables por cualquier persona.

3.1.2.2. Privada

A diferencia de las blockchains públicas, las blockchains privadas restringen el acceso y la participación a un grupo específico de individuos o entidades. En una blockchain privada, solo las partes autorizadas pueden unirse a la red y participar en la validación de transacciones. Por lo general, se requiere un permiso para acceder a una blockchain privada.

Las blockchains privadas son utilizadas por empresas, organizaciones y consorcios que desean mantener un mayor control sobre su red y datos. Al restringir el acceso, pueden establecer niveles de privacidad y confidencialidad más altos en comparación con las blockchains públicas. Además, las blockchains privadas pueden ofrecer un mayor rendimiento y velocidad de transacción, ya que no están limitadas por la necesidad de alcanzar un consenso abierto y global.

3.1.3. Características clave de la tecnología blockchain

3.1.3.1. Descentralización y consenso distribuido

La tecnología blockchain se distingue por su característica de descentralización *(2. Seguridad en el blockchain)*, que elimina la dependencia de una autoridad centralizada. En lugar de confiar en un intermediario, como un banco o una entidad gubernamental, la información en una blockchain se distribuye en una red de nodos interconectados.

Cada nodo contiene una copia completa y actualizada de la cadena de bloques, lo que garantiza la redundancia y la disponibilidad de los datos. La descentralización permite que la toma de decisiones y la validación de las transacciones sean llevadas a cabo por consenso distribuido, donde múltiples nodos en la red deben ponerse de acuerdo para confirmar y registrar una transacción en la cadena de bloques.

Esto brinda mayor seguridad y confianza al evitar puntos únicos de fallo y posibles manipulaciones.

3.1.3.2. Inmutabilidad de los datos

La inmutabilidad es una propiedad fundamental de la tecnología blockchain que garantiza la integridad de los datos almacenados en la cadena de bloques. Una vez que una transacción ha sido registrada y confirmada en un bloque, los datos asociados a esa transacción no pueden ser modificados ni eliminados sin afectar la integridad de toda la cadena.

Esta característica se logra a través del uso de funciones criptográficas que generan un hash único para cada bloque, el cual depende del contenido del bloque anterior. Cualquier intento de alterar un bloque requeriría cambiar su hash, lo que a su vez afectaría a todos los bloques siguientes.

Esta estructura encadenada hace que sea prácticamente imposible modificar los datos sin ser detectado, proporcionando un alto nivel de confianza y seguridad en la integridad de la información almacenada en la blockchain.

3.1.3.3. Transparencia y auditoría de las transacciones

La tecnología blockchain brinda transparencia y posibilita la auditoría de las transacciones realizadas en la red. Cada transacción registrada en la cadena de bloques es accesible para todos los participantes de la red, lo que permite una visibilidad completa de todas las transacciones registradas.

Esto promueve la confianza y la rendición de cuentas al proporcionar a los usuarios la capacidad de verificar la validez y la exactitud de las transacciones. Además, la cadena de bloques facilita la realización de auditorías, ya que el historial transparente de las transacciones permite rastrear el origen y el destino de los activos, garantizando así la trazabilidad y la integridad de los registros.

La transparencia y la auditoría de las transacciones en blockchain reducen la dependencia de terceros de confianza y fomentan la verificación independiente de la información.

3.1.3.4. Seguridad y criptografía en blockchain

La seguridad desempeña un papel fundamental en la tecnología blockchain, y se basa en la utilización de técnicas criptográficas sólidas. Cada transacción en la cadena de bloques es firmada digitalmente con claves criptográficas, lo que garantiza su autenticidad e integridad.

Además, la cadena de bloques emplea algoritmos criptográficos para enlazar los bloques de manera segura, protegiendo así la integridad de los datos almacenados en ellos.

La descentralización de la red y la distribución de la información entre múltiples nodos también contribuyen a reforzar la seguridad, ya que no hay un punto centralizado que pueda ser atacado o comprometido.

En conjunto, la combinación de seguridad y criptografía en blockchain proporciona una capa adicional de protección contra manipulaciones y ataques maliciosos, asegurando la confidencialidad y la integridad de los datos en la cadena de bloques.

3.1.4. Ethereum

Ethereum es una plataforma descentralizada y una criptomoneda que permite a los desarrolladores crear y ejecutar aplicaciones descentralizadas (DApps) y contratos inteligentes. Fue propuesto por primera vez por Vitalik Buterin en 2013 y lanzado en 2015.

A diferencia de Bitcoin, que se centra principalmente en ser una moneda digital, Ethereum va más allá al proporcionar una infraestructura para la creación de aplicaciones descentralizadas en su blockchain. Utiliza su propio lenguaje de programación llamado Solidity para escribir contratos inteligentes.

Ha sido un catalizador para el crecimiento del ecosistema de las criptomonedas y la tecnología blockchain en general. Ha abierto nuevas posibilidades para la creación de aplicaciones descentralizadas y ha fomentado la innovación en el espacio de las finanzas descentralizadas (DeFi), donde se pueden crear y acceder a servicios financieros sin intermediarios tradicionales.

3.1.5. Contratos inteligentes

3.1.5.1 Definición y características de los contratos inteligentes

Los contratos inteligentes, también conocidos como smart contracts, son programas informáticos que ejecutan y hacen cumplir automáticamente acuerdos predefinidos en un contrato. Estos contratos son escritos utilizando lenguajes de programación específicos y se ejecutan en una plataforma blockchain.

Las principales características de los contratos inteligentes son:

1. Autonomía: Los contratos inteligentes funcionan de forma autónoma, sin necesidad de intervención humana una vez que se han implementado en la blockchain. Esto se debe a que están diseñados para ejecutarse automáticamente en respuesta a condiciones predefinidas.

2. Inmutabilidad: Una vez que se ha registrado un contrato inteligente en la blockchain, no se puede modificar. Esto garantiza la transparencia y la confianza en el acuerdo, ya que todas las partes involucradas pueden verificar y auditar el código del contrato.

3. Ejecución precisa: Los contratos inteligentes se ejecutan exactamente como están programados, sin posibilidad de interpretaciones ambiguas o errores humanos. Esto asegura que las condiciones del contrato se cumplan de manera precisa y confiable.

4. Seguridad: Los contratos inteligentes están protegidos criptográficamente y se ejecutan en una red descentralizada y segura, lo que reduce el riesgo de fraudes o manipulaciones. La tecnología blockchain proporciona una capa adicional de seguridad y confianza en la ejecución de los contratos.

Hicimos uso de múltiples fuentes de información para investigar y documentar los contratos inteligentes. *(3. Artículo de introducción a los contratos inteligentes)*

3.1.5.1 Lenguajes de programación para la creación de contratos inteligentes

Existen varios lenguajes de programación que se utilizan para crear contratos inteligentes. A continuación, se mencionan algunos de los lenguajes más populares en la actualidad:

1. Solidity: Solidity es el lenguaje de programación más comúnmente utilizado para desarrollar contratos inteligentes en la plataforma Ethereum. Es un lenguaje de alto nivel inspirado en C++, y su sintaxis se centra en la escritura de contratos inteligentes y la interacción con la plataforma Ethereum. Esto se ampliará en el punto 3.1.6.

2. Vyper: Vyper es otro lenguaje de programación *(4. Vyper)* utilizado en la plataforma Ethereum. A diferencia de Solidity, Vyper está diseñado para ser más seguro y fácil de auditar. Su sintaxis es más simple y restringe ciertos elementos de programación que pueden dar lugar a errores o vulnerabilidades.

3. Chaincode (Go): Chaincode *(5. Chaincode)*, también conocido como "contrato inteligente" en la plataforma Hyperledger Fabric, se puede escribir en Go (Golang). Go es un lenguaje de programación popular debido a su eficiencia y facilidad de uso. Chaincode se ejecuta en una red de

blockchain permissioned (con permisos) y se utiliza en aplicaciones empresariales.

4. Michelson: Michelson es el lenguaje de programación utilizado en la plataforma Tezos para escribir contratos inteligentes. Está diseñado para ser seguro y expresivo, y se utiliza para definir las reglas y condiciones de los contratos en Tezos.

Estos son solo algunos ejemplos de lenguajes de programación utilizados en la creación de contratos inteligentes. Cada plataforma blockchain puede tener su propio conjunto de lenguajes de programación o variaciones de los mencionados anteriormente. Es importante investigar y comprender el lenguaje específico utilizado en la plataforma blockchain en la que se desea desarrollar un contrato inteligente.

3.1.6. Solidity

Solidity es un lenguaje de programación de alto nivel diseñado específicamente para desarrollar contratos inteligentes en la plataforma Ethereum. Es uno de los lenguajes más populares y ampliamente utilizado en el ecosistema de las criptomonedas y contratos inteligentes.

Solidity se inspira en lenguajes de programación como C++ y JavaScript, lo que facilita su adopción para aquellos desarrolladores familiarizados con estos lenguajes. Proporciona una sintaxis similar y ofrece características avanzadas para escribir contratos inteligentes seguros y funcionales.

Una de las características más importantes de Solidity es su capacidad para interactuar directamente con la plataforma Ethereum. Esto significa que los contratos inteligentes escritos en Solidity pueden comunicarse con otros contratos, enviar y recibir criptomonedas, y acceder a la información almacenada en la blockchain de Ethereum.

Solidity también permite la programación de contratos inteligentes complejos, con funciones, estructuras de datos, eventos y herencia. Además, proporciona herramientas para manejar la seguridad y la eficiencia en la escritura de contratos.

A continuación, se describen algunos de los tipos de datos y funciones más comunes en Solidity, información más específica se puede encontrar en la referencia *(6. Solidity)*:

Tipos de datos:

- Enteros (int y uint): Representan números enteros con y sin signo, respectivamente. Pueden tener diferentes tamaños, como int8, uint256, etc.
- Booleanos (bool): Representan los valores booleanos verdadero o falso.
- Direcciones (address): Representa una dirección Ethereum de 20 bytes.
- Puede representar tanto una cuenta externa como un contrato.
- Cadenas de texto (string): Representa una secuencia de caracteres codificados en UTF-8.
- Arreglos (arrays): Pueden ser de tamaño fijo o dinámico y pueden contener elementos del mismo tipo de dato.
- Estructuras (structs): Permite crear tipos de datos personalizados que agrupan varios valores relacionados.
- Enumeraciones (enum): Permite definir un conjunto de valores nombrados.

Funciones:

- Funciones de lectura (view): Son funciones que no modifican el estado del contrato y solo devuelven valores. Estas funciones no consumen gas al ser llamadas.
- Funciones de modificación (non-view): Son funciones que pueden modificar el estado del contrato y potencialmente consumir gas al ser ejecutadas.
- Funciones de pago (payable): Estas funciones permiten recibir Ether (la criptomoneda de Ethereum) al ser llamadas y suelen utilizarse para implementar transacciones de pago en los contratos.

- Modificadores (modifiers): Son fragmentos de código que permiten establecer restricciones de seguridad sobre la ejecución de funciones públicas.

- Eventos (events): Permiten la emisión y registro de eventos en el contrato, que pueden ser escuchados por aplicaciones externas para reaccionar a cambios en el estado del contrato.

Para desarrollar contratos inteligentes en Solidity, se puede utilizar el editor Remix. Remix es un IDE (Entorno de Desarrollo Integrado) basado en web que proporciona un entorno de programación interactivo para Solidity. Permite escribir, compilar y desplegar contratos inteligentes directamente desde el navegador web, lo que facilita el proceso de desarrollo.

Remix ofrece características como resaltado de sintaxis, autocompletado de código, depuración paso a paso y una interfaz intuitiva para interactuar con los contratos inteligentes desplegados. También proporciona herramientas de análisis estático para identificar posibles problemas de seguridad y optimizaciones de código.

Para comenzar a utilizar Remix, simplemente se necesita acceder a su sitio web oficial (https://remix.ethereum.org/) y comenzar a escribir y probar los contratos inteligentes en Solidity.

3.1.7. Tokens

En el caso de los contratos inteligentes, los tokens pueden ser utilizados como unidades de valor dentro de esos contratos. Por ejemplo, imaginemos un contrato inteligente que establece las condiciones para un crowdfunding, donde las personas pueden contribuir con fondos para financiar un proyecto. En este caso, los tokens podrían representar las contribuciones realizadas por los participantes. Cada vez que alguien contribuye con dinero al proyecto, se le asigna un número determinado de tokens en función de su contribución.

Los tokens pueden tener diferentes funciones dentro de los contratos inteligentes. Pueden servir como recompensas, como derechos de voto o incluso como representaciones de activos físicos, como acciones o propiedades. Estos

tokens se crean, registran y transfieren en la cadena de bloques de manera segura y transparente.

Además, los tokens pueden utilizarse como una forma de incentivar a las personas a participar en los contratos inteligentes. Por ejemplo, en un sistema de recompensas basado en tokens, los participantes pueden recibir tokens como incentivo por completar ciertas tareas o cumplir determinados objetivos dentro del contrato inteligente.

3.1.8. DApps

DApps, o aplicaciones descentralizadas, son aplicaciones que se ejecutan en una red blockchain. Estas aplicaciones aprovechan la naturaleza descentralizada de la blockchain para evitar cualquier punto único de fallo. Las DApps pueden incluir todo, desde juegos hasta plataformas de financiación colectiva, y utilizan contratos inteligentes para ejecutar transacciones y almacenar información en la blockchain. En el caso de nuestra plataforma, la gestión de la información del vehículo se llevaría a cabo a través de una DApp, donde cada transacción (por ejemplo, cambio de propiedad, registro de reparaciones, actualización de kilometraje) se registraría a través de contratos inteligentes y se almacenaría de forma segura y transparente en la blockchain. Esto proporciona un nivel de seguridad y transparencia sin precedentes en la gestión de la información del vehículo.

3.2. Tecnologías centralizadas en la gestión de información de vehículos

En relación con las tecnologías actuales en la gestión de información de vehículos, existen diversos sistemas que se encargan de registrar y almacenar información relevante, como por ejemplo el historial de mantenimiento y reparación. Sin embargo, estos sistemas suelen estar centralizados y bajo el control de una sola entidad, lo que puede generar problemas de confidencialidad y manipulación de datos.

Además, la gestión de información de vehículos también puede verse afectada por la falta de estandarización de los datos y la ausencia de un sistema unificado que permita compartir la información entre diferentes entidades.

3.2.1. Libro electrónico de mantenimiento - DGT

El libro electrónico de mantenimiento *(7. DGT)* es un sistema innovador que permite a los talleres comunicar a la Dirección General de Tráfico (DGT) los trabajos de mantenimiento y reparación realizados en los vehículos que atienden. Este sistema digitalizado reemplaza al tradicional libro físico de mantenimiento y ofrece una serie de características y ventajas significativas.

Una de las principales características del libro electrónico de mantenimiento es su capacidad para recopilar y almacenar de manera electrónica toda la información relacionada con los servicios realizados en un vehículo por múltiples empresas. Esto incluye los datos del taller, la identificación del vehículo, las fechas y los detalles de los trabajos realizados, como cambios de aceite, revisiones periódicas, reparaciones, etc.

La comunicación de estos datos se realiza de manera segura y en tiempo real a través de plataformas y sistemas de la DGT. Esto garantiza una mayor transparencia y trazabilidad de los servicios prestados, lo cual resulta beneficioso tanto para los talleres como para los propietarios de los vehículos.

Además, el libro electrónico de mantenimiento ofrece la posibilidad de que los propietarios de los vehículos accedan a esta información de forma sencilla y rápida. A través de portales en línea o aplicaciones móviles, los usuarios pueden consultar el historial de mantenimiento de su vehículo, conocer las fechas y detalles de los servicios realizados, así como tener un control más preciso sobre el mantenimiento de su automóvil.

Otra ventaja importante es la capacidad del sistema para generar alertas y recordatorios automáticos. Estas notificaciones pueden incluir recordatorios de citas de servicio, fechas de renovación de seguros, próximas revisiones periódicas, entre otros. Esto ayuda a los propietarios a mantener su vehículo en buen estado y a cumplir con las obligaciones legales y de seguridad.

El libro electrónico de mantenimiento también facilita la gestión y control por parte de los talleres. Permite llevar un registro detallado de los trabajos realizados, lo cual resulta útil para la facturación, la gestión de garantías y la planificación de tareas futuras. Asimismo, ayuda a mantener una comunicación más eficiente con la DGT al agilizar los trámites administrativos relacionados con los servicios de mantenimiento.

Los talleres integrados en el servicio del libro electrónico de mantenimiento, se encargan de anotar y comunicar a la DGT la información de las reparaciones y mantenimientos que realizan.

Esta información es pública, de manera que se pueden consultar las reparaciones e intervenciones realizadas sobre cualquier vehículo, aunque únicamente aparecerán las actuaciones realizadas por los talleres integrados en el servicio del libro taller.

Poder consultar el historial de mantenimiento y reparaciones aporta transparencia al mercado de compraventa de vehículos y fomenta el correcto mantenimiento de los mismos.

Cuanta más información esté disponible sobre el mantenimiento de un vehículo, más fácil será por parte de un posible comprador conocer y evaluar su estado real.

Cualquier persona puede acceder al informe de su vehículo a través de la plataforma oficial de la DGT, https://sede.dgt.gob.es/es/vehiculos/informe-de-vehiculo/. En esta página el usuario puede elegir entre un "Informe reducido" o "Informe completo", entre otros mucho más específicos como el "Informe datos técnicos", "Informe de cargas", etc….

Para cualquiera de ellos será necesario autentificarse como persona física con el DNI.

Figura 3.1: Pantalla de acceso a la autenticación de usuario en la DGT

Informe reducido sin incidencias

Informe reducido con incidencias

Figura 3.2: Ejemplo de informe reducido

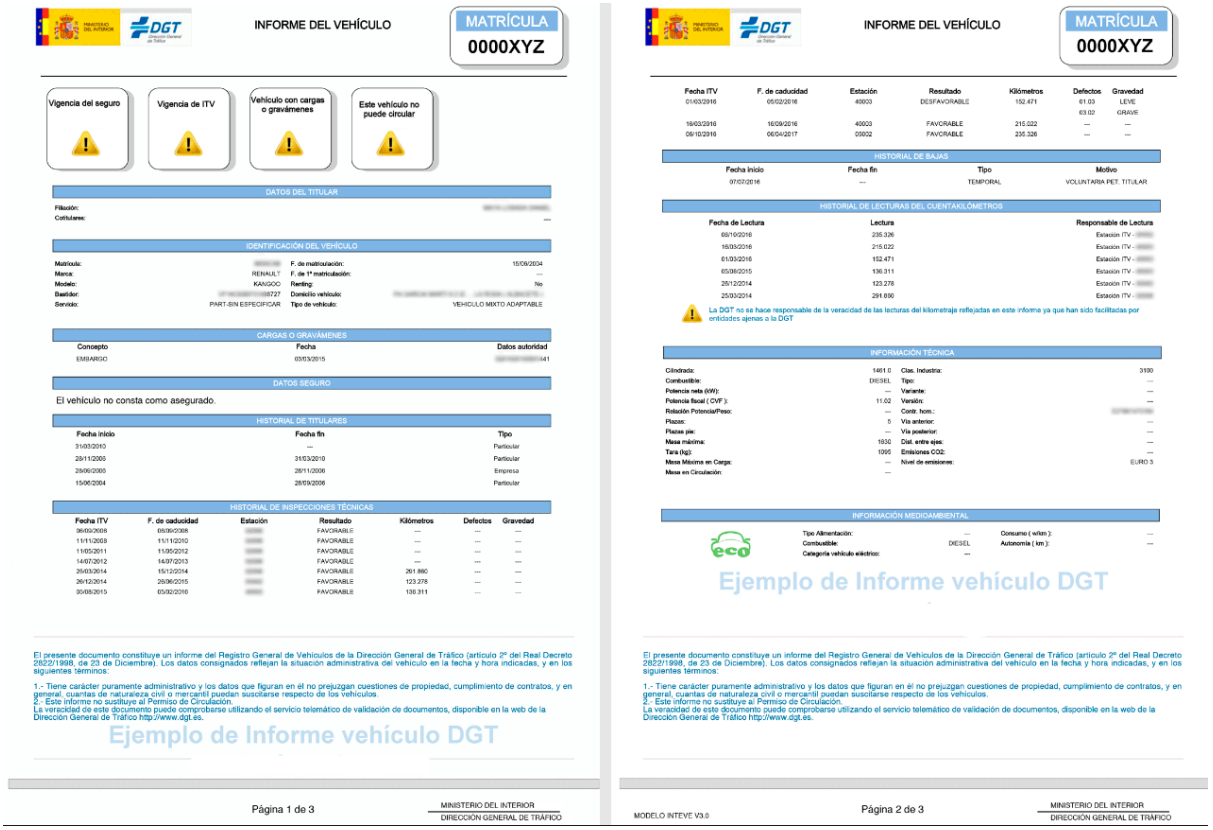


Figura 3.3: Ejemplo de informe completo

3.2.1. Carfax

Carfax *(8.Carfax)* es una empresa internacional que provee historiales de vehículos desde su fundación en Estados Unidos en 1984.

Proporciona información sobre el historial de los vehículos. Esta información incluye datos sobre accidentes, reparaciones, kilometraje y cualquier otro dato relevante que pueda afectar al valor del vehículo. Los usuarios de Carfax pueden acceder a esta información introduciendo el número de matrícula del vehículo y pagando una tarifa.

La plataforma obtiene su información a partir de diferentes fuentes, como talleres, compañías de seguros, departamentos de vehículos a motor y otras fuentes gubernamentales y privadas. La información se compila en un informe detallado que se presenta al usuario. Además, Carfax ofrece una garantía de recompra en caso de que la información proporcionada sea inexacta o incompleta.

La plataforma está estructurada de tal manera que los usuarios pueden acceder fácilmente a la información que necesitan. La página principal de Carfax ofrece una barra de búsqueda donde los usuarios pueden introducir el número de matrícula del vehículo. A continuación, se les dirige a una página de resultados que muestra un resumen de la información disponible. Desde aquí, los usuarios pueden elegir pagar para obtener un informe más detallado.

Aunque Carfax es una plataforma útil para aquellos que desean comprar un vehículo usado, es importante tener en cuenta que la información que proporciona no siempre es completa. Por lo tanto, es importante utilizar Carfax junto con otras herramientas y técnicas para garantizar que se tome una decisión de compra informada.

Figura 3.4: Ejemplo de informe Carfax

3.2.1. Autocheck

AutoCheck *(9. Autocheck)* es un proveedor de informes de historial de vehículos que permite a los usuarios verificar el historial de los automóviles usados antes de comprarlos. La compañía fue fundada en 1996 y tiene su sede en Santa Bárbara, California.

AutoCheck proporciona informes de historial de vehículos que incluyen información sobre los propietarios anteriores del automóvil, historial de accidentes, estado del título y otros detalles importantes.

Para obtener un informe de AutoCheck, los usuarios deben proporcionar el número de identificación del vehículo (VIN) del automóvil que desean verificar. AutoCheck genera un informe que incluye un resumen del historial del vehículo y un desglose detallado de cualquier incidente o accidente reportado. El informe también incluye información sobre propietarios anteriores, así como cualquier problema de título o gravámenes en el automóvil.

Los informes de AutoCheck están disponibles tanto para compradores individuales como para concesionarios. La compañía ofrece varias opciones de precios según la cantidad de informes necesarios, con descuentos para la compra de lotes de informes de múltiples vehículos. AutoCheck también proporciona una aplicación móvil que permite a los usuarios verificar los historiales de los vehículos sobre la marcha.

La tecnología de AutoCheck se basa en datos de múltiples fuentes, incluyendo compañías de seguros, agencias de aplicación de la ley y otros socios de la industria.

Esta plataforma es ampliamente utilizada por compradores de automóviles usados en los Estados Unidos y Canadá para tomar decisiones informadas y garantizar la calidad y la confiabilidad de los vehículos que están considerando adquirir.

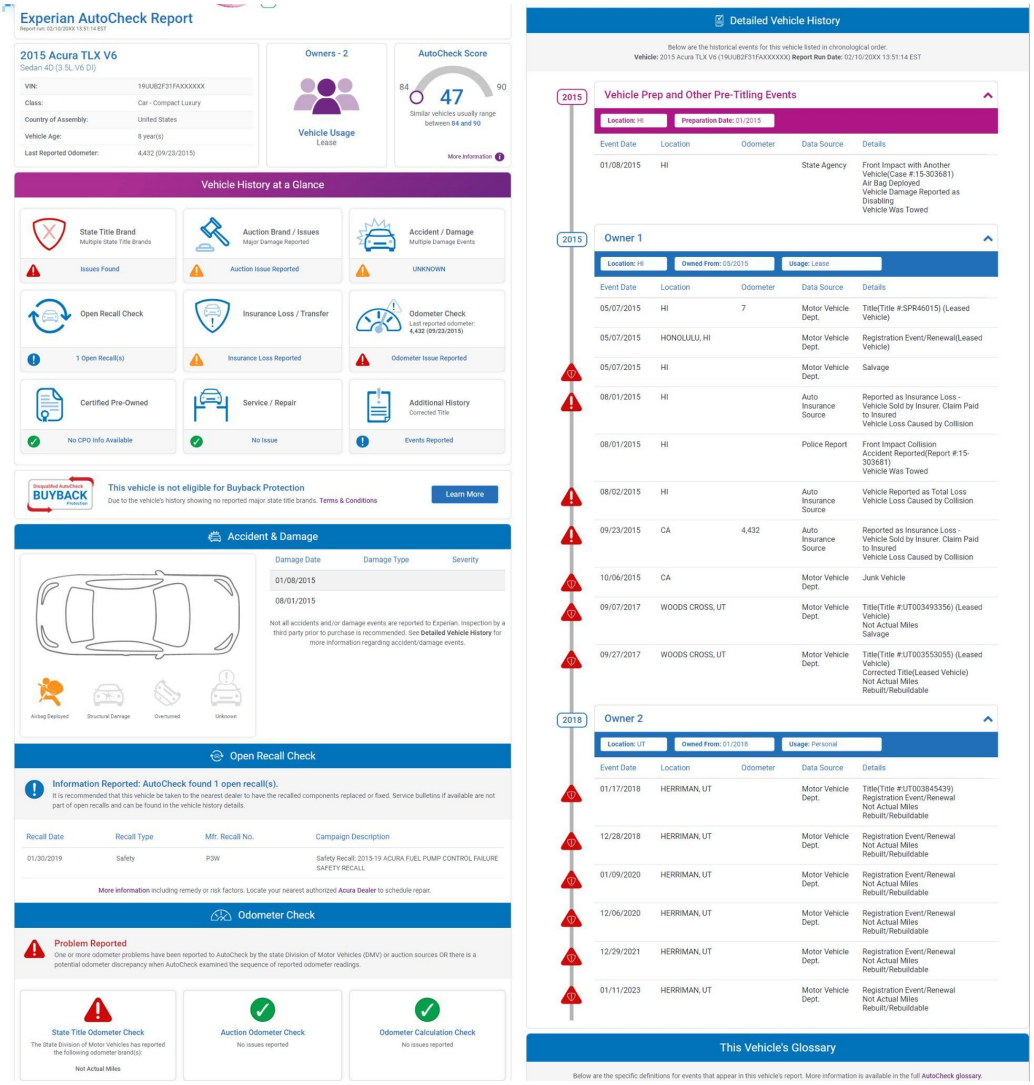


Figura 3.4: Ejemplo de informe AutoCheck

3.3. Tecnologías descentralizadas relevantes

Aunque actualmente no hay plataformas que admitan la implementación de la tecnología blockchain para los registros de vehículos, hemos encontrado referencias a otros proyectos o investigaciones que exploran aplicaciones similares en otras industrias.

Por ejemplo, hemos encontrado un documento de investigación que detalla la implementación de un sistema blockchain para el mantenimiento de aviones con el fin de administrar sus registros.

Además hemos encontrado una propuesta de Renault, XCEED *(10. XCEED)* , como solución para verificar los datos de los componentes de sus vehículos. El

proyecto "Extended Compliance End-to-End Distributed" (XCEED) surgió de la asociación de Renault con otras empresas para desarrollar una solución de cadena de suministro blockchain mediante el intercambio de datos.

En general, lo que XCEED hace es compartir la información sobre el cumplimiento normativo de los componentes del vehículo, entre los fabricantes de piezas y los fabricantes de automóviles.

3.3.1. Tecnología Blockchain para mantenimiento e ingeniería de aeronaves

El documento titulado "Blockchain Technology for Aircraft Maintenance and Engineering" *(11. Artículos blockchain en Aerolinea)* fue preparado por Albert S. Choi para STX Aero Service y la Universidad de Corea. Este documento es una propuesta académica y profesional que explora la aplicación de la tecnología blockchain en la industria de mantenimiento y reparación de aeronaves (MRO).

La propuesta se centra en la creación de una plataforma de registro blockchain que puede mejorar la eficiencia y la trazabilidad en la industria de MRO. Los problemas actuales en esta industria incluyen la operación cerrada del servicio MRO entre los participantes, la trazabilidad difícil en los componentes, incluyendo la Parte Limitada por Vida (LLP), la compartición limitada de documentos por los usuarios relevantes, la generación de documentos onerosa y el almacenamiento de documentos no comunicativos.

La plataforma propuesta utiliza la tecnología blockchain para abordar estos problemas. Por ejemplo, la plataforma puede facilitar la operación abierta del servicio MRO entre los participantes, permitir una fácil trazabilidad de los componentes y su historial, permitir la compartición abierta de documentos por los usuarios relevantes, facilitar la rápida generación de documentos y proporcionar herramientas comunicativas para el almacenamiento de documentos.

El documento también presenta un estudio de caso detallado sobre cómo se puede aplicar la tecnología blockchain en el mantenimiento del tren de aterrizaje de un avión. Este estudio de caso muestra cómo se pueden utilizar los ID distribuidos (DID) y las credenciales para rastrear y registrar las actividades de mantenimiento.

Además, el documento explica el concepto de DID y cómo se puede utilizar en la plataforma de registro blockchain. Cada usuario tiene un DID único con múltiples credenciales. El DID y todas las credenciales se escriben en la cadena de bloques de ID.

En resumen, este documento propone una solución innovadora para mejorar la eficiencia y la trazabilidad en la industria de MRO utilizando la tecnología blockchain. Esta propuesta tiene el potencial de transformar la forma en que se realiza el mantenimiento y la reparación de aeronaves, lo que a su vez puede conducir a mejoras significativas en la seguridad y la eficiencia de la industria de la aviación.

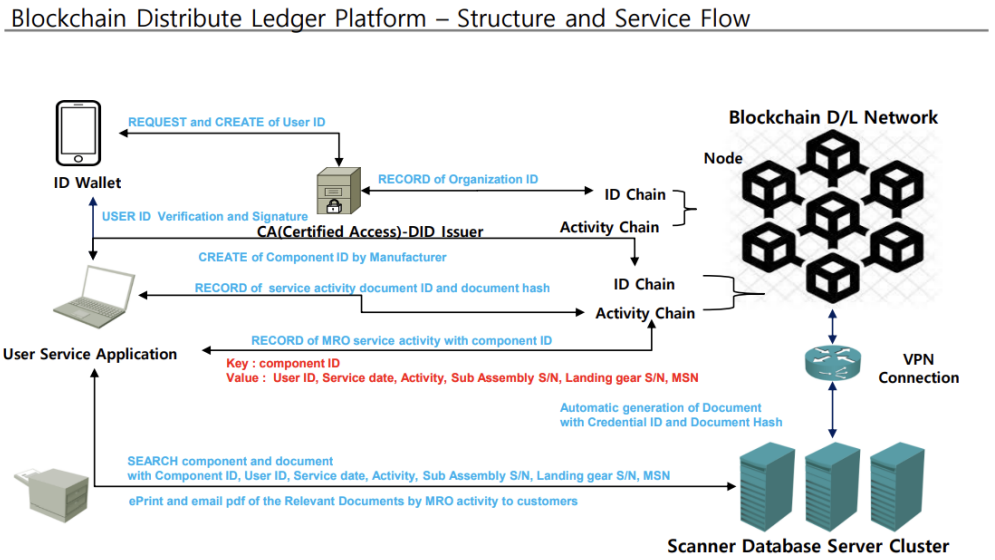


Figura 3.5: Diagrama de la aplicación propuesta a desarrollar

Capítulo 4 - Tecnologías utilizadas

4.1. Desarrollo Front-end

4.1.1. HTML

HTML (HyperText Markup Language) es el lenguaje de marcado estándar para crear páginas web. Se utiliza para estructurar el contenido de la plataforma y es la base sobre la que se construye la interfaz de usuario.

4.1.2. CSS

CSS, o Cascading Style Sheets (Hojas de estilo en cascada), es un lenguaje de diseño utilizado en el desarrollo web para controlar el aspecto y la presentación visual de una página HTML. CSS se utiliza para definir cómo se verán los elementos HTML en términos de color, tamaño, fuente, disposición y otros atributos visuales.

A través de reglas de estilo, CSS permite aplicar estilos de manera consistente en todo un sitio web. Estas reglas consisten en selectores que eligen los elementos HTML a los que se aplicará el estilo, y declaraciones que especifican las propiedades y valores que se asignarán a esos elementos.

Al separar la presentación visual del contenido HTML, CSS proporciona una mayor flexibilidad y mantenibilidad en el desarrollo web. Permite la creación de diseños responsivos y adaptables a diferentes dispositivos, así como la personalización del aspecto de los elementos HTML sin alterar la estructura del documento.

4.1.3. JavaScript

JavaScript es un lenguaje de programación de alto nivel que es fundamental para el desarrollo web. Nos permite crear interactividad en nuestra plataforma, gestionando la lógica de negocio y la interacción con el usuario en el lado del cliente. La elección de JavaScript también está alineada con el uso de Node.js en el lado del servidor, permitiendo un entorno de desarrollo uniforme y eficiente.

4.1.4. EJS

EJS (Embedded JavaScript) es un motor de plantillas de JavaScript que se utiliza en el desarrollo web front-end. Proporciona una forma sencilla y eficiente de generar y renderizar contenido HTML dinámico utilizando JavaScript. EJS se basa en una sintaxis similar a JavaScript, lo que facilita su comprensión y uso para aquellos familiarizados con este lenguaje de programación.

Una de las características principales de EJS es su capacidad para incrustar lógica y código JavaScript dentro de plantillas HTML. Esto permite generar contenido dinámico en función de datos provenientes del servidor o de variables definidas en el cliente. Mediante la inclusión de etiquetas especiales en el código HTML, se pueden insertar expresiones de JavaScript para realizar operaciones, evaluaciones condicionales, bucles y otras lógicas de programación.

4.1.5. Bootstrap

Bootstrap es un framework front-end popular y de código abierto que facilita el diseño y desarrollo de sitios web y aplicaciones web responsivas y visualmente atractivas. Fue creado por Twitter y está diseñado para ser fácil de usar, flexible y compatible con múltiples navegadores.

Combina CSS y JavaScript para estilizar los elementos de una página HTML. Permite mucho más que, simplemente, cambiar el color de los botones y los enlaces.

Es una herramienta que proporciona interactividad en la página, por lo que ofrece una serie de componentes que facilitan la comunicación con el usuario, como menús de navegación, controles de página, barras de progreso y más.

4.2. Desarrollo Back-end

4.2.1. Node.js

Node.js es un entorno de ejecución de JavaScript en el lado del servidor. Nos permite utilizar JavaScript tanto en el cliente como en el servidor, lo que facilita la coherencia del código y reduce la curva de aprendizaje para los desarrolladores. Node.js es conocido por su escalabilidad y eficiencia, lo que lo hace ideal para nuestra plataforma, que necesita manejar múltiples solicitudes y transacciones en tiempo real.

4.2.1.1. Librerías utilizadas

- axios: Axios es una biblioteca de JavaScript que se utiliza para realizar solicitudes HTTP desde el navegador y Node.js. En nuestro proyecto, axios se utiliza para hacer llamadas HTTP a las APIs y a otros servicios web.
- bcrypt: Bcrypt es una biblioteca de Node.js para el hash y el salting de contraseñas. En nuestra plataforma, se utiliza para garantizar la seguridad de las contraseñas de los usuarios.
- crypto: Crypto es un módulo de Node.js que proporciona funcionalidades criptográficas. Se utiliza en nuestro proyecto para asegurar la información y las comunicaciones.
- dotenv: Dotenv es un módulo que carga variables de entorno desde un archivo **.env** a **process.env**. En nuestra plataforma, se utiliza para gestionar las variables de entorno.
- express-mysql-session: Es un almacenamiento de sesiones MySQL para Express. Se utiliza para almacenar la información de la sesión del usuario en la base de datos MySQL.

- express-session: Es un middleware de gestión de sesiones para Express. Se utiliza para manejar las sesiones de usuario en nuestra plataforma.
- express-validator: Es un conjunto de middlewares de validación y saneamiento para Express. Se utiliza para validar y sanear los datos de entrada del usuario en nuestra plataforma.
- fs-extra: FS-extra agrega métodos adicionales de manejo de archivos y directorios a Node.js fs. Se utiliza en nuestro proyecto para leer y escribir archivos.
- ganache: Ganache proporciona una red Ethereum personalizada para pruebas de desarrollo. Se utiliza en nuestro proyecto para probar los contratos inteligentes antes de desplegarlos en la red Ethereum.
- jquery: jQuery es una biblioteca de JavaScript rápida y concisa que simplifica la manipulación de documentos HTML, el manejo de eventos, la animación y las interacciones Ajax. Se utiliza en nuestro proyecto para manipular el DOM y manejar los eventos en la interfaz de usuario.
- jsonwebtoken: jsonwebtoken es una implementación de JSON Web Tokens. Se utiliza en nuestro proyecto para autenticar y autorizar a los usuarios y para mantener las sesiones de los usuarios.
- jwt-encode: jwt-encode es un módulo que codifica JSON a JWT. Se utiliza en nuestro proyecto para codificar y decodificar tokens JWT.
- mysql: MySQL es un módulo de Node.js para interactuar con bases de datos MySQL. Se utiliza en nuestro proyecto para realizar operaciones de base de datos.
- nodemailer: Nodemailer es un módulo para enviar correos electrónicos desde Node.js. Se utiliza en nuestro proyecto para enviar correos electrónicos a los usuarios.

- nodemon: Nodemon es una utilidad que monitorea los cambios en el código fuente y reinicia automáticamente el servidor. Se utiliza en nuestro proyecto para mejorar la productividad durante el desarrollo.
- solc: Solc es un compilador de Solidity. Se utiliza en nuestro proyecto para compilar nuestros contratos inteligentes escritos en Solidity.
- web3: Web3.js es la biblioteca de JavaScript de Ethereum y se utiliza para interactuar con una red Ethereum local o remota. Se utiliza en nuestro proyecto para interactuar con la red Ethereum y para llamar a las funciones de los contratos inteligentes.

4.2.2. Express

Express es un framework de desarrollo web rápido y minimalista para Node.js. Proporciona una capa de abstracción sobre el servidor HTTP de Node.js, lo que simplifica la creación de aplicaciones web y APIs de manera eficiente.

Express se destaca por su enfoque minimalista y su flexibilidad. Permite manejar solicitudes y respuestas HTTP de forma sencilla, además de proporcionar una amplia gama de funciones y middleware para facilitar tareas comunes en el desarrollo web, como enrutamiento, manejo de sesiones, autenticación y renderizado de plantillas.

4.2.3. SQL

SQL (Structured Query Language) es un lenguaje de programación utilizado para administrar y manipular bases de datos relacionales. Es el estándar de facto para la comunicación con sistemas de gestión de bases de datos (SGBD) y se utiliza en una amplia gama de aplicaciones y entornos.

En nuestro caso ha sido utilizado para la creación de la base de datos centralizada de la aplicación web.

Hemos utilizado MySql para la gestión de dicha base de datos.

4.2.4. MySQL

MySQL es un sistema de gestión de bases de datos relacionales (SGBD) muy popular y ampliamente utilizado. Fue desarrollado originalmente por MySQL AB y ahora es propiedad de Oracle Corporation. MySQL se caracteriza por ser de código abierto y está disponible de forma gratuita, aunque también existe una versión comercial con características adicionales.

Se basa en el lenguaje SQL (Structured Query Language) y proporciona una forma eficiente y confiable de almacenar, organizar y administrar grandes cantidades de datos. Es utilizado por una amplia variedad de aplicaciones y sitios web, desde pequeñas aplicaciones personales hasta sistemas empresariales a gran escala.

Hemos usado XAMPP para su gestión de forma local y desarrollo de nuestro sistema.

4.2.5. XAMPP

XAMPP es un paquete de software que proporciona un entorno de desarrollo web local todo en uno.

Permite configurar rápidamente un entorno de desarrollo local que contiene un servidor web, una base de datos y los lenguajes de programación necesarios. Esto te permite desarrollar y probar aplicaciones web en tu propia máquina sin necesidad de una conexión a Internet o un servidor en vivo.

Es fácil de instalar y utilizar, y proporciona una solución conveniente para desarrolladores que desean crear aplicaciones web en un entorno de desarrollo local. Sin embargo, es importante tener en cuenta que XAMPP está diseñado para ser utilizado en entornos de desarrollo y no se recomienda su uso en un entorno de producción en línea debido a consideraciones de seguridad.

4.2.6. Solidity

Como ya hemos definido anteriormente, es un lenguaje de programación orientado a contratos inteligentes (smart contracts) que se ejecutan en la plataforma Ethereum. Fue desarrollado específicamente para permitir la creación y programación de contratos inteligentes en la cadena de bloques de Ethereum.

Lo hemos utilizado para elaborar el código de nuestro contrato inteligente.

4.2.7. Remix

Remix (https://remix.ethereum.org/), como ya hemos explicado anteriormente, es un entorno de desarrollo integrado (IDE) basado en la web para desarrollar contratos inteligentes en Ethereum. Remix proporciona una serie de herramientas y características que facilitan el desarrollo *(12. Remix IDE Documentación)*, la implementación y la prueba de contratos inteligentes escritos en Solidity, el lenguaje de programación utilizado para crear contratos en la red Ethereum.

Su uso en nuestro proyecto ha sido crucial para el despliegue del contrato en nuestro nodo local de Ethereum gestionado por "Geth".

4.2.8. Geth

Geth *(13. Geth)* es una implementación de código abierto del cliente de Ethereum en Go (Golang). Ethereum es una plataforma descentralizada que permite la ejecución de contratos inteligentes y la creación de aplicaciones descentralizadas (dApps).

Geth actúa como un nodo en la red Ethereum, lo que significa que se conecta a la red peer-to-peer de Ethereum y sincroniza con otros nodos para mantener una copia actualizada del libro de contabilidad distribuido de Ethereum, conocido como blockchain. Como cliente de Ethereum, Geth permite a los usuarios

interactuar con la red, enviar transacciones, ejecutar contratos inteligentes y acceder a los datos almacenados en la cadena de bloques.

Capítulo 5 - Arquitectura y diseño del sistema

Los principales actores de nuestro sistema son los propietarios de vehículos y los responsables de talleres.

La parte más importante de nuestra arquitectura es el uso de tanto una blockchain como una base de datos centralizada. Esto es clave ya que así podemos guardar los datos más sensibles en la blockchain y poder acceder a ellos de forma constante desde la base de datos a través de identificadores únicos.

El siguiente esquema describe la arquitectura de nuestra aplicación:

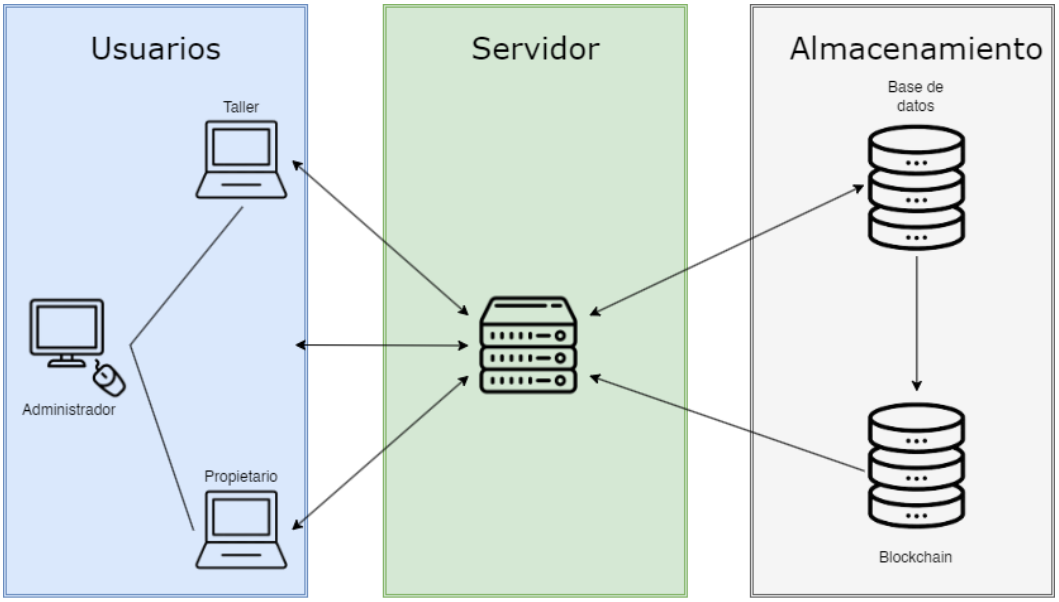


Figura 5.1: Arquitectura de la aplicación

5.1. Arquitectura MVC

Para este proyecto se ha llevado a cabo el patrón Modelo-Vista-Controlador (MVC) para estructurar y organizar nuestra aplicación de una manera eficiente. Este patrón de diseño se implementa de la siguiente manera:

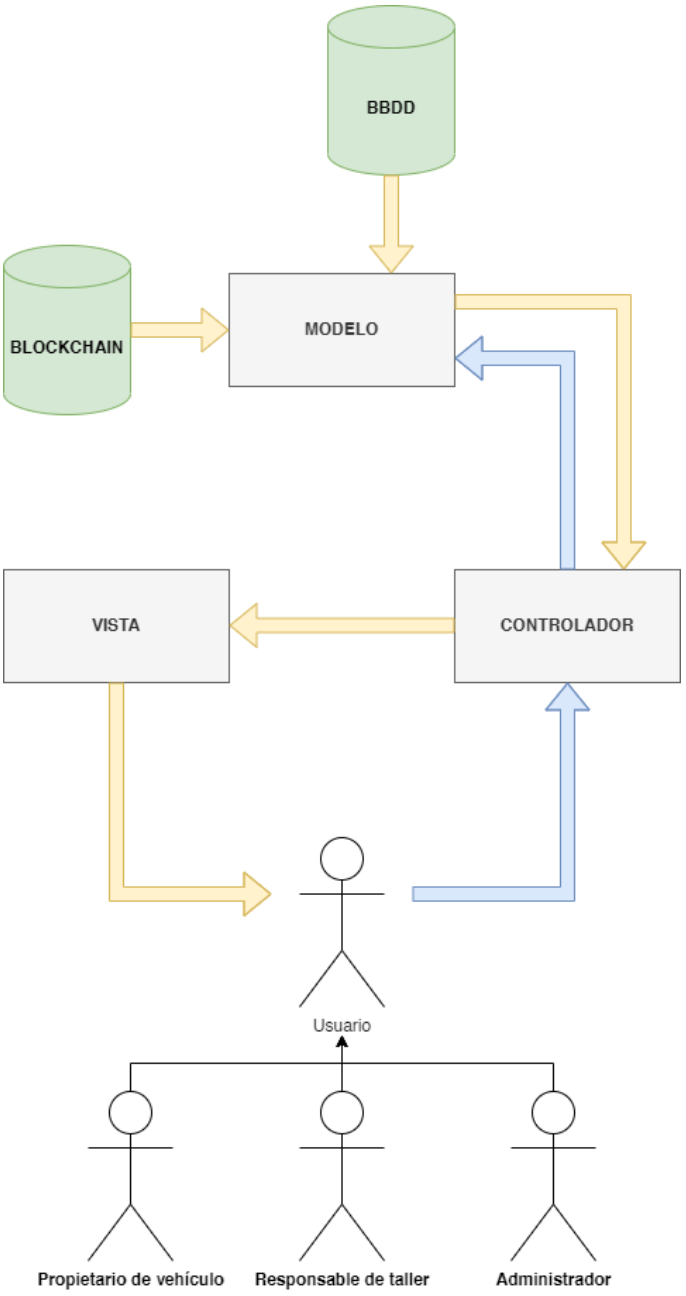


Figura 5.2: Diagrama MVC

Actores:

- **Propietario de Vehículo**: Este usuario está registrado en la plataforma y tiene la capacidad de interactuar con la aplicación para diversas funciones. Pueden ver informes relacionados con diferentes vehículos, administrar sus propios vehículos y también gestionar los permisos relacionados con la visibilidad.
- **Responsable de Taller**: Este tipo de usuario tiene la capacidad de registrar información de servicio o reparación para números de matrícula específicos presentes en la plataforma. El Administrador es responsable de la registración de este tipo de usuario.
- **Administrador**: Como administrador de la plataforma, este usuario puede realizar una amplia variedad de funciones. Son capaces de registrar usuarios, vehículos y talleres. Además, tienen la autoridad para asignar el rol de Responsable de Taller a los talleres registrados.

Modelos: Los modelos, que residen en el directorio **models/**, representan y manejan los datos de nuestra aplicación. En este directorio, **server.js** es responsable de configurar "express" y organizar todos los modelos y rutas que interactúan en el sistema. Además de establecer en qué puerto escuchará el servidor.

Vistas: Las vistas son la representación visual de nuestros modelos. Se encargan de presentar los datos al usuario en un formato legible y manejable. En nuestra estructura de proyecto, las vistas se encuentran en el directorio **views/** y sus subdirectorios. Aquí, los archivos **.ejs** representan diferentes páginas o componentes de la interfaz de usuario en nuestra aplicación.

Controladores: Los controladores son el intermediario entre nuestras vistas y modelos. Se encargan de responder a las acciones del usuario, actualizar los modelos y las vistas en consecuencia. En nuestra estructura de proyecto, los controladores se encuentran en el directorio **controllers/**. Cada archivo **.js** en este directorio actúa como un controlador para un aspecto específico de nuestra aplicación, como la autenticación (auth.js), los usuarios (user.js), los vehículos (vehicle.js), y así sucesivamente.

Además de los componentes MVC, nuestra aplicación incorpora una serie de otros elementos que son esenciales para la funcionalidad y la operación de la plataforma. Por ejemplo, el directorio **ethereum/** contiene el contrato inteligente de Ethereum (en el subdirectorio **contracts/**), así como los scripts de compilación y despliegue para el mismo (compiler.js y deploy.js). Estos contratos inteligentes, escritos en Solidity, son fundamentales para la parte descentralizada de nuestra aplicación.

5.2. Datos almacenados On-chain y Off-chain

La elección de utilizar tanto el almacenamiento on-chain como off-chain en nuestra plataforma se basa en una combinación de factores, incluyendo la necesidad de seguridad, la eficiencia del almacenamiento y la privacidad de los datos.

El almacenamiento on-chain proporciona una fuente de verdad inmutable y segura para los datos críticos del vehículo. Sin embargo, el almacenamiento de todos los datos en la cadena de bloques puede ser costoso y puede resultar en una menor eficiencia debido a las limitaciones de velocidad y tamaño de las transacciones en la cadena de bloques.

Además, la cadena de bloques es inherentemente transparente, lo que significa que cualquier información almacenada en ella es visible para todos los participantes de la red. En nuestro caso, este sistema se implementaría en una blockchain privada, y los datos del contrato no serían públicos para todos, sino que

cada rol dentro de la plataforma está configurado para que solo tenga acceso a una serie de datos.

Por ejemplo, los talleres asociados. Cada usuario autorizado a un taller únicamente podrá tener acceso a los datos que se hayan guardado en ese taller.

Por otro lado, el almacenamiento off-chain en una base de datos centralizada permite una mayor eficiencia y privacidad. Los datos almacenados off-chain pueden ser recuperados más rápidamente y pueden ser protegidos con medidas de seguridad adicionales para garantizar la privacidad de los usuarios.

La combinación de almacenamiento on-chain y off-chain permite a nuestra plataforma aprovechar lo mejor de ambas tecnologías. Los datos críticos del vehículo se almacenan en la cadena de bloques para garantizar su integridad y autenticidad, mientras que los datos menos críticos y los datos sensibles se almacenan de manera segura y eficiente off-chain.

La integración entre los datos almacenados on-chain y off-chain se realiza mediante nuestra plataforma, que se encarga de la sincronización de datos entre la cadena de bloques y la base de datos centralizada. Esto asegura que los usuarios siempre tengan acceso a la información más actual y precisa sobre los vehículos.

5.2.1. Estructura de datos On-chain

La estructura de datos queda definida en el contrato desarrollado en Solidity, este es el encargado de guardar la información más sensible de todas las entidades involucradas.

Es la pieza fundamental del sistema, ya que es la encargada de garantizar la inmutabilidad de los datos guardados, así como la transparencia de los mismos.

5.2.1.1. Variables

1. **contract_owner**: Es el propietario del contrato, probablemente el desplegador del contrato.

2. **reparationInfo**: Almacena información detallada sobre cada reparación realizada en un vehículo. Esto incluye un hash de seguridad, kilómetros del vehículo al momento de la reparación, la fecha de la reparación, una descripción de la reparación y el tipo de revisión.

3. **vehicleInfo**: Almacena información detallada de un vehículo específico. Esto incluye un mapping de reparaciones y datos del vehículo. Utiliza un hash de seguridad para comprobar que cada vez que se accede a una reparación ésta no ha sido vulnerada desde los datos guardados en base de de datos.

4. **vehicleData**: Almacena información adicional sobre un vehículo, como una lista de identificadores de reparaciones, propietarios, historial de kilometraje, fecha de matriculación, estado activo y visibilidad.

5. **workshopInfo**: Almacena información sobre talleres, que incluye una lista de identificadores de vehículos y un mapeo de reparaciones realizadas.

6. **vehicles**: Este mapeo relaciona el hash de un vehículo con su correspondiente información **vehicleInfo**.

7. **workshops**: Este mapeo relaciona el hash de un taller con su correspondiente información **workshopInfo**.

8. **authorized_workshops**: Este mapeo lleva un seguimiento de los talleres autorizados.

9. **ownerXvehicles**: Este mapeo lleva un seguimiento de los vehículos que posee cada propietario.

5.2.1.2. Funciones

1. **registerVehicleReparation**: Registra una reparación para un vehículo específico. Asegura que los kilómetros registrados son mayores que los últimos kilómetros registrados. Asocia la información de reparación con el vehículo y actualiza el historial de kilometraje.
2. **registerVehicle**: Registra un nuevo vehículo en el contrato. Asegura que el vehículo no esté ya registrado. Asocia el vehículo con su propietario y establece su visibilidad en 'privado'.
3. **getVehicleReparation**: Devuelve la información de reparación para una reparación específica de un vehículo en particular.
4. **getVehicle**: Devuelve la información del vehículo para un vehículo específico.
5. **getVehicleCurrentOwner**: Devuelve el propietario actual de un vehículo específico.
6. **addVehicleOwner**: Añade un nuevo propietario a un vehículo específico. Asegura que el vehículo esté activo y que el usuario no tenga vehículos.

7. **setVisibility**: Permite al propietario de un vehículo cambiar la visibilidad de su vehículo.
8. **authorizeWorkshop**: Autoriza a un nuevo taller para realizar reparaciones. Asegura que el taller no esté ya autorizado.
9. **addressOwner**: Devuelve el propietario del contrato.

5.2.1.3. Mofidicadores

Los modificadores son herramientas clave que posibilitan la implementación de medidas restrictivas de seguridad, dirigidas a regular la ejecución de funciones públicas. En el contrato, se aplican los modificadores que se describen a continuación:

1. **isContractOwner:** Comprueba si quien está intentando ejecutar la función es el propietario del contrato, en nuestro caso solamente podría ser el administrador de la aplicación.

2. **isWorkshopAdded:** Comprueba que el taller que se está pasando a la función, está añadido en nuestro sistema.

3. **vehicleActive:** Corrobora que el vehículo está activo en nuestro sistema.

4. **ownerVehiclesEmpty:** Comprueba que el vehículo que se ha pasado a la función tiene dueño.

5. **isVehicleOwner:** Comprueba si la dirección que se está pasando como parámetro coincide con la dirección del propietario del vehículo.

6. **isVehicleInOwnersMap:** Comprueba si la dirección del vehículo está en el mapa donde se guardan los vehículos que tiene cada dueño.

5.2.2. Datos almacenados Off-chain

Además de la información almacenada on-chain, nuestra plataforma también utiliza una base de datos centralizada off-chain para almacenar ciertos tipos de datos.

Esta base de datos es esencial para manejar la información que no necesita la inmutabilidad y la seguridad de la cadena de bloques, pero que aún es crítica para el funcionamiento de la plataforma.

Esto se debe a que gracias a esta base de datos logramos un acceso constante a los datos guardados dentro de nuestro contrato.

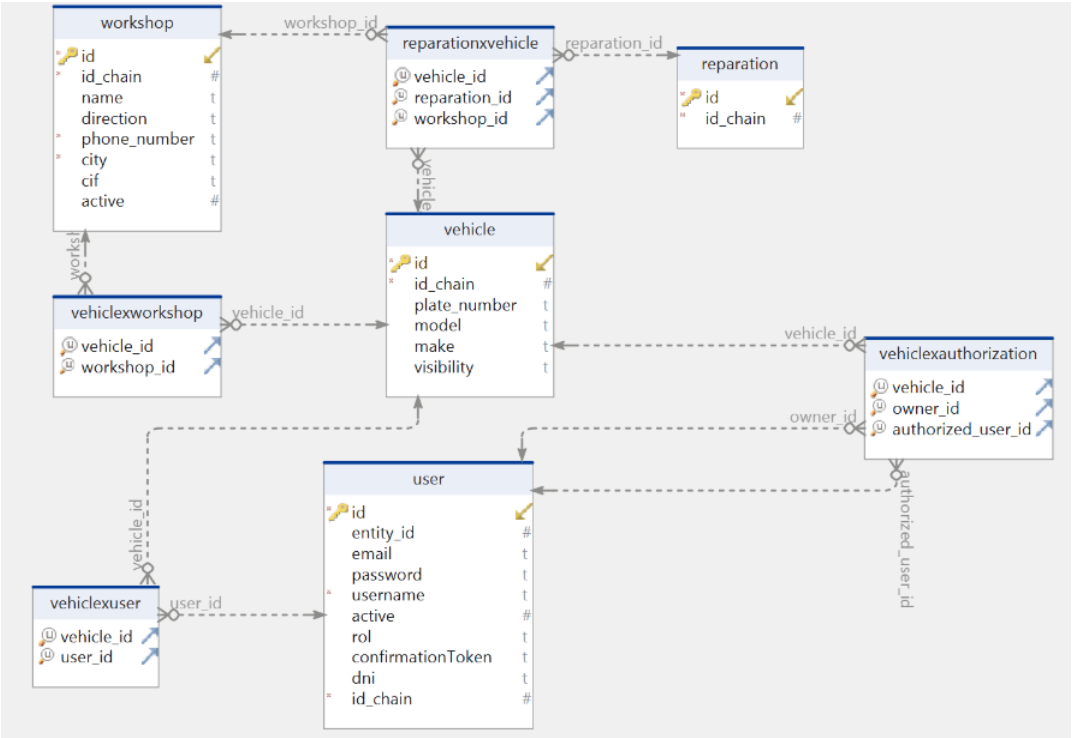


Figura 5.3: Diagrama entidad relación de la base de datos de la aplicación

El esquema incorpora nueve tablas, cada una representando aspectos distintos de las operaciones del sistema. Estas tablas son **reparation**, **reparationxvehicle**, **sessions**, **user**, **vehicle**, **vehiclexauthorization**, **vehiclexuser**, **vehiclexworkshop** y **workshop**. Cada tabla representa un elemento crucial en la gestión y el seguimiento de las operaciones y transacciones del sistema.

Empecemos con la tabla **user**. Esta tabla es fundamental en nuestro sistema ya que almacena la información de cada usuario, incluyendo su dirección de correo electrónico, contraseña, nombre de usuario, y la presencia de un rol dentro del sistema. Aquí también encontramos el campo **id_chain**, un identificador único que se almacena en la cadena de bloques para garantizar la seguridad y la inmutabilidad de los datos del usuario.

Como se puede ver en la figura 5.2, esta tabla está relacionada con **reparationxvehicle**, **vehiclexuser** y **vehiclexauthorization**, lo que indica las diversas interacciones que un usuario puede tener en este sistema, ya sea como propietario de un vehículo, usuario autorizado de un vehículo o gestor de un taller.

La tabla **vehicle** alberga la información esencial de los vehículos, como el número de matrícula, el modelo, la marca y el estado de visibilidad. Como los usuarios, los vehículos también tienen un identificador **id_chain** para asegurar la trazabilidad y la seguridad de sus datos en la cadena de bloques. Esta tabla se relaciona con **reparationxvehicle**, **vehiclexuser**, **vehiclexauthorization** y **vehiclexworkshop**, lo que nos permite rastrear las reparaciones realizadas en un vehículo, los usuarios asociados a un vehículo, las autorizaciones concedidas para un vehículo y los talleres que atienden un vehículo.

La tabla **reparation** es de vital importancia ya que gestiona las reparaciones que se realizan en los vehículos. Al igual que los usuarios y los vehículos, las reparaciones también tienen un identificador **id_chain** que se registra en la cadena de bloques. La relación entre **reparation** y **reparationxvehicle** nos permite vincular cada reparación con los vehículos específicos que se han reparado.

La tabla **workshop** almacena información relevante sobre los talleres, incluyendo su nombre, dirección, número de teléfono, ciudad, CIF y estado de actividad. Esta tabla tiene un identificador **id_chain** y está relacionada con **vehiclexworkshop** y **reparationxvehicle**, lo que nos permite ver qué vehículos se atienden en qué talleres y qué reparaciones se han realizado en cada taller.

Las tablas **vehiclexuser**, **vehiclexauthorization**, **vehiclexworkshop** y **reparationxvehicle** actúan como tablas de unión que facilitan las múltiples relaciones entre las tablas **user**, **vehicle**, **reparation** y **workshop**. Estas tablas son esenciales para obtener una visión completa de las operaciones y transacciones en el sistema.

Por último, la tabla **sessions**, no incluida en la figura 5.2, maneja las sesiones de los usuarios en el sistema, asegurando que cada interacción esté debidamente autenticada y protegida. Aunque esta tabla no se vincula directamente con las demás en términos de relaciones de clave primaria y clave externa, es esencial para el funcionamiento seguro y eficaz del sistema.

Este diseño de base de datos proporciona una gestión eficaz y segura de los datos relacionados con los usuarios, los vehículos, las reparaciones, las autorizaciones y los talleres, garantizando al mismo tiempo la trazabilidad y la inmutabilidad de los datos a través de la integración de la cadena de bloques. A continuación especificamos qué datos necesitamos guardar en nuestra base de datos on-chain y off-chain.

Capítulo 6 - Implementación de la plataforma

La implementación de la plataforma fue un proceso multifacético que implicó el uso de una variedad de tecnologías y componentes. Este capítulo detalla las diversas etapas de la implementación, desde la configuración del entorno de desarrollo hasta la creación de la interfaz de usuario y la implementación de flujos de trabajo.

6.1. Estructura del código

La estructura del código de nuestra plataforma se organiza de manera lógica y eficiente para promover la claridad y la facilidad de mantenimiento del código. Se divide en varios directorios principales que agrupan los archivos según su funcionalidad en el sistema. A continuación, se presenta un resumen general del directorio y los contenidos:

```
├──config
├──controllers
├──database
├──ethereum
│  └──contracts
├──models
├──middlewares
├──public
│  ├──css
│  └──js
├──routes
└──views
   ├──html
   └──partials
```

6.1.1. Carpeta Config

Este directorio contiene los archivos de configuración que establecen parámetros importantes para el sistema.

- **init.js**: El archivo **init.js** es un script de inicialización que crea un usuario administrador cuando la aplicación se lanza por primera vez. Este script genera un hash seguro para la contraseña del administrador utilizando la biblioteca bcrypt y luego intenta insertar un nuevo usuario administrador en la base de datos.
- **truffle-config.js**: Truffle es un marco de desarrollo para Ethereum que proporciona un entorno de desarrollo local para el desarrollo, prueba y despliegue de contratos inteligentes. Este archivo configura Truffle para su uso con la aplicación.

6.1.2. Controllers

Los controladores son responsables de manejar la lógica de la aplicación y las interacciones entre el modelo (los datos) y la vista (la interfaz de usuario). Cada archivo en esta carpeta representa un controlador para una entidad diferente en la aplicación, como usuarios, vehículos, talleres, etc.

- **auth.js**: Este archivo se encarga de la autenticación y registro de usuarios. Se encarga de validar las credenciales de los usuarios, verificar la existencia de un usuario y de registrar nuevos usuarios en el sistema.
- **authorization.js**: Este controlador maneja las autorizaciones de acceso a la información de los vehículos. Gestiona la creación, eliminación y recuperación de todas las autorizaciones de un vehículo específico.
- **factory.js**: Este archivo gestiona todo lo relacionado con los talleres. Se encarga de la creación de nuevos talleres, la eliminación de talleres

existentes y la recuperación de la información de todos los talleres registrados en el sistema.

- **reparation.js**: Este archivo maneja la información sobre las reparaciones de los vehículos. Se encarga de registrar nuevas reparaciones, asociar las reparaciones a vehículos y talleres específicos y recuperar información sobre las reparaciones asociadas a un vehículo o taller específicos.
- **user.js**: Este controlador se encarga de la gestión de los usuarios. Maneja la creación de nuevos usuarios, la activación de usuarios, la validación de usuarios y la recuperación de la información de todos los usuarios registrados en el sistema.
- **vehicle.js**: Este archivo se encarga de todo lo relacionado con los vehículos. Maneja la creación de nuevos vehículos, la asociación de vehículos con usuarios y talleres, y la recuperación de la información de todos los vehículos registrados en el sistema, así como la gestión de autorizaciones y visibilidad de los vehículos.
- **workshop.js**: Este controlador maneja la información relativa a los talleres. Se encarga de la creación y eliminación de talleres, la asociación de vehículos con talleres, y la recuperación de la información de todos los talleres registrados en el sistema.

6.1.3. Ethereum

Contiene todos los componentes relacionados con la red Ethereum y la tecnología blockchain. Se encarga de gestionar los contratos inteligentes y los procesos de compilación y despliegue asociados a estos.

- **compiler.js:** El archivo **compiler.js** es esencial en nuestra plataforma ya que se encarga de la compilación de nuestros contratos inteligentes escritos en Solidity. Es decir, este script transforma el código fuente de Solidity en un formato que la Máquina Virtual Ethereum (EVM) puede interpretar y ejecutar. Este paso es necesario antes de desplegar los contratos en la red Ethereum.
- **deploy.js**: El archivo **deploy.js** es responsable de desplegar nuestros contratos inteligentes en la red Ethereum. Una vez que los contratos inteligentes se han compilado con **compiler.js**, estos scripts gestionan el proceso de migración o despliegue de los contratos a la red Ethereum, lo que permite que se puedan interactuar con ellos a través de transacciones.
- **CarChain.sol**: El archivo **CarChain.sol** contiene el código fuente de nuestro contrato inteligente. Estos contratos definen las operaciones permitidas en nuestros registros de vehículos, como la creación de nuevos registros, la actualización de los existentes y la consulta de la información almacenada.

6.1.4. Routes

La carpeta routes/ juega un papel crucial en nuestra aplicación ya que define las rutas o endpoints de nuestra API. Cada archivo dentro de esta carpeta se encarga de gestionar las solicitudes HTTP a un determinado conjunto de rutas y de dirigir estas solicitudes a los controladores apropiados.

- **admin.js:** El archivo admin.js define las rutas relacionadas con las funcionalidades de administración.
- **auth.js:** El archivo auth.js gestiona las rutas relacionadas con la autenticación de los usuarios.
- **index.js:** Define las rutas de nivel superior de nuestra aplicación.
- **login.js:** El archivo login.js se encarga de las rutas relacionadas con el inicio de sesión de los usuarios. Esto incluye la presentación del formulario de inicio de sesión y la validación de las credenciales de los usuarios.
- **owner.js:** El archivo owner.js define las rutas relacionadas con los propietarios de los vehículos. Esto podría incluir la visualización y actualización de los registros de vehículos, así como cualquier otra funcionalidad relacionada con los propietarios de los vehículos.
- **register.js:** El archivo register.js se encarga de las rutas relacionadas con el registro de nuevos usuarios. Esto incluye la presentación del formulario de registro y la creación de nuevos usuarios en nuestra base de datos.
- **reparation.js:** Este archivo define las rutas relacionadas con las reparaciones de los vehículos. Esto incluye la creación de nuevos registros de reparación, y la visualización de los historiales de reparación.
- **user.js:** El archivo user.js gestiona las rutas relacionadas con las cuentas de usuario.
- **workshop.js:** El archivo workshop.js define las rutas relacionadas con los talleres. Esto incluye la creación de nuevos talleres, la actualización de los talleres existentes y la visualización de los detalles de los talleres.

6.1.5. Views / Public

La carpeta views/ en nuestra aplicación es esencialmente donde se alojan nuestras plantillas de visualización o nuestros archivos de vistas. Estos archivos definen la presentación y el formato de la información que se envía al cliente. En otras palabras, estos archivos determinan cómo se verá nuestra aplicación en el navegador del usuario.

Las vistas se utilizan para generar el HTML que se envía al navegador del usuario. Estas vistas pueden incluir contenido estático, como etiquetas HTML y CSS, así como contenido dinámico, que se genera en tiempo de ejecución utilizando los datos proporcionados por nuestra aplicación.

En nuestra aplicación, estamos utilizando EJS (Embedded JavaScript) como nuestro motor de plantillas. Los archivos .ejs en la carpeta views/, contienen código HTML mezclado con JavaScript que nos permite generar contenido HTML dinámico.

Además, las carpetas views/ y public/ están organizadas en subcarpetas, para facilitar la gestión de las vistas. Por ejemplo, la subcarpeta partials/ contiene fragmentos de vistas que se pueden reutilizar en varias vistas diferentes, en nuestro caso el sidebar de la aplicación.

6.2. Configuración Inicial

Antes de comenzar la implementación de la plataforma, se realizó la configuración del entorno de desarrollo. Esta etapa implicó la instalación de Node.js, que proporciona un entorno de ejecución del lado del servidor que nos permitió ejecutar JavaScript. Node.js se eligió por su escalabilidad y su capacidad para manejar múltiples solicitudes simultáneamente.

6.2.1. Configuración de un nodo local de Ethereum y simulación de una red privada de blockchain

Para simular una red privada de blockchain y probar nuestro contrato inteligente, configuramos un nodo local de Ethereum utilizando Geth y Remix. Un nodo local es una instancia de la red Ethereum que se ejecuta en una máquina local. Esta configuración permite realizar pruebas en un entorno controlado, sin necesidad de conectarse a la red Ethereum pública.

Para configurar el nodo local de Ethereum, se han seguido y se deben seguir estos pasos.

1. Despliegue de GETH en nuestro ordenador.
    1.1. Dirigirse a la carpeta donde se ha instalado GETH

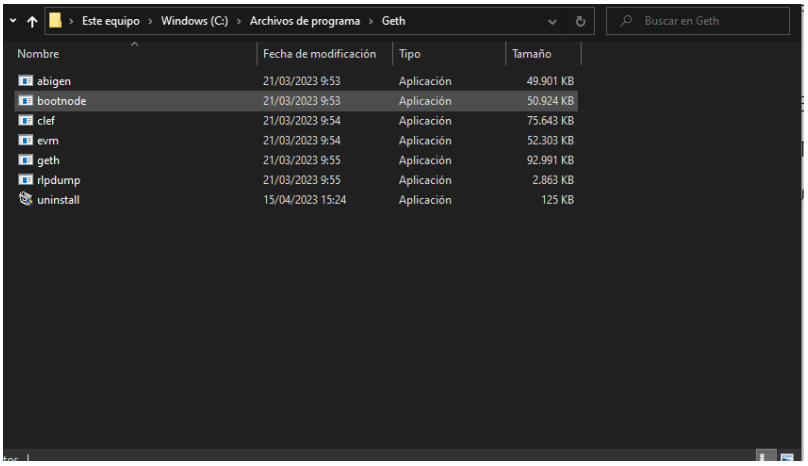


Figura 6.1: Paso 1 de instalación de GETH

1.2. Abrir una terminal de windows sobre la carpeta

Figura 6.2: Paso 2 de instalación de GETH

1.3. Iniciar el cliente de Ethereum a través del siguiente comando:

geth --http --http.corsdomain="https://remix.ethereum.org" --http.api web3,eth,debug,personal,net --vmdebug --datadir /home/user/testnet --dev console

--http: Habilita la interfaz HTTP para comunicarse con el cliente Geth a través de una API HTTP.

--http.corsdomain="https://remix.ethereum.org": Especifica el dominio permitido para las solicitudes CORS (Cross-Origin Resource Sharing). En este caso, se permite que el dominio "https://remix.ethereum.org" realice solicitudes al cliente Geth.

--http.api web3,eth,debug,personal,net: Define los módulos de la API HTTP que se habilitan. En este caso, se habilitan los módulos web3, eth, debug, personal y net. Estos módulos permiten interactuar con diferentes aspectos de la red Ethereum, como la creación de transacciones, el acceso a información de bloques, depuración, manejo de cuentas personales, entre otros.

--vmdebug: Activa el modo de depuración de la máquina virtual Ethereum (EVM). Esto permite obtener información

detallada sobre el proceso de ejecución de contratos inteligentes en la red Ethereum.

--datadir /home/user/testnet: Especifica el directorio de datos en el que se almacenarán los archivos relacionados con la blockchain de prueba (testnet) de Ethereum. En este caso, se utiliza el directorio "/home/user/testnet".

--dev: Inicia el cliente Geth en modo de desarrollo. Esto permite ejecutar una cadena de bloques local de prueba, en lugar de conectarse a la red principal de Ethereum.

console: Inicia el cliente Geth en modo de consola interactiva. Esto permite interactuar directamente con el cliente Geth utilizando comandos de JavaScript.

1.4. Una vez el nodo local está corriendo, hay que seleccionarlo como "ENVIRONMENT" en Remix para poder hacer deploy del contrato que usa el sistema.

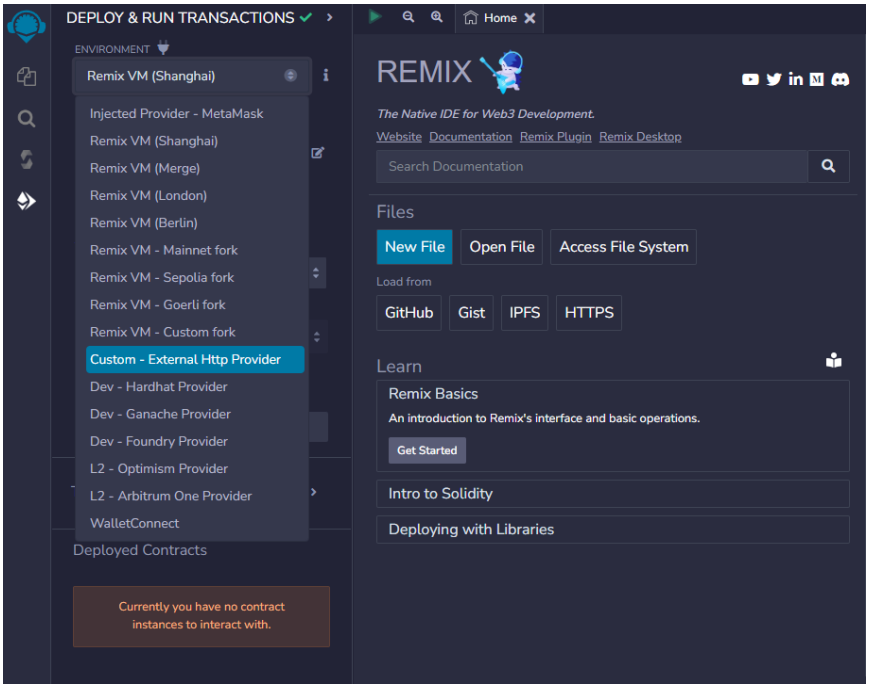


Figura 6.3: Paso 4 de instalación de GETH

1.5. Seguir la interfaz de Remix para el compilado y despliegue del contratro.

6.3. Conexión con la base de datos

Después de establecer el entorno de desarrollo, el siguiente paso fue conectar la plataforma con las bases de datos off-chain.

En cuanto a la base de datos off-chain, elegimos MySQL. Este sistema de gestión de bases de datos relacional es eficiente, fácil de usar y nos permitió almacenar información que no era crítica para la seguridad.

La gestión de la base de datos se ha realizado con XAMPP, y la conexión hacía ella la realiza cada controlador según qué función esté ejecutando gracias al archivo de configuración /database/**config.js.**

6.4 Creación de nuestro Smart Contract

Una vez que se estableció la conexión con las bases de datos, el siguiente paso fue la creación de nuestro Smart Contract. Este contrato, escrito en Solidity, forma la columna vertebral de la plataforma y se utiliza para gestionar las interacciones con la blockchain de Ethereum.

6.5 Implementación de la Interfaz de Usuario

El siguiente paso fue la creación de la interfaz de usuario, que se hizo utilizando HTML, CSS y JavaScript. Diseñamos la interfaz para ser intuitiva y fácil de usar, lo que permitiría a los usuarios interactuar con la plataforma sin tener un conocimiento técnico previo.

Una vez se terminó el diseño en formato html, utilizamos EJS, un motor de plantillas de JavaScript, para generar el HTML dinámicamente. Esto nos permitió presentar a los usuarios una vista personalizada de la plataforma basada en su rol y sus interacciones.

6.6 Implementación de los flujos de trabajo

Finalmente, implementamos los flujos de trabajo de la plataforma. Estos flujos de trabajo, que incluyen el flujo de trabajo del administrador, el flujo de trabajo del usuario y el flujo de trabajo del informe, se diseñaron para reflejar las interacciones típicas que los usuarios tendrían con la plataforma. Para cada flujo de trabajo, tuvimos que considerar las interacciones con la interfaz de usuario, la interacción con las bases de datos on-chain y off-chain, y las transacciones con los smart contracts.

Para cada acción en un flujo de trabajo, implementamos una serie de comprobaciones de seguridad para garantizar que solo los usuarios autorizados pudieran realizar acciones específicas. Por ejemplo, sólo un administrador podría dar de alta a un taller o a un usuario, y solo un usuario autorizado podría ver el informe de un vehículo.

A continuación se describen los distintos flujos de trabajo del sistema:

Register:

En la figura 6.4 se muestra la vista de registro de un usuario del sistema. Si un usuario no tiene una cuenta existente, el proceso de registro es el siguiente:

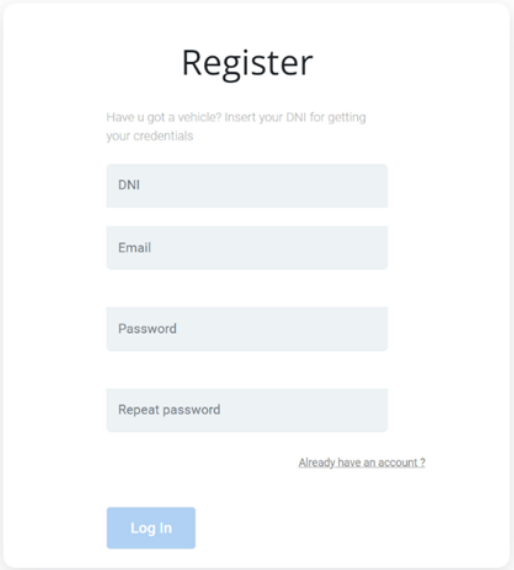


Figura 6.4: Vista de la sección de registro

1. **Ingreso de DNI, Email y Contraseña**: El usuario ingresa su DNI, su email y una contraseña en los campos correspondientes para crear una nueva cuenta.
2. **Verificación de los datos**: La aplicación verifica si el DNI y el email no están registrados previamente en la base de datos.

- Si los datos ya existen: Si el DNI o el email ya están registrados en la base de datos, se notifica al usuario y se le pide que proporcione diferentes datos.
- Si los datos son únicos: Si el DNI y el email son únicos, se continúa con el proceso.

3. **Confirmación de Registro y Almacenamiento de Datos**: La aplicación muestra un mensaje de confirmación al usuario, indicando que se ha registrado con éxito. Los detalles del usuario se almacenan de forma segura en la base de datos.

Login:

El proceso de login para un usuario registrado es el siguiente:

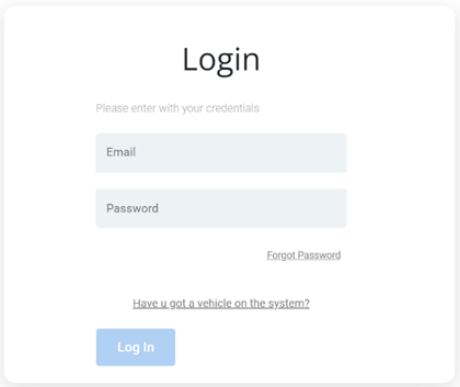


Figura 6.5: Vista de la sección de login

1. **Ingreso de Email y Contraseña**: El usuario ingresa su email y contraseña registrados en los campos correspondientes.
2. **Verificación de los datos**: La aplicación verifica si el email y la contraseña ingresados corresponden a una cuenta existente en la base de datos.

- Si los datos son incorrectos: Si el email o la contraseña son incorrectos, se notifica al usuario y se le invita a intentar ingresar de nuevo.
- Si los datos son correctos: Si los datos ingresados son correctos, se comprueba que clase de usuario es. Si es un usuario con el rol propietario será redirigido a su dashboard de perfil de usuario (Figura 6.15). En caso de ser un usuario con rol taller, será redirigido a la pantalla de gestión de su taller (Figura 6.11).

6.6.1 Flujo de trabajo del Administrador

Este flujo de trabajo implica la creación de talleres, la asignación de administradores de taller y la asignación de vehículos a los usuarios. Dado que nuestra aplicación es un prototipo de sistema, al necesitar de la inserción de información como vehículos y usuarios para la generación de su informe correspondiente, se ha considerado que el administrador pueda dar de alta tanto vehículos con rol propietario como vehículos asociados a los propietarios previamente creados.

En un caso real de despliegue de nuestra aplicación, esta se nutriría de datos oficiales de vehículos por parte de las entidades oficiales correspondientes y constaría de un sistema de verificación de usuarios para poder asignarles las credenciales a nuestro sistema, gestionar sus vehículos y ver informes.

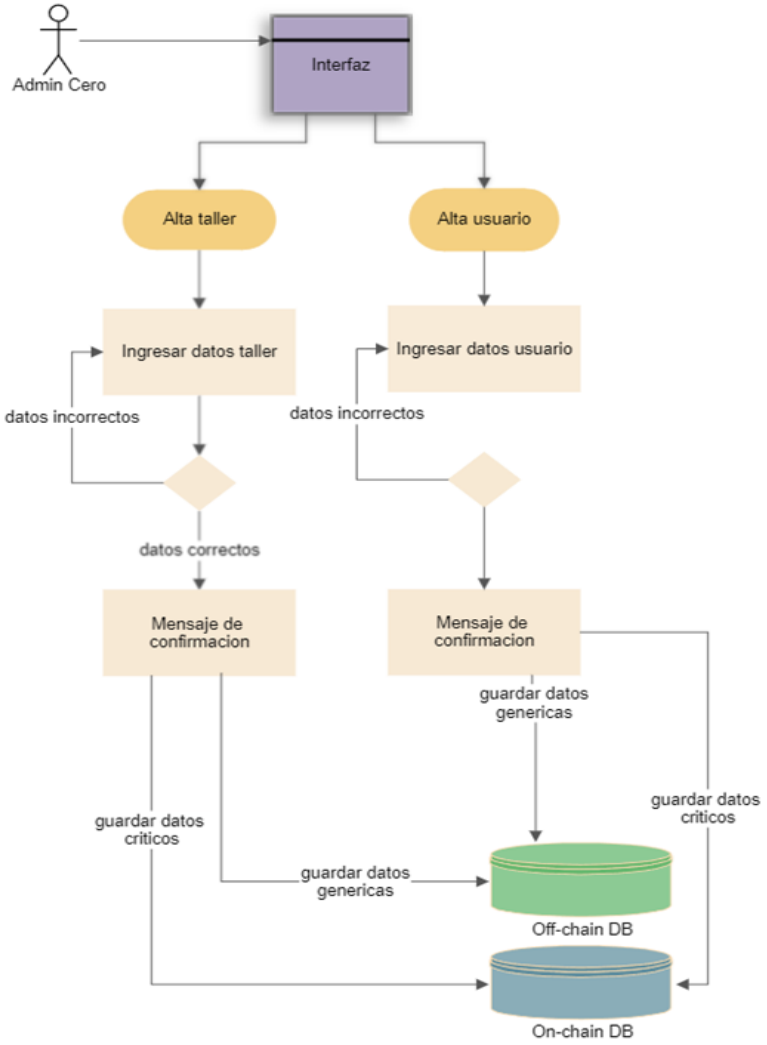


Figura 6.6: Diagrama de flujo de trabajo de la creación de talleres

Los flujos descritos en el diagrama, se explican a continuación:

Alta Usuario

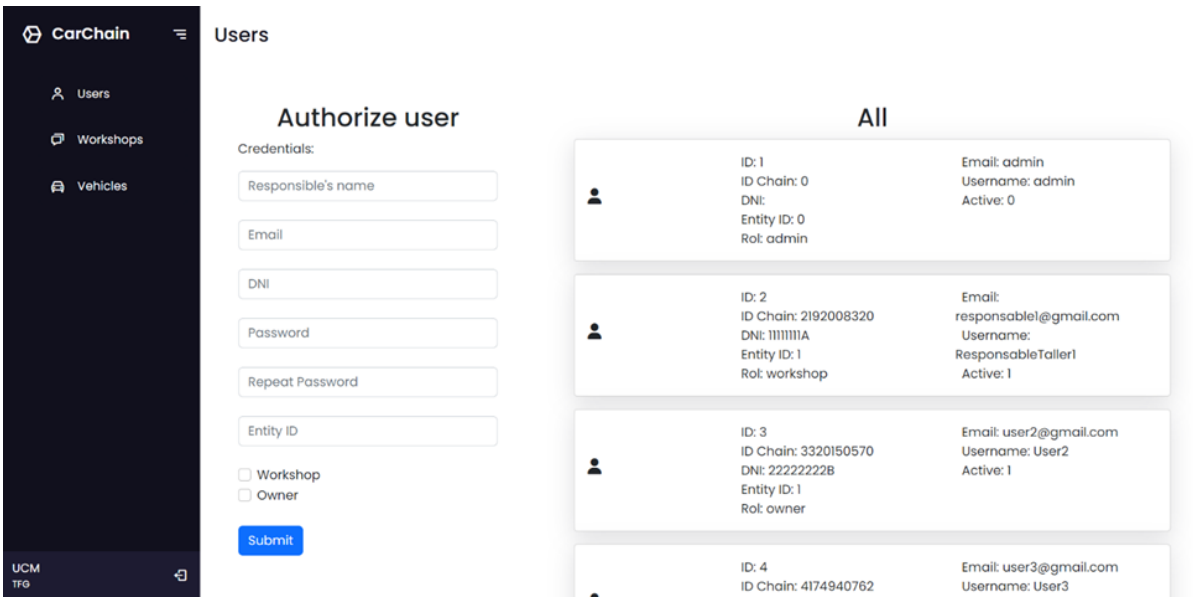


Figura 6.7: Vista de la sección de alta de usuarios

Alta Taller

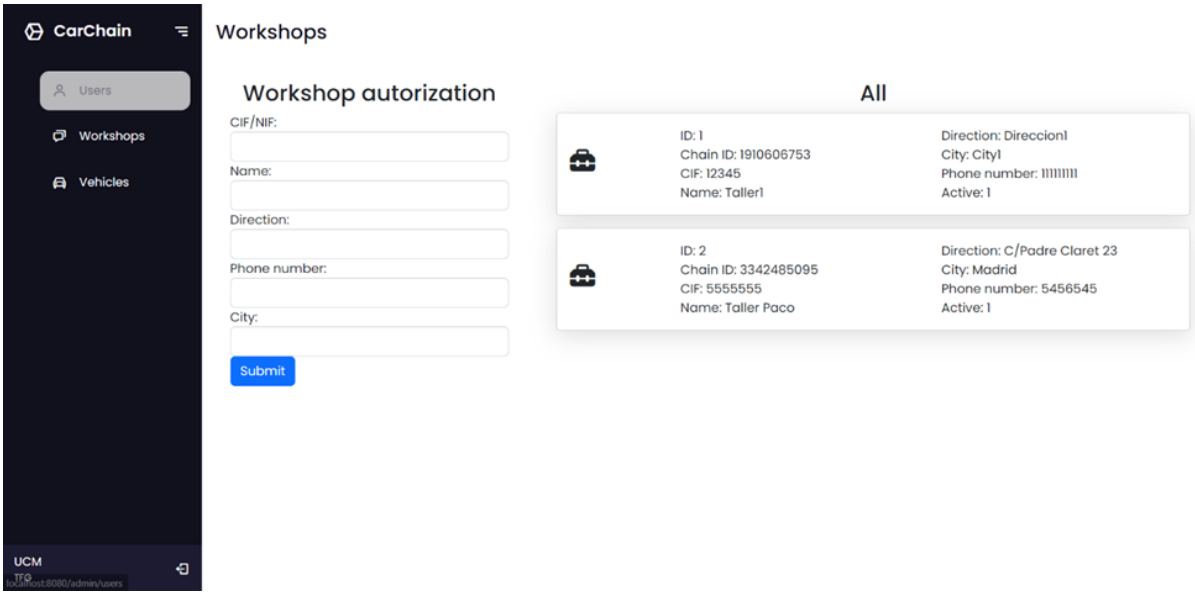


Figura 6.8: Vista de la sección de alta de talleres

Alta Vehiculo

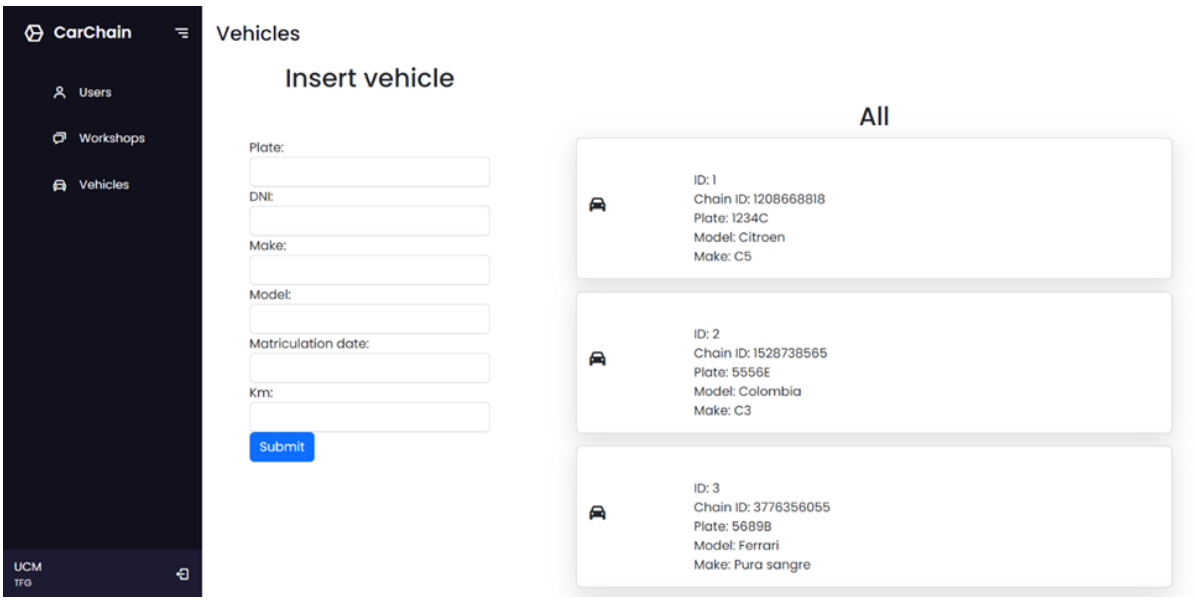


Figura 6.9: Vista de la sección de alta de vehículos

Alta Usuario / Alta Taller / Alta Vehículo

Estos tres flujos siguen un patrón similar de recolección de datos, validación y confirmación:

1. **Ingresar datos**: En cada uno de estos casos de uso, el usuario inicia el proceso ingresando la información relevante:
    - En el caso de "Alta Usuario", los datos son la información personal del usuario, como nombre, DNI, correo electrónico, etc.
    - En "Alta Taller", los datos son información sobre el taller, como nombre del taller, ubicación, etc.
    - En "Alta Vehículo", los datos son detalles del vehículo, como marca, modelo, año de matriculación, número de placa, etc.

2. **Comprobación de datos**: Una vez que se han introducido los datos, la aplicación comprueba si son correctos o no. Verificando la completitud de los datos, la validez y la integridad de los datos.
    - Si los datos son incorrectos, el usuario es redirigido para volver a ingresar los datos.
    - Si los datos son correctos, el proceso avanza al siguiente paso.
3. **Confirmación y almacenamiento de datos**: Cuando los datos han sido validados, la aplicación muestra un mensaje de confirmación al usuario.
    - Los datos críticos (por ejemplo, DNI para usuarios, ID del taller, número de matrícula del vehículo) se almacenan en una base de datos on-chain para asegurar la inmutabilidad y la seguridad.
    - Los datos generales o menos sensibles se almacenan en una base de datos off-chain, lo cual puede ser más eficiente y económico

6.6.2 Flujo de trabajo del Administrador de un taller

Este flujo de trabajo implica la adición de servicios o reparaciones a los vehículos registrados.

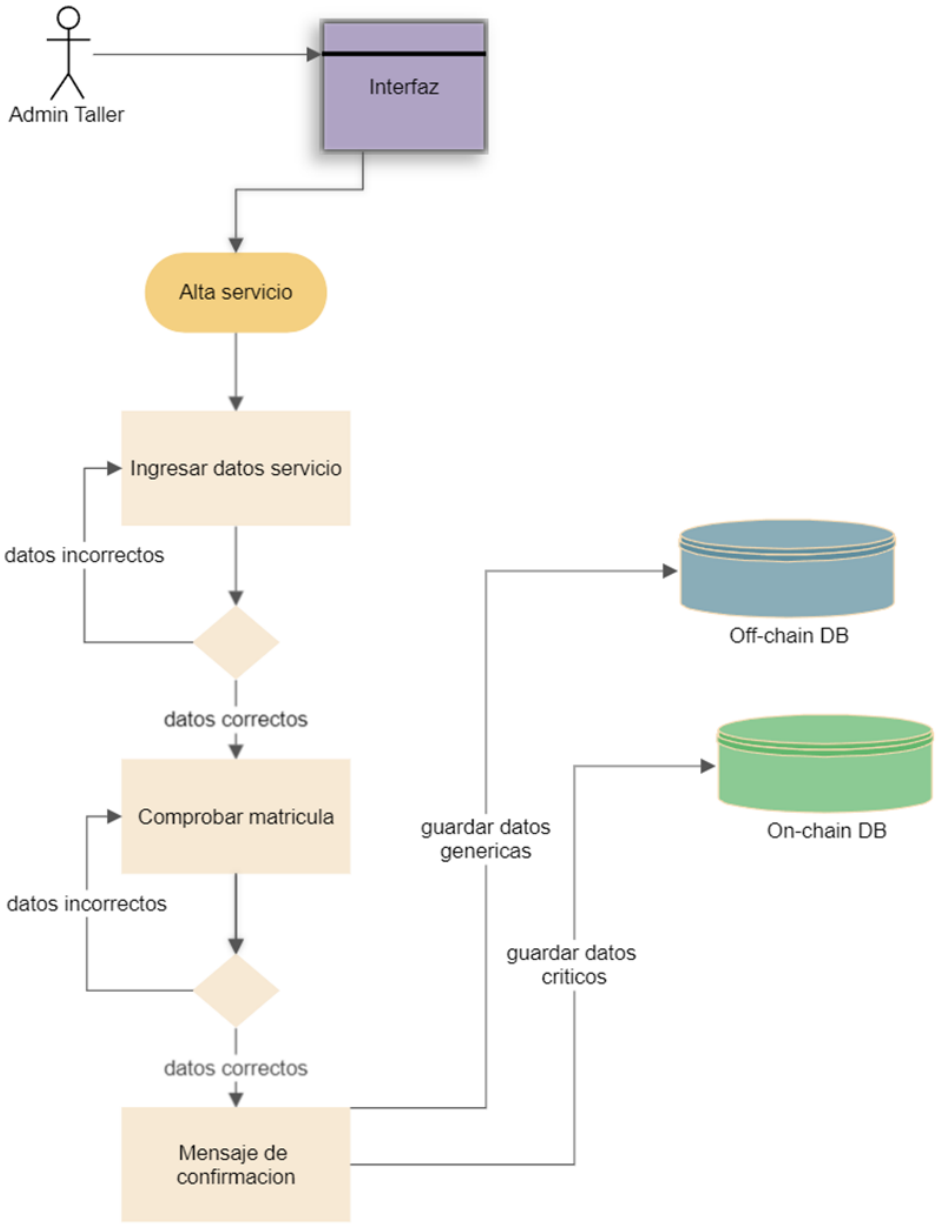


Figura 6.10: Diagrama de flujo de trabajo de alta de servicio por parte de taller

Los flujos descritos en el diagrama, se explican a continuación:

Alta Servicio

El flujo de "Alta Servicio" (6.11) es específico para el usuario "Admin Taller", que tiene permisos para añadir servicios de reparación o mantenimiento en el sistema para un taller específico.

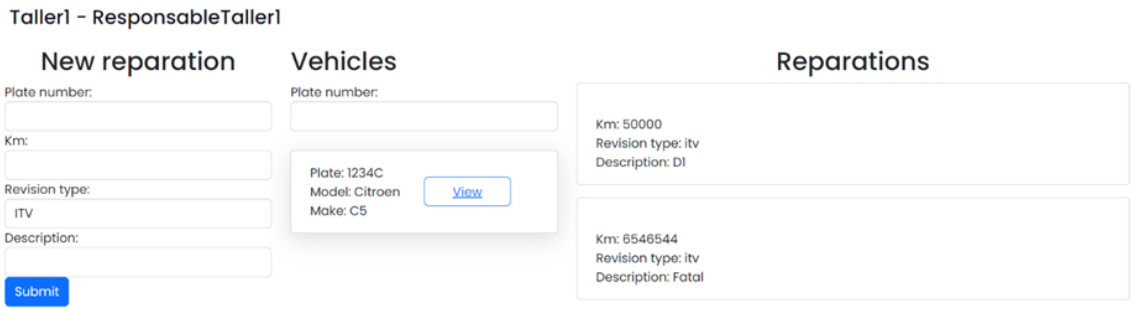


Figura 6.11: Vista de panel de administración de taller

Tiene el siguiente proceso:

1. **Ingresar datos del servicio**: El administrador del taller introduce la información del servicio, incluye detalles como la descripción del servicio, la matrícula del vehículo asociado, el kilometraje que tiene el vehículo en el momento de la intervención y el tipo de revisión.
2. **Comprobación de los datos**: Una vez que el administrador del taller ha ingresado los datos, la aplicación verifica si son correctos y completos. Comprueba la validez de los datos, la integridad de los datos y la

completitud de los datos. Además, como parte de este paso, la aplicación también verifica si la matrícula del vehículo ingresada existe en la base de datos.

- Si los datos son incorrectos o la matrícula del vehículo no existe, el administrador del taller es redirigido para volver a ingresar los datos del servicio.
- Si los datos son correctos y la matrícula del vehículo existe, el proceso avanza al siguiente paso.

3. **Confirmación y almacenamiento de los datos**: Cuando los datos del servicio han sido validados y la matrícula del vehículo ha sido confirmada, la aplicación muestra un mensaje de confirmación al administrador del taller.
    - Los datos críticos del servicio, como el identificador del servicio, la matrícula del vehículo se almacena en la base de datos on-chain.
    - Los datos generales o menos sensibles del servicio, como la descripción del servicio se almacenan en la base de datos off-chain.

Este proceso de "Alta Servicio" permite al administrador del taller agregar nuevos servicios al sistema de una manera eficiente y segura, garantizando al mismo tiempo la exactitud y la validez de los datos del servicio.

6.6.3 Flujo de trabajo para usuarios

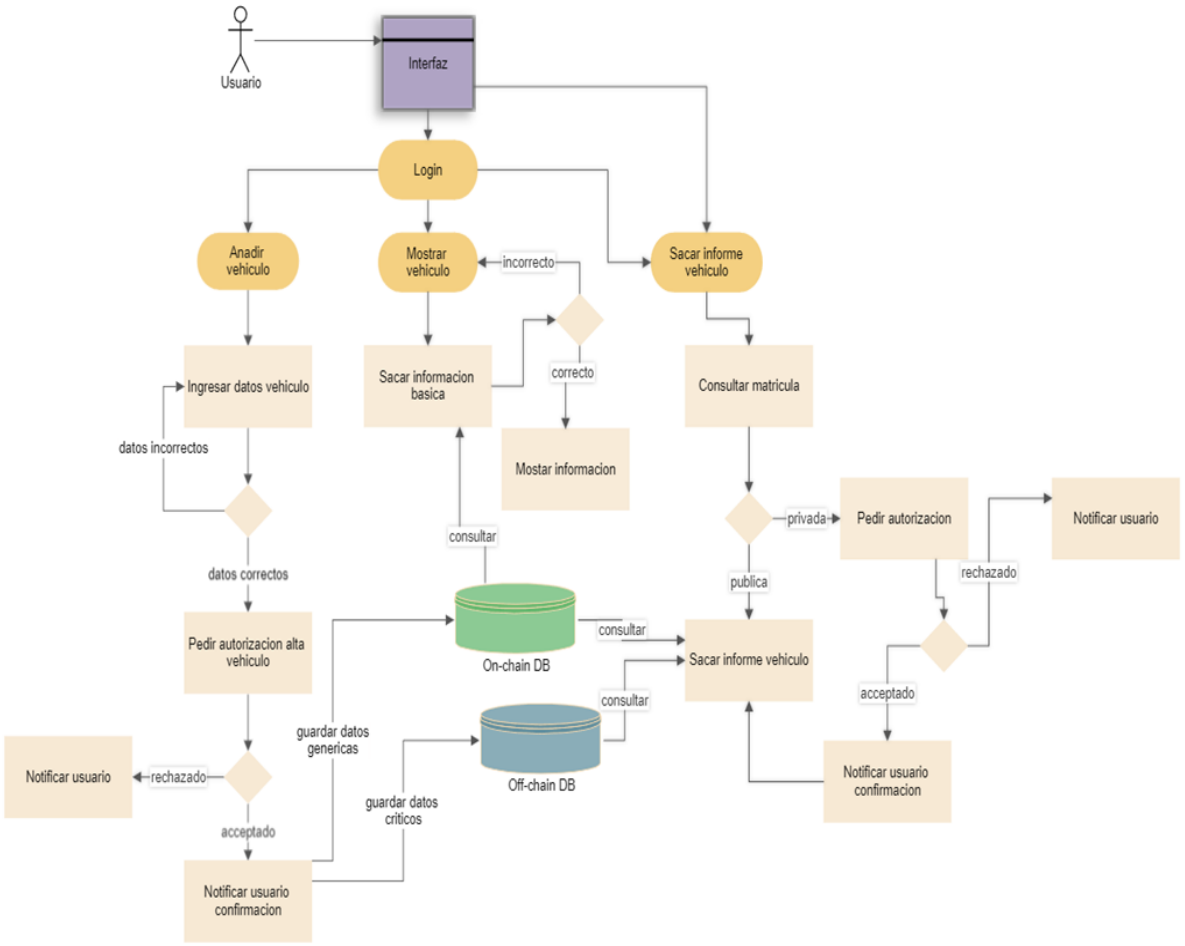


Figura 6.12: Diagrama de flujo de trabajo de los usuarios de la aplicación

En nuestra plataforma, los usuarios registrados tienen la capacidad de controlar la **visibilidad** de las matrículas de sus vehículos registrados. Esto significa que pueden elegir si la información de sus vehículos es pública y accesible para todos los demás usuarios, o privada y accesible solo con su autorización. Este nivel

de control permite a los usuarios mantener la privacidad de sus datos mientras facilita el intercambio de información cuando sea necesario.

Los flujos descritos en el diagrama, se explican a continuación:

Sacar Informe Vehículo

Esta funcionalidad permite a los usuarios obtener informes de vehículos en función de las preferencias de privacidad de los propietarios originales.

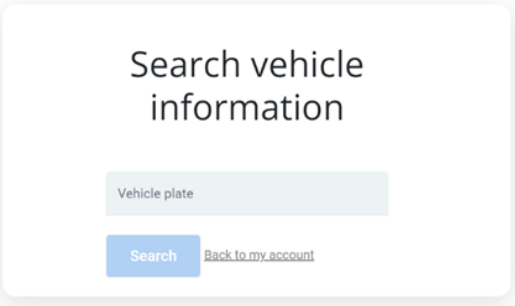


Figura 6.13: Pantalla de acceso al informe de vehículo

Una vez el usuario accede a la vista representada en la figura 6.13, los pasos que debe seguir para generar el informe del vehículo son los siguientes:

1. **Ingresar el número de matrícula de un vehículo**: Para iniciar el proceso de obtención de un informe de vehículo, el usuario ingresa el número de matrícula del vehículo del que desea obtener información.

2. **Comprobación de existencia de la matrícula**: Una vez que se ha ingresado el número de matrícula, la aplicación verifica si existe en la base de datos.
    - Si el número de matrícula no existe, se informa al usuario y se termina el proceso.
    - Si el número de matrícula existe, el proceso continúa al siguiente paso.

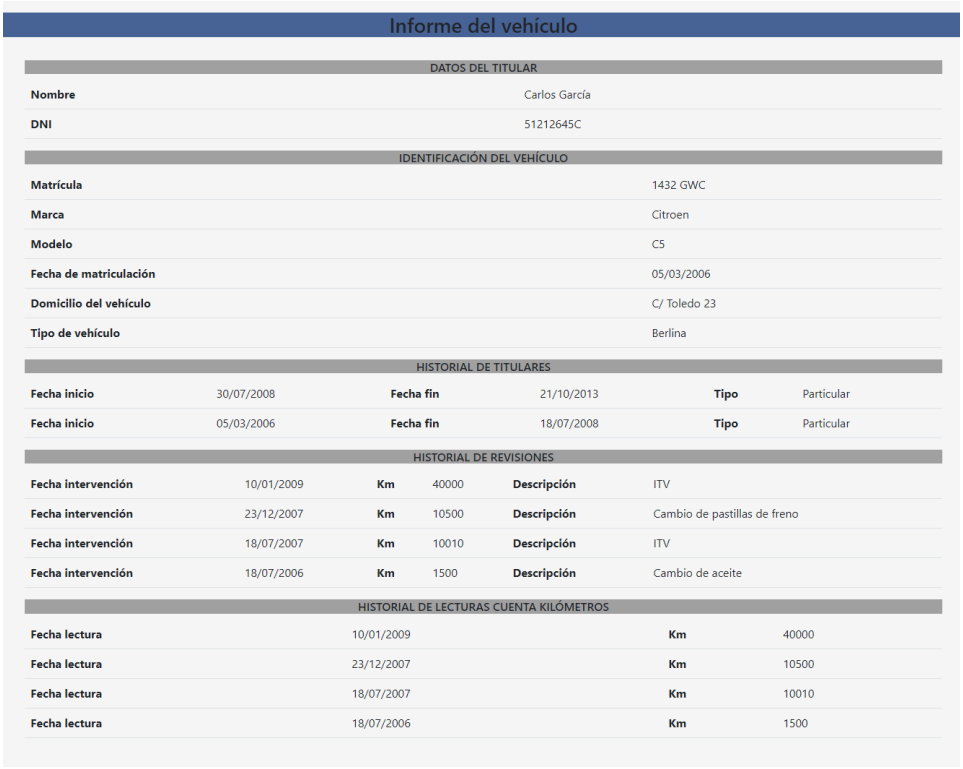
3. **Consulta de la visibilidad de la matrícula**: La aplicación verifica el estado de visibilidad de la matrícula del vehículo.

    - Si la matrícula está en estado "Pública", la aplicación accede a las bases de datos on-chain y off-chain para recopilar la información relevante y generar un informe del vehículo, que se proporciona al usuario.
    - Si la matrícula está en estado "Privada", se solicita la autorización del propietario original del vehículo a través de la aplicación.

4. **Gestión de la autorización**: Si se solicita la autorización, hay dos posibles resultados:
    - Si el propietario original acepta la solicitud, el usuario solicitante recibe una notificación de la aprobación. Entonces, la aplicación accede a las bases de datos y genera el informe del vehículo para el usuario solicitante.

- Si el propietario original rechaza la solicitud, el usuario solicitante recibe una notificación de la denegación, y no se le proporciona el informe del vehículo.

Este es un ejemplo de un informe sacado por la plataforma:



Informe del vehículo

DATOS DEL TITULAR	
Nombre	Carlos García
DNI	51212645C

IDENTIFICACIÓN DEL VEHÍCULO	
Matrícula	1432 GWC
Marca	Citroen
Modelo	C5
Fecha de matriculación	05/03/2006
Domicilio del vehículo	C/ Toledo 23
Tipo de vehículo	Berlina

HISTORIAL DE TITULARES					
Fecha inicio	30/07/2008	**Fecha fin**	21/10/2013	**Tipo**	Particular
Fecha inicio	05/03/2006	**Fecha fin**	18/07/2008	**Tipo**	Particular

HISTORIAL DE REVISIONES					
Fecha intervención	10/01/2009	**Km**	40000	**Descripción**	ITV
Fecha intervención	23/12/2007	**Km**	10500	**Descripción**	Cambio de pastillas de freno
Fecha intervención	18/07/2007	**Km**	10010	**Descripción**	ITV
Fecha intervención	18/07/2006	**Km**	1500	**Descripción**	Cambio de aceite

HISTORIAL DE LECTURAS CUENTA KILÓMETROS			
Fecha lectura	10/01/2009	**Km**	40000
Fecha lectura	23/12/2007	**Km**	10500
Fecha lectura	18/07/2007	**Km**	10010
Fecha lectura	18/07/2006	**Km**	1500

Figura 6.14: Informe de vehículo generado por la aplicación

Dashboard usuario

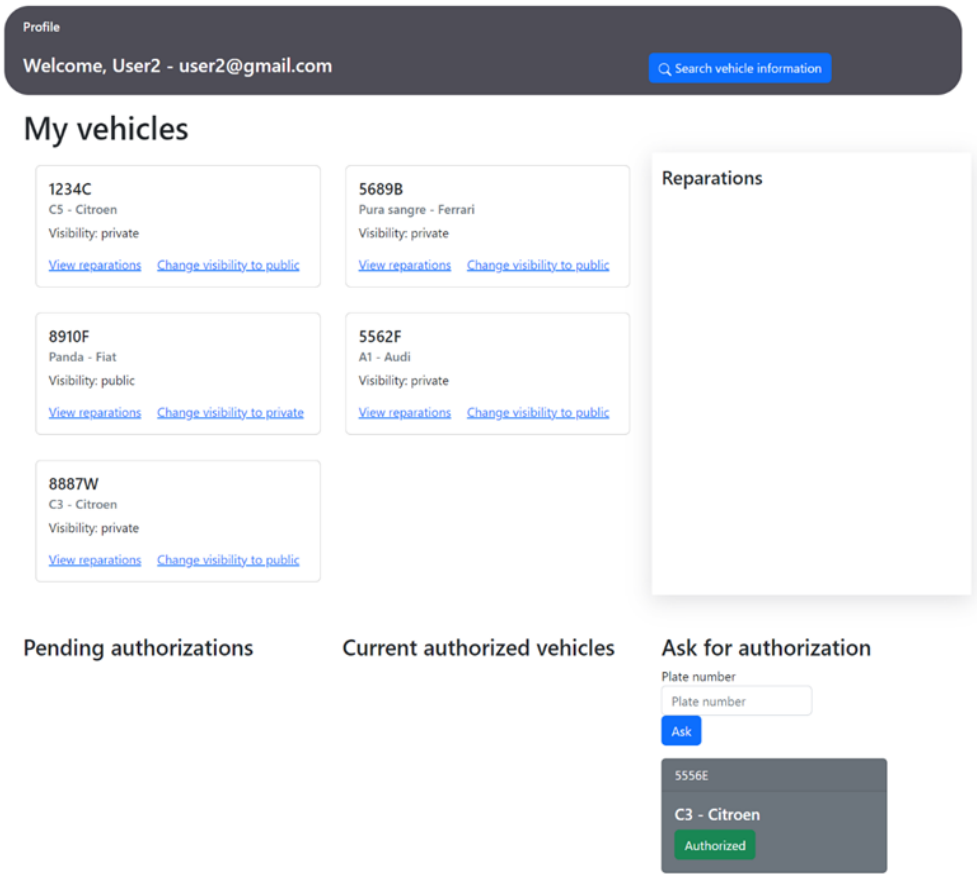


Figura 6.15: Vista del dashboard de un usuario

El panel de un usuario autorizado tiene varias funcionalidades, para ayudar a aclarar cada parte, lo dividiremos en dos partes durante esta explicación.

Mis vehículos:

En esta sección, los usuarios pueden ver sus vehículos registrados. Tendrán acceso a una función de "Mostrar Reparaciones" que muestra los servicios relacionados con cada vehículo a la derecha. Además, una función de "Cambiar privacidad" permite a los usuarios establecer si cada vehículo es público o privado, en función de la visibilidad otorgada el resto de usuarios podrá generar o no el informe del vehículo en cuestión.

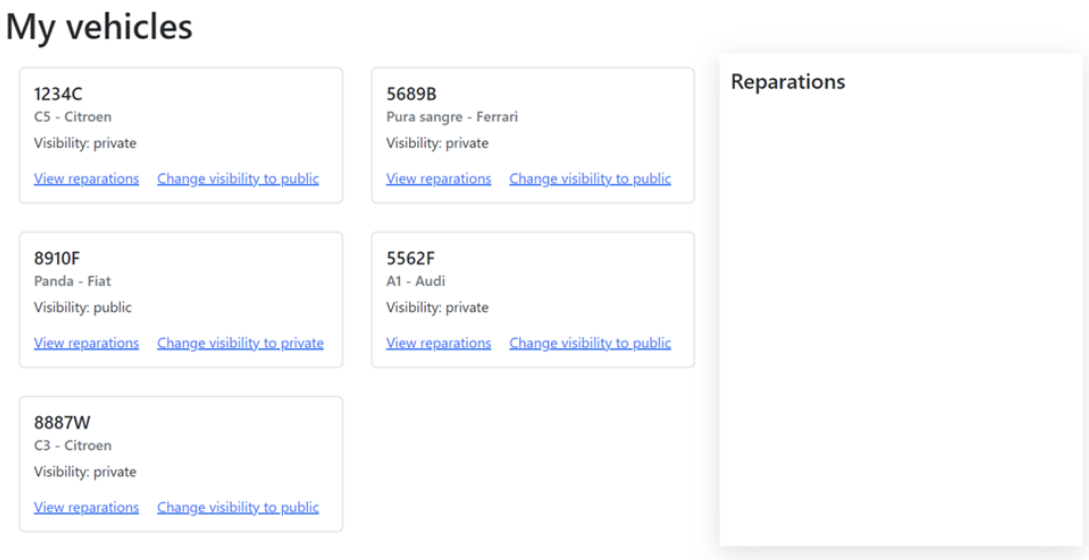


Figura 6.16: Vista del dashboard de un usuario - Sección de vehículos

Mostrar Reparaciones

Esta función permite a los usuarios visualizar una lista simplificada de la información relevante sobre las reparaciones de sus vehículos, como se puede observar en la figura 6.17. Para acceder a esta información, los usuarios simplemente necesitan pulsar en la opción "Mostrar Reparaciones". La aplicación entonces consulta la base de datos, recupera la información pertinente y la presenta de manera clara y concisa al usuario.

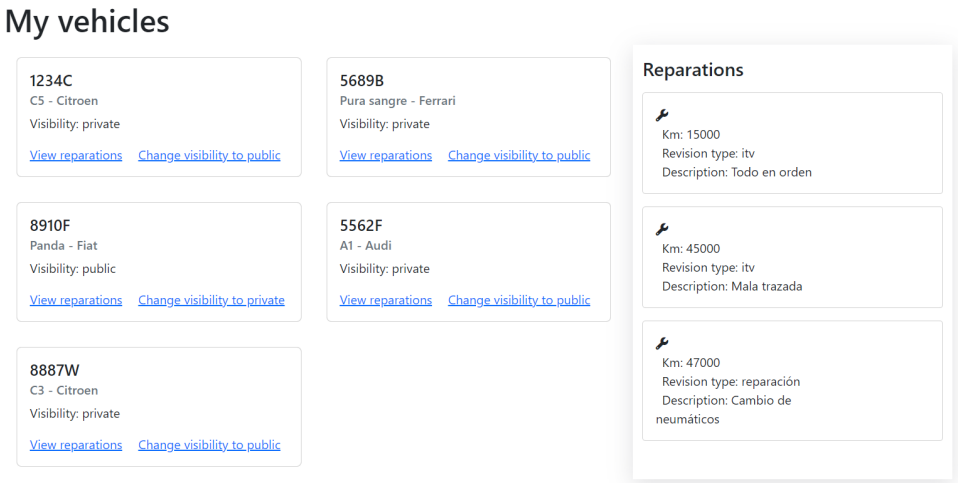


Figura 6.17: Vista del dashboard de un usuario - Reparaciones de un vehículo

Cambiar Privacidad (Pública o Privada)

La segunda función principal de la sección "Mis Vehículos" permite a los usuarios cambiar la visibilidad de la información de sus vehículos. Al pulsar en la opción "Cambiar Privacidad", los usuarios pueden alternar entre los estados "Público" y "Privado" para la matrícula de cada uno de sus vehículos.

- Si eligen "Público", otros usuarios pueden ver la información del vehículo y acceder a los informes sin necesidad de solicitar permiso.
- Si eligen "Privado", cualquier usuario que desee acceder a la información del vehículo debe solicitar autorización a través de la aplicación.

Autorizaciones:

En esta sección, los usuarios pueden enviar solicitudes de autorización y gestionarlas. También puede supervisar sus vehículos autorizados actualmente, incluyendo los usuarios designados para cada uno. Además, también pueden administrar de manera efectiva las solicitudes de autorización recibidas.

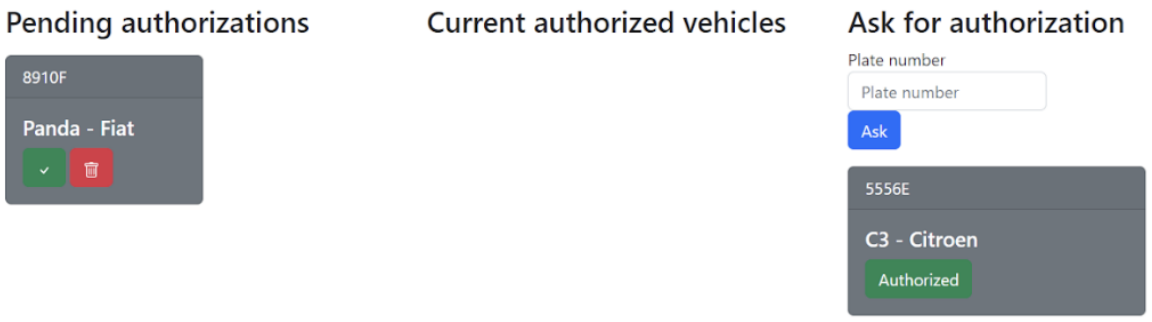


Figura 6.18: Vista del dashboard de un usuario - Sección de autorizaciones

Autorizaciones Pendientes

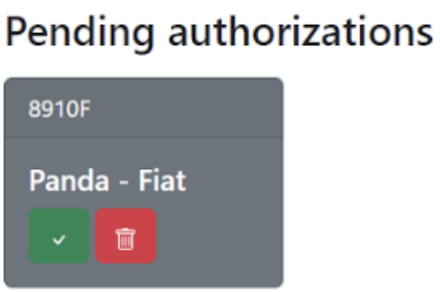


Figura 6.19: Vista del dashboard de un usuario - Sección de autorizaciones pendientes de aprobación

En "Autorizaciones Pendientes", los usuarios pueden ver las solicitudes de autorización que otros usuarios han enviado para acceder a la información de sus vehículos. Aquí, pueden revisar cada solicitud y tomar una decisión sobre si aceptar o rechazar cada una de ellas. Este control facilita el manejo de la privacidad y el acceso a los datos del vehículo, asegurando que la información solo se comparta con aquellos usuarios que el propietario original apruebe.

Vehículos Autorizados

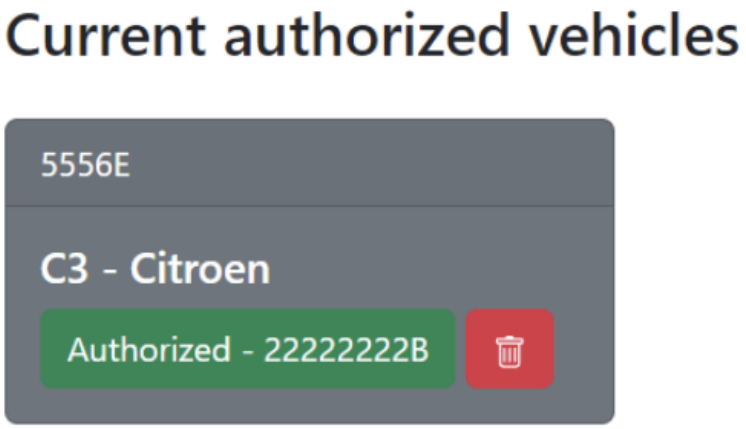


Figura 6.20: Vista del dashboard de un usuario - Sección de vehículos en los que ya ha sido autorizado el usuario

La sección "Vehículos Autorizados" proporciona una lista de los vehículos del usuario para los que se ha otorgado autorización a otros usuarios. Aquí, los usuarios pueden ver qué vehículos de su flota están autorizados y a quién se le ha concedido el acceso. Esta funcionalidad permite a los usuarios mantener un seguimiento constante de quién tiene acceso a la información de sus vehículos y realizar ajustes según sea necesario.

Pedir Autorización

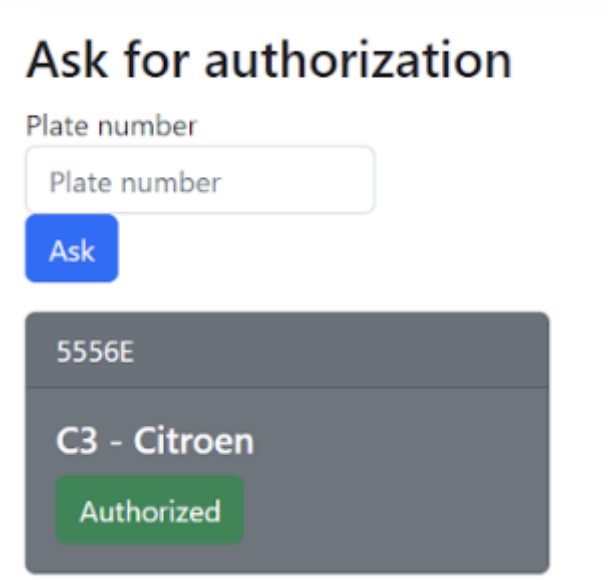


Figura 6.21: Vista del dashboard de un usuario - Sección para pedir autorización a un vehículo concreto y vista de los que ya han sido autorizados

Finalmente, en "Pedir Autorización", los usuarios pueden solicitar acceso a la información de un vehículo específico introduciendo el número de matrícula y pulsando "Pedir". Una vez enviada la solicitud, se mostrará el estado de la misma en esta sección. Los posibles estados son "Pendiente", "Aceptado" y "Rechazado". Esto permite a los usuarios seguir el progreso de sus solicitudes y conocer el resultado tan pronto como la decisión sea tomada por el propietario original del vehículo.

Capítulo 7 - Conclusiones y trabajo futuro

En este capítulo final, revisaremos las implicaciones a largo plazo y el alcance de nuestra aplicación. Se discutirá la viabilidad del sistema, incluyendo cómo se introducirá a la sociedad y su posible colaboración con las entidades relacionadas. Profundizaremos en los acuerdos con talleres autorizados y la incorporación de fábricas, concesionarios y seguros. También exploraremos posibles mejoras o ampliaciones de nuestra aplicación. Buscamos no solo cerrar la memoria en este capítulo, sino también abrir nuevas puertas para futuras investigaciones y desarrollos.

El código de nuestro prototipo de aplicación se puede ver y descargar a través del siguiente enlace: https://github.com/jaaavi/TFG-CarChain

7.1 Viabilidad del sistema en un contexto real

El blockchain, al ser una tecnología emergente, aún no está totalmente comprendida ni adoptada por la sociedad en general. Por ello, introducir una aplicación basada en blockchain para el sector automotriz supone un desafío significativo. Sin embargo, con un enfoque estratégico, esto se puede lograr de manera efectiva.

7.1.1 Introducción a la sociedad y colaboración con la DGT

Primero, es esencial educar al público sobre qué es el blockchain y cómo puede beneficiar al sector automotriz. Las campañas de información y concienciación pueden desempeñar un papel crucial en este aspecto. Se pueden realizar seminarios web, talleres, publicaciones en blogs y redes sociales, y distribuir folletos informativos en los talleres y concesionarios de coches. La clave es desmitificar el blockchain y presentarlo de una manera accesible y fácil de entender para el público en general.

Segundo, la aplicación es fácil de usar y ofrece una clara ventaja sobre los métodos existentes. Los usuarios son más propensos a adoptar una nueva tecnología si ven que mejora su vida de alguna manera. En este caso, la aplicación ofrece mayor transparencia, confiabilidad y seguridad en la información del vehículo, lo cual sería una mejora significativa respecto al sistema actual.

En cuanto a la competencia con la Dirección General de Tráfico (DGT), es importante señalar que, en lugar de ser una competencia directa, la aplicación puede considerarse como un complemento a los servicios ya ofrecidos por la DGT. Mientras que la DGT tiene la responsabilidad de mantener y regular el registro de vehículos en el país, la aplicación basada en blockchain podría servir como una capa adicional de verificación y transparencia.

Para trabajar eficazmente con la DGT, podría ser útil establecer una colaboración, en la que la DGT pueda reconocer la utilidad de la aplicación y posiblemente incluso promover su uso. Esto requerirá demostrar el valor y la seguridad de la aplicación, así como garantizar que cumple con todas las regulaciones y estándares pertinentes.

En última instancia, introducir la aplicación a la sociedad y colaborar con la DGT requerirá una combinación de educación pública, desarrollo de una aplicación fácil de usar y valiosa, y una colaboración efectiva con las entidades reguladoras. Con el enfoque adecuado, la adopción de la aplicación puede ser exitosa, lo que a su vez puede conducir a una mayor transparencia, confianza y eficiencia en el sector automotriz.

7.1.2 Acuerdos con talleres autorizados e incorporación de fábricas, concesionarios y seguros

Para que nuestra aplicación tenga éxito, será fundamental obtener la participación y colaboración de todas las partes interesadas en la industria del

automóvil. Esto incluye a los talleres autorizados, fábricas, concesionarios y empresas de seguros.

Los talleres autorizados desempeñan un papel crucial en el mantenimiento de los vehículos y, por ende, en la generación de datos confiables y verificables. Al incorporar estos talleres en la red blockchain, cada servicio o reparación realizado en un vehículo se registra de manera inmutable y transparente. Para lograr esto, será necesario establecer acuerdos con los talleres, lo que podría implicar demostrar los beneficios que nuestra aplicación puede ofrecer, como la posibilidad de aumentar la confianza de los clientes y mejorar la eficiencia de sus procesos.

Las fábricas y concesionarios también son actores clave. Las fábricas pueden registrar la información inicial de cada vehículo en la cadena de bloques, garantizando su autenticidad desde el momento en que el vehículo sale de la fábrica. Los concesionarios, por su parte, pueden utilizar la blockchain para verificar la información del vehículo en el momento de la venta, lo que puede aumentar la confianza de los compradores y facilitar las transacciones.

Por último, las empresas de seguros pueden beneficiarse enormemente de nuestra aplicación blockchain. Con el acceso a los datos del vehículo verificados y en tiempo real, las empresas de seguros pueden evaluar con mayor precisión el riesgo de cada póliza, lo que puede resultar en primas de seguro más justas y personalizadas. Además, en caso de accidente, la blockchain puede facilitar el proceso de reclamación al proporcionar un registro confiable de los servicios y reparaciones realizados en el vehículo.

Para incorporar todas estas entidades en el sistema, será fundamental establecer una comunicación clara y abierta, demostrar los beneficios que nuestra aplicación puede ofrecer y proporcionar el soporte necesario para su implementación. Además, será importante garantizar que el sistema cumple con

todas las regulaciones y estándares pertinentes, para proteger los derechos y la privacidad de los usuarios.

7.1.3 Posibles mejoras o ampliaciones de nuestra aplicación

Como parte de las posibles expansiones de nuestra aplicación basada en blockchain, hay que considerar la integración de una funcionalidad que permita la evaluación automática del cumplimiento de las recomendaciones del fabricante. Esta característica pretende mejorar la confiabilidad y transparencia de la información relativa al mantenimiento y reparación de los vehículos.

Las recomendaciones del fabricante son fundamentales para mantener un vehículo en óptimas condiciones y preservar su valor a lo largo del tiempo. Hasta ahora, ha sido un desafío para los propietarios de vehículos y las partes interesadas verificar si estas recomendaciones se han seguido estrictamente.

Con esta expansión, cada vez que se realice un servicio o reparación en un vehículo, los detalles se registrarán en la blockchain. Este registro incluirá la naturaleza del servicio o reparación, las piezas utilizadas, la fecha y la ubicación, así como cualquier otra información relevante. Una vez registrada, esta información se vuelve transparente e inmutable, es decir, no puede ser alterada o falsificada después de su registro.

Además, esta nueva funcionalidad permitirá comparar automáticamente los servicios y reparaciones realizados en un vehículo con las recomendaciones del fabricante. Si se identifica alguna discrepancia, la aplicación podría alertar al propietario del vehículo o a la parte interesada correspondiente, como una empresa de seguros que podría querer verificar si un vehículo ha sido mantenido correctamente antes de emitir una póliza.

Esta expansión no solo proporcionará una mayor transparencia y confiabilidad en la información de mantenimiento y reparación, sino que también

podría incentivar a los propietarios de vehículos a seguir las recomendaciones del fabricante. Un vehículo que ha sido mantenido correctamente puede tener un valor más alto y ser más atractivo para los compradores potenciales. Así, tanto los propietarios de vehículos como las partes interesadas en la industria del automóvil se beneficiarán de esta posible ampliación de nuestra aplicación.

7.2 Conclusiones

Este proyecto explora la posibilidad y el potencial de aplicar la tecnología blockchain en el sector automotriz, a través de nuestra aplicación. Hemos discutido cómo nuestra aplicación puede aumentar la transparencia, mejorar la eficiencia y fortalecer la confianza entre las partes interesadas al proporcionar un registro inmutable y verificable de la historia de un vehículo.

Las ventajas de nuestra aplicación son significativas. Primero, puede ayudar a reducir el fraude y la manipulación de datos en la industria automotriz al proporcionar un historial de mantenimiento y reparación seguro e inalterable. Esto no solo beneficia a los consumidores, sino también a los concesionarios, aseguradoras y talleres de reparación que se benefician de un sistema más transparente y confiable.

En segundo lugar, nuestra aplicación puede mejorar la eficiencia de los procesos de la industria automotriz. Al automatizar la verificación de información y eliminar la necesidad de intermediarios, podemos acelerar las transacciones y reducir los costos.

Además, nuestra aplicación tiene un gran potencial para crecer y expandirse. La inclusión de talleres autorizados y la integración con fabricantes, concesionarios y compañías de seguros pueden aumentar aún más el valor y la utilidad de nuestra aplicación. Asimismo, la posibilidad de evaluar automáticamente si un vehículo cumple con las recomendaciones del fabricante

es una expansión prometedora que podría incentivar a los propietarios de vehículos a mantener adecuadamente sus vehículos.

Sin embargo, la integración de nuestra aplicación en la sociedad no está exenta de desafíos. La adopción de una nueva tecnología siempre implica un cierto grado de resistencia, y la blockchain no es una excepción. Para superar esto, será crucial educar a los consumidores y a las partes interesadas sobre los beneficios de la blockchain y cómo nuestra aplicación la utiliza para mejorar la industria automotriz.

Además, el éxito de nuestra aplicación también depende de la cooperación de las partes interesadas. Necesitaremos trabajar en estrecha colaboración con las entidades de la industria automotriz, desde fabricantes y concesionarios, talleres de reparación y aseguradoras, hasta entidades reguladoras como la DGT para garantizar que nuestra aplicación se integre de manera efectiva en los procesos existentes.

Capítulo 8 - Conclusions and Future Work

In this final chapter, we will review the long-term implications and scope of our application. The feasibility of the system will be discussed, including its introduction to society and possible collaboration with related entities. We will delve into agreements with authorized workshops and the incorporation of factories, dealerships, and insurance companies. Additionally, we'll explore potential improvements or extensions of our application. Our goal is not only to conclude the thesis in this chapter, but also to open new doors for future research and development.

The code of our application prototype can be viewed and downloaded via the following link: https://github.com/jaaavi/TFG-CarChain

8.1 System viability in the real world

Blockchain, being an emerging technology, is not yet fully understood or adopted by society at large. Therefore, introducing a blockchain-based application for the automotive sector presents a significant challenge. However, with a strategic approach, this can be achieved effectively.

8.1.1 Introduction to society and collaboration with the DGT

First, it is essential to educate the public about what blockchain is and how it can benefit the automotive sector. Information and awareness campaigns can play a crucial role in this regard. You can hold webinars, workshops, blog and social media posts, and distribute informational brochures in garages and car dealerships. The key is to demystify the blockchain and present it in a way that is accessible and easy to understand for the public.

Second, the application is easy to use and offers a clear advantage over existing methods. Users are more likely to adopt a new technology if they see that it

improves their lives in some way. In this case, the application offers greater transparency, reliability and security in vehicle information, which would be a significant improvement over the current system.

Regarding the competition with the Directorate General of Traffic (DGT), it is important to note that, instead of being a direct competition, the application can be considered as a complement to the services already offered by the DGT. While the DGT has the responsibility of maintaining and regulating vehicle registration in the country, the blockchain-based application could serve as an additional layer of verification and transparency.

In order to work effectively with the DGT, it might be useful to establish a collaboration, in which the DGT can recognize the usefulness of the application and possibly even promote its use. This will require demonstrating the value and security of the app, as well as ensuring that it complies with all relevant regulations and standards.

Ultimately introducing the app to society and collaborating with the DGT will require a combination of public education, development of a user-friendly and valuable app, and effective collaboration with regulatory entities. With the right approach, app adoption can be successful, which in turn can lead to greater transparency, trust, and efficiency in the automotive sector.

8.1.2 Agreements with authorized workshops and incorporation of factories, dealers and insurance

In order for our application to be successful, it will be essential to obtain the participation and collaboration of all the interested parties in the automotive industry. This includes authorized workshops, factories, dealers and insurance companies.

Authorized workshops play a crucial role in vehicle maintenance and thus in generating reliable and verifiable data. By incorporating these workshops into the blockchain network, every service or repair performed on a vehicle is recorded immutably and transparently. To achieve this, it will be necessary to establish agreements with the workshops, which could involve demonstrating the benefits that our application can offer, such as the possibility of increasing customer confidence and improving the efficiency of their processes.

Factories and dealers are also key players. Factories can record the initial information of each vehicle on the blockchain, guaranteeing its authenticity from the moment the vehicle leaves the factory. Dealers, for their part, can use the blockchain to verify vehicle information at the time of sale, which can increase buyer confidence and make transactions easier.

Lastly, insurance companies can greatly benefit from our blockchain application. With access to real-time, verified vehicle data, insurance companies can more accurately assess the risk of each policy, which can result in fairer and more personalized insurance premiums. Furthermore, in the event of an accident, the blockchain can facilitate the claim process by providing a reliable record of the services and repairs performed on the vehicle.

To incorporate all these entities into the system, it will be essential to establish clear and open communication, demonstrate the benefits that our application can offer and provide the necessary support for its implementation. In addition, it will be important to ensure that the system complies with all relevant regulations and standards, in order to protect the rights and privacy of users.

8.1.3 Possible improvements or extensions of our application

As part of the possible expansions of our blockchain-based application, it is necessary to consider the integration of a functionality that allows the automatic evaluation of compliance with the manufacturer's recommendations. This feature aims to improve the reliability and transparency of information related to vehicle maintenance and repair.

The manufacturer's recommendations are essential to keep a vehicle in top condition and preserve its value over time. Until now, it has been a challenge for vehicle owners and stakeholders to verify whether these recommendations have been fully followed.

With this expansion, every time a service or repair is performed on a vehicle, the details will be recorded on the blockchain. This record will include the nature of the service or repair, the parts used, the date and location, as well as any other relevant information. Once registered, this information becomes transparent and immutable, that is, it cannot be altered or falsified after registration.

In addition, this new functionality will automatically compare the services and repairs performed on a vehicle with the manufacturer's recommendations. If any discrepancies are identified, the app could alert the vehicle owner or an appropriate interested party, such as an insurance company that might want to verify whether a vehicle has been properly maintained before issuing a policy.

This expansion will not only provide greater transparency and reliability in maintenance and repair information, but could also incentivize vehicle owners to follow manufacturer recommendations. A vehicle that has been properly maintained may have a higher value and be more attractive to potential buyers. Thus, both vehicle owners and stakeholders in the automotive industry will benefit from this possible extension of our application.

8.2 Conclusions

This project explores the possibility and potential of applying blockchain technology in the automotive sector, through our application. We have discussed how our app can increase transparency, improve efficiency, and strengthen trust among stakeholders by providing an immutable and verifiable record of a vehicle's history.

The advantages of our application are significant. First, it can help reduce fraud and data manipulation in the automotive industry by providing a secure and unalterable maintenance and repair history. This not only benefits consumers, but also dealers, insurers and repair shops who benefit from a more transparent and trustworthy system.

Secondly, our application can improve the efficiency of processes in the automotive industry. By automating the verification of information and eliminating the need for intermediaries, we can speed up transactions and reduce costs.

Also, our app has great potential to grow and expand. The inclusion of authorized workshops and the integration with manufacturers, dealers and insurance companies can further increase the value and usefulness of our application. Also, the ability to automatically assess whether a vehicle meets manufacturer recommendations is a promising expansion that could incentivize vehicle owners to properly maintain their vehicles.

However, integrating our app into society is not without its challenges. The adoption of a new technology always comes with a certain degree of resistance, and the blockchain is no exception. To overcome this, it will be crucial to educate consumers and stakeholders on the benefits of blockchain and how our app uses it to improve the automotive industry.

In addition, the success of our application also depends on the cooperation of the interested parties. We will need to work closely with entities in the automotive industry, from manufacturers and dealerships to repair shops and insurers, to ensure our app integrates effectively into existing processes.

Capítulo 9 - Contribuciones personales

9.1. Haroon Sammaraie Salih

- **Investigación**

Estuve implicado en una serie de actividades de recopilación de datos y análisis, con el fin de entender el contexto y las necesidades de nuestro proyecto. Revisé la literatura existente, estudié tendencias emergentes y evalué las tecnologías relevantes.

- **Diseño**

Utilicé la información obtenida de la investigación para crear bocetos y propuestas de diseño preliminares. Me enfoqué en crear diseños que no sólo fueran estéticamente agradables, sino también funcionales y fácilmente implementables.

- **Redacción de la Memoria**

La redacción de la memoria del proyecto fue una tarea que asumí con mucha responsabilidad. Mi objetivo era garantizar que todo el trabajo realizado se documentara de manera precisa y completa. Trabajé en la elaboración de secciones detalladas que explicaban cada etapa del proyecto, desde la investigación hasta el diseño y la implementación. Este proceso aseguró que todas nuestras experiencias, desafíos y logros quedaran correctamente registrados para su revisión futura.

- **Diagramas**

Finalmente, contribuí en la creación de varios diagramas importantes que se utilizan en todo el proyecto. Mis diagramas ayudaron a visualizar conceptos complejos, flujos de trabajo y estructuras. La habilidad para convertir ideas abstractas en representaciones gráficas claras y concisas nos ayudó para el éxito de esta etapa.

9.2. Javier García Suárez

- **Investigación**

He profundizado en el uso de tecnologías blockchain para la integración en un proyecto y tuve que realizar una extensa investigación sobre cómo poder integrar Node.js con la ejecución de un smart contract en Solidity. En un primer momento, desarrollé un sistema que compilaba y desplegaba el contrato en una red de prueba local sin necesidad de estar corriendo un nodo de Ethereum en el sistema. Este sistema consistía de un método singleton que utilizaba la librería ganache para desplegar el contrato y poder utilizarlo si previamente no estaba instanciado, si ya estaba instanciado lo comprobaba y lo usaba directamente. Finalmente, investigué la idea de correr un nodo de Ethereum, como se ha explicado anteriormente en la memoria, para realizarlo de una forma más real a lo que sería un despliegue en producción en una blockchain privada en la red de Ethereum.

- **Creación de las vistas de la aplicación**

Inicialmente, diseñé las vistas de la aplicación en formato HTML, aplicando estilo mediante CSS, y generando contenido de ejemplo para asegurar una correcta maquetación. Una vez completadas, las transformé en plantillas EJS para permitir la generación de contenido dinámico.

- **Configuración del servidor**

Al desarrollar este código, opté por estructurarlo en forma de clase para aprovechar la modularidad y organización que proporciona este enfoque. La clase "Server", del archivo "/models/server.js", encapsula la lógica del servidor, con métodos separados para configurar los middlewares y las rutas. Esto permite organizar y separar la lógica de manejo de cada ruta en archivos individuales, lo cual facilita la comprensión y el mantenimiento del código. Finalmente, implementé el método "listen()" que inicia el servidor y lo pone en funcionamiento en el puerto especificado. Esto permite que el servidor empiece a escuchar las solicitudes entrantes y responda en consecuencia.

- **Desarrollo de controladores y rutas**

Después de discutir y aclarar las ideas junto a mi compañero, procedimos a finalizar los diseños correspondientes de los diagramas de flujo de la aplicación. Posteriormente, me encargué de diseñar y desarrollar la lógica necesaria para llevar a cabo dichos diagramas de flujo. Creando la funcionalidad necesaria para poder conectar la base de datos con la blockchain de la forma más eficiente posible.

- **Redacción de la memoria**

Durante el desarrollo de la memoria, tuve una participación activa y desempeñé un papel crucial en su redacción. Desde el inicio, me encargué de planificar la estructura y el enfoque general, sobre todo en la parte de redacción de la arquitectura e implementación de la plataforma.

- **Revisión de memoria y código fuente**

Una vez desarrollada la base de la memoria, me enfoqué en pulir y mejorar el contenido de la memoria, asegurando que su narrativa fuera coherente y clara, y reestructurando las secciones cuando fue necesario para mejorar su fluidez. Paralelamente, también revisé el código fuente en profundidad, corrigiendo errores, optimizando secciones y garantizando su correcta funcionalidad a través de pruebas rigurosas. Mi objetivo en todo momento fue maximizar la calidad del proyecto y facilitar su entendimiento, y estoy convencido de que mis esfuerzos han contribuido significativamente a la excelencia final del proyecto.

Capítulo 10 - Referencias

1. **Blockchain privada**
    - https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/#:~:text=Public%20blockchains%20allow%20anyone%20to,the%20administrators%20to%20do%20so.
2. **Seguridad en el blockchain**
    - https://originstamp.com/blog/what-makes-the-blockchain-secure/
3. **Articulos contratos inteligentes**
    - https://www.investopedia.com/terms/s/smart-contracts.asp
    - https://www.simplilearn.com/tutorials/blockchain-tutorial/what-is-smart-contract
    - https://www.techtarget.com/searchcio/feature/Examples-of-smart-contracts-on-blockchain
4. **Vyper**
    - https://docs.vyperlang.org/en/stable/
5. **Chaincode**
    - https://hyperledger-fabric.readthedocs.io/en/release-1.3/chaincode.html
6. **Solidity**
    - https://solidity.readthedocs.io/
    - https://www.geeksforgeeks.org/introduction-to-solidity/
    - https://www.alchemy.com/overviews/solidity-smart-contract
7. **DGT**
    - https://www.dgt.es/nuestros-servicios/tu-vehiculo/tus-vehiculos/libro-electronico-de-mantenimiento/
8. **Carfax**

- https://www.carfax.eu/es/comprobar-bastidor-vehiculo
- https://www.carfax.eu/es

9. **Autocheck**
    - https://www.autocheck.com/vehiclehistory/
    - https://www.autocheck.com/vehiclehistory/sample-vehicle-history-report
10. **XCEED**
    - https://www.renaultgroup.com/en/news-on-air/news/xceed-a-new-blockchain-solution-for-renault-plants-in-europe/
11. **Articulos blockchain en aerolinea**
    - https://www.leewayhertz.com/blockchain-aviation-better-transparency-trust/
    - https://www.iata.org/contentassets/81005748740046de878439e6c54f2355/d1-1630-1700-blockchain-tech-for-mtc-and-engineering_stxaero.pdf
12. **Remix IDE Documentacion**
    - https://remix-ide.readthedocs.io/en/latest/
    - https://github.com/ethereum/remix-project
    - https://www.youtube.com/watch?v=sqN3PzPv1mM
13. **Geth**
    - https://geth.ethereum.org/
    - https://geth.ethereum.org/docs/getting-started
14. **Documentación oficial de Ethereum:**
    - https://ethereum.org/
15. **Documentación oficial de Web3.js:**
    - https://web3js.readthedocs.io/
16. **Artículo de introducción a Ethereum:**

- https://www.investopedia.com/terms/e/ethereum.asp

17. Tutorial de Web3.js y Ethereum:

- https://www.dappuniversity.com/articles/web3-js-intro