NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

JOSÉ MIGUEL PEREIRA MURTA

Bachelor in Computer Science and Engineering

TRUSTWORTHY CLASSIC CAR RESTORATION'S HISTORY USING BLOCKCHAIN

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon

February, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

TRUSTWORTHY CLASSIC CAR RESTORATION'S HISTORY USING BLOCKCHAIN

JOSÉ MIGUEL PEREIRA MURTA

Bachelor in Computer Science and Engineering

Adviser: Vasco Miguel Moreira do Amaral
Associate Professor, FCT-NOVA University Lisbon

Co-adviser: Fernando Brito e Abreu
Associate Professor, ISCTE University Institute of Lisbon

Examination Committee

Chair: João Ricardo Viegas da Costa Seco
Associate Professor, FCT-NOVA University Lisbon

Rapporteur: Paulo Rupino da Cunha
Associate Professor, FCTUC

Adviser: Vasco Miguel Moreira do Amaral
Associate Professor, FCT-NOVA University Lisbon

Co-adviser: Fernando Brito e Abreu
Associate Professor, ISCTE University Institute of Lisbon

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
February, 2024

Trustworthy Classic Car Restoration's History using Blockchain

Copyright © José Miguel Pereira Murta, NOVA School of Science and Technology, NOVA University Lisbon.
The NOVA School of Science and Technology and the NOVA University Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.5) [1].

Acknowledgements

I would like to start by expressing my profound gratitude to both my advisors, Prof. Vasco Amaral and Prof. Fernando Brito e Abreu, whose constant availability, support, and guidance were essential throughout the entire duration of this dissertation. Their consistent encouragement, weekly meetings, and insightful guidance not only shaped the progress of this work but also steered me toward the most correct path, by discussing several topics and potential improvements to the project. Their unwavering motivation and enthusiasm towards this project inspired me to bring the most value from my efforts. This work would not be possible without them.

I would also like to thank the support offered by all the team at Raimundo Branco workshop for their generous support. Their kindness and readiness to assist provided me with insights that significantly enriched the development of this work.

Special appreciation goes to my family, close friends, and girlfriend, who provided moments of joy and unwavering support during the most challenging times. Their motivation and encouragement inspired me to strive for the best possible results.

Throughout my journey at FCT-NOVA, I had the opportunity to meet inspiring people, including colleagues and professors, who consistently motivated and assisted me during the most difficult moments. With them, this arduous path became less daunting and more enjoyable and fruitful. I am sure that the friendships forged during this time will be cherished for a lifetime, and I take great pride in these meaningful connections.

This work was also produced with the support of INCD funded by FCT and FEDER under the project 01/SAICT/2016 nº 022153, and partially supported by NOVA LINCS (FCT UIDB/04516/2020), and ISTAR-Iscte (FCT UIDB/04466/2020 and UIDP/04466/2020).

Abstract

Classic automobiles have played a pivotal role in shaping the automotive industry and have special recognition worldwide. These vehicles are much more than simply old cars or means of transportation. They represent the cultural, historical, and technological accomplishments of specific eras and can be considered masterpieces, often holding significant financial value. However, to be considered true works of art, vintage cars must be conserved in mint condition or properly restored following strict guidelines and expert restoration services, accompanied by meticulous documentation, encompassing all pertinent information and other relevant details.

For this reason, all the relevant information respective to a classic car, such as fundamental details, technical documentation, certification information, and especially data about restoration or conservation processes, containing information from each performed step on these procedures, and other important documentation records must be adequately recorded to guarantee that they are not possible to be tampered or suffer any unauthorized modifications, allowing it to be shared with authorized entities.

This research addresses the mentioned challenge by proposing a blockchain-based decentralized solution. Comprising a web application utilizing Hyperledger Fabric blockchain network features, this solution ensures the secure and trustworthy recording of classic cars' information, restoration procedures, and associated documentation. It guarantees data immutability, traceability, and transparency, while also promoting data sharing and collaboration among diverse stakeholders.

The research and resulting solution underwent thorough validation, garnering positive feedback and interest from various entities in the classic vehicles sector. It is anticipated that this solution will contribute to the digital transformation of the classic automotive domain, enhancing credibility and trustworthiness through improved documentation. This enhanced documentation can also increase the financial value of classic cars and aid in elevating their status to that of masterpieces.

Keywords: Blockchain, Smart Contracts, Classic Car Documentation, Auto Repair Shop Software, Software Engineering

Resumo

Os automóveis clássicos desempenharam um papel fundamental na definição da indústria automóvel e têm um reconhecimento especial a nível mundial. Estes veículos são muito mais do que carros antigos ou meios de transporte. Representam realizações culturais, históricas e tecnológicas de épocas especificas e podem ser considerados obras-primas, frequentemente associadas a um valor financeiro significativo. No entanto, para serem considerados verdadeiras obras de arte, os carros clássicos devem estar conservados em perfeitas condições ou devidamente restaurados seguindo diretrizes rigorosas e serviços de restauro especializados, acompanhados de documentação minuciosa, contendo todas as informações pertinentes e outros detalhes relevantes.

Por este motivo, toda a informação relevante relativa a um automóvel clássico, tais como detalhes fundamentais, documentação técnica, informações de certificação e, especialmente, dados sobre processos de restauro, contendo informações de cada etapa realizada nestes procedimentos, e outros registos importantes devem ser devidamente registados para garantir que não são passíveis de ser adulterados ou sofrer quaisquer modificações não autorizadas, permitindo a sua partilha com entidades autorizadas.

Esta pesquisa aborda o desafio mencionado propondo uma solução descentralizada baseada em blockchain. Composta por uma aplicação web que utiliza as funcionalidades de uma rede blockchain Hyperledger Fabric, esta solução garante o registo seguro e confiável de informação de carros clássicos, procedimentos de restauro e documentação associada. Garante a imutabilidade, rastreabilidade e transparência de dados, promovendo a partilha de informações e a colaboração entre diversas partes interessadas.

A pesquisa e solução resultante foram submetidas a uma validação rigorosa, recebendo feedback positivo e suscitando interesse de várias entidades no setor de veículos clássicos. Antecipa-se que esta solução contribuirá para a transformação digital da indústria de automóveis clássicos, reforçando a credibilidade e confiabilidade por meio de uma documentação aprimorada. Esta documentação melhorada pode, adicionalmente, aumentar o valor financeiro dos carros clássicos e elevar o seu estatuto ao de obras-primas.

Palavras-chave: Blockchain, Contratos Inteligentes, Documentação de Carros Clássicos, Software de Oficinas, Engenharia de Software

Contents

List of Figures viii

List of Tables x

Acronyms xi

1 Introduction 1
- 1.1 Context and Description . . . 1
- 1.2 Institutional Context . . . 2
- 1.3 Challenges . . . 2
- 1.4 Problem Statement and Final Goals . . . 3
- 1.5 Contributions . . . 4
- 1.6 Scientific Methodology . . . 5
- 1.7 Document Structure . . . 6

2 Background 9
- 2.1 Classic Car Restoration Workshop Digital Transformation Project . . . 9
- 2.2 Blockchain Technology . . . 12
  - 2.2.1 Blockchain Network Types . . . 15
  - 2.2.2 Blockchain Architecture . . . 16
  - 2.2.3 Consensus Protocols . . . 18
  - 2.2.4 Smart Contracts . . . 20
- 2.3 Blockchain Platforms . . . 21
  - 2.3.1 Bitcoin . . . 21
  - 2.3.2 Ethereum . . . 22
  - 2.3.3 Hyperledger Fabric . . . 22
  - 2.3.4 Blockchain Platforms Comparison . . . 25

3 State-of-the-Art 27
- 3.1 Rapid Review . . . 27

3.1.1 Planning the Review . . . 28
3.1.2 Conducting the Review . . . 30
3.2 Generic Primary Studies . . . 33
3.3 Blockchain-based Vehicle Documentation Solutions . . . 35
3.3.1 Grey Literature . . . 35
3.3.2 Formal Scientific Literature . . . 37
3.4 Summary of the Research . . . 40

4 System Proposal **41**
4.1 Software Requirements Engineering . . . 41
4.1.1 Stakeholders . . . 41
4.1.2 Functional Requirements . . . 42
4.1.3 Non-functional Requirements . . . 46
4.2 Architecture . . . 46
4.2.1 Components . . . 47
4.2.2 Technologies . . . 49
4.2.3 Deployment Diagram . . . 54

5 Implementation **55**
5.1 ClassicsChain . . . 55
5.1.1 Blockchain Network . . . 55
5.1.2 Smart Contracts . . . 62
5.1.3 Fabric Application and API . . . 68
5.1.4 User Interface . . . 76
5.2 Charter of Turin Monitor . . . 85
5.2.1 Enhancements and Additional Features . . . 86
5.2.2 Integration with *ClassicsChain* . . . 88

6 Verification and Validation **91**
6.1 Overview . . . 91
6.2 Performance Evaluation . . . 92
6.2.1 Blockchain Network Evaluation . . . 92
6.2.2 Fabric Application and API Evaluation . . . 94
6.3 Relevance and Usability Evaluation . . . 99
6.3.1 Results . . . 102
6.3.2 Validation Threats . . . 105
6.4 Communication . . . 106

7 Conclusions **107**
7.1 Conclusions . . . 107
7.2 Future Work . . . 109

Bibliography 111

Appendices

A Web Application API 117

B Questionnaire 121

List of Figures

1.1 Design Science Research Process Model. Taken from [8]. 5

2.1 Example of one restoration procedure Business Process Model and Notation (BPMN) from the *Charter of Turin Monitor*. Taken from [5]. 10
2.2 Example of one of the sections of the PDF file. Taken from [5]. 11
2.3 Example of the platform's activities live map. Taken from [6]. 11
2.4 Example of a video conference. Taken from [5]. 12
2.5 Example of blockchain structure. 17

3.1 Diagram of the studies obtained through our literature review. 33
3.2 The life cycle of a vehicle in the system proposed in [38]. 38

4.1 Use Case Diagram of the Blockchain-based Documentation System 43
4.2 Component Diagram of our system, based on the component diagram in [5]. The colored components and artifacts identify the work developed in the current research. 48
4.3 Deployment Diagram of our solution. 54

5.1 Hyperledger Fabric Network Design Diagram. 60
5.2 The Login page on the ClassicsChain website. 77
5.3 The Register page on the ClassicsChain website. 77
5.4 The "My Classics"page on the ClassicsChain website. 78
5.5 The "New Classic"page on the ClassicsChain website. 78
5.6 The "Fundamental Vehicle Data"section of the "Vehicle Card"page on the ClassicsChain website. 79
5.7 The "Restorations"section of the "Vehicle Card"page on the ClassicsChain website. 80
5.8 The "Certifications and Other Documentation"section of the "Vehicle Card"page on the ClassicsChain website. 80
5.9 The "Edit Classic"page on the ClassicsChain website. 81
5.10 The "New Restoration Step"page on the ClassicsChain website. 81

5.11 The "History of the Vehicle Card"page on the ClassicsChain website. 82
5.12 The "Transaction information"pop-up of the "History of the Vehicle Card"page on the ClassicsChain website. 82
5.13 The "Access Permissions"page on the ClassicsChain website. 83
5.14 The "Access History"page on the ClassicsChain website. 84
5.15 The "Transaction information"pop-up of the "Access History"page on the ClassicsChain website. 84
5.16 The "User Details"page on the ClassicsChain website. 85
5.17 Images accompanied by captions in the generated restoration PDF document. 87
5.18 Updated version of the classic car project page in the *Charter of Turin Monitor* with the "Vehicle Card"button. 90

6.1 Throughput results of our Hyperledger Fabric blockchain network. 93
6.2 Duration of user management requests, in seconds. 95
6.3 Duration of the simplest write requests, in seconds. 95
6.4 Duration of write requests that interact with the media storage, in seconds. 96
6.5 Duration of read requests to retrieve the current recorded data of a specific classic car, in seconds. 97
6.6 Duration of read requests to list the owned or authorized classic cars, in seconds. 97
6.7 Duration of read requests to retrieve the chronological history of a classic car, in seconds. 98
6.8 System Usability Scale (SUS) results for the *ClassicsChain* web platform. . . . 104

B.1 Questionnaire for the relevance of our solution. The first set of questions (1). 121
B.2 Questionnaire for the relevance of our solution. The first set of questions (2). 122
B.3 Questionnaire for the SUS of *ClassicsChain*. The second set of questions (1). . 123
B.4 Questionnaire for the SUS of *ClassicsChain*. The second set of questions (2). . 124
B.5 Questionnaire for the final considerations on usability and relevance of *ClassicsChain*. The third set of questions (1). 125
B.6 Questionnaire for the final considerations on usability and relevance of *ClassicsChain*. The third set of questions (2). 126

List of Tables

1.1 Overview of the Design Science Research methodology: descriptive steps and corresponding sections. 5

2.1 Description of the work and contributions developed in this project. The work developed in the current research is highlighted in yellow. 13
2.2 Comparison between public, private, and consortium blockchain networks . 17
2.3 Comparison between Bitcoin, Ethereum and Hyperledger Fabric. 25

3.1 Comparison of formal related work on the automobile context. 39

5.1 Comparison of Requests Duration: Old Approach vs. New Approach. The time is presented in seconds. 75

6.1 Responses to the questionnaire regarding the significance of the implemented features in this work. 103
6.2 SUS scoring conventions. 104
6.3 Responses to the questionnaire regarding the final opinions of *ClassicsChain*. 105

Acronyms

ACP Automóvel Club de Portugal *(pp. 1, 2, 100)*
API Application Programming Interface *(pp. 46–48, 51–53, 68, 70–72, 74–76, 86–89, 91, 92, 94, 108)*
ASE Automated Software Engineering *(p. 2)*

BPMN Business Process Model and Notation *(pp. viii, 9, 10, 86)*

CA Certificate Authority *(pp. 23, 57, 60, 61, 69, 70)*
CFT Crash Fault Tolerance *(pp. 18, 19, 24, 57, 59)*
CID Content Identifier *(pp. 52, 65, 66, 72–75)*

DLT Distributed Ledger Technology *(pp. 13, 22)*
DMN Decision Model and Notation *(pp. 9, 86)*
DSR Design Science Research *(pp. 5, 6, 91, 106, 107)*

FCT NOVA School of Science and Technology *(pp. 3, 9)*
FIVA Fédération Internationale des Véhicules Anciens *(pp. 1, 2, 9, 10, 37, 44, 45, 47, 57, 63, 86, 88)*

GDPR General Data Protection Regulation *(p. 109)*
GUI Graphical User Interface *(pp. 9, 49, 51, 53, 71, 76, 85, 86, 91, 99, 103, 106, 108, 110)*

HTTP Hypertext Transfer Protocol *(pp. 53, 72–74)*
HTTPS Hypertext Transfer Protocol Secure *(pp. 51, 53, 68, 85)*

INCD Infraestrutura Nacional de Computação Distribuída *(p. 53)*
IoT Internet of Things *(pp. 3, 10, 13, 34)*
IPFS InterPlanetary File System *(pp. 52, 65, 66, 73, 74, 109)*

JSON JavaScript Object Notation *(pp. xii, 51, 57, 71, 72)*
JWT JSON Web Token *(pp. 71, 75, 76, 88, 117)*

ML Machine Learning *(p. 10)*
MSP Membership Service Provider *(p. 23)*

NFT Non-Fungible Token *(p. 36)*

P2P Peer-to-Peer *(pp. 12, 17, 21, 25, 35, 52)*
PBFT Practical Byzantine Fault Tolerance *(pp. 18, 19, 24, 32, 34, 37, 57)*
PICOC Population, Intervention, Comparison, Outcomes and Context *(p. 28)*
PoA Proof of Authority *(p. 37)*
PoS Proof of Stake *(pp. 18, 19, 22, 25, 32, 37)*
PoW Proof of Work *(pp. 18, 19, 21, 22, 25, 32)*

REST Representational State Transfer *(pp. 47, 51–53, 68, 92, 94)*
RR Rapid Review *(pp. 7, 27, 33, 107)*

SDK Software Development Kit *(pp. 51, 68, 94, 99)*
SLR Systematic Literature Review *(p. 27)*
SMTP Simple Mail Transfer Protocol *(p. 76)*
SRE Software Requirements Engineering *(p. 41)*
SSH Secure Shell *(p. 53)*
SUS System Usability Scale *(pp. ix, x, 101, 103, 104, 123, 124)*

TPS Transactions Per Second *(pp. 93, 94, 99)*

VIN Vehicle Identification Number *(pp. 36–38, 50, 63, 78, 88)*

1

Introduction

This chapter will start by presenting the context and description of the work developed during the elaboration of this research, introducing its theme, and will be followed by a brief description of its institutional environment in section 1.2. In section 1.3, the challenges expected to be encountered during this work will be described, followed by the problem statement and final goals in section 1.4. In section 1.5, the results and contributions expected to be accomplished by the end of this research work, will be presented. The subsequent section (section 1.6) presents the scientific methodology and research executed in the context of this research. Finally, in section 1.7, this document's structure is described by giving an overview of the topics discussed in each subsequent chapter.

1.1 Context and Description

Classic automobiles are much more than old cars. The materials used in their original construction, the craftsmanship that went into building them, and the historical significance they may hold make them true works of art. Like other pieces of artwork, classic vehicles tend to be collectible items and can be sold for millions of euros.

To consider vintage cars masterpieces and consequently to associate them with a large price tag, these need to be equipped with meticulous records, and the majority of them are required to be restored following strict guidelines and expert restoration services to keep their authenticity, i.e., keeping them as close to their original factory condition as possible, and to not cause any damages, like slight scratches or flaws in the paint application, that could potentially depreciate their value by thousands of euros. The guidelines mentioned were proposed by both national entities, such as Automóvel Club de Portugal (ACP), and international ones, like Fédération Internationale des Véhicules Anciens (FIVA), which published a guide providing practical advice on restoration and maintenance of historic vehicles called the 'Charter of Turin Handbook' [2], that will be followed during this work, mainly because it is an international benchmark, but also due to the level of detail presented in it.

It is noticeable that a vehicle's restoration process has a huge impact on the automobile, especially in its price but also in the overall quality of it resulting from this procedure. Because of this, it is mandatory that the restoration's history, with all the activities performed during each process and other important information, is properly recorded and documented in tamper-proof immutable pieces of evidence to be available for the classic car owners, workshops, and institutions that may want or need this type of evidence for certification or evaluation purposes. This is exactly what this work aims to achieve.

1.2 Institutional Context

The execution and research conducted in this work derived from a partnership between the Automated Software Engineering (ASE), group at NOVA LINCS, and the Software Systems Engineering Group at ISTAR-IUL from ISCTE-IUL, in collaboration with Raimundo Branco, a famous Portuguese classic car restoration shop, by serving as our case study, where our research and work will be developed and evaluated.

We are also collaborating and cooperating with the Classic Cars Division of ACP, which is the largest automobile association in Portugal, and BASF Portugal, a division of the German multinational company, which is the largest chemical producer in the world that produces the Glasurit brand of paints, specially sorted to classic cars restorations.

1.3 Challenges

The classic automotive sector is a niche one, sometimes even associated with old practices and outdated mores. Consequently, there is a lack of literature and innovative initiatives within this context, especially concerning the integration of web technologies to create immutable, tamper-proof, and transparent documentation of all details and associated evidence. This is especially true when related to restoration activities. Moreover, existing solutions and initiatives do not meet the necessary level of detail demanded by the meticulous nature of classic car enthusiasts, failing to adhere to the best practices and guidelines outlined in the FIVA's Charter of Turin.

Another aspect to consider is that restoring a classic car is a complex, expensive, and thorough process [2]. Throughout the phases of the restoration process, a large amount of data and evidence are collected, which can be of diverse media types and formats, such as text, photos, or even videos, adding another layer of complexity to the documentation requirements within this sector.

Furthermore, the car's history must be available to all interested parties in the restoration process. Some parts may be interested in only visualization of the records, such as the cars' owners, possible vehicle buyers, or automobile regulators for certification and authentication purposes, but other entities may also need to update the vehicle's documentation. The latter can consist of several workshops since a vehicle can participate in different restoration processes at different shops and individual car restorers may want

to carry out some minor recondition activity according to the mentioned guidelines [2]. This can easily become a challenge since all the documentation generated, regardless of the entity that desires to update it and regardless of the documentation's type and format, needs to be well organized and properly recorded to build or update the detailed, consistent, and trustworthy web portfolio of the car's history.

Additionally, all the mentioned stakeholders must be granted or revoked the necessary permissions to access or update the car's restoration history. For instance, when the restoration procedures are finished in a specific shop, the owner may want to remove the shop's permission to alter the car's registry, or even when the owner of the classic decides to sell it, he may want to give them access to a possible buyer to check the vehicle's restoration activities that took place. For this purpose, a user access management feature is required.

1.4 Problem Statement and Final Goals

Our project will follow the work developed in four preceding research initiatives [3–6] from NOVA School of Science and Technology (FCT), that together aimed to produce a platform to be integrated into the classic car shop mentioned (Raimundo Branco), where the progress of the car's restoration process is available, displaying the procedures that have already been finished, what is being done at the moment, and what is expected to be done next, through the usage of process-aware Business Process Models as graphical interfaces. This progress is updated with the least interaction from the workshop employees, made possible by implementing an Internet of Things (IoT) system that can infer the stages of the restoration procedure using sensors that measure humidity, temperature, vibration, and track position.

Along with the progress of the car's restoration, pieces of evidence documenting the activities executed are generated, and ensuring that they are adequately recorded and stored is the work we are in charge of in this current research. The trustworthy car's history should be accessible to every involved part, with specific permissions to visualize or modify it. Considering this objective, our solution should be integrated into the mentioned repair shop's current platform. However, it will also function as a standalone option, catering to other users like classic car owners, additional restoration shops, and certification authorities.

With the previously developed work and the challenges presented, the main goal for this research is to develop a solution to properly record all relevant information about classic automobiles, related documentation, and all the pieces of evidence generated during the stages of a classic car restoration process, assuring that this evidence is tamper-proof, immutable, transparent and traceable, based on the blockchain technology to make available a complete chronologically continuous online portfolio of restoration activities and any other modifications to it. This solution must also provide data sharing and collaboration between the relevant entities within the classic automotive ecosystem. With

this solution, classic car owners benefit by having a way to prove the authenticity of their vehicles, while restoration shops can prove that the restoration procedures that are performed are trustworthy, and facilitate a straightforward and uncomplicated certification process for certification regulators. This can combine into an overall increase in the value and recognition of the vehicles, accompanied by an increase in trust and confidence in the classic automobile sector.

Several approaches have been applied to validate software relevance, namely by trying to validate its requirements [7]. In our case, the relevance of our solution, given that it was aligned with the evolution of a global project to digitally transform the restoration of classics, seemed to be guaranteed from the outset. However, potential users of a software platform often find it challenging to validate its requirements in the absence of a mockup or running prototype for several reasons. Firstly, without a visual representation or hands-on experience with the software, users may struggle to conceptualize how the proposed solution will address their needs or fit into their workflow. Verbal or written descriptions alone may not sufficiently convey the intended functionality or user experience, leading to misunderstandings or misinterpretations of the requirements. Additionally, users may have diverse backgrounds, preferences, and levels of technical expertise, making it difficult for designers to anticipate and address all potential user needs without direct feedback. Furthermore, without a prototype to interact with, users may find it challenging to provide constructive feedback or identify potential issues with the proposed solution, hindering the refinement of requirements and the overall success of the software platform.

To mitigate the aforementioned handicap of users in recognizing the relevance of our platform and validating its requirements, we opted to carry out its post-factum evaluation with a set of representatives of the most important stakeholders, i.e. in the presence of a consolidated prototype of the platform described. This confirmation will be presented in a later chapter.

1.5 Contributions

The main contributions resulting from the development of this work:

- A study and review of the literature, including methodologies, techniques, and tools to generate trustworthy documentation, more specifically on the car industry;
- Design and implement a decentralized, blockchain-based platform to adequately register all the evidence and documentation generated in the process of car restorations;
- Deliver a finished and practical product for classic car owners and restoration shops to record trustworthy, transparent, traceable documentation and all associated information.

- Offer classic car certification and authentication bodies a product that can facilitate their tasks by providing evidence on the restoration procedures performed and associated documentation;

1.6 Scientific Methodology

The Design Science Research (DSR) methodology will be followed during this work. This methodology is a systematic, iterative, and scientific outcome-based approach that aims to acquire and develop the necessary knowledge to create new artifacts, processes, or systems to solve a well-defined problem in a specific research area, offering specific guidelines for evaluation within research projects [8, 9]. Figure 1.1 illustrates the typical DSR Process Model, highlighting its process iterations and outputs.

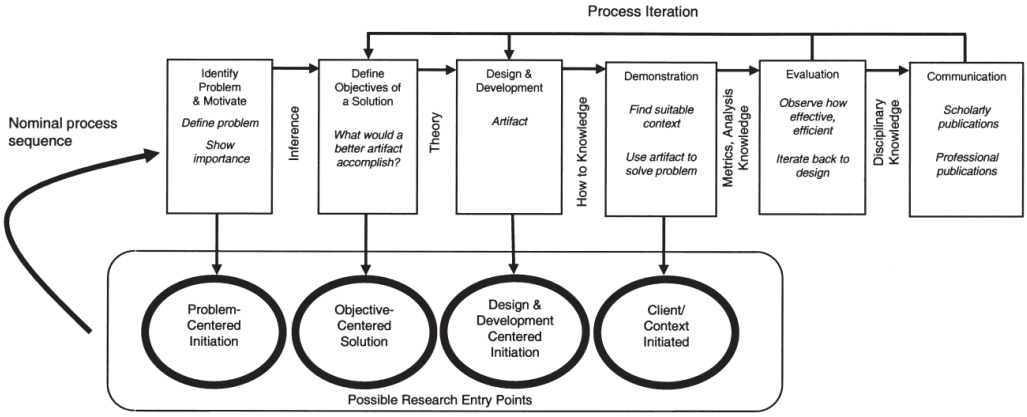


Figure 1.1: Design Science Research Process Model. Taken from [8].

The DSR process model steps, which were followed iteratively, are described in Table 1.1, accompanied by the respective sections and chapters where each is explained.

Table 1.1: Overview of the Design Science Research methodology: descriptive steps and corresponding sections.

Step	Description	Sections
Identify Problem & Motivate	Identify the research problem, understanding the context, motivation and insights into relevant concepts.	Sections 1.1 to 1.4, Chapters 2 and 3
Define Objectives of a Solution	Clearly define the objectives that the designed solution should achieve.	Section 1.5, Chapters 2 and 3
Design & Development	Develop the necessary artifacts and systems to solve the research problem.	Chapter 4 and 5
Demonstration	Present and demonstrate the designed artifact's functionality and capabilities, to obtain constructive feedback.	Section 6.3
Evaluation	Assess the effectiveness, usability, and performance of the developed solution in addressing the identified problem.	Sections 6.2 and 6.3
Communication	Share the findings, results, and knowledge gathered with the community through academic publications, presentations, or other means of communication.	Section 6.4

We now describe how these DSR steps were reified in the current work:

- **Identify Problem & Motivate**: In [3–6], the vehicles' data and evidence generated in each restoration activity need to be reliable, tamper-proof, auditable, and not possible to manipulate without the specific permissions. Additionally, comprehensive information regarding the vehicles, encompassing both technical and non-technical details, assessment certificates, and historical ownership records, must be trustworthy and properly recorded. In the previously developed work, a feature serving that purpose is not implemented. Moreover, there is also the need for all the aforementioned information to be securely shared, enabling collaboration between interested entities.
- **Define Objectives of a Solution**: It was clear from the beginning of the project that our solution would rely on blockchain technology, smart contracts, and its unique features to record trustworthy vehicle information properly. Based on a review of relevant concepts, and both academic and grey literature, various technologies were evaluated to determine the most suitable options for our problem and clearly define our objectives.
- **Design and Development**: A working solution was implemented to tackle our problem, ensuring that it is advantageous for all the entities involved and applicable in the automotive industry.
- **Demonstration**: The developed solution was demonstrated to a set of relevant stakeholders to obtain feedback and reiterate the need for our work.
- **Evaluation**: It was also crucial to assess the usability and effectiveness of our solution. For usability assessment, questionnaires and interviews with several stakeholders were employed. Performance evaluation comprised the execution of performance tests to assess diverse measures and metrics.
- **Communication**: A conference paper with preliminary material from our work was presented at an international conference, and a journal paper, already available as a preprint, is under preparation.

1.7 Document Structure

The rest of this document is divided into the subsequent chapters:

- **Chapter 2 (Background)**: In this chapter, an insight into the core concepts for this project will be presented and detailed to allow a better understanding of the problem in question and the possible solutions introduced. In the first section of this chapter, an overview of the work already developed in the previous studies, which will be continued in the current one, is presented. The second section explores the concept

of Blockchain, its underlying technologies, and its unique characteristics. In the last section of this chapter, the blockchain platforms that are most relevant to our work are described and compared.

- **Chapter 3 (State of the Art)**: In this chapter, we present a selection of the studies and solutions that are relevant to our research topic. We begin by describing the literature review technique used, named Rapid Review (RR), followed by a review of the articles and projects previously developed with similar objectives and research questions in both the automotive industry and other sectors.

- **Chapter 4 (System Proposal)**: In this chapter, we present and discuss the proposed solution. We start by detailing the system's stakeholders, their roles, associated requirements, and use cases. Our solution's extended architecture is also explained, meticulously describing the components and technologies that form this project.

- **Chapter 5 (Implementation)**: In this chapter, the details of our solution's implementations are described. The development details and implementation decisions made during the development process are explained.

- **Chapter 6 (Verification and Validation)**: In this chapter, we validate the developed work by addressing the performance, usability, and relevance tests, subsequent results, and their corresponding interpretations.

- **Chapter 7 (Conclusions)**: In this chapter, we conclude the document by providing a concise overview of the accomplished work and proposing potential future work.

This page was intentionally left blank.

2

Background

This chapter presents the building blocks we will use in our work and gives insight into some essential concepts. We will start by presenting an overview of the work previously developed in the precursor research in section 2.1. In section 2.2, blockchain technology will be introduced, describing overall important details and explaining relevant information to the work being developed, such as consensus protocols and smart contracts. Finally, in section 2.3, the most popular and pertinent to our work blockchain platforms are described.

2.1 Classic Car Restoration Workshop Digital Transformation Project

As mentioned in section 1.4, the work undertaken follows the research work already developed in four other research works [3–6] of FCT. It is deemed appropriate to provide an overview and some background details on the work previously developed.

The previous work consisted of creating and implementing a system that facilitates and refines the quality of the documentation of all the processes involved in classic car restorations, improving the efficiency and reliability of the restoration shop's work while also making it less time-consuming.

In [3, 5], a Business Management System was integrated with a process monitoring platform that combines data collected from sensors distributed along the plant shop with input provided manually by a plant shop manager using a Workflow Editor as a Graphical User Interface (GUI) to create Business Process Model and Notation (BPMN) and Decision Model and Notation (DMN) models that represent the restoration process (Figure 2.1) supporting the guidelines from FIVA's Charter of Turin [2], this way naming this platform as *Charter of Turin Monitor*. Clients cannot only track the progress of their vehicles' restoration or preservation process through the graphical interface mentioned but can also interact and visualize the pieces of evidence generated from some of the restoration procedures applied to their cars.

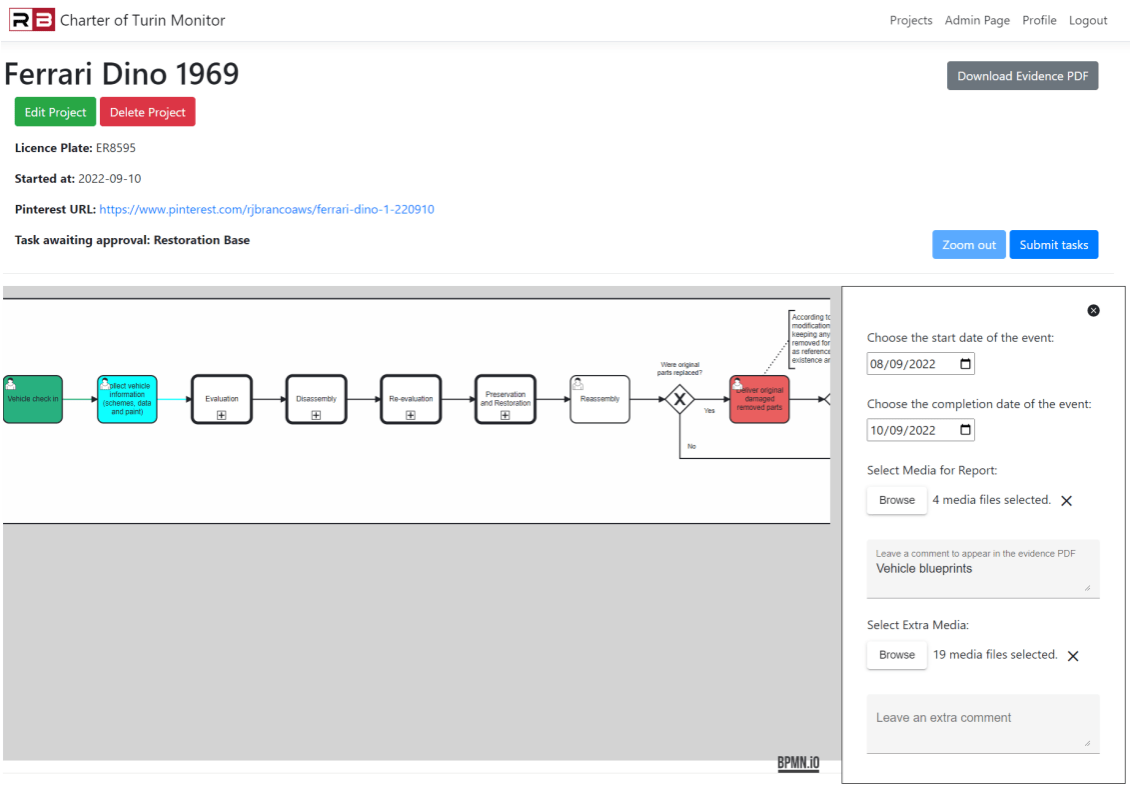


Figure 2.1: Example of one restoration procedure BPMN from the *Charter of Turin Monitor*. Taken from [5].

A PDF file can be automatically generated, containing a summary of all the steps performed and the evidence, mainly photos and diagrams, related to these steps. The car's owners can use this PDF as proof of all the details of the work carried out to restore the vehicle. An example of one of the automatically generated PDF file sections can be found in Figure 2.2.

It is important to mention that the *Charter of Turin Monitor* is in continuous use at Raimundo Branco workshop and obtained valuable feedback from Museu do Caramulo and FIVA, national and international entities, respectively.

As mentioned previously, the process monitoring platform gathers data from manual input by the employees of the workshop. However, the progress of the car's restoration procedure should be updated with the least human interaction. To settle this requirement, [4, 6] developed an IoT system to detect the type of restoration activity performed on the vehicle, combining Machine Learning (ML) algorithms with energy consumption sensing and other sensing techniques, such as measuring temperature, vibration, humidity and tracking indoor location (Figure 2.3).

Last but not least, the developed platform allows video-conference sessions with remotely controlled cameras for classic car owners to visualize, inspect, and record relevant milestones in the restoration process. An example of a video conference performed on the platform is depicted in Figure 2.4.

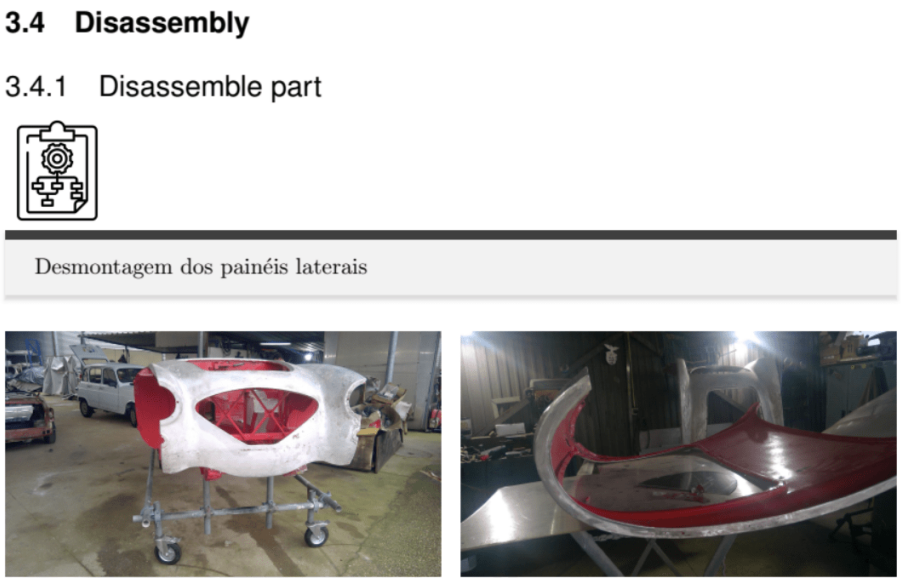


Figure 2.2: Example of one of the sections of the PDF file. Taken from [5].

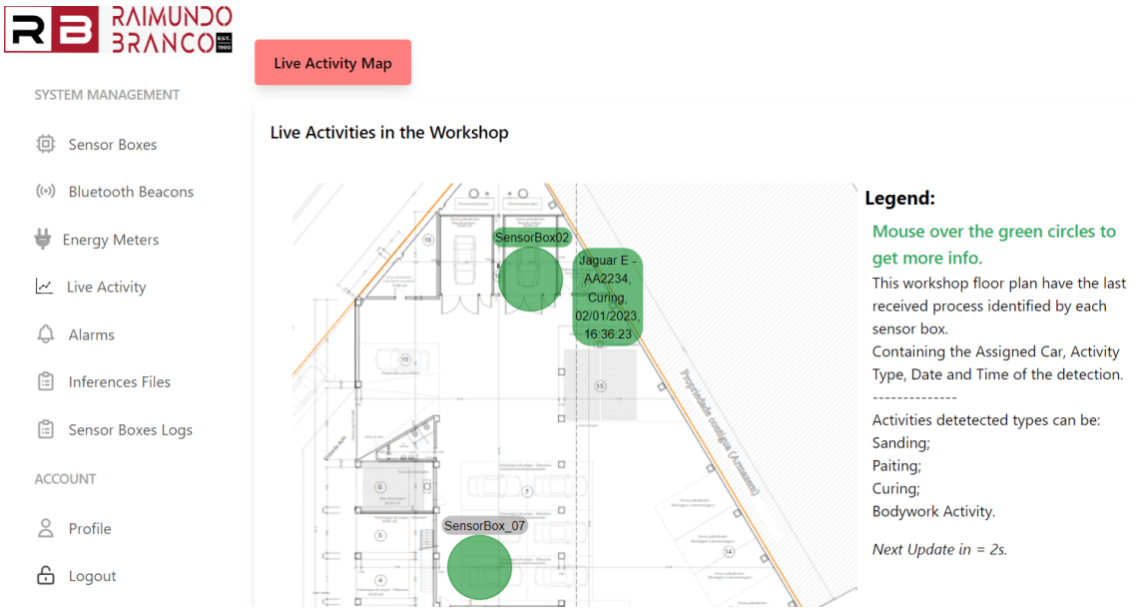


Figure 2.3: Example of the platform's activities live map. Taken from [6].

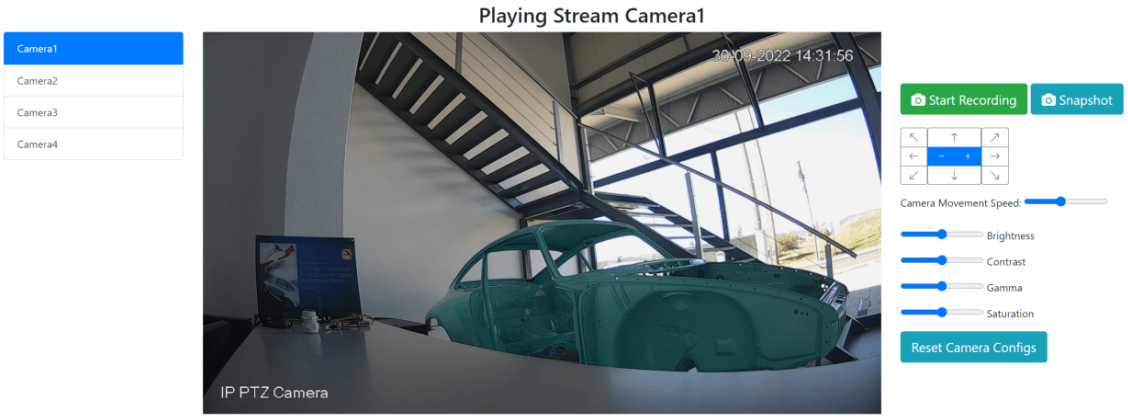


Figure 2.4: Example of a video conference. Taken from [5].

Table 2.1 presents a concise description of the previous contributions to this project [3–6]. The table highlights the main distinctions and original contributions of the work undertaken in the current research.

2.2 Blockchain Technology

Blockchain technology is often described as an exceptional solution for the longstanding challenge of establishing trust among humans [10]. It is a decentralized and distributed digital ledger that allows the recording and verification of transactions without the need for a central authority. Essentially, it is a distributed database that stores, manages, and shares data among all participants in the network, maintaining a continuously growing list of transaction data records, which are cryptographically secured to prevent tampering and revision.

Block chain technology, as it was initially named and later changed to Blockchain, was first proposed in a white paper published by Satoshi Nakamoto (whose identity is still unknown) in 2008 and was introduced as the underlying technology for the cryptocurrency, *Bitcoin* [11]. This paper introduced the idea of a Peer-to-Peer (P2P) online payment system that did not require the use of a third party, like a bank or financial institution, and that solved the problem of double spending in digital currency by linking every transaction to the preceding one in a decentralized public ledger creating an immutable record of all the transactions.

In the context of our work, we anticipated since the beginning of our research, that blockchain technology would be used in our project. To better understand blockchain, an introduction to its key underlying technologies needs to be performed. Though not new, their combination makes blockchain unique. These key technologies are the following:

- **Cryptography** deals with securing digital data, as security is a prime concern in almost every application that runs on the Internet [12]. It establishes a secure digital identity for each network participant and validates transactions. This is

Table 2.1: Description of the work and contributions developed in this project. The work developed in the current research is highlighted in yellow.

Title	Author	Year	Description of the work
Process-based Monitoring in Industrial Context - The Case of Classic Cars Restoration [3]	Diogo Lívio	2021	Web platform integrated with a Workflow engine that executes BPMN/DMN process models that represent classic car restoration procedures matching the *Charter of Turin* requirements. Additionally, remotely controlled IP cameras facilitate video conferencing with classic car owners during restoration processes.
An automated system for monitoring and control classic cars' restorations: An IoT-based approach [4]	Diogo Pereira	2021	Web platform that leverages an Internet of Things (IoT) smart sensor system to identify classic car restoration processes and pinpoint the internal workshop location.
Developing and validating process-aware GUIs in an industrial setting [5]	Pedro Moura	2022	Continuing the work made by Diogo Livio, the platform was enriched with additional features, improved functionalities, and the integration of automatically generated documents. These documents succinctly capture the steps and associated media evidence involved in classic car restorations.
Combining Different Data Sources For Industrial Process Monitoring [6]	Rodrigo Gomes	2023	Continuing the work made by Diogo Pereira, enhancing the capabilities of the IoT sensor system. This involves integrating ML algorithms with energy consumption sensing and various sensing techniques, including temperature, vibration, and humidity measurements.
(Current work) Trustworthy Classic Car Restoration's History Using Blockchain	José Murta	2023	A blockchain-based web platform for recording and securing all pertinent information about classic cars, documentation, and evidence generated during restoration processes. Ensuring trustworthy, immutable, transparent, and traceable records, the system aims to create a continuous registry of restoration activities. Additionally, it facilitates data sharing and collaboration among relevant entities in the classic automotive ecosystem. These functionalities were also integrated with the previous work and respective web platforms.

achieved by cryptographic hashing and digital signatures, a standard aspect of most cryptographic protocols used in several transactional models where it is crucial to detect and prevent forgery or tampering. Digital signatures are used to prove and authenticate the sources of messages or, in blockchain terms, transactions. A correct digital signature ensures authentication and non-repudiation since it proves that a known authenticated sender created the transaction or message and the sender cannot deny having sent that message. Additionally, it also guarantees the integrity of the transaction data [13].

- **Distributed Ledger Technology (DLT)** allows for decentralized digital management of information with high transparency and security. Fundamentally, it is a database for recording the transaction of assets that can be shared across multiple places

simultaneously, holding a synchronized copy of the records. Consensus protocols are required to guarantee that the records are properly replicated within a network.

- **Consensus Protocols** consist of the mechanism by which the network reaches an agreement on the current state of the ledger, and the validity of transactions [13]. After an update on the ledger, it is crucial that all the nodes in the network have a single and accurate copy of the ledger, but this is only possible after the nodes reach a consensus on which version of the ledger is correct. The higher the number of nodes in the network verifying and validating the transactions, the more secure the network. Consensus protocols are better explained in subsection 2.2.3.

The union of the presented technologies and their noteworthy development makes blockchain a novel technology with unique and distinctive characteristics that has been gaining interest and relevance among several sectors all around the globe [14]. Next follows a brief description of the most notable characteristics of blockchain technology as well as the provided benefits:

- **Decentralization**: Blockchain allows for the direct sharing of a database through a distributed ledger to record and verify transactions instead of relying on an intermediary or central authority. This decentralized structure provides increased transparency and security by making the ledger accessible to all network participants and validating transactions by consensus among the nodes that process and store these transactions.

- **Immutability and Persistency**: Blockchain transactions are stored in blocks, each of which is linked to the previous block via a cryptographic hash function. Any alteration to the content of a block will also change the subsequent blocks. So to modify a specific block, all the following blocks would have to be modified too. This is an extremely difficult task to execute and practically infeasible due to the decentralized nature of the network. Besides that, the transactions need to be validated and recorded in blocks distributed in the whole network [15]. This way, blockchain enables immutable, tamper-proof transaction records that cannot be altered or deleted.

- **Transparency**: As mentioned previously, since the transactions are replicated among the network nodes, and the blocks where these transactions are recorded are chained to the previous block, modifications to the stored data are visible to all network participants providing increased transparency, and accountability [16].

- **Traceability**: Traceability is also enhanced with blockchain technology. Since each transaction on the blockchain needs to be validated and recorded with a timestamp in blocks linked together, the network's participants can simply trace previous records and updates on the ledger iteratively [15].

- **Trustless**: Blockchain allows for transactions between entities that do not necessarily trust each other. The decentralized nature of the ledger and the fact that ledger updates are validated via a consensus protocol ensure the validity of transactions in an untrustworthy environment. Smart contracts, essentially described as code deployed and running in the blockchain (which are going to be better explained in subsection 2.2.4), also provide the ability to enforce predefined terms and conditions of an agreement, regardless of the level of trust between the parties involved. Smart contracts have significantly contributed to the widespread acceptance of blockchain [13].

- **Security**: As aforementioned, due to its decentralized structure and the cryptographic techniques used to verify data transactions and, consequently, the data stored in the ledger, blockchain technology guarantees high security and integrity.

2.2.1 Blockchain Network Types

Depending on the use, requirements, and access permission levels, blockchain systems are typically categorized into three types: public, private, or consortium. Each type has unique features and characteristics as well as advantages and disadvantages [16].

Public blockchains, also referred to as permissionless, are typically open source, and are accessible by anyone in the world with no restrictions being applied. The network is open to participation by anyone who can interact with it and contribute. Network participants can perform read and write operations on the ledger and participate in the consensus process, validating transactions and blocks to be added to the chain.

Public blockchains' participants, can typically join the network as a simple node, full node, or mining node. A simple node can only send and receive transactions and does not maintain a copy of the ledger or validate transactions, while a full node does both. Mining nodes are full nodes capable of adding transactions to blocks and broadcasting them to the network. Mining and full nodes are responsible for executing the consensus protocols.

This network provides economic incentives for users to participate in the consensus process to attract more participants to join the network. This incentive is a payment in the form of cryptocurrency native to the blockchain in question.

Bitcoin blockchain [11] was the first and most notorious implementation of a public blockchain. Another extremely popular public blockchain is *Ethereum* [17]. However, the latter blockchain platform can be extended to provide well-suited applications for enterprise-focused sectors, as will be discussed next.

When first idealized, blockchain technology was designed following a philosophy of transparency and inclusivity, where if any user desired to participate, use, or audit the system, he would be able to do it without restrictions. However, new use cases have arisen where additional requirements, such as volume scalability, data privacy, and access control, are intended together with some of the blockchain's original characteristics. From

these needs, permissioned blockchains emerged, which can then be separated into private and consortium blockchains.

A **private** blockchain is a digital ledger that provides proprietary networks where a central organization can grant access to the potential participants and assign them roles [16]. Write permissions are kept centralized to a group of users within one organization, while read permissions can be public or restricted [18]. This type of blockchain requires a level of trust between the network peers, and the number of peers is reduced.

Industries specifically design this kind of blockchain for their internal and commercial use, and recently, many companies have been implementing this type due to its efficiency and auditability. *Ethereum* and *Hyperledger Fabric* [19] are the most commonly used platforms to deploy and maintain private blockchains.

For similar use cases as private blockchains, another interesting solution that is important to consider is the *Amazon Quantum Ledger Database*. Although not classified as a blockchain since it is a centralized ledger fully managed by *Amazon Web Services* allowing its transactions to execute without the need for multi-party consensus, it has garnered notable attention in private scenarios, such as finance, banking, and insurance sectors. However, it is a proprietary solution with substantial deployment and maintenance expenses and does not support the execution of smart contracts.

A **consortium** blockchain is derived from the two previously mentioned blockchain types. As the name suggests, a group of organizations collaborate to maintain the network in this blockchain category. The consensus process is controlled by a pre-selected set of nodes from the group's organizations. Read permissions can be public or restricted to the participants, and write permissions are only acquired by known and trusted participants from the organizations involved.

This type of ledger is best suited for inter-organizational collaboration. As in private blockchains, both *Ethereum* technology and *Hyperledger Fabric* are the most popular and well-suited platforms for consortium blockchains' purposes.

Table 2.2 depicts a comparison between the blockchain environments presented. By analyzing this table, it is noticeable that public blockchains are completely decentralized and transparent, where no intermediaries are necessary, but with that, scalability issues arise, decreasing efficiency and transaction speed. In contrast, private and consortium blockchains are less decentralized and transparent but offer increased scalability, efficiency, and data privacy.

2.2.2 Blockchain Architecture

As it has been mentioned during this section, blockchain is a continuously growing list of records that are cryptographically linked. These records are called blocks. A block is divided into two main components: block header and list of transactions.

The block header of each block is usually constituted of three important components. The first element is a cryptographic hash of the previous block. It is understandable

Table 2.2: Comparison between public, private, and consortium blockchain networks

	Public blockchain	**Private blockchain**	**Consortium Blockchain**
Network join permission	Open	Restricted	Restricted
Consensus determination	All block generators	Selected nodes from one organization	Selected nodes
Transaction visibility	Public	Public or restricted	Public or restricted
Permission	Permissionless	Permissioned	Permissioned
Decentralized	Full	No	Partially
Trust in the network	Not required	Required	Required
Data privacy	Low	High	High

that the first block of a chain (called genesis block), is an exception to this since there is no antecedent block to the first one. The second component usually consists of more than one element and its composition may vary depending on the blockchain platform, however, these elements consist of statistics that are used to create the block [16], such as the timestamp of its creation or its version. The last component included is the Merkle tree root and consists of the hash value of all the transactions in the block [15]. The block body is composed of the rest of the Merkle tree, comprising the list of the hashed valid transactions. The block size and transaction size determine the number of transactions in a block.

The architecture of blockchain, detailing the structure of one block and the chain created by the linkage of blocks, is presented in Figure 2.5.

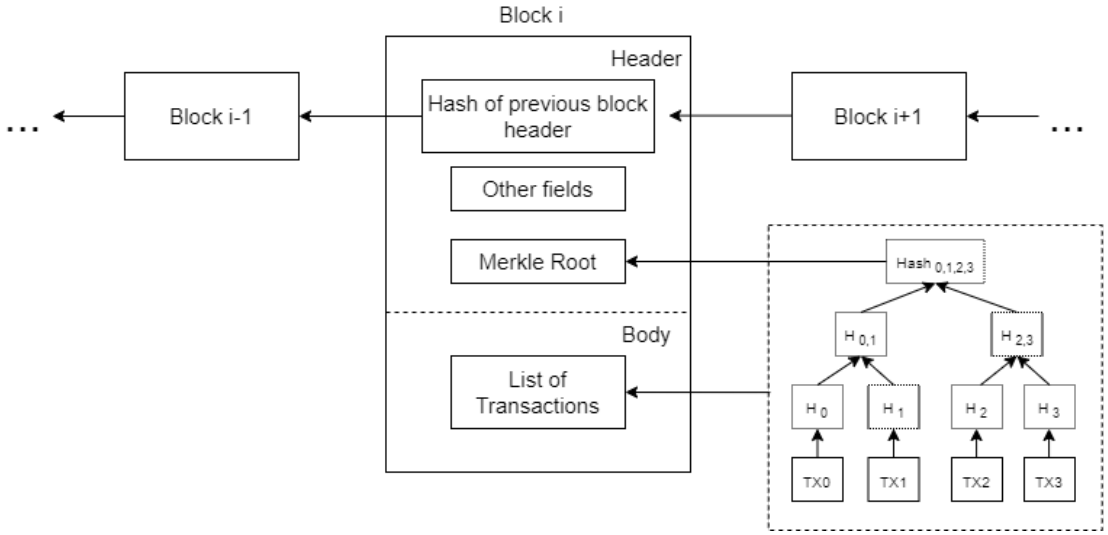


Figure 2.5: Example of blockchain structure.

Blockchain technology runs distributively along communicating nodes that operate on the same ledger via the copy that each one holds, and this way forming a P2P network. Nodes can be thought of as clients or users, that, besides interacting with the system, can also help to execute it properly and maintain it. The interaction between users and the

blockchain is made using asymmetric cryptography, where the private key is utilized to sign the user's transactions, whereas the public key has the purpose of decrypting those transactions and making the user addressable.

2.2.3 Consensus Protocols

After an update on the blockchain, nodes must agree on the transactions and their order. Sometimes even different blocks can be produced concurrently, called a fork, creating a temporarily different version of the ledger. To solve these unique situations, consensus protocols are necessary to ensure that a single copy of the ledger is achieved in all nodes. In other words, consensus protocols maintain the consistency and stability of the blockchain.

It is a demanding challenge to reach a consensus in a decentralized and trustless network. In a fair scenario, the order of transactions for the next block would be selected by a vote among all validating nodes, and the majority's decision would be followed. However, this would be catastrophic since an entity could join the network with multiple validating nodes, get multiple votes to get the majority's decision, and this way tamper the ledger or influence it with a damaging intent [13].

Consensus protocols come in a wide diversity of implementations and have numerous names. Consensus is reached in a public blockchain through the mining process, where validating nodes compete, in diverse means according to the blockchain platform, to add the next block to the chain. In a private or consortium blockchain, consensus can be reached using other techniques, such as employing a central authority or trusted nodes.

Next follows an explanation of the most popular consensus protocols, those being *Proof of Work (PoW)*, which was introduced to the blockchain technology with Bitcoin's creation, *Proof of Stake (PoS)*, which consists of a viable alternative to *PoW* and which the Ethereum blockchain recently moved to, *Crash Fault Tolerance (CFT)* and *Practical Byzantine Fault Tolerance (PBFT)*, consensus protocols widely used in permissioned blockchains.

- ***PoW***: As previously mentioned, this is the consensus protocol that Bitcoin relies on. However, this protocol was originally introduced, with some slight differences, in 1993 by Cynthia Dwork and Moni Naor [20]. In this protocol, the nodes are given a compute-intensive mathematical problem to solve that is extremely difficult to resolve but easy to verify its correctness. The mining nodes compete against each other to solve this problem, and the one that gets it first gets to add the next block to the blockchain and receive an agreed incentive in the form of the native cryptocurrency. For instance, in the Bitcoin blockchain, miners compete to find the right random number, called *nonce*, that makes the block hash have a specific amount of leading zeros [13]. There is no other way to find this number than cycling to all possible values until the correct one is reached. As expected, this requires a lot of computation power, resources, and energy, exponentially increasing as the blockchain grows. In fact, in recent years, the annual electricity consumption due to

bitcoin mining has reached the annual energy consumption of some medium-sized countries, such as Austria and Norway [21].

- ***PoS***: This consensus protocol is an energy-saving alternative to *PoW* that uses significantly fewer CPU computations and overall resources and electricity. In *PoS*, the validating node is selected proportionately to the node's balance. The higher the node's staked cryptocurrency, the higher the chances for that node to mine the block. There is no competition among the mining nodes and no mining reward, nevertheless, in this consensus protocol, these nodes also need the incentive to stake a high amount of cryptocurrency to keep appending the blocks, and this is done by charging a fee to add the block to the chain, which is called the block reward. It is likely to think that in *PoS*, a malicious entity with a high amount of staked cryptocurrency can attack the network. However, this situation would only be possible if this entity had at least 51% of the total cryptocurrency. Besides this being practically improbable, it is also implausible to damage a network if the entity owns that much quantity of cryptocurrency [16]. On September 15, 2022, the Ethereum platform transitioned from *PoW* to *PoS* consensus protocol, expecting to reduce energy consumption by up to 99.95% [22].

- ***CFT***: *CFT* is a consensus protocol that integrates a level of resilience within the protocol to ensure the network's stability and agreement, especially in scenarios where nodes might experience crashes or failures while still achieving consensus among the nodes. This type of protocol is usually effective to function even when up to half of the nodes are inaccessible at any given moment. This is a protocol typically used in private or consortium scenarios. One well-known example of a *CFT* protocol is *Raft*, which is implemented in platforms like Hyperledger Fabric [19]. Further elaboration on this topic will be provided in subsection 2.3.3.

- ***PBFT***: This consensus protocol was first introduced in 1999 as a type of asynchronous communication protocol [23]. With blockchain technology, this algorithm guarantees that all nodes concur on the network's current state, even when dishonest or malicious nodes are present. It can be considered as an upgrade of *CFT* protocols, as it not only accommodates node failures but also addresses situations where nodes exhibit malicious intent. For this protocol to function properly, the number of faulty nodes must not be greater than 1/3 of the total available nodes in the network, and each node must be known. In *PBFT* protocol, a primary node, selected through a round-robin model, proposes the system's next state and the secondary nodes, known as validators, vote on whether to accept the proposal. The proposed state is accepted if a predetermined number of nodes approve it. As with *CFT*, this protocol is typically used in private and consortium blockchains since the set of nodes is known, usually running on a trusted environment, and sometimes it is even fixed. As a result,

consensus can be reached more quickly and effectively, but less decentralization emerges as a consequence.

2.2.4 Smart Contracts

The concept of smart contracts was first introduced in 1994 by Nick Szabo, explaining that smart contracts "combine protocols with user interfaces to formalize and secure relationships over computer networks" [24]. It was presented as a way to facilitate, verify and enforce the negotiation of a contract, translating contractual clauses involving multiple stakeholders and organizations into computer algorithms that self-execute when pre-defined conditions and requirements are reached, minimizing the need for trusted intermediaries, the possibilities of maliciously altering the agreement or other unexpected situations. Additionally, smart contracts make it possible to automate complicated procedures, increasing efficiency and lowering costs.

Within the blockchain context, smart contracts were originally adapted and supported by the Ethereum platform and can be described as scripts stored on the blockchain. The physical machine that has to be employed as the back-end for smart contract execution is replaced by the blockchain nodes. Smart contracts are triggered when desired conditions are met or by specific transactions that execute functions of the smart contract, forwarded to the unique address of the contract stored on the ledger. The smart contract execution runs independently on each network's node. These nodes have special features, namely the need to run a virtual machine to execute the smart contract's logic [13].

Before the adaption of smart contracts to blockchain technology, the process of storing data other than cryptocurrency transaction data was inefficient, not secure, and not possible to store complex and large types of data. With smart contracts, it is possible to record different types of fields, with a diverse variety of types, functions, and data structures, such as arrays, structs, mappings, and even classes and objects, to document complex scenarios, logic, and use cases. Smart contracts have significantly contributed to the broad adoption and exploration of blockchain technology across various use cases, industries, and environments.

Smart contracts are mostly written in domain-specific programming languages such as *Solidity*, which is the language in which Ethereum smart contracts are written. However, some blockchain platforms, which is the case for Hyperledger Fabric, enable the possibility to write smart contracts in more widely used programming languages, such as *Java*, *Go* or *Node.js*.

At this time, a smart contract using blockchain technology can automate most of a company's system and procedures. This is also applied to multi-organizational environments, where several organizations share business logic through the deployment and execution of decentralized smart contracts on the blockchain.

2.3 Blockchain Platforms

In this section, an overview of the most popular and relevant to our work blockchain platforms will be presented. We will be discussing the most important features, unique approaches, and pertinent implementation details, as well as some limitations of these platforms that can help us develop an adequate solution to our problem.

2.3.1 Bitcoin

It is infeasible to describe and explain blockchain technology without giving noteworthy detail to the most popular implementation of it: Bitcoin. As previously mentioned at the beginning of section 2.2, Bitcoin was first introduced in 2008 as P2P online payment system that did not need a trusted intermediary. The Bitcoin blockchain is a publicly distributed ledger that stores validated transactions in blocks. The first Bitcoin block was mined in 2009.

In the Bitcoin network, a digital signature mechanism is established to authenticate and verify transactions, where a user or a node is associated with a pair of public and private keys, which are needed to propose and validate those transactions. In Bitcoin, for a transaction to be broadcasted to the network, the transaction's data is first hashed and then it is encrypted using the user's private key, generating a digitally signed transaction. The encrypted transaction and the original transaction's data are broadcasted together to the network. Then, to prove the authenticity and integrity of the transaction, each of the network's validating nodes executes the validation process, decrypting the signed transaction with the original signer's public key to obtain the original hash of the transaction's data and hashing the unaffected original transaction data. If both hash values match, it means that the transaction is valid, authentic, and was not tampered with.

Once a Bitcoin transaction is validated and verified as authentic, it is transmitted to the network, where the mining nodes will try to append it to a block with a predefined size. Mining nodes compete against each other to mine new blocks, and to append them to the chain, it is necessary to reach a consensus on the network. To reach this consensus, the protocol used in bitcoin is *PoW* as it was previously explained in subsection 2.2.3. The mining nodes receive a percentage of the amount of cryptocurrency being transferred in the transaction as an incentive to keep running the node and, this way, support the network's integrity in exchange for the consumed energy in trying to mine new blocks. In order for a Bitcoin transaction, where cryptocurrency is exchanged from one user to another, to be completely confirmed, it can take up to 6 appended blocks after the one that was mined with the transaction, depending on the amount of cryptocurrency being transferred. In the bitcoin blockchain, the process of mining one block takes an average time of 10 minutes, which adds substantial latency to the system and prevents instantaneous transactions.

2.3.2 Ethereum

Ethereum was launched in 2015 [17] and it is the second most popular blockchain platform. Ethereum is open-source and programmable, which means that it enables developers to build and deploy decentralized applications on its network due to its integration with smart contracts. These applications can serve a variety of purposes by using the blockchain to store data or control the desired logic for specific use cases.

Ethereum has a native cryptocurrency called *Ether*, which is used to pay for transaction fees and computational services on the network, such as running decentralized applications and smart contracts.

As explained in subsection 2.2.4, Ethereum was the first blockchain technology to support smart contracts, which are digital scripts, stored on the blockchain, that can contain complex logic and that self-execute once desired requirements and terms are met. Ethereum smart contracts are written in a domain-specific programming language called *Solidity*.

As mentioned in subsection 2.2.3, the Ethereum platform recently moved from a PoW consensus protocol to a PoS, reducing drastically the energy consumption required to mine new blocks to the blockchain. This switch to a different consensus protocol was also made with the intention to increase transaction performance and efficiency, however, this is not yet noticeable since it is a long-term objective.

2.3.3 Hyperledger Fabric

Hyperledger Fabric is an open-source permissioned DLT platform, designed specially to be applied in enterprise contexts, that offers modularity, resiliency, flexibility, scalability, and confidentiality for a diverse variety of industry use cases [25]. These application cases cover but are not limited to topics including identity management, contract management, supply chain and trade logistics, and food safety [19].

One important area of distinction is that Hyperledger was created under the Linux Foundation business model, which has a long history of developing open-source projects with transparent governance and vibrant ecosystems. The Fabric project, in particular, is composed and maintained by a community with multiple organizations involved.

Hyperledger Fabric presents a highly modular and configurable architecture, to support and optimize a broad range of industries and companies' requirements. As mentioned, it is also the first distributed ledger platform to support smart contracts written in general-purpose programming languages such as *Java*, *Go*, and *Node.js*, instead of domain-specific programming languages [25]. In Fabric nomenclature, smart contracts go by *chaincode*.

Most of the blockchain platforms precedent to Hyperledger Fabric used an order-execute architecture, meaning that the transaction flow consists of, first ordering the transactions, then propagating the ordered transactions to all peers in the network, and finally all the nodes executing all the transactions sequentially. This architecture is simple and widely used in different blockchain systems, ranging from public to permissioned.

However, this architecture brings some limitations to the system [19], specially in permissioned platforms, such as:

- Transactions must be deterministic to guarantee that the same result is established in all peers.
- All peers must also maintain the smart contract to execute the transactions which can lead to its logic being viewed easily.
- The smart contracts must be written in domain-specific languages which can lead to programming errors that are irreversible.
- The sequential execution of all transactions by all nodes limits performance.
- The consensus protocol is fixed to the blockchain platform and, for specific use cases, it might not be feasible to use that platform.

In contrast, Hyperledger Fabric presents an execute-order-validate architecture to tackle the stated limitations, increasing the system's resiliency, flexibility, scalability, and performance with a modular design for permissioned blockchains [26]. One of the most important features that arise with this architecture is the support for pluggable consensus protocols, which allow the platform to be more effectively modified to match certain use cases [25]. Additionally, in Fabric, the consensus protocols do not require a native cryptocurrency to incentivize mining or support the execution of smart contracts, which is a beneficial feature since, in private and consortium blockchains, the users, as expected, do not want to have a cost associated with any type of transaction when they are the entities maintaining the network, running the nodes that validate and add those transactions into blocks.

To understand the Fabric's transaction flow it is important to understand some of the aspects of how its network is constituted. *Peer* nodes are nodes, maintained by the member organizations of the network, which store and execute *chaincode* processing transactions (called *endorsing peers*), access ledger data, verify transactions, and interact with client applications. Fabric's network is also comprised of an *Ordering Service*, formed from a collection of special nodes called *orderers*, essentially described as the nodes that participate in the consensus protocol, ordering transactions into blocks to ensure the consistency of the blockchain. The system is maintained by one or multiple Organizations which are the entities that own the peers, and these organizations can join the network and are identified and authenticated by the *Membership Service Provider (MSP)*, which is an interface component that validates, verifies and generates the certificates of the organization's clients and peers, usually implemented as the organization's *Certificate Authority (CA)*.

Since the CA is responsible for creating X.509 certificates, which include both public and private keys, along with additional attributes, provides Hyperledger Fabric the support

of attribute-based access control for selected network members to communicate with data isolation and confidentiality.

In Hyperledger Fabric, a ledger comprises two distinct but related components. Put succinctly, there is the **world state** or state database, which is a database that holds current values, expressed as key-value pairs to optimize the efficiency of data retrieval and querying by *chaincode*, allowing for quicker access to current values. Additionally, there is the **blockchain**, which is a transaction log that records the transactions that have resulted in the current world state, as outlined by Hyperledger Fabric [25].

Different organizations communicate along abstract relations called *Channels*, where each channel presents a different ledger in order to keep data shared only with peers of that same channel [26]. Note that every peer node within a channel keeps a copy of the ledger of the channel, which is updated with each new block, however, orderer nodes only contain the blockchain portion of that ledger, excluding the state database. In essence, while the ledger is physically hosted on peer nodes, it is logically hosted within a channel.

Although Hyperledger Fabric supports pluggable consensus protocols to tailor to a particular deployment or solution, the default protocol for a production network is the *RAFT* protocol. It uses a "leader and follower" model, in which a leader is dynamically elected among the ordering nodes in a channel, replicating the transactions to the follower nodes [19]. As previously noted, it is a *CFT* consensus protocol since it sustains the loss or failure of nodes, including leader nodes, as long as a majority of nodes continue to function, establishing high availability for the ordering service. At the time of developing the solution of this work, Hyperledger Fabric did not natively support a *PBFT* consensus protocol, however, the developers claimed that its integration was in progress and that the *RAFT* protocol was the first step towards it [27]. In fact, Hyperledger Fabric has recently announced its support for a PBFT consensus protocol.

With the presented explication, it is now simpler to explain an overview of the Fabric's transaction flow:

1. A client application sends a transaction proposal to one or multiple endorsing peers.

2. The endorsing peers verify the correctness and execute the proposed transaction returning to the client a proposal response, that includes a read-write set of the proposed changes to the ledger.

3. The client application, after receiving the responses, verifies if they satisfy the endorsement policy and read-write sets, and transmits the endorsed proposal to the Ordering Service, which orders the transactions, combines them into blocks according to the channel, and broadcasts the blocks to all peers of the corresponding channel.

4. Each peer in the network validates the transactions by checking the endorsement policy. If the endorsement is not satisfied, the transaction is marked as invalid,

Table 2.3: Comparison between Bitcoin, Ethereum and Hyperledger Fabric.

	Bitcoin	**Ethereum**	**Hyperledger Fabric**
Restricted access	Public	Public or permissioned	Permissioned
Purpose	P2P online payment system	Business to customers	Business to Business
Consensus Protocols	PoW	PoS (recently)	Pluggable protocol
Governance	Developers and Community	Ethereum Developers	Linux Foundation
Cryptocurrrency	Bitcoin	Ether	None
Smart Contracts	Not supported	Supported	Supported
Speed of Transactions	Low	Medium	High
Transactions Cost	Yes	Yes	No

otherwise, if the transaction is valid, it is committed to the ledger and broadcasted to all peers in order to produce the same updated state of the network in all peers.

2.3.4 Blockchain Platforms Comparison

Table 2.3 depicts a comparison between the blockchain platforms presented: Bitcoin, Ethereum, and Hyperledger Fabric.

This page was intentionally left blank.

3

State-of-the-Art

This chapter aims to present some of the studies found relevant to the topic of study, starting by, in section 3.1, detailing the chosen Review Methodology, the Rapid Literature Review, and how it was performed, to provide adequate and necessary knowledge from the solutions already researched on the topic. In the following sections, it will be presented the analysis and review of the literature found and the respective solutions that best fit our research questions, with section 3.2 explaining overall blockchain-based solutions to record trustworthy documentation and section 3.3 focused on studies, articles, and projects that explore blockchain technology to produce immutable and credible documentation to the automotive industry. Finally, section 3.4 presents a summary of the research performed.

3.1 Rapid Review

A Rapid Review (RR), also referred to as Rapid Literature Review, is a research methodology to identify, examine, evaluate, and clarify all the available findings, evidence and analyses of other authors on a specific research question or questions, topic area, or phenomenon of interest [28]. In our work, a state-of-the-art rapid literature review was performed to guarantee rigorous and detailed research on this study's topics, providing the necessary background on the subject by obtaining relevant evidence about the available technologies as well as limitations in current research. Later in this chapter, in subsection 3.3.1 we will explain how this literature review also encompasses elements and adheres to guidelines from a Multivocal Literature Review [29].

To conduct this RR, Parsifal was used. Parsifal is a web tool that supports researchers, by helping with planning and managing rapid and systematic literature reviews.

Since a RR can be considered a lightweight, more streamlined version of a Systematic Literature Review (SLR), we followed some of the guidelines typical of this latter type of literature review [30], to ensure the most precise and aligned review possible. This way, the review was divided into three stages, where sub-processes took place as well.

1. Planning the review:

- Specifying the need for a literature review;
- Identifying and detailing the research question(s);
- Developing a review protocol;
- Evaluating the review protocol;

2. Conducting the review:
   - Identifying research;
   - Selecting studies with inclusion and exclusion criteria;
   - Assessing study quality;
   - Extracting and monitoring data;
   - Synthesising data;
3. Reporting the review:
   - Specifying dissemination mechanisms;
   - Formatting the main report;
   - Evaluating the report;

3.1.1 Planning the Review

At the beginning of this phase, we identified the specific problems that we wanted to solve with this work and the objectives to reach with this review. These problems are the ones presented in section 1.3 and section 1.4. At the end of this review, we expect to have a complete study of methodologies, practices, techniques, and tools on the usage of blockchain technology to control and manage the immutability and trustworthiness of documentation, preferably concerning the vehicle's market and its proper evidence, particularly within the classic car domain.

3.1.1.1 Population, Intervention, Comparison, Outcomes and Context (PICOC)

Before defining our research questions, we identified the PICOC components, as suggested in [31]. These are next described:

- **Population** in which the evidence is gathered: Research publications utilizing at least one of the interventions in the context mentioned;
- **Intervention** applied in the empirical study: Utilize blockchain technology for documentation purposes within the automotive sector, particularly focusing on the domain of classic cars;
- **Comparison** which the intervention is measured against traditional documentation management;

- **Outcome** of the experiment: A review and study of methodologies, practices, techniques, tools, and procedures that define the current state-of-the-art in the mentioned area and context;

- **Context** of the study: The study is carried out in the context of documentation management, blockchain technologies, and the car restoration industry.

3.1.1.2 Research Questions

Defining the research questions is a crucial step in planning any literature review to later help the researchers identify primary studies that assess these questions, apply the criteria of inclusion, and extract and analyze the necessary data to answer them.

The following **research questions** were formulated in order to obtain the most adequate answers possible through our review:

- **RQ1**: How is blockchain technology used for documentation and evidence purposes?

- **RQ2**: How is documentation stored in trustworthy systems that are not prone to tampering?

- **RQ3**: How is blockchain technology used in the vehicle industry?

- **RQ4**: Is blockchain being explored for documentation purposes in the vehicle industry, especially in the classic car domain?

- **RQ5**: Which network types and blockchain platforms are used in evidence documentation?

3.1.1.3 Keywords

To better plan our review and, taking into account the formalized questions, we identified the next **keywords** together with synonyms:

- Blockchain, Cryptographic Ledger, Distributed Ledger

- Documentation System, Evidence Documentation, Product History

- IoT, Internet of Things, Internet of Everything

- Automotive Industry, Vehicle Market

- Classic Cars, Classic Vehicles, Vintage Cars

- Car Documentation, Vehicle Documentation, Car Evidence

3.1.1.4 Develop a Review Protocol

A review protocol outlines the methods that will be followed in a specific literature review and it is necessary to reduce the potential researcher bias [30].

We decided to select Scopus as our main database source where the desired searches will be performed due to its larger index of publications, encompassing a broad spectrum of journals and conference papers [32]. Additionally, we employed Google Scholar as a secondary resource to ensure a broader coverage of studies.

3.1.1.5 Search Strings

To execute these searches, the keywords identified previously were combined and the next search strings were obtained:

- ("Blockchain" OR "Cryptographic Ledger" OR "Distributed Ledger") AND ("Documentation System" OR "Evidence Documentation")
- "Blockchain" AND ("Trustful Documentation" OR "Trustworthy Documentation" OR "Evidence Documentation")
- "Blockchain" AND ("IoT Data" OR "Industrial IoT Data")
- "Blockchain" AND ("Vehicle Market" OR "Automotive Industry")
- "Blockchain" AND ("Car Documentation" OR "Vehicle Documentation" OR "Car Evidence")
- "Blockchain" AND ("Classic Car" OR "Classic Vehicle" OR "Vintage Car")

The identified search strings can be divided into two different, but related research categories: the first three strings were used to obtain a broader collection of studies to better analyze the state-of-the-art technologies that use blockchain to produce trustworthy documentation and the last three strings were used to acquire specific studies and articles that connect the blockchain technology to the automobile sector and market.

In order to better explain the review protocol, the remaining parts of it are precisely detailed and elaborated in the next subsection. The results from each step are also available for a more explicit view of our protocol.

3.1.2 Conducting the Review

3.1.2.1 Identification of Research

Before starting the actual review process, some initial searches were performed, in both Scopus and Google Scholar to check if the identified keywords produced the desired results. The keywords selected seemed to be appropriate for our search and when intersected, some relevant studies and articles on the research topic were already picked for further

analysis. From Google Scholar, only peer-reviewed articles, published by major publishers, were considered for inclusion in this review.

Then, the search strings previously mentioned were executed in our main database, Scopus. With these searches' results and the ones selected from the preliminary searches, mentioned in the last paragraph, and after a brief analysis of their titles, abstracts, keywords, and publish date, a total of 49 studies were collected.

For our selection not to be limited by the results found automatically by the mentioned search databases and consequently, so that the knowledge acquired in certain topics is not limited either, together with the rapid literature review we decided to additionally use the snowballing approach which consists in using the reference list of an article or the citations to the article to identify additional relevant papers, articles, and studies [33]. Both backward and forward snowballing were executed, which respectively consist of searching the citations that were used to build a study's case to find new ones and to search the studies that reference the one in question.

By applying the snowballing method we obtained 19 more studies, resulting in a total of 68 articles gathered. Regardless of whether the studies were originally obtained by executing the search strings or by using snowballing, the subsequent phases consisting of study selection and quality assessment were conducted identically.

3.1.2.2 Study selection with inclusion criteria

A more in-depth review of the abstracts and documents' overall structure of the selected studies was carried out, to determine whether or not they were relevant to our research. To decide if a study should be included in our selection, the following inclusion criteria must be met:

- Be written in English;
- Be a primary study;
- Have at least six pages;
- Have been published in the last eight years;
- Match at least one of the literature review objectives;
- Be the most up-to-date and comprehensive version of the document;
- The study must be grounded in practical, real-world applications.

After the execution of this first thorough selection, a total of 25 studies were obtained. The criterion imposing the most significant limitation was the mandatory requirement for a study to be considered a primary study, given the abundance of secondary research and literature reviews in the overall selection.

3.1.2.3 Study quality assessment

In this stage of the literature review, the studies acquired in the previous phase were thoroughly reviewed and analyzed with the objective to evaluate the quality of each one of them, and for that purpose, a score was given to each study according to how appropriate the answers to the following questions are presented:

- Were the goals and research questions clearly defined in the study?
- Was previously published related work exposed and compared with the research results claimed in the study?
- Was the research design clearly outlined?
- Were the limitations of the study evaluated and identified?
- Was the data used for validation described in sufficient detail and made available?
- Were answers to the research questions provided?
- Were negative or unexpected findings reported about the study?

Each question was answered on an ordinal scale with either "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree" or "Strongly disagree", each associated with unitary scores ranging from 4.0 to 0.0. The cumulative weights of each study were calculated and subsequently compared against a predefined threshold of 20.0. Studies scoring below this cut-off value were excluded from further analysis.

After this phase, a total of 9 studies found relevant to our topic and field of research were accepted. It is also important to notice that, although some of the studies were not selected, they could have been analyzed to better understand topics related and explained in chapter 2.

Refer to the diagram depicted in Figure 3.1 for a visual representation of the formal literature review, showcasing the number of studies obtained across the various phases of this review.

3.1.2.4 Data extraction and monitoring

From the studies that remained after the quality assessment phase, the following data items were extracted: Citation, Year, Venue, Title, Publication, Number of Citations, Core Rank, Scimago Rank, Methodology, Blockchain Platform Used (*Bitcoin*, *Ethereum*, *Hyperledger Fabric*, other), Blockchain Network Type (Public, Private, Consortium), Consensus Protocol Used (*PoW*, *PoS*, *PBFT*, other), Blockchain Architecture and Implementation Description, support for Off-Chain Storage, and implementation of a proper Access Control Scheme.

This information was then assembled and organized in a structured manner to facilitate the subsequent synthesis of the results. Ensuring that we accurately gathered the necessary

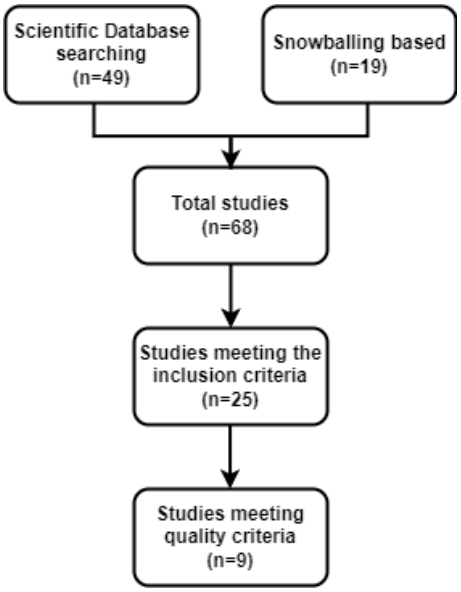


Figure 3.1: Diagram of the studies obtained through our literature review.

information from all of the studies was an essential aspect of conducting this literature review.

3.2 Generic Primary Studies

Blockchain technology and its unique features have recently captivated the interest of stakeholders in several sectors and industries, such as finance, healthcare, real estate, logistics with IoT support, supply chain, especially in the luxury market and food supply chain, and finally, the government sector. In fact, there are some interesting concepts, platforms and systems, that utilize blockchain technology in the sectors mentioned, which are being used daily, bringing many benefits to all the involved entities.

In this section, it will be reviewed the literature that presents a novel implementation that uses blockchain technology to facilitate, store, and generate trustworthy documentation, but not necessarily related to the automotive industry. These studies share common requirements, encompassing process tracking, trusted recording, and solutions to record data that is not adequate to store on the blockchain, such as large-volume files (e.g., photos, videos, and various documents). To align with our focus on vehicle restorations and address our research questions, this exploration draws parallels with these solutions from different industries, such as healthcare and supply chain management.

We acknowledge that there may exist a larger corpus of articles exploring blockchain technology for documentation purposes. However, given the Rapid Review (RR) nature of our literature review, the selection process prioritized those most aligned, with similar requirements and results with the work expected to be developed in this research.

Zhong et al. [34] proposed a framework for using blockchain technology to manage the impact of hospitals during the COVID-19 pandemic, tracking the fulfillment of crucial

infection control measures in the design, construction, or renovation, and operation stages of hospitals. In a COVID-19 emergency, the proposed framework can ensure an immutable, tamper-proof record of events and confirm the authenticity and accuracy of records. In the architecture of this framework, Hyperledger Fabric is the platform selected to implement the blockchain network, mainly due to the data privacy requirements and permission mechanisms that are necessary for the healthcare industry. The blockchain is a consortium blockchain, where there are several organizations, with several members each, that have different permission levels, interacting together. This framework is also supported by off-chain cloud data storage to store larger data files (e.g., design files and IoT sensor data) that are not suited to be recorded in the blockchain. However, the hash value of these files is stored on-chain, to guarantee that the files are not modified.

Another relevant solution in the healthcare sector is the one presented in [35]. The authors proposed a lightweight blockchain architecture for managing healthcare data. This solution addresses the need for reduced computational energy consumption and increased transaction throughput compared to public blockchain networks by dividing the network participants into clusters and maintaining one copy of the blockchain per cluster. Additionally, a *PBFT* protocol is used to achieve consensus on the ledger. As patient data is sensitive, the authors opted for a permissioned blockchain network that limits network participation to authorized members. However, this architecture does not support smart contracts, a significant limitation compared to Hyperledger Fabric or the Ethereum networks. While this solution offers several benefits, such as reduced computational energy consumption and increased transaction throughput, its inability to support smart contracts limits its potential applications. Nonetheless, it presents an interesting option for healthcare organizations that require a permissioned blockchain network to manage their sensitive patient data.

Zhang et al. [36] proposed a system architecture to track and store data of all the operations involved in the entire grain supply chain based on blockchain technology, in order to guarantee food quality and safety-process traceability. Compared to traditional systems, this system presents a major improvement in data security, reliability, information interconnection, real-time sharing of information, and trustworthy tracking. The authors designed a multimode storage mechanism to improve blockchain storage efficiency. The blockchain network, developed with the Hyperledger Fabric platform, ensures that the information is traceable, credible and supports the execution of smart contracts to implement the business logic. Moreover, each of the blockchain's nodes also has an off-chain database component in order to solve the problem of blockchain's insufficient storage capacity when there is the need to store large amounts and varied types of data or when the data to be stored does not need a consensus to be reached.

In [37], the authors proposed a smart contract-based supply chain framework using a permissioned blockchain network. This framework was optimally designed to be applied to the US beef cattle industry but can be adapted to other supply chains with minimal modifications. This system ensures that all operations along the supply chain are recorded

in the blockchain, to guarantee that the generated data is immutable, confidential, and tamper-proof. The proposed solution is based on the Ethereum blockchain platform and presents a P2P network, where each of the nodes, which can be executed by the business owners, contains not only the blockchain itself with transaction data from all the farms with all the animals registered on the system and smart contracts bytecodes, but also needs to have a local database service to store raw farm data, such as animal inspections, vaccinations, and other relevant information. When there is an update on the local database, a cryptographic hash of all the farm's raw data combined is computed, which is then recorded in the blockchain via the execution of a smart contract specially designed for that purpose. Furthermore, the system implements an access control policy and enables animal trading between users, changing the ownership of the animal from the old owner to the new one.

Although in a completely different sector and with a completely distinct set of use cases, the technologies presented in all these studies can serve as an inspiration for the work that we are developing.

3.3 Blockchain-based Vehicle Documentation Solutions

To be more precise about applications of blockchain-based solutions applied to the automotive industry, there are few relevant studies available in the literature that utilize blockchain technology to store and secure trustworthy information about vehicles. In fact, a significant amount of the studies on this topic found in the literature are not peer-reviewed. Others are oversimplified proofs of concept that do not have any kind of real application. There were also some studies found that were based on blockchain platforms or technologies that are no longer properly working, due to discontinuing some dependent components or because the development team disbanded. Furthermore, some of the published literature is being applied in several areas of the automotive industry beyond the scope of our research, such as in enhancing communication in autonomous vehicles. Those other applications are not reviewed here.

3.3.1 Grey Literature

As mentioned at the beginning of this chapter, this literature review can also be categorized as a Multivocal Literature Review, since not only formally-published scientific literature was analyzed but also included grey literature, which involves the review of websites, blogs, white papers, and proprietary solutions [29]. By adopting this multivocal approach, our review bridges the gap between pure scientific research and industry projects, incorporating cutting-edge solutions applied in real-world applications.

falar aqui em como usamos o Google e fizemos as pesquisas

In the upcoming paragraphs, we will present some projects, that although not found directly in the formal literature, as are included in grey literature and rely on proprietary

implementations, have a similar scope to ours. These are recent and relevant projects that have been launched to explore and produce real-world practical applications that utilize blockchain technology in securing immutable vehicle documentation.

Although not directly related to our research, as it does not include any type of important vehicle details and relevant documentation, we find it interesting to offer a concise overview of the earliest steps in applying blockchain technology in the automotive industry, that to a limited extent represent a form of digital documentation of vehicles. One of the first ways that blockchain technology was applied in the vehicle industry, was with the *tokenization* of luxury cars. Each token, commonly referred to as Non-Fungible Token (NFT), is backed by a real specific luxury car. Similar to investing in stocks, individuals can buy these tokens, which may represent the entire ownership of the vehicle or just a percentage of it, becoming co-owners of the underlying cars, hoping their value will increase and this way also increase the matching NFT's value. One example of this *tokenization* platforms, is *CurioInvest*, a platform where investors can purchase tokens representing fractional ownership of rare and exotic cars, without the full financial commitment.

VINchain is a blockchain-based solution to transparently and securely record important information about vehicles. Clients of the platform can access the data stored on the ledger on a specific vehicle, such as insurance plans or accident history, using its Vehicle Identification Number (VIN), which is essentially a unique identifier of the vehicle (there are not two operating vehicles with the same VIN). The permission to access this information is granted by exchanging *VIN tokens*, which consist of the cryptocurrency of this solution. This cryptocurrency can fluctuate in value and this way alter the cost of obtaining access to a vehicle's data. This platform enables other organizations, such as insurance companies and repair shops, to be participants in the network. One limitation of this implementation is that anyone with access to a vehicle's VIN and willing to pay to get the data associated with that VIN will have it. In contrast, even the owner of the vehicle would have to pay to get information about his vehicle. Furthermore, participants of the network, have equal rights to access and interact with all the data. This brings some data privacy concerns since some vehicle data may want to be kept confidential to avoid undesired behaviors from malicious participants. To the best of our knowledge, *VINchain* has not been receiving continual updates, as there has been an absence of new car records and new information on the web platform for the past 2 years. Furthermore, certain functionalities within their solution are currently non-operational or inaccessible due to unresponsiveness from the host.

There are also initiatives from big companies in the vehicle industry, such as Renault and BMW with *XCEED* and *VerifyCar* respectively, but these are mainly focused on creating a secure, transparent, and trusted network throughout the supply chain to vehicle manufacturers. However, to our best knowledge, it is not possible to find details of the implementation or significant information about the projects and it seems that they were abandoned since there have not been recent updates.

To finish the review of commercial implementations with a similar scope, it is also crucial to present *The Motor Chain*. *The Motor Chain* is the platform most similar to our solution since it is focused on the documentation of classic cars with the goal of increasing the monetary value of the registered vehicles. It is advertised as a blockchain-based platform where classic car owners can register their vehicles and any updates on the documentation performed by the owners or professional entities (e.g. maintenance services, restoration shops, appraisal entities) are reflected in a timeline of the vehicle documentation. *The Motor Chain* claims the use of Hyperledger Fabric as its chosen blockchain platform to record all the desired information, however, due to its proprietary nature, there exists no accessible means to verify the actual usage of this platform or the technology. Furthermore, there is still insufficient information accessible on the access management mechanism and therefore insufficient credibility of the provided information, and the cars' owners have to pay in order to register their vehicles in the system. Another point of consideration lies in the platform's primary focus on a basic timeline of events of a classic car, such as exhibition events and summaries of restoration procedures, and does not follow the guidelines and best practices expressed in FIVA's Charter of Turin, providing thorough details of the vehicles and its restoration procedures. To the best of our knowledge, this system lacks the capability to incorporate other types of media evidence besides photos, and, in fact, there is no information available regarding media storage. Nonetheless, *The Motor Chain* shares similar requirements and goals with ours, contributes to reinforcing and corroborating the need for our work's development, and aids us in identifying certain challenges and issues that we aim to address.

3.3.2 Formal Scientific Literature

Proceeding to the actual formal scientific literature reviewed on the automotive context, Y. T. Jiang and H. M. Sun [38] proposed a trusted vehicle data source system based on blockchain technology. This system was developed to provide transparency and immutability to the second-hand vehicle market. The proposed system uses a consortium blockchain network that is built on the Ethereum platform. The developed blockchain platform uses a *Proof of Authority (PoA)* consensus protocol, which is similar to *PoS* but instead of using the nodes to generate new blocks based on the amount of staked cryptocurrency, transactions are verified and combined in blocks by nodes that are selected as credible entities, so in the aspect of transactions' validation, it is also similar to *PBFT*. Figure 3.2 presents a diagram of the life cycle of a vehicle in the system. This solution provides access control management to support different levels and sets of permissions to access or interact with some specific data stored on the ledger, since there are several intervening entities, such as repair shops, government branches, car owners and second-hand vehicle dealers. The vehicle's VIN is also used as a unique identifier of the car in the system, however, having access to the VIN of a vehicle does not grant access to the vehicle's data record since it is also necessary to have a token provided by the car

dealer. This token will expire after a desired period of time. Since the blockchain platform opted for is Ethereum, every write operation to record data on the ledger will have an associated cost as a transaction fee in the native cryptocurrency *Ether*. Eventual variations in cryptocurrency price will affect the cost of performing these transactions [39].

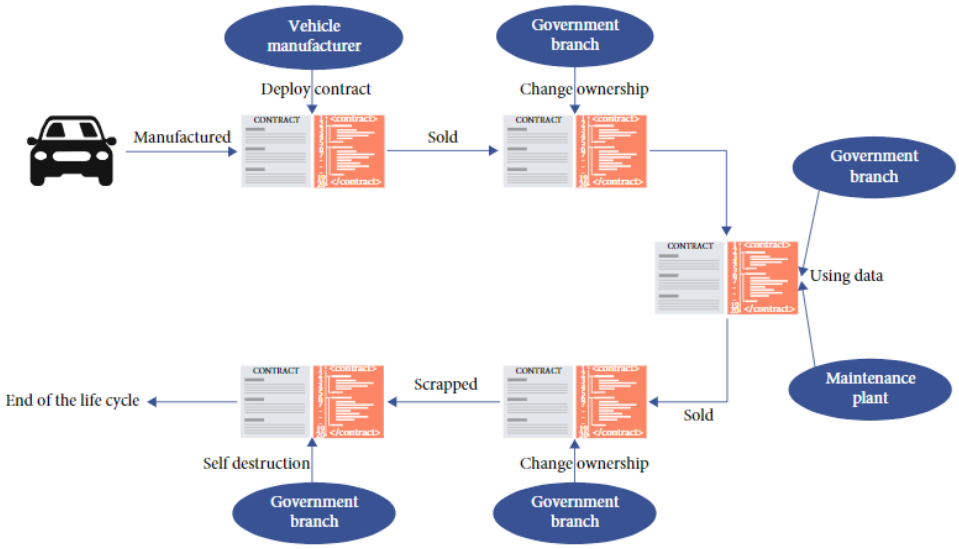


Figure 3.2: The life cycle of a vehicle in the system proposed in [38].

In [40] and [41], both articles from the same authors, it was proposed a vehicle data and processes ledger framework to allow collaboration, secure sharing of vehicle data and maintenance history between several stakeholders over a consortium blockchain. The objective of this system is to provide a transparent history of a vehicle through its lifecycle, from the initial automotive manufacturer to the car owner, registering all the operations and data (e.g., repair, maintenance, accidents). The vehicles' information is recorded on the ledger using smart contracts, which the authors designated as *car books*, linked to the car by its VIN. Access to a vehicle's data is restricted first to the manufacturer, car owner, and brand official repair shops. This access can then be also conceded to other entities, such as third-party maintenance shops or insurance companies. This solution mixes blockchain technology and classic databases for storage purposes, with information that needs to be secured and transparent to all participants being stored on the blockchain, and complementary data or member-specific data stored in local databases. The blockchain platform chosen for this implementation was *Quorum*, an open-source project derived from the Ethereum platform, customized to use in permissioned blockchain networks. Quorum enables control nodes to access the ledger using a whitelist, validates blocks using a specially designed consensus protocol that does not require mining, and removes the cost of executing transactions on the blockchain. It also supports a modular architecture. It is possible to discern that Quorum is similar to the Hyperledger Fabric platform and both can be applied in similar use cases, however, while in the latter, the smart contracts can be written in general-purpose programming languages, in Quorum the smart contracts

Table 3.1: Comparison of formal related work on the automobile context.

Work	Author(s)	Year	Purpose	Blockchain category	Blockchain platform	Transaction fee	Documentation detail	Applied to Classic's Industry
[38]	Y. T. Jiang and H. M. Sun	2021	Second-Hand vehicle market	Public	Ethereum	Yes	Medium	No
[40, 41]	K. L. Brousmiche et al	2018	Vehicle's history throughout its life cycle	Consortium	Quorum	No	Medium	No
[42]	K. Louchoomun, S. Pudaruth and S. Kishnah	2020	Prevent falsification of mileage	Consortium	Hyperledger Fabric	No	Low	No
[43]	G.Submanian and A.S. Thampy	2021	Lifecycle of electric/smart vehicles	Consortium	Ethereum	Yes	Low	No

must be written in *Solidity*, the same programming language as Ethereum.

In [42], the authors proposed a blockchain-based smart contract application to avoid some of the common challenges of the automotive industry, such as tampering with vehicle information and falsification of mileage. With the proposed solution, the vehicles' data is transparent, accurate, and immutable while also offering the possibility to streamline some of the business processes, which leads to an increase in trust among the customers and an improved speed and efficiency of the vehicle registration process while minimizing the need for paperwork and reducing costs for buyers. This solution's architecture consists of a Hyperledger Fabric blockchain network together with *Spring Boot* framework for the server-side application. To the best of our knowledge, the proposed solution is a proof of concept, and there was no validation or analysis of the results performed by the authors. However, the authors performed interviews with several stakeholders of the automotive industry and surveys to collect information and opinions that corroborate the need for our research, namely the lack of trust towards sellers and the general agreement that blockchain technology would be an upgrade to the sector that the interested entities are willing to endorse.

Another relevant solution was introduced by G. Subramanian and A. S. Thamp [43]. The authors proposed a consortium blockchain solution, similar to the ones already presented, but focusing on the electric car market, where the major actors consist of the vehicle manufacturer, charging station, and battery manufacturer. Since electric vehicles and their charging stations have more software support, smart IoT sensors can be used to enable automatic interaction with the blockchain and the overall system, such as recording vehicle mileage during charging. This solution also supports the transfer of vehicle ownership. *Ethereum* serves as the blockchain platform for this solution, with each transaction incurring an associated cost.

Table 3.1 provides a concise overview of the formally reviewed scientific literature within the automotive industry, highlighting the contributions and limitations of each work and offering a comparative analysis.

3.4 Summary of the Research

Through our literature review, we identified and understood the technologies and tools needed to tackle similar issues, enabling us to draw certain conclusions. Even though certain literature examined was used in various fields, not specifically relevant to the automotive industry, or not formally published due to the multivocal approach of this review, it is vital in understanding cutting-edge technologies and can serve as a source of inspiration for our work. All the research questions posed previously in this chapter have been promptly addressed. This process has provided valuable insights into the context and current research related to documentation within the automotive sector, with a specific focus on classic vehicles. It was also possible to conclude that consortium is the network type preferred to record reliable documentation and that Hyperledger Fabric and Ethereum, or derivations from it, are the favored blockchain platforms.

Some relevant studies, projects, and initiatives specific to the automotive sector aim to preserve vehicle data, documentation, and overall history using blockchain technology. However, none of these have been adequately tailored to meet the specific needs and demands of the classic automobile context. These solutions primarily target the conventional automotive market, and the recorded data is typically plain and straightforward, like maintenance logs. The solution most tailored to the classic vehicle industry is *The Motor Chain*, however, it offers only a relatively limited timeline of events with no minute details and presents other issues identified previously, such as uncertainty about the authenticity of the information and the entity responsible for its registration. Both the owners of classic cars and the certification authorities, along with the restoration procedures, require a higher level of meticulousness, namely a precise and detailed record for every stage of the restoration process. Proper documentation outlining each step of the restoration process, including supporting evidence and the responsible parties, grants authenticity that, as for most works of art, is highly considered for determining the market value.

The lack of trust and transparency and the necessity to provide rigorous information in the classic car sector is the problem we are trying to tackle with our system, offering a unique solution to record every detail of a classic car, every restoration procedure, with all related pieces of evidence, like media files or any other type of vehicle documents (e.g., registration certificates or inspection reports), together with an appropriate access control mechanism, capable of effectively handling diverse scenarios within the classic car industry, to benefit all the entities involved and streamline some of their tasks. As expected, addressing these problems in this sector entails specific challenges, which is something that this work aims to overcome, namely how to best structure and configure the blockchain network to align with the classic car ecosystem, specifying all the intervening organizations and their corresponding access levels.

4

System Proposal

This chapter aims to present the conceptualization and solution proposal of this work. Firstly, in section 4.1, the software requirements, the stakeholders, and use cases will be presented. Afterwards, in section 4.2, the proposed system's architecture, its components, and the technologies used are described.

4.1 Software Requirements Engineering

Software Requirements Engineering (SRE) is the disciplined application of proven principles, methods, tools, and notations to discover, analyze, and document the services and constraints that a system should provide, based on the needs of customers and stakeholders [44]. The SRE process should occur before any actual designing, coding, testing, or maintenance takes place.

We started by defining who is affected and who can affect our system, through the identification of the distinct stakeholders, which not only have different roles and requirements (functional and non-functional) but also have different levels of access and permissions to interact with the system [45].

Requirements elicitation drew upon the valuable expertise of industry specialists. This included renowned classic vehicle collectors, managers from reputable restoration shops, and key stakeholders within the automotive sector.

4.1.1 Stakeholders

In our project, we have identified these four stakeholders:

- **Classic Car Owner**: The stakeholder that owns a classic vehicle and registers it in the system with all the desired associate documentation. Can also share this documentation with other stakeholders.

- **Certified Restoration Shop Administrator**: The stakeholder responsible for performing most of the restoration or conservation procedures in the classic cars registered in the system.

- **Certification Body Officer**: The stakeholder that is accountable for reviewing the documentation and issuing certificates for registered classic cars.

- **Potential Buyer**: The stakeholder who can be granted permission to visualize the documentation of specific cars to verify their authenticity and the procedures carried out to restore them.

4.1.2 Functional Requirements

Functional requirements refer to the features or services that the developed system must incorporate to achieve its goals. These are statements that describe how the system should behave in certain situations and how it should respond to certain inputs [46].

The system's functional requirements are represented through a Use Case Diagram shown in Figure 4.1, and are going to be thoroughly detailed in the following subsections. For simplicity, the authentication use cases are not shown explicitly, but it is important to note that every single actor in the system requires prior authentication.

4.1.2.1 System Requirements

The system requirements go beyond specific user requirements and result from general functionalities. They describe what the system should be capable of performing by itself. The following system requirements have been identified:

- Allow user authentication through credentials.

- Enable smooth communication and seamless integration among system components.

- Maintain up-to-date vehicle information and complete documentation of processes, pieces of evidence, and any additional details.

- Automatically update the car's documentation's chronological modification log with all executed changes.

- Ensure vehicle information is only available for authorized users with granted permissions.

- Maintain up-to-date records of accesses and corresponding types that different users have on a given car.

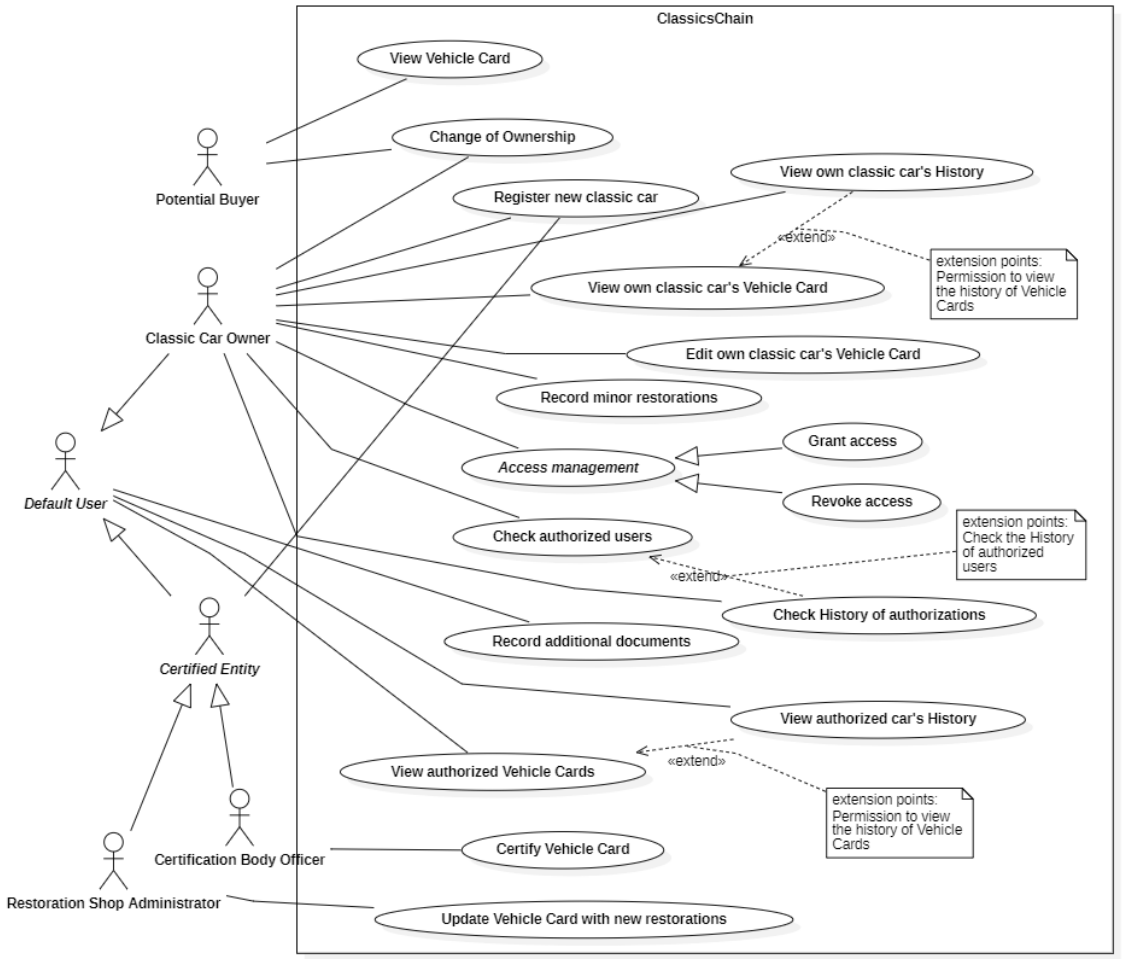


Figure 4.1: Use Case Diagram of the Blockchain-based Documentation System

4.1.2.2 Classic Car Owner Requirements

The Classic Car Owner is the primary stakeholder in our project, who can be regarded as the normal user of our system's implementation. The following requirements for a Classic Car Owner were identified:

- Registers an account with the normal user role;
- Logins as user;
- Resorts to a certified restoration shop or certification body to register a classic car in the system;
- Controls the access to the information on its owned classic cars;
- Checks all the details, documentation, pieces of evidence, restoration procedures, certifications, and any other information about its owned cars, at any time with no restrictions. We named this web portfolio as *Vehicle Card*;

- Checks the chronological update log/history of modifications of any information of their owned vehicles, along with information on the individuals or entities responsible for these modifications. The system creates this log when the vehicle is first registered and uses it thereafter;
- Edit fundamental details of their vehicles, such as the year of manufacture, license plate, or engine number, in case there is some sort of error;
- Adds minor maintenance procedures to the vehicles' documentation, such as change of tires or oil;
- Update the information details and related pieces of evidence on minor restoration procedures previously made by the user. Restoration procedures applied by other users, being restoration workshops, previous owners, or any other entity with write permissions cannot be altered by the owner, only by those who originally made these interventions;
- Adds extra documents to the *Vehicle Card*, such as inspection reports, exhibition awards, or any other document that the owner finds relevant and that can potentially increase the vehicle's value and reputation;
- Grants or revokes, at any time, three different types of access permissions to other users according to the desired role: viewer, modifier, and certifier. These permissions can also be checked at any time;
- Check the chronological log/history of prior owners, authorized users, and the corresponding access level of their cars. This log is created and available since the initial registration of the vehicle in the system;
- Changes the ownership of its cars to another user and consequently grants complete access to all the documentation, access control, and historical logs to the other user which will be the new owner of the vehicle;

4.1.2.3 Certified Restoration Shop Administrator Requirements

Workshops registered in our system for classic car restoration must comply with FIVA's Charter of Turin guidelines and best practices. All restoration procedures must be meticulously documented and universally recognized within the classic automobile sector. Therefore, we categorize these restoration shops as 'Certified.' The requirements for a Certified Restoration Shop Administrator are as follows:

- Registers an account with the role of restoration shop administrator;
- Logins as restoration shop administrator;

- Registers a new classic car in the system and adds its detailed information provided by the owner;
- Visualizes the *Vehicle Cards* of all available vehicles (i.e. those that were previously granted the *modifier access level* to the restoration shop);
- Visualizes the historical log of modifications of all the available vehicles;
- Adds new restoration procedures to available vehicles, following the FIVA's Charter of Turin, providing a meticulous title and description, as well as other pieces of evidence relevant to that specific procedure, such as photos, documents, or videos;
- Updates the information details and related pieces of evidence on restoration procedures previously made by the workshop. As with classic car owners, only the user who originally made that particular restoration procedure can modify it.
- Adds extra documents to the *Vehicle Card*, such as inspection reports or proofs of restoration;

4.1.2.4 Certification Body Officer Requirements

Several requirements were derived from the interaction between Certification Body Officers and the system. The following requirements were identified:

- Registers an account with the role of certification body officer;
- Logins as certification body officer;
- Registers a new classic car in the system and adds its detailed information provided by the owner;
- Visualizes the *Vehicle Card* of all the available vehicles (i.e. those that were previously granted the *certifier access level* to this Certification Body);
- Visualizes the historical log of modifications of all the available vehicles;
- Marks a specific available vehicle as certified by the Certification Body in question after a review of the *Vehicle Card* and historical log of modifications;
- Adds extra documents to the *Vehicle Card*, such as a proof of certification;

4.1.2.5 Potential Buyer Requirements

Several requirements were derived from the interaction between Potential Buyers and the system. Although Classic Car Owners and Potential Buyers are different stakeholders, with distinct requirements and interests, it is noteworthy that they share the same implementation within the system. Both are considered normal users and must be registered

and logged in as so, with the only distinction being that a Classic Car Owner has at least one owned vehicle registered in the system and a Potential Buyer does not. Importantly, there is no limitation preventing them from concurrently representing both stakeholders, as a user can have their vehicles registered and still check other vehicles' documentation for potential purchase.

It is also important to mention that there are no restrictions on a Classic Car Owner holding additional roles such as a Restoration Shop Administrator or a Certification Body Officer. However, these roles have different implementations, requiring users to be registered and logged in with the appropriate role.

The following requirements for a Potential Buyer were identified:

- Registers an account with the normal user role;
- Logins as user;
- Visualize the *Vehicle Card* and a historical log of modifications of a vehicle that was previously granted the viewer permissions to the specific user. A user's view permission expires after a pre-determined time;
- Ultimately, if the buyer decides to buy a specific car, he gets complete access to the *Vehicle Card* with all associated information and evidence, while the original owner gets his permissions revoked. Naturally, this operation can only be executed by the original owner.

4.1.3 Non-functional Requirements

Non-functional requirements outline the quality attributes of a software system as a whole or its services and functions, focusing on how the system should perform. These include aspects that ensure the system's usability, effectiveness, and overall performance.

Our system must comply with the next non-functional requirements:

- The system must have an adequate response time;
- The system must be resilient, exhibiting fault tolerance to ensure uninterrupted operation, and all components should be robust.
- The Web Platform with Application Programming Interface (API) integration must be easy to use and understand, avoiding a steep learning curve;
- The system must not rely on a pay-per-transaction blockchain;

4.2 Architecture

With the requirements of our project clearly defined, it is appropriate to start explaining the developed system's structure, which relies on these constraints. It is important to

note, that although our project was integrated into the project described in section 2.1, our system mainly consists of an independent solution for all the users, entities, or interested organizations in the classic automobile context who may want to use our solution but are not necessarily associated with the restoration shop where our work was developed.

To facilitate understanding, we first briefly discuss the various components that make up our solution and how they are integrated. Next, the technologies used to implement these components are rigorously detailed.

4.2.1 Components

The component diagram illustrated in Figure 4.2 presents an architectural overview of our system, built upon the requirements and goals previously mentioned. It is an upgraded version of the component diagram originally made for the system in usage at the classic restoration shop, which is described in [5]. The colored components and parts identify the artifacts that were developed during the current work. This system was named *ClassicsChain* to highlight the dual focal points of this project: classic cars and blockchain.

As brought up beforehand, although the component diagram represents this as a subsystem integrated into the repair shop platform already developed by interacting through the Representational State Transfer (REST) API that we developed, it is crucial to understand that it will also function as a standalone system that other entities besides the repair shop will use, namely using the Web Application or, if desired to integrate with other systems, using the API, which we provide for these external entities. Here, our system is illustrated as integrated into the rest of the project to point out that the restoration evidence will be automatically added to the car's documentation on the blockchain ledger when added to the original platform, such as, when a restoration procedure is submitted on the *Charter of Turin Monitor*.

4.2.1.1 ClassicsChain

As previously mentioned, this component was named *ClassicsChain* to elucidate the two main aspects of this project, so it will be referred to as such throughout this document. *ClassicsChain* is the backbone of our project, with all components integrated to properly develop a system for all interested entities.

ClassicsChain web platform allows all entities and car enthusiasts with permission to do so to visualize and update all the information, documents, and evidence available on each *Vehicle Card*. This applies regardless of whether these entities are affiliated with the workshop where this work originated, as long as it adheres to the guidelines outlined in FIVA's Charter of Turin. This way, every participant in the classic car ecosystem contributes to achieving an enhancement in the documentation's quality, thereby adding substantial value to the entire sector, rather than benefiting only a specific workshop.

To better suit the specific needs and requirements of each entity and corresponding users, *ClassicsChain*'s functionalities are also available through a REST API (*ClassicsChain*

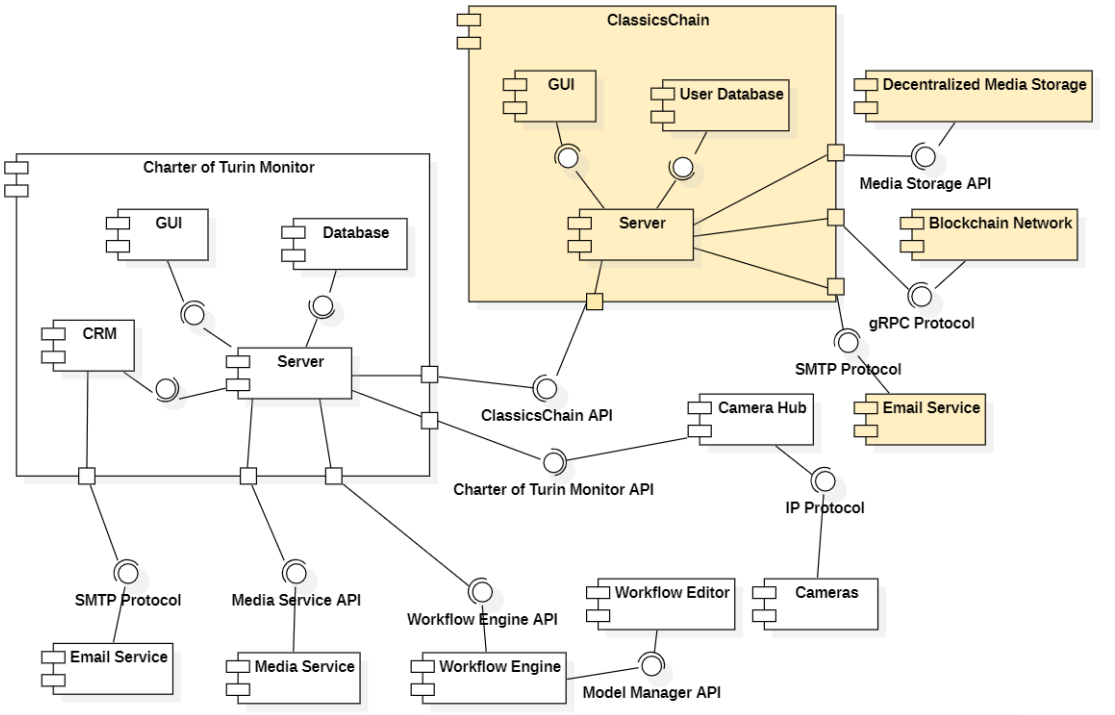


Figure 4.2: Component Diagram of our system, based on the component diagram in [5]. The colored components and artifacts identify the work developed in the current research.

API). In this manner, the entities, such as restoration workshops or certification bodies, can use this API to integrate with their existing platforms, clients, or other APIs. By doing so, it simplifies the process of recording information into our system and streamlines various tasks for them, enhancing the efficiency of their operations.

A user database is also incorporated to facilitate user registration and authentication with typical credentials and decouple the information stored on the blockchain network from the user account details.

4.2.1.2 Blockchain Network

This component constitutes the live production blockchain network, comprising essential elements such as organizations, their respective nodes, the associated consensus protocol, and the deployed smart contracts. Unlike a blockchain test network deployed on a single local environment, this blockchain network operates as a production network deployed on the cloud. The logic and functions embedded within the smart contracts are accessible for utilization in the *ClassicChain* application with API integration.

The configuration, development, and deployment of our blockchain network will be further detailed and described in subsection 5.1.1.

4.2.1.3 Off-chain Media Storage

An off-chain decentralized media storage component is crucial in our system, aligning with the decentralized computation and storage principles inherent in both the system and its core technology, which is blockchain. Blockchain technology is not suitable for high-volume data, such as media files. This is precisely the purpose of the off-chain media storage component, which is to efficiently manage media files such as images of technical documents, photos, videos from restoration procedures, and various documents like inspection reports or warranties for specific car parts. The hash or unique identifier of the content is securely recorded on the blockchain.

The technology chosen for this media storage is better described in subsubsection 4.2.2.6 and implementation details in subsubsection 5.1.3.4.

4.2.1.4 Email Service

Lastly, the *Email Service* component is the service that will be used by the *ClassicsChain* system to enable custom emails to be sent automatically, informing users about specific operations that took place, such as a user registering its account for the first time, the granting of access to a particular car to that user, and other pertinent operations.

4.2.1.5 Charter of Turin Monitor

As previously mentioned, the *Charter of Turin Monitor* is the platform that was developed before our work but is still within our project [5]. It serves as the Graphical User Interface (GUI) that Raimundo Branco workshop employees use to interact with the restoration processes and classic car owners use to check the progress and details of restoration procedures. Some modifications were made to this system to integrate with *ClassicsChain*. This was made through the *ClassicsChain API* mentioned previously. With the integration completed, some of the functions performed in the *Charter of Turin Monitor* are automatically reflected to *ClassicsChain*. A more detailed explanation regarding the changes that took place to this element can be found in section 5.2.

There were also some other minor modifications and bugs fixed that, although unrelated to *ClassicsChain*, are important for the continuous project's progression, ultimately increasing suitability to the workshop's customer demands and overall vintage car sector. Some of these adjustments include minor updates to the GUI to make it more visually appealing and added information to the automatically generated PDF. For more information about the rest of the project, please consult section 2.1 in this document or refer to the respective original documents [3–6].

4.2.2 Technologies

In this subsection, we will provide an appropriate description of the primary technologies used to develop our work, aligning with the architecture described previously.

4.2.2.1 Blockchain Network / Hyperledger Fabric

For all the documentation of restoration processes and any other relevant information to be recorded in an immutable, tamper-proof, and credible way, it was clear from the beginning of our project that blockchain technology and its unique features would properly suit our requirements. The blockchain network will be used to maintain data about fundamental and technical documentation of the classic (e.g., VIN, engine number, mechanical and electrical schemas), every detail from restoration procedures (e.g., materials and tools used), independent assessment certificates (e.g. quality and authenticity evaluations, awards in Concours d'Elegance), list of previous owners of the vehicle, and any other important documentation (e.g., inspections report and insurance records). The cryptographic hash of associated media evidence will also be recorded on the blockchain.

The chosen technology to implement the blockchain network in our system is Hyperledger Fabric. As previously described, Hyperledger Fabric is a consortium-oriented blockchain platform, that presents a highly modular and configurable architecture. Smart contracts, or *chaincode* in Fabric's nomenclature, will be used to handle the logic to record the previously mentioned data. Fabric allows for the creation of organizations comprising different users for entities connected to the network. It also supports the creation of channels and enables private data collections and attribute-based access control for selected network members to communicate with data isolation and confidentiality. Other benefits obtained by choosing Fabric over other permissioned blockchain platforms are the fact that there are no transaction costs associated, and smart contracts can be written in general-purpose programming languages. In our case, the programming language chosen for the smart contracts was Go due to its optimization and widespread adoption within the Fabric community developers.

The superior performance of Hyperledger Fabric was a crucial factor in our decision to choose it over other blockchain platforms. This decision was informed by the performance assessments outlined in [47] and [48]. In both these studies, the performance metrics of various blockchain platforms, such as Corda, Quorum, and Ethereum, were evaluated and compared. Both papers acknowledge that Hyperledger Fabric outperforms the others, exhibiting lower latency, higher throughput, and greater scalability. Moreover, Fabric's privacy mechanisms have been significantly improved to suit enterprises with varying permission levels. Fabric also presents a larger and more robust community, along with more informative documentation, and its widespread adoption across several blockchain projects, are notable factors that influenced our decision to select this platform over others.

4.2.2.2 Docker

Docker is a powerful containerization platform that streamlines the workflow of developers by using virtualization and encapsulating software and their dependencies into containers that can run consistently across different computing environments.

Since Docker was used in the previous work developed in this project [5], it was logical to keep this technology for containerizing and deploying our system's components, mainly the *ClassicsChain*'s application with API integration and GUI. This ensures a standardized deployment of all our project components, leading to a simplification and acceleration of both current and future development workflows.

Furthermore, Docker was used to containerize the developed *chaincode* to install it on the network's peers. Additionally, Docker was also employed to provide and execute the Docker Images that compose our Hyperledger Fabric blockchain network.

4.2.2.3 Go

Go or Golang, is an open-source statically typed programming language developed by Google. It excels due to its simplicity, efficiency, security, and concurrent nature.

As stated in subsubsection 4.2.2.1, Go was the programming language selected to develop the smart contracts of our project. While Hyperledger Fabric supports smart contracts (or *chaincode*) to be written in several general-purpose languages, Go stands out as the most optimized one and predominantly adopted by the community and blockchain developers [49]. One of the features that makes Go the optimal choice for *chaincode* purposes is the remarkable performance and scalability in concurrent operations.

4.2.2.4 Node.js

Node.js is a powerful, cross-platform open-source Javascript runtime environment. Node.js leverages an asynchronous event-driven model, making it ideal for developers to build scalable, high-performance real-time applications and microservices.

To interact with the developed *chaincode*, a Fabric application was implemented that directly invokes the operations available through the smart contracts deployed on the blockchain network. This component was developed using Node.js, primarily since we needed to use the Software Development Kit (SDK) offered by Hyperledger Fabric. While this SDK is accessible in other programming languages such as Go and Java, the Node.js version is the most refined. This application constitutes the integration bridge for our system, connecting other important components and respective functionalities, such as the user database, email service, and off-chain media storage, and acts as our REST API, exposing the developed HTTPS endpoints.

4.2.2.5 MongoDB

MongoDB is a document database known for its flexibility and scalability. It stores data in flexible JSON-like documents, allowing for fields and data structure to vary between documents, making it suitable for a wide range of applications.

To facilitate user registration and, consequently, authentication with typical credentials, besides the registration within the blockchain platform that records the user as an entity of the network attributing it a public and private key, the user is also automatically

registered in a MongoDB database. This ensures that the user can be authenticated in the system with credentials, such as email and password. Still, when interacting with the blockchain network, it uses the expected certificates. This approach further helps to decouple the users' credentials from the actual assets stored on the blockchain, dedicating the blockchain storage exclusively to information regarding classic cars.

4.2.2.6 IPFS

As our system is based on blockchain and one of its unique characteristics consists in its decentralization, we also found it important to store media files associated with classic cars, such as images, videos and other files, decentralized. However, blockchain is not suitable for high-volume data storage. To address this issue, InterPlanetary File System (IPFS) is used as off-chain media storage. IPFS is a decentralized, Peer-to-Peer (P2P) file storage and sharing protocol that provides redundant, efficient, and permanent storage for different types of files, employing a content-addressed data approach.

The IPFS network, being a decentralized network, eliminates the need for a central authority, retrieving files from multiple sources at once. All the data in the network undergoes verification through hash functions, ensuring its reliability and integrity.

Within our system, while these media files are not recorded directly on our blockchain, their respective Content Identifier (CID) is. Each file is identified by a singular CID, generated through a content-based address that functions identically to a cryptographic hash. This guarantees that identical files will produce matching CIDs, confirming their uniqueness and integrity. The CID, typically a fixed 32-byte length irrespective of the original file's size, is well-suited for recording on the blockchain. This characteristic ensures its efficient representation on the blockchain network, providing a concise and immutable reference to the associated content stored on the IPFS network.

In subsubsection 5.1.3.4, we will properly explain the decision details regarding the implementation of IPFS in our solution.

4.2.2.7 Swagger

Swagger is an open-source specification framework for designing, documenting and testing REST APIs. By using this tool on a API, a *YAML* document is generated, containing the list of available resources and endpoints. The Swagger UI provides an intuitive and user-friendly interface that allows for the visualization, interaction, and testing of the API's endpoints directly from any web browser. Throughout the development and testing stages of the *ClassicsChain* API, this interface proved crucial in streamlining the testing and debugging processes. Moreover, it serves as complete documentation for entities interested in potentially integrating it with their systems, platforms, and APIs.

4.2.2.8 Angular

Angular is a widely adopted Typescript-based open-source framework tailored for the

development of dynamic and responsive GUIs. This framework's popularity can be attributed to its comprehensive set of tools, cross-platform support and robust features that facilitate the creation of responsive, efficient, and maintainable applications.

In our solution, a GUI was developed to allow users to visually interact with different functionalities based on their roles. We opted for Angular as the framework to build this interface. The choice to utilize Angular for the *ClassicsChain* Web Platform was influenced by the fact that the existing *Charter of Turin Platform* was already built with Angular. Given our efforts to understand its development features and make modifications to it, it made practical sense to continue with Angular for the *ClassicsChain* GUI.

The vehicles' information is displayed as a registry with all the relevant data that can function as an Identification Card of the vehicle, which we called *Vehicle Card*. A more complete explanation of the *ClassicsChain* GUI is provided in subsection 5.1.4.

4.2.2.9 Nginx

To make our Angular web application accessible beyond the local network, we used Nginx, an open-source web server that delivers static content and websites fast, in a robust and configurable nature. Moreover, it also functions as a reverse proxy, load balancer, and HTTP cache, which alleviates concerns regarding incoming traffic, including access volume and concurrent connections to the website, allowing developers to focus on implementing other functionalities for their projects. This enhances our Web Application's performance and scalability, contributing to a smoother user experience. Our Angular web application was deployed using the *Nginx* web server, specifically configured to enforce HTTPS protocols, thereby fortifying the overall security of this component of our solution.

4.2.2.10 OpenStack Server

In order to host our system's back-end and front-end for our Web application as well as to make the REST API accessible to every interested entity, we utilize a server hosted on a remote machine provided by the Infraestrutura Nacional de Computação Distribuída (INCD) and accessible via *OpenStack*. To establish a connection to the server from our machines, we used the Secure Shell (SSH) protocol.

This server had already been set up during the previous development of the *Charter of Turin Monitor* with Ubuntu 20.04.3, 4 Virtual CPUs, 8GB of RAM, and 100GB of Disk memory. These specifications proved adequate for our work development, without any compromise on performance. For our work, it was necessary to expose additional ports to ensure remote accessibility for the newly created components.

4.2.2.11 Kubernetes

Kubernetes is an open-source platform that serves to automate the deployment, scaling, and management of containerized applications. It is a technology with a remarkable

capacity to orchestrate containerized workloads efficiently, providing features like automated load balancing, self-healing, and seamless scaling to ensure optimal performance and availability.

In our project, simply developing a blockchain network in a local environment would not suffice to meet our initial objectives and desired contributions. For this reason, Kubernetes was leveraged to deploy our blockchain network onto the cloud, elevating it to a production network implemented and managed following well-researched best practices, ensuring it is accessible and usable by all interested parties.

4.2.3 Deployment Diagram

With the components and technologies used described and combining our system's architecture in the component diagram with these technologies, we can present the deployment diagram of the developed system as shown in Figure 4.3.

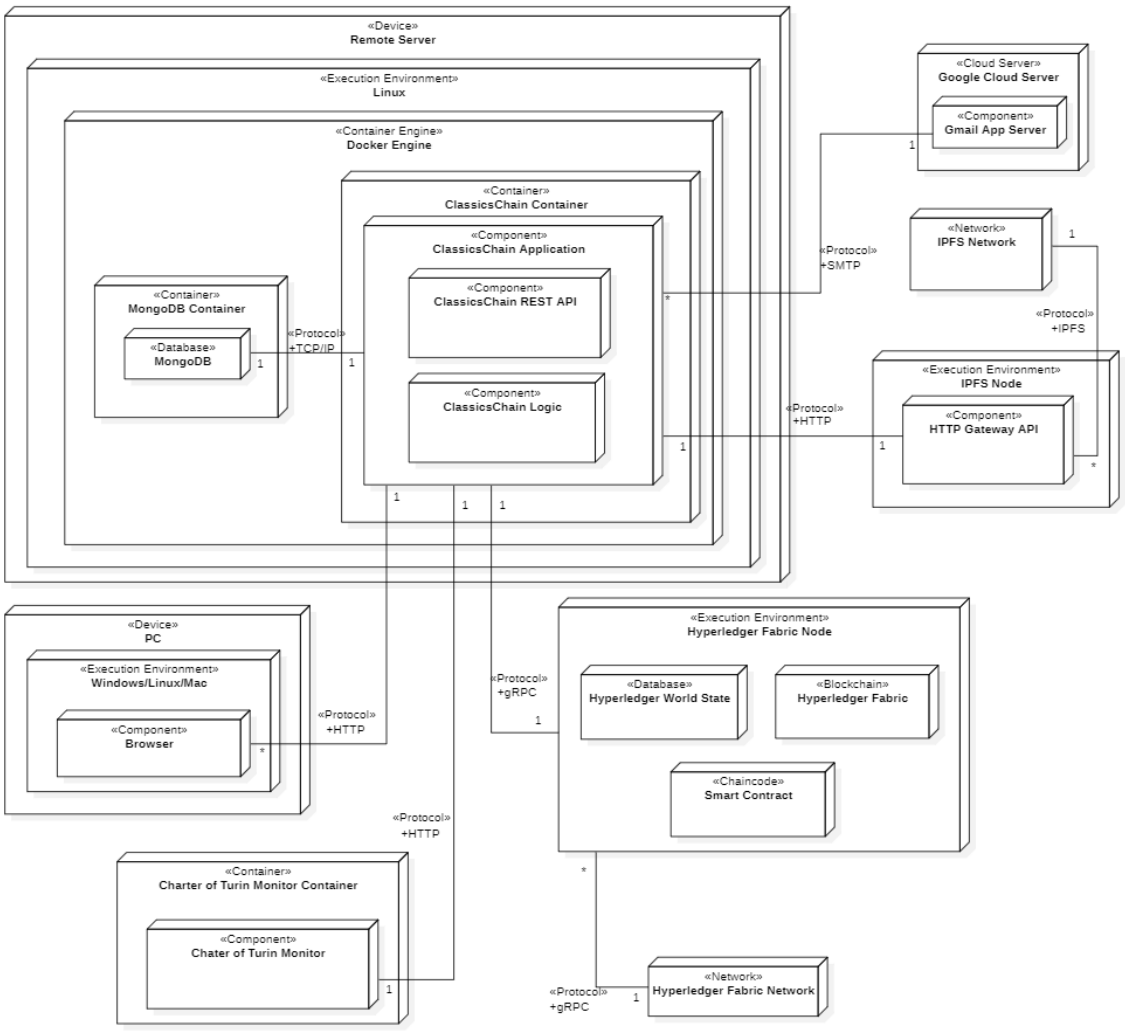


Figure 4.3: Deployment Diagram of our solution.

5

Implementation

This chapter is dedicated to presenting and explaining the implementation details of our work. In section 5.1, we describe the implementation decisions and developed features of the components of our main contribution, which is ClassicsChain. Subsequently, in section 5.2, we described the additional features implemented in the Charter of Turin Monitor, along with its integration with ClassicsChain.

5.1 ClassicsChain

As previously mentioned, *ClassicsChain* is the backbone of our project. *ClassicsChain* is a blockchain-based application established on the principles of decentralized web applications, that leverages the unique features of blockchain technology to facilitate the adequate recording of every detail and information of classic automobiles, restoration procedures applied, and all associated documentation by ensuring this information is trustworthy, immutable and traceable. Moreover, our platform provides data sharing and facilitates collaboration between interested entities.

Following, we will describe the implementation and development procedures of each component within our system, while also providing an understanding of the the implementation decisions made.

The complete code repository for *ClassicsChain* is available at this link.

5.1.1 Blockchain Network

In order to develop *ClassicsChain*, we had to begin with the foundational element of our work: a blockchain network, particularly a Hyperledger Fabric Network. As described in subsection 2.3.3, a Fabric blockchain network is a technical infrastructure that provides ledger services for distributed applications [25].

In the initial development phases, we utilized the Fabric Test Network, a simple network provided by the Hyperledger Fabric developers to facilitate user learning about Fabric's network functionalities, setup fundamentals, test smart contracts and applications. That

was the exact purpose for which we used it. The Fabric's developers make this test network available with Docker images and configurable through shell scripts. These scripts are used to:

- Configure some of the network's configurations and policies;
- Instantiate the network, with peer and orderer organizations;
- Create channels;
- Join peers to the channels;
- Install the *chaincode*, which includes, package, install on the peers, approve, and commit it to the channels;
- Interact with and invoke the smart contract's functions;

Deployed using Docker Compose, the test network operates nodes on a local machine, ensuring their isolation within the Docker Compose network environment. This network is composed of two peer organizations, running one peer each to store the blockchain ledger and validate transactions, and one orderer organization, using a single node ordering service, that uses the Raft consensus algorithm to order the transactions across the network [25].

Although the Test Network was vital to our learning processes, understanding how a Fabric network works, and deploying the first versions of our smart contracts as well as testing them, it was evident from the beginning that this oversimplified network would not fit our needs and that it would not align with the classic cars ecosystem and its specific requirements, although, at that time, the final network topology and design were still undecided. This decision was one of the most challenging and demanding decisions that we faced throughout this work, so we actively sought opinions from industry experts, as well as past project participants, and other blockchain developers (unrelated to this project).

Because of the aforementioned reasons, upgrading our network components and topology to better align with the classic sector requirements and moving our infrastructure to a production network deployed on the cloud was the next logical step, However, it was first necessary to decide definitively on how we would organize the network topology and design, as well as other crucial network configurations.

5.1.1.1 Network Design

After a thoughtful reflection and thorough analysis of the blockchain network topology and design to properly suit this unique sector, we determined that an optimal configuration would entail four organizations, structured as follows:

- **Peer organizations**: We decided to employ three peer organizations, with each one representing the groups of our main stakeholders. Each organization runs two peers. The first organization is the **Owners Org**, representing the classic car owners and potential buyers. Then, there is also the **Workshop Org**, representing the certified restoration workshops that utilize our system. The third organization is the **Certification Org** representing the certification bodies' authorities. For a clear understanding, it is important to emphasize that specific entities, being specific classic car owners, specific restoration shops, or specific certification bodies, are registered as unique users/entities of the specific organization with unique private and public keys. This implies that, for instance, the Raimundo Branco restoration shop is a registered entity under the **Workshop Org**.

- **Ordering organization**: We decided to employ one ordering organization. This organization provides the Ordering service running three orderer nodes, also known as the consenter set. We opted for the RAFT consensus algorithm, due to its straightforward setup and its superior performance. In our conception, we considered that the ordering service must be maintained by a trusted authority, that is unbiased and has no prevailing interest in disrupting the consensus protocol, which we identified as Fédération Internationale des Véhicules Anciens (FIVA). As explained in subsection 2.2.3, at the time of this project's development, the most secure consensus protocol available natively with Fabric is the RAFT protocol since it uses a Crash Fault Tolerance (CFT) protocol, meaning that it is resilient to node loss or failure but does not address malicious nodes. Hence, it is crucial that these nodes are administered by a trusted organization in this domain. Using the same rationale, if these nodes are operated by a trusted authority and a Practical Byzantine Fault Tolerance (PBFT) algorithm is employed, would be unnecessary and would lead to performance and throughput degradation [25].

- **Four Certificate Authorities**: As proposed by Hyperledger Fabric developers, in a production network it is important to employ one Certificate Authority (CA) for each one of the organizations. In Hyperledger Fabric, CAs need to be the first component to be deployed on a Fabric network because the certificates associated with a node or user must be created before the actual creation of the node or user. A CA generates the certificates used to authenticate and authorize the organization's admins, nodes owned by that organization, and other participants in the network.

- **State Database**: As stated in subsection 2.3.3, in Hyperledger Fabric, current state values are stored in a state database for efficient reads and queries to the *chaincode*. There are two options for this database: *LevelDB* and *CouchDB*. We opted for the default database, *LevelDB*. While *CouchDB* supports JSON rich queries, deeming such queries not strictly necessary for our system. The database type has a significant impact on system performance. In comparison to *CouchDB*, *LevelDB* exhibits up to

three times higher throughput [50]. Moreover, *CouchDB* performance also degrades more than *LevelDB* does as the volume of state data increases [49].

There were also some additional considerations, especially related to performance, that had to be taken prior to our final network implementation. The reasons for the majority of these decisions were derived from the performance analysis of T. Guggenberger et al. [50] and from the official Hyperledger Fabric documentation [25, 49] and are explained as follows:

- **Single channel**: Given that every participant organization is invoking, storing, and sharing data about classic vehicles, and none of the organizations have exclusive interests to share with one another through an exclusive channel, we determined that maintaining this ledger data within a single channel would be the most fitting approach for the classic cars use case. This simplifies the network architecture, enhancing manageability and maintenance, while also ensuring that all transactions on the channel are private and visible only to the participants on that channel.

- **Attribute-Based Access Control**: In order to maintain privacy using a single channel architecture, enabling selected network members to communicate while ensuring data isolation and confidentiality, we implemented Attribute-Based Access Control. Given our use of a single-channel architecture and the fixed nature of the organizations, employing additional channels or using Private Data Collections to guarantee data privacy would not be suitable since both of these privacy mechanisms operate at the organization level rather than at the entity level. Thus, these mechanisms can only keep data isolated and confidential between organizations, not between individual users or entities. Using Private Data Collections also degrades performance, with throughput for public transactions being around two times higher than for private transactions (when using *LevelDB*). Consequently, Attribute-Based Access Control was selected, restricting data access to certain roles or users within each organization. This is achieved by embedding the necessary attributes in each user's certificate and integrating access control into the *chaincode* logic.

- **Number of peers**: As mentioned in subsection 2.3.3, peer nodes form the building blocks of a Fabric blockchain network. Every network's peer maintains a replica of the ledger, executes *chaincode*, verifies transactions and interacts with client applications. In our setup, each peer organization is comprised of two peers, resulting in a total of six peer nodes. We also decided to provide each peer as an anchor peer for client applications to connect to and interact with. This approach ensures we can achieve redundancy, and load balancing of client requests across each organization. Consequently, we establish a level of fault tolerance as the network remains unaffected as long as at least one peer from each organization remains operational.

- **Number of ordereres**: As previously stated in this section, we used a three-orderer node architecture. These orderers use the RAFT consensus protocol, a CFT protocol that withstands node failures. With three orderer nodes deployed, this architecture can tolerate the failure of one node. It is worth noting that due to the RAFT protocol using a "leader and follower" model, an increase in the number of orderer nodes would also increase the communication overhead, as the leader must communicate with all of the nodes for the ordering service to function effectively. This way, we considered that a three-node ordering service aligns well with the requirements of our project, however, a five or even seven orderer architecture could also be considered without compromising performance. Regarding node failures, if a non-leading orderer crashes the impact on performance is minimal, however, if the crashed node is a Raft leader, it takes about 5 seconds for the system to resume normal operations.

- **Endorsement Policy**: When specifying a *chaincode* definition, an Endorsement Policy is also defined. This policy describes which organizations must endorse transactions for them to be committed to the ledger. We chose to implement a Majority policy, which mandates that a transaction must receive endorsements from at least two out of the three peer organizations. We initially tried to use a policy where every organization had to endorse transactions, however, a decrease in the network's throughput was noticed.

Our final Hyperledger Fabric Network architecture proposal is illustrated in Figure 5.1. It is important to note that, with this network structure and distinctive configurations, our blockchain network aligns with the characteristics of consortium blockchain since the network is comprised of different organizations, that collaborate to maintain it. However, it also inherits some aspects of a public permissioned blockchain, as the participants from these organizations are regular public users, such as classic car owners, potential buyers, or administrators of restoration shops, but require permission for specific information access or operations. They do not actively contribute to the network, aside from recording and retrieving data if they have been granted permission to do so, which is managed on a role and attribute-based access control.

5.1.1.2 Network Deployment on Kubernetes

As previously outlined, to achieve the pre-established objectives for this project, it was vital to deploy a production blockchain network, a network that had to be deployed on the cloud and not in a local environment. As suggested in the official Hyperledger Fabric documentation [25], we decided to deploy our Fabric network using Kubernetes.

A precondition to deploy a blockchain network to Kubernetes is to have an available Kubernetes Cluster. A Kubernetes cluster is a set of nodes that run containerized applications. Containers are put into *pods* that execute inside these nodes. Due to time constraints

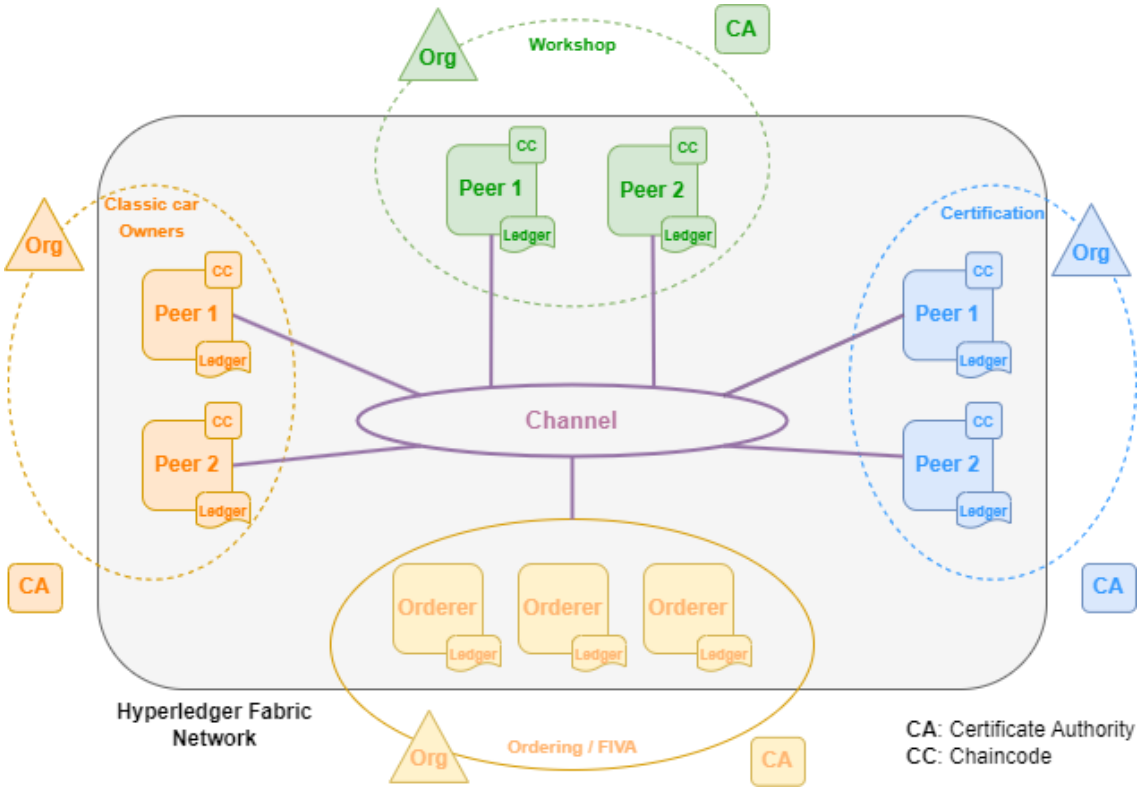


Figure 5.1: Hyperledger Fabric Network Design Diagram.

and a lack of experience in setting up a Kubernetes Cluster from scratch, as well as limited resources for a cluster with multiple nodes, we opted to host our cluster in Digital Ocean Web Services, which allowed for a simpler setup and reduced management efforts. However, we acknowledge that managing a cluster with on-premises nodes entirely under our control would offer superior data security, privacy, and a higher level of overall control over the network infrastructure since we would not have to dependend on a third-party service provider and its security measures and policies.

For our Fabric network deployment, our Kubernetes Cluster was structured to align with the four distinct organizations within our network, so it is composed of four different nodes. Each node in the cluster corresponds to one of these organizations, responsible for running each specific Fabric Docker container in a designated *pod*, such as each peer, each orderer, and each CA. Collectively, the cluster nodes have a computational capacity of 8 virtual CPUs, 16GB of RAM, and 240GB of Disk memory.

With our Kubernetes Cluster running successfully, we achieved the capability to deploy our blockchain network. In the process of selecting a suitable technology for deploying Hyperledger Fabric networks on Kubernetes, we thoroughly explored various options. Ultimately, we opted for Hyperledger Fabric Operator, since it was notable for its recent updates and substantial contributions from the development community. This choice was further informed by its simple and straightforward installation, which does not demand advanced knowledge of Kubernetes, along with its properly detailed documentation.

These factors collectively played a pivotal role in accelerating the deployment of our blockchain network within a reasonable time period.

The complete process of transitioning from an empty Kubernetes Cluster, with only the nodes deployed, to a fully operational production-grade Hyperledger Fabric network is an intricate process, even when leveraging technologies such as Hyperledger Fabric Operator. Next, we will concisely explain the steps we performed to correctly deploy our network. These steps are executed through command-line instructions using the Kubectl command tool incorporated with the Hyperledger Fabric Operator plugin. On a related note, it is important to highlight that we employed Hyperledger Fabric version 2.5.3 for the peer and orderer images, and version 1.5.6 for the deployment of CAs.

1. Deploy the CAs for each peer organization;
2. Register a user in the CA for the peers of each organization;
3. Deploy two peer nodes for each peer organization;
4. Deploy the CA for the orderer organization;
5. Register a user in the orderer organization CA;
6. Deploy the three orderer nodes;
7. Register and enroll an admin identity for each organization;
8. Create the channel. Besides the performance considerations described previously, it is also worth highlighting some of the channel configuration settings derived directly from the recommendations provided by the developers of Hyperledger Fabric Operator. These settings are related to the ordering service executing in the channel and include: a batch timeout set at 2 seconds; a maximum transaction limit per block, established at 120; and both absolute and preferred maximum block size values, set at 1MB and 512KB respectively;
9. As an admin of each organization, join the respective peers to the channel;
10. Build the developed *chaincode* Docker image (this step could be undertaken earlier, but it is presented in this sequence for clarity of understanding);
11. Deploy and install the *chaincode* as an External Service across all peers. The installation and deploying of *chaincode* as an External Service is a suggestion from the developers of Hyperledger Fabric Operator since it facilitates the deployment of *chaincode* on cloud deployments, such as Kubernetes. This entails initiating a *pod* containing the *chaincode* Docker image, which then makes its installation available to all peers. This is important to keep the *chaincode* manageable uniformly, eliminating the need for manual installation on each peer.

12. Approve the *chaincode* definition for each organization;

13. Commit the *chaincode* definition, with at least one organization, in order for the *chaincode* definition to be implemented on the channel.

With all the aforementioned steps successfully completed, our production Hyperledger Fabric network has been deployed on Kubernetes, permitting to invoke transactions of the smart contracts and securely record them onto the ledger.

5.1.2 Smart Contracts

To execute logic on our blockchain network, smart contracts are needed to invoke operations that generate new transactions to be added to the ledger. In Fabric, packaged smart contracts that run on the network are referred to as *chaincode*, while the term smart contracts encompasses the functions and transaction logic. It is imperative to provide a detailed explanation of how we implemented such an important pillar of our work, elucidation specific information on how smart contracts govern the logic of *ClassicsChain*.

As previously described, we decided to choose Go as the programming language for our smart contracts. This decision was guided by its superior efficiency and optimization, making it the most adopted language on Hyperledger Fabric. Additionally, Go's simplicity and comprehensive documentation further facilitated our choice. The official Hyperledger Fabric contract interface for Go, *Fabric Contract API*, is used to implement our smart contracts, while also leveraging the *Shim package* in order to access the smart contract state variables and transactions context.

Initially, our *chaincode* was comprised of three separate smart contracts, each handling essential aspects of our system: one for classic car functions, another for restoration procedures, and a third for permission access. It is important to note that this separation did not imply isolation of their functionalities, as some of the functions in different smart contracts interact and rely on one another. However, since we installed our *chaincode* as an External Service to deploy our Hyperledger Fabric network, we encountered challenges in maintaining the smart contracts separated, so the smart contracts were unified into a single smart contract. Nevertheless, the logical structure of the functions remains organized in the same manner as described, providing an abstract distinction between the various logic components of our *chaincode*.

5.1.2.1 Assets Definition

To represent the distinct objects on the ledger, assets are defined in the form of Go structs. To encapsulate the mentioned fundamental elements of our system, we specified the following structs, each with its corresponding parameters, as presented in Listing 5.1.

```
type Classic struct {
    Make                string `json:"make"`
    Model               string `json:"model"`
```

```
    Year                int     `json:"year"`
    LicencePlate        string  `json:"licencePlate"`
    Country             string  `json:"country"`
    ChassisNo           string  `json:"chassisNo"`
    EngineNo            string  `json:"engineNo"`
    OwnerEmail          string  `json:"ownerEmail"`
    PriorOwners         []string `json:"priorOwners"`
    Restorations        []RestorationStep `json:"restorations"`
    Certifications      []string `json:"certifications"`
    Documents           map[string]string `json:"documents"`
}

type RestorationStep struct {
    ID                  string   `json:"id"`
    Title               string   `json:"title"`
    Description         string   `json:"description"`
    PhotosIds           []string `json:"photosIds"`
    MadeBy              string   `json:"madeBy"`
    When                string   `json:"when"`
}

type Access struct {
    OwnerEmail          string `json:"ownerEmail"`
    Viewers             map[string]string `json:"viewers"`
    Modifiers           map[string]string `json:"modifiers"`
    Certifiers          map[string]string `json:"certifiers"`
}
```

Listing 5.1: The defined structs of our implementation.

The *Classic* struct represents the classic cars to be registered to the ledger, and its parameters are tailored to capture the fundamental details of a classic car, its related documentation and the restoration procedures applied. The selection of these parameters is guided by their inclusion in the *Charter of Turin Monitor* and their alignment with recommendations from FIVA. Each classic car is uniquely identified by its chassis number or VIN, and all operations pertaining to vehicles rely on this attribute, acting as the primary key of this object. This decision was made because the VIN is inherently unique to each vehicle, providing a singular and distinguishable identifier for every classic car in our context. What we named the *Vehicle Card* essentially encapsulates all the relevant information of the *Classic* object.

The *RestorationStep* struct represents each restoration procedure and its most relevant details, including its title, description, associated photos (or any other media evidence), who performed it, and the date of execution. The primary key is derived by the concatenation of both the chassis number and the current total count of restoration procedures, regarding that particular vehicle.

The *Access* struct represents the current access permissions for each classic vehicle. This information is crucial in our implementation since our access control mechanism

is entirely executed and handled by smart contracts, ensuring that specific entities have restricted access to particular vehicles' data. This way, this struct maintains information about the owner, users with viewer, modifier, and certifier access for each classic vehicle, along with the timestamp indicating when these permissions were granted. The primary key for this struct is likewise based on the chassis number of each vehicle, however, it is also concatenated with the word "Access" to differentiate it from the *Classic* struct.

5.1.2.2 Smart Contract Functions

Due to the extensive amount of code and functionalities within our smart contracts, we will next focus on describing the most significant functions, the functions that most contribute to the unique features of *ClassicsChain*, while also providing insights into specific implementation details.

The following functions are the most relevant, warranting a more thorough examination and explanation for a proper comprehension of their implementation.

- **CreateClassic**: This function requires validation to ensure its successful execution and subsequent addition to the ledger. First and foremost, it verifies whether the entity submitting the function holds a role within the **Workshop Org** or **Certification Org**, as detailed in the functional requirements section. It is also checked if the classic vehicle is not already registered in the system. Subsequently, if all the necessary fields of the *Classic* struct are correctly provided, the classic is created with the given parameters, the owner is granted complete ownership over its classic's information and the submitting entity is automatically added to the access permissions for that vehicle, granting the modifier or certifier access level according to their affiliation with the **Workshop Org** or **Certification Org**, respectively.

- **UpdateClassic**: This function enables the update of the fundamental details of a classic car. For this function to be valid, it is first confirmed that the car exists and that the submitting user is the owner of the car, as only the owner of the car is authorized to execute this operation. If validated, the fundamental details are updated to the ones provided.

- **UpdateClassicEmail**: This is the function that transfers the total ownership of the classic vehicle to another user. For this function to be valid, it is first confirmed that the car exists and that the submitting user is the owner of the car, as only the owner of the car is authorized to execute this operation. If confirmed, the ownership of the classic vehicle is updated to the new proprietor, and the previous owner is recorded as a prior owner of the vehicle.

- **CreateRestorationStep**: This function facilitates the creation of a new restoration procedure for a particular vehicle. To ensure the validity of this function, it is first verified if the particular car exists and if the entity submitting this transaction

holds modifier access permissions. These permissions must have been granted previously to a workshop entity or, alternatively, can be executed by the owner in cases involving minor restoration procedures, such as a change of tires or oil change. The essential parameters, including the title and description of the restoration step, are mandatory, while others, such as the IPFS CID of associated photos (or other media files), are optional. The date and time of the restoration procedure and the entity that performed it are automatically added based on the current time and the submitting entity, respectively.

- **UpdateStepAndPhotos**: This function is responsible for the process of updating the details and associated photos (or other media files) of a specific restoration procedure, and also presents some important validation steps. It is first checked if the specific restoration procedure exists. Additionally, it is confirmed if the submitting entity holds modifier access permissions and whether it is the entity that originally performed that specific restoration. This dual authorization check ensures that the information of a restoration step can only be updated by who originally made it if its permissions were not revoked in the meantime. Adding media files continues to be optional, as it is possible to update a restoration without providing new photos as evidence. Upon satisfying these conditions, the restoration procedure is updated successfully. It is worth noting that two additional variations of this function were implemented: **UpdateStep** and **UpdateStepPhotos**. These allow for the updating of only the textual details (title and description) or solely the media, respectively.

- **AddDocument**: This is a simple function that adds a new document to a particular vehicle. However, it is crucial to outline the conditions for its successful execution. It is first verified if the particular classic vehicle exists, and then if the entity submitting the transaction holds at least one of the following permissions: owner, modifier, or certifier access permissions. The document is recorded as an IPFS CID and added to that particular classic vehicle.

- **MarkAsCertified**: This function enables a certification body to certify or authenticate a particular classic vehicle. To execute this function, the preconditions involve confirming the existence of the car and verifying if the entity submitting the transaction holds certifier access permissions over the particular vehicle. Upon successful validation, the specified classic car is officially marked as certified by the entity submitting the transaction, serving as conclusive proof of certification for that vehicle.

- **ReadClassicAsViewer**: This is a simple read function that returns the classic vehicle associated with the provided chassis number passed as a parameter. However, this function also needs to satisfy some conditions to execute successfully, namely that the car is registered in the system and the submitting entity holds viewer access permissions. This access level is considered less restrictive, as any other

permission level, including ownership, modifier, and certifier access, encompasses it. This means that all entities that hold at least the viewer access level, regardless of their access level, can access this same information. In essence, this function returns the complete *Classic* object, which represents a *Vehicle Card* within the broader context of our system. Despite being a simple function, it is one of the most vital functions for our system, for the context, pre-established goals, and requirements of this project. This function has three alternative variations based on the executing user's access level: **ReadClassic**, which is performed by the vehicle's owner; **ReadClassicAsModifier**, which is performed by entities with modifier access level; and **ReadClassicAsDocumenter**, which is carried out by the entities with certifier access level.

- **GetStep**: This function is a simple read function that returns the title, description, registration date, performer, and IPFS CIDs of the media files associated with a specific restoration step identified by the restoration step identifier passed as argument. To execute this function, a prerequisite check is performed to confirm the existence of the car and validate that the submitting user is indeed the owner. Upon successful validation, the fundamental details are then updated to the values provided.

- **GetClassicHistory**: This is also one of the vital functions of the smart contracts and the overall system. It provides the complete chronological history of all updates and transactions that led to the current state of a classic vehicle, tracing back to its initial registration in the system. As in the aforementioned functions, it requires the condition that the car must be registered in the system and the submitting entity must have at least a viewer access level. Upon success, the function retrieves all versions of a specific *Classic* object, along with the corresponding IDs and timestamps of the transactions that generated those versions. Essentially, within the broader context of our system, it returns a timeline of the different versions of the *Vehicle Card*.

- **QueryClassics**: While technically comprised of three distinct functions, for the sake of simplicity, they are presented as a unified function. Collectively, these functions serve the purpose of, respectively: listing all the classics owned by a specific user (**QueryClassicsByOwner**); listing the classics for which a restoration workshop holds modifier access (**QueryClassicsByModifier**); listing the classics to which a certification body has certifier access (**QueryClassicsByCertifier**).

- **GetAccess**: This is a straightforward read function that retrieves the current access permissions for a particular vehicle. It requires the condition that the vehicle must be registered, and the only user authorized to perform this operation is the classic car owner. No other entity, regardless of the level of access permission they hold under a car, is able to access this data. Upon success, all current access permissions

are returned, identifying the user to whom the permission was granted, the access level, and the date and time of authorization.

- **GiveAccess**: This function holds significant importance as it governs the access permissions of each user for a specific vehicle. This function validation steps include checking if the classic car is registered, and if the submitting user is the owner of the vehicle since only the car's owner is authorized to execute this function. Upon successful validation, the permission is granted to another user, while also setting the access level (viewer, certifier, or modifier), both provided in the arguments of the function and automatically recording the date and time of the granting process. It is important to emphasize that when a user is granted access to the details of a classic car, regardless of their level of access, all information (excluding permission details, which are exclusive to the owner) is shared in a transparent and complete manner. This ensures that the owner of the classic cannot selectively share inaccurate or outdated information, nor can he choose to share only some specific details that may serve his interests.

- **RevokeAcess**: This function mirrors the implementation of the **GiveAccess** function, with the same conditions needed to be established. However, instead of granting access permission to another user, it revokes previously granted access from a specified user.

- **CheckUserAccess**: This function is used to determine the access permission level held by a specific user for a particular classic car. The only condition to be verified is that the car must be registered. If it executes successfully, a concise string containing "owner", "modifier", "certifier" or "viewer" is returned, according to the current level of access permissions a specified user holds over a specified vehicle. If not successful, an error is returned informing that the user has no permissions over the vehicle. It is important to note that modifier and certifier access permissions persist until manually revoked by the owner, however, viewer permissions are automatically revoked after a one-day period from the time of granting.

- **GetAccessHistory**: Similarly to the **GetClassicHistory** functions, this function also provides a complete chronological history, but in this case, of all updates and transactions that led to the current state of the access permissions for a classic vehicle. This encompasses details of the prior owners, authorized users, their respective access levels, and the date and time of the granted permissions. This chronological record traces back to the vehicle's initial registration in the system. To execute successfully, the car must be registered in the system, and the submitting entity must be the owner of the classic. Upon success, the function retrieves all versions of the *Access* object of a specific classic, along with the corresponding IDs and timestamps of the transactions that generated those versions.

It is important to note that the aforementioned functions, which include conditions regarding access permissions, are managed by an Attribute-Based Access Control scheme. With this approach, unique attributes identifying specific users or roles are embedded into each user's network certificate. Whenever functions require verification of these attributes, the relevant attributes are extracted directly from the certificates. It is worth emphasizing that the control of access permissions is completely achieved through this Attribute-Based Access Control approach on the smart contracts side. The underlying logic and corresponding error handling are implemented inside the smart contract's functions, and the essential attributes are integrated into each user's certificate.

5.1.3 Fabric Application and API

In order to interact with the *chaincode*, a Fabric application was developed to support the direct invocation of the available smart contract's functions. A Fabric application is also referred to as a client application because it interfaces with a blockchain network using the Hyperledger Fabric client SDK. Fabric applications are the only place outside the network where transaction proposals can be generated, invoking specific smart contract functions. To be able to make these functions available through endpoints, we implemented this Fabric application to serve as a REST API, that once hosted on our server, acts as our back-end server application that exposes HTTPS endpoints for each main function on the smart contract's logic. These endpoints can then be invoked from other clients and applications.

As mentioned in subsection 4.2.2, this component was developed using Node.js due to being the most refined option that can integrate with the SDK provided by Hyperledger Fabric. This SDK is composed of different modules, with the two primary ones being *fabric-network* and *fabric-ca-client*. The first is responsible for interacting with the Fabric network and deployed *chaincode*, handling operations such as transaction invocation and data querying. The latter is responsible for registering and enrolling new users to the blockchain, assigning a network identity, and providing the respective certificate.

To implement the application's API and corresponding endpoints and functionalities we used Express, which is a Node.js framework, known for its simplicity and flexibility in building REST APIs, streamlining the processes of defining routes, handling requests and corresponding responses, authentication, and error handling. The functionalities of this API are exclusively accessible through secure data transmission via HTTPS. This requirement ensures that all communication between clients and the API is encrypted, enhancing data security and privacy concerns.

5.1.3.1 Initial Configuration and Connection with the Fabric Network

The first stage in developing a Fabric application using the Hyperledger Fabric client SDK involves establishing and configuring a connection between the application and the Fabric Network layer and consequently, between the *chaincode* deployed on the network. A

prerequisite to enable this connection establishment is the need for a *YAML* file detailing the network's components, in order to create what is called a *connection profile*. This file has been previously generated upon the installation of the *chaincode* into the blockchain network.

A *connection profile* is simply described as a view of the network and it describes the set of the network's components, including the addresses and certificates of peer nodes, orderer nodes, and CAs associated with the respective organization on the Fabric network. This *connection profile* is only loaded once when the application starts its execution. The primary role of the *connection profile* lies in configuring the network gateway, which establishes the communication layer between the application and the blockchain network.

Furthermore, upon initiation of the Fabric application, the *connection profile* is also necessary to build the client for each organization's CA to enable user registration and enrollment through the application. For each organization's users to have their identities properly isolated on the respective organization, one *wallet* for each organization is also generated during the execution of the Fabric application. Each *wallet* is necessary to contain a set of user identities along with their associated cryptographic material, which includes the X.509 certificates, public and private keys, and identification of the respective organization.

Once the *connection profile* is loaded, enabling the initialization of each organization's CA client and each organization's dedicated identity *wallet*, and once an instance of the network gateway is established, this network gateway can be set up for a registered user, providing the associated *wallet* where its identity is defined and the loaded *connection profile*. With the gateway instance configured, a network instance is created, identifying the channel where the smart contract is deployed (in our case, there is only a single channel), and the name of the smart contract, to be able to interact with the Fabric network by submitting transactions and querying data.

It is important to note that a network instance is built for each user when the user first queries or submits a transaction to the network. This instance is then persisted for potential retrieval when the user decides to query or submit another transaction. This way, by having a network instance per user in the system, we ensure a multi-user approach, facilitating the handling of concurrent operations to the Fabric application.

5.1.3.2 User Authentication and Authorization

To interact with the Fabric blockchain, network identities for each user need to be defined. This definition involves a unified process of registering and enrolling users in the network.

1. The initial phase of this procedure involves the **registration** of a user as a new identity in the network. This registration is conducted by the CA administrator, and to perform it, it is first checked the precondition that the user is not already registered as a participant in the specific organization currently under registration. Then the *userId*, organization, and extra attributes are defined and associated with a

new identity with the role of *client*, which is stored by the respective organization's CA. To singularly identify and distinguish the users in our system, we decided that the *userId*, which is the unique identifier of an entity on the network, would be the email of the user. Additional relevant fields are included as extra attributes to the user's certificate. It is crucial to mention that the *userId* and extra attributes defined to each user's certificate are fields that are used to identify the user on the network and its corresponding permissions, when submitting a transaction or querying data, and are accessible within the function's logic of the smart contracts, governing authorization for specific functions and data access. In essence, these attributes are integral to our Attribute-based access control and data privacy approach.

2. The second phase of this process is the **enrollment** of the user. In this stage, the user's X.509 certificate is generated and associated with the newly created entity. The generated certificate is composed of the public and private keys of the user, the associated organization, and the extra attributes desired to be added. Finally, the identity is imported into the respective organization's *wallet* completing the coordinated process of registration and enrollment of a new user to the network.

Nevertheless, the complete process of registering a user in our system extends beyond the conventional network registration and enrollment. As mentioned in 4.2.2.5, to facilitate user registration and subsequent authentication within our system with standard credentials, the users are also automatically registered in a MongoDB database. This approach lets users authenticate in the system with normal credentials, such as email and password, and guarantees a decoupling of the user data and credentials from the actual assets stored on the blockchain. Nonetheless, when interacting with the blockchain via the network instance, the users utilized their network identity and corresponding certificate.

Through this dual-registration approach, we store in the MongoDB database the following for each user: email (enforced to match the *userId* for uniqueness), a hash of the password (hashed using the *bcrypt* algorithm with a 10-round salt [51]), organization, country, and either the first name and surname or, in the case of administrators of workshops or officers of certification bodies, the name of the company or entity.

To enhance security and data privacy, as previously outlined, each network's organization must have an entity *wallet*. Due to the adoption of this dual-registration approach, each organization must also maintain a MongoDB user database. This ensures that only the information of users affiliated with that specific organization is stored in the respective database. The connection to these databases is established only once during the initiation of the Fabric application.

The complete user registration process in our system is made available in our API via a POST request to the "*/api/Users*" endpoint. This involves passing all the essential parameters previously listed for storage in the user database within the request body. Subsequently, the detailed registration process is executed in its entirety.

Similarly, the process for a user to log in to the system is also accessible through a POST request to the "*/api/Users/Login*", however, this method involves some implementation details that are worth mentioning. The user's email, password, and organization are passed in the request body. Initially, a verification is made to check if the user is registered in both the organization's wallet and user database. If the user authentication is successful, a JSON Web Token (JWT), composed of the user's email and organization and valid for five hours, is generated and returned. This JWT, will serve for the identification of the user in all the requests to other endpoints of the API, using it in other requests' authentication header with the format "Bearer TokenString".

For all other available requests, the JWT identifying a particular user undergoes verification as a precondition to ensure validity and non-expiration. The user's email and organization are extracted and decoded from the JWT to be used on the application's logic. Subsequently, with the user's email and organization, the corresponding identity is retrieved from the respective organization's *wallet* to enable interaction with the blockchain network. As previously explained, each identity establishes a connection to the blockchain network via a dedicated network instance that is instantiated upon the user's initial query of data or submission of a transaction to the network and persisted for subsequent interactions. The extracted information from the token is also used by the GUI to know what elements to show to the given user and its organization, a topic discussed in subsection 5.1.4.

One thing that would significantly enhance our system, particularly in user authentication, would be the implementation of a mediated identity verification process. However, due to the constraints of our project deadline and its implicit exclusion from the defined scope, this enhancement is deferred to potential future work (refer to section 7.2).

5.1.3.3 Interaction with *Chaincode*

As previously explained, every function available on our API (excluding the user's registration and login functions) starts by verifying the JWT to get the user's email and organization. With this information, it is then retrieved or instantiated a network instance and smart contract instance for that specific user to interact with the Fabric network. Although not all the available functions interact with the Fabric network, namely the function that lists all the users registered in the system and the function to return the user's details given the user email, most of the functions have a similar implementation when it comes to interacting with the blockchain through the smart contract's functions.

With the network instance obtained for a particular submitting user, the first phase to interact with the Fabric network through the Fabric application is usually some type of data validation. This includes checking if the number of arguments passed in the request body is correct and if the mandatory arguments are set. Moreover, when the function being executed involves other users, such as granting/revoking permissions to another user or registering a classic as owned by a specific user, it is also confirmed that the particular user is registered in the system.

Following the successful completion of data validation, there can also be some other logic handling details. One that is common throughout some of the functions, is the need to obtain the current time, for instance, to register the time a user was given viewer permission to a classic car. At the beginning of this work's development, we used to source the current time directly on the smart contract's logic. However, when moving to a non-local Fabric's network we noticed that these functions were not being executed. This was due to the fact, that during the execution of the proposed transaction by the network's peers, their individual current time would slightly differentiate from one another, which would lead to disparate transaction responses from each peer, resulting in the invalidation of the transaction and its failure to get committed.

After the data validation and additional prerequisite logic, to actually invoke the smart contract's methods by proposing a transaction or querying the stored data, two different functions are used:

- **submitTransaction**: As implied by its name, is used when it is desired to record a new transaction and, consequently, new information on the ledger. This is done exactly as it was explained in the transaction flow of a Hyperledger fabric transaction described in subsection 2.3.3.

- **evaluateTransaction**: Conversely, this function is employed for read operations that exclusively query the ledger without recording any new or modified transactions. Unlike the *submitTransaction* function, this transaction is not transmitted to the ordering service, and no consensus algorithm is executed, leading to a quicker execution.

Finally, the result obtained from the transaction's execution is returned, whether it is a query or submit transaction. This result is obtained as a JSON-formatted string and is then converted into a Javascript object. Subsequently, this object is sent as a response to the original API request, along with the status code 200, denoting the successful completion of the request. In case of an unsuccessful request, an appropriate error status code, following Hypertext Transfer Protocol (HTTP) best practices, is sent along with an error description.

It is also important to mention that certain API operations invoke both the *evaluateTransaction* and *submitTransaction*. This applies to operations involving data recording in our media storage, such as registering a new restoration procedure with associated photos or a new legal document in the Vehicle Card. As previously explained in 4.2.2.6, the unique CID of each file is recorded on the blockchain, and to obtain this CID, the corresponding media file needs to be previously recorded onto the media storage or, at least, generated to be used in said media storage. If we only used the *submitTransaction* to invoke a particular function to record that to the blockchain with implemented access control logic on the corresponding *chaincode* function, the media files would always be registered, even in cases where the submitter does not have permission to do such operation.

To address this, in operations recording data on the media storage, we initially use the *evaluateTransaction* function to verify that the submitter user has enough permission to perform such operation on the particular vehicle. If verified, the necessary logic is applied to record the media data to our media storage, retrieve the generated CID of each media file, and subsequently use the *submitTransaction* method to register this information on the blockchain. If the submitter does not hold permission to perform that operation in that specific vehicle, an error is promptly returned after the execution of the *evaluateTransaction* function. This approach not only reduces the operation duration, given that media file recording on the media storage is time-consuming but more importantly, prevents the unauthorized recording of media data.

5.1.3.4 Media Storage Integration

As mentioned throughout this document, from the beginning of this project's research, we recognized that off-chain decentralized media storage would be a vital component of our implementation, due to the fact that blockchain technology is not suitable for high-volume data. This component ensures that the media files of various types and formats are adequately stored and an identifier of their content, based on a cryptographic hash, is subsequently recorded in our blockchain network. This approach is consistently applied to functionalities that involve interaction with media files, specifically in the registration of new restoration procedures, the updating of existing procedures, or the addition of additional documents.

With the study's initiation, it was also promptly acknowledged that IPFS would align with our requirements as the chosen off-chain media storage system. Hence, the integration of IPFS with our Fabric application became imperative. To enable this integration, we decided to use an IPFS Gateway, which enables an HTTP-based service to allow software to access content on the IPFS network. It serves as the bridge between HTTP and the decentralized storage protocol, IPFS.

In the initial stages of integrating IPFS to our project, we opted to use a Public Gateway, in this case, the official Public Gateway of IPFS, *ipfs.io*. This was implemented using the *ipfs-http-client* client library for our Node.js application. However, it was quickly evident that this approach would not be a decent option for our work. The uploading and retrieving of content proved considerably time-consuming and sometimes even unsuccessful. This was due to the public nature of this Gateway, as when it is overloaded with traffic, its functionality degrades significantly or ceases altogether. Furthermore, the finite storage on IPFS nodes resulted in data uploaded through the public Gateway being regularly subjected to garbage collection, an automatic resource management process.

To avoid the performance drawbacks of relying on a Public IPFS Gateway, support long-term media storage, and avoid incurring storage costs, we opted decided to use a combined solution of IPFS and *Filecoin*. Filecoin can be described as a decentralized storage network in which the storage providers rent their storage space to the desired

clients, establishing an agreement between the client and provider specifying the terms of how much data is going to be stored, the duration, and at what price.

Fortunately, the proliferation of tools and services has simplified the process of making content available over IPFS while also persisting the data via Filecoin. Our opted solution, *NFT.Storage*, is a combined solution of IPFS and Filecoin, ensuring that every content is discoverable and addressable while also guaranteeing data persistence. In this solution, storage providers commit their hard drive capacity to the Filecoin network and earn significant block rewards for doing so. This compelling incentive encourages the storage providers to offer free storage and retrieval, ensuring users incur no associated costs.

NFT.Storage operates similarly to traditional IPFS. Each uploaded content generates a unique CID based on the hash of the content and can be retrieved through a CID-based HTTP Gateway. However, this solution employs multiple Gateways for retrieval, dynamically selecting the fastest one or a combination of several to provide the fastest response possible to end-user requests.

Upon integrating NFT.Storage into our Fabric application, we noticed a significant decrease in file retrieval time but encountered an increase in the duration of the upload process. This increased duration of the uploading process was due to the fact that we followed a sequence of operations: uploading media files to NFT.Storage, waiting for complete upload to obtain the resulting CID, and then registering this information on our Hyperledger Fabric network. This approach proved time-consuming, particularly when uploading multiple media files (e.g., 4 to 5 files of around 500KB each), taking an average of 15 seconds for the operation. Given that registering media files is an important feature and is heavily used in our system, this delay would definitely be noticeable to an end-user.

To address this prolonged duration, we opted for an alternative approach: before the media files are uploaded, we generate their unique CIDs locally; subsequently, these CIDs are immediately registered on the blockchain as the media files are uploaded on the background, and this way ensuring minimal impact to the end-user. This revised approach significantly improved our system's efficiency, reducing the overall duration of such requests to a consistent average time of approximately 2.5 seconds and exhibiting greater resilience to variations in file quantity and size.

Table 5.1 illustrates the improvements in request duration resulting from the transition to our new implementation approach compared to the old method when simulating requests to our application through the available API endpoints. A consistent decrease in the duration of all requests is evident, highlighting the enhanced efficiency achieved.

Another alternative considered was the utilization of a private IPFS network to store data using personal IPFS nodes under our control and management. However, implementing such a solution is an intricate task and does not align with the specific challenges addressed in this work. Nonetheless, we recognize that this approach could potentially represent the most optimal option, particularly concerning data privacy and access control for uploaded content. However, although NFT.Storage uses Public Gateways, images can not be searched and retrieved in any other way besides having the corresponding CID,

Table 5.1: Comparison of Requests Duration: Old Approach vs. New Approach. The time is presented in seconds.

Request	Old approach avg. duration (s)	New approach avg. duration (s)
Add new document (1MB to 5MB)	6.2	2.9
Add new restoration with 2 files (up to 1MB)	6.4	2.4
Add new restoration with 5 files (up to 1MB)	14.8	2.5
Add new restoration with large files (up to 15MB)	8.8	3.1

which is accessible only to the authorized users via our Attribute-based Access Control scheme built into the *chaincode* logic, with the CID registered in the corresponding classic car asset stored on the blockchain. These considerations were acknowledged as a possible point for the future work of this project (please refer to section 7.2).

5.1.3.5 API Methods

Finishing the explanation of the Fabric application, it is important to describe the API functionalities and their respective endpoints. Almost every main function available on the smart contract, namely storing all information about classic vehicles, documentation, and associate restoration procedures as well as the access control logic, is available via the functionalities of our API, excluding some auxiliary and testing functions. Additionally, some functions have been introduced exclusively in the API, particularly encompassing user-related and transaction-related functionalities.

The API's functionalities are divided into the following resources:

- **/api/Users**: Encompasses user registration, authentication, and listing features.
- **/api/Classics**: Manages classic cars, including retrieval, registration, update of information, and listing with specific queries.
- **/api/Restorations**: Handles restoration procedures-related operations such as creating, updating, and retrieving restoration data.
- **/api/Access**: Encompasses access control and permissions to specific classic vehicles, including user roles and authorization.
- **/api/Transaction**: Manages the retrieval of transaction information.

As mentioned previously, the functionalities available in each endpoint present some type of data and logic validation. Additionally, there is a verification step for the session JWT token, reasonably excluding the user registration and login endpoints.

Furthermore, certain methods interact with another important component of our system, the Email Service. These operations specifically involve the necessity of sending emails to users, such as notifying them about a newly registered classic car on their account, providing updates on the information of an owned classic car, granting permissions to a particular vehicle, and other related actions. To implement this mailing service feature, we employed the *nodemailer* module for Node.js. Within this module, the Simple Mail Transfer Protocol (SMTP) protocol facilitates the delivery of electronic emails, and to integrate this feature with our project some minimal configurations needed adjustment. For the sake of simplicity, we utilized the email service from the Raimundo Branco workshop, requiring us to configure the service to use Gmail.

Appendix A provides a comprehensive list of all the developed endpoints within the respective resources, each accompanied by brief descriptions and additional details provided when necessary for clarification purposes. When applicable, it also specifies the invoked function within the smart contract.

For a more in-depth documentation of the developed API's functionalities, resources, and endpoints, including thorough descriptions, information on all required request path parameters, details about parameters passed in the body request, and the expected structure of responses and error messages, please refer to the available Swagger interface.

5.1.4 User Interface

To finish the overview of *ClassicsChain*, it is essential to delve into its GUI and front-end application. This is a crucial component of our solution since it is the "face" of *ClassicsChain*. It provides visual functionalities for users to interact with our solution, serving classic car owners and other entities or users, from restoration workshops or certification and regulation bodies. As explained in subsubsection 4.2.2.8, Angular was selected to develop this GUI.

Next, the primary features that users can leverage are outlined, displaying the main pages of the *ClassicsChain* Web Platform. A comprehensive video walkthrough of this GUI is also available at this link, offering a detailed showcase of the platform's features and usage flow.

As shown in Figure 5.2, a simple **Login** page was implemented to provide the authentication features. This page serves as the designated *home page* for users who are not authenticated. To authenticate on the platform, users are required to input their email address and password. Additionally, they must select the organization they wish to log in to in the dropdown select menu, with the options being User, Workshop and Certification that represent the **Owners Org**, **Workshop Org**, and **Certification Org**, respectively. This information is sent to the respective API's endpoint for authentication, and if the user is authenticated correctly, the JWT token is transmitted to the front end and stored in a cookie used for authorization. The user's email address and organization are extracted from this token, to present the expected elements according to the logged-in organization.

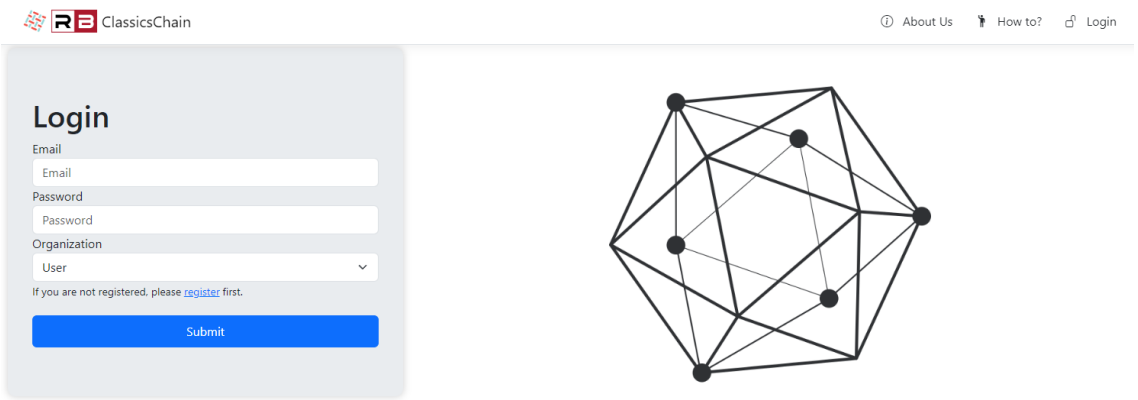


Figure 5.2: The Login page on the ClassicsChain website.

Figure 5.3 depicts the **Register** form, which new users in the system are required to complete to register in *ClassicsChain*. The fields to be filled align with those mentioned in the login process, with the addition of some demographic details, including the user's first name, surname, and country.

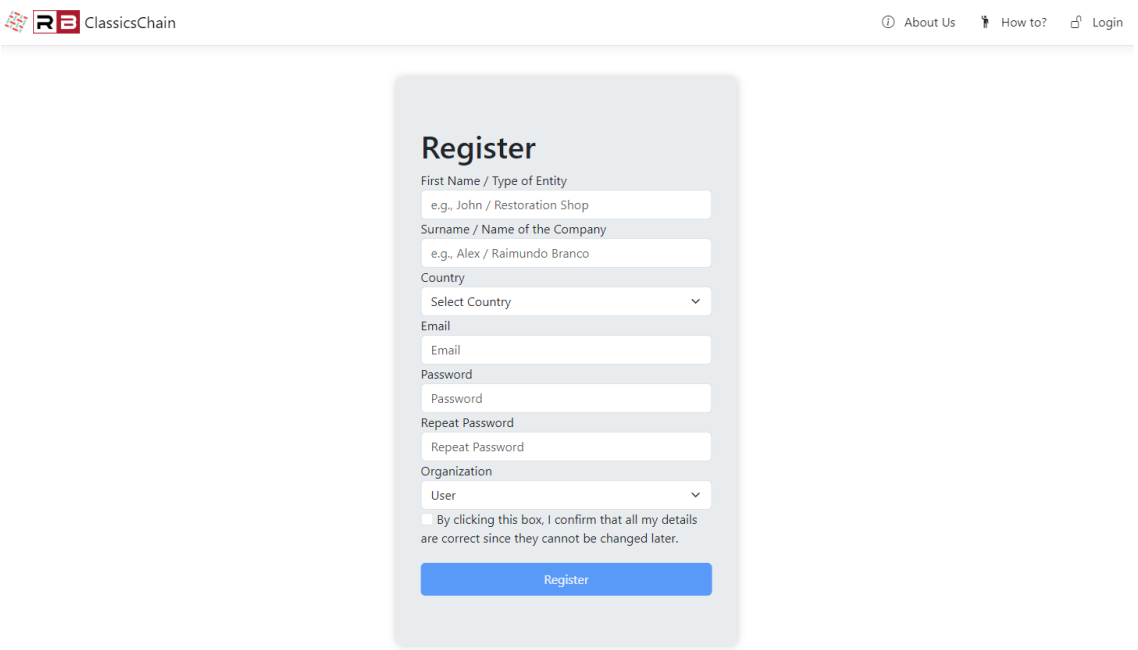


Figure 5.3: The Register page on the ClassicsChain website.

Upon user login, the platform presents the user's owned classic cars if logged in as a User (or no vehicles, if the user does not own any registered vehicles). Conversely, if logged in as a Workshop or Certification, the displayed vehicles are those for which the user has been granted permission to modify or certify, respectively. For each classic, a section is depicted containing the most relevant details of the car, along with a "Vehicle Card" button for further insights. This page is shown in Figure 5.4 and functions as the *home page* for authenticated users.

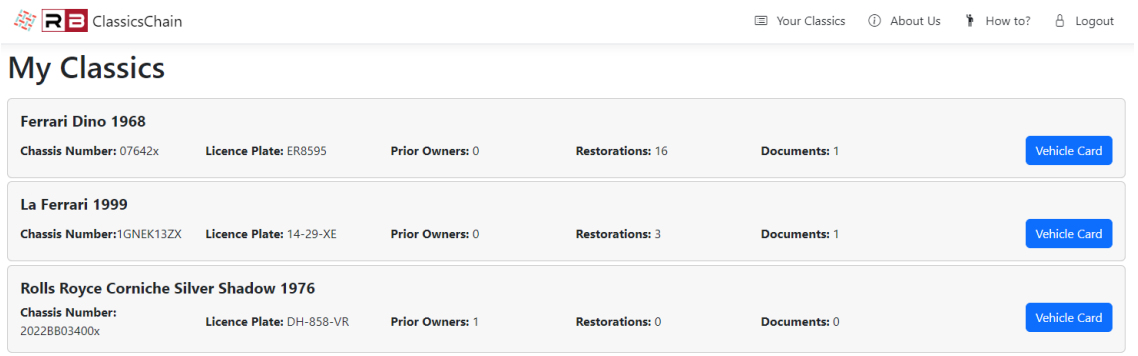


Figure 5.4: The "My Classics" page on the ClassicsChain website.

Furthermore, when logged in as Workshop or Certification entity, the page features a button allowing the registration of a new classic automobile in the system, as only these entities are authorized for such actions. Clicking this button opens a page with a form for registering a new classic. The form includes fields representing the fundamental details of a classic car, as illustrated in Figure 5.5.

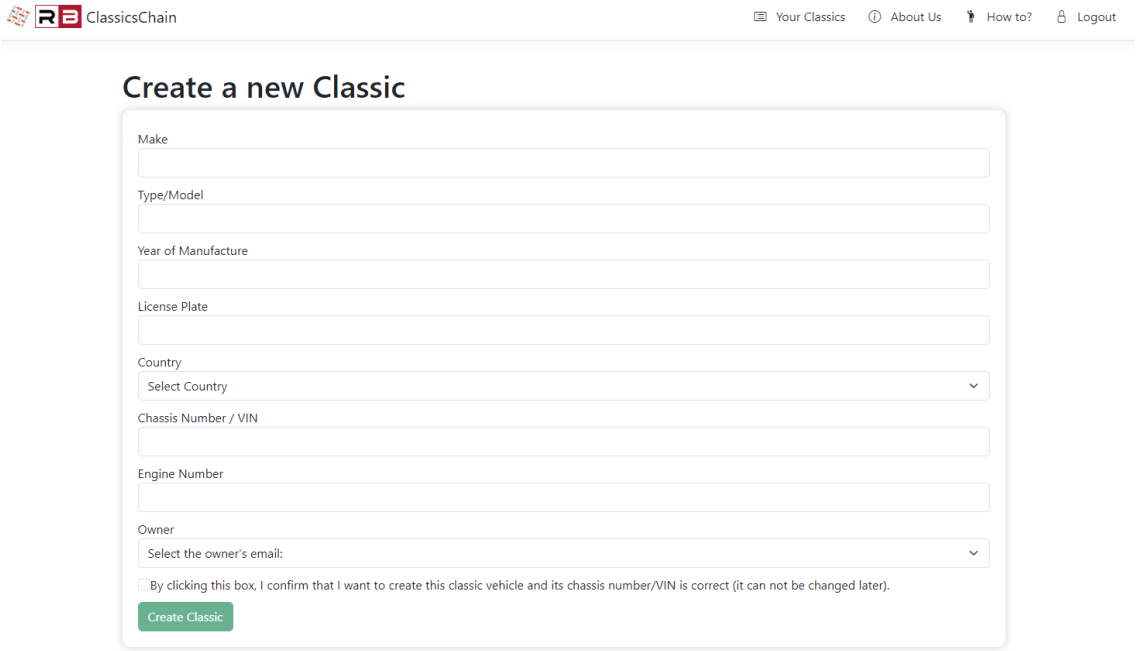


Figure 5.5: The "New Classic" page on the ClassicsChain website.

When clicking on the "Vehicle Card" button, the user is redirected to the *Vehicle Card* of the classic car. As mentioned previously, the *Vehicle Card* works as the ID card of the classic, where all information is detailed and it is the most important page of our solution. The *Vehicle Card* is divided into three distinct sections:

1. **Fundamental Vehicle Data**: This section provides the fundamental technical details of the car, such as the year and country of manufacture, license plate, and VIN. Other relevant information, such as the current owner of the car and a list of prior

owners, is also included. A button to navigate to the History of the *Vehicle Card* is also presented, and, if the current user is the owner, additional buttons are visible for editing fundamental details, managing access permissions, and reviewing the history of these permissions. This section is displayed in Figure 5.6.

2. **Restorations**: As depicted in Figure 5.7, this section documents all the restoration procedures applied to a classic car: Each restoration step includes the title, description, timestamp of registration, and the user responsible for the restoration. Steps with associated media feature buttons that, when clicked, reveal corresponding photos, or redirect to other media files such as videos or documents, as exemplified in the first step of this figure. Additionally, if the user viewing the *Vehicle Card* is the same that originally performed a particular restoration step, they have the option to edit details or associated media. Users with a modifier access level can also register new restoration procedures by clicking on the "Add new Procedure" button.

3. **Certifications and Other Documentation**: The last section of the *Vehicle Card*, presented in Figure 5.8, is comprised of a list of the entities that emitted certifications or authentications to that particular vehicle and a list of additional documentation to be appended to the *Vehicle Card*. Such supplementary materials may encompass legal documents, inspection or insurance records, warranties for specific parts of the vehicle, or any other relevant documentation. In this instance, the additional documentation is the certificate of restoration issued by Raimundo Branco and automatically generated through the *Charter of Turin Monitor*.

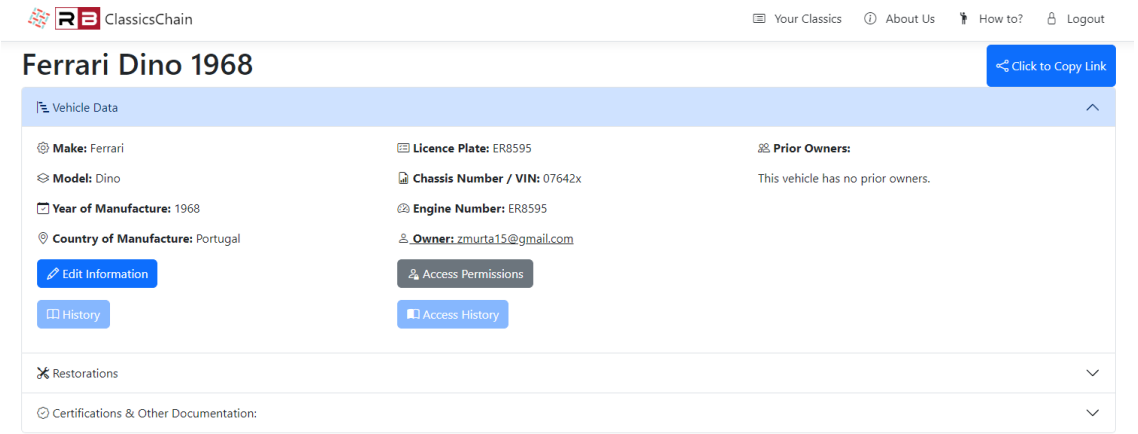


Figure 5.6: The "Fundamental Vehicle Data" section of the "Vehicle Card" page on the ClassicsChain website.

The page for **editing fundamental details of classics** is a straightforward form page accessible only to the vehicle owner. Here, the owner can modify all the fields, with the exception of the chassis number. While all other fields are modifiable, it is imperative that these are not left empty. For an illustrative representation of this page, refer to Figure 5.9.

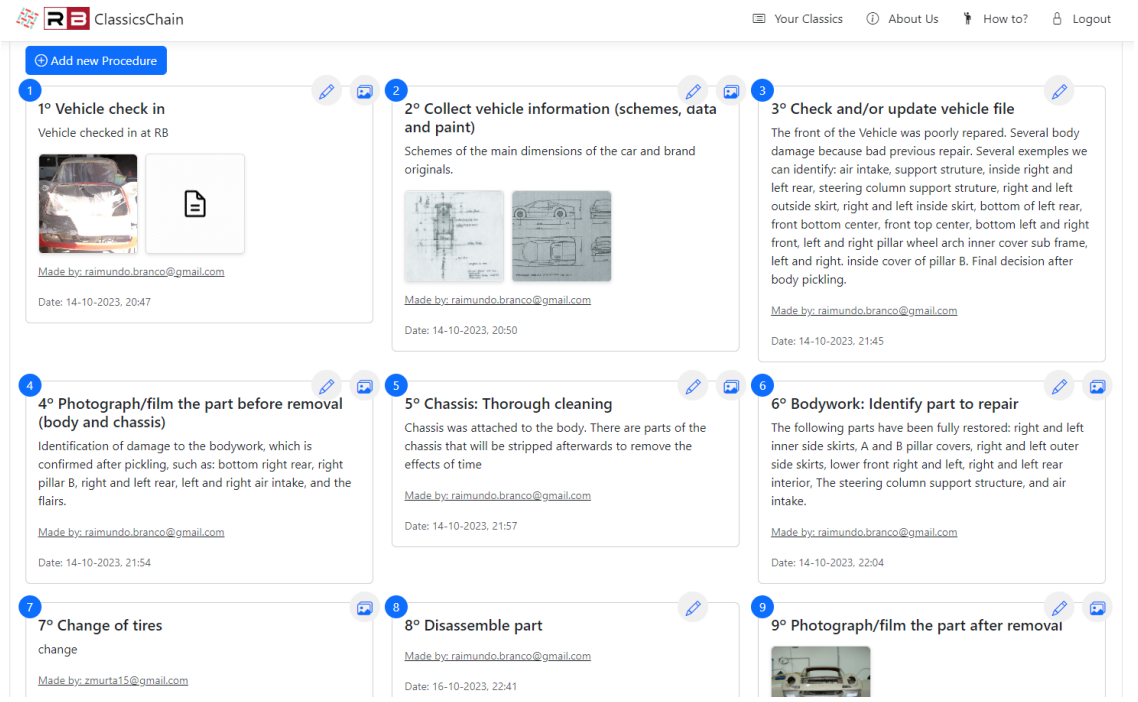


Figure 5.7: The "Restorations" section of the "Vehicle Card" page on the ClassicsChain website.

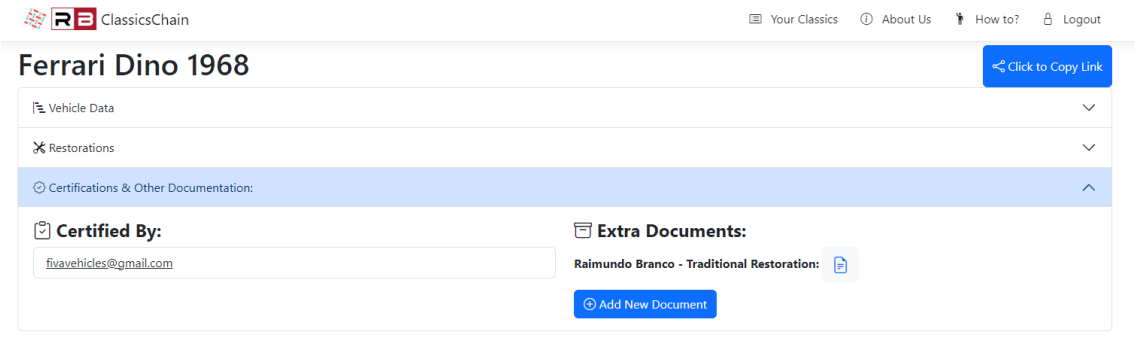


Figure 5.8: The "Certifications and Other Documentation" section of the "Vehicle Card" page on the ClassicsChain website.

Figure 5.9: The "Edit Classic" page on the ClassicsChain website.

The page to **register a new restoration procedure** is depicted in Figure 5.10. It is a simple form page where the user submitting the new step is required to furnish a title and description and, optionally, include supplementary materials such as photos, documents, or any other pertinent media files. Access to this page and the permissions to register a new procedure are exclusive to users with a modifier access level.

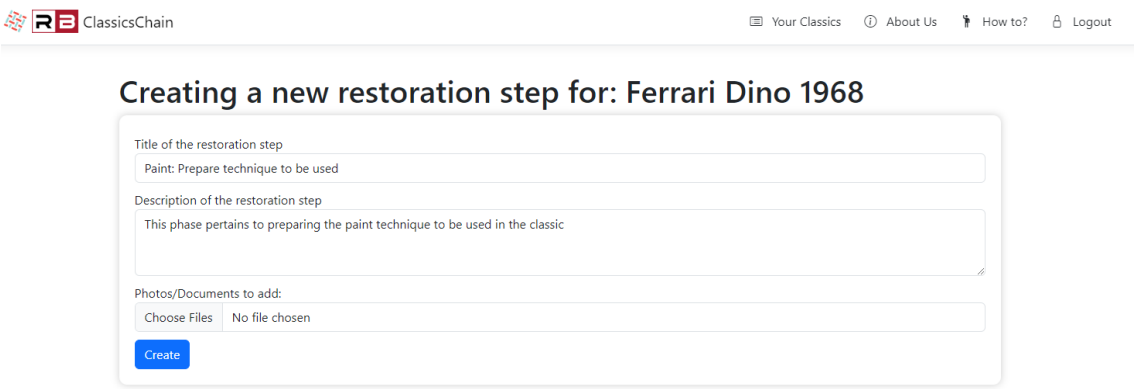


Figure 5.10: The "New Restoration Step" page on the ClassicsChain website.

The **History of the Vehicle Card** is also one of the focal points of our solution, displayed in Figure 5.11. In this page, it is showcased a chronological log of every version of a Vehicle Card. Whether it is a minor modification, such as a single-word modification in a restoration step's description, or a complete transfer of ownership to a new user, every update is documented here. Each version is displayed with the transaction ID that originated it, the timestamp of the transaction, and a concise version of the *Vehicle Card*.

To ensure absolute transparency and traceability, users have the option to access details about each version by clicking on the respective transaction ID. A pop-up window is

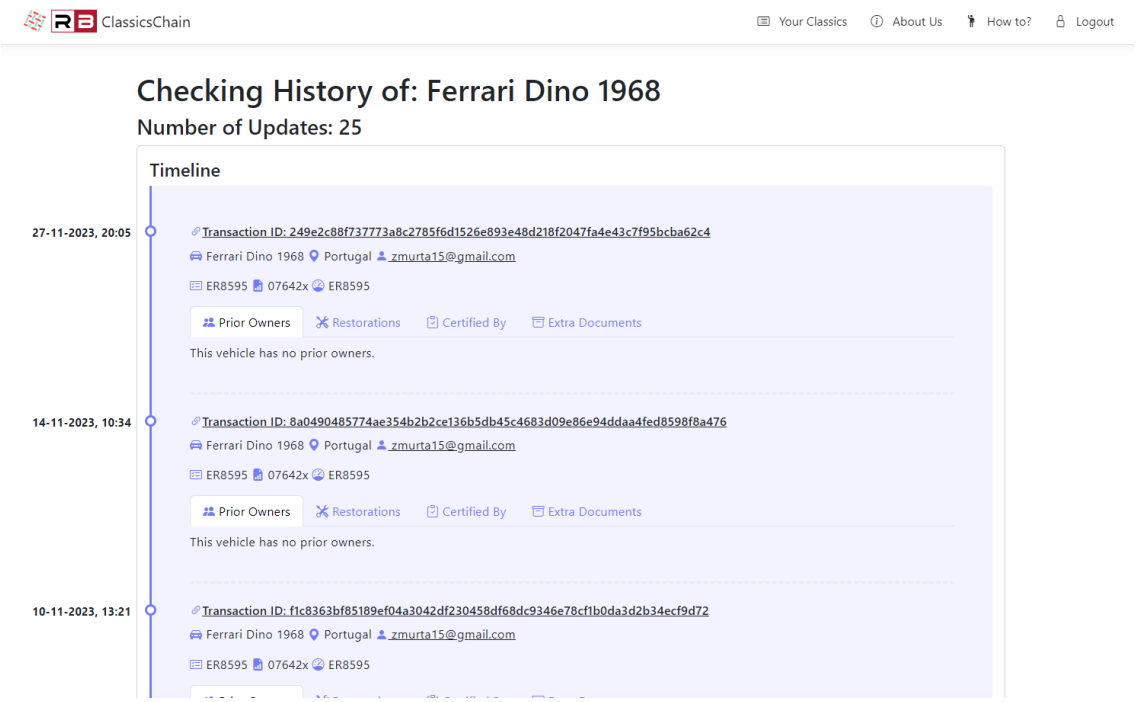


Figure 5.11: The "History of the Vehicle Card" page on the ClassicsChain website.

displayed, presenting key information associated with the transaction. This includes the transaction timestamp, the email and organization details of the individual transaction submitter, the name and arguments of the executed smart contract function, as well as the message and status code of the function's result. This pop-up is illustrated in Figure 5.12

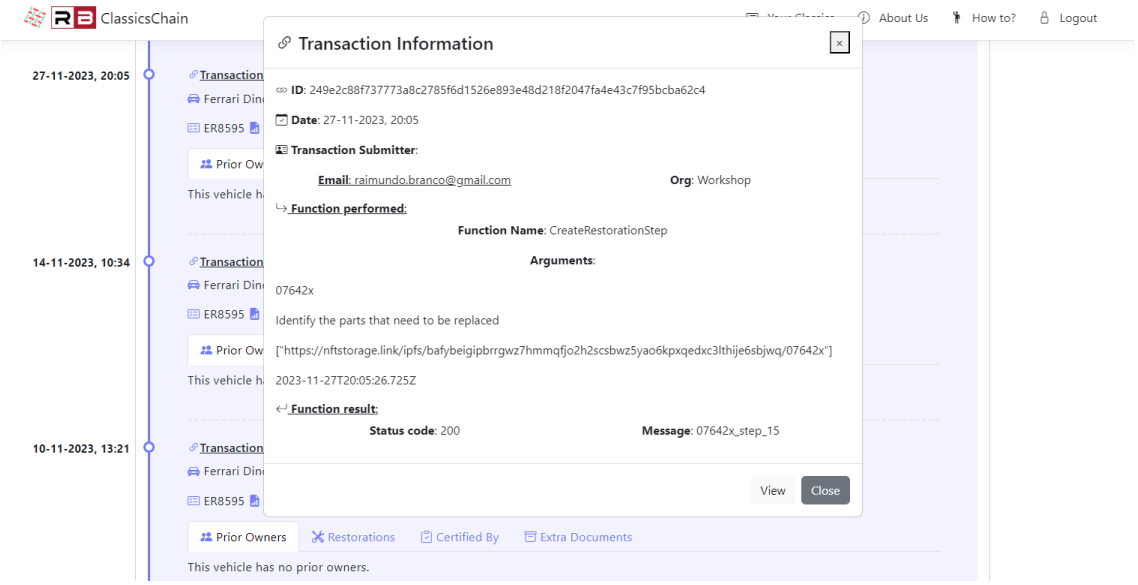


Figure 5.12: The "Transaction information" pop-up of the "History of the Vehicle Card" page on the ClassicsChain website.

Figure 5.13 depicts the **Access Permissions** page. Exclusively available to the owner of the classic car, this page provides the interface for managing access permissions to

the owned vehicle. The owner can verify which entities have permissions and their corresponding level, along with the corresponding timestamps indicating when these access levels were granted. Additionally, the owner can also grant or revoke new access permissions by selecting the action (grant or revoke), the access level, and the targeted user for permission adjustment in drop-down menus. Ultimately, the owner can also transfer the complete ownership of his classic vehicle to another user.

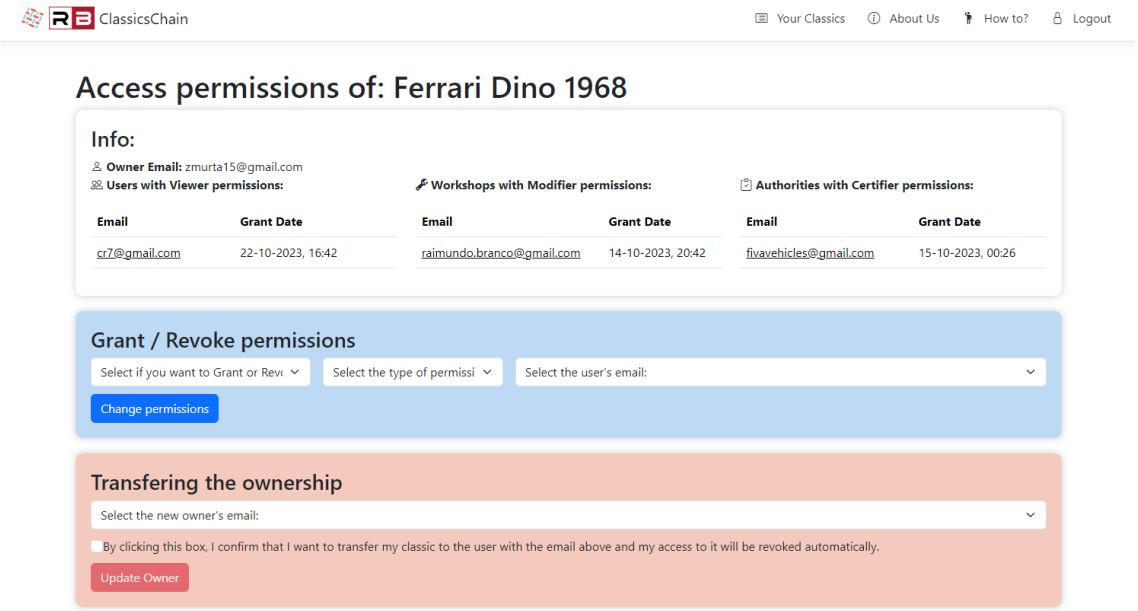


Figure 5.13: The "Access Permissions" page on the ClassicsChain website.

The car owners also have exclusive access to another important page, which is the **Access History** page. As depicted in Figure 5.14, this page has a similar purpose and structure to the **History of the Vehicle Card** page. However, instead of documenting the history of updates of a *Vehicle Card*, it presents the history of updates of the granted and revoked permissions for a specific classic automobile. This page showcases a chronological timeline view of all the permissions granted and revoked to a particular classic car. Each version of the access permissions is displayed with the transaction ID that originated it, the timestamp of the transaction, info about the owner, and a list of users along with the respective timestamp of permission granting for each type of access permission.

Similar to the **History of the Vehicle Card** page, the transaction IDs in the **Access History** page are clickable facilitating a proper look into the details of each version and enhancing the overall transparency and traceability of our solution. A pop-up window is displayed, presenting the transaction timestamp, the email and organization details of the user who submitted the transaction, the name and arguments of the executed smart contract function, as well as the message and status code of the function's result. A pop-up for the **Access History** transaction is illustrated in Figure 5.15.

All the email addresses from the users displayed throughout the platform are clickable to return more information about that particular user as shown in Figure 5.16.

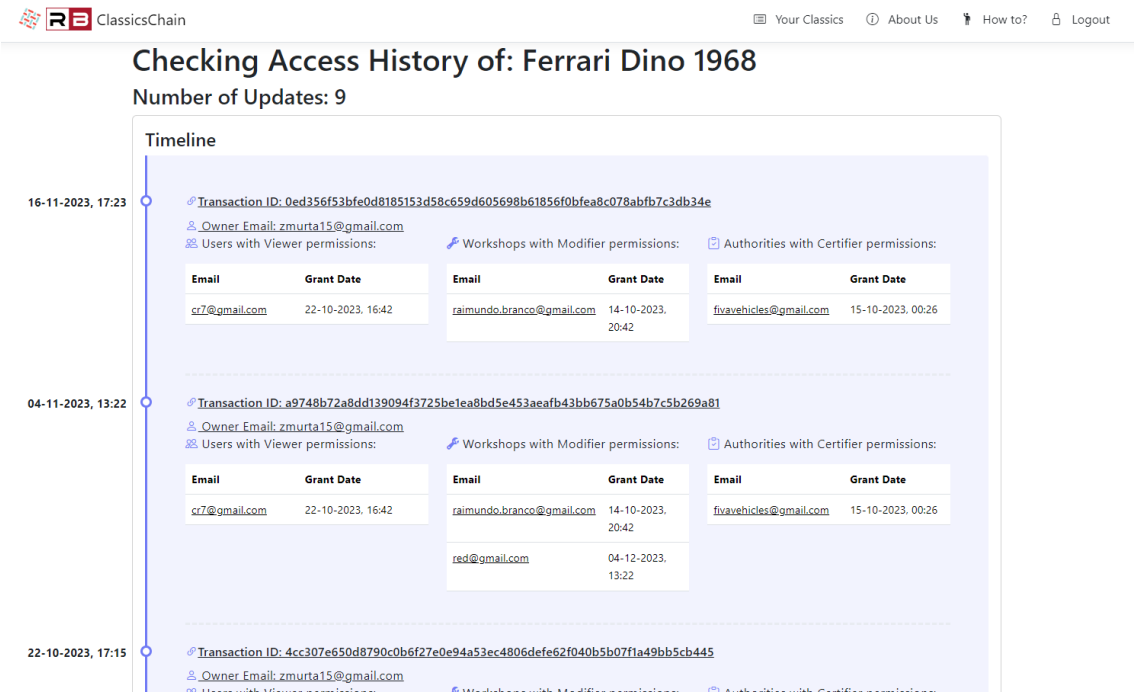


Figure 5.14: The "Access History" page on the ClassicsChain website.

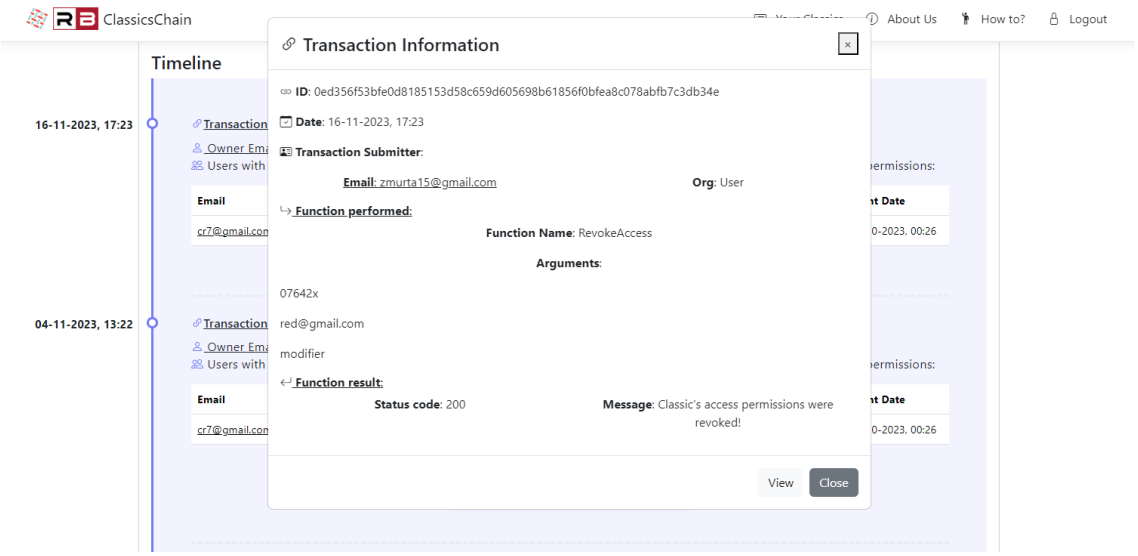


Figure 5.15: The "Transaction information" pop-up of the "Access History" page on the ClassicsChain website.

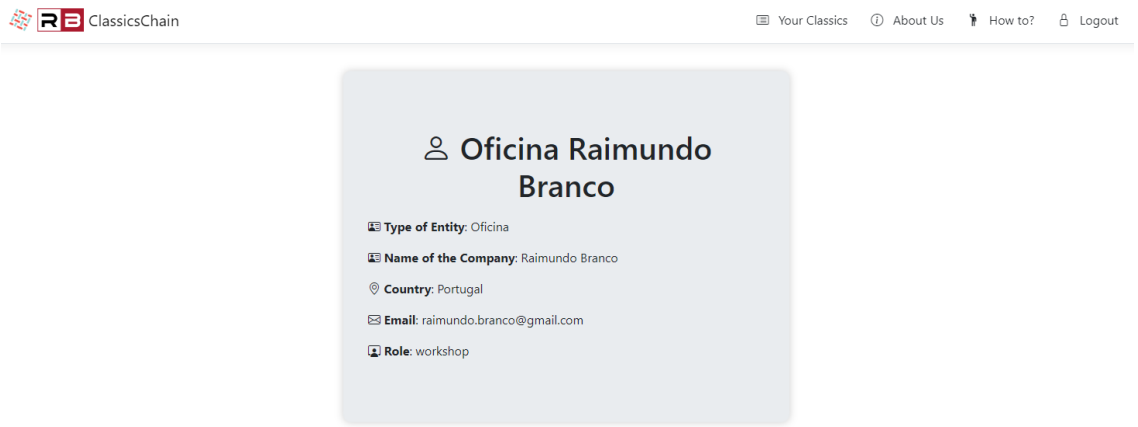


Figure 5.16: The "User Details" page on the ClassicsChain website.

The development of this GUI followed some of the best practices expected for a front-end implementation, encompassing security measures. In addition to server-side input validation, the front-end application reinforces mandatory fields and data validation, introducing an additional layer of verification. Moreover, every request (logically excluding the login and register requests) made through this GUI requires a valid authentication token. In the event of an error, (e.g., the user is not authorized to view a specific classic car's Vehicle Card) an informative error message is displayed without revealing any sensitive information. It is also important to note that we have deployed and configured the front-end application to enforce the HTTPS protocol. This configuration enhances the application's overall security and privacy. By ensuring that all data transmitted between the client-side application and the server application is encrypted, security and privacy concerns are addressed, guarding against potential threats or unauthorized access.

Transitioning to usability best practices, several measures have been incorporated into the *ClassicsChain* GUI to enhance the user experience. Firstly, this interface was developed with a responsive design and consistent navigation, ensuring seamless navigation flow and interactions regardless of the device and screen size. Confirmation alerts, confirmation checkboxes, and loading indicators are also utilized to contribute to an intuitive user experience, especially in operations that involve sensitive data or require additional processing time. As mentioned in the preceding paragraph, appropriate informative errors are also displayed, and a help guide is also available to assist users in using this platform and executing its functionalities.

5.2 Charter of Turin Monitor

As stated multiple times throughout this document, the current work is the continuation of the work developed in [5], which led to the creation of the most recent version of the *Charter of Turin Monitor*, superficially discussed in subsubsection 4.2.1.5. To briefly reintroduce the context, the *Charter of Turin Monitor* is a web application, currently in use

at Raimundo Branco, specially developed to record restoration workflows as Business Process Model and Notation (BPMN) models in adherence to FIVA's Charter of Turin. This platform facilitates progress tracking and evidence documentation through a graphical interface. Users can also download an automatically generated document summarizing all the restoration steps and associated evidence.

In the next paragraphs of this subsection, we will delve into the implementation and development that took place during our work to refine the *Charter of Turin Monitor*, implement new functionalities, and how we integrated it with *ClassicsChain*. The complete code repository for the *Charter of Turin Monitor* is available at this link.

In the beginning stages of the current research work, it was imperative to have a firm understanding of the work already done by P. Moura [5]. For this, we followed the steps provided in his research to set up and deploy the preexisting system. Furthermore, a study of the technologies already in use had to be conducted to understand how the system is built fully and facilitate the implementation of new functionalities. This initial knowledge-gathering phase was also essential to have a more refined understanding of the classic car industry, and the particular demands of customers in this sector, which indirectly contributed to the development of *ClassicsChain*.

From this learning phase, it was possible to promptly understand the key features and aspects from the previous work that would be valuable and relevant to address the functionalities needed to be implemented and the further integration with *ClassicsChain*. These include:

- The web application's back end and API implementation using ASP.NET, a framework for building web apps and services with .NET and C#.
- *Camunda*, an open-source platform, was used to create the BPMN process diagrams Decision Model and Notation (DMN) decision tables, as well as to manage the workflows of these models. Its API is also crucial to access the details of the diagrams and workflows.
- The building of the application's GUI using Angular, with *BPMN.io* employed to integrate a viewer for visualizing and editing BPMN models in the front end.
- Storage of media files on *Pinterest*, with retrieval and storage managed through the Pinterest API.
- The generation of an automatically created document detailing the complete restoration process that took place for a specific classic car is achieved using LateX. This document is generated in pdfLaTeX format.

5.2.1 Enhancements and Additional Features

Initially, we focused on implementing some simple features and modifications, completely unrelated to *ClassicsChain*, that have been collected from the hands-on experience and

feedback of the main users of the *Charter of Turin Monitor*, namely the workshop workers and administrators. These primarily include features aimed at improving user experience and usability. Notable additions include a loading indicator to represent ongoing operations, providing users with feedback during execution. Additionally, alert messages were incorporated for operations with extended durations, and confirmation dialog boxes were introduced for operations involving sensitive data.

Moreover, adjustments were made to the reading and handling of responses from specific Pinterest API requests that previously caused failures in generating the PDF file that details the restoration process.

The automatically generated PDF also had some visual upgrades. Each photographic evidence present in this document is now displayed accompanied by a caption, a suggestion from classic car owners who may not always be aware of the specific procedure depicted in a photo. To develop this feature, we utilized the *Note* string field that every item recorded in Pinterest possesses, allowing workshop administrators to manually add captions directly on the Pinterest web platform to specific photos. This implementation decision was made according to the preferences expressed by workshop administrators regarding how they would prefer to use this feature. Figure 5.17 illustrates a section of the restoration PDF, showcasing images with their corresponding captions.

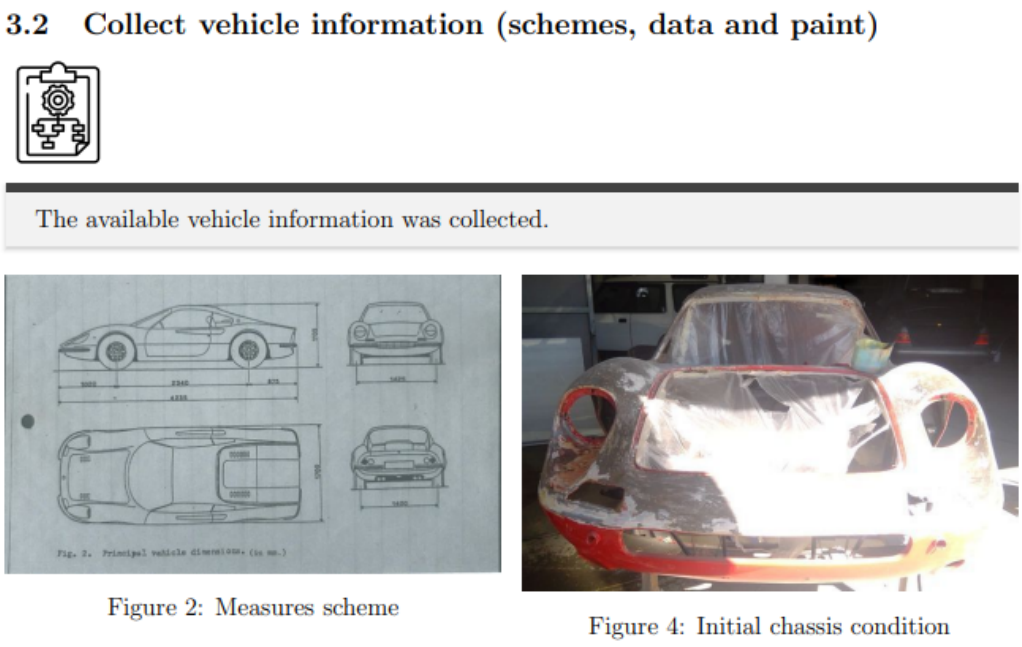


Figure 5.17: Images accompanied by captions in the generated restoration PDF document.

Another relevant addition to the generated PDF involves the incorporation of a verified Glasurit painting certificate appended to the end of the document. This certificate includes the color reference, applied technique, and other painting information. It serves as an exclusive certificate issued by the Raimundo Branco workshop, validating the authenticity

of the paintwork using Glasurit paint, adhering to FIVA guidelines. This certificate is automatically generated and attached to the PDF, once the necessary painting details are registered to a specific car. Furthermore, the certificate template was constructed in LaTeX and attached to the original PDF template.

5.2.2 Integration with *ClassicsChain*

The integration of the *Charter of Turin Monitor* with *ClassicsChain* was made through the invocation of requests to the endpoints from the *ClassicsChain* API. Existing functionalities within the *Charter of Turin Monitor*, utilized daily in the Raimundo Branco workshop, were augmented with additional features from *ClassicsChain*. Through this approach, specific functionalities within the *Charter of Turin Monitor* extend their reach by making requests to the *ClassicsChain* API. This collaborative integration enhances the overall capabilities of the *Charter of Turin Monitor*, recording all the details and restoration tasks of classic cars under intervention and making this documentation accessible to all interest entities that contribute to the classic automotive sector, not necessarily related to Raimundo Branco workshop. This integration approach enables the administrators from the workshop to continue using the already familiar *Charter of Turin Monitor* as the main platform for most of the functionalities, but simultaneously ensure the recording of all relevant information and pieces of evidence with *ClassicsChain*.

When logging into the *Charter of Turin Monitor* with an *admin* role, representing a workshop administrator, the system simultaneously logs into the corresponding Raimundo Branco administrator account and the **Workshop Org** on *ClassicsChain* through the Login endpoint. The returned JWT token is stored in a cookie to facilitate subsequent requests. With this approach, it ensures the maintenance of distinct tokens for a specific admin session: one for *Charter of Turin Monitor* functionalities and another for *ClassicsChain* functionalities. Both these tokens are valid for five hours so when the user gets logged out, it gets logged out automatically from both.

The user registration, which in the *Charter of Turin Monitor* is a process exclusive to administrators, was extended to facilitate automatic user registration as a classic car owner in *ClassicsChain* if that particular user was not already registered in the system. In this case, a randomly generated password is assigned, and the credentials are dispatched to the user via email.

When a workshop administrator is logged into the *Charter of Turin Monitor*, he is able to register a new classic car for a future restoration at Raimundo Branco. As known, in *ClassicsChain*, to be able to register a new classic, the submitting user needs to hold a role within the **Workshop Org** or **Certification Org**, and a workshop administrator from Raimundo Branco in the *Charter of Turin Monitor* corresponds to a user within the **Workshop Org** of *ClassicsChain*. This way, when an admin registers a new classic on *Charter of Turin Monitor*, it automatically reflects the registering on *ClassicsChain* if that particular car, identified by its unique chassis number or VIN, is not already registered in

the system. This process mirrors the direct registration of a classic car in *ClassicsChain* but utilizes the interface and functionalities of the *Charter of Turin Monitor*.

The most significant feature of this integration is the automatic recording of each restoration procedure in *ClassicsChain* once this restoration task is registered by an administrator on the *Charter of Turin Monitor*. To achieve this, the classic car must be initially registered on the *Charter of Turin Monitor* or granted modifier access permission beforehand. When an administrator registers a new restoration task on the *Charter of Turin Monitor* with a title, description, and associated media (photos, videos or diagrams), this operation is reflected to the corresponding *Vehicle Card* on *ClassicsChain* with the same arguments passed in the request's body, along with the registration current time and the submitting user, who corresponds to a workshop administrator of Raimundo Branco. Upon completion on the *ClassicsChain* side, the resulting task identifier is retrieved and stored in the Task object of the *Charter of Turin Monitor*, representing the executed restoration task. To implement this, a "BlockChainId" field was introduced to the Task object, storing the identifier on the *ClassicsChain* system. This approach establishes a One-to-One relationship between the restoration task in the *Charter of Turin Monitor* and its counterpart in *ClassicsChain*.

The edit of the details such as title, description, or associated media of a particular restoration procedure on *ClassicsChain* is also possible once edited by an administrator through the *Charter of Turin Monitor*. When edited on the *Charter of Turin Monitor*, the submitted transaction originates a new version of the *Vehicle Card* on *ClassicsChain*, akin to the process when utilizing *ClassicsChain* directly.

Regardless of whether the logged-in user on the *Charter of Turin Monitor* is a workshop administrator or a classic car owner, they can access the details of a specific car restoration project, provided they have access to that particular vehicle. A newly added "Vehicle Card" button on the project details page facilitates the transition from the *Charter of Turin Monitor* to the specific classic vehicle's *Vehicle Card* on *ClassicsChain*. This button is visible in the project details page of the *Charter of Turin Monitor* and is illustrated in Figure 5.18.

The overall integration process was straightforward, primarily involving the submission of requests to the *ClassicsChain* API and subsequent response handling, which included parsing, extracting, processing, and storing data to perform various actions.

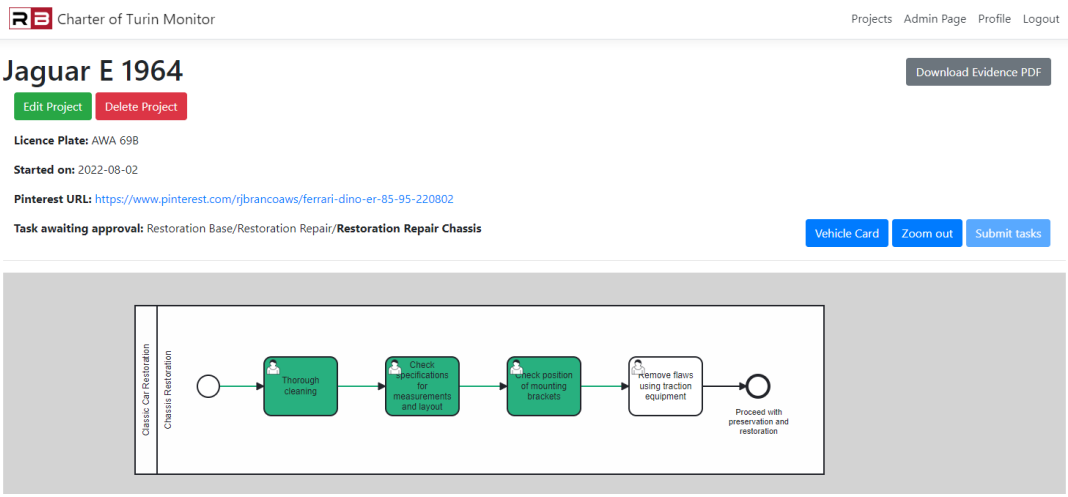


Figure 5.18: Updated version of the classic car project page in the *Charter of Turin Monitor* with the "Vehicle Card" button.

6

Verification and Validation

This chapter aims to present the validation process of the system proposed in this work. In section 6.1, we provide an overview of the complete validation process and efforts undertaken. Subsequently, we delve into the specifics of performance evaluation, outlining the tests conducted in detail in section 6.2. Following this, in section 6.3, we explore the relevance evaluation of our solution as well as the usability evaluation of the ClassicsChain's GUI, while also addressing the potential threats to the validity of this evaluation. Finally, section 6.4 describes the communication efforts to the dissemination of knowledge and the sharing of results from this work.

6.1 Overview

As presented in section 1.6, validating our solution is an essential phase of the Design Science Research (DSR) methodology, verifying that our solution can fulfill all the proposed objectives and contributions. We will now describe how it was evaluated using meticulous methods and various techniques to determine its overall quality, performance, relevance, and usability.

Not only were performance tests iteratively conducted on our solution, assessing the blockchain network's performance and evaluating the duration of specific requests to the developed API as the system evolved and integrated new features, but also continuous interaction and feedback was sought from workshop employees, past project participants, and knowledgeable stakeholders in the classic car ecosystem, including classic car owners and industry specialists.

This ongoing process of monitoring and gathering feedback provided a clear understanding of the essential features to be incorporated into *ClassicsChain* and identified potential enhancements for this solution. Moreover, this consistent interaction with workshop workers also helped us upgrade and modify the features of the *Charter of Turin Monitor*. While not the primary focus of this work, the *Charter of Turin Monitor* remains essential to the project's overall progress.

It is also important to mention that throughout our project's development, we emphasized code quality and adherence to best practices. To ensure this, we used the *Sonarlint* extension within our IDE to meticulously refine the code of all individual components of our solution. Furthermore, we employed the *Sonarcloud* platform, which enabled a complete evaluation of the entire project, ensuring that our code met established quality standards and best practices guidelines. This commitment to code quality was geared towards delivering a well-developed solution following the predefined goals and contributions.

Another significant aspect of our validation, as it will be properly described in section 6.3, was the participation of two prominent stakeholders in the classic car ecosystem in Portugal in our solution's relevance and usability evaluation.

6.2 Performance Evaluation

To assess the performance of our solution, our main focus centered on the *ClassicsChain* Fabric application and API through stress tests, both sequential and concurrent, since this is the component that users will interact with. Users engage with it either through the *ClassicsChain* front-end client or other clients invoking requests from our REST API deployed on our web server. It is important to note that end-users, such as classic car owners or any other entities that use the *ClassicsChain* web platform, do not interact directly with our blockchain network.

Nevertheless, we also conducted evaluations of the individual performance of the Hyperledger Fabric blockchain network, the cutting-edge technology of our solution. This assessment aimed to comprehend the unique performance characteristics of this component to gauge its impact on the overall system and dissociate it from the performance of the Fabric application.

Both evaluation techniques were executed iteratively during the development process. This iterative approach provided a comprehensive understanding and facilitated comparisons with previous iterations of our solution. These comparisons were particularly valuable given the evolving feature set, allowing us to maximize performance in the course of this work.

6.2.1 Blockchain Network Evaluation

For the individual blockchain network's performance evaluation, we employed *Hyperledger Caliper*. Caliper stands out as a benchmark tool designed specifically for assessing the performance of blockchain networks, not limited to Hyperledger Fabric but also extending to others, such as Ethereum networks. In our context, it was used to assess the performance of our Hyperledger Fabric blockchain network individually, analyzing the performance through the invocation of various functions directly from the smart contracts deployed in the network.

To execute the evaluation of our blockchain using Caliper, we needed to develop Node.js tests using the *workload* module provided by Hyperledger Caliper. These tests must be configured to perform either read or write operations corresponding to the invoked function. Other parameters, such as the send transaction rate, number of test rounds, and test duration, need to be configured. Multiple iteration measures were executed, encompassing diverse workloads and varying transaction rates for both read and write operations to the blockchain. The performance metric was quantified in terms of Transactions Per Second (TPS).

The results from the tests using Caliper are displayed in Figure 6.1. These results were obtained from evaluating the most up-to-date version of the blockchain network and smart contract and allowed us to conclude the limits of our blockchain network in terms of supported TPS for both read and write operations, clearly defining a maximum threshold for these operations.

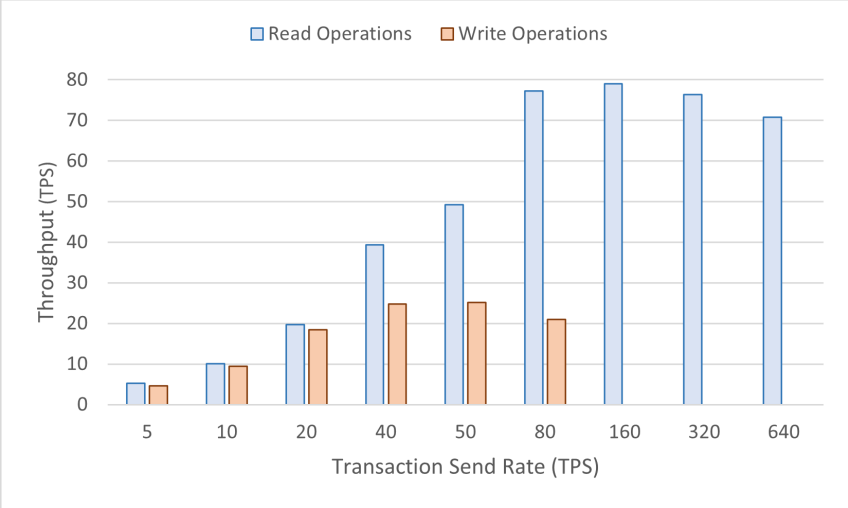


Figure 6.1: Throughput results of our Hyperledger Fabric blockchain network.

It was observed that the TPS for read operations reaches approximately 70 TPS before experiencing a decline in throughput as the send rate increases, accompanied by a notable rise in average latency. Importantly, no transaction failures were recorded regardless of the increasing send rate and increasing latency.

In contrast, the peak throughput for write operations was measured at around 22 TPS. As the transaction send rate increased, the latency showed a more pronounced increase and noticeable transaction failures occurred when the send rate exceeded 80 TPS (this explains the absence of additional values beyond this threshold in Figure 6.1). Nonetheless, despite increasing transaction failures attributed to the network overload, no complete network failures were identified. Instead, only the communication between components experienced temporary interruptions, quickly reestablishing connectivity.

The disparity between the throughput of the two operation types is justified by the fact that write operations involve the consensus mechanism and consequent communication

among the network's components, significantly impacting the maximum achievable TPS. In contrast, read operations do not involve the consensus process and typically only query the state database of the network's nodes.

The Caliper tool also allows us to record latency duration in seconds. In typical network conditions and transaction send rates, we obtained average values of 0.16 seconds for read operations and 1.52 seconds for write operations.

We recognize that the results obtained from stress tests using Caliper, particularly in assessing the maximum achievable throughput, do not reach optimal performance, especially compared to similar works employing network topologies and configurations. However, it is crucial to note that these limitations may not carry significant weight within this project's scope. Given that classic car documentation is intended for a very specific niche, the classic car sector, rather than a wide audience, the impact of these constraints on the end-user experience should not be significant.

Nevertheless, it is imperative to acknowledge these limitations conscientiously since the reliability and integrity of our solution can be impacted. Potential factors contributing to the observed impact on performance include the network topology, channel configuration, the endorsed policy definition, and the intricacy of the deployed smart contracts. It is also important to remain mindful of the possibility of flaws in the configuration of the Caliper tool and the development of testing scripts.

6.2.2 Fabric Application and API Evaluation

Moving to the evaluation of the Fabric Application and API, as emphasized earlier in this section as the primary performance evaluation for end-users, this evaluation is crucial whether users interact through the *ClassicsChain* front-end client (connecting to our API) or other clients making requests directly via our REST API. This performance evaluation was conducted using both shell scripts and *Postman* performance features, to simulate real users using this application. Postman was mainly used to perform parallel tests, simulating a desired number of virtual users, and integration tests comprised of diverse functionalities and activity flows.

This evaluation aimed to determine the duration of the most important functionalities available through the respective endpoints. This assessment is essential as it evaluates the waiting time for end-users, from executing a specific request to obtaining the expected response, commonly referred to as load time.

A dedicated shell script was developed for each major functionality in *ClassicsChain* to assess its duration. Each test was iteratively executed 10 to 30 times to mitigate potential performance concerns, obtaining the minimum, average, and maximum duration.

The duration of user management and related requests is illustrated in Figure 6.2. Notably, a significant difference is observable for the registration of a new user in the system. This is the only write function from this set, and the only one that uses both the Hyperledger Fabric SDK to generate a new identity in the blockchain network and

establish a connection with the MongoDB database to register this new user. In contrast, the remaining functionalities demonstrate lower duration times, primarily involving straightforward read operations or a login function (verifying previous registration and returning a session token), solely interacting with the MongoDB database.

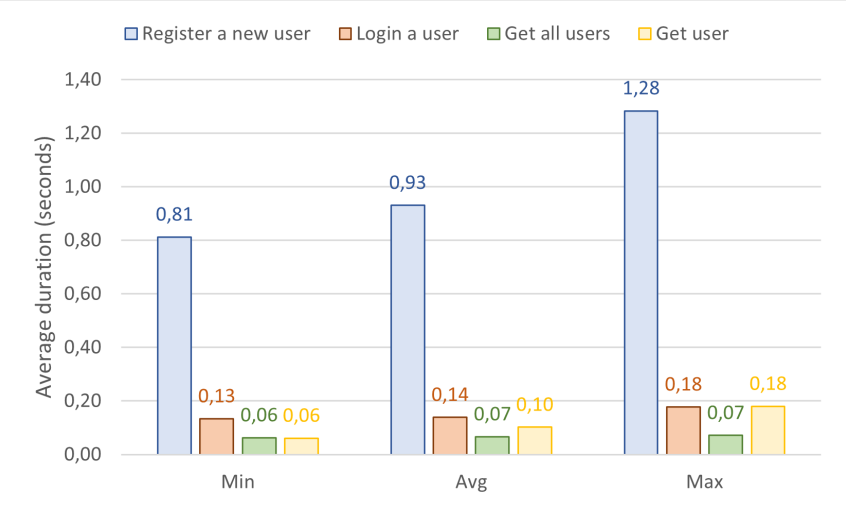


Figure 6.2: Duration of user management requests, in seconds.

Similar duration times are exhibited regarding the simplest functions that invoke write operations on the blockchain. These functions do not interact with the media storage or other components, except for some that trigger email sending through the Email Service, and record only a few hundred bytes. These encompass requests such as registering a new classic car in the system, altering the ownership of a specific vehicle, certifying a particular car, and granting or revoking access and their corresponding duration times are depicted in Figure 6.3, resulting in an average time of 2.35 seconds for this specific set of requests.

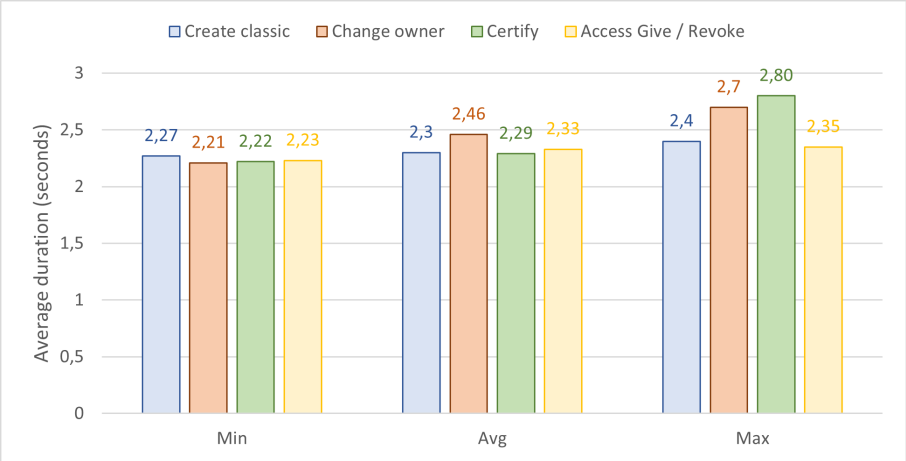


Figure 6.3: Duration of the simplest write requests, in seconds.

Considering the most logically complex functions, we focus on those that interact with our media storage component and perform write operations on the Fabric network.

From this set, the performance of all the functions was analyzed, emphasizing the most critical functionalities of our system. For this purpose, we assessed the duration times of registering a new restoration procedure under various scenarios: with zero associated files, up to two files, five or more files, and finally, with one or two associated videos. The time taken to register a new document was also evaluated. It is worth clarifying that, the files encompass any media files of any type within a capacity range of 10KB to 1MB. Larger files, such as documents like PDFs, have a size between 1MB and 5MB, while videos refer to video content files ranging from 2MB to 15MB.

The results for this set of functions are showcased in Figure 6.3. As expected, the duration time for registering a restoration procedure without any associated files aligns with the previously mentioned average for the least complex write operations. As the number and size of files to be recorded increase, the duration time also increases, albeit not by a significant margin, as depicted in the duration times of both the requests adding video files and adding documents. Importantly, across all requests, the average duration times consistently stay within the 3-second range, except in cases involving the registration of video content. Although not displayed, the behavior and respective duration times for updating restoration procedures mirror these results.

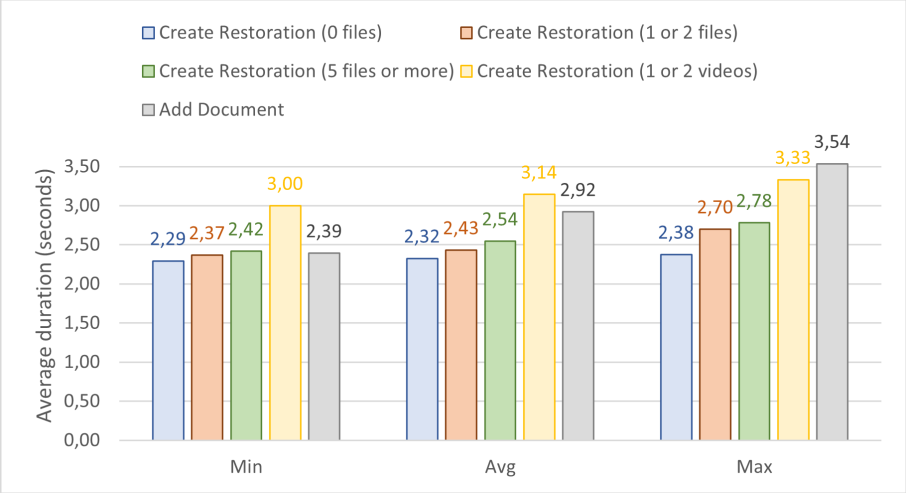


Figure 6.4: Duration of write requests that interact with the media storage, in seconds.

As previously explained when describing the media storage integration and implementation, in earlier iterations of our solution, these types of requests were considerably slower, prompting the need to revise their implementation to enhance performance and achieve more acceptable duration times.

Concerning the requests involving read operations to the Fabric network, The results of the query request invoked to retrieve current information about a classic car, essentially the *Vehicle Card*, are displayed in Figure 6.5. This figure shows that reading a relatively low amount of data from the network (up to 50KB) typically demands short periods, averaging around 0.10 seconds. As the data increases reaching a standard threshold expected for this request (up to 100KB), there is a noticeable increase in duration times, potentially

tripling the time required to complete the request, reaching up to 0.30 seconds.

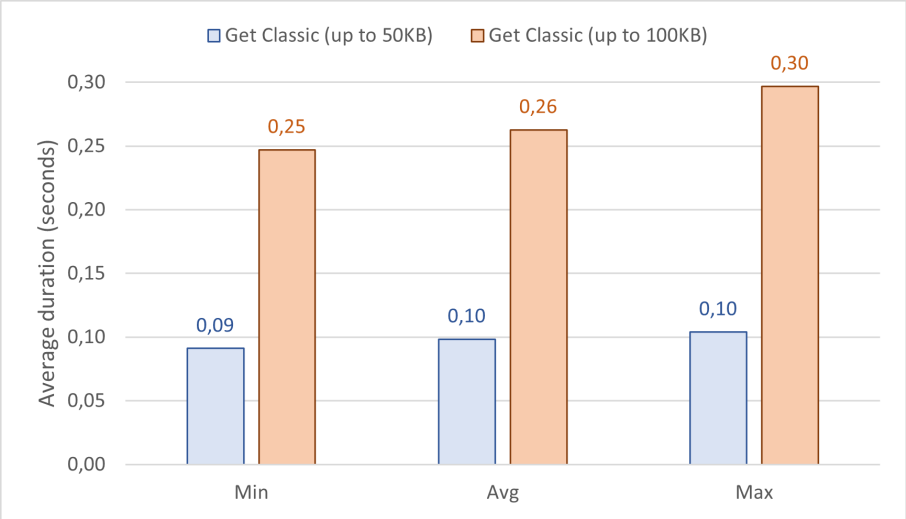


Figure 6.5: Duration of read requests to retrieve the current recorded data of a specific classic car, in seconds.

The requests to list the classic cars owned by a particular user or authorized to be modified or certified by a workshop or certification authority, respectively, are also important query functionalities to our system. The duration of these requests is illustrated in Figure 6.6, showing an average duration of 0.13 seconds for approximately 30KB of data to be retrieved, irrespective of the user's role in making the requests. As anticipated, when the data retrieval size increases (up to 100KB), the duration of the requests also increases to an average of 0.31 seconds, again independent of the user's role.

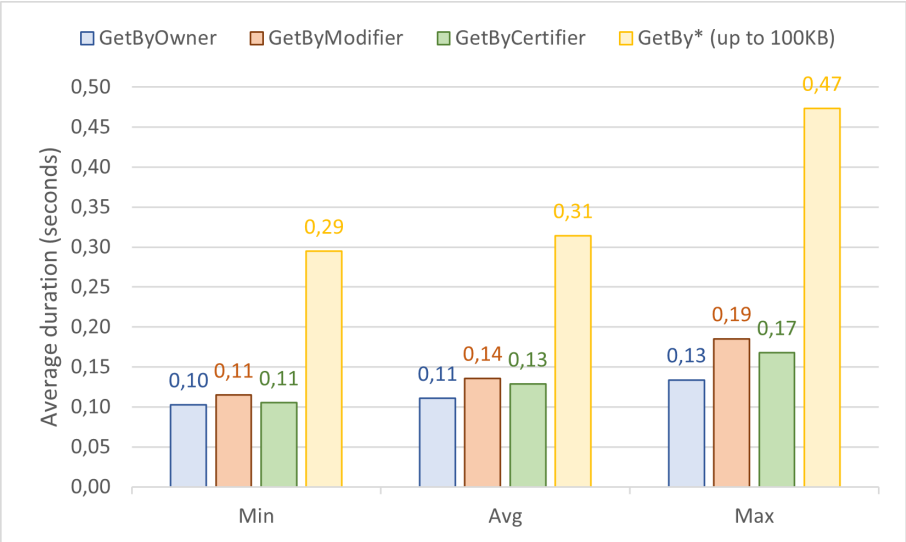


Figure 6.6: Duration of read requests to list the owned or authorized classic cars, in seconds.

The most time-intensive request within our system is retrieving the chronological history of all updates that lead to the current state of the classic car, which is the History of the

Vehicle Card and corresponding transactions and timestamps associated with each version. The results of this request are detailed in Figure 6.7. We subjected this request to extensive evaluation, considering varying amounts of data retrieval to ensure a comprehensive performance assessment.

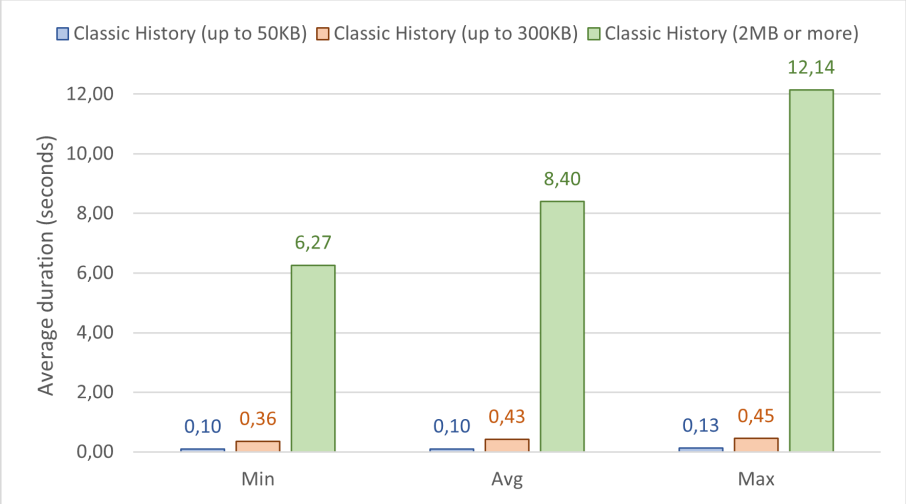


Figure 6.7: Duration of read requests to retrieve the chronological history of a classic car, in seconds.

For smaller datasets (up to 50KB) representing around 10 different versions of the classic, the average request time was 0.10 seconds. However, this duration noticeably escalates with an increasing number of versions to be retrieved, consequently expanding the data volume. Retrieving around 30 different versions (equivalent to up to 300KB) resulted in an average request time of 0.43 seconds. More significantly, when retrieving more than 100 versions (equivalent to at least 2MB of data), the average request time surged to 8.4 seconds, with a maximum time exceeding 12 seconds.

While not explicitly displayed, a similar pattern is observed when executing the request to retrieve the history of granted and revoked access permissions for a specific car. However, this request typically entails much less data to be retrieved. The increased time consumption in these requests stems from the fact that they are the only read operations actually retrieving data from the blockchain to obtain past values of the specific asset, rather than only querying the state database. Such more time-intensive executions are certainly noticeable to end-users.

Finally, it is also worth mentioning additional read requests that, while not complex in terms of logic, bear significance due to the amount of data to be retrieved being typically constant. For instance, retrieving information about a specific transaction based on its transaction ID or returning an expected error, such as attempting to access unauthorized vehicle data or querying details of an unregistered classic car. In these scenarios, the average duration for a request resulting in an expected error (regardless of the error type) is 0.12 seconds, which is a pretty acceptable response time to inform the end-users of an error occurrence effectively. Similarly, the duration for retrieving transaction information

stands at 0.15 seconds, reflecting the prompt response for accessing specific transaction details.

It is also important to mention that our implementation can also handle peak loads of requests due to its concurrency support. This is accomplished by instantiating each user's dedicated gateway and network instance. Our application can support similar TPS rate as the ones described in the previous subsection, with variations mainly observed in the duration and latency of available functionalities. Notably, some overhead is introduced due to the connection between the Hyperledger SDK used in our Fabric application and the actual Fabric network.

To conclude this performance evaluation, it is discernible that the request duration times for the functionalities within our solution are acceptable and reasonable within the context of blockchain-based systems and in the context of web applications for the classic car ecosystem. These durations offer users tolerable load times, ensuring a responsive experience. Upon analyzing all available requests, the average duration was calculated, obtaining an average of 2.5 seconds for write operations and 0.75 seconds for read functions, regardless of the data size being uploaded or retrieved. These duration values fall well within the acceptable threshold, remaining under the commonly tolerated 3-second waiting time for end-users in web systems, as proposed in [52]. In *ClassicsChain*, under normal blockchain and application conditions, end-users experience minimal disturbances or insignificant delays, especially in read-only operations, which represent the most frequently utilized functionalities.

6.3 Relevance and Usability Evaluation

To validate the relevance and usability of our solution, evaluating the user's interaction with the system, we employed an interview approach [53]. This interview, structured with a survey-based methodology, was divided into three stages, employing a questionnaire with distinct sets of questions and statements.

The first set of questions in the survey focused on the relevance of features implemented in *ClassicsChain*, presented before explaining our solution or without mentioning it explicitly. Following a presentation of our project, the second set of questions was introduced to evaluate the usability and user-friendliness of the web platform's GUI. The final set of questions aimed to capture participants' opinions, satisfaction levels, and suggestions regarding the developed solution, platform, and respective features.

To execute this validation correctly, it was important to thoughtfully select and invite participants who genuinely held the potential for interest in our solution. We sought individuals who could be interested in our platform and provide valuable and constructive feedback. By including diverse participants from various perspectives and environments within the classic car ecosystem, we aimed to gather insights and alternative opinions that could contribute to refining our solution. Moreover, these participants needed to understand the area, enabling them to provide impartial feedback that could thoroughly

evaluate the relevance and suitability of our work, pointing out strengths and weaknesses without hesitation. For the same reasons, it was also important that the participants had limited prior knowledge of our solution. Only a few of the selected individuals had some, yet limited knowledge of the overall digital transformation project at the Raimundo Branco workshop.

In light of the aforementioned considerations, five participants were selected: one classic car owner, an automotive industry manager with extensive experience, the director of ACP Clássicos, a division within Automóvel Club de Portugal (ACP) that certifies vehicles of historical interest, attesting to their quality, and both a business sales manager and commercial director from BASF, focusing on the BASF's Glasurit brand specialized in providing high-quality automotive coatings, including paints, primers, clear coats, and related products, used for refinishing classic car vehicles.

As mentioned, this validation process was divided into three distinct stages. The first stage consisted of the relevance evaluation, presenting the first set of questions of the questionnaire regarding features implemented in *ClassicsChain* deliberately avoiding any explicit mention of our solution and its development. For this purpose, the first set of statements within this stage was designed and divided into two subsets. The first aims to obtain the participants' opinions on how important they would consider the statements regarding their relevance on the market and the historical value of a classic car. The second subset pertains to the potential importance of having the most relevant features of *ClassicsChain* available in an online platform to enhance the classic car sector, but again without mentioning *ClassicsChain* or specific development details. These statements must be classified within a 5-point response range, ranging from "Strongly Disagree" to "Strongly Agree". The statements for both subsets are as follows:

- **First subset**:
  - Detailed technical documentation (e.g. VIN, engine number and other characteristics, mechanical and electrical schemas).
  - Documented evidence of past restoration procedures.
  - Independent assessment certificates (e.g. quality and authenticity evaluations performed by accredited sector organizations, awards in Concours d'Elegance, etc...).
  - List of previous owners.
  - Other important documentation (e.g. photographs and videos of the vehicle, inspection reports, insurance records, warranties for specific parts, etc...).
- **Second subset**:
  - Guarantee that uploaded contents cannot be tampered with, are immutable, and are completely transparent.

- Classic cars' documentation traceability features (history of versions, updates, and who performed them).
- Secure access control with the ability to grant and revoke specific permissions to a specific vehicle (e.g. permission to view the documentation or permission to modify it and, ultimately, transfer of ownership).
- Access auditing features (ability to track which entities have been granted/revoked access to a specific vehicle and the corresponding timestamps for such granted/revoked access).

The second phase of this evaluation process consisted of explaining our solution, *ClassicsChain*. Participants were given an overview of our solution's goals, functionality, underlying technologies, and feature set. This presentation was accompanied by a live demonstration of the web platform, showing various scenarios and features as they would be experienced by users in different roles. Next, participants were allowed to try the platform for themselves. Unfortunately, due to time constraints, some participants were unable to explore the platform during the session.

The third and final stage of this evaluation focused on the actual usability assessment of the web platform. This stage involved presenting the survey's second and third sets of questions. The second set of questions specifically addressed usability and user-friendliness, employing the System Usability Scale (SUS) methodology [54]. The SUS is a straightforward yet reliable questionnaire-based tool designed to assess the perceived usability of a system. It covers various usability aspects, including system complexity, ease of use, and overall satisfaction. It consists of a ten-item questionnaire, alternating between positive and negative tone statements, which must be rated in a range of "Strongly Disagree" to "Strongly Agree". We opted to use the standard version of this technique, which includes the following statements:

- I think that I would like to use this system frequently.
- I found the system unnecessarily complex.
- I thought the system was easy to use.
- I think that I would need the support of a technical person to be able to use this system.
- I found the various functions in this system were well integrated.
- I thought there was too much inconsistency in this system.
- I would imagine that most people would learn to use this system very quickly.
- I found the system very cumbersome to use.
- I felt very confident using the system.

- I needed to learn a lot of things before I could get going with this system.

The final set of questions of the validation process was created to gather the participants' opinions on the platform's concept, the overall solution, their satisfaction with it, and suggestions for the implemented features. This set is composed of six statements, each to be rated on a scale ranging from "Strongly Disagree" to "Strongly Agree," similar to the previous ones, along with two final open-ended questions. The six statements are as follows:

- The information presented in the Vehicle Card is comprehensible.
- The information presented in the Vehicle Card is well structured.
- I find it important to have all this information, details, and documentation stored and accessible through this online system.
- I find it important to have an access control mechanism to grant/revoke permissions to other users or entities to view or modify my Vehicle Card.
- I plan to use this online system to enhance the documentation of the restoration and preservation processes of my classic cars.
- I would recommend the use of this online system to a friend/acquaintance who is also interested in the classic automobile context, e.g., owning a classic vehicle or looking to acquire one.

The last two open-answer questions are:

- What other information, if any, would you like to appear in the Vehicle Card?
- Do you have any other suggestions or improvements to the system?

For a complete understanding of the questionnaire, including its respective divisions, questions, and flow, please refer to the Appendix B.

6.3.1 Results

From the interviews and initial feedback of the live demonstration and questionnaire completion, it became evident that the platform garnered interest from the different participants, each representing distinct stakeholder perspectives within this context. The diverse range of perspectives offered valuable insights and ideas about the platform's future development. In particular, one noteworthy suggestion came from a participant who raised the question of whether owners and other authorized users could modify information about their classic cars. While such modification is indeed possible, it does not extend to all aspects, contrary to the participant's initial assumption. The functionality allows owners to modify only the basic technical details of the car and other authorized

Table 6.1: Responses to the questionnaire regarding the significance of the implemented features in this work.

Statement	**Strongly Disagree**				**Strongly Agree**
Detailed technical documentation.	0	0	0	1	4
Documented evidence of past restoration procedures.	0	0	0	0	5
Independent assessment certificates.	0	0	0	1	4
List of previous owners.	0	0	1	1	3
Other important documentation.	0	0	0	1	4
Guarantee that uploaded contents are not possible to be tampered with, immutable, and completely transparent.	0	0	0	0	5
Classic cars' documentation traceability features (history of versions, updates, and who performed them).	0	0	0	0	5
Secure access control with the ability to grant and revoke specific permissions to a specific vehicle.	0	0	0	0	5
Access auditing features.	0	0	0	0	5

users are only allowed to modify the restoration procedures they initially registered. However, other users with view access to a classic car can verify the chronological history of versions of the *Vehicle Card* to check for these types of updates, and the classic car owners are informed of such updates via email (the permissions necessary for each available functionality are explained in the functions' descriptions in subsubsection 5.1.2.2). This ensures that *Vehicle Cards* and our solution remain transparent to all authorized users.

The responses to the statements in the initial set of questions are presented in Table 6.1. This set was designed to evaluate the relevance and initial impact of some of the most crucial features implemented on *ClassicsChain*, without explicitly mentioning their development or our solution. The answers substantiate the necessity of our work, emphasizing the potential upgrades of existing solutions within the automotive context by introducing these features, proving that these are important and valuable for various stakeholders within this sector. Furthermore, it reinforces that the functionalities in our solution are interesting and captivating, appealing even to individuals without extensive knowledge of web technologies.

Regarding the SUS section of the questionnaire to assess the usability and user experience of the *ClassicsChain* GUI, the results of the questions can be consolidated into a total score. As each of the ten questions has 5 possible answers, these can be scored ranging from 0.0 to 10.0 with a 2.5 interval, resulting in a possible score for each question of 0.0, 2.5, 5.0, 7.5, or 10.0 depending on the tone of the question being positive or negative which alters the order of this sequence. By summing the values of all questions, a final score ranging from a minimum of 0.0 to a maximum of 100.0 is calculated. This cumulative

Table 6.2: SUS scoring conventions.

SUS Score range	Grade	Percentile range
84.1 - 100	A+	96 - 100
80.8 - 84.0	A	90 - 95
78.9 - 80.7	A-	85 - 89
77.2 - 78.8	B+	80 - 84
74.1 - 77.1	B	70 - 79
72.6 - 74.0	B-	65 - 69
71.1 - 72.5	C+	60 - 64
65.0 - 71.0	C	41 - 59
62.7 - 64.9	C-	35 - 40
51.7 - 62.6	D	15 - 34
0.0 - 51.6	F	0 - 14

value can then be better converted into a grade, ranging from A+ to F, or represented as a percentile [54]. Table 6.2 depicts the scoring conventions for a SUS score.

The participants' final scores in the SUS section of the questionnaire are presented in Figure 6.8. The average score yields a final result of 81, corresponding to a grade of A when converted. This score is considered above average for user experience and usability, surpassing the score of 80, as suggested by J. R. Lewis [54].

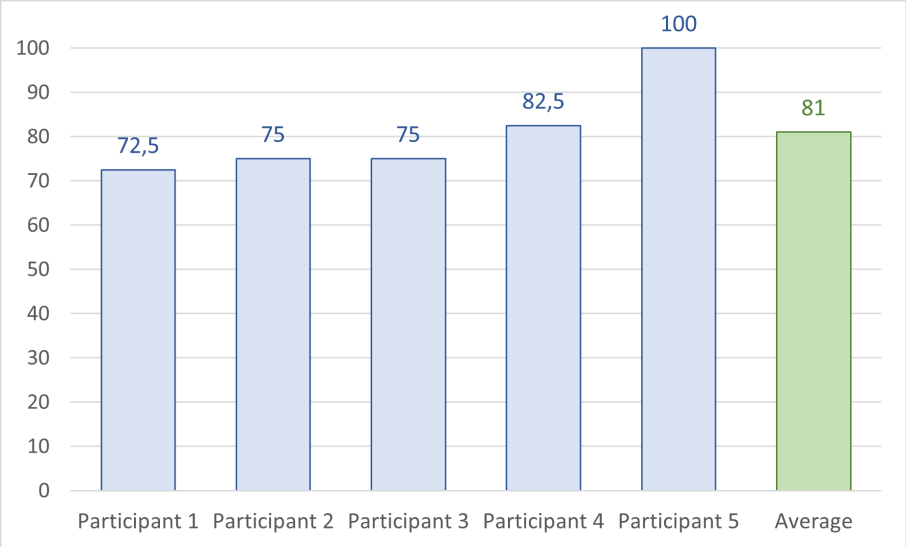


Figure 6.8: SUS results for the *ClassicsChain* web platform.

The responses to the final set of questions are displayed in Table 6.3. These further emphasize the necessity for the work undertaken in this research and the relevance of *ClassicsChain* and its distinctive features. Particularly noteworthy are the positive responses indicating participants' intentions to utilize the platform for future restoration and preservation procedures and their willingness to recommend it to other enthusiasts and entities involved in the classic automobile context.

From the last two open-answer questions, we received only one response to the first

Table 6.3: Responses to the questionnaire regarding the final opinions of *ClassicsChain*.

Statement	**Strongly Disagree**				**Strongly Agree**
The information presented in the Vehicle Card is comprehensible.	0	0	0	1	4
The information presented in the Vehicle Card is well structured.	0	0	0	1	4
I find it important to have all this information, details, and documentation stored and accessible through this online system.	0	0	0	0	5
I find it important to have an access control mechanism to grant/revoke permissions to other users or entities to view or modify my Vehicle Card.	0	0	0	1	4
I plan to use this online system to enhance the documentation of my classic cars' restoration and preservation processes.	0	0	0	1	4
I would recommend the use of this online system to a friend/acquaintance who is also interested in the classic automobile context.	0	0	0	0	5

question. This response reiterated the suggestion mentioned at the beginning of this subsection about authorized individuals being able to modify certain details on a *Vehicle Card* and other users would need to fact-check it later. This concern was already addressed (please refer to the beginning of subsection 6.3.1). Nevertheless, it offered valuable and prompt input that we have considered and is actively being elaborated for future work, as it will be explained in section 7.2.

To conclude the result analysis of the relevance and usability of our work, the feedback received during interviews and the corresponding results of the questionnaire across all its sections were considerably positive. There were no negative responses or serious concerns regarding the effectiveness of our solution. The feedback gathered from a diverse group of participants ranging from classic car owners to workers in classic car workshops and representatives of classic car certification entities, indicates that *ClassicsChain* is a valuable and significant contribution to the classic automobile sector. Some participants even expressed that this kind of research and its respective solution represents a substantial improvement over existing practices in the industry, even when comparing offerings from both national and international entities.

6.3.2 Validation Threats

Upon completion of the relevance and usability assessment, it is important to clarify the validation threats identified. One validation threat arises from the presence of a moderator during the interviews and questionnaire sessions. This factor introduces a potential bias, wherein participants may alter their responses to either appease or displease the moderator

and, consequently, the research.

Moreover, the participant sample for this assessment was limited to only five individuals, a constraint imposed by time and the availability of suitable participants. This limitation raises concerns about the precision of the results. It would be advantageous to replicate the experiments with a more extensive participant group. This improvement is anticipated as part of the continuation of this work and overall project.

Additionally, despite being unrelated to our work or the solution developed, some participants in this assessment had prior exposure to the workshop where the solution was developed. This may introduce a positive bias, potentially influencing responses to maintain a favorable relationship.

Specific to the validation of the *ClassicsChain* GUI, one identified threat was that some participants could not try and explore the platform themselves. While all participants observed the platform in operation, mere observation might not suffice to evaluate certain usability aspects comprehensively. The absence of hands-on experience could have influenced the outcomes, potentially yielding results that are either more favorable or less so than they would have been with direct interaction. Furthermore, the participants who utilized the interface did so in the same controlled environment of the interview, also with the presence of the mediator. This can impact the authenticity of the results as participants might interact with the system differently than they normally would in their natural environment.

6.4 Communication

As explained in section 1.6, the dissemination of the knowledge gathered, findings, and corresponding results, is a crucial step of the DSR methodology. This step not only validates the significance of the research and respective solution but also fosters collaboration between academic and scientific domains. For this reason, in the course of this research, we submitted a paper to the International Conference on Industry 4.0 and Smart Manufacturing (ISM 2023) conference. This paper [55] was written during an intermediate phase of the development of our work. It describes our solution's overall structure and implementation and some of the tests, specifically performance tests and evaluation of the components developed so far. Additionally, the paper covers background topics and describes the literature review conducted. The submitted paper was peer-reviewed and accepted for presentation at the ISM 2023 conference, held November 22-24, 2023, with proceedings expected to be published by Elsevier in early 2024.

A more comprehensive publication, including a more detailed explanation of the validation step and corresponding outcomes, is online as a preprint at the ArXiV server made available by Cornell University, in the Computer Science / Computers and Society section, intended to be submitted as a journal paper soon.

7

Conclusions

This chapter presents the conclusions of this work, outlining the involved work and the resulting solution as well as the contributions reached in section 7.1. Additionally, in section 7.2, we describe potential future work proposals to refine our proposed solution further.

7.1 Conclusions

This research work aimed to create and validate a solution to record and maintain all relevant information about classic cars and all associated documentation, especially for the restoration processes that these cars typically undergo to preserve their authenticity and further classify them as masterpieces. Given the meticulous nature of classic car enthusiasts and the industry's emphasis on maintaining authenticity, every detail plays a crucial role in upholding the credibility of owners, workshops, and the sector as a whole.

All the information proposed for implementation in our solution is intended to be trustworthy, immutable, transparent, and traceable. Furthermore, it should facilitate the exchange of information among various entities, fostering collaboration within the industry, even among unrelated stakeholders. Moreover, our solution intends to promote best practices within this specific sector, meticulously recording all important information and transparently presenting it to all authorized entities.

To execute this work, we adopted a Design Science Research (DSR) methodology. This involved a systematic assessment of the problem and final goals, exploration of cutting-edge technologies applicable to this domain, and a literature review (Rapid Review (RR)) to examine and evaluate existing solutions with objectives similar to ours, drawing insights from both scientific literature and industry innovations. Notably, we observed a lack of studies and projects addressing these objectives, particularly when tailored to the unique demands of the classic car industry and its detailing requirements in restoration for authenticity's sake.

The developed solution, *ClassicsChain*, consists of a blockchain-based application specially tailored to record every relevant information about classic cars. This encompasses technical specifications, evidence of restoration procedures, certificates, details of current and past owners, and any other crucial documentation necessary for achieving the highest degree of detail possible for a classic vehicle. *ClassicsChain* leverages a Hyperledger Fabric blockchain network, with the functions available on our smart contracts accessible through a Fabric application functioning as our server application. This application was also integrated with decentralized media storage to store the media evidence associated with vintage cars.

Among the most important and unique features of *ClassisChain* is its inherent immutability and complete transparency once information is stored. It provides a chronological history of the data, specifying details of the restoration procedures, including the type of procedure, the associated media, the registration time, and the responsible entity. An access control mechanism was also implemented, allowing users to grant or revoke specific permissions on the owned vehicles' data, while also including access auditing features. All these features are available through the developed GUI or through the public API, for any unrelated entity that wants to integrate these unique features with their systems. Continuous monitoring and feedback from experts in the field, classic workshop workers, and past project participants have shaped and refined these features.

Furthermore, *ClassicsChain* functionalities were integrated with the platform *Charter of Turin Monitor*, currently in use at the Raimundo Branco workshop for tracking restoration progress. Some other modifications were also performed to the *Charter of Turin Monitor* for an ongoing enhancement of this solution and continuous improvement of the overall project.

Our work and the resulting solution were also rigorously validated. The performance assessment met the standards expected for this type of system and both the relevance and usability evaluations garnered significant positive feedback from all the participants. It was even noticed by some of the participants that this work represents a substantial improvement over existing solutions in the field.

Moreover, this study hopes to contribute to the enhancement and credibility of the digital transformation of the specific domain of classic automotive, especially when related to the restoration procedures executed for these vehicles, reinforcing best practices and improving the efficiency of information flow transparently. Ultimately, it can also help to increase the value of vintage cars and the trust in this industry, as the registered vehicles are associated with a meticulous and trustworthy record of all respective information.

This research also aspires to be a valuable contribution to other endeavors and projects sharing similar goals within the realm of blockchain-based systems and the digital transformation of the classic car ecosystem.

7.2 Future Work

In terms of future developments, there is potential for enhancements and the introduction of new features and ideas related to both the ongoing progression of this work and the further refinement of our solution. As an important consideration, it is worth emphasizing that the credibility of our solution relies on the accurate registration of information by the users. If users choose to register incorrect or false information, we lack a mechanism to prevent that, neither was the focus of this research.

Nevertheless, it is relevant to acknowledge some of the potential directions for future work, namely to integrate a mechanism to automatically detect data quality issues and misinformation using Generative Artificial Intelligence and Machine Learning algorithms [56, 57]. Such an approach would aid in preventing the incorrect registration of user-provided information, enhancing data accuracy, and ensuring a more reliable input of information.

To ensure compliance with data privacy regulations like the General Data Protection Regulation (GDPR), it is advisable to implement best practices for confidentiality. Instead of using email addresses as unique identifiers for users, a more suitable approach would involve generating unique random identifiers. This method eliminates the need to store any personal data directly on the blockchain, thereby enhancing data privacy and minimizing potential risks associated with storing sensitive information.

Another significant prospect for future work would be to explore the transition of our blockchain network infrastructure to a network with nodes completely managed by the represented organizations. This shift would grant the respective organizations or entities full control and management of these nodes, as opposed to depending on the services offered by a cloud service provider. However, it is essential to recognize that this is an intricate task that demands considerable attention, dedicated resources, and support from the involved organizations.

Applying the same rationale, a private IPFS network completely managed by the participating organizations would further enhance data privacy for uploaded media files. However, undertaking this is yet another highly intricate task that requires substantial resources and entails significant associated costs.

Concerning the information being stored for classic cars on our solution, it would be valuable to incorporate additional fields for specific details. This could include particulars like events or Concours d'Elegance in which the classic car participated, along with its competition history. Moreover, introducing fields to document components or parts added or removed during a specific restoration process would add detail to the information. While this data can be captured through restoration procedure descriptions or additional documentation, explicitly specifying these occurrences in dedicated fields would provide clearer and more direct clarification.

Considering a suggestion made by one of the participants during the interviews for our validation endeavors (outlined in the final considerations in subsection 6.3.1), aimed at

facilitating the process of fact-checking some information recorded on a vehicle, it would be highly beneficial to introduce a feature to the chronological history of the *Vehicle Cards* to display some type of warning message when an update is noticed from one version of the *Vehicle Card* to the subsequent one. For instance, if a user modifies the country of registration or any other pivotal technical detail, or updates the information related to a previously registered restoration procedure, a warning message would be returned, indicating that there was an update on already stored data between the two versions. Such functionality would significantly simplify and streamline the fact-checking process for classic car owners, potential buyers, and certification authorities alike.

Another highly valuable improvement of our work would be the mediated identity verification upon user registration, as it was previously outlined during the description of the user registration and authentication of *ClassicsChain* (refer to subsubsection 5.1.3.2). The envisioned identity verification process would involve validating each user's identity with a document or certificate that uniquely identifies them, such as an identity card, passport, or driver's license. This is a practice commonly employed in banking and government platforms. This feature would present immense potential for elevating our authentication mechanisms, introducing means to ensure the authenticity of each user's identity, and guaranteeing a higher level of trust in the stored data. Moreover, this verification process enhances traceability, as the data associated with a user is associated with a tangible individual with specific details. In the classic car sector, where reputation is highly valuable, this approach incentivizes accuracy in data entry, further strengthening the integrity and trustworthiness of the information. The result of this feature would be a system with enhanced security, credibility, and traceability, in line with the best practices expected in the classic car ecosystem. This enhancement also opens up the possibility for classic car owners to register their vehicles directly in the system, eliminating the need to rely on a certified restoration shop or certification authority.

Other considerable improvements for future developments include fostering greater interaction among user-related functionalities. This encompasses features like enabling users to grant view permissions with customized duration times and facilitating users to request permissions from one another. Additionally, the implementation of communication channels allowing all users to exchange text messages would be beneficial. This could be particularly useful in scenarios where a classic car owner needs to communicate with the classic workshop where their vehicle is undergoing restoration or when an owner is communicating with a potential buyer.

Several minor improvements can also be made to the developed Graphical User Interface (GUI) to achieve an even more favorable evaluation, including more visually appealing and user-friendly components, particularly on critical pages like the Vehicle Card and the History of the Vehicle Card.

Lastly, considering the limited number of participants (5) in both the relevance and usability evaluations, a more extensive assessment with participants immersed in the classic car environment is essential.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. i).

[2] K. Gibbins. *Charter of Turin Handbook*. Tech. rep. Version 2. Accessed on 2023-06-18. Fédération Internationale des Véhicules Anciens (FIVA), 2018-11. URL: https://fiva.org/download/turin-charter-handbook-updates-2019-english-version/ (cit. on pp. 1–3, 9).

[3] D. Lívio. "Process-based Monitoring in Industrial Context - The Case of Classic Cars Restoration". MA thesis. Monte da Caparica, Portugal: Nova School of Science and Technology, 2021-02. URL: http://hdl.handle.net/10362/141079 (cit. on pp. 3, 6, 9, 12, 13, 49).

[4] D. Pereira. "An automated system for monitoring and control classic cars' restorations: An IoT-based approach". MA thesis. Monte da Caparica, Portugal: Nova School of Science and Technology, 2021-11. URL: http://hdl.handle.net/10362/138798 (cit. on pp. 3, 6, 9, 10, 12, 13, 49).

[5] P. Moura. "Developing and validating process-aware GUIs in an industrial setting". MA thesis. Monte da Caparica, Portugal: Nova School of Science and Technology, 2022-11. URL: http://hdl.handle.net/10362/151100 (cit. on pp. 3, 6, 9–13, 47–49, 51, 85, 86).

[6] R. Gomes. "Combining Different Data Sources For Industrial Process Monitoring". To be published. MA thesis. Monte da Caparica, Portugal: Nova School of Science and Technology, 2022-04 (cit. on pp. 3, 6, 9–13, 49).

[7] I. Atoum et al. "Challenges of Software Requirements Quality Assurance and Validation: A Systematic Literature Review". In: *IEEE Access* 9 (2021), pp. 137613–137634. DOI: 10.1109/ACCESS.2021.3117989 (cit. on p. 4).

[8] K. Peffers et al. "A design science research methodology for information systems research". In: *Journal of Management Information Systems* 24 (2007-01), pp. 45–77 (cit. on p. 5).

[9] V. K. Vaishnavi and W. L. Kuechler. "Design Science Research in Information Systems". In: *Ais* (2004), pp. 1–45. DOI: 10.1007/978-1-4419-5653-8 (cit. on p. 5).

[10] I. Bashir. *Mastering Blockchain: A deep dive into distributed ledgers, consensus protocols, smart contracts, DApps, cryptocurrencies, Ethereum, and more*. Packt Publishing Ltd, 2020 (cit. on p. 12).

[11] S. Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: *SSRN Electronic Journal* (2008). DOI: 10.2139/ssrn.3977007 (cit. on pp. 12, 15).

[12] D. R. Stinson. *Cryptography: theory and practice*. Chapman and Hall/CRC, 2005 (cit. on p. 12).

[13] K. Christidis and M. Devetsikiotis. "Blockchains and Smart Contracts for the Internet of Things". In: *IEEE Access* 4 (2016), pp. 2292–2303. ISSN: 2169-3536. DOI: 10.1109/access.2016.2566339 (cit. on pp. 13–15, 18, 20).

[14] B. Tavares et al. "A survey of blockchain frameworks and applications". In: *International Conference on Soft Computing and Pattern Recognition* (2018-10). DOI: 10.13140/RG.2.2.34042.95685 (cit. on p. 14).

[15] Z. Zheng et al. "Blockchain challenges and opportunities: a survey". In: *International Journal of Web and Grid Services* 14.4 (2018), p. 352. ISSN: 1741-1114. DOI: 10.1504/ijwgs.2018.095647 (cit. on pp. 14, 17).

[16] B. Shrimali and H. B. Patel. "Blockchain state-of-the-art: architecture, use cases, consensus, challenges and opportunities". In: *Journal of King Saud University - Computer and Information Sciences* 34 (9 2022-10), pp. 6793–6807. ISSN: 1319-1578. DOI: 10.1016/J.JKSUCI.2021.08.005 (cit. on pp. 14–17, 19).

[17] V. Buterin. *A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM*. Accessed on 2023-04-23. 2015 (cit. on pp. 15, 22).

[18] V. Buterin. *On Public and Private Blockchains | Ethereum Foundation Blog*. Accessed on 2023-04-23. URL: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains (cit. on p. 16).

[19] "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains". In: *Proceedings of the Thirteenth EuroSys Conference*. EuroSys '18. ACM, 2018-04. DOI: 10.1145/3190508.3190538 (cit. on pp. 16, 19, 22–24).

[20] C. Dwork and M. Naor. "Pricing via Processing or Combatting Junk Mail". In: *Advances in Cryptology — CRYPTO' 92*. Ed. by E. F. Brickell. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 139–147. ISBN: 978-3-540-48071-6 (cit. on p. 18).

[21] J. Sedlmeir et al. "The Energy Consumption of Blockchain Technology: Beyond Myth". In: *Business and Information Systems Engineering* 62 (6 2020-12), pp. 599–608. ISSN: 18670202. DOI: 10.1007/S12599-020-00656-X (cit. on p. 19).

[22] *The Merge | ethereum.org*. Accessed on 2023-04-23. URL: https://ethereum.org/en/upgrades/merge/ (cit. on p. 19).

[23] M. Castro and B. Liskov. "Practical Byzantine Fault Tolerance". In: *ACM Transactions on Computer Systems* 20.4 (2002-11), pp. 398–461. ISSN: 1557-7333. DOI: 10.1145/571637.571640 (cit. on p. 19).

[24] N. Szabo. "Formalizing and Securing Relationships on Public Networks". In: *First Monday* 2.9 (1997-09). ISSN: 1396-0466. DOI: 10.5210/fm.v2i9.548 (cit. on p. 20).

[25] *A blockchain platform for the enterprise*. Accessed on 2023-04-23. URL: https://hyperledger-fabric.readthedocs.io/ (cit. on pp. 22–24, 55–59).

[26] T. S. L. Nguyen et al. "Impact of network delays on Hyperledger Fabric". In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019-04. DOI: 10.1109/infcomw.2019.8845168 (cit. on pp. 23, 24).

[27] *The Ordering Service | Raft*. Accessed on 2023-07-26. URL: https://hyperledger-fabric.readthedocs.io/en/latest/orderer/ordering_service.html#raft (cit. on p. 24).

[28] A. C. Tricco et al. "A scoping review of rapid review methods". In: *BMC Medicine* 13.1 (2015-09). ISSN: 1741-7015. DOI: 10.1186/s12916-015-0465-6 (cit. on p. 27).

[29] V. Garousi, M. Felderer, and M. V. Mäntylä. "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering". In: *Information and Software Technology* 106 (2019-02), pp. 101–121. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2018.09.006 (cit. on pp. 27, 35).

[30] B. Kitchenham and S. Charters. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Tech. rep. EBSE-2007-01. School of Computer Science and Mathematics, Keele University, 2007-07 (cit. on pp. 27, 30).

[31] C. Wohlin et al. "Experimentation in software engineering". In: *Experimentation in Software Engineering* 9783642290442 (2012-07), pp. 1–236. DOI: 10.1007/978-3-642-29044-2 (cit. on p. 28).

[32] S. Stahlschmidt and D. Stephen. *Comparison of Web of Science, SCOPUS and Dimensions databases*. Tech. rep. KB Forschungspoolprojekt 2020. Hannover, Germany: German Centre for Higher Education Research and Science Studies (DZHW), 2020-10, pp. 1–37 (cit. on p. 30).

[33] C. Wohlin. "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering". In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. EASE '14. ACM, 2014-05. DOI: 10.1145/2601248.2601268 (cit. on p. 31).

[34] B. Zhong et al. "A Blockchain-Based Life-Cycle Environmental Management Framework for Hospitals in the COVID-19 Context". In: *Engineering* 20 (2023-01), pp. 208–221. ISSN: 2095-8099. DOI: 10.1016/j.eng.2022.06.024 (cit. on p. 33).

[35] L. Ismail, H. Materwala, and S. Zeadally. "Lightweight Blockchain for Healthcare". In: *IEEE Access* 7 (2019), pp. 149935–149951. ISSN: 2169-3536. DOI: 10.1109/access.2019.2947613 (cit. on p. 34).

[36] X. Zhang et al. "Blockchain-Based Safety Management System for the Grain Supply Chain". In: *IEEE Access* 8 (2020), pp. 36398–36410. ISSN: 2169-3536. DOI: 10.1109/access.2020.2975415 (cit. on p. 34).

[37] T. Ferdousi, D. Gruenbacher, and C. M. Scoglio. "A Permissioned Distributed Ledger for the US Beef Cattle Supply Chain". In: *IEEE Access* 8 (2020), pp. 154833–154847. ISSN: 2169-3536. DOI: 10.1109/access.2020.3019000 (cit. on p. 34).

[38] Y.-T. Jiang and H.-M. Sun. "A Blockchain-Based Vehicle Condition Recording System for Second-Hand Vehicle Market". In: *Wireless Communications and Mobile Computing* 2021 (2021-05). Ed. by L. Cuthbert, pp. 1–10. ISSN: 1530-8669. DOI: 10.1155/2021/6623251 (cit. on pp. 37–39).

[39] R. R. Rezende et al. "Using blockchains to protect critical infrastructures: a comparison between Ethereum and Hyperledger Fabric". In: *International Journal of Security and Networks* 17.2 (2022), p. 77. ISSN: 1747-8413. DOI: 10.1504/ijsn.2022.10047903 (cit. on p. 38).

[40] K. L. Brousmiche et al. "Digitizing, Securing and Sharing Vehicles Life-cycle over a Consortium Blockchain: Lessons Learned". In: *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2018-02. DOI: 10.1109/ntms.2018.8328733 (cit. on pp. 38, 39).

[41] K. Leo Brousmiche et al. "Hybrid Cryptographic Protocol for Secure Vehicle Data Sharing Over a Consortium Blockchain". In: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018-07. DOI: 10.1109/cybermatics_2018.2018.00223 (cit. on pp. 38, 39).

[42] K. Luchoomun, S. Pudaruth, and S. Kishnah. "Implementation of a Proof of Concept for a Blockchain-based Smart Contract for the Automotive Industry in Mauritius". In: *International Journal of Advanced Computer Science and Applications* 11.3 (2020). ISSN: 2158-107X. DOI: 10.14569/ijacsa.2020.0110309 (cit. on p. 39).

[43] G. Subramanian and A. S. Thampy. "Implementation of Hybrid Blockchain in a Pre-Owned Electric Vehicle Supply Chain". In: *IEEE Access* 9 (2021), pp. 82435–82454. ISSN: 2169-3536. DOI: 10.1109/access.2021.3084942 (cit. on p. 39).

[44] B. H. Cheng and J. M. Atlee. "Research Directions in Requirements Engineering". In: *Future of Software Engineering (FOSE '07)*. IEEE, 2007-05. DOI: 10.1109/fose.2007.17 (cit. on p. 41).

[45] A. Aurum and C. Wohlin. *Engineering and managing software requirements*. Vol. 1. Springer, 2005 (cit. on p. 41).

[46] I. Sommerville. *Software Engineering 10th Edition*. Pearson, 2015. ISBN: 0133943038 (cit. on p. 42).

[47] A. A. Monrat, O. Schelen, and K. Andersson. "Performance Evaluation of Permissioned Blockchain Platforms". In: *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*. IEEE, 2020-12. DOI: 10.1109/csde50874.2020.9411380 (cit. on p. 50).

[48] J. Polge, J. Robert, and Y. Le Traon. "Permissioned blockchain frameworks in the industry: A comparison". In: *ICT Express* 7.2 (2021-06), pp. 229–233. ISSN: 2405-9595. DOI: 10.1016/j.icte.2020.09.002 (cit. on p. 50).

[49] *Performance Considerations*. Accessed on 2023-10-19. URL: https://hyperledger-fabric.readthedocs.io/en/latest/performance.html (cit. on pp. 51, 58).

[50] T. Guggenberger et al. "An in-depth investigation of the performance characteristics of Hyperledger Fabric". In: *Computers & Industrial Engineering* 173 (2022-11), p. 108716. ISSN: 0360-8352. DOI: 10.1016/j.cie.2022.108716 (cit. on p. 58).

[51] N. Provos and D. Mazières. "A Future-Adaptable Password Scheme". In: *Proceedings of the FREENIX Track: USENIX Annual Technical Conference*. 1999 (cit. on p. 70).

[52] F. F.-H. Nah. "A study on tolerable waiting time: how long are Web users willing to wait?" In: *Behaviour & Information Technology* 23.3 (2004-05), pp. 153–163. ISSN: 1362-3001. DOI: 10.1080/01449290410001669914 (cit. on p. 99).

[53] I. Seidman. *Interviewing as qualitative research: A guide for researchers in education and the social sciences*. Teachers college press, 2006 (cit. on p. 99).

[54] J. R. Lewis. "The System Usability Scale: Past, Present, and Future". In: *International Journal of Human–Computer Interaction* 34.7 (2018-03), pp. 577–590. ISSN: 1532-7590. DOI: 10.1080/10447318.2018.1455307 (cit. on pp. 101, 104).

[55] J. Murta, V. Amaral, and F. B. e Abreu. "Trusted Evidences on the Digital Transformation of Classic Cars Restoration". In: *Proc. of 4th International Conference on Industry 4.0 and Smart Manufacturing*. Procedia Computer Science. DOI: to be assigned. Lisbon, PT, 2024 (cit. on p. 106).

[56] P. S. Dhoni. *Enhancing Data Quality through Generative AI: An Empirical Study with Data*. Tech. rep. 24470032. TechRxiv / Authorea, 2023. DOI: 10.36227/techrxiv.24470032.v1 (cit. on p. 109).

[57] L. Poon et al. “Unsupervised anomaly detection in data quality control”. In: *2021 IEEE International Conference on Big Data (Big Data)*. IEEE. 2021, pp. 2327–2336 (cit. on p. 109).

A

Web Application API

Next follows a list of the developed resource and endpoints along with some implementation considerations. When employing {param}, it indicates that the parameter is passed as a path parameter. Other specific parameters passed in the request's body will be described accordingly.

1. **/api/Users**

    - **POST** /api/Users - Enables the registration of a new user in the system by providing the email address and password in the request's body, along with other user information, such as the organization. An automated email confirming the registration is sent to the newly registered user.
    - **POST** /api/Users/Login - Enables user login, generating a JWT access token valid for five hours. This token is used for authentication in subsequent endpoints upon successful authentication.
    - **GET** /api/Users/GetAll - Returns information for all registered users with all respective information, such as email address, organization, and country of registration. This function is exclusive to logged-in users.
    - **GET** /api/Users/Get/{email} - Returns the information for a specific user identified by the provided email address as a path parameter. This function is exclusive to logged-in users.

2. **/api/Classics**

    - **POST** /api/Classics/Create - Enables the registration of new classic vehicles in the system, providing the fundamental details (defined in the *Classic* struct as depicted in Listing 5.1) and the owner's email in the request's body. An automated email confirming the registration of the classic vehicle is sent to the respective owner. Within the smart contract, the "CreateClassic" function is invoked and it is restricted to entities from the Workshop Org or Certification Org.

- **GET** /api/Classics/Get/{chassisNo} - Returns the information of a particular classic car identified by the chassis number provided as a path parameter. The returned information constitutes the *Vehicle Card.* Within the smart contract, one of the variations of the "ReadClassic" function is invoked and only users holding at least view permission for that particular vehicle can perform it.
- **PUT** /api/Classics/Update/{chassisNo} - Enables the updating of fundamental details for the classic car identified by the chassis number provided as a path parameter. Only the current owner of the vehicle can perform this function. An automated email confirming the update of these fundamental details is sent to the car owner.
- **PUT** /api/Classics/UpdateEmail/{chassisNo}/{newEmail} - Enables the transfer of ownership of a specific classic car identified by the chassis number provided as a path parameter to the user with the email address also passed as a path parameter. Within the smart contract, the "UpdateClassicEmail" function is invoked and only the current owner of the vehicle can perform this function. Automated confirmation emails are sent to both the former and new owners, acknowledging the successful transfer of ownership of the vehicle.
- **GET** /api/Classics/History/{chassisNo} - Returns a complete chronological history of all updates and transactions that led to the current state and details of the classic vehicle identified by the chassis number provided as a path parameter. It is retrieved the History of the *Vehicle Card*, along with the transaction and timestamp that originated that state of the *Vehicle Card*. Within the smart contract, the "GetClassicHistory" function is invoked and only users holding at least view permission to that particular vehicle can perform this function.
- **GET** /api/Classics/GetBy* - Returns the classic cars owned or authorized for the particular user making the request. The asterisk (*) serves as a wildcard for three different endpoints based on the user's role: "Owner," "Modifier," or "Certifier". Within the smart contract, one of the variations of the "QueryClassics" function is invoked.
- **PUT** /api/Classics/AddDocument/{chassisNo} - Enables the addition of documents (e.g., certificates of restoration or legal documents) to the *Vehicle Card* of a classic car identified by the chassis number provided as a path parameter. Within the smart contract, the "AddDocument" function is invoked and only users holding owner, modifier, or certifier permission levels for the particular vehicle can perform this operation.
- **PUT** /api/Classics/Certify/{chassisNo} - Enables the certification of a *Vehicle Card* and respective information for the classic car identified by the chassis number provided as a path parameter. Within the smart contract, the "MarkAsCertified" function is invoked and only entities with certifier access for that particular vehicle are authorized to perform this operation.

3. **/api/Restorations**

   - **POST** /api/Restorations/Create/{chassisNo} - Enables the creation of a restoration step to be added to the *Vehicle Card* of a classic car identified by the chassis number provided as a path parameter. The request's body includes the title, description, and optional media files as form-data for the specific restoration procedure. Upon successful completion, the unique identifier for the restoration procedure is returned as a response. Within the smart contract, the "CreateRestorationStep" function is invoked and only entities with modifier access for the particular vehicle and the vehicle's owner can perform this operation.
   - **GET** /api/Restorations/Get/{chassisNo}/{stepId} - Returns the details of a particular restoration step identified by the chassis number and step identifier provided as path parameters. Within the smart contract, the "GetStep" function is invoked and this function is authorized only to users with at least view permission for that particular vehicle.
   - **PUT** /api/Restorations/Update/{chassisNo} - Enables the update of the title and description of a particular restoration step identified by the chassis number provided in the path parameter and the step identifier passed in the request's body. Within the smart contract, the "UpdateStep" function is invoked and only the entity that originally registered the specified restoration step is authorized to perform this operation.
   - **PUT** /api/Restorations/Update/{chassisNo}/Photos - Enables the update of the media files of a particular restoration step identified by the chassis number provided as a path parameter and the step identifier passed in the request's body. Within the smart contract, the "UpdateStepPhotos" function is invoked and only the entity that originally registered the specified restoration step is authorized to perform this operation.
   - **PUT** /api/Restorations/UpdateAndPhotos/{chassisNo} - Enables the update of the title, description as well as the media files of a particular restoration step identified by the chassis number provided as a path parameter and the step identifier passed in the request's body. This endpoint combines the functionalities of the two previous endpoints into a single operation. Within the smart contract, the "UpdateStepAndPhotos" function is invoked and only the entity that originally registered the specified restoration step is authorized to perform this operation.

4. **/api/Access**

   - **GET** /api/Access/Get/{chassisNo} - Returns access information, including users who have been granted access permissions, along with their respective levels and timestamps of granting, for a specific classic car identified by the

chassis number provided as a path parameter. Within the smart contract, the "GetAccess" function is invoked and only the current owner of the vehicle can perform this function.

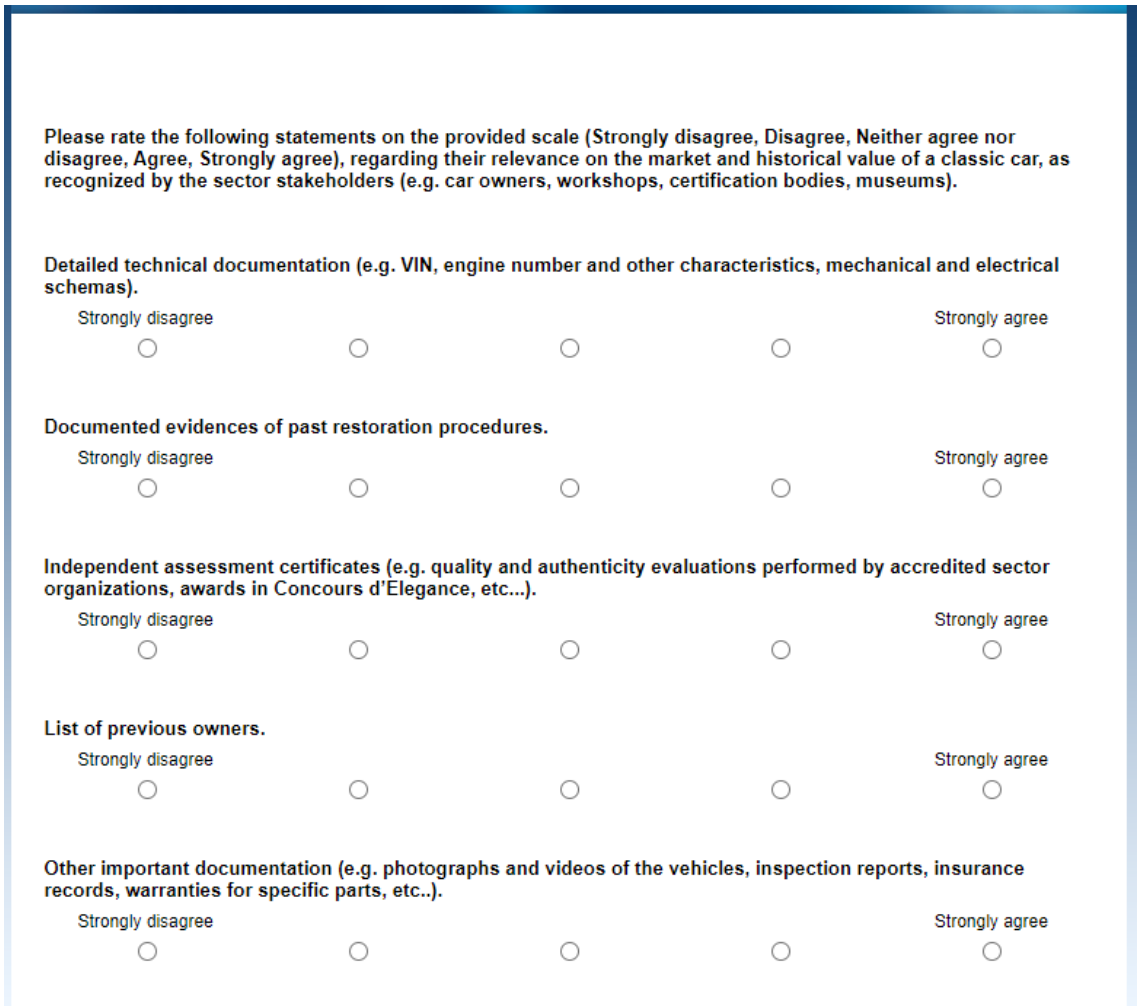
- **PUT** /api/Access/Give/{chassisNo} - Enables the granting of access permission to a particular user identified by the email address provided in the request's body. The permission level is also specified in the request's body and can be a string with the value "viewer," "modifier," or "certifier" for the specific classic car identified by the chassis number in the path parameter. Within the smart contract, the "GiveAccess" function is invoked and only the current owner of the vehicle can perform this function. An automated email is sent to the user who received permissions, informing details about the granted access level and the specific classic car associated with the permission.
- **PUT** /api/Access/Revoke/{chassisNo} - Similar to the previous endpoint, but instead of granting permission, it revokes it for a specific user identified by the email address provided in the request's body. The permission level is also specified in the request's body ("viewer," "modifier," or "certifier") for the classic car identified by the chassis number in the path parameter. Within the smart contract, the "RevokeAccess" function is invoked and only the current owner of the vehicle can perform this function.
- **GET** /api/Access/CheckLevel/{chassisNo} - Returns a string containing the value "owner," "viewer," "modifier," "certifier," or "unauthorized" indicating the access permission level held by the user invoking the request for the specific classic car identified by the chassis number in the path parameter. Within the smart contract, the "CheckUserAccess" function is invoked.
- **GET** /api/Access/History/{chassisNo} - Returns a complete chronological history of all updates and transactions that led to the current state of access permissions of the classic vehicle identified by the chassis number provided as a path parameter. The transaction identifier and timestamp that originated that state are also displayed. Within the smart contract, the "GetAccessHistory" function is invoked and only the current owner of the vehicle can perform this function.

5. **/api/Transaction**

- **GET** /api/Transaction/{chassisNo}/{txId} - Returns information about the particular transaction identified by the transaction identifier and the chassis number provided as path parameters. The information includes the transaction timestamp, user details of the transaction submitter, the function that initiated the transaction, along with its arguments, and the response generated by that function. Only users holding at least view permission to that particular vehicle can perform this function.

B

Questionnaire



Please rate the following statements on the provided scale (Strongly disagree, Disagree, Neither agree nor disagree, Agree, Strongly agree), regarding their relevance on the market and historical value of a classic car, as recognized by the sector stakeholders (e.g. car owners, workshops, certification bodies, museums).

Detailed technical documentation (e.g. VIN, engine number and other characteristics, mechanical and electrical schemas).

Strongly disagree ○ ○ ○ ○ ○ Strongly agree

Documented evidences of past restoration procedures.

Strongly disagree ○ ○ ○ ○ ○ Strongly agree

Independent assessment certificates (e.g. quality and authenticity evaluations performed by accredited sector organizations, awards in Concours d'Elegance, etc...).

Strongly disagree ○ ○ ○ ○ ○ Strongly agree

List of previous owners.

Strongly disagree ○ ○ ○ ○ ○ Strongly agree

Other important documentation (e.g. photographs and videos of the vehicles, inspection reports, insurance records, warranties for specific parts, etc..).

Strongly disagree ○ ○ ○ ○ ○ Strongly agree

Figure B.1: Questionnaire for the relevance of our solution. The first set of questions (1).

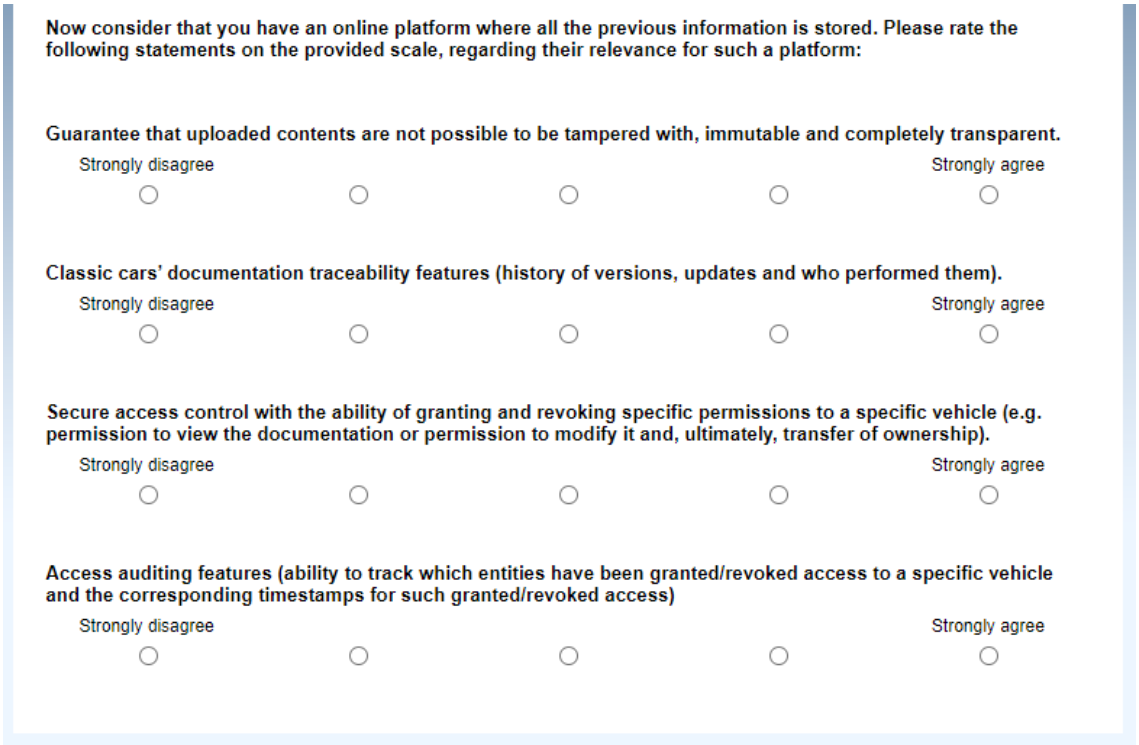


Figure B.2: Questionnaire for the relevance of our solution. The first set of questions (2).

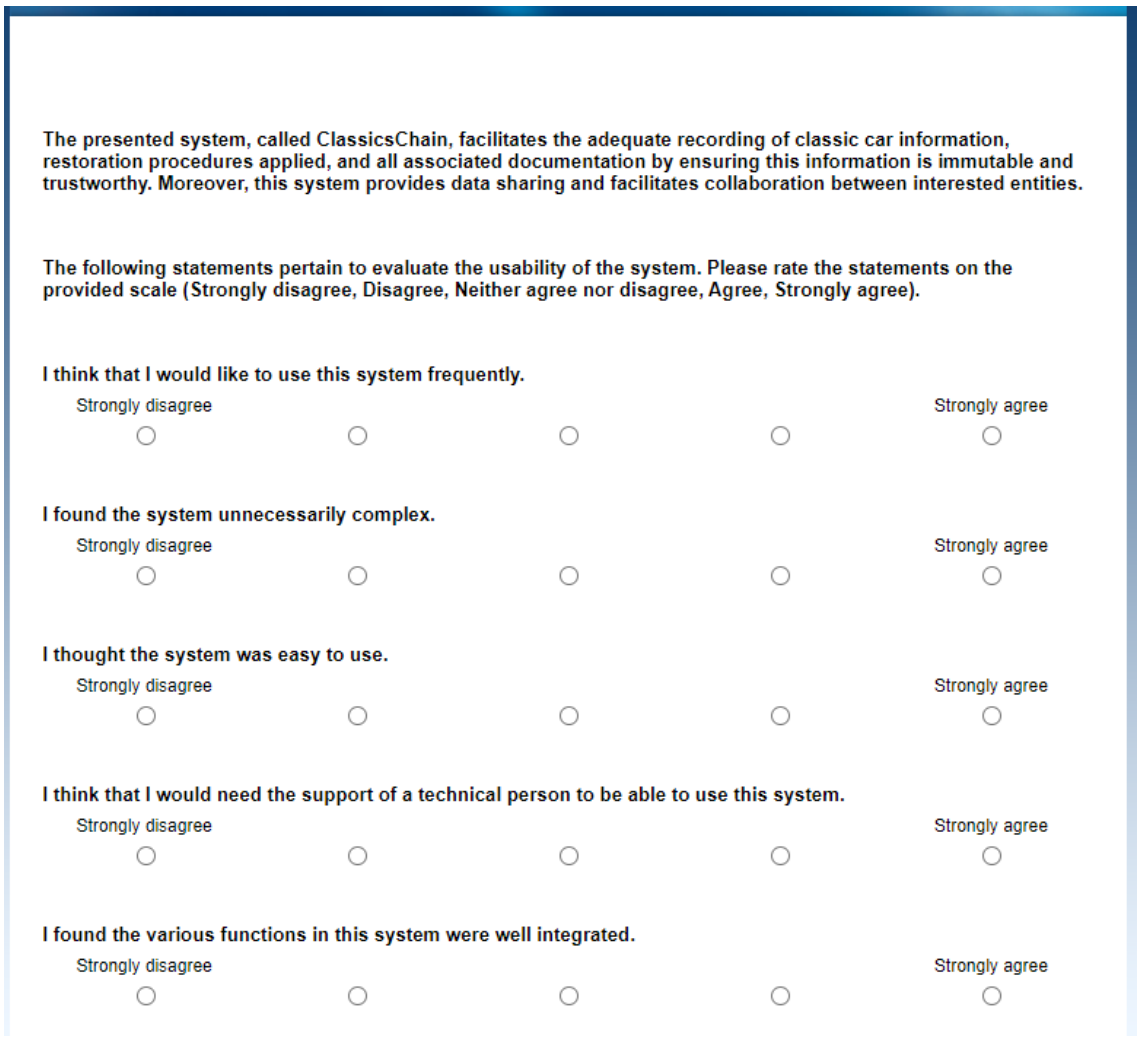


Figure B.3: Questionnaire for the System Usability Scale (SUS) of *ClassicsChain*. The second set of questions (1).

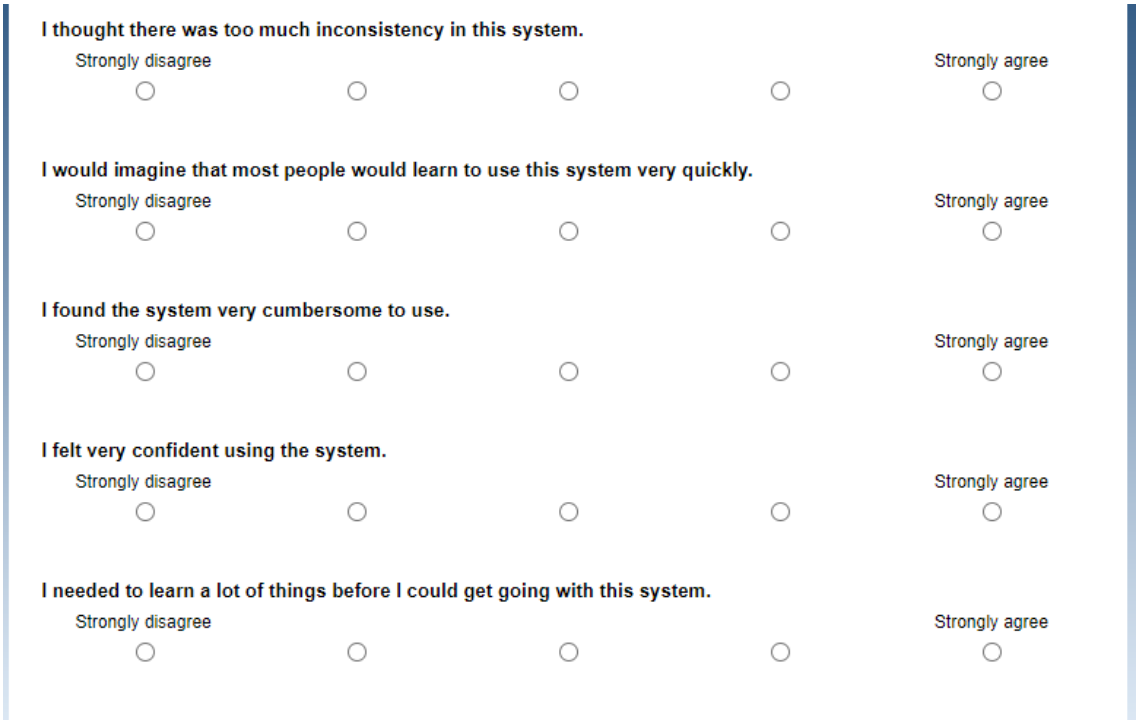


Figure B.4: Questionnaire for the SUS of *ClassicsChain*. The second set of questions (2).

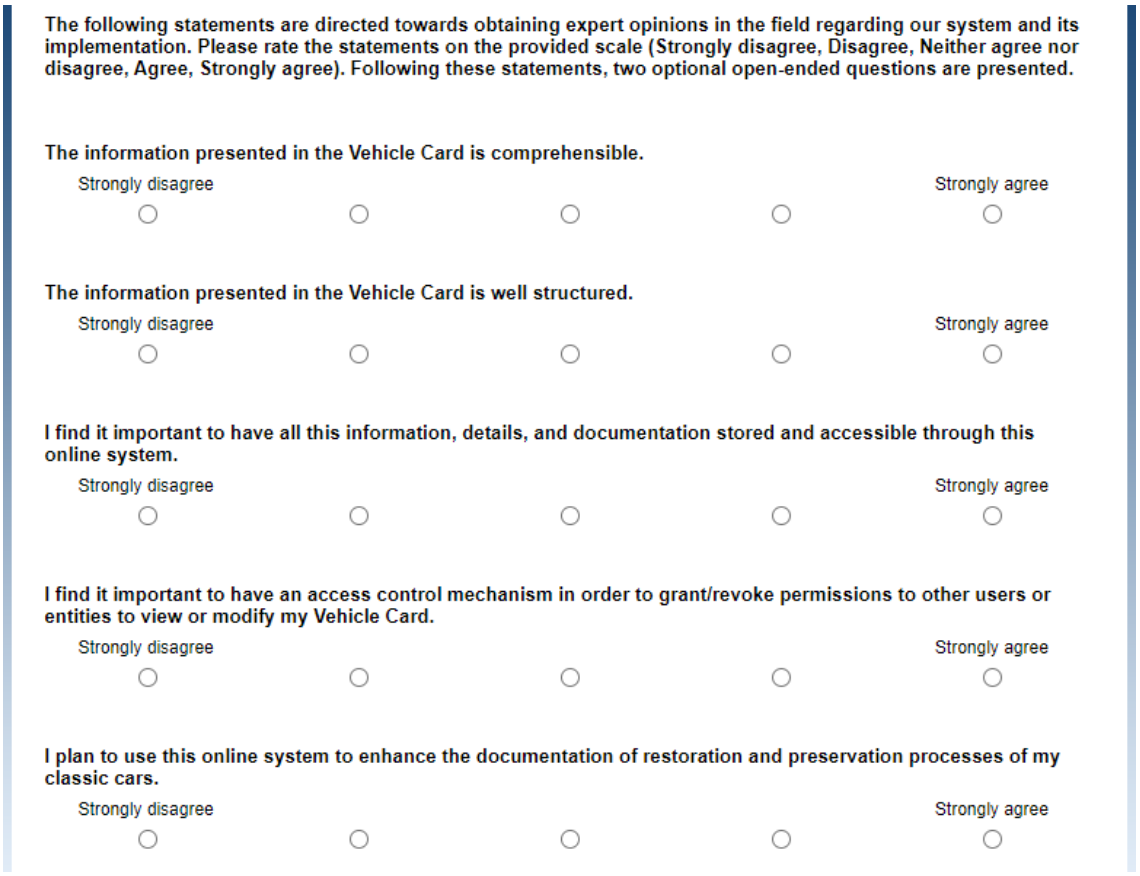


Figure B.5: Questionnaire for the final considerations on usability and relevance of *ClassicsChain*. The third set of questions (1).

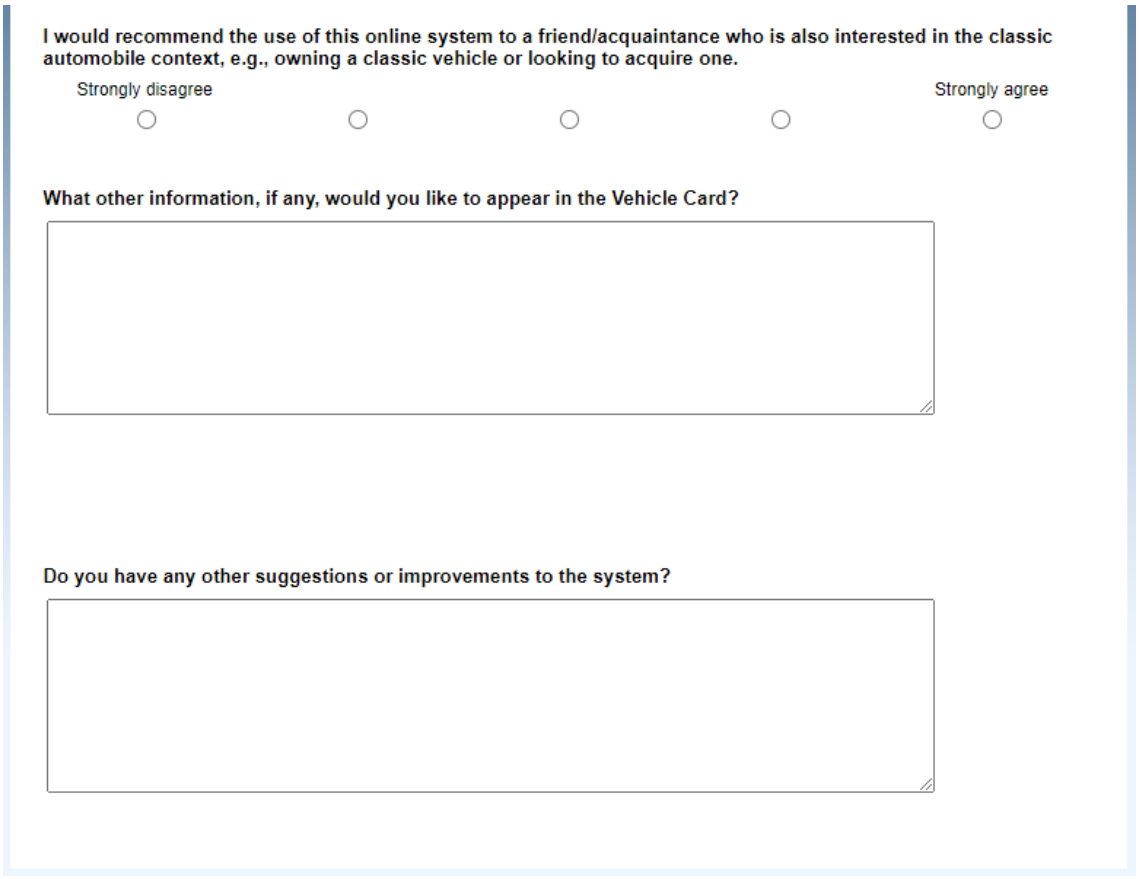


Figure B.6: Questionnaire for the final considerations on usability and relevance of *ClassicsChain*. The third set of questions (2).