



Instituto Federal da Bahia  
Departamento de Pós-Graduação e Qualificação

Programa de Pós-Graduação em Engenharia de Sistemas e Produtos

**A BLOCKCHAIN FRAMEWORK FOR  
TRACEABILITY IN SUPPLY CHAIN  
MANAGEMENT**

Edivaldo Mascarenhas Ferreira de Jesus Júnior

MASTER'S DISSERTATION

Salvador  
November 15, 2021



EDIVALDO MASCARENHAS FERREIRA DE JESUS JÚNIOR

**A BLOCKCHAIN FRAMEWORK FOR TRACEABILITY IN  
SUPPLY CHAIN MANAGEMENT**

Dissertation presented to Programa de Pós-Graduação em Engenharia de Sistemas de Produtos of PPGESP of Instituto Federal da Bahia as a partial requirement for obtaining the Master of Science degree on Systems and Products Engineering.

Advisor: Manoel Carvalho Marques Neto  
Co-advisor: Allan Edgard Silva Freitas

Salvador  
November 15, 2021

Biblioteca Raul V. Seixas – Instituto Federal de Educação, Ciência e Tecnologia da Bahia - IFBA – Campus Salvador/BA.

Responsável pela catalogação na fonte: Samuel dos Santos Araújo - CRB 5/1426.

J58b Jesus Júnior, Edivaldo Mascarenhas Ferreira de.  
A blockchain framework for traceability in supply chain management / Edivaldo Mascarenhas Ferreira de Jesus Júnior. Salvador, 2021.

90 f. ; 30 cm.

Dissertation (Mestrado Profissional em Engenharia de Sistemas e Produtos) – Instituto Federal de Educação, Ciência e Tecnologia da Bahia.

Advisor: Prof. Dr. Manoel Carvalho Marques Neto.

Co-advisor: Allan Edgard Silva Freitas.

1. Blockchain. 2. Supply chain management. 3. Traceability. 4. Smart contracts. I. Marques Neto, Manoel Carvalho. II. Freitas, Allan Edgard Silva. III. Instituto Federal de Educação, Ciência e Tecnologia da Bahia. IV. Título.

CDU 2 ed. 004

**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE SISTEMAS E PRODUTOS - PPGESP**

**A BLOCKCHAIN FRAMEWORK FOR TRACEABILITY IN SUPPLY CHAIN MANAGEMENT**

**EDIVALDO MASCARENHAS FERREIRA DE JESUS JÚNIOR**

Produto(s) Gerado(s): Relatório Final de Pesquisa

Orientador: Prof. Dr. Manoel Carvalho Marques Neto

Coorientador: Prof. Dr. Allan Edgard Silva Freitas

Banca examinadora:

---

Prof. Dr. Manoel Carvalho Marques Neto

Orientador – Instituto Federal da Bahia (IFBA)

---

Prof. Dr. Allan Edgard Silva Freitas

Coorientador – Instituto Federal da Bahia (IFBA)

---

Prof. Dr. Billy Anderson Pinheiro

Membro Externo – Amazônia Blockchain Solutions (AMACHAINS)

---

Profa. Dra. Flavia Maristela Santos Nascimento

Membro Externo – Instituto Federal da Bahia (IFBA)

Trabalho de Conclusão de Curso aprovado pela banca examinadora em 13/10/2021

Em 15 de outubro de 2021.

conforme decreto nº 8.539/2015.



Documento assinado eletronicamente por **FLAVIA MARISTELA SANTOS NASCIMENTO, Professor do Ensino Básico, Técnico e Tecnológico do Câmpus Salvador**, em 18/10/2021, às 09:02, conforme decreto nº 8.539/2015.



Documento assinado eletronicamente por **Billy Anderson Pinheiro, Usuário Externo**, em 18/10/2021, às 14:41, conforme decreto nº 8.539/2015.



Documento assinado eletronicamente por **MANOEL CARVALHO MARQUES NETO, Professor Titular**, em 18/10/2021, às 18:38, conforme decreto nº 8.539/2015.



A autenticidade do documento pode ser conferida no site [http://sei.ifba.edu.br/sei/controlador\\_externo.php?acao=documento\\_conferir&acao\\_origem=documento\\_conferir&id\\_orgao\\_acesso\\_externo=0](http://sei.ifba.edu.br/sei/controlador_externo.php?acao=documento_conferir&acao_origem=documento_conferir&id_orgao_acesso_externo=0) informando o código verificador **2032354** e o código CRC **4609E783**.

*This work is dedicated to the memory of  
Edivaldo M. F. de Jesus (1951–2021)*

## RESUMO

Os avanços nas tecnologias da informação e comunicação reduziram as barreiras físicas, políticas e culturais entre as nações. Essas tecnologias permitiram a globalização ao acesso de matérias-primas, bens e serviços. A complexa rede de relacionamentos que envolve quem fornece materiais, fabrica componentes ou subprodutos, monta ou mistura as partes e entrega o produto final no mercado é conhecida como cadeia de suprimentos. O rápido crescimento das tecnologias da Internet permitiu o surgimento de soluções emergentes aplicadas em sistemas de rastreabilidade, na área da cadeia de suprimentos. No entanto, esses sistemas tendem a ser centralizados, monopolistas, assimétricos e opacos. Como consequência, essas aplicações estão susceptíveis à problemas de confiança como fraude, corrupção, adulteração e falsificação de informações. Da mesma forma, por ser um ponto único de falha, o sistema centralizado é vulnerável ao colapso. Atualmente, uma emergente tecnologia chamada blockchain apresenta uma nova abordagem baseada na descentralização. No entanto, por estar em seus estágios iniciais, ela tem alguns desafios a enfrentar, nos quais a rastreabilidade a escalabilidade e o desempenho se tornam principalmente um desafio para encarar a enorme quantidade de dados no mundo real. Este trabalho pretende fornecer uma estrutura baseada em blockchain para facilitar o desenvolvimento de aplicações para rastreabilidade no gerenciamento da cadeia de suprimentos.

**Palavras-chave:** Blockchain; gerência de cadeia de suprimento; rastreabilidade; contratos inteligentes.





## ABSTRACT

Advances in information and communication technologies have reduced the physical, political and cultural barriers between nations. These technologies have enabled globalization to access raw materials, goods, and services. The complex web of relationships that provide materials manufacture the components, assemble or mix the parts and deliver the final product to market is known as the supply chain. The rapid growth of Internet technologies allowed the onset of many emerging technologies applied in traceability systems, in the supply chain area. However, these systems tend to be centralized, monopolistic, asymmetric, and opaque. Consequently, these systems result in trust problem, such as fraud, corruption, tampering and falsifying information. Likewise, by being a single point of failure, a centralized system is vulnerable to collapse. Nowadays, a new technology called the blockchain presents a whole new approach based on decentralization. Nonetheless, being in its early stages has some challenges to deal with, in which traceability, scalability, and performance become mainly defiance to face the massive amount of data in the real world. This work is intended to provide a blockchain-based framework to facilitate the development of applications for traceability in supply chain management.

**Keywords:** Blockchain; supply chain management; traceability; smart contracts.



# CONTENTS

<b>Chapter 1—Introduction</b>	1
<b>Chapter 2—Theoretical Background</b>	5
2.1 Blockchain . . . . .	5
2.1.1 Blockchain Properties . . . . .	6
2.2 Fundamentals of blockchain . . . . .	7
2.2.1 Cryptography . . . . .	7
2.2.1.1 Cryptographic Hashes . . . . .	8
2.2.1.2 Digital Signatures . . . . .	8
2.2.2 Consensus . . . . .	9
2.2.3 Distributed Ledger . . . . .	10
2.2.3.1 Transactions . . . . .	10
2.2.3.2 Blocks . . . . .	11
2.3 Public Blockchain Versus Permissioned Blockchain . . . . .	12
2.3.1 Public blockchain . . . . .	13
2.3.1.1 Consensus for Public Blockchain . . . . .	14
2.3.1.2 The pros and cons of public blockchain . . . . .	15
2.3.2 Permissioned Blockchain . . . . .	16
2.3.2.1 Consensus for permissioned blockchain . . . . .	17
2.3.2.2 The pros and cons of permissioned blockchain . . . . .	17
2.4 Smart Contracts . . . . .	18
2.4.1 Smart Contracts Security . . . . .	18
2.5 Supply Chain Management and blockchain . . . . .	19
2.5.1 Transparency . . . . .	21
2.5.2 Efficiency . . . . .	21
2.5.3 Safety and protection . . . . .	22
2.6 Hyperledger Fabric . . . . .	22
<b>Chapter 3—Related Work</b>	25
3.1 Traditional Systems . . . . .	25
3.2 Blockchain-based Systems . . . . .	26
3.3 Comparison with the presented work . . . . .	26

<b>Chapter 4—Proposed Solution</b>	29
4.1 Application Architecture . . . . .	29
4.1.1 WebApp - FrondEnd . . . . .	31
4.1.1.1 User Interaction . . . . .	31
4.1.1.2 Backend Integration . . . . .	32
4.1.2 WebApp - BackEnd . . . . .	32
4.1.2.1 API Gateway . . . . .	33
4.1.2.2 Service Layer . . . . .	34
4.1.2.3 Resource Locator . . . . .	34
4.1.3 Blockchain . . . . .	34
4.1.3.1 Smart contract . . . . .	34
4.1.3.2 Chaincode . . . . .	35
4.1.3.3 Ledger . . . . .	35
4.1.4 Data Storage . . . . .	35
4.1.4.1 Filesystem . . . . .	36
4.1.4.2 Database . . . . .	36
4.1.4.3 Blockchain . . . . .	36
4.2 Actions And Actors . . . . .	36
4.2.1 Setup . . . . .	36
4.2.2 Data Insertion . . . . .	37
4.2.3 Visualization . . . . .	37
4.3 Implementation Details . . . . .	37
4.4 Proof of Concept . . . . .	39
4.5 Use Example . . . . .	39
<b>Chapter 5—Conclusion and Future work</b>	47
<b>Appendix A—Project Management</b>	55
A.1 Activities . . . . .	55
A.1.1 Front end . . . . .	55
A.1.2 Back end . . . . .	56
A.1.3 Hyperledger Blockchain . . . . .	56
A.2 User stories . . . . .	56
A.3 Non-functional requirements . . . . .	58
<b>Appendix B—Smart Contract and backend endpoints</b>	59
B.1 Template for config file . . . . .	59
B.2 Assets, asset items, steps, and actors structs . . . . .	60
B.3 Main function . . . . .	61
B.4 Create Asset . . . . .	61
B.5 Move asset item . . . . .	62
B.6 Track asset item . . . . .	63

B.7 Audit Methods . . . . .	65
B.8 Backend Endpoints . . . . .	68
<b>Appendix C—Data Structure</b>	<b>69</b>



# LIST OF FIGURES

2.1	Signatures validated by public/private key (CHEGG, 2017). . . . .	9
2.2	Structure of a block. (ZHENG et al., 2016) . . . . .	11
2.3	Blockchain representation (MICHAEL; COHN; BUTCHER, 2018) . . . . .	12
2.4	Hyperledger Fabric transaction flow (MANEVICH; BARGER; TOCK, 2018). . . . .	23
4.1	Application architecture of Supply Chain Management - Blockchain Plat- form (SCM-BP) . . . . .	30
4.2	API Gateway. . . . .	33
4.3	SCM User flow . . . . .	38
4.4	Fill in asset info. . . . .	40
4.5	Adding actors. . . . .	41
4.6	Defining steps. . . . .	41
4.7	Review asset details before submitting. . . . .	42
4.8	Asset Items list. . . . .	42
4.9	Actors list. . . . .	43
4.10	Steps list. . . . .	43
4.11	Create asset item form. . . . .	44
4.12	Move asset item form. . . . .	45
4.13	Track an asset item forward. . . . .	46
4.14	Track an asset item backward. . . . .	46
C.1	SCM-BP data structure . . . . .	69





## LIST OF TABLES

2.1	Public Vs. Permissioned blockchain (101BLOCKCHAINS, 2020): . . . .	13
3.1	Related Works and their main characteristics, strategies and results. . . .	28
A.1	SCM-BP User Stories . . . . .	57
A.2	Non-functional requirements of SCM-BP . . . . .	58



## LIST OF ACRONYMS

<b>SCM</b>	Supply Chain Management . . . . .	1
<b>P2P</b>	peer-to-peer . . . . .	5
<b>SK</b>	secret key . . . . .	8
<b>PK</b>	public key . . . . .	8
<b>BG</b>	Byzantine generals . . . . .	9
<b>BTC</b>	Bitcoin . . . . .	13
<b>ETH</b>	Ethereum . . . . .	13
<b>XRP</b>	Ripple . . . . .	13
<b>PoW</b>	Proof-of-work . . . . .	13
<b>BFT</b>	Byzantine fault tolerance . . . . .	16
<b>SC</b>	Smart contract . . . . .	18
<b>KYC</b>	Know-Your-Customer . . . . .	22
<b>AML</b>	Anti-Money Laundering . . . . .	22
<b>IoT</b>	Internet of Things . . . . .	25
<b>RFID</b>	Radio frequency identification . . . . .	25
<b>SaaS</b>	Software-as-a-Service . . . . .	25
<b>SPA</b>	Single-page application . . . . .	31
<b>SSO</b>	Single Sign-on . . . . .	32
<b>API</b>	Application programming interface . . . . .	36
<b>SQL</b>	Structured Query Language . . . . .	36
<b>DBMS</b>	Database Manager System . . . . .	39
<b>SCM-BP</b>	Supply Chain Management - Blockchain Platform . . . . .	57



## Chapter

# 1

## INTRODUCTION

Hundreds of years ago, supply chains were reasonably straightforward. Mines and farms provided natural resources to skilled craftsmen like blacksmiths and tailors, creating and selling finished products. Today's supply chains are much more complicated, fragmented, and difficult to understand. Hundreds or even thousands of supplies worldwide contribute to making and shipping products purchased by a customer. Most of the time, various companies do not know each other, and a final consumer likely doesn't know anything about how, where, when, or under what conditions the products passed through. This is not just a problem for consumers. Today's supply chains are so complex that even big industry players have difficulty tracking how their goods get made.

Many systems have been developed to solve problems that come with this complexity, such as supply chain visibility and traceability. However, these systems typically store information in standard databases controlled by service providers. This centralized data storage becomes a single point of failure and risks tampering. Products' origin information may be essential, and a central server may be changed, generating doubt about confiability in that spot. In this scenario, a centralized organisation may become a vulnerable target... for bribery, and then the whole system can not be trusted anymore.

The complex web of relationships that provide materials manufacture the components, assemble or mix the parts and deliver the final product to market is known as the supply chain (BUURMAN, 2002).

Management is on the verge of a breakthrough in understanding how industrial company success depends on the interactions between information flows, materials, money, human resources, and capital equipment. The way these five flow systems interlock to amplify one another and cause change and fluctuation will form the basis for anticipating the effects of decisions, policies, organizational forms, and investment choices (FORRESTER, 1958).

The term Supply Chain Management (SCM) has risen to prominence over the past fifteen years (COOPER; LAMBERT; PAGH, 1997). For example, at the 1995 Annual Conference of the Council of Logistics Management, 13.5% of the concurrent session titles contained the words "supply chain." At the 1997 conference, just two years later,

the number of sessions containing the term rose to 22.4%. Moreover, the term is frequently used to describe executive responsibilities in corporations (LONDE, 1997). SCM has become such a "hot topic" that it is difficult to pick up a periodical on manufacturing, distribution, marketing, customer management, or transportation without seeing an article about SCM or SCM-related topics (ROSS, 1997).

La Londe and Masters proposed that a supply chain is a set of firms that pass materials forward. Usually, several independent firms are involved in manufacturing a product and placing it in the hands of the end-user in a supply chain—raw material and component producers, product assemblers, wholesalers, retailer merchants, and transportation companies are all members of a supply chain (LONDE; MASTERS, 1994). By the same token, Lambert, Stock, and Ellram define a supply chain as the alignment of firms that brings products or services to market.

Another definition notes that a supply chain is the network of organizations involved, through upstream and downstream linkages, in the different processes and activities that produce value in the form of products and services delivered to the ultimate consumer (CHRISTOPHER, 2017). In other words, a supply chain consists of multiple firms, both upstream (i.e., supply) and downstream (i.e., distribution), and the ultimate consumer.

As summarizing, (MENTZER et al., 2001) defines a supply chain as a set of three or more entities (organizations or individuals) directly involved in the upstream and downstream flows of products, services, finances, or information from a source to a customer.

To begin with, the starting point of a supply chain is the extraction of raw materials and how they are first processed (preprocessed) by suppliers for delivery in the next step. The next step is called manufacturing, where the conversion process for raw materials takes place. Following this, the constructed products are passed to the distributors responsible for allocating them to multiple intermediaries, such as wholesalers and retailers. Distributors also maintain an active inventory of products, as previous products are connected to suppliers. Subsequently, wholesalers do not sell products directly to the public but other retailers, whereas retailers dispose of products purchased to end-users. Finally, consumers are who buy or receive goods or services for personal use and not for commercial resale or trade purposes (LITKE; ANAGNOSTOPOULOS; VARVARIGOU, 2019).

The manufacturer needs to validate crucial information about the natural resources they collected by reading and verifying all tags included in its transactions and then proceeding to the proper manufacturing step. Some products have more significant value by region of origin or guarantee a fair process, so traceability is essential. New transactions with new information, such as manufacturer name, field experience, and more, are sent after the internship has been completed. Then the products are delivered to distributors. Distributors can sell products to wholesalers and retailers. This process is represented by transactions that contain essential data, such as merchant and customer address, exchange value, product raw material quality, and more (SAUER; SEURING, 2018).

As the distributors sell products to intermediates (generally not end-users), they must check the progress route until that stage, such as the raw material origin, manufacturer's company popularity, distributor address, and others. Retailers can audit a product's natural resource quality and get the appropriate feedback before selling it to the consumer.

After that, when a distributor sends the product to the wholesaler, some details, such as manufacturer name, field experience, and others, are submitted after similarly completing acts. A wholesaler needs to check this information and execute their selling to another wholesaler or retailer company. The same applies to retail companies. Finally, end-users obtain the final product and should track and verify all aspects from the beginning of its supply chain journey (LITKE; ANAGNOSTOPOULOS; VARVARIGOU, 2019).

There are billions of products being manufactured daily through complex supply chains extending to all parts of the world. However, there is very little information on how, when, and where these products originated, were manufactured, and used during their life cycle (HORIUCHI, 2015).

Before reaching the end consumer, the goods go through an often vast network of retailers, distributors, carriers, warehousing facilities, and suppliers who participate in the design, production, delivery, and sales process of a product, but in many cases. These steps are a dimension invisible to the consumer (B., 2015).

Gryna e Juran (1998) defines traceability as the ability to preserve the identity of the products and their origins so that the collection, documentation, and maintenance of information related to all processes in the production chain must be ensured. For a food product, traceability represents the ability to identify where and how it was grown, as well as the ability to track its post-harvest history and to identify the processes performed at each step in the production chain through records. Traceability is required primarily for (HORIUCHI, 2015): Improve credibility with customers and consumers, ensure that only quality materials and components are present in the final product, better allocate responsibilities, identify products that are distinct but may be confused, enable the return of defective or suspect products, and find the causes of failures and take steps to repair them at the lowest possible cost.

Consumers consider traceability as part of a standard protection package when purchasing products. Traceability can improve credibility in this scope since all the related providers and dealers or another agent between the raw manufacturers and the final consumer can be tracked. In a traceability system, the responsibility papers are well defined. A traceability system allows users to track products by providing information about them (e.g., originality, components, or locations) during production and distribution. Suppliers and retailers typically require independent, government-certified traceability service providers to inspect products throughout the supply chain. Vendors and retailers request traceability services for different purposes. Suppliers want to receive certificates to showcase their products. Retailers want to verify the origin and quality of products (LU; XU, 2017).

Supply chain visibility, or traceability, is one of the key challenges encountered in the business world. Most companies have little or no information about their own second and third-tier suppliers. Transparency and end-to-end visibility of the supply chain can help shape product, raw material, test control, and end product flow, enabling better operations and risk analysis to ensure better chain productivity (ABEYRATNE; MONFARED, 2016).

Folinas et al. (FOLINAS; MANIKAS; MANOS, 2006) identified that the efficiency of a traceability system depends on the ability to track and trace each asset and logistics unit



in a way that enables continuous monitoring from firstly processed until final clearance by the consumer.

Aung e Chang (2014) and Golan et al. (2004) set three main traceability objectives, namely: (1) better supply chain management, (2) product differentiation and quality assurance, and (3) better identification of non-compliant products. An additional objective is to maintain assurance of traceability following applicable regulations and standards.

Traceability systems typically store information in standard databases controlled by service providers. This centralized data storage becomes a single point of failure and risks tampering. Consequently, these systems result in trust problems, such as fraud, corruption, tampering, and falsifying information. Likewise, a centralized system is vulnerable to collapse (TIAN, 2017).

Nowadays, a new technology called the blockchain presents a whole new approach based on decentralization. Blockchain enables end-to-end traceability, bringing a standard technology language to the supply chain while allowing consumers to access the asset's history of these products through a software application. This characteristic is provided by an immutable register ledger that facilitates recording transactions and tracking assets across a network (GALVEZ; MEJUTO; SIMAL-GANDARA, 2018).

Blockchain and smart contracts could make supply chain management more straightforward and transparent. The idea is to create a single source of information about products and supply chains via a global ledger. Each component would have its entry on the blockchain that gets tracked over time. Both untrust companies could then update the status of goods in real-time. The result is that once the clients receive their products, they can track every piece back to its manufacturer. Theoretically, users could trace the supply chain back to the mines where the raw materials came from (GREVE et al., 2018).

Companies can also use the blockchain supply chain as a single source of truth for their products. They can manage and monitor risks within the supply chain, ensure the quality of delivery parts and track the delivery status. Additionally, companies can use smart contracts to manage and pay for the supply chain autonomously. This would reduce the need for large contract invoices on the back-and-forth of refund requests for faulty components. Those same smart contracts could assist with shipping and logistics tracking valuable products as they travel around the world. Using blockchain, companies can finally have a complete picture of their products at every stage in the supply chain, bringing transparency to the production process while reducing the cost of manufactured goods.

This work presents Supply Chain Management - Blockchain Platform (SCM-BP), a generic open source framework intended to be used in any supply chain correlated to assets and products. It also presents a use case of this framework applied.

This dissertation is structured as follows: Chapter 2 presents several technologies involved in preparing this dissertation, introduces essential concepts of the Computer area in which the context of this project is inserted, and presents business concepts related to supply chain management. Chapter 3 shows correlated works. Chapter 4 presents the solution, its architecture, details of the executed implementation, and exposes the validation and results. The last chapter presents the conclusions and future work.

*In this section, the main concepts studied are presented, which provided subsidies for the proposed project's development.*

## THEORETICAL BACKGROUND

### 2.1 BLOCKCHAIN

Recently, cryptocurrency has attracted extensive attention from both the industry and the academy. Bitcoin, often called the first cryptocurrency, had massive success, with the capital market coming to \$10 billion in 2016 (COINDESK, 2016). Blockchain is the central mechanism of Bitcoin and was first proposed in 2008 and implemented in 2009 (NAKAMOTO et al., 2008). The blockchain can be considered a public ledger, in which all committed transactions are stored in a block chain. This chain grows continuously when new blocks are attached to it (ZHENG et al., 2016).

At the origin of the blockchain is the Bitcoin protocol, proposed by Satoshi Nakamoto (NAKAMOTO et al., 2008). This paper proposes a peer-to-peer (P2P) network where transactions with the cryptocurrency bitcoin, proposed by customers, are received by servers, who will decide, through a consensus protocol based on cryptographic challenges, on the order in which they will be carried out and permanently stored in a chain of blocks, replicated on each server. According to FORMIGONI (2017), it was the creation of a digital currency that worked in a P2P network that allowed the sending of online payments in a completely secure way, without the involvement of financial institutions, for all participants from the web. In this sense, blockchain was created motivated by an efficient, economical, reliable, and secure system to conduct and record financial transactions. Hence the question: what is the relationship between blockchain and Bitcoin? Blockchain is the platform used for the operation of the Bitcoin network and several other cryptocurrencies.

While the system of financial institutions that serve as third parties reliable processors for processing payments works well for most, still it suffers from the shortcomings inherent in the model based on confidence. In addition, the cost of mediation increases transaction costs, limiting the minimum practical size of the transaction and eliminating the possibility of occasional small transactions. To solve these problems, (NAKAMOTO et al.,

2008) defined an electronic payment system called Bitcoin based on cryptographic proof rather than reliability, allowing either party to be willing to transact directly without the need for a reliable third party.

This revolution began with a new marginal economy on the Internet. Bitcoin emerges as an alternative currency issued and not backed by a central authority but by automated consensus among networked users. However, its true uniqueness lay in the fact that it did not require that users trust each other. Through self-policing algorithmically, any malicious attempt to circumvent the system would be rejected. In a precise and technical definition, Bitcoin is digital money transacted via the Internet in a decentralized system without bail, using a ledger called blockchain. It is a new way of combining peer-to-peer file sharing with public key encryption (SWAN, 2015).

For (SWAN, 2015), besides the currency ( "Blockchain 1.0"), smart contracts ("2.0") demonstrate how the blockchain is in a position to become the fifth disruptive computing paradigm after mainframes, PCs, Internet, and mobile/ social networks. Bitcoin is starting to become a digital currency, but the technology blockchain behind it can be much more significant.

The rapid growth in blockchain technology adoption and the development of applications based on this technology have revolutionized financial services industries. In addition to bitcoin, typical applications of blockchain usage vary from proprietary networks used to process financial and insurance claims to platforms that can issue and trade equity and corporate bonds (MICHAEL; COHN; BUTCHER, 2018).

Blockchain exists with real-world implementations beyond cryptocurrencies, and these solutions deliver powerful benefits to healthcare organizations, bankers, retailers, and consumers. The potential benefits of blockchain are more than just economic. They extend to the political, humanitarian, social, and scientific domains. Specific groups are already harnessing their technological capacity to solve real-world problems (MICHAEL; COHN; BUTCHER, 2018).

### 2.1.1 Blockchain Properties

Blockchain technology has key features such as centralization, persistence, anonymity, and auditability. Blockchain can function in a decentralized environment that is activated by integrating several key technologies such as cryptographic hash, digital signature (based on asymmetric encryption), and distributed consensus engine. With blockchain technology, a transaction may occur in a decentralized manner. As a result, blockchain can significantly save costs and improve efficiency (ZHENG et al., 2016). The primary properties of the blockchain are considered innovative and enable rapid adoption for technology (GREVE et al., 2018):

- Decentralization: Applications and systems run in a distributed manner, through the establishment of trust between the parties, without the need for a trusted intermediary entity. This is the primary motivator for the growing interest in the blockchain;
- Availability and integrity: All datasets and transactions are securely replicated in

different nodes to keep the system available and consistent;

- **Transparency and auditability:** All transactions recorded in the ledger are public and can be verified and audited. Furthermore, technology codes are often open, verifiable;
- **Immutability and Irrefutability:** Transactions recorded in the ledger are immutable. Once registered, they cannot be refuted. Updates are possible based on the generation of new transactions and the realization of a new consensus;
- **Privacy and Anonymity:** It is possible to offer privacy to users without the third parties involved having access and control of their data. In technology, each user manages their keys, and each server node stores only encrypted fragments of user data. Transactions are somewhat anonymous, based on the address of those involved in the blockchain;
- **Disintermediation:** Blockchain enables the integration between different systems directly and efficiently. Thus, it is considered a connector of complex systems (systems of systems), allowing the elimination of intermediaries to simplify the design of systems and processes;
- **Cooperation and Incentives:** Offer of an incentive-based business model in the light of game theory. On-demand consensus is now offered as a service at different levels and scopes.

## 2.2 FUNDAMENTALS OF BLOCKCHAIN

In this section, the key elements of the blockchain that contribute to its properties will be presented such as integrity integrity, immutability, transparency, availability, disintermediation, decentralization.

### 2.2.1 Cryptography

Blockchain relies heavily on encryption to satisfy system and application security requirements. As the word suggests, cryptocurrencies also make heavy use of encryption. Encryption provides a mechanism for safely encoding the rules of system encryption on the system itself. This can prevent tampering, and misconceptions and coding in a mathematical protocol the rules for creating new currency units. So, before understanding blockchains correctly, it is necessary to understand the cryptographic foundations they trust (NARAYANAN et al., 2016).

Cryptography is a profound academic field of research that uses many advanced mathematical techniques that are notoriously subtle and complicated. In this chapter, cryptographic hashes and digital signatures will be defined, which stand out among the most used resources. For more details, the reader should refer to the book (NARAYANAN et al., 2016).

**2.2.1.1 Cryptographic Hashes** A hash function is a mathematical function with the three properties to be followed (NARAYANAN et al., 2016):

- Its input can be any string of any length;
- Produces a fixed size output;
- It is efficiently computable. Intuitively, this means that it is possible to find out the hash function output within a reasonable time for a given input string. Technically, hashing a  $n$ -bit string must have an  $O(n)$  runtime.

These properties define a general hash function. Cryptographic hash functions (or cryptographic summaries) are unidirectional and hardly allow retrieving the original value  $x$  from the hash  $h$ . For a hash function to be cryptographically secure, it must satisfy the following three properties: (1) collision resistance, (2) hiding, and (3) puzzle friendliness (GREVE et al., 2018).

A collision occurs when two distinct inputs produce the same output. A hash function  $H$  is collision-resistant when it is impossible to find two values  $x$  and  $y$ , such that  $x \neq y$  and  $H(x) = H(y)$  (NARAYANAN et al., 2016).

The hidden property states that having the hash function output  $y = H(x)$ , there is no possible way to find the  $x$  input (GREVE et al., 2018).

A hash function  $H$  is considered puzzle friendliness if for each possible output value of  $n$  bits  $y$  if  $k$  is chosen from a distribution with high min-entropy, then it is impracticable to find  $x$  such that  $H(k||x) = y$  in time significantly less than  $2^n$  (NARAYANAN et al., 2016).

**2.2.1.2 Digital Signatures** A digital signature is supposed to be a digital analog of a handwritten paper signature. Two signature properties are desired, which correspond well to the analogy of the handwritten signature. First, only one person can make their signature, but anyone can verify if it is valid. Secondly, it is desired that the signature be linked to a specific document, so it cannot indicate the agreement or endorsement to a different document (MERKLE, 1989). Moreover, it is not possible to forge a signature in such a way as to reuse it in some other context. That is, signatures must be irrefutable.

To implement digital signatures, asymmetric key encryption is used. A secret key (SK) is used for signing the document, and a public key (PK) is used to attest the signature's authenticity (GREVE et al., 2018).

A digital signature consists of the following three algorithms (NARAYANAN et al., 2016):

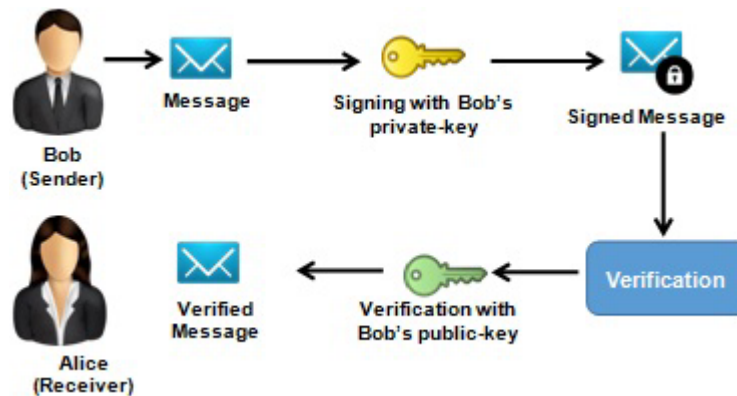
- $(sk, pk) := generateKeys(keysize)$  – The  $generateKeys()$  method receives a key size ( $keysize$ ) in the input and return a pair of public ( $pk$ ) and private ( $sk$ ) keys.
- $sig := sign(sk, msg)$  – The method  $sign()$  receives a message  $msg$  and a secret key ( $sk$ ) on entry and returns the signature  $sig$  f that message under  $sk$ .

- $isValid := verify(pk, msg, sig)$  – The method *verify* receives a public key ( $pk$ ), a message  $msg$ , and a signature ( $sig$ ) as input, and returns a boolean value:  $isValid = true$  if  $sig$  is a signature valid for  $msg$  under  $pk$ ;  $isValid = false$ , otherwise.

The following two properties must be maintained:

- Authenticity: Signatures can be validated:  
 $verify(pk, message, sign(sk, message)) == true$ .
- Signatures are existentially unfalsifiable: signature cannot be forged.

It is noted that *generateKeys()* and *sign()* can be random algorithms. Generating keys should be randomized because it should be generating different keys for different people. On the other hand, *verify()* will always be deterministic.



**Figure 2.1** Signatures validated by public/private key (CHEGG, 2017).

## 2.2.2 Consensus

The key to blockchain operation is that the network must agree collectively on the ledger's content. Instead of a central entity maintain control over information (such as a bank, for example), the data is shared among all. This requires the network to maintain the consensus around the information recorded in the blockchain. How this consensus is reached affects the security and economic parameters of the protocol (KOSTAREV, 2017).

In this context, consensus emerges as a fundamental problem since it allows distributed participants to coordinate their actions to reach joint decisions, ensuring the consistency of safety and system progress (liveness) despite failures (GREVE et al., 2018). More specifically, in the context of blockchain, consensus enables an agreement on the next block that will be added to the blockchain.

In the blockchain, reaching consensus between untrusted nodes is a transformation of the problem of Byzantine generals (BG) (LAMPORT; SHOSTAK; PEASE, 1982). In the BG problem, a group of generals who command a portion of the Byzantine army circles

the city. The attack would fail if only part of the generals attacked the city. Generals need to communicate to agree on the attack or not. However, there may be traitors in the generals. The traitor could send different decisions to different generals. This is an environment without trust.

Reaching consensus in such an environment is a challenge. This is also a challenge for blockchain because its network is distributed, and there is no central node that ensures that ledgers on distributed nodes be all the same. Nodes do not need to trust other nodes. Thus, some protocols are required to ensure that ledgers on different nodes are consistent (KOSTAREV, 2017). To solve the consensus problem, several algorithms have been proposed, the mainly used are Proof-of-Work, Proof-of-Stake, Delegated Proof-of-Stake, Leased Proof-Of-Stake, Proof of Elapsed Time, Practical Byzantine Fault Tolerance, Simplified Byzantine Fault Tolerance, Delegated Byzantine Fault Tolerance, Directed Acyclic Graphs, Proof-of-Activity, Proof-of-Importance, Proof-of-Capacity, Proof-of-Burn and Proof-of-Weight (MINGXIAO et al., 2017).

A good consensus algorithm means efficiency, security, and convenience. Current standard consensus algorithms still have many shortcomings. New consensus algorithms are created to solve some blockchain-specific problems (ZHENG et al., 2016).

### 2.2.3 Distributed Ledger

A distributed ledger is a data structure distributed by several nodes or computing devices. Each node replicates and saves an identical ledger copy. Each participating node in the network updates independently (GREVE et al., 2018).

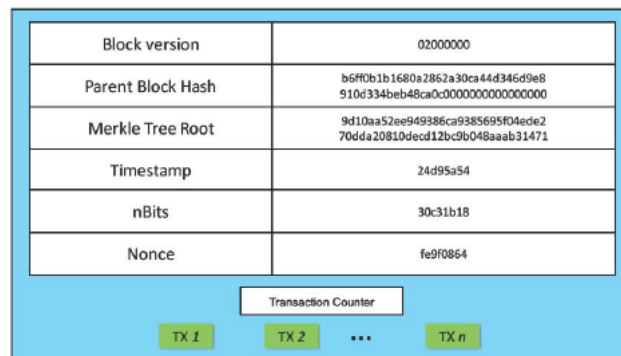
The innovative feature of distributed accounting technology is that any central authority does not maintain the ledger. Updates to the ledger are independently constructed and recorded by each node. The nodes then vote on these updates to ensure that most agree on the conclusion, based on some previous consensus algorithm. Once consensus has been reached, the distributed ledger updates itself, and the latest agreed version is saved on each separate node (SWAN, 2015). Thus, the distributed ledger is replicated and immutable.

**2.2.3.1 Transactions** The most fundamental definition of a transaction is an atomic event allowed by the underlying protocol from a technical standpoint. A transaction determines a sequence of state operations. It adds a transfer of an asset or, generally speaking, a smart contract. In a basic case, the transaction girds a digital signature of the issuer holding the asset and the receiver's address and inputs and outputs for the transaction. Each transaction must contain both Inputs and Outputs, just like in an accounting book. Entries indicate the previous transaction hash related to the current one (GREVE et al., 2018). Validating a Transition involves:

1. signature verification;
2. confirmation of existing values from hashes of previous referenced transactions;
3. confirmation that any other transactions did not previously spend the amount.

In this case, it is necessary to search the blockchain between the block from the referenced transaction to the last block of the structure. Each node of the blockchain network independently validates transactions, and this feature contributes to the decentralization of the process (GREVE et al., 2018).

**2.2.3.2 Blocks** Blocks contain a header with the information needed for the current maintenance and its validation. A block consists of the block header and block body, as shown in Figure 2.2.



**Figure 2.2** Structure of a block. (ZHENG et al., 2016)

In particular, the header pack includes:

- Block version: indicates which set of block validation rules to follow;
- Parent block hash: A 256-bit hash value that points to the previous block;
- Merkle tree root hash: The hash value of all transactions in the block;
- Timestamp: Current date and time as seconds since 1970-01-01T00:00 UTC;
- nBits: current hashing target in a compact format;
- Nonce: A 4-byte field, usually starting with 0 and increasing for each hash.

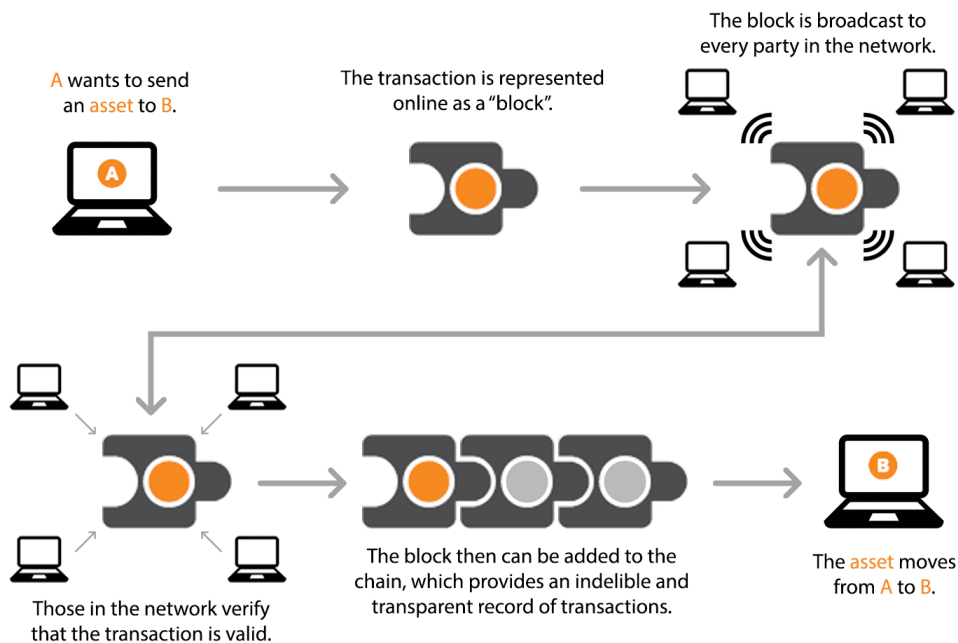
The body's block consists of a transaction counter and transaction. The maximum number of transactions a block can hold depends on block size and the size of each transaction. Blockchain uses an asymmetric encryption mechanism to validate transaction authentication. A digital signature based on asymmetric encryption is used in an untrusted environment (ZHENG et al., 2016).

The validation of a block consists in verifying (i) if its structure is well-formed (ii) its hash is valid (meets the challenge), (iii) its size is within the network accepted limit, (iv) the set of transactions within the block is valid, (v) the first transaction (and only the first) is the coinbase transaction - which incorporates the generation of new cryptocurrencies in the system, besides acting as a reward mechanism. The blocks are validated



independently by each node of the blockchain network, and this feature contributes to the process of decentralization (GREVE et al., 2018).

Figure 2.3 presents a visual representation of a blockchain (TIAN, 2017). When a person "A" wants to send an asset to person "B", the transaction is represented as a "block". This block is broadcast to every participant in the network to verify if this transaction is valid. Once validated, the block is added to the chain, providing an indelible and transparent record of transaction. The asset then, moves from A to B.



**Figure 2.3** Blockchain representation (MICHAEL; COHN; BUTCHER, 2018)

## 2.3 PUBLIC BLOCKCHAIN VERSUS PERMISSIONED BLOCKCHAIN

Blockchain networks can be categorized into two groups: public (or permissionless) blockchain and private (or federated or permissioned) blockchain (with permission and controlled access) (GREVE et al., 2018).

Since the beginning of blockchain technology, people have debated about public vs. permissioned blockchain. In an enterprise environment, it is essential to know the significant differences between these two. Public and permissioned blockchain examples play a considerable role in the companies looking for the perfect blockchain type for their solutions (101BLOCKCHAINS, 2020). Table 2.1 presents a comparison between public and permissioned blockchain.

**Table 2.1** Public Vs. Permissioned blockchain (101BLOCKCHAINS, 2020):

	<b>Public blockchain</b>	<b>Permissioned blockchain</b>
Access	Anyone	Single Organization
Authority	Decentralized	Partially decentralized
Transaction Speed	Slow	Fast
Consensus	Permissionless	Permissioned
Transaction Cost	High	Low
Data Handling	Read and Write access for anyone	Read and Write access for a single organization
Immutability	Full	Partial
Efficiency	Low	High

The primary difference between public and private blockchain is the level of access participants are granted. In order to pursue decentralization to the fullest extent, public blockchains are entirely open. Anyone can participate by adding or verifying data. The most common examples of public blockchain are Bitcoin (BTC) and Ethereum (ETH). A public blockchain is about accessibility, which is evident in its use (SELFKEY, 2020).

Conversely, permissioned blockchain (also known as private blockchain) only allows certain entities to participate in a closed network. Participants are granted specific rights and restrictions in the network, to have full access or limited access at the network's discretion. As a result, permissioned blockchain is more centralized as only a small group controls the network. The most common examples of permissioned blockchains are Ripple (XRP) and Hyperledger (BLOCKGEEKS, 2018).

Additionally, some public blockchains also allow anonymity, while permissioned ones do not. For example, anyone can buy and sell Bitcoin without having their identity revealed. It allows everyone to be treated equally. Whereas with permissioned blockchains, the identities of the participants are known. This is typical because permissioned blockchain is used in the corporate and business-to-business sphere, where it is crucial to know who is involved (101BLOCKCHAINS, 2020).

### 2.3.1 Public blockchain

On a public or permissionless blockchain, any person can participate without a specific identity. There are no restrictions when it comes to participation. Public blockchains typically involve a native cryptocurrency and often use Proof-of-work (PoW) consensus and economic incentives (ANDROULAKI et al., 2018). Anyone can audit a public blockchain, and each node has as much transmission power as any other. For a transaction to be considered valid, it must be authorized by all nodes constituents via the consensus process. As long as each node meets protocol-specific stipulations, their transactions can be validated and thus added to the chain (COMSTOR, 2017).

In a public blockchain, all the participants have equal rights. People can join in and participate in consensus and transact with their peers as they please. Everyone can see the ledger as well, thus maintaining transparency in any time (101BLOCKCHAINS,

2020).

As the P2P network node-set is unknown, its membership is dynamic, allowing random node entrances, exits, and anonymity. Blockchain can act globally without the control of its participants, who do not even trust each other mutually. Are examples of public blockchain: the Bitcoin, Ethereum, and several other cryptocurrencies (BASHIR, 2018; ANTONOPOULOS, 2017).

If a fully decentralized network system is required, then public blockchain is the way to go. However, it can get problematic when incorporating a public blockchain network with the enterprise blockchain process (101BLOCKCHAINS, 2020).

The main characteristics of a public blockchain are:

- Open environment: public blockchain is open for all. The system typically has an incentive mechanism to encourage more participants to join the network;
- Anonymous nature: everyone is anonymous. Users will not use their real name or real identity here. Everything would stay hidden, and no one can track data based on that;
- No regulations: public blockchain does not have any regulations that the nodes have to follow. So, there is no limit to how one can use this platform for their betterment. However, the main issue is that enterprises cannot work in a non-regulated environment;
- True decentralization: public blockchain provides true decentralization. This is something that is quite absent in private blockchain networks. As everyone has a copy of the ledger, it creates a distributed nature as well. Basically, in this type of blockchain, there is not a centralized entity. Thus, the responsibility of maintaining the network is solely on the nodes. With help from a consensus algorithm, they are updating the ledger, promoting fairness;
- Full transparency: public blockchain companies tend to design the platforms fully transparent to anyone on the ledger. It means that users can see the ledger anytime they want. So, there is no scope for any corruption or discrepancies. Anyhow, everyone has to maintain the ledger and participate in consensus;
- Immutability: the public blockchain network is fully immutable. Once a block gets on the chain, there is no way to change it or delete it. So, it ensures that no one can alter a particular block to benefit from others.

**2.3.1.1 Consensus for Public Blockchain** Due to the uncertainties regarding the participants, public blockchains generally adopt mining-based consensus mechanisms. In these mechanisms, miners vie with each other for consensus leadership, using computational power, possession power over cryptocurrency or other election-relevant powers that cannot be monopolized such that the same knots always come out victorious) (GREVE et al., 2018).

Compensation to these miners for their work is often cryptocurrency. These incentives are critical to preventing Byzantine attacks by solving the fundamental challenge of agreement globally. Currently, proof of work is one of the few prosperous and resilient consensus approaches to Sybil attacks (DOUCEUR, 2002) (impersonation attacks, when malicious users become impersonate others).

In the mining process, new transactions are vilified by the nodes in the whole system known as “miners” before being added to the blockchain. Miners add new blocks on the chain or new transactions on the block by a consensus algorithm, which must be confirmed by the majority of all the nodes in the system, like a voting operation, as the valid data. Blockchain-based systems rely on miners to aggregate transactions into blocks and append them to the blockchain. Once a sufficient number of nodes confirms the transaction, it becomes a valid and permanent part of the database.

**2.3.1.2 The pros and cons of public blockchain** One of the most significant advantages of a public blockchain is that there is no need for trust. Everything is recorded, public, and cannot be changed. Everyone is encouraged to do the right thing for the betterment of the network. There is no need for intermediaries (BLOCKGEEKS, 2018).

The other significant advantage is security. The more decentralized and active a public blockchain is, the more secure it becomes. As more people work on the network, it becomes harder for any attack to succeed. It is nearly impossible for malicious actors to band together and control the network (SELFKEY, 2020).

The final piece of what makes public blockchain successful is transparency. All data related to transactions are open to the public for verification. The transparency of public blockchain increases potential use cases, such as decentralized identity (COMSTOR, 2017).

One of the biggest problems with public blockchain is speed. Public blockchains like Bitcoin are extremely slow, only managing to process seven transactions per second. Compare that to Visa, which can do 24,000 transactions per second, and we see where the problem is. Public blockchains are slow because it takes time for the network to reach a consensus. Additionally, the time needed to process a single block takes a long time compared to a private blockchain (BLOCKGEEKS, 2018).

Public blockchains also face scalability concerns. With the current state of things, public blockchains cannot compete with traditional systems. The more a public blockchain is used, the slower it gets because more transactions clog the network. However, steps are being taken to remedy this problem. An example is Bitcoin’s Lightning Network (SELFKEY, 2020).

Lastly, energy consumption has been a concern when it comes to the public blockchain. Bitcoin’s algorithm relies on Proof-of-Work, which relies on using much electricity to function. That being said, other algorithms such as Proof-of-Stake use far less electricity (SELFKEY, 2020).

### 2.3.2 Permissioned Blockchain

Permissioned, federated, or private blockchains, on the other hand, perform a blockchain between a set of known and identified participants. A permissioned blockchain provides a way to protect the interactions between entities with a common goal but does not trust each other, like companies that trade funds, assets, or information. Relying on peer identities, one permissioned blockchain may use the traditional consensus of Byzantine fault tolerance (BFT) (ANDROULAKI et al., 2018).

Federated blockchain has its known composition. It is formed by  $n$  processes whose inputs and outputs are subject to permissions. Participants are identified, authenticated, and authorized. This blockchain model aims to serve better corporate or private interests where participants have well-defined roles and can even organize themselves into groups. Examples of permissioned blockchain are Hyperledger Fabric (CACHIN et al., 2016) and some other projects (CACHIN; VUKOLIĆ, 2017).

As enterprises need privacy, permissioned blockchain use cases seem a perfect fit in this case. Without proper privacy, their competition can enter the platforms and leaks valuable information to the press. This, in the long term, can influence the brand value greatly. So, in some instances, companies need privacy greatly (101BLOCKCHAINS, 2020).

The main characteristics of public blockchains are:

- High efficiency: compared with a public blockchain, which tends to lack efficiency because it assumes that everyone (or every node) is part of the network. As a result, when more people try to use the features from network, it takes up many resources that the platforms cannot back up. Thus, it slows down rapidly. On the other hand, private blockchain only allows a smaller number of people in the network. In many cases, they even have specific tasks to complete. So, there is no way they can take up extra resources and slow down the platform;
- Full privacy: Private blockchain tends to focus on privacy concerns. Enterprises always deal with security and privacy issues. Moreover, they also deal with such sensitive information daily. If even one of them gets leaked, it can mean a massive loss for the company;
- Empowering enterprises: private blockchain solutions work to empower the enterprises as a whole rather than individual employees. In fact, companies do need great technology to back up their processes. Furthermore, these solutions are mainly for the internal systems of an enterprise. This is one of the best use cases of the private blockchain;
- Stability: private blockchain solutions are stable. Basically, in every blockchain platform, users have to pay a certain fee to complete a transaction. However, the fee can increase significantly in public platforms due to the pressure of nodes requesting transactions. When there are too many transaction requests, it takes time to complete them. More so, as time increases, the fee increases drastically;

- **Low fees:** in private blockchain platforms, the transaction fees are meager. Unlike public blockchain platforms, the transaction fee does not increase based on the number of requests. So, no matter how many people request a transaction, the fees will always stay low and accurate;
- **Saves money:** in reality, private blockchain saves much money. Maintaining a private blockchain is relatively simple compared to public blockchains. Private blockchain platforms take up only a few resources;
- **No illegal activity:** private blockchain platforms have authentication processes before any user logs into the network. What this process does is filter any intruders trying to get into the system;
- **Regulations:** for enterprise companies that have to follow many rules and regulations, private blockchain outlines all the rules, and the peer nodes have to follow them.

**2.3.2.1 Consensus for permissioned blockchain** Since it is a controlled network with  $n$  known participants and identified by the federation, classic Byzantine fault tolerance (BFT) protocols and deterministic Byzantine consensus can be adapted to the blockchain (ANDROULAKI et al., 2018).

In addition, there is no need to use incentives to the agreement, as the federation of stakeholders can establish its financial model of remuneration. Incentives, however, may be used for other purposes but, different from the evidence-based consensus, they are not essential to consensus (GREVE et al., 2018).

In the BFT literature, replication consistency is maintained by two principles:

- **No mistake:** leaders are prevented from making mistakes, so only one possible proposal per leader per rating;
- **Proposal Security:** a (higher-ranked) proposal can extend, but not modify, any lower-ranking compromised log prefix (ABRAHAM; MALKHI et al., 2017).

**2.3.2.2 The pros and cons of permissioned blockchain** A significant advantage of permissioned blockchain is speed. Permissioned blockchains have far fewer participants, meaning it takes less time for the network to reach a consensus. As a result, more transactions can take place. Permissioned blockchains can process thousands of transactions per second. Comparing that to Bitcoin's seven transactions per second is a massive difference (IORIO, 17).

Permissioned blockchains are also far more scalable. Since only a few nodes are authorized and responsible for managing data, the network can process more transactions. The decision-making process is much faster because it is centralized (SELFKEY, 2020).

However, the centralization of permissioned blockchain is one of its most significant disadvantages. Blockchain was built to avoid centralization and permissioned blockchain inherently becomes centralized due to its private network (BLOCKGEEKS, 2018).

Trust is also a more significant issue when it comes to permissioned blockchain. The credibility of a permissioned blockchain network relies on the credibility of the authorized nodes. They need to be trustworthy as they are verifying and validating transactions. The validity of records also can't be independently verified (BLOCKGEEKS, 2018).

Security is another concern with permissioned blockchain. With fewer nodes, it is easier for malicious actors to gain control of the network. Unfortunately, a permissioned blockchain is far more at risk of being hacked or having data manipulated (ABRAHAM; MALKHI et al., 2017).

## 2.4 SMART CONTRACTS

A smart contract is a computerized transaction protocol that executes the terms of a contract (SZABO, 1997). Its model has proposed a long time ago, and now this concept can be implemented with blockchain.

Nick Szabo has standardized the term Smart contract (SC) in the 90s (GREVE et al., 2018). It means: "an internal transaction protocol format that executes the terms of a contract. Their overall goals are to ensure common contractual conditions (such as payment terms, liens, confidentiality, and even compliance), minimize malicious and accidental exceptions, and the need for reliable intermediaries. Related economic objectives include reducing fraud losses, arbitration and execution costs, and other transaction costs." (SZABO, 1997).

Smart contract searches can be classified into two types: development and evaluation. Development can be the creation of smart contracts or smart contract platform development. Recently, many smart contracts have been implemented on the Ethereum blockchain (WOOD, 2018). Regarding the platform development, many smart contracting platforms like Ethereum (WOOD, 2018) and Hawk (KOSBAA et al., 2016) are emerging (ZHENG et al., 2016). Evaluation means code analysis and performance evaluation. Errors in smart contracting can bring catastrophic damage. Smart contract attack analysis is fundamental. On the other hand, smart contract performance is also vital to the contract. With the rapid development of blockchain technology, more and more smart contract based applications will be used, and companies need to consider the application's performance (ZHENG et al., 2016).

In the blockchain, smart contracts are created as scripts, stored with exclusive addressing on the blockchain itself (GREVE et al., 2018). They are triggered when addressing a transaction to it. Then the script is executed independently and automatically, as prescribed in all nodes in the network according to the data included in the transaction (CHRISTIDIS; DEVETSIKIOTIS, 2016).

### 2.4.1 Smart Contracts Security

As an innovative technology, smart contracts have been applied in various business areas, such as digital asset exchange, supply chains, crowdfunding, and intellectual property. Unfortunately, many security issues in smart contracts have been reported in the media, often leading to substantial financial losses. These security issues pose new challenges

to security research because the execution environment of smart contracts is based on blockchain computing and its decentralized nature of execution. Thus far, many partial solutions have been proposed to address specific aspects of these security issues. The trend is to develop new methods and tools to detect common security vulnerabilities automatically. However, smart contract security is systematic engineering that should be explored from a global perspective, and a comprehensive study of issues in smart contract security is urgently needed (HUANG et al., 2019).

Smart contracts interpret the code objectively - "The Code is the law". However, a code error was the target of a cyberattack, resulting in a deviation of about 50 million dollars, forcing Ethereum to perform a hard fork to perform a recovery (BASHIR, 2018). Performing recoveries like this are not trivial, so the risks must be evaluated and minimized.

Smart contracts should be concerned with blockchain threats:

- State of contract;
- Random generation;
- Time Restrictions.

Regarding the status of the contract, field and balance values determine the smart contract state. A user, when invoking it, may not have certainty under its state, as other transactions may modify it or a fork may have occurred. In some cases, this can create vulnerabilities and lead to asset theft. In a random generation, some contracts generate pseudo-random numbers with the same seed for all miners. This allows everyone to have the same view as blockchain, providing a malicious miner to influence the network. About time restriction, if a miner holds a stake in a contract, he can gain an advantage by choosing an appropriate time stamp for the block he is exploring (GREVE et al., 2018).

## 2.5 SUPPLY CHAIN MANAGEMENT AND BLOCKCHAIN

In order to solve some problems with supply chain visibility and traceability, many Internet of things technologies, such as RFID and wireless sensor network-based architectures and hardware, have been applied. However, these technologies do not guarantee that the information shared by supply chain members in the traceability systems can be trustful. As a centralized organization, it can become a vulnerable target for bribery, and then the whole system can not be trusted anymore (TIAN, 2017).

Suppose companies A, B, and C exercise such roles in the chain: producer, distributor, and retailer. There is data centralization, single point of failure, and trust among the parties in traditional systems. Sometimes, a third-part as a certificate authority must guarantee some regulation, audition, or provide security among the participants. The use of blockchain technology can help solve these problems since blockchain provides better transparency, enhanced security, immutability of data, traceability, and decentralization.

Blockchain and distributed ledger technology underpinning cryptocurrencies such as Bitcoin represent a new and innovative technological approach to realizing decentralized, trustless systems. Indeed, the inherent properties of this digital technology provide



fault-tolerance, immutability, transparency, and full traceability of the stored transaction records, as well as coherent digital representations of physical assets and autonomous transaction executions (CARO et al., 2018).

Blockchain enables end-to-end traceability, bringing a standard technology language to the supply chain while allowing consumers to access the asset's history of these products through a web app. The need to track products across the complex supply chain to the end consumer is often common: checking environmental impacts or simply ensuring transparency for consumers (GALVEZ; MEJUTO; SIMAL-GANDARA, 2018).

Instead of storing data in a shadowy network system, blockchain allows all the goods' information to be stored in a shared and transparent system for all the members along the supply chain (TIAN, 2017). Monfared (ABEYRATNE; MONFARED, 2016) argued about the potential benefit of blockchain technology in the manufacturing supply chain. They proposed that the inherited characteristics of the blockchain enhance trust through transparency and traceability within any transaction of data, goods, and financial resources. Moreover, it could offer an innovative platform for a new decentralized and transparent transaction mechanism in industries and businesses.

Many members are among the supply chain, including suppliers, producers, manufacturers, distributors, retailers, consumers, and certifiers. Each of these members can add, update and check the information about the product on the blockchain as long as they register as a user in the system. Each product also has a unique digital cryptographic identifier that connects the physical items to their virtual identity in the system. Users in the system also have their digital profile, which contains information about their introduction, location, certifications, and association with products (TIAN, 2017).

Supply chain members can register themselves in the system through the register, providing credentials and a unique identity to the members. After registration, public and private cryptographic key pairs will be generated for each user. The public key can be used to identify the user's identity within the system, and the private key can be used to authenticate the user when interacting with the system. This enables that each product can be digitally addressed by the users when being updated, added, or exchanged to the following user in the downstream position of the supply chain (CARO et al., 2018).

All members of the business network agree with the information acquired in each transaction. Once consensus is reached, no permanent record can be changed. Each information provides critical data that can potentially reveal security issues with the product in question (GALVEZ; MEJUTO; SIMAL-GANDARA, 2018).

A smart contract encodes the combination of services and other conditions defined in the contract. Therefore, the smart contract can automatically verify and apply these conditions. It also verifies all information required by regulation to enable automated verification of regulatory compliance (LU; XU, 2017). A smart contract, by default, has no owner. Once deployed, its author has no special privileges. Unauthorized users may accidentally trigger a function without permission. Therefore, smart contracts must have internal permission to verify contract permissions.

The smart contract structural design has a significant cost impact if the blockchain is public. The cost of contract implementation depends on its size because the code is stored in the blockchain, which implies data storage fees proportional to the size of the

contract. Therefore, a structural design with more lines of code costs more. A blockchain consortium does not have a coin or token, so the monetary cost is not a problem. However, blockchain size is still a design concern because it grows with each transaction, and each participant has a replica of the entire blockchain. In addition, a more structural design may affect performance as it may require more transactions (LU; XU, 2017).

A blockchain must be universal and adaptable to specific situations (VALENTA; SANDNER, 2017). The need to agree on a particular type of blockchain to be used puts the parties under pressure. This is a significant disadvantage as blockchain technology is progressing rapidly, and predicting the best choice for the future is quite tricky (GALVEZ; MEJUTO; SIMAL-GANDARA, 2018).

On the other hand, there are advantages to applying the blockchain concept to the enterprise supply chain. One of the most important is that all stakeholders involved in the supply chain (raw material / producer, manufacturer, distributor, wholesaler, retailer) are motivated by the need to demonstrate to customers the superior quality of their methods and products (LU; XU, 2017).

In addition to serving the functions of a traceability system, a blockchain can be used as a marketing tool. Because blockchains are fully transparent (IANSITI; LAKHANI, 2017) and participants can control the assets in them (LIAO; CHANG; CHANG, 2011), they can be used to enhance the image and reputation of a company (RIEL; FOMBRUN, 2007), drive loyalty among existing customers (PIZZUTI; MIRABELLI, 2015) and attract new ones (SVENSSON, 2009).

Companies can easily distinguish themselves from competitors by emphasizing transparency and monitoring product flow along the chain. In addition, quickly identifying a source of contamination or loss can help protect a company's brand image (MEJIA et al., 2010) and alleviate the adverse impact of media criticism (DABBENE; GAY, 2011).

### **2.5.1 Transparency**

The main goals of a blockchain are to facilitate information exchange, create a digital twin of information and its workflow, and validate the quality of assets as they move along the chain. These goals are achieved by allowing each participant to share claims, evidence, and assessments of each other's claims about the product. The journey of mineral resources along the supply chain is captured in a blockchain object called a "mineral bundle". At the journey's end, the package combines all information provided by stakeholders over the life of the mineral item. This information can be used to establish the provenance, quality, sustainability, and many other attributes of mineral assets (MARTIN; LEURENT, 2017).

### **2.5.2 Efficiency**

Blockchain is an infrastructure that allows new transactions between players who do not yet know or trust each other. Smart contracts are instructions that interface with the blockchain protocol to automatically evaluate and possibly post transactions on the blockchain (RASKIN, 2017).

Similarly, smart libraries are specialized sets of blockchain-compatible functionality

that can be used locally or privately or shared and licensed to other blockchain participants and agents. All participants meet at the blockchain, evaluate the statements made and notify their account holders when matches are found in quality, time, quantity, etc. Buyers and sellers are matched by a shared but reliable need for data combined and used by either party. So traceability does not have to wait for large company consortia to use patterns and semi-mandatory or concentrated business practices to access the information (GALVEZ; MEJUTO; SIMAL-GANDARA, 2018).

### 2.5.3 Safety and protection

Blockchains can also be used to emit and manage the creation of unique cryptographic tokens (NYSTRÖM, 1999). Tokens can be made to represent the collateral value between two participants (for example, future production to be sent in a specific field lot). Tokens do not have to take value exchange for the financial settlement of invoices and contracts. Instead, they represent a license to publish information that becomes uniquely valued in proportion to the needs of others on the blockchain. The strategy around issuing these encryption tokens, which need not be implemented in the initial system, is still being defined (GALVEZ; MEJUTO; SIMAL-GANDARA, 2018).

## 2.6 HYPERLEDGER FABRIC

As the popularity of public blockchain and a few other derivative technologies grew, interest in applying the underlying technology of the blockchain, distributed ledger, and distributed application platform to more innovative enterprise use cases also grew. However, many enterprise use cases require performance characteristics that permissionless blockchain technologies are unable (presently) to deliver. In addition, in many use cases, the identity of the participants is a hard requirement, such as in the case of financial transactions where Know-Your-Customer (KYC) and Anti-Money Laundering (AML) regulations must be followed (POLGE; ROBERT; Le Traon, 2020).

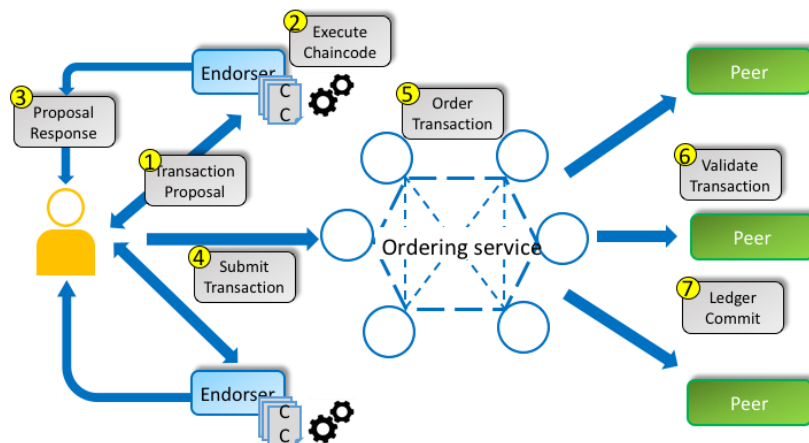
For enterprise use, it is necessary to consider the following requirements:

- Participants must be identified/identifiable;
- Networks need to be permissioned;
- High transaction throughput performance;
- Low latency of transaction confirmation;
- Privacy and confidentiality of transactions and data of business transactions.

These requirements are a good fit with the required non-functional requirements for an SCM project and the ones specified in Table A.2. While many early blockchain platforms are currently being adapted for enterprise use, Hyperledger Fabric has been designed for enterprise use from the outset.

The Hyperledger project is a collaborative effort to create an enterprise-grade, open-source distributed ledger framework, and codebase. It aims to advance blockchain technology by identifying and realizing a cross-industry open standard platform for distributed ledgers, transforming how business transactions are conducted globally (CACHIN et al., 2016).

Hyperledger Fabric implements a distributed ledger platform for running smart contracts, leveraging familiar and proven technologies, with a modular architecture allowing pluggable implementations of various functions (CACHIN et al., 2016). Hyperledger Fabric is a widely used permissioned blockchain-primarily in enterprise settings to make transactions between multiple businesses more efficient. This implementation defines assets and the transaction instructions. Members of each permissioned network interact with the ledger using chaincode. The membership identity service manages IDs and authenticates participants. Also, the Access Control List provides additional layers of permission (BLOCKGEEKS, 2018).



**Figure 2.4** Hyperledger Fabric transaction flow (MANEVICH; BARGER; TOCK, 2018).

There are two types of nodes in Hyperledger Fabric: peer nodes and ordering nodes. Peer nodes are responsible for executing and verifying transactions while ordering nodes are responsible for ordering transactions and propagating the correct history of events to the network. This increases efficiency and scalability by allowing peer nodes to batch and process multiple transactions (BUTERIN, 2016).

Fabric Ledger comprises two components: a blockchain log to stores the immutable sequenced record of transactions in blocks and a State Database to maintain the blockchain's current state. In a public blockchain, there is no state database. It means that the chain's current state is always calculated by going through all the transactions in the ledger. For speed and efficiency, the Fabric stores the current state and allows network members to query it as a SQL-like transaction (BLOCKGEEKS, 2016).

Figure 2.4 shows the Hyperledger Fabric transaction flow. The client proposes a transaction to the endorsing peers and collects transaction responses. The client then

submits a transaction to the ordering service, which orders incoming transactions and cuts them into blocks. Peers pull blocks from the ordering service, validate the transactions, append them to the ledger, and apply valid transactions to the state.

The log aims to trap an asset's provenance or place of origin as it exchanges among multiple parties. To track an asset's provenance means to track where and when it was created, and every time it was exchanged. Tracking an asset's provenance is extremely important in business because it ensures that the business selling an item possesses a chain of titles verifying their ownership of it. In typical databases, where only the current state is kept and not a log of all transactions, tracking an asset's provenance becomes very difficult. Add to this the fact that transacting businesses each keep an incomplete record of the asset transaction, and it becomes nearly impossible (BLOCKGEEKS, 2016).

Fabric uses private channels to solve the problem of sensitive data which other parties or competitors could see in a public blockchain. Private channels are restricted message paths that can provide transaction privacy and confidentiality for a specific subset of network members. All data are invisible and inaccessible to any network members not explicitly granted access to that channel. This allows competing business interests and any groups that require private, confidential transactions to coexist on the same permissions network (BRABBANI, 2017).

*This section presents some conventional and Blockchain-based SCM systems, their main characteristics, and how this work presents a different approach.*

## RELATED WORK

The rapid growth of Internet technologies allowed the onset of lots of technologies applied in traceability systems. In order to solve some problems with Supply Chain traceability, many Internet of Things (IoT) technologies, such as Radio frequency identification (RFID) and wireless sensor network-based architectures, have been applied. However, these technologies do not guarantee that the information shared by supply chain members in the traceability systems can be trusted (TIAN, 2017).

Blockchain presents a whole new approach based on decentralization. Nonetheless, being in its early stages has some challenges to deal with, in which scalability and performance become mainly defiance to face the massive amount of data in the real world. Through this technology, some solutions have been raised, as follows.

### 3.1 TRADITIONAL SYSTEMS

Microsoft's Dynamics 365 is excellent for simple SCM needs, and it is just as accessible as every other Microsoft suite on the market. Dynamics integrates with third-party management systems, but it is primarily for "simple needs." Microsoft's offering is not so great for complex supply chain demands - but then, the more significant majority of organizations have not got overly complex networks (BELLU, 2018).

Plex Systems was one of the very first supply chain and manufacturing Software-as-a-Service (SaaS) ERP systems. The software is a cloud-based SCM that is very popular with industry-leading companies - especially in the aerospace and automotive industries. Unfortunately, though, for all its maturity and complex capabilities, Plex lets organizations down with its inability to support several implementation partners (PLEX, 2020).

Oracle NetSuite is a cloud-based supply chain and ERP system for the less complex needs of moderately-sized companies and most SCM and ERP systems (ROLLING, 2016). As ERP focused system, this project is focused on business management instead of SCM traceability and information transparency (ROLLING, 2016).

SAP Supply Chain Management harnesses AI and the Internet of Things to provide visibility and analytics to help the users plan, source, and deliver the goods and materials. This is a real-time supply chain planning software that connects stakeholders (SNAPP, 2010).

These systems typically are ERP solutions focused on business management, controlled by service providers. This centralized data storage becomes a single point of failure and risks tampering. As a centralized organization, it can become a vulnerable target for bribery, and then the whole system can not be trusted anymore. Also, as proprietary systems, there are some concerns about reliability, security, decentralization, immutability, transparency, and lack of trust among participants. There is no trusted third party to ensure data reliability.

### **3.2 BLOCKCHAIN-BASED SYSTEMS**

There are advantages of applying the Blockchain concept to a supply chain. One of the most important is that all stakeholders involved in the supply chain are motivated to demonstrate to customers the superior quality of their methods and products (LU; XU, 2017). In addition, a Blockchain can be used as a marketing tool. As Blockchains are fully transparent and participants can control the assets in them, they can be used to enhance the image and reputation of a company (RIEL; FOMBRUN, 2007), drive loyalty among existing customers (PIZZUTI; MIRABELLI, 2015), and attract new ones (SVENSSON, 2009). In fact, companies can easily distinguish themselves from competitors by emphasizing transparency and monitoring product flow along the chain.

In (TIAN, 2017), it is proposed a system that combines HACCP (a food safety protocol), Blockchain, and IoT in order to provide food safety traceability. Each member can add, update and check the information about the product on the Blockchain as long as they register as a user in the system. Each product also has a unique digital cryptographic identifier that connects the physical items to their virtual identity in the system. This virtual identity can be seen as a product information profile.

The Everledger Diamonds project provides a Blockchain-based solution to facilitate tracking from mine to consumer, enabling easier compliance against increasingly strict measures from diamonds produced (CROSBY et al., 2016).

IBM Food Trust is a pilot project motivated by food contamination scandals worldwide. The main goal is to tackling food safety in the supply chain using Blockchain technology. This platform tracked pork in China and mangoes in the Americas (KAMATH, 2018).

### **3.3 COMPARISON WITH THE PRESENTED WORK**

These projects are focused on specific products only and are closed projects. Still, there is a general lack of standards for implementing a Blockchain approach for traceability. A Blockchain must be universal and adaptable to specific situations (VALENTA; SANDNER, 2017). In addition, the need to agree on a particular type of Blockchain to be used puts the parties under pressure.

Compared to traditional systems, the great difference of this work is to provide the non-functional requirements acquired by using the blockchain:

- More reliable operations;
- More democratic transactions;
- Process optimization;
- Ease of coordination between companies;
- Recording data in chronological order;
- Cost reduction;
- Distributed and autonomous platform.

In relation to blockchain-based systems:

- Not specific to a product type;
- Open source project and not closed;

Our work is focus on providing a Blockchain-based platform to facilitate the development of applications for traceability in supply chain management.



**Table 3.1** Related Works and their main characteristics, strategies and results.

<b>Related work</b>	<b>Blockchain</b>	<b>Solution</b>	<b>Main Deficiency</b>
Microsoft Dynamics 365	No	Simple SCM needs	Centralized solution. Not a distributed and autonomous platform.
Plex System	No	SaaS ERP system	Centralized solution. Not easy to coordinate between companies. High cost. Private System.
Oracle NetSuite	No	cloud-based supply chain and ERP system	Centralized solution. High cost. Private System.
SAP SCM	No	AI and the Internet of Things	High cost. Private System.
Tian	Yes	Oroposed system that combines HACCP, Blockchain, and IoT	Theoretical application. Food related only.
Everledger Diamonds	Yes	Blockchain-based solution	Product specific. Not an open source project. High cost.
IBM Food Trust	Yes	Blockchain-based solution	Product specific. Not an open source project. High cost.
SCM-BP	Yes	Blockchain-based solution	Relies on a minimum amount of participation to obtain a reliable forecast.

## PROPOSED SOLUTION

Supply Chain Management - Blockchain Platform (SCM-BP) has the general objective to create a generic open source framework aiming at to be used in any supply chain correlated to assets and products using a stateless microservices architecture.

A SCM platform relies on three main items: assets (the goods themselves), steps (phases which products go through), and actors (people who transact assets during the steps). Our approach is based on this triad that must be defined to create a new supply chain. The workflow is divided into two phases: design time and execution time. Design time is when an administrator configures the asset flow: define asset info, steps, actors and create the relationship between these entities. Execution time is composed of creating, moving, and tracking asset items through the supply chain.

In the design time, initially, a configuration file in JSON format is generated and read in the Blockchain platform, adding the primary information for the correct functioning of the chain. The mechanism for creating this configuration file is detailed in the following section.

### 4.1 APPLICATION ARCHITECTURE

The main objective of this work is to create a generic platform for Supply Chain Management (SCM). SCM-BP is divided into four main modules described below: WebApp - frontend, WebApp - backend, Blockchain network, and Data Storage. Figure 4.1 shows the application architecture and its components. The WebApp - frontend is the component which interacts with the user by providing web pages. WebApp - backend is the component that works like a middleware by receiving REST calls from the front end, and interacting with the lower layers. The Data Storage module main responsibility is to storage content specific data and also files updated by the user. The Hyperledger Platform Service is the Blockchain network where all the transactions are stored and the chaincode is executed. Figure C.1 presents the main data structure.

Table A.1 defines the main functional requirements in order to be able to meet the objectives proposed by this project. They are characterized as non-functional requirements for the proper functioning of the items in table A.2.

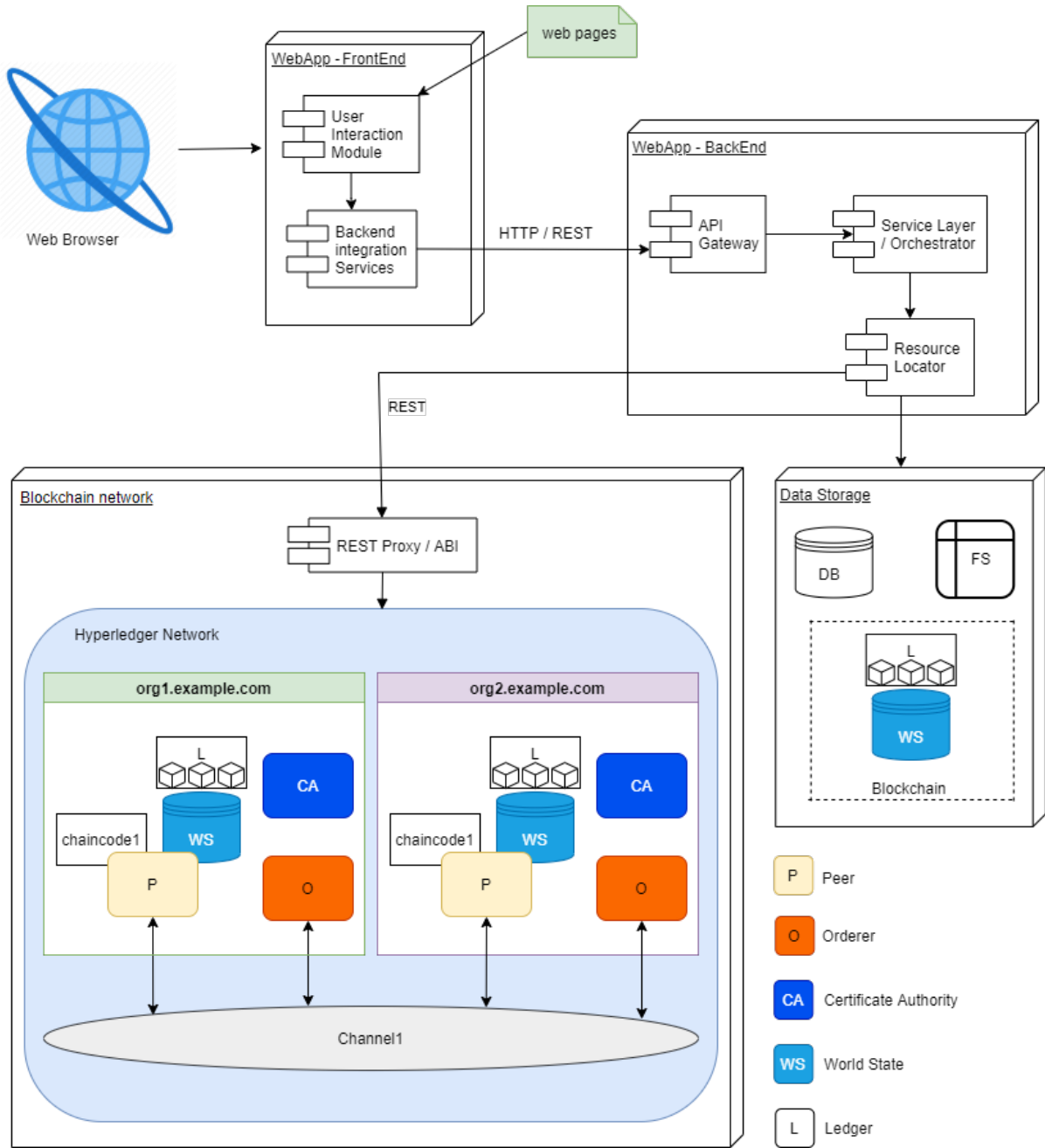


Figure 4.1 Application architecture of SCM-BP

### 4.1.1 WebApp - FrontEnd

WebApp - FrontEnd is a client-server computer application that the client (including the user interface and client-side logic) runs in a web browser. This is a Single-page application (SPA), a web application that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server. This approach avoids interruption of the user experience between successive pages, making the application behave more like a desktop application.

The application is built with Angular, a JavaScript library for building user interfaces. It is a TypeScript-based open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. Used as a base in developing of single-page or mobile applications, Angular is optimal for fetching rapidly changing data that needs to be recorded. However, fetching data is only the beginning of what happens on a web page, which is why complex Angular applications usually require additional libraries for state management, routing, and interaction with an API.

The Webapp - FrontEnd is divided into two main blocks, and these are classified according to the interactions: User Interaction Modules and Backend Interactions Services.

**4.1.1.1 User Interaction** The User Interaction modules are responsible for providing web pages that will be rendered on the client's web browser. These interactions are provided by web pages grouped by the following components:

- Login page;
- Application configuration module;
- User handling module (actors - CRUD);
- Data entry module (forms);
- Data visualization module;
- Reporting module.

The Login Module is responsible displaying the login and authentication alternatives pages (e.g. 'forgot my password', 'reset my password'). The Application Configuration module provides the features of the creation/configuration of supply chain items and supply chain flows (steps). This module is responsible for getting the information from the user to generate the configuration JSON file in the backend. The User handling module provides the features for the creation/configuration of Actors and Steps, complementing the configuration file. The Data Entry module provides form pages that allow the actors to enter data in the application, search and move asset items from a step to another. The Data Visualization module is responsible for displaying the information about asset items in the supply chain flow through steps. In the Reporting module, users can generate reports/files organized in a narrative, graphic, or tabular form, prepared on ad hoc, periodic, recurring, regular, or as required. Reports may refer to specific periods, events, occurrences, or subjects presented in written form or any other format.

**4.1.1.2 Backend Integration** Backend interactions happen via a service layer consisting of:

- Authentication service;
- Application setup service;
- User creation service (actors);
- Data entry service (forms);
- Data visualization service;
- Reporting service.

The Authentication Service role is to request information from an authenticating party and validate it against the configured identity repository using the specified authentication module. After successful authentication, the user session is activated and validated across all web applications participating in an Single Sign-on (SSO) environment. For example, when a user or application attempts to access a protected resource, credentials are requested by one (or more) authentication modules. Gaining access to the resource requires that the user or application be allowed based on the submitted credentials.

Application setup service provides methods to configure and edit supply chain items, and supply chain flows, defining which steps and sub-tasks will be present in this flow and which information will be present in these steps.

The User creation Service is responsible for creating users and roles to log in and use the application's features. Only administrators are allowed to create new users (see Actions and Actors). The Data entry service receives data from UI forms and sends them to the backend to be processed and stored. The Data visualization services provide information about the supply chain: Assets, users, and transactions, to be used by the data visualization module. The Report services generate files (Doc/PDF/XSL, etc...) from a specific period with information about the supply chain: Assets, users, and transactions.

## **4.1.2 WebApp - BackEnd**

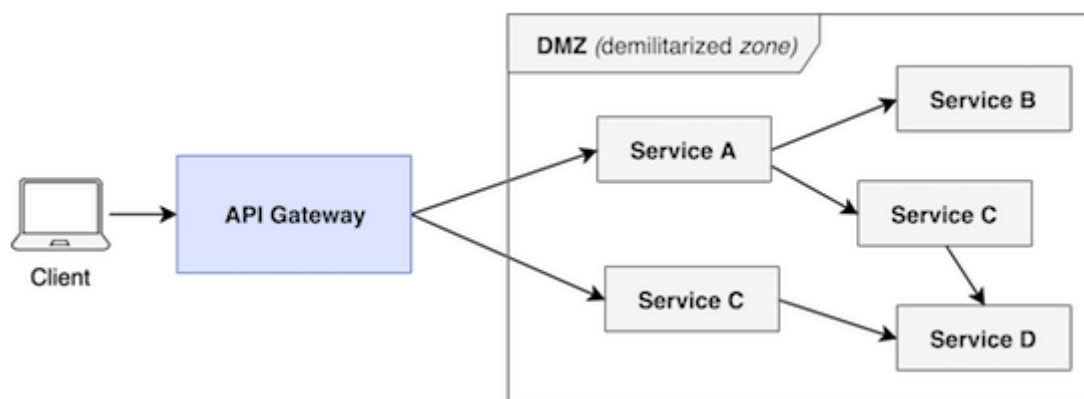
WebApp - BackEnd is a Middleware that runs on the server. This Middleware (server-side software) facilitates client-server connectivity, forming a middle layer between the app(s) and the network: the server, the database, the operating system, and more. It receives requests from the clients (in this case, the WebApp - FrontEnd) and contains the logic to send the appropriate data back to the applicant over HTTP and REST by providing a standard way to interact with the front end and in the future with external applications. These are the main conventions that provide structure to the request-response cycle between clients and servers.

WebApp - BackEnd is an application built with Node.js, an application platform where developers can write Javascript programs compiled, optimized, and interpreted by the V8 virtual machine. Node.js can create quick, reliable websites and products in a much

efficient manner. Developing easy-to-scale real-time applications in other technologies is a bit difficult, but JavaScript technologies made it more accessible.

The WebApp - BackEnd is composed of the API Gateway, Service Layer, and Resource Locator, more detailed below.

**4.1.2.1 API Gateway** API Gateway is a managed service that enables easy creation, publish, maintain, monitor, and secure REST APIs to act as a "gateway" for applications to access data, business logic, or functionality in the backend services, such as workloads. The API Gateway provides a simple uniform view of external resources to the internals of this application. It manages all tasks involved in receiving and processing API calls, including traffic management, authorization and access control, and monitoring and managing API versions.



**Figure 4.2** API Gateway.

Basically, the gateway is an interface that receives calls to its internal systems, being a large gateway. It acts in five different ways:

- Filter for call traffic from different media;
- A single gateway to the various API's that are exposed;
- Router: API and Rate Limit traffic router;
- Security engine with authentication and logging.

Gateway access can be done from many different devices. Therefore, it must have the power to unify outgoing calls and deliver to the user content that can be accessed from any browser and system. In this project, the gateway interaction happens with the frontend web app. The Gateways as a Security Feature: In the APIs world, one of the most subjects talked about issues is always security, and having an API Gateway is one of the best solutions on the market to get complete control of API's, because this pattern addresses the so-called CIA (Confidentiality, Integrity, Availability) almost flawlessly.

**4.1.2.2 Service Layer** A Service Layer defines an application's boundary and its set of available operations from the perspective of interfacing client layers. It encapsulates the application's business logic, controlling transactions and coordinating responses in the implementation of its operations.

Enterprise applications typically require different interfaces to the data they store and the logic they implement: data loaders, user interfaces, integration gateways, and others. Despite their different purposes, these interfaces often need common interactions with the application to access and manipulate its data and invoke its business logic. The interactions may be complex, involving transactions across multiple resources and coordinating several responses to an action. Encoding the logic of the interactions separately in each interface causes much duplication. The service layer:

1. Centralizes external access to data and functions;
2. Hides (abstracts) internal implementation and changes;
3. Allows for versioning of the services.

The service layer acts as an orchestrator, controlling the flow of incoming and outgoing information requests and responses. Orchestration allows to directly link process logic to service interaction within workflow logic. This combines a business process model with service-oriented modeling and design, realizing workflow management through a process service model. Orchestration brings the business process into the service layer, positioning it as a master composition controller.

**4.1.2.3 Resource Locator** Resource locators are components that abstract the persistence layer. Their job is to provide an object that can help services discover and persist information from/to the Data Storage Module. Information can be stored in the Blockchain, Filesystem, or Database, and resource locators should know precisely where to get/put data within them.

### **4.1.3 Blockchain**

SCM-BP uses blockchain as a supply chain that tracks parts and service provenance, ensures the authenticity of goods, blocks counterfeits, and reduces conflicts. In SCM-BP, the Blockchain module consists of a smart contract, chaincode, and ledger. From the application developer's perspective, a smart contract and the ledger form the heart of a Hyperledger Fabric blockchain system. Whereas a ledger holds facts about the current and historical state of a set of business objects, a smart contract defines the executable logic that generates new facts added to the ledger. Based on the discussion in Section 2.3 and to achieve the non-functional requirements exposed in Chapter A.3, the permissioned blockchain was chosen and Hyperledger Fabric as its implementation.

**4.1.3.1 Smart contract** Before businesses can transact with each other, they must define a common set of contracts covering common terms, data, rules, concept definitions,

and processes. Together, these contracts lay out the business model that governs all of the interactions between transacting parties.

A smart contract defines the rules between different organizations in executable code. Applications invoke a smart contract to generate transactions that are recorded on the ledger.

**4.1.3.2 Chaincode** Hyperledger Fabric users often use the terms smart contract and chaincode interchangeably. In general, a smart contract defines the transaction logic that controls the lifecycle of a business object contained in the world state. It is then packaged into a chaincode which is then deployed to a blockchain network. Think of smart contracts as governing transactions, whereas chaincode governs how smart contracts are packaged for deployment.

**4.1.3.3 Ledger** At the simplest level, a blockchain immutably records transactions that update states in a ledger. A smart contract programmatically accesses two distinct pieces of the ledger: a blockchain, which immutably records the history of all transactions and a world state that holds a cache of the current value of these states, as it's the current value of an object that is usually required.

The ledger is the sequenced, tamper-resistant record of all state transitions in the fabric. State transitions are a result of chaincode invocations ('transactions') submitted by participating parties. Each transaction results in a set of asset key-value pairs committed to the ledger as creates, updates, or deletes. The ledger comprises a blockchain ('chain') to store the immutable, sequenced record in blocks, as well as a state database to maintain the current fabric state. There is one ledger per channel. Each peer maintains a copy of the ledger for each channel of which they are a member.

Smart contracts primarily put, get, and delete states in the world state and query the immutable blockchain record of transactions.

- A **get** typically represents a query to retrieve information about the current state of a business object.
- A **put** typically creates a new business object or modifies an existing one in the ledger world state.
- A **delete** typically represents the removal of a business object from the current state of the ledger but not its history.

Smart contracts have many APIs available to them. Critically, in all cases, whether transactions create, read, update or delete business objects in the world state, the blockchain contains an immutable record of these changes.

#### 4.1.4 Data Storage

Data storage is a general term for archiving data in electromagnetic or other forms for use by a computer or device. Different types of data storage play different roles in a computing



environment. In addition to forms of hard data storage, there are now new options for remote data storage, such as cloud computing and blockchain, that can revolutionize how users save and access data.

SCM-BP uses three applications as data storage: Blockchain, Cloud filesystem, and relational database, which are better detailed in the following subsections. Blockchains grow continuously because of the amount of data and code in them, which is unchanging. Therefore, an important design decision is to choose which data and calculations to keep in and out of the chain.

**4.1.4.1 Filesystem** A cloud file system is a tiered storage system that provides shared access to file data. Users can create, delete, modify, read and write files and logically organize them into directory trees for intuitive access.

Cloud file-sharing can be defined as a service that gives multiple users simultaneous access to a cloud file data set. Cloud file sharing security is managed with the user and group permissions, allowing administrators to control access to shared file data tightly.

For all files uploaded and stored in the filesystem, a locally stored digital fingerprint (hash) is saved in the blockchain, separately from the original files or content, to make it easier to confirm whether data has been altered or manipulated in a particular organization.

**4.1.4.2 Database** A relational database is a set of formally described tables from which data can be accessed or reassembled in many different ways without reorganizing the database tables. The standard user and Application programming interface (API) of a relational database is the Structured Query Language (SQL). SQL statements are used for interactive queries for information from a relational database and for gathering data for reports.

**4.1.4.3 Blockchain** Since the blockchain consists of a Ledger and a world state database (among other components), this can also be seen as part of the data storage module as it stores data. Component 4.1.2.3 has the intelligence to decide where to search and store data to optimize storage consumption.

## 4.2 ACTIONS AND ACTORS

A set of rules governs the system. These rules define how users interact with the system and how the data is shared among the users. Moreover, once the rules are stored in the blockchain, they can not be altered without broadcasting to all nodes and verified by most of them.

### 4.2.1 Setup

Setup is the set of actions to configure the application. The setup phase is when a new supply chain is created, or an existing one is updated or deleted. Users can also be created, updated, and deleted by setup actions. When creating or editing a supply chain,

Admin users will define which steps, sub-steps, and information the supply chain flow will contain. Users from member groups can add info and move an asset to each step in the logistics network. Administrators are the only users in the Admin group. This actor type has access to all areas of the program. It has the same abilities like all other user types (configuring, moving the asset, and viewing flow). However, his primary responsibility is to configure the application and perform the setup actions.

#### 4.2.2 Data Insertion

Data insertion is the action that will fill the supply chain flow with data. Once the admin user creates a new supply chain, it is ready to be populated with information. Member and admin users are responsible for performing these actions. In the Data Insertion phase, users can update information from a specific step and sub-step and move assets depending on the rules applied in the setup phase. The most common actor types in SCM are Raw material/Producer, Manufacturer, Distributor Wholesaler, and Retailer.

#### 4.2.3 Visualization

Any user in the system can perform visualization actions. However, its main purpose is to provide the end-user the capability to track the flow of an asset from the point of origin to the point of consumption.

### 4.3 IMPLEMENTATION DETAILS

The chaincode is written in Golang and provides all contracts needed to proceed with traceability in the application. All contracts for use in chaincode must implement the interface *contractapi.ContractInterface*.

The first step is to create a JSON config file providing all information about these three items. A configuration file includes *assetId*, a list of actors, and a list of ordered steps. The chaincode processes this file through *initLedger* and *createNewAsset* functions. A template for the config file can be found in appendix B.1.

Front-end WebApp enables users to define settings through a Configuration Page, adding these to the configuration file, as shown in Figure 4.7. Assets, asset items, steps, and actors are described in appendix B.2. There are create methods for each one, responsible for creating an instance of these *structs* and save the state into the blockchain. Query methods are responsible for interact with the information of any item in the blockchain.

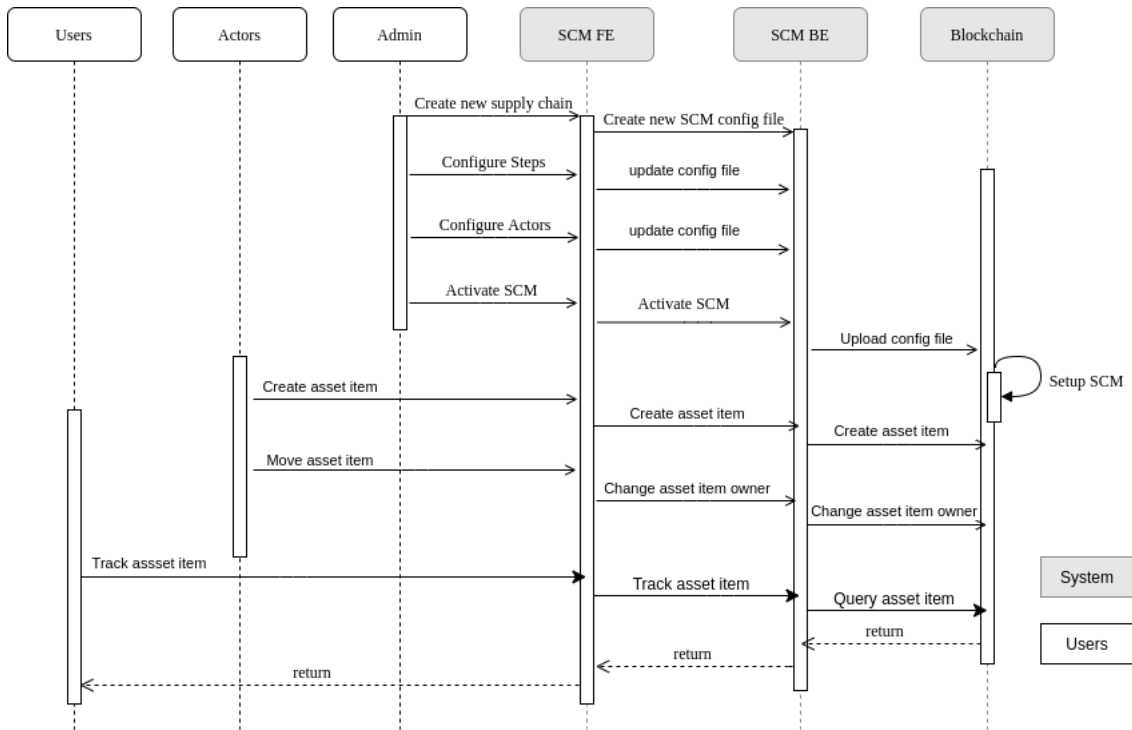
The chaincode *main* function invokes the *initLedger* function, reads the configuration files, and raises the platform enabling users to interact with the blockchain via exposing its API.

When creating an asset item, an *AssetItemId* is generated. Each entity in the chain will have its unique entity ID and timestamp when processing the transaction. By querying *AssetItemId*, the user can easily track the current transaction information and status. Finally, completed all steps, the blockchain will update *deliverDate* and mark the status as completed once the last actor (generally the consumer) has received the order. The *CreateAsset* function is detailed in appendix B.4.

*MoveAssetItem* is the method to update an asset item when moved from one step to another. It updates the *CurrentOwnerId*, the *ProcessDate*, information about prices, and many other details of the transactions by the key/value map *additionalInfo*. Its details are in appendix B.5.

*TrackAssetItem* is the method responsible for tracking an asset item. It returns the children’s tree of the given element and its ancestors from the beginning of the Supply chain. This method uses the function *getChildrenTree* to get the current item’s children nodes. Appendix B.6 contains its implementation.

For audit purposes, some methods were implemented to get info about the blocks and transactions stored in the ledger. These are the methods: *getBlockByTxID*, *getChainInfo*, *getBlockByHash*, *getBlockByNumber*, and *getTransactionByID*. Appendix B.7 contains their implementation.



**Figure 4.3** SCM User flow

Figure 4.3 shows the interaction flow from users with the Arion platform. Initially, an admin persona creates and configure the SCM, adding information about the steps and the users. After that, the admin can activate this SCM, and from that point, the actors can interact with the SCM to provide information about an asset item and also move this asset item through the supply chain. From that point, any user can track an asset item to get information about the required goods.

The backend gateway exposes all the endpoints shown in table B.8. These endpoints are integrated with the frontend and can be used in future work to integrate with a mobile app or an external application.

## 4.4 PROOF OF CONCEPT

For this project development, the agile Scrum method has been used. In Scrum, projects are divided into cycles called sprints, with frequent meetings where the team can inform what is being done and think of ways to improve the process quickly. Scrum proposes constant project monitoring. Often the team will be meeting, exchanging experiences, evaluating what has been done, and re-planning what will be done next.

During the requirements gathering, developers and other stakeholders sought to raise and prioritize the needs of future software users (referred to as requirements). After the requirements gathering, in the requirements specification stage, developers made a detailed study of data collected in the previous activity, from where models were built to represent the software system being developed.

At the architectural design stage of the system, two basic activities were performed: architectural design (or high-level design) and detailed design (or low-level design). Some aspects were considered at this stage of system design, such as system architecture, the platform used, Database Manager System (DBMS) used, and graphical interface standard.

In the application development period, the backend and frontend components were created from the computational description of the design phase. Pre-existing software tools and class libraries were used to streamline activity. These tools and libraries were defined during the architectural design and were referenced in sections 4.1.1 and 4.1.2.

For system validation, two main requirements were evaluated: the components and the behavior of who will use the application. For the first point, functional, integration, and security tests will be performed. For the second, this proof of concept was applied.

Appendix A presents the activities for project management, the user stories, and non-functional requirements.

## 4.5 USE EXAMPLE

To accomplish this usage example, one Coffee supply chain will be defined and configured accordingly. The supply chain of coffee beans is a lengthy process that involves growing the beans, harvesting, hulling, drying, packing, bulking, blending, and finally roasting. In between this process, the beans go through international transporters, export sellers, and retailers like grocery stores, cafes, and specialty shops. A coffee tree can take four to seven years before it yields its first crop of beans. The harvesting process is a very labor-intensive exercise. Parts of the cherry must be removed to access the beans and need to be laid out to dry. Once the beans are dried, they are packaged into large sacks and passed onto the exporters. They are distributed to big companies in the coffee business who take these beans and put them in industrial roasting and distribution centers. The inventory stock from the roasting and distributing centers must be passed forward to retailers. Through a web of transport, these coffee beans are delivered to thousands of roasters, cafes, restaurants, grocery stores, and large chain retailers, where they finally will come to the final users who will taste its flavor.

For this scenario, five steps will be taken: Extraction is the step where the coffee

is growing and picking. Processing is when the beans are dry, roasting, grinding, and packaging. Distribution is when the packages are shipping. Retail is about selling the product, and the final step is the final user consumption.

Five fictitious business enterprises and a final customer are named as follows: Tasty Coffee Farm, situated in Brazil, is an extractor responsible for all the information regarding the extraction step in the supply chain. Café Brazilium is the manufacturer and is responsible for the processing step. There will be two distribution companies. The first is Edgard Cargo which is the Distributor company and is in charge of distribution details. This company will send the coffee produced in Brazil to the United States. The second is O'Neil DistInc, a competitor of the first one. Marques BigSales is a United States situated Delicatessen store and is at the helm of selling details. Allan Manoel Jr. is a customer who wants to experiment with Brazilian coffee for the first time. Being very curious, the customer would like to know the provenance and the information about the coffee he drinks and where the coffee has passed.

As actors, each company above will have one user configured in the platform:

- Extractor: James Johnson.
- Manufacturer: Donald Jackson.
- Distribution: Elizabeth Taylor.
- Distribution: Charlotte O'Neil.
- Retailer: BigSales.
- Customer: Allan Manoel Jr.

The first step when creating a new Supply Chain is to create and configure the asset. The admin user makes this action. By going through the setup wizard, first, the asset's info is requested:

The screenshot shows a 'Create Asset' wizard interface. At the top, there's a search bar and user profile icons. Below that, a progress bar indicates four steps: 1. Asset, 2. Actors, 3. Steps, and 4. Summary. The first step, 'Asset', is currently active and highlighted in blue. Underneath, there's a section titled 'Fill asset info' with three input fields: 'Asset ID' (with a note that a new ID will be generated if not provided), 'Asset Name' (filled with 'Coffee'), and 'Description' (filled with 'Roasted coffee beans SCM'). At the bottom right, there are 'CANCEL' and 'NEXT >' buttons.

**Figure 4.4** Fill in asset info.

Then the admin will add actors to the SCM, informing its types. Actors can be added, updated, or deleted later on the actors' list page.

**Figure 4.5** Adding actors.

The next phase in the wizard is to define the Supply chain steps, specifying the order and binding it to the previous created actors' types:

**Figure 4.6** Defining steps.

The final step, before submitting the form, is to review all the information previously added in the review asset details page, under the wizard:

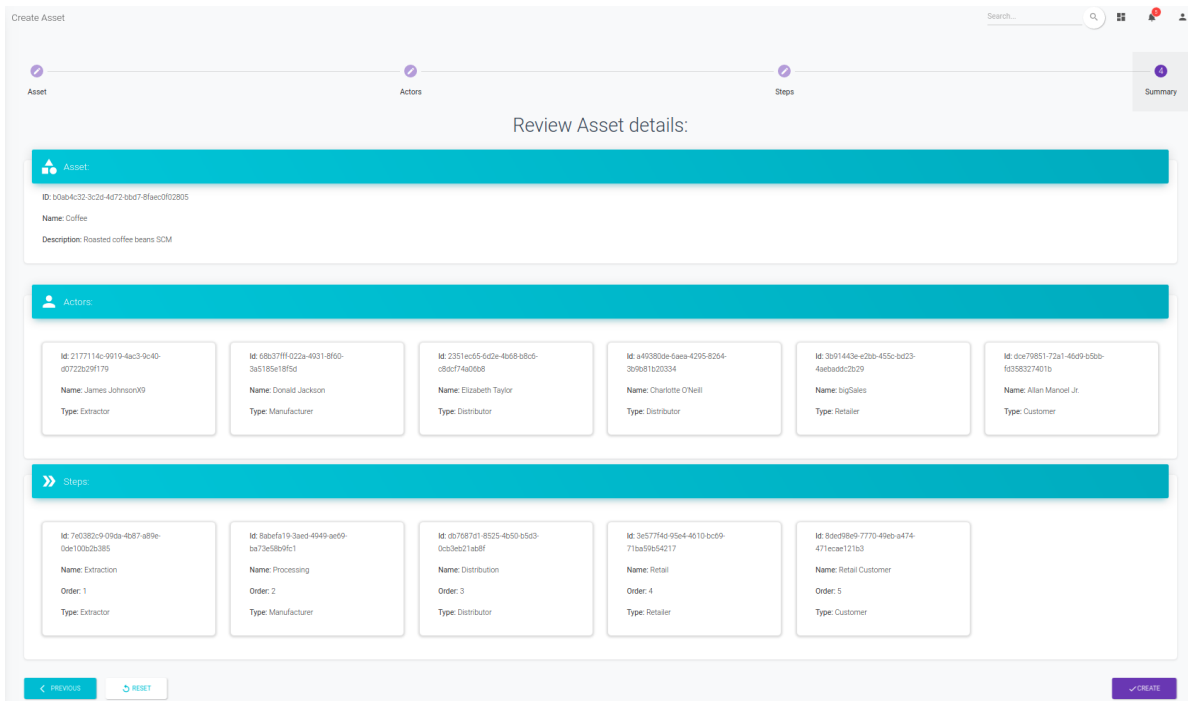


Figure 4.7 Review asset details before submitting.

Once created, the asset can be seen in the assets list, where the admin can perform crud operations by the actions items. When clicking in the asset details, the current user is redirected to the asset details page, where the main information about asset items.





















ID	Owner	Step	Process Date	Status	Quantity	Actions
a544bb43-61c6-48e9-9908-3edc17837dc3	James JohnsonX9	Extraction	2021-04-11T21:49:49	Sold	4	    
600c17ca-9b87-4769-9038-d9899989ab42	James JohnsonX9	Extraction	2021-04-11T21:49:58	Order initiated	2	    
10043c19-baab-4083-bf52-bcb1e0009475	Donald Jackson	Processing	2021-04-11T21:51:26	Sold	2	    
ff446b7b-3709-41f2-9839-e6abe6912158	Donald Jackson	Processing	2021-04-11T21:51:38	Sold	2	    

Figure 4.8 Asset Items list.

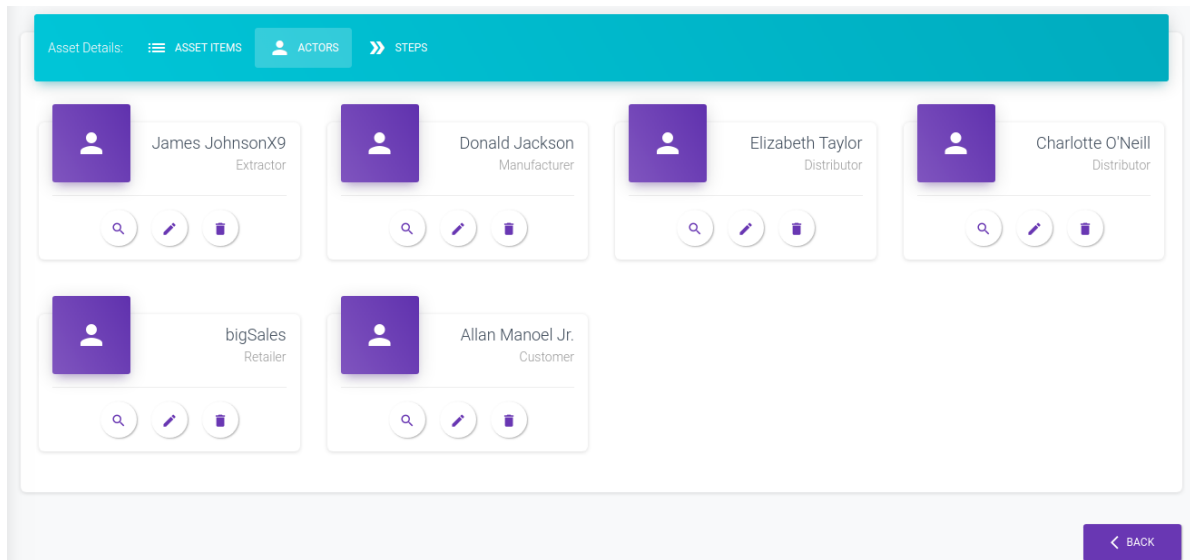


Figure 4.9 Actors list.

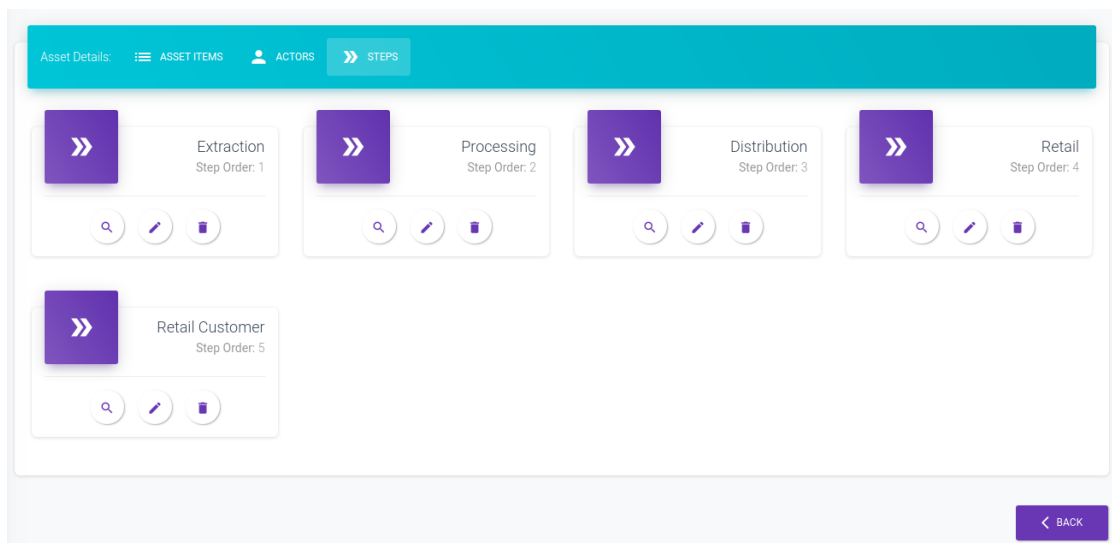


Figure 4.10 Steps list.

As an admin, there are also button actions to perform crud operations. Beyond Admins, actors responsible for the first step are allowed to create an asset item. This action will redirect the user to the create asset item form shown below. In the use case, a coffee crop was harvested by the extraction company. Information about this step is added at this point.



The screenshot shows a web form titled "Create Asset Item". The form is contained within a light gray border. At the top, there is a teal header bar with the text "Create Asset Item" in white. Below the header, the form is organized into several sections. On the left, there is a text input field for "Asset Item ID" with a small note below it: "A new ID will be generated if not provided". To the right of this is another text input field for "Parent ID" with the value "0" displayed. Below these are a dropdown menu for "Owner" and a text input field for "Step ID" with the value "1". Further down is a text input field for "Delivery Date" and another for "Process Date" with the value "16/05/2021 20:42:36". At the bottom of the form, there are four text input fields: "Order Price", "Shipping Price", "Status", and "Quantity". To the right of these fields are two buttons: a white "CANCEL" button and a teal "SUBMIT" button. At the very bottom right of the form area, there is a purple button with a white left-pointing arrow and the text "BACK".

**Figure 4.11** Create asset item form.

Beyond the crud actions in the asset items list, there are also two new actions: move asset item and track asset item. The first one shows a form where the user can move an asset through the SCM steps. The user can only move this asset item to the next or the previous step in the supply chain. Move an item back to the previous step is a feature that can be used when a customer needs to return the product to whoever sold it, for example, when a product comes defective, or the customer wants to exchange it.

To the use case, the companies are using this feature to move towards the products. First, the Manufacturer company bought two lots from the same coffee crop of the extraction company. The manufacturer sold them into three lots after processing the products. The first and the second one were sent to the first distributor company and the third to the second distributor. The first distributor delivered the first lot to the retail company, and from this lot, a cup of coffee was sold to the final customer. All these actions were made from the moving asset item form below:

The screenshot shows a web form titled "Move Asset Items" with the subtitle "Move Assets Items between Steps". The form is divided into two main sections, connected by a double arrow icon (➡).

**Left Section (Current State):**

- Asset Item ID: 10043c19-baab-4083-bf52-bcb1e0009475
- Parent ID: a544bb43-61c6-48e9-9908-3edc17837dc3
- Step: Processing
- Owner: Donald Jackson
- Process Date: 04-11-2021 21:51:26
- Delivery Date: 2021-05-11T15:06:54
- Order Price: 50
- Shipping Price: 5
- Status: Sold
- Quantity: 2

**Right Section (Target State):**

- Asset Item ID: (empty)
- Parent ID: 10043c19-baab-4083-bf52-bcb1e0009475
- Step: Extraction (selected in a dropdown menu)
- Process Date: 05-16-2021 19:28:34
- Delivery Date: (empty)
- Order Price: (empty)
- Shipping Price: (empty)
- Status: (empty)
- Quantity: (empty)

At the bottom right of the form, there are two buttons: "CANCEL" and "SUBMIT".

**Figure 4.12** Move asset item form.

Track asset item displays the tracked info about the chosen asset item. It shows the children's tree of the selected element and its ancestors. Figure 4.14 shows the current status of the scenario described above by viewing the first asset item selected (the extracted coffee crop).

When clicking on a node in the chart, the information about the selected node is displayed under the diagram. Using this feature, the final user can track and see back that its cup of coffee has been passed. This information would be essential if a problem occurred. All the companies involved in this process could also track this information to understand better what happened, identify the possible step where a problem arose, and provide input for decision making.

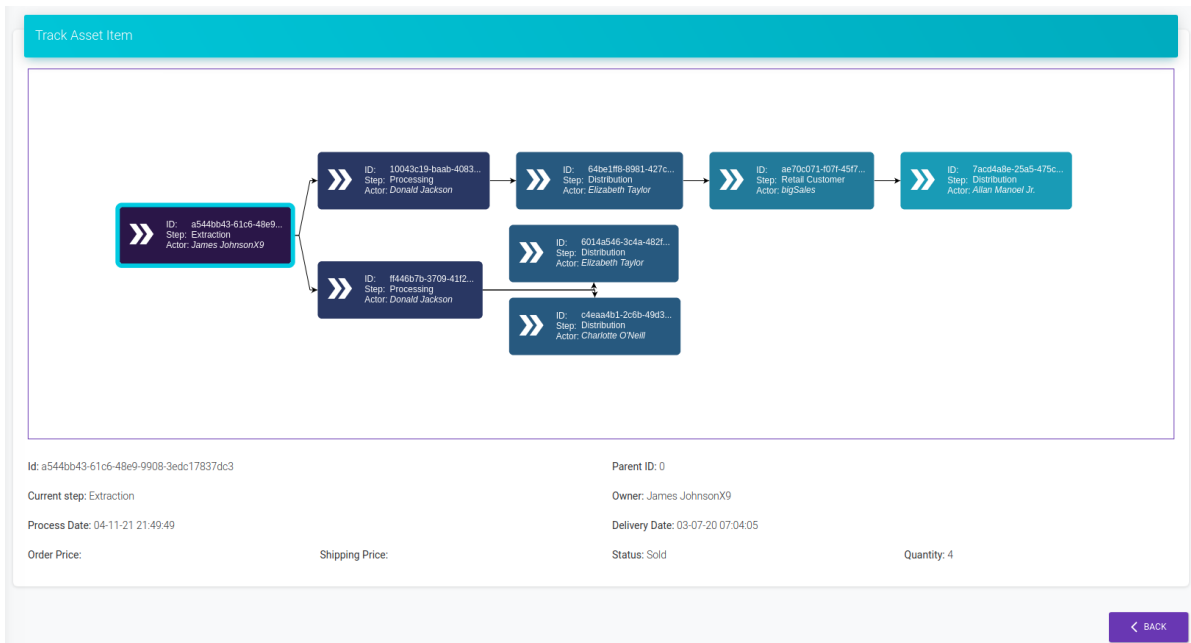


Figure 4.13 Track an asset item forward.

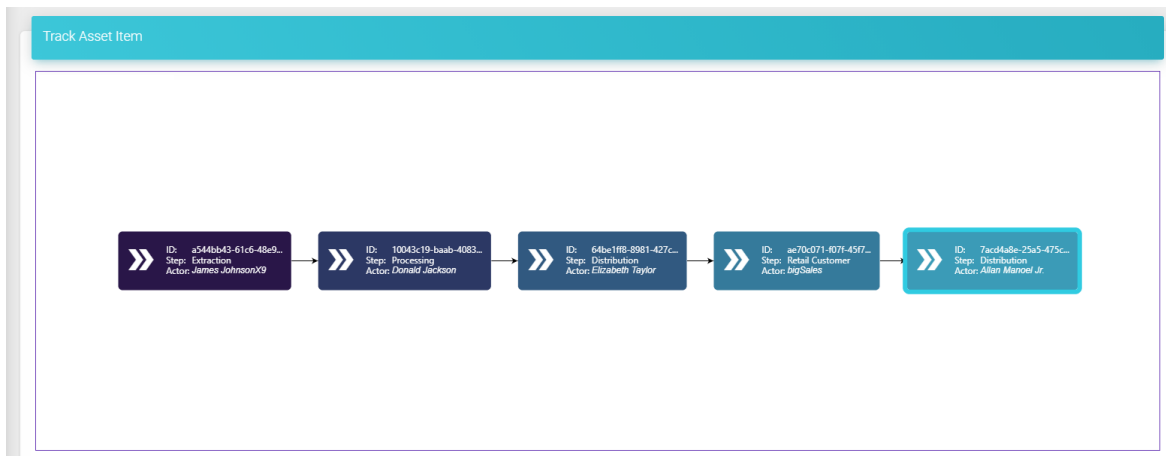


Figure 4.14 Track an asset item backward.

Usage of blockchain helped to solve the problem since all the information under the blockchain is immutable. It can be audited since the information is stored and cannot be modified or deleted. The system allows anyone to check the previous' records traceability, which can be achieved by arriving at the beginning of the chain. A blockchain does play a key role in traceability, as it ensures the data logged is not tampered with once it has been saved to the blockchain.

## **CONCLUSION AND FUTURE WORK**

Lately, Blockchain technology has been the subject of extensive research but rarely related to supply chain traceability. Although some companies have launched pilot projects using blockchain technology to manage their supply chains, no detailed information on the technical implementation of such projects has been reported. Either way, the retail industry has the potential to use this technology to improve traceability.

Even if some properties of blockchain implementation may be beneficial for supply chain management, there are still few uses to support this claim. With so little research on this subject, it is difficult for industry stakeholders to understand exactly how blockchain technology can be used in their specific business.

This dissertation has presented a framework for new decentralized traceability systems based on blockchain technology. Moreover, an example scenario was given to demonstrate how it works in an enterprise supply chain. This system delivers real-time information to all supply chain members on the safety status of goods, significantly reduces the risk of centralized information systems, and brings more secure, distributed, transparent, and collaborative to the supply chain management. The Framework can significantly improve the development time of Supply Chain Management applications and provide efficiency and transparency of products in a supply chain.

by joining the blockchain consortium, stakeholders can obtain benefits, but adopting new technology such as blockchain is challenging for traditional industries due to the learning curve and cost of integrating blockchain into existing systems. Negotiating business details also takes time. In addition, the development of smart contracts must take into account quality and adaptability. Transparency and data sharing are most important in this regard. In general, blockchain is a good option for providing traceability in supply chain management. However, the industry needs to look more closely at its risks and opportunities.

Blockchain enables end-to-end traceability by bringing a common technological language to the supply chain, while allowing consumers to access the story of goods on their labels through any connected device. This characteristic has raised the need to

trace products through the complex supply chain from retail back to the farm: to trace an outbreak; to verify that a product is kosher, organic, or allergen-free; or to assure transparency to consumers. Digital product information such as farm origination details, batch numbers, factory and processing data, expiration dates, storage temperatures, and shipping details are digitally connected to items. Their information is entered into the blockchain at each step of the process. All members of the business network agree upon the information acquired in each transaction. Once consensus is reached, no permanent record can be altered. Each piece of information provides critical data that may potentially reveal safety issues with the product concerned. The record created by the blockchain can also help retailers to manage the shelf life of products in individual stores and further strengthen safeguards relating to food authenticity. Across ecosystems, business model changes enabled by blockchain can bring strengthened trust, transparency and a new link to value exchange. Whether it is individuals seeking to complete transactions involving many parties, or enterprises collaborating across multiple organizational silos, wherever any documents or transactions must be confirmed, settled, exchanged, signed, or validated, there are usually frictions that can be avoided by using blockchain technology to unlock greater economic value.

We propose a deeper evaluation that may analyze different product types and accomplish performance tests as future work. Besides, the role permission could be applied to guarantee that only allowed users could read/write sensitive information in the blockchain. This could be made by using a flag in the additional info to show the field as public/private information or, better, use the private data collection feature provided by Hyperledger. Also, the asset item's data structure could be changed to a tree data structure for better performance results.

## BIBLIOGRAPHY

- 101BLOCKCHAINS. *public vs private blockchain*. 2020. (<https://101blockchains.com/public-vs-private-blockchain>). [Online; accessed 17-September-2020].
- ABEYRATNE, S. A.; MONFARED, R. P. Blockchain ready manufacturing supply chain using distributed ledger. © The Authors. Published by eSAT, 2016.
- ABRAHAM, I.; MALKHI, D. et al. The blockchain consensus layer and bft. *Bulletin of EATCS*, v. 3, n. 123, 2017.
- ANDROULAKI, E. et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In: ACM. *Proceedings of the Thirteenth EuroSys Conference*. [S.l.], 2018. p. 30.
- ANTONOPOULOS, A. M. *Mastering Bitcoin: Programming the open blockchain*. [S.l.]: " O'Reilly Media, Inc.", 2017.
- AUNG, M. M.; CHANG, Y. S. Traceability in a food supply chain: Safety and quality perspectives. *Food control*, Elsevier, v. 39, p. 172–184, 2014.
- B., H. *Blockchain: the solution for transparency in product supply chains*. 2015. (<https://www.provenance.org/whitepaper>). [Online; accessed 17-september-2019].
- BASHIR, I. *Mastering blockchain: Distributed ledger technology, decentralization, and smart contracts explained*. [S.l.]: Packt Publishing Ltd, 2018.
- BELLU, R. *Microsoft Dynamics 365 for Dummies*. [S.l.]: John Wiley & Sons, 2018.
- BLOCKGEEKS. What is blockchain technology? a step-by-step guide for beginners. *BlockGeeks*, 2016.
- BLOCKGEEKS, I. A deeper look at different smart contract platforms. *Blockgeeks Inc*, 2018.
- BRABBANI, H. *What is Hashing & Digital Signature in The Blockchain?* *Blockgeeks*. [S.l.]: Blockgeeks, 2017.
- BUTERIN, V. *What Are Smart Contracts? A Beginner's Guide to Smart Contracts*. [S.l.]: Blockgeeks, 2016.
- BUURMAN, J. *Supply chain logistics management*. [S.l.]: McGraw-Hill, 2002.

CACHIN, C. et al. Architecture of the hyperledger blockchain fabric. In: CHICAGO, IL. *Workshop on distributed cryptocurrencies and consensus ledgers*. [S.l.], 2016. v. 310, p. 4.

CACHIN, C.; VUKOLIĆ, M. Blockchain consensus protocols in the wild. *arXiv preprint arXiv:1707.01873*, 2017.

CARO, M. P. et al. Blockchain-based traceability in agri-food supply chain management: A practical implementation. In: IEEE. *2018 IoT Vertical and Topical Summit on Agriculture-Tuscany (IOT Tuscany)*. [S.l.], 2018. p. 1–4.

CHEGG. *Forging digital signature*. 2017. (<https://www.chegg.com/homework-help/questions-and-answers/forging-digital-signature-working-procedure-digital-signature-illustrated-figure-21>). [Online; accessed 17-September-2021].

CHRISTIDIS, K.; DEVETSIKIOTIS, M. Blockchains and smart contracts for the internet of things. *Ieee Access*, Ieee, v. 4, p. 2292–2303, 2016.

CHRISTOPHER, M. I. *Logistics & supply chain management*. [S.l.: s.n.], 2017.

COINDESK. *State of Blockchain q1 2016: Blockchain Funding Overtakes Bitcoin*. 2016. (<https://www.coindesk.com/state-of-blockchain-q1-2016>). [Online; accessed 17-March-2019].

COMSTOR, C. *Blockchain pública e privada qual a diferença?* 2017. (<https://blogbrasil.comstor.com/blockchain-publica-e-privada-qual-a-diferenca>). [Online; accessed 22-Julho-2019].

COOPER, M. C.; LAMBERT, D. M.; PAGH, J. D. Supply chain management: more than a new name for logistics. *The international journal of logistics management*, Emerald Group Publishing Limited, v. 8, n. 1, p. 1–14, 1997.

CROSBY, M. et al. Blockchain technology: Beyond bitcoin. *Applied Innovation*, v. 2, n. 6-10, p. 71, 2016.

DABBENE, F.; GAY, P. Food traceability systems: Performance evaluation and optimization. *Computers and Electronics in Agriculture*, Elsevier, v. 75, n. 1, p. 139–146, 2011.

DOUCEUR, J. R. The sybil attack. In: SPRINGER. *International workshop on peer-to-peer systems*. [S.l.], 2002. p. 251–260.

FOLINAS, D.; MANIKAS, I.; MANOS, B. Traceability data management for food chains. *British Food Journal*, Emerald Group Publishing Limited, v. 108, n. 8, p. 622–633, 2006.

FORMIGONI, J. *Tecnologia Blockchain: uma visão geral*. 2017. (<https://www.cpqd.com.br/wp-content/uploads/2017/03/cpqd-whitepaper-blockchain-impresso.pdf>). [Online; accessed 17-September-2020].

- FORRESTER, J. W. Industrial dynamics. a major breakthrough for decision makers. *Harvard business review*, v. 36, n. 4, p. 37–66, 1958.
- GALVEZ, J. F.; MEJUTO, J.; SIMAL-GANDARA, J. Future challenges on the use of blockchain for food traceability analysis. *TrAC Trends in Analytical Chemistry*, Elsevier, v. 107, p. 222–232, 2018.
- GOLAN, E. H. et al. *Traceability in the US food supply: economic theory and industry studies*. [S.l.], 2004.
- GREVE, F. et al. Blockchain e a revolução do consenso sob demanda. *Livro de Minicursos do SBRC*, v. 1, p. 1–52, 2018.
- GRYNA, F. M.; JURAN, J. M. *Juran's quality control handbook*. [S.l.]: McGraw-Hill, 1998.
- HORIUCHI, F. S. Rastreabilidade de um modelo de cadeia produtiva agrícola generalizado em uma rede blockchain. 2015.
- HUANG, Y. et al. Smart contract security: A software lifecycle perspective. *IEEE Access*, IEEE, v. 7, p. 150184–150202, 2019.
- IANSTITI, M.; LAKHANI, K. R. The truth about blockchain. *Harvard Business Review*, v. 95, n. 1, p. 118–127, 2017.
- IORIO, E.-D. D. *Blockchain Applications That Are Transforming Society*. [S.l.]: Block-geeks, 17.
- KAMATH, R. Food traceability on blockchain: Walmart's pork and mango pilots with ibm. *The Journal of the British Blockchain Association*, The British Blockchain Association, v. 1, n. 1, p. 3712, 2018.
- KOSBAA, M. et al. The blockchain model of cryptography and privacy preserving smart contracts. *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016.
- KOSTAREV, G. *Review of blockchain consensus mechanisms*. [S.l.]: Waves Platform. [Consulta: 12 Julho, 2018]. Disponível em: [https://blog ...](https://blog...), 2017.
- LAMPORT, L.; SHOSTAK, R.; PEASE, M. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, ACM, v. 4, n. 3, p. 382–401, 1982.
- LIAO, P.-A.; CHANG, H.-H.; CHANG, C.-Y. Why is the food traceability system unsuccessful in taiwan? empirical evidence from a national survey of fruit and vegetable farmers. *Food Policy*, Elsevier, v. 36, n. 5, p. 686–693, 2011.
- LITKE, A.; ANAGNOSTOPOULOS, D.; VARVARIGOU, T. Blockchains for supply chain management: Architectural elements and challenges towards a global scale deployment. *Logistics*, Multidisciplinary Digital Publishing Institute, v. 3, n. 1, p. 5, 2019.



- LONDE, B. J. L. Supply chain management: myth or reality? *Supply Chain Management Review*, v. 1, n. 1, p. 6–7, 1997.
- LONDE, B. J. L.; MASTERS, J. M. Emerging logistics strategies. *International journal of physical distribution & logistics management*, MCB UP Ltd, 1994.
- LU, Q.; XU, X. Adaptable blockchain-based systems: A case study for product traceability. *IEEE Software*, IEEE, v. 34, n. 6, p. 21–27, 2017.
- MANEVICH, Y.; BARGER, A.; TOCK, Y. Service discovery for hyperledger fabric. In: *Proceedings of the 12th ACM International Conference on Distributed and Event-Based Systems*. [S.l.: s.n.], 2018. p. 226–229.
- MARTIN, C.; LEURENT, H. Technology and innovation for the future of production: Accelerating value creation. In: *World Economic Forum, Geneva Switzerland*. [S.l.: s.n.], 2017.
- MEJIA, C. et al. Traceability (product tracing) in food systems: an ift report submitted to the fda, volume 2: cost considerations and implications. *Comprehensive Reviews in Food Science and Food Safety*, Blackwell Publishing, v. 9, n. 1, p. 159–175, 2010.
- MENTZER, J. T. et al. Defining supply chain management. *Journal of Business logistics*, Wiley Online Library, v. 22, n. 2, p. 1–25, 2001.
- MERKLE, R. C. A certified digital signature. In: SPRINGER. *Conference on the Theory and Application of Cryptology*. [S.l.], 1989. p. 218–238.
- MICHAEL, J.; COHN, A.; BUTCHER, J. R. Blockchain technology. *The Journal*, 2018.
- MINGXIAO, D. et al. A review on consensus algorithm of blockchain. In: IEEE. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.], 2017. p. 2567–2572.
- NAKAMOTO, S. et al. Bitcoin: A peer-to-peer electronic cash system. *Citeseer*, Working Paper, 2008.
- NARAYANAN, A. et al. *Bitcoin and cryptocurrency technologies: A comprehensive introduction*. [S.l.]: Princeton University Press, 2016.
- NYSTRÖM, M. Pkcs# 15-a cryptographic token information format standard. In: *Smart-card*. [S.l.: s.n.], 1999.
- PIZZUTI, T.; MIRABELLI, G. The global track&trace system for food: General framework and functioning principles. *Journal of Food Engineering*, Elsevier, v. 159, p. 16–35, 2015.
- PLEX. *supply chain and collaboration*. 2020. (<https://www.plex.com/products/supply-chain/supply-chain-and-collaboration>). [Online; accessed 17-September-2020].

- POLGE, J.; ROBERT, J.; Le Traon, Y. Permissioned blockchain frameworks in the industry: A comparison. *ICT Express*, 2020. ISSN 2405-9595. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405959520301909>.
- RASKIN, M. *The law and legality of smart contracts. 1 Georgetown Law Technology Review 304*. [S.l.]: GeorgeTown, 2017.
- RIEL, C. B. V.; FOMBRUN, C. J. *Essentials of corporate communication: Implementing practices for effective reputation management*. [S.l.]: Routledge, 2007.
- ROLLING, T. Using netsuite in business curriculum. *Journal of Higher Education Theory and Practice*, North American Business Press, v. 16, n. 5, p. 42, 2016.
- ROSS, D. F. *Competing through supply chain management: creating market-winning strategies through supply chain partnerships*. [S.l.]: Springer Science & Business Media, 1997.
- SAUER, P. C.; SEURING, S. Extending the reach of multi-tier sustainable supply chain management—insights from mineral supply chains. *International Journal of Production Economics*, Elsevier, 2018.
- SELFKEY. *Understanding Public vs. Private Blockchain*. 2020. <https://selfkey.org/understanding-public-vs-private-blockchain/>. [Online; accessed 17-September-2020].
- SNAPP, S. *Discover SAP SCM*. [S.l.]: Galileo Press, 2010.
- SVENSSON, G. The transparency of scm ethics: conceptual framework and empirical illustrations. *Supply Chain Management: An International Journal*, Emerald Group Publishing Limited, v. 14, n. 4, p. 259–269, 2009.
- SWAN, M. *Blockchain: Blueprint for a new economy*. [S.l.]: " O'Reilly Media, Inc.", 2015.
- SZABO, N. The idea of smart contracts. *Nick Szabo's Papers and Concise Tutorials*, v. 6, 1997.
- TIAN, F. A supply chain traceability system for food safety based on haccp, blockchain & internet of things. In: IEEE. *2017 International Conference on Service Systems and Service Management*. [S.l.], 2017. p. 1–6.
- VALENTA, M.; SANDNER, P. Comparison of ethereum, hyperledger fabric and corda. [ebook] *Frankfurt School, Blockchain Center*, 2017.
- WOOD, E. *A Secure Decentralised Generalised Transaction Ledger*. 2018.
- ZHENG, Z. et al. Blockchain challenges and opportunities: A survey. *Work Pap.–2016*, 2016.



## **PROJECT MANAGEMENT**

### **A.1 ACTIVITIES**

Once the software architecture and its main modules were determined, these components were divided into activities for better project management. These activities are listed below, segregated by the main modules.

#### **A.1.1 Front end**

- Create login page
- Create application configuration module
- Create user creation module (actors)
- Create data entry module (forms)
- Create data visualization module
- Create report module
- Create authentication service
- Create application setup service
- Create user creation service (actors)
- Create data entry service (forms)
- Create data visualization service
- Create reporting service

### A.1.2 Back end

- Gateway Creation
  - Create authentication endpoint
  - Create application configuration endpoint
  - Create user creation endpoint (actors)
  - Create data entry endpoint (forms)
  - Create data visualization endpoint
  - Create report endpoint
  
- Service Creation
  - Create authentication service
  - Create application setup service
  - Create user creation service (actors)
  - Create data entry service (forms)
  - Create data visualization service
  - Create reporting service
  
- Resource Locator Creation
  - Create Connector with File system
  - Create Connector with Hyperledger Blockchain
  - Create Connector with Oracle Database

### A.1.3 Hyperledger Blockchain

- Configure the blockchain network
  
- Create chaincode (Smart contract)
  
- Package and install the chaincode in the network

## A.2 USER STORIES

Table A.1 presents all the user stories for artifacts development, used in the system and managed according to agile methodologies.

**Table A.1** Supply Chain Management - Blockchain Platform (SCM-BP) User Stories

US-1	As an administrator, clicking “new” on the supply chain list page takes you to the chain configuration page, with empty settings (no phases, sub-phases, and fields).
US-2	As an administrator, clicking “edit” on the supply chain list page takes you to the chain configuration page, with the settings filled in (with phases, sub-phases, and fields already registered).
US-3	As an administrator, when clicking delete on the supply chain list page, a modal should appear requesting deletion confirmation.
US-4	As an administrator, when clicking confirm deletion on the supply chain list page, an alert should appear stating the deletion result: alert-success or alert-danger.
US-5	As an administrator, on the creation or editing screens of a chain, the administrator must tell from each section which user types can enter information in that section.
US-6	As an administrator, when clicking on the User List page redirects to the user creation page with its empty settings.
US-7	As an administrator, on the user creation page, the admin have to enter the type of user (Admin, Producer, Manufacturer, Distributor, Wholesaler, Retailer, End User).
US-8	As an administrator, clicking “edit” on the User List page takes you to the user creation page, with the settings filled in (with the previously entered data).
US-9	As an administrator, when clicking delete on the User List page, a modal should appear asking for deletion confirmation.
US-10	As an administrator, when clicking confirm deletion on the User List page, an alert should appear stating the deletion result: alert-success or alert-danger.
US-11	As a ”Member” User (Admin, Producer, Manufacturer, Distributor, Wholesaler, Retailer), clicking Move Asset redirects to the information entry page in the chain.
US-12	As a ”Member”, each user can only enter information regarding the allowed phase in the access rules (e.g., a distributor cannot enter exploration information) as defined in use case 5.
US-13	As any user (Admin, Member, or End User), clicking Track Asset will take you to a page with a list of all assets paged and filtered by date in descending order (most current to oldest).
US-14	As any user (Admin, Member, or End User), by clicking on “Track Asset”, the user can enter an Id in the input search to search.
US-15	As any user (Admin, Member, or End User), by clicking on “Track”, the user will go to a page with all information of the respective asset, from its conception to the current state.

### A.3 NON-FUNCTIONAL REQUIREMENTS

**Table A.2** Non-functional requirements of SCM-BP

NF-1: Usability	The available product, corresponding API's and documentation should be clear enough to allow for the developers to perform the implementation.
NF-2: Performance	<p><b>Speed and latency:</b> The throughput and latency on Hyperledger have already been tested, and the throughput is not expected to be as high as a centralized data system. But, overall, the time to synchronize the information from one company to another might increase; The goal is to make the product be as fast as needed to support the businesses, even if it does not have better performance than other alternatives, since what the target here is the addition of new functionalities (shared ledger);</p> <p><b>Precision and accuracy:</b> The product shall record the data just as it was entered, and predictions as to whether a product has any mismatching entries shall always be justifiable;</p> <p><b>Reliability and availability:</b> The product shall not always be available unless all of the nodes fail at once, which is almost impossible, unless a coordinated attack were to happen; If some of the nodes happen to fail, the response time of the system might be lower than expected;</p> <p><b>Scalability:</b> The product should scale to hundreds of companies, which would require a similar number of nodes;</p>
NF-3: Maintainability and portability	The product is expected to run on Linux-based systems, compatible with the Docker, nodejs, and go-lang versions that Hyperledger Fabric uses. More specifically, Oracle Cloud services have servers with the required setup for this. Creating new nodes or moving an existing one should be an easy process, without much complication, other than starting the node software on the environment and closing an existing one, if needed.
NF-4: Security	<p><b>Privacy:</b> The system must ensure appropriate visibility of transactions and products, which might be privacy sensitive; sharing some data would pose a threat or could have negative effects for some of the companies; otherwise, transactions should also be secure, authenticated, and verifiable;</p> <p><b>Immutability:</b> No one can make changes to the contents of the ledger;</p> <p><b>Authorization:</b> All changes to any data should be approved by the people that possess the data or will be affected by these changes directly. A shipment delivery transaction should, for instance, be approved by both the person delivering and the person receiving the shipment.</p>

## SMART CONTRACT AND BACKEND ENDPOINTS

### B.1 TEMPLATE FOR CONFIG FILE

```
{
  "AssetId": "assetID",
  "AssetName": "assetName",
  "AssetDescription": "assetDescription",
  "Actors": [
    {
      "actorType": "type",
      "aditionalInfo": [
        {
          "key": "value"
        }
      ]
    }
  ],
  "Steps": [
    {
      "StepId": "stepID",
      "StepName": "stepName",
      "StepOrder": 1,
      "ActorType": "actorType",
      "aditionalInfo": [
        {
          "key": "value"
        }
      ]
    }
  ]
}
```



## B.2 ASSETS, ASSET ITEMS, STEPS, AND ACTORS STRUCTS

```

type Actor struct {
    ActorId    string `json:"actorId"`
    ActorType  string `json:"actorType"`
    ActorName  string `json:"actorName"`
    Deleted    bool   `json:"deleted"`
    AdditionalInfo map[string]string `json:"additionalInfo"`
}

```

```

type Step struct {
    StepId     string `json:"stepId"`
    StepName   string `json:"stepName"`
    StepOrder  uint   `json:"stepOrder"`
    ActorType  string `json:"actorType"`
    Deleted    bool   `json:"deleted"`
    AdditionalInfo map[string]string `json:"additionalInfo"`
}

```

```

type AssetItem struct {
    AssetItemId string `json:"assetItemId"`
    OwnerId     string `json:"ownerId"`
    StepID      string `json:"stepID"`
    ParentID    string `json:"parentID"`
    Children    []string `json:"children";`
    ProcessDate string `json:"processDate"`
    DeliveryDate string `json:"deliveryDate"`
    OrderPrice  string `json:"orderPrice"`
    ShippingPrice string `json:"shippingPrice"`
    Status      string `json:"status"`
    Quantity    string `json:"quantity"`
    Deleted     bool   `json:"deleted"`
    AdditionalInfo map[string]string `json:"additionalInfo"`
}

```

```

type Asset struct {
    AssetId     string `json:"assetId"`
    AssetName   string `json:"assetName"`
    Description string `json:"description"`
    AssetItems  []AssetItem `json:"assetItems"`
    Actors      []Actor    `json:"actors"`
    Steps       []Step     `json:"steps"`
    Deleted     bool       `json:"deleted"`
    AdditionalInfo map[string]string `json:"additionalInfo"`
}

```

```
}

```

### B.3 MAIN FUNCTION

```
func main() {
    chaincode, err := contractapi.NewChaincode(new(SmartContract))
    if err != nil {
        fmt.Printf("Error create chaincode: %s", err.Error())
        return
    }
    if err := chaincode.Start(); err != nil {
        fmt.Printf("Error starting chaincode: %s", err.Error())
    }
}
```

### B.4 CREATE ASSET

```
func (s *SmartContract) CreateAsset(
    ctx contractapi.TransactionContextInterface, assetId string,
    assetName string, description string, assetItems []AssetItem,
    actors []Actor, steps []Step, additionalInfo map[string]string) error {

    if err != nil {
        return fmt.Errorf(
            "Failed to read the data from world state: %s", err
        )
    }

    if assetJSON != nil {
        return fmt.Errorf("The asset %s already exists", assetID)
    }

    asset := Asset {
        AssetId:      assetId,
        AssetName:    assetName,
        Description:  description,
        AssetItems:   assetItems,
        Actors:      actors,
        Steps:       steps,
        Deleted:     false,
        additionalInfo: additionalInfo,
    }
    assetAsBytes, _ := json.Marshal(asset)
    if err != nil {
```

```

    return err
}
return ctx.GetStub().PutState("ASSET_"+assetId,assetAsBytes)
}

```

## B.5 MOVE ASSET ITEM

```

func (s *SmartContract) MoveAssetItem(
    ctx contractapi.TransactionContextInterface,
    assetItemID string, newAssetItemID string, stepID string,
    newOwnerID string, orderPrice string, shippingPrice string,
    status string, quantity string,
    additionalInfo map[string]string ) error {

    assetItemJSON, err := s.QueryAssetItem(ctx, assetItemID)
    if err != nil {
        return err
    }
    if assetItemJSON == nil {
        return fmt.Errorf("The assetItem %s does not exists", assetItemID)
    }

    newAssetItem := AssetItem{
        AssetItemID:      newAssetItemID,
        OwnerID:          newOwnerID,
        StepID:           stepID,
        ParentID:        assetItemID,
        Children:         []string{},
        ProcessDate:      time.Now().Format("2006-01-02 15:04:05"),
        OrderPrice:       orderPrice,
        ShippingPrice:    shippingPrice,
        Status:           status,
        Quantity:         quantity,
        Deleted:          false,
        AditionalInfoMap: additionalInfo,
    }

    assetItemAsBytes, err := json.Marshal(newAssetItem)
    if err != nil {
        return err
    }

    return ctx.GetStub().PutState(
        "ASSET_ITEM_"+newAssetItemID, assetItemAsBytes

```

```
)
}
```

## B.6 TRACK ASSET ITEM

```
func (s *AssetTransferSmartContract) TrackAssetItem(
    ctx contractapi.TransactionContextInterface,
    assetItemID string) ([]*AssetItem, error) {

    assetItem, err := s.QueryAssetItem(ctx, assetItemID)
    log.Print("tracking info from assetItem id: ", assetItem.AssetItemID)
    if err != nil {
        return nil, fmt.Errorf(
            "Failed to read from world state. %s", err.Error()
        )
    }

    if assetItem == nil {
        return nil, fmt.Errorf("%s does not exist", assetItemID)
    }

    trackedItems := make([]*AssetItem, 0)

    //first add the children to tracked items
    children, err := s.getChildrenTree(ctx, assetItem)
    if err != nil {
        return nil, fmt.Errorf(
            "Failed to read from world state. %s", err.Error()
        )
    }

    for _, child := range children {
        fmt.Println(child)
        trackedItems = append(trackedItems, child)
    }

    //then, add the current item to tracked items
    trackedItems = append(trackedItems, assetItem)

    //finally, add the ancestor of current item to tracked items
    for {
        currentParentId, err := strconv.Atoi(assetItem.ParentID)
        if (currentParentId <= 0) {
            break
        }
    }
}
```

```

    }
    parentAssetItem, err := s.QueryAssetItem(ctx, assetItem.ParentID)
    if err != nil {
        return nil, fmt.Errorf(
            "Failed to read from world state. %s", err.Error()
        )
    }
    newParentId, err := strconv.Atoi(parentAssetItem.ParentID)
    log.Print("newParentId: ", newParentId)
    trackedItems = append(trackedItems, parentAssetItem)
    assetItem = parentAssetItem
}
return trackedItems, nil
}

func (s *AssetTransferSmartContract) getChildenTree(
    ctx contractapi.TransactionContextInterface,
    assetItem *AssetItem) ([]*AssetItem, error) {

    tree := make([]*AssetItem, 0)
    log.Print("len(assetItem.Children): ", len(assetItem.Children))
    if len(assetItem.Children) == 0 {
        tree = append(tree, assetItem)
    } else {
        for _, childId := range assetItem.Children {
            childAssetItem, err := s.QueryAssetItem(ctx, childId)
            if err != nil {
                return nil, fmt.Errorf(
                    "Failed to read from world state. %s", err.Error()
                )
            }
            if childAssetItem == nil {
                return nil, fmt.Errorf("%s does not exist", childId)
            }

            childrenTree, err := s.getChildenTree(ctx, childAssetItem)
            for _, child := range childrenTree {
                tree = append(tree, child)
            }
        }
        tree = append(tree, assetItem)
    }
    return tree, nil
}
}

```

**B.7 AUDIT METHODS**

```

func getTransactionByID(
    vledger ledger.PeerLedger, tid []byte) pb.Response {

    if tid == nil {
        return nil, fmt.Errorf("Transaction ID must not be nil.")
    }

    processedTran, err := vledger.GetTransactionByID(string(tid))
    if err != nil {
        return nil, fmt.Errorf(
            "Failed to get transaction with id %s, error %s",
            string(tid), err.Error()
        )
    }

    bytes, err := protoutil.Marshal(processedTran)
    if err != nil {
        return nil, fmt.Errorf(err.Error())
    }

    return shim.Success(bytes)
}

func getBlockByNumber(
    vledger ledger.PeerLedger, number []byte) pb.Response {

    if number == nil {
        return nil, fmt.Errorf("Block number must not be nil.")
    }

    bnum, err := strconv.ParseUint(string(number), 10, 64)
    if err != nil {
        return nil, fmt.Errorf(
            "Failed to parse block number with error %s", err
        )
    }

    block, err := vledger.GetBlockByNumber(bnum)
    if err != nil {
        return nil, fmt.Errorf(
            "Failed to get block number %d, error %s", bnum, err
        )
    }
}

```

```

bytes, err := protoutil.Marshal(block)
if err != nil {
    return nil, fmt.Errorf(err.Error())
}

return shim.Success(bytes)
}

func getBlockByHash(
    vledger ledger.PeerLedger, hash []byte) pb.Response {

    if hash == nil {
        return nil, fmt.Errorf("Block hash must not be nil.")
    }
    block, err := vledger.GetBlockByHash(hash)
    if err != nil {
        return nil, fmt.Errorf(
            "Failed to get block hash %s, error %s", string(hash), err
        )
    }

    bytes, err := protoutil.Marshal(block)
    if err != nil {
        return nil, fmt.Errorf(err.Error())
    }

    return shim.Success(bytes)
}

func getChainInfo(vledger ledger.PeerLedger) pb.Response {
    binfo, err := vledger.GetBlockchainInfo()
    if err != nil {
        return nil, fmt.Errorf(
            "Failed to get block info with error %s", err
        )
    }
    bytes, err := protoutil.Marshal(binfo)
    if err != nil {
        return nil, fmt.Errorf(err.Error())
    }
    return shim.Success(bytes)
}

func getBlockByTxID(

```

```

vledger ledger.PeerLedger, rawTxID []byte) pb.Response {
txID := string(rawTxID)
block, err := vledger.GetBlockByTxID(txID)
if err != nil {
    return nil, fmt.Errorf(
        "Failed to get block for txID %s, error %s", txID, err
    )
}
bytes, err := protoutil.Marshal(block)
if err != nil {
    return nil, fmt.Errorf(err.Error())
}
return shim.Success(bytes)
}

func (e *LedgerQuerier) Invoke(stub shim.ChaincodeStubInterface) pb.Response {
    args := stub.GetArgs()
    fname := string(args[0])
    cid := string(args[1])
    sp, err := stub.GetSignedProposal()
    name, err := protoutil.InvokedChaincodeName(sp.ProposalBytes)
    targetLedger := e.ledgers.GetLedger(cid)
    qsccllogger.Debugf("Invoke function: %s on chain: %s", fname, cid)
    res := getACLResource(fname)

    switch fname {
    case GetTransactionByID:
        return getTransactionByID(targetLedger, args[2])
    case GetBlockByNumber:
        return getBlockByNumber(targetLedger, args[2])
    case GetBlockByHash:
        return getBlockByHash(targetLedger, args[2])
    case GetChainInfo:
        return getChainInfo(targetLedger)
    case GetBlockByTxID:
        return getBlockByTxID(targetLedger, args[2])
    }

    return nil, fmt.Errorf(
        "Requested function %s not found.", fname
    )
}

```



## B.8 BACKEND ENDPOINTS

Actors	URI	Description
POST	/actors	creates a new actor
GET	/actors	retrieves the actors' list
PUT	/actors/:id	updates an existing actor
GET	/actors/:id	retrieves an existing actor
DELETE	/actors/:id	removes an existing actor
Steps		
POST	/steps	creates a new step
GET	/steps	retrieves the steps' list
PUT	/steps/:id	updates an existing step
GET	/steps/:id	retrieves an existing step
DELETE	/steps/:id	removes an existing step
Asset Items		
POST	/asset-items	creates a new asset item
GET	/asset-items	retrieves the asset items' list
PUT	/asset-items/:id	updates an existing asset item
GET	/asset-items/:id	retrieves an existing asset item
DELETE	/asset-items/:id	removes an existing asset item
POST	/asset-items/:id	moves an asset item through the SCM
GET	/asset-items/track/:id	tracks an asset item through the SCM
Assets		
POST	/assets	creates a new asset
GET	/assets	retrieves the assets' list
PUT	/assets/:id	updates an existing asset
GET	/assets/:id	retrieves an existing asset
DELETE	/assets/:id	removes an existing asset
Audit		
GET	/blocks/:number	Return the block specified by block number
GET	/blocks/hash/:hash	Return the block specified by block hash
GET	/blocks/tx-id/:tx-id	Return the transaction by Transaction ID
GET	/transactions:id	Return the transaction by ID
GET	/chain	Return a blockchain Info object

# Appendix

# C

## DATA STRUCTURE

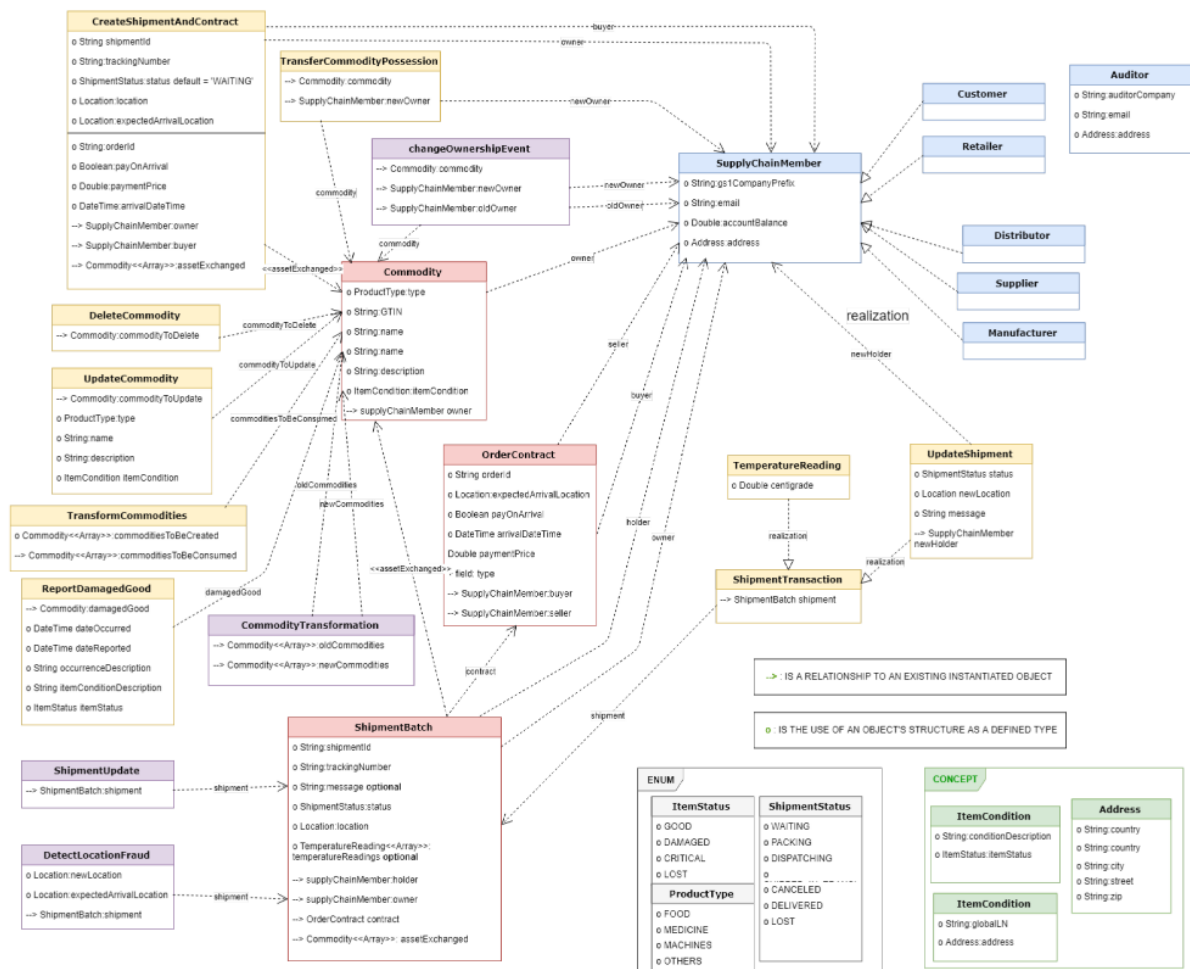


Figure C.1 SCM-BP data structure