



BACHELOR'S THESIS
COMPUTER SCIENCE DEGREE
INFORMATION TECHNOLOGIES MENTION



Traceability and tracing of pharmaceutical distribution through Blockchain and Smart Contracts

Student: Javier Escribano Salgado
Direction: José Manuel Vázquez Naya
Direction: Cristian Robert Munteanu

A Coruña, September 2022

Dedicated to my family, Martín and Bruno, you were always there when no one else was.

Acknowledgements

I would like to thank my family and friends for accompanying me throughout the project, and especially throughout these four years.

Abstract

Pharmaceutical supply chains require a large number of actions and resources to track products circulating there. However, the appearance of Blockchain implies a substantial advance in identifying products since it adapts perfectly to the conditions imposed by the sector.

Therefore, this Bachelor's thesis investigates technologies such as Blockchain to reinforce the mechanisms and guarantee security in the traffic of products throughout the pharmaceutical supply chain. The developed system allows users to interact with it, facilitating a graphical interface with all the functionalities offered to add, update and view information on both medicines and workers. In addition, this system offers great reliability when it comes to ensuring the integrity of the information assigned to medicines, where data is accessible throughout the supply chain, ensuring total transparency between members of the chain and end users. Therefore, in terms of security, this system makes great strides to offer a solution against falsifications in the supply of medicines and their subsequent sale.

Resumo

As cadeas de suministro farmacéuticas requiren un gran número de accións e recursos para poder realizar o seguimento dos produtos que por alí circulan. Sin embargo, a aparición de Blockchain implica un avance substancial á hora de identificar os produtos que se amolda perfectamente ás condicións impostas polo sector.

Por ello, este Traballo Fin de Grao investiga en tecnoloxías como Blockchain para reforzar os mecanismos e garantir a seguridade no tráfico de produtos ao longo da cadea de suministro farmacéutico. Desenvolveuse un sistema co que os usuarios poden interactuar facilitando unha interface gráfica con todas as funcionalidades ofrecidas para engadir, actualizar e visualizar información tanto de medicamentos como dos traballadores. Ademais, este sistema ofrece unha gran fiabilidade á hora de garantir a integridade da información asignada aos medicamentos, onde os datos son accesibles en toda a cadea de suministro, garantindo a total transparencia entre os membros da cadea e os usuarios finais. Por iso, en materia de seguridade, o sistema dá grandes avances para ofrecer unha solución contra as falsificacións na subministración de medicamentos e a súa posterior venda.

Keywords:

- Supply Chain
- Medicines
- Blockchain
- Smart Contracts
- NFT
- IPFS
- Solidity
- Ethereum

Palabras chave:

- Cadea de suministro
- Medicinas
- Blockchain
- Contratos Intelixentes
- NFT
- IPFS
- Solidity
- Ethereum

Contents

1	Introduction	1
1.1	Goals	3
2	Fundamentals	5
2.1	Blockchain	5
2.2	Smart Contracts	6
2.3	Ethereum	7
2.4	Tokenization	8
3	State of the Art	9
3.1	Typical supply chain features	9
3.2	Pharmaceutical supply chain	11
3.3	Limitations in the supply chain	13
3.4	Solutions currently proposed	14
3.4.1	Mass Serialization	15
3.4.2	Tamper Proof Packaging	16
3.4.3	Track and trace technologies	18
3.4.4	DLT approaches	21
4	Planning and cost evaluation	25
4.1	Project Planning	25
4.2	Costs evaluation	27
5	Software Development Methodology	29
5.1	Kanban	29
5.1.1	Roles	30
5.1.2	Practices	30
5.1.3	Lifecycle	31

5.1.4	Workflow	31
6	Project Development	33
6.1	Requirements analysis	33
6.1.1	Functional requirements	33
6.1.2	Non-functional requirements	34
6.2	Design and Implementation	34
6.2.1	Iteration 1: Blockchain environment selection	34
6.2.2	Iteration 2: System Design	38
6.2.3	Iteration 3: Smart Contracts	41
6.2.4	Iteration 4: Tests	50
6.2.5	Iteration 5: Interface design	55
6.2.6	Iteration 6: Web development	58
7	Results	65
8	Conclusion	73
8.1	Future Work	74
A	Glossary of terms	79
	Bibliography	83

List of Figures

3.1	Supply Chain Stages [1].	10
3.2	Example of Flow Commercial Pharmaceutical Supply Chain in the U.S [2]. . .	12
3.3	Damage from counterfeit goods, by industry [3].	14
3.4	Mass Serialization and Track-and-Trace around the world [3].	15
3.5	Effectiveness and Cost of Hologram Counterfeiting [4].	18
3.6	Example of a serialized US drug code [5].	19
3.7	RFID vs 2 Dimensional Barcode[5].	20
3.8	Example of how fake medicines are being detected through the chain [6]. . . .	22
4.1	Project workflow visualized in a Gantt chart.	25
5.1	Part of the Workflow in Kanbanize.	32
6.1	Some types of Blockchain [7].	35
6.2	<i>MedToken</i> Class UML design.	38
6.3	Storages structures UML design.	39
6.4	<i>MedChain</i> Class UML design.	40
6.5	General System UML design.	41
6.6	<i>MedToken</i> contract code.	42
6.7	Structures in <i>MedChain</i> contract.	44
6.8	<i>Concat</i> function.	45
6.9	<i>AddWorker</i> function.	46
6.10	<i>AddMedicine</i> function.	47
6.11	<i>AddState</i> function.	48
6.12	<i>GetRestMedicine</i> function.	49
6.13	Example of an automated test function.	50
6.14	Functions to test in <i>MedChain</i> Smart Contract.	52
6.15	Example of gas cost in a transaction.	52

6.16	<i>MedChain</i> performance. Gas vs Functions.	54
6.17	<i>MedToken</i> performance. Gas vs Functions.	54
6.18	Mockup for the home page.	55
6.19	Mockup for the register page.	56
6.20	Mockup for the main page.	57
6.21	Model View Controller pattern graph [8].	58
6.22	Function connection to the network.	59
6.23	Controller function to register an user.	60
6.24	Routes for all the main components.	60
6.25	Example of forms available for a manufacturer.	61
6.26	Example of forms available for a client.	62
6.27	Example of Visible Component inside a role.	62
6.28	<i>useNotification</i> Hook for React Components.	63
6.29	Successful notification for a new medicine added.	63
7.1	Medicine's path.	66
7.2	NFT and drug data.	66
7.3	Ganache addresses.	68
7.4	Workers medicine information.	69
7.5	Worker data searched for its form.	69

Introduction

AN *estimated 1 in 10 medical products circulating in low- and middle-income countries is either substandard or falsified* [9]. That is a quote mentioned in a research conducted by WHO (World Health Organization) in 2017. That percentage has not decreased by even 1% in nearly five years. That is to say, some medicines are ineffective or cannot treat the disease they were designed for. As a result, it causes not only economic loss for the affected people, but also the consequences of severe health damage or even death.

The estimation could probably be a tiny fraction of the problem due to countless unreported cases and the ignorance of the medicine integrity during the distribution, supplying, and selling processes. Substandard medical products reach patients when tools and technical capacity to enforce quality standards in the development and provisioning phases are limited. Moreover, globalization is making it harder to regulate medical products. Many falsifiers manufacture and print packaging in different countries and ship their components to a final destination where they are assembled and distributed. Sometimes, offshore companies and bank accounts have been used to facilitate the sale of falsified medicines. However, faked products tend to transit from stages where regulation is affected by unethical methods from wholesalers, distributors, retailers, and health care workers. A high proportion of cases reported to WHO occur in countries with constrained access to medical products.

Nevertheless, in order to fight against this problem, several approaches try to serialize and trace each product to know the path and stages made from each drug accurately. For instance, Mass Serialization consists of adding machine readable codes containing serial numbers to individual packs of medicines [10]. It enables product authentication at the point of dispense (POD) and provides a potential barrier between harmful drugs and the patient. While the pharma industry is driven by financial success and risk management, there is a widespread acknowledgment that efforts to promote public safety in order to generate a positive return

on investment.

However, this solution is not even close to perfect. Mass Serialization implies an overwhelming and complex process of product authentication using traditional serialization techniques and community locations that administer medical countermeasures (MCMs) to the public. Tracing each product manually and maintaining those community locations have an implicit cost that increases massively when drugs are highly demanded. Therefore, to assure stability, there must be a selection process where certain products will remain unchecked for the anti-counterfeit systems and continue being part of the inconsistent supply network. Moreover, when we talk about a massive task, it is usually related to a government or a merger of private firms. In terms of fault tolerance, the system is sustained by one layer that, in case of failure, will cause the inoperability of the system and its consequent actions.

As a result of those drawbacks, a system where validation and authenticity are vital features needs to be brought into focus. It is necessary to offer reliable end-to-end tracking and monitoring of drugs through the supply chain, which is why companies have started using Blockchain-based systems. With Blockchain, if an issue is detected, the user could look at all previous data entries to trace the origin of the medicines, the manufacturer, or the batch where they came from. So, it will allow drugs in the supply chain to be readily traced and verified, enabling easier detection and issues rectification. This is the case of the UK National Health Service (NHS) [6], which worked with other companies to assure the authenticity of the COVID-19 vaccines using temperature-sensitive sensors logged in a decentralized network. The nodes from the network acted as an unbiased neutral party to coordinate and timestamp sensor data. The database existed across several locations and parties; avoiding being reliant on a single centralized authority (SPOF), the information was only available to trustworthy parties. However, the unique identifiers assigned to each vaccine batch had no usage outside the enterprise network. Hence, customers did not have any relation with the supply chain, and they could not verify the authenticity of the element, although the UK government previously assured it.

In order to improve validation, traceability, and authenticity, this Bachelor's thesis will investigate decentralization technologies such as Blockchain and enforce several mechanisms to assure safety and security through drug flow in the supply chain. However, throughout the project development process, it is necessary to face difficulties related to the validation, integration, and reliability of the information on medicines. In other words, it must comply with the standards that, today, are met under the tutelage of more complex and massive supply techniques. Therefore, this approach will be based on an environment that can be

changeable over time because it will depend on the basic needs of clients and workers that will be all part of the system itself. Additionally, the system will be sustained by distributed life nodes so that there will not be a central controller in the network, and all the participants will talk to each other directly. This property will allow transactions to be conducted directly among peers without third-party involvement. Moreover, thanks to the distributed ledger, each node will hold a copy of it, and it will be consensually shared and synchronized across multiple sites accessible by multiple users. That is, transactions will have public "witnesses," and participants can access the recordings shared across the network by owning an identical duplicate. Therefore, any changes or additions made to the ledger are reflected and copied to all participants in a matter of time.

Furthermore, this project tries to improve several points that, in a pharmaceutical supply chain, were not trivial to implement or deploy, that is the case of the identifiers used in drug traceability. Since we are dealing with a Blockchain environment, this project will introduce technologies based on Non Fungible Tokens (NFT) to maintain the untransferability of medicines identifiers. They will work together with the IPFS protocol (Interplanetary File System) that will offer a high-performance block storage model so that the information of each medicine will be duplicated and distributed among peers simultaneously. In order to provide the minimum risk and maximum safety for the user in terms of validation and authenticity, the system will offer different actions based on the "actor" that executes the function itself. That is why the project will offer the possibility of spreading functionalities in several applications and guarantee its accurate performance in private or public environments; depending on the system's context, it will be flexible enough to perform successfully in different backgrounds or situations.

1.1 Goals

The main goal of this Bachelor's Thesis is to deploy a Blockchain system-based that let users assure the reliability and integrity of medicines. In order to achieve that goal first, the project will include an exhaustive and detailed study of the different Blockchain-based application development environments to become familiar with its structure and implement the system efficiently and effectively. Moreover, there would be an analysis of advantages and drawbacks of those environments, as well as the proper choice of the environment in which the Smart Contracts will be hosted.

Next, there would be the implementation and test of the Smart Contracts, in which medicine data will be validated and stored in order to assure different features related with the integrity, transparency, reliability, and authenticity of each medicine. Within the Smart Contracts im-

plementation we can point out the addition of new medicines, the edition of drugs state and the visualization of their information, as well as the certification of medicines identifiers.

To achieve the objectives mentioned above, an interface must be created in order to achieve user interaction, as well as carry out basic actions by communicating directly with the Blockchain without intermediaries in between. It should be noted that this interface must have usability and efficiency features so that any user can interact with the application intuitively and effectively.

Fundamentals

On account of the complexity of the knowledge expressed in this project, this chapter will try to define those concepts that, whether due to work, nature or ideology, are not widely known.

2.1 Blockchain

Blockchain stands for a “*peer-to-peer, distributed ledger that is cryptographically secure, append-only, immutable (extremely hard to change), and updateable only via consensus or agreement among peers*” [11]. The definition really means that there is no central controller in the network and that all participants talk to each other directly. This allows transactions to be directed between peers without interference from a third party. In addition, a ledger is spread across the network among all its peers, and each one holds a copy of the complete ledger. Essentially, we can compare it with a database, but it is consensually shared and synchronized across multiple sites, institutions or geographies, accessible by multiple users. It allows transactions to have public “witnesses”. Moreover, cryptography is being used to provide security services that make this ledger safe against tampering and misuse, as well as the append-only property which means that data can only be added to the Blockchain in *time-sequential* order. Hence, once data is added to the Blockchain, it is almost impossible to change, and it has to be considered practically immutable. Lastly, its consensus, which is the most critical attribute due to the chain is only updatable if all peers agree with each other, so that any update made in the Blockchain is validated against strict criteria defined by the Blockchain protocol and added to it only after a consensus has been reached.

Within the Blockchain there are several elements that makes the system follow an architecture based on transaction validation and peer-to-peer node network [11]:

- **Address:** they are unique identifiers used in a Blockchain transaction to denote senders and recipients. An address is usually a public key or derived from a public key.

- **Transaction:** it is the fundamental unit of a Blockchain. A transaction represents a transfer of value from one address to another.
- **Block:** it is composed of multiple transactions and other elements, such as the previous block hash (hash pointer), timestamp, and nonce. A block is composed by a block header and a selection of transactions bundled together and organized logically. A block contains several elements:
 - A reference to a previous block is also included in the block unless it is a **genesis block** (first). This reference is the hash of the header of the previous one. A **genesis block** is the first one in the Blockchain that is hardcoded at the time the Blockchain was first started. The structure of a block is also dependent on the type and design of a Blockchain.
 - A **nonce** is a number that is generated and used only once. A nonce is used extensively in many cryptographic operations to provide replay protection, authentication, and encryption. In Blockchain, it's used in PoW consensus algorithms and for transaction replay protection. A block also includes the nonce value.
 - A **timestamp** is the creation time of the block.
 - **Merkle root** is the has of all of the nodes of a Merkle tree. In a Blockchain block, it is the combined hash of the transactions in the block. Merkle tress are widely used to validate large data structures securely and efficiently.
 - In addition to the block header, the block contains transactions that make up the block body. A **transaction** is a record of an event, for example, the event of transferring cash from a sender's account to a beneficiary's account. A block contains transactions and its size varies depending on the type and design of the Blockchain.

2.2 Smart Contracts

A **Smart Contract** "is a secure and unstoppable computer program representing an agreement that is automatically executable and enforceable" [11]. Dissecting the definition reveals that a Smart Contract is, fundamentally, a computer program that is written in a language that a computer or target machine can understand. Also, it encompasses agreements between parties in the form of business logic. Another fundamental idea is that Smart Contracts are automatically executed according to the instruction that is coded in, for example, when certain conditions satisfy.

They are enforceable, which means that all contractual terms perform as specified and expected, even in the presence of adversaries. Moreover, they are secure and unstoppable,

which means that these computer programs are fault-tolerant and executable in a reasonable (finite) amount of time. These programs should be able to execute and maintain a healthy internal state, even if external factors are unfavorable.

Enforcement is a broader term that encompasses traditional enforcement in the form of a law, along with the implementation of specific measures and controls that make it possible to execute contract terms without requiring any intervention. Preferably, Smart Contracts should not rely on any traditional methods of enforcement. Instead, they should work on the principle that *code is the law*, which means that there is no need for an arbitrator or a third party to enforce, control, or influence the execution of a Smart Contract. They are self-enforcing as opposed to legally enforceable. This idea is entirely possible and it is in line with the true spirit of Smart Contracts.

Lastly, they are secure and unstoppable, which means that these computer programs are fault-tolerant and executable in a reasonable (finite) amount of time. These programs should be able to execute and maintain a healthy internal state, even if external factors are unfavorable. However, if the environment it is running in or the external factors it relies on deviate from the usual or expected state, the program may react arbitrarily or abort. Smart Contracts must be immune to this type of issue.

2.3 Ethereum

It is an open source platform, which its main goal is to execute and deploy Smart Contracts. It is programmable, which stays as base for developers that can use it in order to create decentralized applications [12]. Ethereum, just like an other Blockchain, can be visualized as a transaction-based state machine. The core idea is that in the Ethereum Blockchain, a genesis state is transformed into a final state executing transactions incrementally.

The Ethereum network is a peer-to-peer network where nodes participate in order to maintain the Blockchain and contribute to the consensus mechanism. Networks can be divided into types based on the requirements and usage. For instance, the **Mainnet**, that is the current live network of Ethereum. Its network ID is 1 as its chain ID, which are used to identify the network. Another example are the **Testnets** whose aim is to provide a testing environment for Smart Contracts and Decentralized Apps before being deployed to the production live Blockchain. Moreover, they also allow experimentation and research.

This Blockchain is made up from many features that stand for the contribution and main-

tenance of the Blockchain, but there is one characteristic that makes the network different from, for example, Bitcoin the first public Blockchain. That feature is the **Ethereum Virtual Machine**:

- The EVM is a simple stack-based execution machine that runs bytecode to transform the system state from one state to another [11]. The word size of the EVM is set to 256-bit and the stack size is limited to 1024 elements which is based on the LIFO queue. The EVM is a Turing-complete machine but is limited by the amount of gas that is required to run any instruction. This means that infinite loops that can result in DOS attacks are not possible due to gas requirements. The EVM is an entirely isolated and sandboxed runtime environment. The code that runs on the EVM does not have access to any external resources such as a network or a filesystem. This results in increased security, deterministic execution, and allows untrusted code to be executed on the Ethereum Blockchain.

2.4 Tokenization

In the context of Blockchain, it is the process of representing an asset digitally on a Blockchain. It can be used to represent commodities, real estate, ownership of art, currency, or anything else of value [11]. There are several types of them, but as this project uses non-fungible tokens, we will bring the focus to that type specifically.

NFTs are unique and different from other tokens of the same type or in the same category. In addition, since they are unique and represent specific attributes, these tokens are not interchangeable with tokens of the same type. We can look at it as a certificate of authenticity that represents an specific asset. Lastly, they are available only as a complete unit. They are associated with an unique individual, it is thus not interchangeable, and it is not rational for it to be divided into fractions, making it indivisible [11]. Its standard is called **ERC-721** and it mandates a number of rules that must be implemented in a Smart Contract for it to be ERC-721 compliant. These rules govern how these tokens can be managed and traded. ERC-721 was made by a Blockchain in which was available the assests traffic in order to trade and represent unique entities within the network.

State of the Art

THIS chapter will try to analyze in depth the supply chains and how problems related to fraud and counterfeiting of pharmaceutical products are currently faced throughout all the processes of the chain.

3.1 Typical supply chain features

When we talk about supply chain, we can adopt the term provided by CSCMP (Council for Supply Chain Management Professionals), which is, "the planning and management activities that ensure the availability of needed products from the manufacturers to end users" (CSCMP, 2011) [13]. Essentially, it consists of fulfilling a customer request involving all parties directly or indirectly, and including not only the manufacturer and suppliers, but also transporters, warehouses, retailers, and even the customers themselves [14].

If we break down the term itself, it conjures up images of products or supplies moving from suppliers to manufacturers to distributors to retailers to customers along the chain. However, it is also important to visualize information, funds, and product flows along different directions of the chain. Therefore, if we want to describe the structure of a supply chain, it would be more accurate if we use the terms "network" or "web". So, based on the bare definition, we can identify several typical stages that are found in every supply network:[1]

- Component/Raw Material Suppliers
- Manufacturers
- Wholesalers/Distributors
- Retailers
- Customers/Clients

In general, these stages are widely interconnected via communication flows (figure 3.1), allowing the process to bypass incoming flaws and achieve an optimal result. Furthermore, these flows are typically bidirectional and may be managed by one of the stages or by an intermediary. In the end, the appropriate design of the supply network depends on both the customer's needs and the roles played by the stages involved.

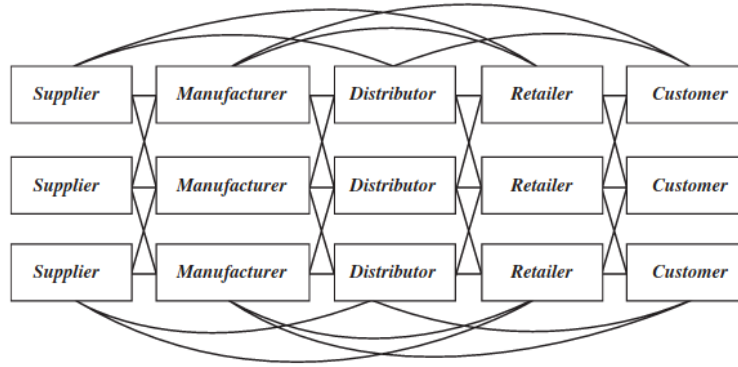


Figure 3.1: Supply Chain Stages [1].

Apparently, these stages need to be managed in a certain way that the overwhelming process could have a combination of strategy and activity in order to provide an accurate service, indeed, the bare definition of Supply Chain Management can be described by Gartner: *"The processes of creating and fulfilling demands for goods and services. It encompasses a trading partner community engaged in the common goal of satisfying end customers"* [15]. In essence, the collectively focus is that organizations must plan and coordinate supply chain activities among their network of suppliers and customers to ensure that the end product is available, that is why the assurance of delivering and authentic, valid and genuine product becomes a dramatic challenge.

The strength of supply chain management is, in reality, its ability to integrate the customer as a partner in supplying the goods or services given by a network, which enhances the information course owing to customer information analysis. In the typical supply chain, the further the members of a chain are from the end customer, the less understanding these members have of the needs of the customer, which increases uncertainty differently (increasing inventory or increasing lead times). In order to improve this situation, there has to be a strong relationship between the end client and all the processes the product made towards the final stage, where the customer gets its "value". So, authenticity and reliability are crucial terms that enable each stakeholder in a supply chain to ensure the value and security of the product, allowing the improvement of distribution channels integrity to provide end-to-end protection against counterfeit products. However, firms have to face three major challenges with conventional product authentication in the supply chain:

- Inadequate security if they are easy to use and economical.
- If we are facing a robust system, it requires a high level of technical expertise and dedicated hardware to operate.
- It is difficult and tedious to integrate them with enterprise solutions.

This overall view of a typical supply chain shows us how it is struggling with insufficient resources for efficient authenticity verification. Providing solutions that are either easy to use but insecure or uneconomical to develop. [16].

3.2 Pharmaceutical supply chain

In particular, these type of supply chains usually involve many stakeholders and stages from the API provider, drug manufacturer, packaging and distribution companies, regulators, all the way through to hospitals, pharmacies and ultimately, the patient [6]. In fact, we are dealing with many intermediaries who add complexity and makes it difficult to track and ensure authenticity, meaning counterfeit medicines are more likely to enter the distribution chain. This network is actually very complex, requiring a number of steps that must be taken to ensure medications are available and accessible to patients [17]. In this type of process, drugs that are incorrectly distributed affect both company's reputation and client satisfaction, as well as pure profit, therefore an ineffective network might disrupt the healing processes of patients and produce negative effects on public health [9].

At the most basic level, if we had to gather all the flows of goods and financial transactions among players in the commercial pharmaceutical supply chain we will get with five main stages, that can be split in two parts, Drug Flow with Contracting, and Patient Access Involving Payment [2]:

- **Manufacturers:** who produce drugs and manage the distribution from manufacturing facilities to wholesalers, and in some cases, directly to retail pharmacy chains, mail-order and specialty pharmacies, hospitals, clinics or other entities. Moreover, they manage contracts with the wholesalers in order to get some service fee in terms of inventory management, financial transactions or data processing.
- **Wholesalers:** they purchase drugs directly from manufacturers and distribute them to a variety of customers. They are usually specialized in selling certain products to specific customers.
- **Pharmacies:** in this stage pharmacies buy squarely the drugs, which must be safely stored and dispensed rightfully to patients. This is the last stage involving Contracting

and, as we have seen, final customers do not take part in any transaction of the flow so it is invisible for them.

- **PBM:** starting the Patient Access Involving, there is the Pharmacy Benefit Manager, who essentially negotiates with health plans and insurances to cover a list of drugs and display information on tier placement and coverage criteria that correspond with patient cost sharing for each drug. PBMs also offer services like drug utilization review, disease management, and consultative services to assist plans with their benefit structure.
- **Pharmacies:** they specifically consist of actors of the flow that have agreed to dispense prescription drugs and provide pharmacy services to a health plan's enrollees under a set of specified terms and conditions, which can sometimes require pharmacy payment of certain fees such as network access fees. PBMs use a point-of-sale system to link pharmacies in their networks and distribution centers to verify coverage, formulary restrictions, and patient cost sharing.
- **Client:** the final stage of the flow who can face the full cost of the drug at the pharmacy, based on the drug's list price or require a cost sharing payment which would involve a PBM with its proper health plan.

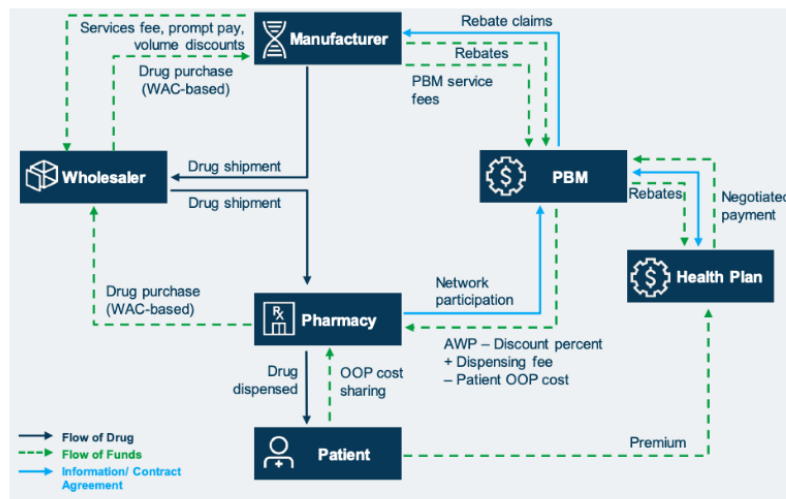


Figure 3.2: Example of Flow Commercial Pharmaceutical Supply Chain in the U.S [2].

3.3 Limitations in the supply chain

Even though we are able to see an almost flawless system of product transaction, demand management, distribution and efficient client provisioning (figure 3.2), it also has several limitations that makes the whole process struggle with shortages, tenders and epidemics, although one devastating human toll is the counterfeit pharmaceuticals. As we already know from WHO research [9], we are dealing with toxic ingredients (China, 2008) [3], serious harm from drugs purchased on the Internet (United Kingdom 2009) or even estimations of 450,000 people who die of Malaria every year after taking ineffective counterfeit pills [18]. Another big example took place in September 2013 when a counterfeit network of fake Viagra was finally arrested [19]. The amount of pills added up to 40000 and if they had been real, the value of the drugs would have risen to \$750000, but the pills did not contained active ingredients at all. Even though this crime was successfully sorted out, new cases of trafficking these fake pills were carried out without the implementation of preventing methods that could minimize that dishonest traffic of pills.

Moreover, we are dealing with a system where many elements are thought to work together to produce a successful product flow. Still, it cannot provide security and safety in all its stages because the supply chain was invented with the goal of efficiency. That is why it is difficult to grant the natural source of the fake product when we face counterfeit actions. Studies performed in Ghana drew conclusions about drug quality because their data represented a random sample of drugs from an almost exhaustive sampling frame of known pharmacies, chemical shops, and other dispensaries in their study area. The chain worked adequately, but the mischievous actions were performed in the first stage when raw materials were supplied [20]. Nevertheless, the usual method used by individuals is the introduction of faked drugs during the flow of the chain, that is, during the distribution or sale stages. The perfect example is vaccines, they are very complicated to make, and few manufacturers supply the world market. Vaccines are generally procured in bulk by governments or UN agencies in a supply chain with few intermediaries [20]. In 1995, about 60,000 Nigeriens were injected with water disguised as a meningitis vaccine during a meningitis epidemic. The substandard vaccine caused about 2,500 to 3,000 excess deaths. Another example occurred when about 200,000 doses of substandard rabies vaccine circulated in Jiangsu province in 2010 before a manufacturer recall. Assuming the patients survive injection with unidentified liquids, they are still at risk for death from the disease they were not inoculated against.

Furthermore, if we take into account the financial consequences of counterfeiting, along with suffering about €1.6 trillion is being collected every year from fraudulent goods world-

wide [3]. Moreover, as we can see in figure 3.3, keeping the system economically alive is a real struggle that gets bigger over time. The drug manufacturing ecosystem gets seriously punished by counterfeit actions in prevention methods, even reaching expense limits exceeding 2% of the total global economic output. Drug companies are spending billions of dollars every year to combat this growing threat to their bottom lines and public health.

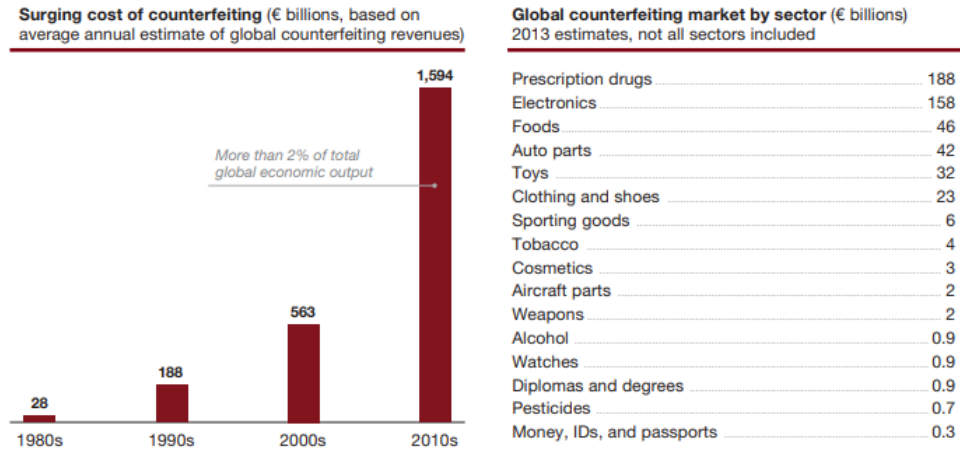


Figure 3.3: Damage from counterfeit goods, by industry [3].

Eli Lilly, for example, recently spent more than US\$100 million [21] on a new system to thwart reproductions of bigname remedies such as Cialis, Cymbalta, and Zyprexa. The reputational cost is also a crucial factor that can be hard to quantify, it is undeniable that damages related with counterfeit truly damages a company's brand name and its integrity. However, according to a survey conducted by PWC [3], pharmaceutical industry leaders tend to underestimate the threat even if they acknowledge weaknesses in their defenses. Executives interviewed showed little appetite for new spending on anti-counterfeiting measures, in part because they have already spent so much to comply with mandates such as the European Union's Falsified Medicines Directive (FMD). This view misses the opportunity to create new generation anti-counterfeiting technologies and other services that could provide strong defense against fake products.

3.4 Solutions currently proposed

From the previous exposed cases, firms and governments seem to increase public awareness of fraud and spurred regulators to action.

3.4.1 Mass Serialization

One of the main techniques is called "Mass Serialization", which is not an authentication method itself but it can be used to trace products in a supply chain. It basically encodes each drug package with an unique identifier like a scannable barcode, so that, each package that comes off the product line has an identifier code that enters into an online database. Therefore, with track-and-trace technologies those packages can be checked against the database at each point of the chain [3]. Hence, an effective Mass Serialization program should act as a safety net which would prevent counterfeit products from falling into the hands of patients by identifying them at the medical countermeasures locations. This level of security helps protect pharmaceutical companies from the negative publicity associated with a drug scare, but most importantly removes the danger of patients falling ill – or worse – through the consumption of fake, ineffectual or unsafe medicines [10]. Additionally, there would be a significant amount of goodwill to be gained from the successful implementation of these programs. Under the right conditions and with the right incentives, pharmaceutical manufacturers stand to benefit significantly from the successful adoption of such programs.

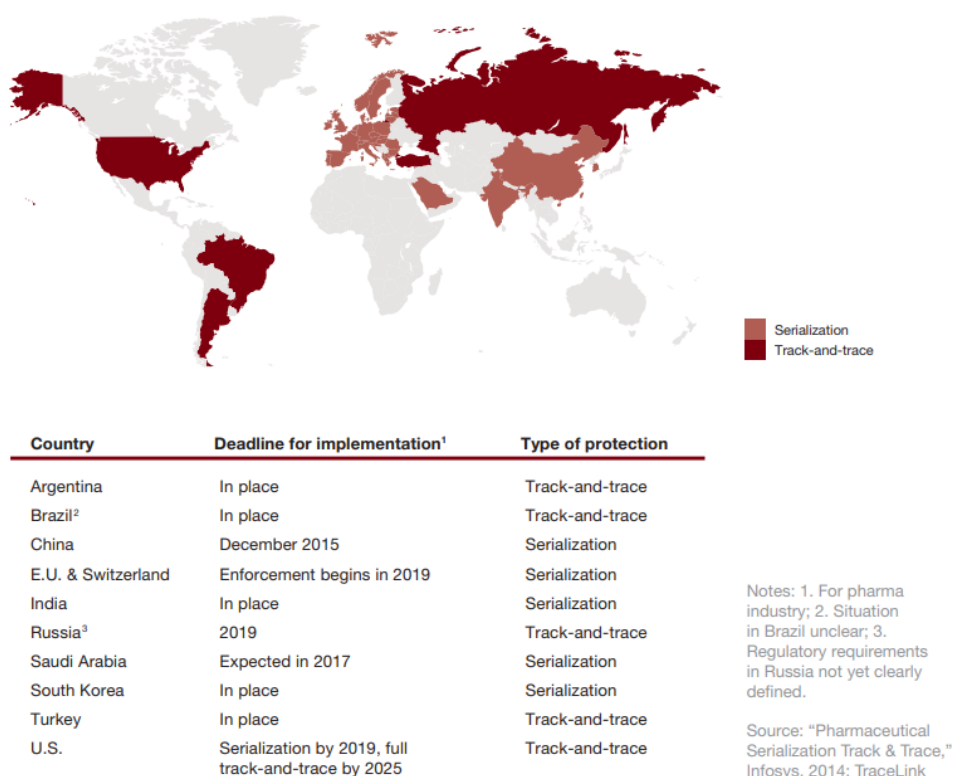


Figure 3.4: Mass Serialization and Track-and-Trace around the world [3].

Unfortunately, the new rules have some vulnerable features that allow counterfeiters to apply methods to subvert them. This method is based on centralized systems that computes all the serialization actions, as well as identifiers production and database management. Nowadays both the technology and the regulatory controls are porous. Counterfeiters usually crack today's systems forcing companies to shell out regularly for expensive upgrades, so that if a malicious actor has access to the infrastructure where all the main components of the system are being managed, there would be a massive weak point that would be inefficiently solved spending huge amounts of money in updating or in system management. Following this alternative, Mass Serialization implies several processes that need to coordinate the communication with state agencies and administer medical countermeasures (MCMs) to the public.

Additionally, a critical weak spot is the barcode attached to the package. This is not the weak point itself, but since the whole identifying system is hidden from the user, if a counterfeiter has access to those codes the attacker can compromise them, creating new images or barcodes to develop decent forgeries. So, once the fraud is done, they will only have to slap those codes on the fake packages and slip them into distribution channels. Logically, there would be irregularities that will be managed at one point in the chain. Still, by the time that action is done, the counterfeiter would have already introduced new fake drugs in the chain that are lost in the process, and they will reach customers long before duplications are detected. Most worrying is that manufacturers must attach unique identifiers to product containers and enter the codes in a central database that the pharmacist later consults. However, vague FMD language [3] mandates only random container-level scanning, so for most products, wholesalers record only batch numbers which create a potential opening for counterfeiters. This drawback is also based on geographical features because regulations are applied to a single country or a group of them. As a result, global drug manufacturers do not follow regulations applied in the US or Europe uniquely, so a proper detection of a counterfeit is highly difficult or extremely inaccurate.

3.4.2 Tamper Proof Packaging

Another technique that tries to avoid counterfeiting in a physical way is the use of tamper proof packaging for drugs. Essentially, packaging is defined as the collection of different components which surround the pharmaceutical product from the time of production until its final use [4]. In order to achieve an optimal package it has to provide containment, protection, information, identification and convenience. If one of those features is non-existent, firms or groups involved cannot grant the authenticity and validation of the product. That is why when we talk about anti-counterfeit technologies in packaging, it is crucial the fight against:

- *Duplication*: label copying, packaging, instructions or usage information.

- *Substitution*: placing products in authentic or reused packaging.
- *Tampering*: altering packages and using spiked or stolen goods in place as real.
- *Returns and Warranty frauds*.

The number of anti-counterfeiting solutions is wide and many of the options are introduced with some sort of variation of another one. So it is easier to classify them into categories and extract the most significant examples with their advantages and disadvantages [4]:

- **Overt features**: they intend to enable end users to verify the authenticity of the pack with features that will generally be prominently visible and difficult or expensive to reproduce. They also require utmost security in supply, handling, and disposal procedures to avoid unauthorized diversion. Some examples are film wrappers that are wrapped securely around a product or container, shrink seals that are shrunk by heat or drying to seal the cap and container union, breakable caps, and sealed tubes. All those options "grant" the final user the authenticity of the product because the customer understands that the product was not changed during the supply chain process since the drug was manufactured. However, suppose a counterfeiter has accurate information about the sealing or wrap process. In that case, it is trivial to substitute the fake product with the real one, being undetectable to the final user.
- **Covert features**: the purpose of these features is to enable the brand owner to identify a counterfeiter product. So they are not visible to the final customers, and they will not be able to validate the drug's authenticity because they do not have the means to verify them. One of the most used systems is RFID (Radio Frequency Identification), which joins the tag, the reader, and the software to build a circuit that contains a unique tracking identifier. The reader captures the transmitted signal and provides connectivity between tag data and software. This process enables the product owner to validate the product through all the chain and get information even in the final stages. Nevertheless, RFID is a method that can be attacked from different techniques, one of them is the Tango Attack [22], which is based on listening to various iterations to break the protocol and find crucial values that will allow the attacker to find the shared secrets and impersonate the identifier. So, not only will the counterfeiter access the identifier to make new fake products, but the attacker will also know protocol information to build hidden deceptive products that are undetectable to final users.
- **Holograms**: a hologram typically incorporates an image with an illusion of 3-dimensional construction of apparent depth and particular separation. Holograms and similar optically variable devices (OVD) can be more effective when incorporated in a tamper-evident feature or as an integral part of the primary pack. This technique can make

unique sequential product numbering, making counterfeits easier to detect in the supply chain. The main disadvantages of sequential numbering are that the sequence is predictable and easily replicated, and end users require some means of access to the database. On the other hand, canes are used to make hidden marks or embedded images invisible to customers and difficult to see or even to make a copy of it, as well as laser coding. However, it requires really expensive equipment and results in recognizable artifacts which may be difficult to simulate. Despite being challenging to make a copy, it is not truly impossible. Some studies revealed that with reduced periods and relatively low cost, it is effortless to make new counterfeit labels [23].

	Effectiveness	Lab Cost	Time	Product Cost
Mechanical Copying	5	\$2,500	5 days	\$.05
Contact Printing	4.5	\$15,000	7 days	\$.05
Two-Step Copying	4	\$20,000	10 days	\$.05
Re-Mastering	3	\$25,000	20 days	\$.05
Simulation	1	\$ 0	0 days	\$.25

Figure 3.5: Effectiveness and Cost of Hologram Counterfeiting [4].

As we have seen, tamper-proof packaging is an interesting method to prevent counterfeiting in the pharmaceutical supply chain, providing strategies that physically imply the product and the customer. Final clients can "assure" the drug's authenticity by quickly consulting the package, labels, or stickers. Hence, these methods make life more difficult for counterfeiters, but they do not prevent irregularities from being committed that compromise the supply chain.

3.4.3 Track and trace technologies

This is the process of assigning an unique identity to each stock unit during manufacture which then remains with it through the supply chain until its consumption. Information is attached in the form of an unique pack coding, enabling access to the same information on a secure database [5]. As it is indicated in the Mass Serialization section 3.4.1, track and trace methods are highly used in pharmaceutical supply chains. Therefore, several techniques came up in order to improve the security and integrity of data which also provide physical traceability and real identification. However, this techniques were thought to be present in an operational environment where outsiders are not facilitated with methods that could assure

the veracity of the drug, so those identifiers where only useful to actors present in the supply chain, beginning with manufacturers, passing by distributors and ending with retailers. We can highlight several track and trace practices, exposing their utilities and drawbacks:

- **Pedigree:** This is a track and trace system of the drugs in a location. A drug pedigree is a paper document or electronic file that records details of the distribution of a prescription drug from its manufacture through wholesale transactions until it is received by the dispenser, which is usually a pharmacy or physician. Essentially, the person who gets that pedigree along with the drug shipment must verify that each recorded distribution took place and that the drug information is correct. The main goal of the pedigree system is to ensure that prescription drugs cannot easily be diverted or replaced with counterfeit products. However, through this process, the final person needs to verify the path and trace done by the drug, confirming its authenticity or discarding it. Even though we are dealing with a method that follows a measured approach in which every change of state or process is recorded, it is crucial to assure that the transaction cannot be reverted, faked, or deleted. In this case, those actions are probably done in a system where all transactions go through a central environment where only people involved in the chain are able to see that information. [5]
- **GTIN:** Global Trade Item Numbers are digitally unique identification numbers assigned by the manufacturer in accordance with GS1 allocation rules [24]. Parallel to this method, there is a technique called sGTIN (Serialized Global Trade Item Number). It uses unique numbers that identify a particular trade item, created by appending a serial number to the previous GTIN of the product. In fact, in the draft guidance for serialized identifier prescription drugs, the FDA proposed using NDC (a number that gathers the labeler's code, product code, and package code) combined with an eight-digit serial number. Nevertheless, we are dealing with complex serialization data (figure 3.6) that needs to be tracked in every transaction of the process by, probably, the need for potentially huge, multi-access databases. In terms of efficiency, this process is compromised and committed to a thorough process in which time is a crucial factor. That is why we are dealing with a method that would be useful for chains where product manufacturing rhythm is not high.

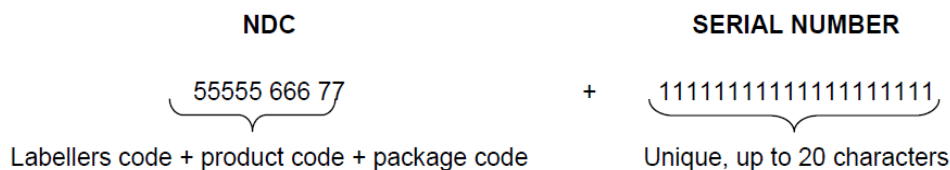


Figure 3.6: Example of a serialized US drug code [5].

- **Data Carriers:** They are graphical systems used to convey the product identifiers and associated information in a computer or human-readable formats [5]. This technique was previously mentioned in the Tamper Proof section 3.4.2. However, RFIDs and Bar codes are frequently used in Track and Trace actions. Following the disadvantages introduced in that section, we are dealing with two methods with high drawbacks for safety and validity reasons. Those methods were thought to be present in middle and extensive supply chains where processes were sealed to the final user and costs were over boundaries due to introducing those "high technological" identifiers in products (this is the case of RFIDs). Research done by Tata Consultancy Services concluded that those two methods implied more drawbacks than accurate functioning.

Features	2D BARCODE	RFID
Direct line of sight requirement	Yes	No
Difficult to duplicate or alter	No	Yes
Readability robustness (interference with liquids/metals)	No	Yes
Cost of tags	Low	High
Tag data storage	Low	High
Bulk tag reading	No	Yes
Initial technology set up cost	Low	High
Eco-system and/or standards maturity	High	Medium
Tag feature's extendibility (ex. Tag with sensors)	Low	High

Figure 3.7: RFID vs 2 Dimensional Barcode[5].

All of this track and trace technologies make the product unique in a way that the stages into the chain can know exactly the path of the medicine and its source. However, checking actions are forced to be done against centralized systems where the main information is not visible to the final user and clients have to deposit confidence in pharmacies or retail shops which might not have accurate data about that particular medicine. Moreover, these methods imply high costs in terms of tracking and permanence, that is, each identifier needs to be permanent through the chain and maintain its validity, even in stages where the checking actions were already done.

So far, from all previous sections (3.4.1, 3.4.2, 3.4.3), we have seen that, in order to guarantee a secure transaction of products through the chain, it is almost mandatory to implement complex and tedious systems that make the fake process difficult to counterfeiters. These implementations are not trivial to maintain, update or fix if a problem arises. Additionally, all these methods revolve around a centralized system that elaborates and manages most of the drug flow. That is why vulnerabilities of the main structure might result in failures related to the authenticity or integrity of products. The record of transactions, product data,

manufacturers' data, or client's data is stored in centralized databases, which may be exposed to attackers, and jeopardize the whole structure, even being able to make several malicious actions without being tracked or detected. Therefore, it is important to expand and distribute data from the supply chain structure, without the increase of resources usage or complexity management. Moreover, it is essential to introduce a technology that allows exhaustive monitoring of products without the danger of the information being manipulated or eliminated from the data chain. For all these reasons, new ideas are born that use Blockchain as the primary basis for technology development to obtain precise and detailed monitoring information.

3.4.4 DLT approaches

Distributed Ledger Technologies consist of electronic systems or data storage databases that are not executed by only one entity. This method allows users to store and use decentralized (stored in various places) and distributed privately or publicly [6]. However, Blockchain is a type of DLT with some particular features. It is also a shared database, but, in this case, it is structured by blocks that result in a chain. These blocks are closed by a cryptographic sign (hash); the next block is opened with that hash. This method allows information to be reliable, encrypted, and not easily manipulated. Essentially, altering one piece of information will change the contents of the block, which will require the alteration of every single other block. Such is the nature of distributed ledger technology, every node in the network will have a copy of this Blockchain, and therefore someone would need to simultaneously alter 51% of all nodes in order to falsify a record. This adds a layer of additional security over and above a manual system, where one central ledger is held by one trusted party.

In terms of the pharmaceutical supply chain, there are several firms and governments that have recently adopted Blockchain as a mean to track and trace products along the chain. Specifically, there is the example of the COVID-19 vaccine roll-out [6], which the main goal was that users could look at all previous data entries, touch points, locations and timestamps to trace all the way back to find the origin of the product, the specific manufacturer and even the specific batch that it came from. Blockchain would therefore allow products in the supply chain to be readily traced and verified (figure 3.8), and could enable easier detection and issues rectification. This in turn would allow faster and more efficient detection and removal of counterfeit, faulty or expired products from distribution.

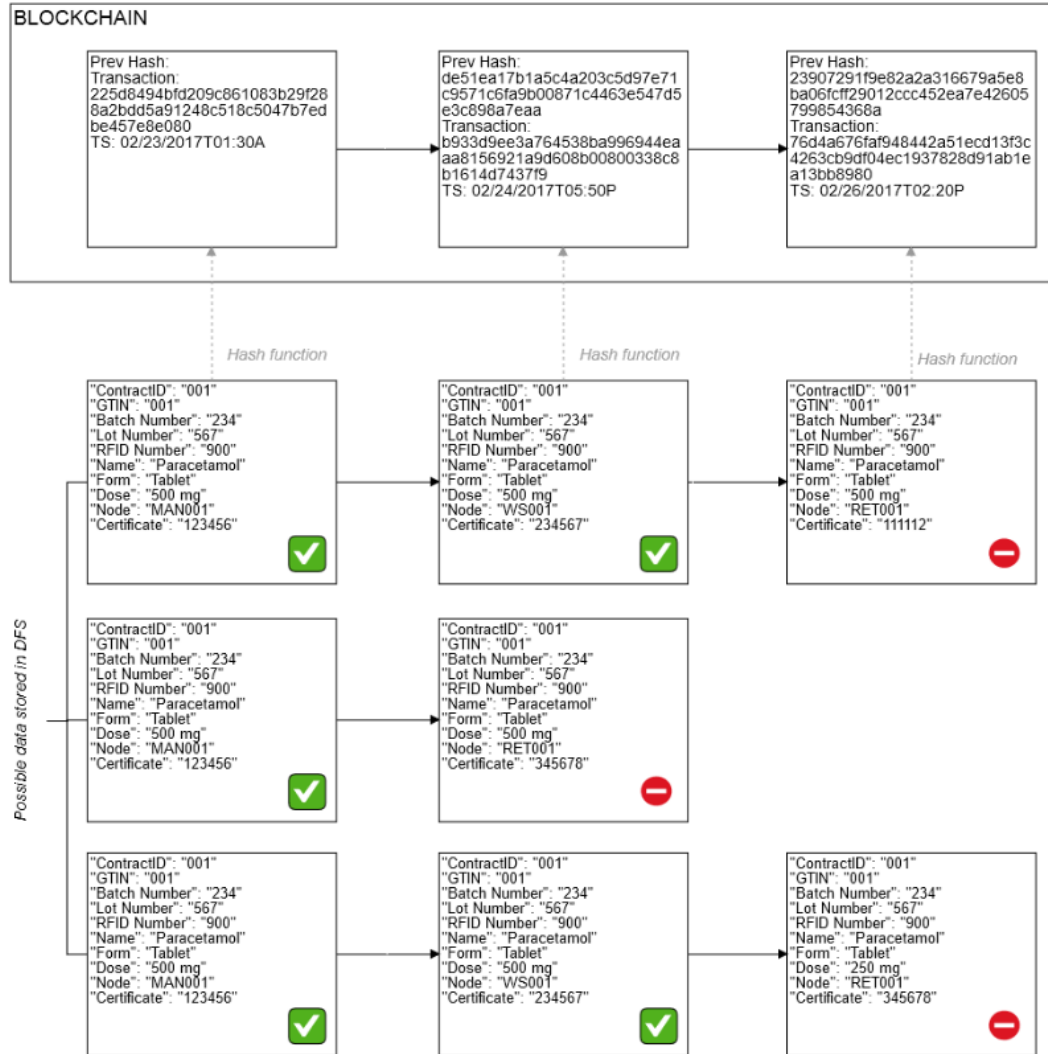


Figure 3.8: Example of how fake medicines are being detected through the chain [6].

Blockchain could also be used for monitoring the COVID-19 vaccine management, as it can be combined with the use of internet of things (‘IoT’) technology (e.g. temperature sensors). Temperature monitoring and storage time is essential for the COVID-19 vaccine distribution chain in particular, as different vaccines have different storage requirements. Blockchain technology could be used to monitor temperature and other important information during transport and storage of vaccine batches. These methods were adopted by the UK National Health Service (NHS) [25] who worked with firms like Everyware Ltd and Hedera Hashgraph that provided a system which information collected from Everyware’s temperature-sensitive monitoring sensors is logged on Hedera’s decentralized platform. The Hedera network “nodes” acted as the unbiased neutral party to coordinate and timestamp the

sensor data. The database exists across several locations and several parties, and is therefore not reliant on one single centralized authority. The information is then immediately available to any parties who have been given permission to access it. This provides a reliable and tamper-proof record-keeping system where integrity and trustworthiness of data is maintained and results are recorded, validated and shared across the supply chain in a cost-effective, secure and efficient manner.

Another early adopter is the MediLedger Project [25] which is made up of pharmaceutical manufacturers and wholesalers who are exploring the potential Blockchain to meet the Drug Supply Chain Security Act requirements for a track and trace system for US drugs by 2023. MediLedger counts some of the industry's largest pharmaceutical companies as its members and it has developed a Blockchain-based system to track prescription drugs across the supply chain to better tackle the movement of counterfeit medicines. The core function of the MediLedger Network is to validate the authenticity of drug identifiers throughout the supply chain, all of which can be done without any proprietary data being shared openly on the Blockchain and without it ever leaving a company's control.

Even though these solutions are highly innovative and beneficial for the treatment of medicines throughout the supply chain, it is important to highlight various drawbacks that appear during the product display and administration methods. In the processes followed by the NHS and the MediLedger Project, capabilities are offered to track medicines and know their path during their course within the chain. However, these identifiers that appear on the product packaging do not offer any type of utility for the end user, that is, the end customer cannot get to know any type of information, since the identifiers offer data to only manufacturers, distributors or owners of the structure. In addition, it is important to cover all the characteristics that allow identifiers not to be forged. In other words, if we introduce the ability for the identification element to be non-transferable, it will allow the repeated use of the identifier without the need to worry about whether there is replication of ids during different stages of the supply chain, so it is interesting to introduce NFTs as a mean to achieve that end. Also, because this is an early adopter of the Blockchain-based solution, it does not cover all the points that offer a global view of the medicine journey. That is, during the course of the product, information is collected and cannot be falsified due to the block-based structure, but there is no obvious relationship between the "owner" of the medicine manufacturer and the final counterfeit. In other words, if the product had all the complete information about the process, a negative act could easily be associated with it at the beginning of the chain and an exhaustive tracking could begin. For example, associating a manufacturer's wallet address to a medicine. In this way we will be able to know first-hand the representative or owner of the

final information that is shown to the client.

Planning and cost evaluation

4.1 Project Planning

Below is the planning of the project using a Gantt Chart, which has different phases of the project, including its duration, start, and end dates.

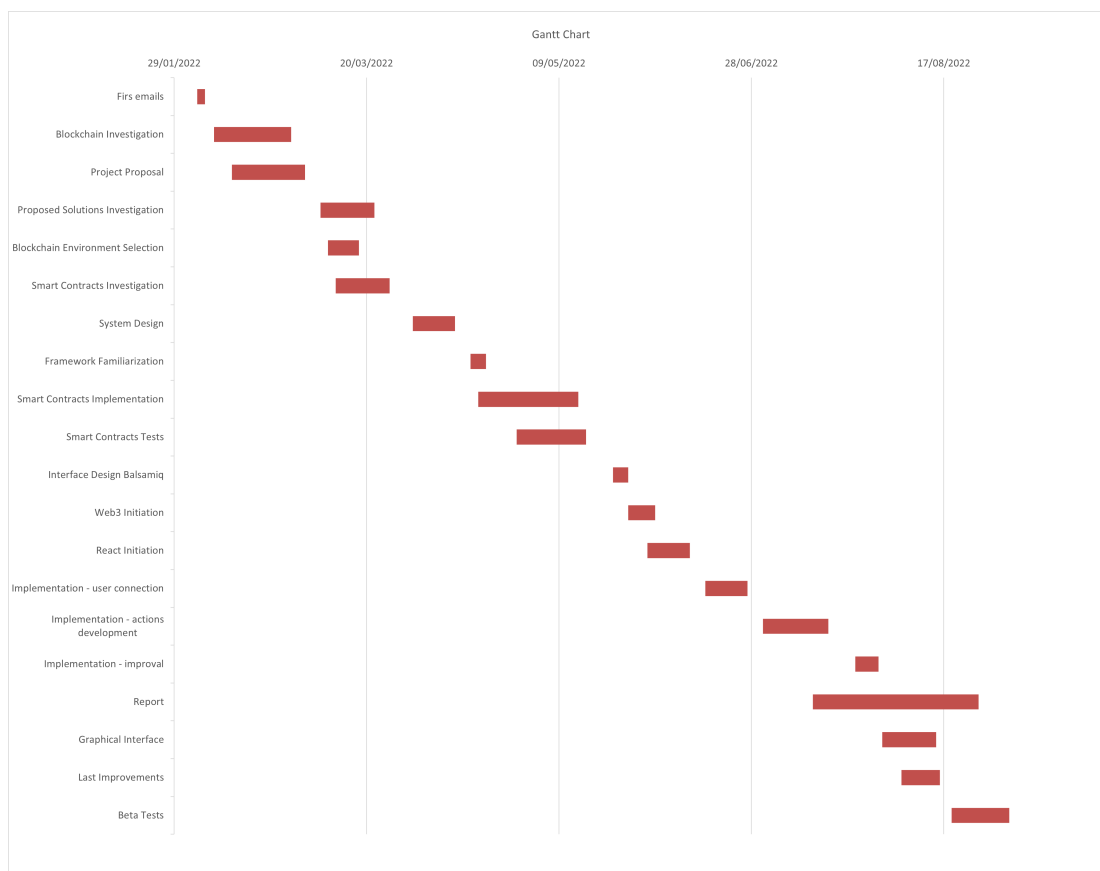


Figure 4.1: Project workflow visualized in a Gantt chart.

Now, all the tasks that were carried out throughout the project flow will be detailed:

- **First emails:** these were the first messages exchanged with the directors, detailing related projects and leading to a final idea for development.
- **Blockchain investigation:** this is a theoretical part in which research from multiple sources is carried out to collect as much information as possible from Blockchain and related elements. Specifically, the book *Mastering Blockchain* [11] was used extensively as a fundamental basis for the knowledge acquired.
- **Project Proposal:** it consists of the document drafting that will present the project as a whole, together with the objectives, methodologies, iterations and techniques used.
- **Proposed Solutions Investigation:** it is related with all the solutions currently proposed to the problem exposed in this project. It consists of a thorough investigation that tries to gather all the methods and techniques currently used to solve the decadent situation.
- **Blockchain Environment Selection:** this is the first task related with the system implementation. It is based on an analysis of different environments and their subsequent elections. There are several features that command the final choice such as cost, efficiency, transparency or maturity.
- **Smart Contracts Investigation:** this task is based on an investigation of how Smart Contracts work in a specific environment and how their transactions are validated. This task is directly related with the implementation due to its basic components and design.
- **System Design:** it consists of the charts implementation to illustrate how the components of the system interact with each other. Also, it implies the capacity of the system, as well as its behaviour in terms of modularity. This task is developed following design patterns and UML properties.
- **Framework Familiarization:** in this case Smart Contracts need to interact with a framework in order to be implemented. So, in this task frameworks like *truffle* will be analyzed to get used to its commands and capabilities.
- **Smart Contracts Implementation:** this is the task in which Smart Contracts are fully implemented. *Solidity* is the language used and *Remix* is an IDE that helps with the compilation and deployment of the contracts.
- **Smart Contracts Tests:** in order to implement efficient and secure contracts, tests need to be developed almost parallel to the contracts implementation. This tests will check all the functionalities of the contracts and their efficiency.

- **Interface Design Balsamiq:** phase dedicated to the design of the user interface. *Balsamiq* is the tool used to develop the mockups of each interface tab, improving its appearance and usability for the final user.
- **Web3 Initiation:** familiarization with the web3 library in order to interact with the Blockchain from an external application.
- **React Initiation:** familiarization with the *React* framework in order to implement a secure, modular and actual application that could be compatible with other environments and frameworks.
- **Implementation - user connection:** first part of the implementation, where the connection between the user wallet and the Blockchain is carried out.
- **Implementation - actions development:** it is the main part of the implementation. Here, all features related with contracts functions are developed. Its main goal is to interact directly with the Blockchain via read or write transactions.
- **Implementation - improval:** debugging of the functionalities of the previous section where there is an improval of each one of the interactions, incorporating extra functions. It also consists of adding more layers of security and data persistence to the system.
- **Report:** writing of the final report of the Bachelor's thesis.
- **Graphical Interface:** dedicated to an aesthetic improval of the web application.
- **Last Improvements:** the last details of the system are polished, optimizing functionalities and eliminating minor errors.
- **Beta Tests:** last testing phase where all use cases and devices on which the app runs are tested. It is the most similar test to a production process and it is also the phase that helps to more accurately describe the chapter of future works.

4.2 Costs evaluation

During the course of this project, no tools or methods involving monetary expenditure were used. In other words, *Open Source* frameworks and tools were used to facilitate the work and allow to carry out without any associated economic expense. However, a proprietary computer was used in all phases of the project. Its cost is around €2,000 if divided by its useful life, which is calculated in four years, that is, forty-eight months, we obtain €42 per month. Since the project lasted eight months (from February to September), we must include

an expense of €336. Although it is not exact, according to the previous figure 4.1, due to the thesis implementation in which its duration in days is indicated for each phase of the previous ones, and with a dedication of 5 hours per day of work, we obtain that the project took a total of approximately 1030 hours. From these hours, we must subtract the days when any work was not carried out, which were around 30 hours throughout eight months. Therefore we can conclude with approximately 1000 final hours invested in the Bachelor's thesis

Since no agreement is precisely adapted to the price/hour of computer science tasks in general, the following assumptions are made when calculating the total cost of the work. In addition, as one person only carried out the roles of manager, analyst, and designer, we will base ourselves on a programmer profile that would be the one who would have interacted with the system the most hours:

- **Programmer:** having to take care of the entire implementation and all the modules (React, Solidity, Web3, Truffle) as well as their tests, along with deployments from both parties. Some 450 hours dedicated to these tasks are estimated, so assuming a generic price for all of them of €20/hour according to the same criteria as in the previous cases, we would have: $450 \text{ hours} * €20/\text{hour} = €9,000$

Software Development Methodology

IN order to implement the software tool, there has been used the agile methodology Kanban, which splits the development process into iterations resulting with completely functional products.

5.1 Kanban

The Kanban Method is a mean to design, manage, and improve flow systems for knowledge work. The method also allows organizations to start with their existing workflow and drive evolutionary change. They can do this by visualizing their work flow, limit work in progress (WIP) and stop starting and start finishing [26]. Additionally, Kanban can be used almost in any knowledge work setting, and is particularly applicable in situations where work wants to be deployed as soon as it is ready, rather than waiting for other work items. Applying Kanban will embrace the following values:

- **Transparency:** information is shared openly using clear language that improves the flow of the product value.
- **Balance:** different viewpoints and capabilities must be balanced in order to achieve effectiveness.
- **Understanding:** individual and organizational self-knowledge of the starting point is necessary to move forward and improve.
- **Manage Flow:** The work flow in a service should maximize value delivery, minimize lead times and be as predictable as possible. Teams use empirical control through transparency, inspection and adaption in order to balance these potentially conflicting goals. A key aspect of managing flow is identifying and addressing bottlenecks and blockers.

- **Agreement:** Everyone involved with a system are committed to improvement and agree to jointly move toward goals while respecting and accommodating differences of opinion and approach.

5.1.1 Roles

In every Kanban project there are two roles who make different decisions and serve particular purposes [26].

- **Service Request Manager:** Understands the needs and expectations of customers, and facilitates the selection and ordering of work items at the Replenishment Meeting. This function is often filled by a product manager, product owner, or service manager.
- **Service Delivery Manager:** Responsible for the work flow to deliver selected items to customers. Facilitates the Kanban Meeting and Delivery Planning. Other names for this function include flow manager, delivery manager, or flow master.

In this particular case, as only one person manages the project development, the two roles will be gathered and performed equally. However, that is not a critical problem for the methodology. While the main features and iterations can be performed during the established time, the project will not suffer date damages.

5.1.2 Practices

In Kanban projects the following practices need to be assured in order to provide a successful workflow [26]:

- **Visualize:** mechanisms such as the *kanban board* are used to visualize the workflow and processes it goes through. Suppose we want to work as effectively as possible. In that case, it is needed to show agreements of workers to do a specific job, delivery points to the customer, policies in particular stages, and WIP limits. This visualization process will be achieved with the tool *kanbanize*, which offers a friendly interface that can help with the creation and management of iterations.
- **Make policies explicit:** Explicit policies help explain a process beyond just the listing of different stages in the workflow. Policies should be sparse, simple, well-defined, visible, always applied, and readily changeable by the people working on the service.
- **Limit work in progress:** When you establish limits to the amount of work you have in progress in a system and use those limits to guide when to start new items, you can smooth out the flow of work and reduce lead times, improve quality, and deliver more frequently.

5.1.3 Lifecycle

Since job items tend to flow through the system in a single piece flow, the best way to describe the lifecycle in Kanban is via the feedback stages involved [26]:

- **Strategy Review:** Selects the services to provide and the context in which those services are appropriate.
- **Operations Review:** Understands the balance between and across services, including deploying people and resources to maximize value delivery.
- **Risk Review:** Understands and responds to delivery risks in services.
- **Service Delivery Review:** Examines and improves the effectiveness of a service. This is similar to a retrospective that is focused on improving the Kanban system.
- **Replenishment Meeting:** Identifies items that the team will work on and determined, which work items may be selected next. This is analogous to a planning meeting for a sprint or iteration.
- **The Kanban Meeting:** A team working on a service coordinates their activities for the day. This is analogous to a daily standup.
- **Delivery Planning Meeting:** Monitors and plans deliveries to customers.

Some of those stages are thought to be executed monthly, quarterly or bi-weekly. However, since we are dealing with a project meant to be implemented individually, it is optimal to rearrange those stages and transform the dates to half of them or even a third. Following that schema, developing the project in time and with all the cycles performed successfully is realistic.

5.1.4 Workflow

As mentioned above, the work has been carried out following the agile Kanban methodology where the Kanbanize tool has been used to help with iteration management. Changes made from an original Kanban schedule are based on the number of project members. By only having one person, the two Managers have been united in a single worker. However, meetings regarding risks, delivery of services or strategy have been respected.

The iterations handling has been done through the "requests", "in progress" and "done" tabs. Where the original plans were followed to carry out the iterations consecutively. Each of these has subtasks that must be fully completed for the iteration to move to the next stage.

We can see in the figure 5.1 how these iterations were managed in a specific stage of the project.

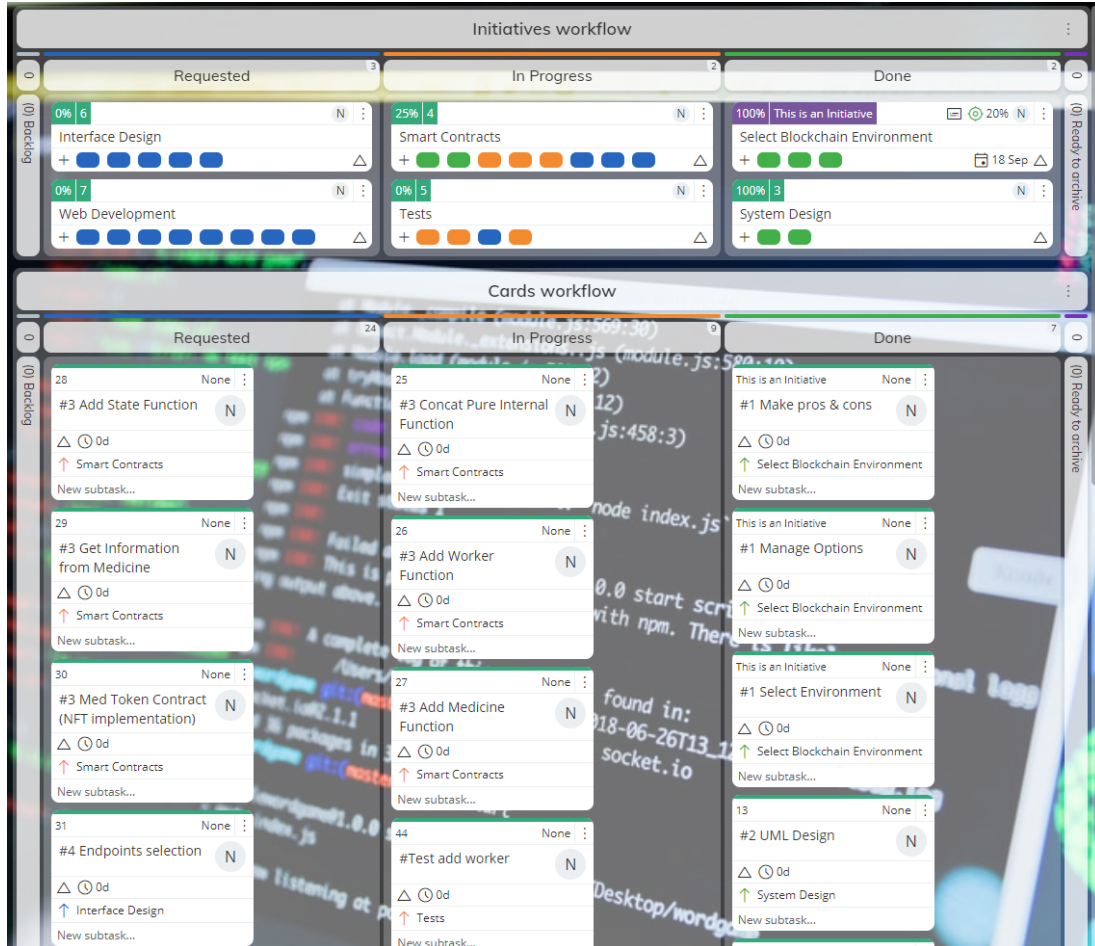


Figure 5.1: Part of the Workflow in Kanbanize.

On the other hand, version management was based on two branches in GitHub: the "dev" branch, which gathers all project development, and the "master" branch, which stores all ended versions of the software iterations.

Project Development

THIS chapter's purpose is to accurately detail the project's analysis, design, implementation, and test processes. The web tool will provide a friendly interface that every person can easily use, dividing several functions depending on the person that is executing the application. The development will follow the Kanban methodology (chapter 5), dividing the implementation into several iterations that will gather several functions and roles in executing the development processes. First, there will be an analysis and enumeration of requirements to help with the project's planning. Then, iterations are going to follow their natural process fulfilling those requirements. So, essentially, there will be a few iterations that will be part of the project backend, and the rest of them will follow the implementation of the frontend.

6.1 Requirements analysis

In order to start with the development of the tool, it is essential to carry out an analysis of the main requirements that will help us better understand the software tool's limitations.

6.1.1 Functional requirements

Those are the ones that the software must fulfill intrinsically. Due to we have an architecture similar to a client-server, we can distinguish several requirements depending on the part from which we see the application:

- **Interface:**
 - Add Worker to the Blockchain via register way.
 - Add Medicine to the Blockchain.
 - Add State to a medicine (state change into the Blockchain).
 - Visualize data from medicines.

- Visualize data from workers.
- **Service:**
 - Expose capabilities in medicine functions.
 - Expose capabilities in worker functions.
 - Gather functionalities from NFT creation to medicine addition.
 - Gather functionalities from visualization functions (medicines and workers).

6.1.2 Non-functional requirements

Completing the requirements above can be carried out in different ways, all of them in terms of software quality. It is here where non-functional requirements, those not associated with the goals of the final software, occur. In order to develop the software tool, the following requirements must also be met:

- **Usability:** Regarding the application's ability to facilitate its use by people outside the development of any kind. It must have an intuitive and easily understood design to meet the objectives so the end user can carry out their actions without difficulties or impediments.
- **Performance:** The objective will be that the end user does not have to wait long periods for an action to be executed and that the information to be displayed is displayed quickly without waiting for other processes.
- **Multiplatform:** The application must be able to run on any infrastructure without depending on its basic characteristics. In addition, the frontend can be migrated and run on top of another compatible Blockchain without worsening its performance.

6.2 Design and Implementation

6.2.1 Iteration 1: Blockchain environment selection

Within the options present in the development of Blockchain-based applications, there are many categories of environments in which Smart Contracts can be implemented. In addition, there are languages for its development ranging from Python, Rust, Java to Solidity. All these languages operate under a work environment that involves configuration and administration changes when designing and implementing Smart Contracts. However, in order to know the best possible environment, it is necessary to know the characteristics that will make possible a good integration, efficiency, and guarantee of operation:

- **Decentralization:** The vital point of projects based on Blockchain is the guarantee of precise decentralization and that no type of information depends on a central node, but rather that all the parts of the chain form a group and all of them, or most of them, are who carry out the actions that imply updating or adding data. We can talk about decentralization has to be full and total, because there are semi-decentralized environments [11] where intermediaries compete with each other, for example when multiple service providers compete to win a contract. That is not what we are looking for, the goal is to achieve with a system that any user can thoroughly check data directly to the Blockchain without the need to pass through and intermediary. Resulting in benefits like transparency, efficiency and cost saving.
- **Opening:** In Blockchain-based software development environments, it is essential to know the limits of the application itself. That is, for private or business systems, it is not necessary to use public Blockchains whose primary function is the possibility of access by any user. However, the environment's capacity to offer different options to developers to open applications to specific users or clients with a single purpose is vital. For this reason, open source environments are more likely to be used in production environments since they offer a wide range of options and possibilities without any economical added cost.

4 main types of blockchain technology				
	Public (permissionless)	Private (permissioned)	Hybrid	Consortium
ADVANTAGES	+ Independence + Transparency + Trust	+ Access control + Performance	+ Access control + Performance + Scalability	+ Access control + Scalability + Security
DISADVANTAGES	- Performance - Scalability - Security	- Trust - Auditability	- Transparency - Upgrading	- Transparency
USE CASES	■ Cryptocurrency ■ Document validation	■ Supply chain ■ Asset ownership	■ Medical records ■ Real estate	■ Banking ■ Research ■ Supply chain

Figure 6.1: Some types of Blockchain [7].

- **Cost:** In terms of ROI and profitability, cost is a highly important point that needs to be brought into focus. Actually expenses on the Blockchain is thought in points like maintenance where apps based on that environment could be switched to different platforms depending on their scalability, flexibility and confidentiality. In addition, it

is important to point out the expense that developers will make in the deployment of Smart Contracts and in the use of the implemented applications. If the application has very high costs when executing functions or actions, it will not be possible to expand the use of the tool to the whole world, and therefore it will be necessary to update the code or migrate to another development environment.

- **Development Tools:** Offering developers a wide variety of options is an essential requirement when it comes to hosting the implemented Smart Contracts. Many of today's environments carry out their development using programming languages such as Python, Rust or Go. However, the most important thing is not the language itself but the ability to be extended to other platforms without additional costs or difficulties.
- **Maturity:** The maturity time of a system is an exciting point, and it leads to various problems not being present in the short term. For example, environments like Ethereum have been in deployment for more than seven years and offer more stability than other environments. However, many recently emerging environments, such as Solana or Polkadot, use Ethereum to learn and start from scratch without making mistakes when ignorance was greater. Moreover, these new additions use the latest technologies to add to their systems in the development stages. At the same time, older environments need some restructuring to be able to integrate new technological methods into their solutions.
- **Efficiency:** Ensuring the correct functioning of the deployed code is vital for the platform's growth and its subsequent development. Some environments use various consensus methods for managing acceptance of transactions that directly implies an improvement or loss of performance in the execution of Smart Contracts. However, it is essential to correctly choose the environment that best suits the requirements of the application to be implemented, since if we do not have a high demand for transactions, a constant flow of them is not necessary. Furthermore, when we include multiplatform applications, the environment must be capable of migrating correctly, and the application needs to operate as close as possible to the previous environment or try to improve it.

Therefore, the goal was to find an environment that meets or at least satisfies most of these characteristics. We could highlight the development-based options of Hyperledger Fabric, Solana, or Tron. However, the project's objective does not match what these Blockchains offer at the implementation level. For example, Tron offers great interoperability between different environments. Still, due to its recent appearance [27] (2018), it does not present a friendly interface when interacting with Smart Contracts, a disadvantage that Hyperledger

Fabric also suffers from. Although it is a very stable, user-friendly, and secure environment, it operates in a rather unopened way [28]. It does not allow diversification when it comes to being able to expand Smart Contracts.

Due to these reasons, the option that we believe best suits the development, implementation, and deployment of the project is the Blockchain offered by the Ethereum environment. The first thing that was sought when developing the project was the possibility of interoperability between different environments to be able to execute the same Smart Contracts in different Blockchains. This function is offered thanks to the EVM (Ethereum Virtual Machine) which is an entirely isolated and sandboxed runtime environment. The code that runs on the EVM does not have access to any external resources such as a network or filesystem. This results in increased security, deterministic execution, and allows untrusted code (code that can be run by anyone) to be executed on Ethereum Blockchain [11]. Essentially, what happens in the deployment of the Smart Contracts is that every node in the network get a copy of the bytecode resulted from the compilation of the Smart Contracts. When an individual calls our Smart Contract to be deployed, all these nodes take a copy of the relevant Smart Contract, run its bytecode, and give the code to that individual. The result of this is the so-called "state change". It means that the current state of the Blockchain is changed, or not, depending on the purpose of action of the Smart Contract (writing or reading actions). So, tanks to this Virtual Machine the bytecode is able to be executed in several Blockchains that can offer different purposes. For instance, Avalanche or Cardano present greater speed, low transaction costs, and feature higher capacity compared to Ethereum, so in the end it is all about optimization and the best decentralized user experience in the crypto space.

Actually, Ethereum is not the Blockchain that will be used, but only that it is compatible with EVM will be enough to later be able to choose the deployment platform that best suits the needs of users and developers. This will also be offered to us by the programming language, which is Solidity. In addition, numerous tools are compatible with this language, making developing and implementing Smart Contracts easier. In addition, apart from choosing a specific Blockchain for deployment, we can also choose the characteristics of that environment. For example, since this project does not involve token swaps and end users do not make changes to the Blockchain at the write level, we can limit the amount of Gas in transactions, as well as the maximum capacity allowed. Logically we can also limit actions to users or allow access to only clients with permissions, although that is a decision in the use stage, not in the development stage.

In other words, the final decision has been the choice of Ethereum as the Blockchain

environment to be used during the development and deployment of Smart Contracts, not because of its transaction execution capabilities but because of its interoperability capacity with current tools, its maturity, and its wide range of options in terms of deployment.

6.2.2 Iteration 2: System Design

Once the production environment of the Smart Contracts is known, it is necessary to carry out a phase of design and recognition of the limits and characteristics of the system to be implemented. The main point is that the tool will have the Blockchain itself as its backend. That is, a large number of transactions will be validated per second; therefore, an overestimated execution of the application's actions should not be done. That is, we must carry out a simple structure design with activities that do not imply an overload of the system. In addition to the fact that we are facing the EVM under the Ethereum environment, it is necessary to use the Solidity programming language. So we need to adjust to the limitations of the language and design a good structure that can be maintained and flexible over time. The explanations of the design will be made under the sample of illustrations of the UML design, which will allow to better clarify the main characteristics.

- **NFT Class:** This class will be in charge of creating each of the NFTs that will be included with their respective medications. This class will have an integer identifier (Counter implies value types inherited from other contracts detailed in the development iteration of Smart Contracts). It will also have the primary function that will obtain as input the address of the manufacturer, this person being responsible for the drug. Moreover, it will receive an URL in the form of a string that will indicate the characteristics of the medicine. Finally, it will return the identifier that corresponds to the created NFT.

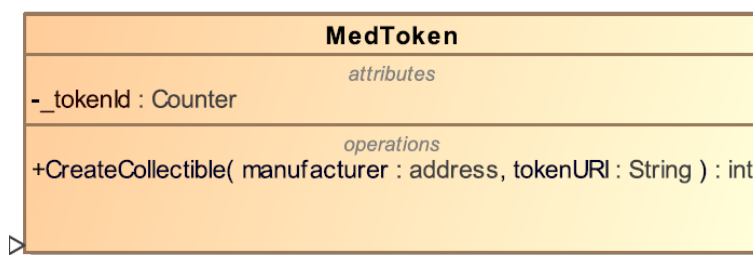


Figure 6.2: *MedToken* Class UML design.

The explanation of the decision made for this class is based on the recommendations given for implementing and developing OpenZeppelin NFTs, which basically consists of a standard to secure Blockchain applications. In addition to offering a good development experience and an outlet for handling and fixing problems in decentralized environments [29].

- **Storage structures:** Before designing the main class, we must know the types that will store the main information of the medicines and the workers/users. In addition, we must make it as simple as possible to not overload the Blockchain with information and directly influence its performance. First, we will have to specify the states through which the medicine will go, making an enum in which all attributes are private and static due to the characteristic of enums. On the other hand, the struct of workers will be made. Since we do not know the workers in the supply chain, we can group them all under the "agent" scheme and simplify the design, including the same types of parameters in each one of them. It will also be advantageous when searching directly for employees and obtaining their information. Finally, we get the primary structure, which corresponds to each medicine. Since it is the main one, it is essential to fill it as much as possible with relevant data to develop the application. In this case, we enter information related to the manufacturer, identifier, dates, states, and agents. All this information will be handy for the last deployment in the frontend part.

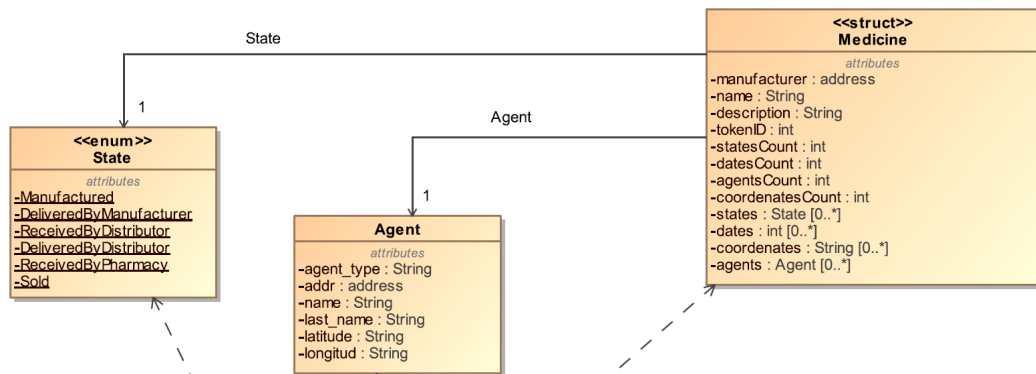
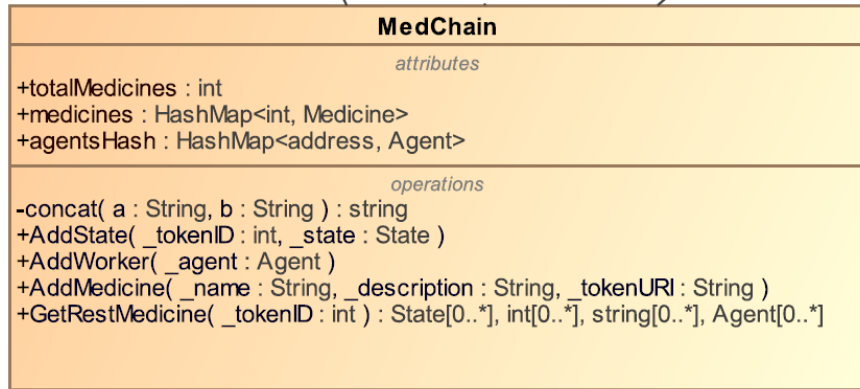


Figure 6.3: Storages structures UML design.

- **MedChain Class:** The main class will have some unique attributes that will be visible by making them public. Since the design is as simple as possible, a medicine counter and two HashMaps have been made to store the information on each medicine and each worker within the Blockchain. HashMaps have been chosen because they will be frequently required; therefore, we will need to access the data quickly and without having to iterate through lists. We will also have the main methods with which we can add new states to medicines, add workers to the HashMap, add drugs to the other HashMap, and obtain information about the medications that Solidity does not offer us directly. Finally, we have a private method that performs the concatenation of two strings and returns it. It is done directly since Solidity does not have a function that can do it, as of today.

Figure 6.4: *MedChain* Class UML design.

Actually, the class's public attributes are not visible outside of the class. That is, Solidity, for those attributes that are global and public, automatically implements a "get" function that returns that attribute. Therefore, the variable is not directly accessed, but the public method that reads the variable is really accessed [30].

- **Relationships:** For all these blocks to work together, relationships are needed to make them come together. The *MedChain* class will inherit the *MedToken* class to use its NFT creation method when creating a medicine, that is, in the *AddMedicine* function. The *MedChain* class will use the structs and the enum to store its data so that they will have usage relationships to the main class. Lastly, the *Medicine* structure will have cardinality relationships and the name of the attribute that makes it up. In this case, the attributes are the enum and the structure itself. It is seen in figure 6.5 how they connect all the relations with the different blocks.

It is important to emphasize that Solidity does not have classes as such, but each "class" would actually be a contract on which constructors or methods can be implemented, being able to inherit from other contracts or implement them (functioning as abstract classes and interfaces). Therefore, the UML schema can be easily adjusted to Solidity, although it is true that some relationships may change or work differently. Even so, from now on, we will talk about contracts instead of classes in order to clarify the code explanations.

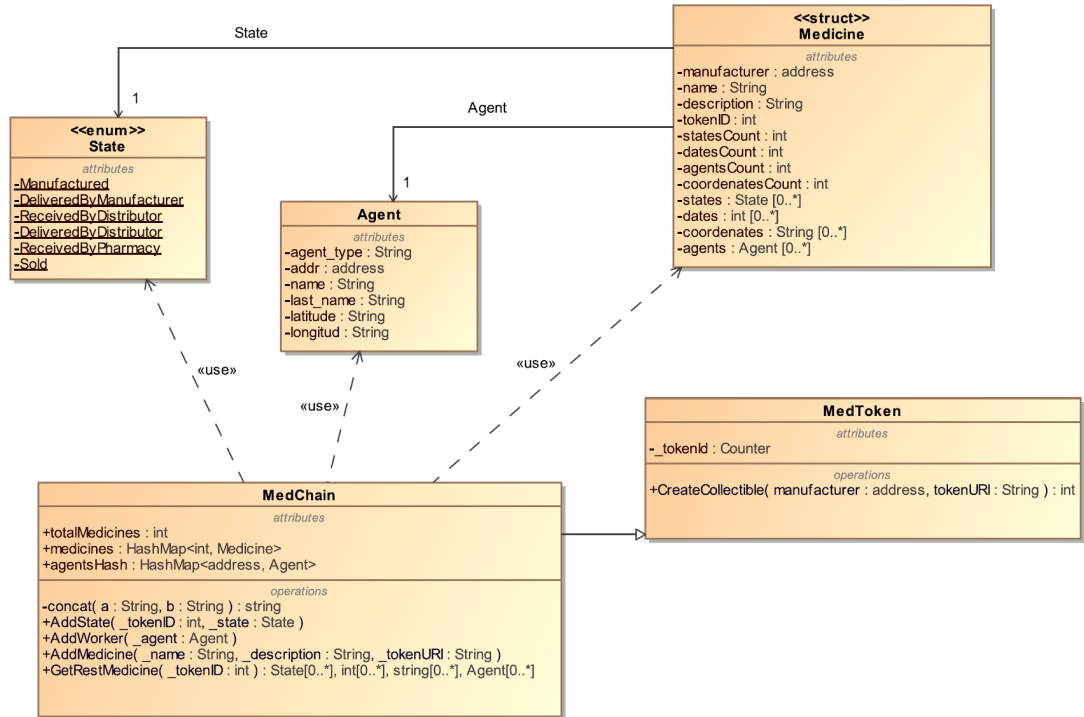


Figure 6.5: General System UML design.

6.2.3 Iteration 3: Smart Contracts

Once the application design has been determined, the implementation of the Smart Contracts will be arranged. They will have to be developed keeping in mind points related to the optimization, cost, and speed of transaction validation. Since we are in a Blockchain environment, each transaction needs to be validated before being entered, and later, it will be replicated for all nodes in the newly mined block. Therefore, it is necessary not to overload the system with useless or meaningless actions. It is required to optimize the functionalities of the functions as much as possible and choose the most appropriate types of variables to achieve almost constant performance. In order to visualize the code, it is accessible through this link: <https://github.com/javier-escribano/DecenPharm>

First, as explained in the design part (subsection 6.2.2), we will have two contracts that will be inherited, one over the other. The first is *MedToken*, the contract that will be in charge of minting the NFTs and the one that will have the weight of the medicine information outside the Blockchain environment.



```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

import '@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol';
import '@openzeppelin/contracts/utils/Counters.sol';

contract MedToken is ERC721URIStorage {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenId;

    constructor() ERC721('MedToken', 'MTK') {}

    function CreateCollectible(address manufacturer, string memory tokenURI)
        public
        returns (uint256)
    {
        _tokenId.increment();
        uint256 newItemID = _tokenId.current();
        _safeMint(manufacturer, newItemID);
        _setTokenURI(newItemID, tokenURI);

        return newItemID;
    }
}
```

Figure 6.6: *MedToken* contract code.

This contract basically consists of creating the token that will be associated with a particular medicine. It is simply a function that receives the address of the wallet of the person who owns the token and an URL with the medical information. Later, through OpenZeppelin [29] contract functions, the NFT is minted, and the information is assigned to it, also increasing the identifier number. Even so, it is important to emphasize several aspects that occur in the creation of the NFT. The owner will correspond to the medicine manufacturer and to whom the property value is attributed. The responsibility of ownership of the token falls on that person, and therefore, when obtaining information about the medicine, that person will appear in it as responsible for manufacturing and minting. Due to the particularity of the NFTs, they cannot be exchanged for others (fungibles), and, furthermore, we do not grant any action that may allow users a possible exchange of the tokens. That is, if a malicious user tries to change, forge or create a token, it will not be possible, even if they intercept the contract as a whole. The NFT will only be made once the drug is created and is not done again in any other part of the process.

On the other hand, the second parameter is an URL with the main characteristics of the medicine. This URL is not a standard one using the HTTP protocol, but the IPFS (Interplan-

etary File System) protocol is introduced. It consists of a distributed system to store access files, websites, applications, or data [31]. This system allows downloading files or viewing data from many locations that any organization does not manage. It basically consists of a peer-to-peer (p2p) storage network whose content is accessible through peers located anywhere in the world that might relay information, store it, or do both. IPFS knows how to find what you ask for using its content address rather than its location. In addition, each piece of information is unique within the decentralized storage, thanks to the CID (Content Identifier) that allows content addressing through hashes that enable content to be linked together. Additionally, it uses data structures such as DAGs (Directed acyclic graphs) or DHTs (Distributed Hash Tables) through which actions can be performed such as optimizing the representation of directories or files, breaking content into blocks, being able to refer to the same subset of data or perform requests to blocks quickly and efficiently. Finally, in terms of privacy IPFS itself isn't explicitly protecting knowledge about CIDs and the nodes that provide or retrieve them. This isn't something unique to the distributed web; on both the d-web and the legacy web, traffic and other metadata can be monitored in ways that can infer a lot about a network and its users [31]. Some key details on this are outlined below, but in short: While IPFS traffic between nodes is encrypted, the metadata those nodes publish to the DHT is public. Nodes announce a variety of information essential to the DHT's function — including their unique node identifiers (PeerIDs) and the CIDs of data that they're providing — and because of this, information about which nodes are retrieving and/or reproviding which CIDs is publicly available. Therefore, as can be seen, IPFS is a perfect protocol for storing NFT information. It does not break the harmony of decentralization that the entire system has since if the NFT data were stored in a centralized service, it would not make any sense.

Finally, the function will return the identifier that will be the number that identifies the NFT within the Blockchain in addition to the medicine itself.

The following contract is *MedChain*, which initially contains the structures to store all the information on medicines and workers.

However, in the beginning, there is a function known as an event. Events are inheritable members of contracts that, when called, cause arguments to be stored in transaction logs, which is a particular data structure in the Blockchain [30]. These logs are associated with the caller's address and are accessible in the block. Therefore, this event's function is to keep a kind of record apart from the public record of transactions. In addition, by having indexed values, we can quickly and efficiently access any transaction log.


```

// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

import "./MedToken.sol";

contract MedChain is MedToken{

    event Added(uint indexed index);

    enum State {
        Manufactured,
        DeliveredByManufacturer,
        ReceivedByDistributor,
        DeliveredByDistributor,
        ReceivedByPharmacy,
        Sold
    }

    struct Agent {
        string agent_type;
        address addr;
        string name;
        string last_name;
        string latitude;
        string longitud;
    }

    struct Medicine {
        address manufacturer;
        string name;
        string description;
        uint tokenID;
        uint manufacturing_date;
        string tokenURI;
        uint statesCount;
        uint datesCount;
        uint agentsCount;
        uint coordinatesCount;
        State[] states;
        uint[] dates;
        string[] coordinates;
        Agent[] agents;
    }

    uint public totalMedicines = 0;
    mapping(uint => Medicine) public medicines;
    mapping(address => Agent) public agentsHash;
}

```

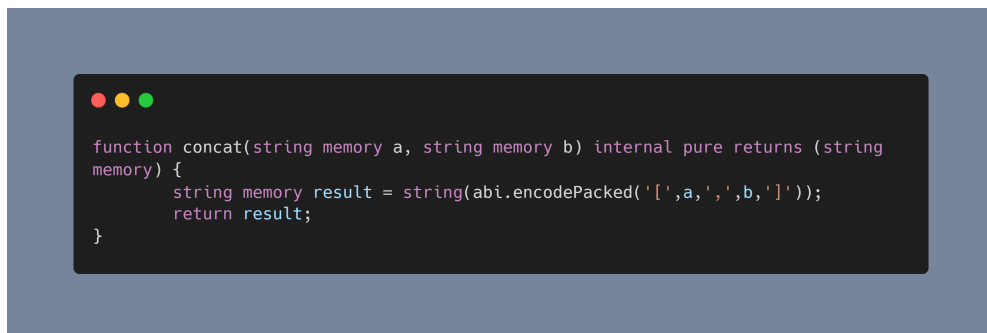
Figure 6.7: Structures in *MedChain* contract.

Next, we have an enum that will have the states a medicine can have. In particular, since this system is an example of a simple use case, we have indicated six maximum states for the drugs. Still, it is logical that these can be increased to improve the precision in the scope and tracking of the drug's steps during the supply chain. Finally, we have the two main structures in terms of primary data. The first consists of data referring to a worker. The design assumes that all workers have the same data types; thus, a more straightforward strategy can be achieved. Additionally, each worker will use this structure to store their information. We see that all the information is string type, and no one uses the byte type of Solidity because the maximum number of characters that the final fields can have is not precisely known. In addition, the agent's address is of type address, and thus we will be able to carry out later faster lookups in the main storage data structure. The other structure consists of quite similar

fields, but in this case, we have lists within the structure. These lists correspond to the number of states, dates, coordinates, and agents intervening in a particular medicine. These fields will be used to track drugs simply by looking at the information on the lists. In addition, it has been decided to use a list as the data type instead of a `HashMap`. Due to the peculiarity of Solidity, we would be unable to perform flexible actions on them, as is the case of returning the entire `HashMap`. When obtaining data, the performance would be significantly affected by iterating over the `HashMap` whenever the whole set is needed.

Lastly, there are the main `HashMaps` that were mentioned in the design section and that will have the information on medicines and total workers. The interesting thing to note is that the key to finding the medicines is an integer, that is, it is the identifier returned by the *CreateCollectible* function of the *MedToken* contract, hence its identity. The workers' key is the address of their wallet that they add when registering a new worker. Both values are unique and recent replicas cannot be added to either of them since if it exists in the `HashMap`, Solidity itself will throw an exception that will be caught. Lastly, we see that total medicine value that Solidity will "transform" into a getter function in order to access the value, just like with the `HashMaps`.

In other matters, we observe the "concat" function (figure 6.8) that receives two strings to concatenate them and return the final result. This function is performed to refine and fix the code for better understanding. In addition, the operation is internal, that is, it can only be accessed from the current contract or contracts derived from it. It cannot be accessed externally since it is not exposed to the outside through the ABI of the contract. Additionally, it is pure, meaning there is a promise not to change the state of the Blockchain once it is executed.



```
function concat(string memory a, string memory b) internal pure returns (string memory) {
    string memory result = string(abi.encodePacked('[' ,a,',',b,']'));
    return result;
}
```

Figure 6.8: *Concat* function.

Next, we can see (figure 6.9) the *AddWorker* function, which simply adds a new agent to the main `HashMap` of agents, which will be received as a parameter. The agent's address

is indicated as a key and is entered in the HashMap. If by any chance the same address is entered, what will happen is that the information in its position will be overwritten, so it is essential to know that the addresses must be unique and unrepeatable.

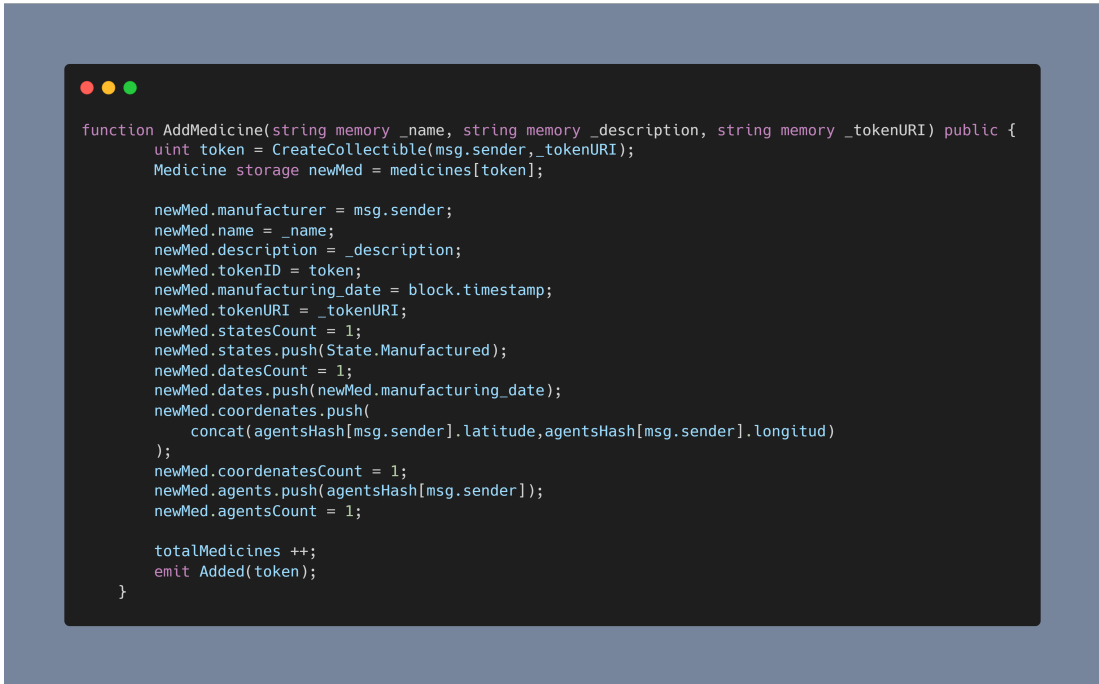


```
function AddWorker(Agent memory _agent) public {  
    agentsHash[_agent.addr] = _agent;  
}
```

Figure 6.9: *AddWorker* function.

A critical function is the *AddMedicine* function. It receives the medicine's name, description, and NFT's information URL as input parameters and returns nothing in the output. The URL will be made in the frontend, where the necessary data will be captured. The first part of the function corresponds to the formation of the token, the *CreateCollectible* function, inherited from the *MedToken* contract, is called and we add that number to the new medicine created of type *Medicine* struct. Then the drug will be filled step by step since Solidity does not allow it to be done simultaneously.

On the other hand, it is convenient to highlight different types that we see in the fields (figure 6.10), one of them is *msg.sender*, which is the address of the person who calls the contract, and we also see *block.timestamp*, which gives us the current date in epoch format. We then fill in the remaining simple fields with the parameter information and insert new fields into the lists. The new state will be "Manufactured" since it has just been created, we have already obtained the new date, the *concat* function is used to enter the coordinates simply so that they are then easily accessible, and finally, the agent information is added getting it from the main HashMap of agents. Finally, the global counter of total medicines is increased, and the event visible in the figure 6.7 is emitted, that is, the event will be triggered, and it will record the log that a medicine has been created with a specific identifier, which is the one that is introduced.



```

function AddMedicine(string memory _name, string memory _description, string memory _tokenURI) public {
    uint token = CreateCollectible(msg.sender, _tokenURI);
    Medicine storage newMed = medicines[token];

    newMed.manufacturer = msg.sender;
    newMed.name = _name;
    newMed.description = _description;
    newMed.tokenID = token;
    newMed.manufacturing_date = block.timestamp;
    newMed.tokenURI = _tokenURI;
    newMed.statesCount = 1;
    newMed.states.push(State.Manufactured);
    newMed.datesCount = 1;
    newMed.dates.push(newMed.manufacturing_date);
    newMed.coordinates.push(
        concat(agentsHash[msg.sender].latitude, agentsHash[msg.sender].longitude)
    );
    newMed.coordinatesCount = 1;
    newMed.agents.push(agentsHash[msg.sender]);
    newMed.agentsCount = 1;

    totalMedicines ++;
    emit Added(token);
}

```

Figure 6.10: *AddMedicine* function.

It is important to emphasize that, among all the functions that imply a change in the state of the Blockchain, this (figure 6.10) will be the most used, and therefore more transactions will be completed. Consequently, it is convenient to keep it as simple as possible and try not to carry out many validation configurations that imply a loss of performance. However, validations would still not be necessary thanks to the behavior of HasMaps and Solidity's own error handling. If, for example, there are agent existence errors or type assignment problems, Solidity will not allow the execution of the function and, therefore, of the transaction. We can catch these errors later, show them to the end user, and report the problem.

Of all the functions that involve a change in the state of the Blockchain, this is the last one, *AddState* (figure 6.11). This function receives as parameters an identifier of a drug and a status of the enum, it does not return anything at all. Still, its task will be to add that status to a particular medicine. This function is the most important in terms of drug tracking knowledge. The reason is that when adding a new status, relevant information is also being added to the follow-up, so it is vital to make configurations that validate in which cases the status must be updated and in which cases not. For this reason, we see (figure 6.11) how validation is already carried out to check the identifier as soon as the function starts. In this case, it has to be greater than 0, otherwise, there would be no medications to add states to.

```

function AddState(uint _tokenId, State _state) public {
    require(_tokenId > 0, "Token ID must be valid");
    uint count = 0;
    uint countStates = 0;

    for (uint i = 0; i < medicines[_tokenId].agents.length; i++) {
        if (medicines[_tokenId].agents[i].addr == msg.sender)
            count ++;
    }

    for (uint j = 0; j < medicines[_tokenId].states.length; j++) {
        if (medicines[_tokenId].states[j] == _state) {
            countStates ++;
        }
    }

    if (countStates == 0) {
        medicines[_tokenId].states.push(_state);
        medicines[_tokenId].statesCount ++;
        medicines[_tokenId].dates.push(block.timestamp);
        medicines[_tokenId].datesCount ++;
    }

    if (count == 0) {
        medicines[_tokenId].agents.push(agentsHash[msg.sender]);
        medicines[_tokenId].agentsCount ++;
        medicines[_tokenId].coordinates.push(
            concat(agentsHash[msg.sender].latitude,agentsHash[msg.sender].longitud)
        )
        medicines[_tokenId].coordinatesCount ++;
    }
}

```

Figure 6.11: *AddState* function.

This function in Solidity has the same behavior as an if statement that will return, in case the first field is false, an error message written in the second field [30]. Later there are two validations regarding the addition of information. The first corresponds to adding the state itself along with the current date. This validation goes through the list of drug states and sees if it already exists in it. If so, the new state is not entered. The second validation is performed because a single agent can enter more than one state; however, the agent's data cannot be replicated, so the list of agents is iterated, and the calling address is compared to the list's address, if it matches, agent information is not added. As we can see, we are performing iterations within the function, which is not a very common practice in developing Smart Contracts. However, in this case it is necessary since the main objective is not to overload the Blockchain, so if we introduce HashMaps in each drug instead of lists, we will cause each drug to weigh much more and involve a more significant amount of computation. For this reason, the decision was to use lists that are not going to be large, and therefore, the iteration carried out on them will be fast and will not harm the performance in the validation of transactions.

The last function included in the Smart Contract is *GetRestMedicine* (figure 6.12).

The image shows a code editor window with a dark background and light-colored text. The code is a Solidity function named `GetRestMedicine`. It takes a `uint` parameter `_tokenId` and returns four arrays: `State[] memory`, `uint[] memory`, `string[] memory`, and `Agent[] memory`. The function is marked as `public view`. It includes a `require` statement to ensure `_tokenId` is greater than 0, with an error message "TokenID must be valid". The function returns a tuple of four arrays, each corresponding to a property of the `medicines[_tokenId]` struct: `states`, `dates`, `coordinates`, and `agents`.

Figure 6.12: *GetRestMedicine* function.

This function has been made due to the limitations of Solidity when it comes to returning value types. In Solidity, you cannot return structures or objects that wrap them. The `medicines` structure has four lists that Solidity does not return by itself, so it is necessary to make a function that knows the type of lists it will return. In addition, a validation is performed as in the previous function (figure 6.11) so as not to perform a false search. Finally, we see that the function is of type `view`, that is, we will indicate to Solidity that the function is read-only and, therefore, there will be no change in the state of the Blockchain.

Throughout all the functions of both contracts we have seen that there are references to values that are passed as parameters, as returns, or within functions. They have "memory" or "storage" labels. These are called reference types [30] and they have to be handled more carefully than value types, because they will provide the data area where the type is stored. For instance, *memory* types lifetime is limited to an external function call, however, *storage* location is where the state variables are stored (structs, mapping, ...), where the lifetime is limited to the lifetime of a contract. So, for instance, in the *MedChain* contract there is only one *storage* reference type, it is shown in figure 6.10 where a new medicine was created. That drug has to remain permanently in the Blockchain as long as the contract is alive and it has to be present in external calls as well. Nevertheless, parameters passed in functions, or return types are values that are not going to be permanent in the lifetime of the contract so they will be of type *memory* and they will be cleared out once the external call is closed.

Indeed, making Smart Contracts to help carrying out the system's functions efficiently is essential. However, it is important to point out that these contracts must be cheap in terms of gas for their own use within the Blockchain. Gas limits the use of a function if it exceeds the imposed one. Logically it is advantageous if we see tasks that can perform infinite loops, causing the entire system to collapse. However, we are dealing with a system in which only

potential users can make changes to the Blockchain at the writing level, and final clients are the ones who only carry out reading operations. So we can play with the gas limits within our Blockchain to provide users with a more lasting and permanent experience. Although if there were certain functions that imply concurrent writing in time, the use of gas would have to be limited, as is the case with Ethereum in its leading network.

6.2.4 Iteration 4: Tests

These tests were carried out thinking about the main features that would be useful for exchanging information from the backend to the frontend. For this reason, it is essential to carry out tests that involve checking the type reference in the returned parameters or the possible non-definition of an element during its access. In addition, it is interesting to test execution means of detailed analysis and evaluation of Smart Contracts to assess the quality of their source during the development cycle. In this way, we will be able to discover bugs and vulnerabilities and reduce the possibility of software errors that can cause costly exploits. Also, Smart Contracts deployed on the Ethereum Virtual Machine are immutable by default.

```
it("Should not add a new state if it is already added", async () => {
  const medchain = await MedChain.new()

  await medchain.contract.methods.AddMedicine("Ibuprofeno", "Medicine based on inflammations", "")
    .send({ from: accounts[0], gas: 800000 })

  await medchain.contract.methods
    .AddState(1, 1)
    .send({ from: accounts[0], gas: 600000 })

  await medchain.contract.methods
    .AddState(1, 1)
    .send({ from: accounts[0], gas: 600000 })

  x = await medchain.contract.methods.medicines(1).call({ from: accounts[0] })
  y = await medchain.contract.methods.GetRestMedicine(1).call({ from: accounts[0] })
  countStates = x.statesCount

  assert.equal(2, countStates, "New state has been added incorrectly")
  chai.expect(x.datesCount).to.be.equal('2')
  chai.expect(x.agentsCount).to.be.equal('3')
  chai.expect(x.coordinatesCount).to.be.equal('3')
  chai.expect(x.statesCount).to.be.a('string')
  chai.expect(x.datesCount).to.be.a('string')
  chai.expect(x.coordinatesCount).to.be.a('string')
  chai.expect(x.agentsCount).to.be.a('string')
  chai.assert.oneOf('1', y[0], "Not found in list")
  chai.assert.oneOf('0', y[0], "Not found in list")
})
return go(f, seed, [])
}
```

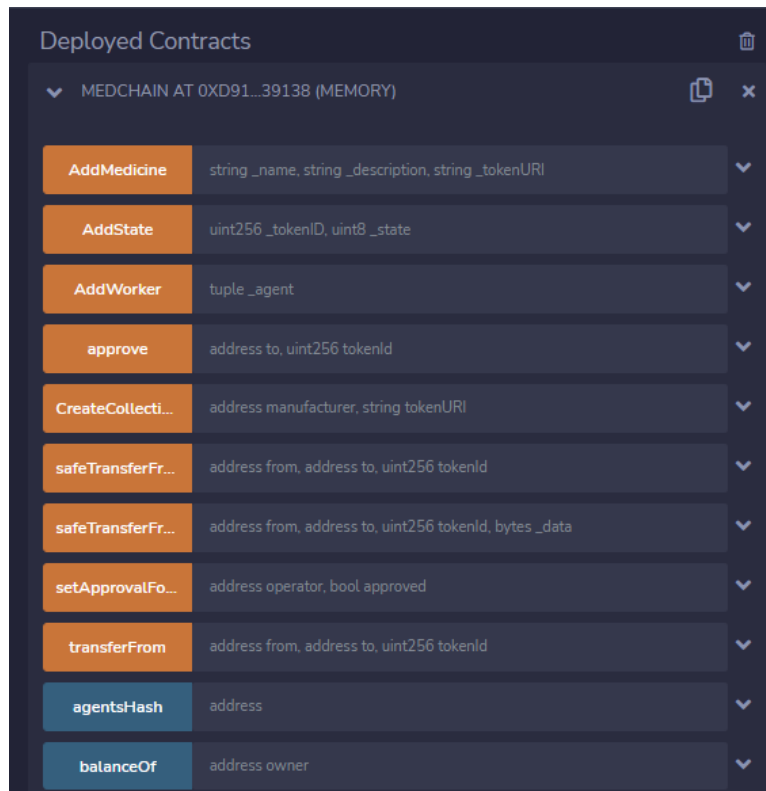
Figure 6.13: Example of an automated test function.

Therefore, unlike conventional development, where software fixes can be done after launch,

Ethereum development leaves little room for patching security flaws once the Smart Contract is deployed on the Blockchain. For the most part, upgrades should be considered a last resort and avoided unless necessary. Detecting potential vulnerabilities and flaws in the Smart Contracts during the pre-launch phase reduces the need for a logic upgrade. Following the agile methodology, tests were carried out during the implementation and development processes of the Smart Contracts. Actually, the integrity tests are not directly done since the joint functionality of both contracts is made only in one function. That is, they are carried out indirectly in the unit tests. These tests were performed following the guidelines promoted by the *truffle* framework and the *chai* library. More than 60 tests, including types validation, definitions, equalities, functionalities, and performance, were validated. We can see in the figure 6.13 an example of a function test.

On the other hand, given that at this point in the development of the project, the frontend has not been implemented, in order to know the correct performance and operation of the contracts, it is necessary to use test networks that facilitate the execution of the contracts and can be seen at first glance, errors, bugs or vulnerabilities. That is the case of Ethereum Testnets, these are networks used by protocol developers or Smart Contract developers to test both protocol upgrades as well as potential Smart Contracts in a production-like environment before deployment to Mainnet. However, an IDE called Remix allows the compilation, deployment, and testing of Smart Contracts through a graphical interface where each contract function can be interacted with individually. Unlike the previous (automated) tests, these are manual tests directly affecting a specific function and its displayed data. This part of the testing is significant since, apart from seeing how the contracts work, we can see at the bytecode level everything that happens with the transactions and messages that inform about how the transaction has been validated. So, for instance if we deploy the *MedChain* Smart Contract we will be able to interact with its functions directly as if they were buttons (figure 6.14). It is also important to mention that there are several functions that were not written previously like *transferFrom* or *balanceOf*, but those functions are part of every contract in Solidity, that is, it provide users with functions to get information from the Smart Contracts and help developers to write useful and efficient code [30].

So, if we interact with those functions, we will be able to get conclusions in terms of performance, cost and efficiency, only by looking to the information that the transaction offers to us. In order to get any conclusion from the transaction we are also able to debug it and go step by step to gather interesting data and see in which steps the system interacts with valuable or useful information.

Figure 6.14: Functions to test in *MedChain* Smart Contract.

For instance, in this figure 6.15, we are able to see crucial information about gas and transaction cost.

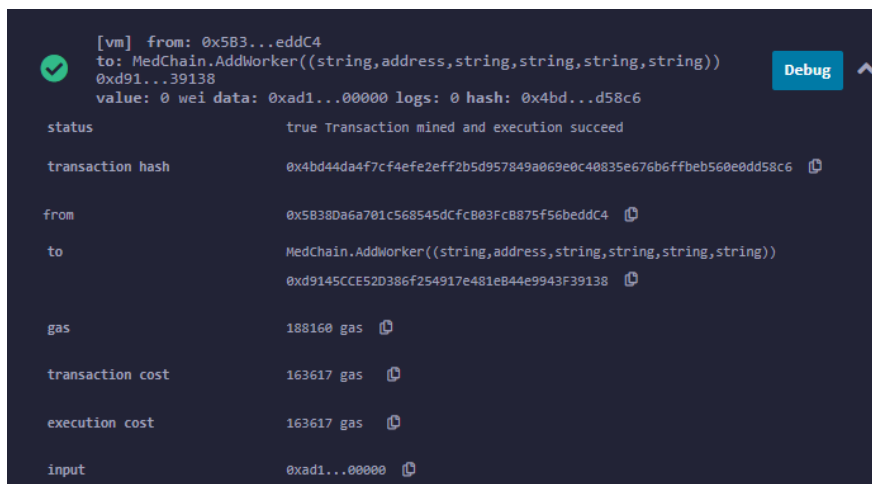


Figure 6.15: Example of gas cost in a transaction.

Having this values in mind, we are able to make some moves and understand how the

transaction works and if it needs to reduce the amount of effectiveness in order to improve the performance of the contract. In this case the gas limit provided for transactions is set to 3000000, so this transaction would only be executed with a 19% of that amount, what is highly favourable in order to maintain an optimal and efficient environment, even with huge amounts of transactions being executed.

So, in order to test the efficiency and performance of the functions, we are able to perform those transactions consequently and store information about gas in terms of execution cost. That way we will be able to know exactly how those functions perform and if they are optimal to be included in the final version of the code.

Functions	AddWorker	AddMedicine	AddState	Agents	Medicines	GetRest
Performance (Gas)	163617	611077	93579	40635	46868	63348
Percentage	5,45%	20,37%	3,12%	1,35%	1,56%	2,11%

As we see in the table above, there is a function that needs a more considerable amount of gas to execute the transaction, *AddMedicine*. This is understandable because, at first, we call a function external to the contract, so it needs no locally accessible information. Secondly, this function assigns a storage type, which implies that the fixed type is kept constant in the Blockchain as long as the contract is available, as we saw in the implementation and development section 6.2.3. Nevertheless, the rest of the functions work with a tiny amount of gas, that is, in terms of performance and optimization, the functions work ideally. Especially in the case of *AddState*, which, despite having several iterations, by not having to make long runs, it achieves a gas use of just 3%, in the same way that happens with *AddWorker*. In addition, we see how effectively the functions that do not imply a change in the state of the Blockchain do not require a large amount of gas, the only one that exceeds 2% is *GetRestMedicine* because it does not return simple types but rather it returns complex types such as the case of enum lists or simple lists.

In order to visualize better the performance we can see in the figures (6.16, 6.17) how those contracts perform in a simulated real environment.

We can verify (figure 6.17) that the performance of *MedToken* concerning *MedChain* is relatively low (we see the comparison since figure 6.16 is in hundreds of thousands and figure 6.17 in tens of thousands). This is because the function carried out in *MedToken* does not imply an excellent processing capacity and does not access external values outside the contract. It simply performs actions of the packages it imports, which is OpenZeppelin's property [29].

All in all, we have seen throughout the iteration how testing Smart Contracts is extremely

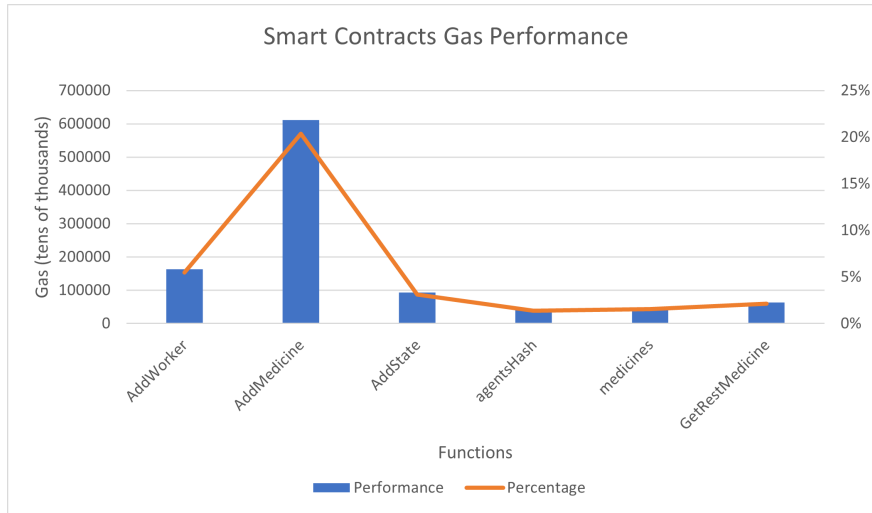


Figure 6.16: *MedChain* performance. Gas vs Functions.

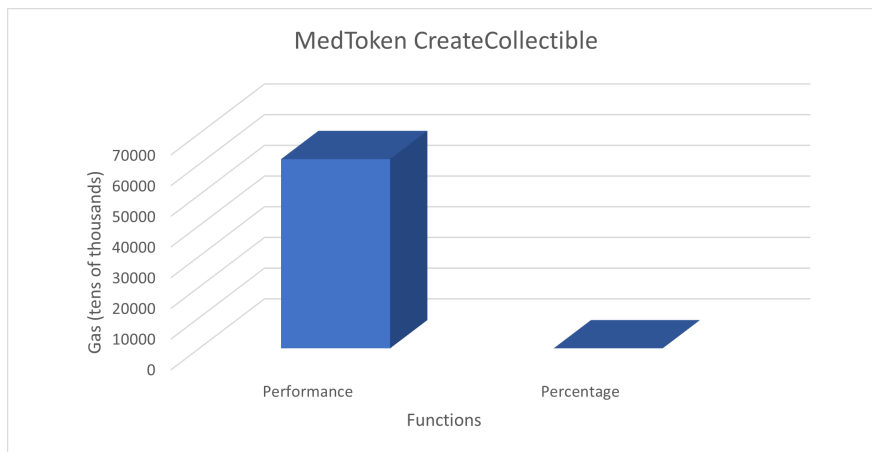


Figure 6.17: *MedToken* performance. Gas vs Functions.

important to know how the code operates and to be able to safely launch and deploy bugs, errors, or vulnerabilities. This will give us enough confidence not to deploy the contracts again and lose confidence, security, and economic assets regarding the cost of gas or tokens.

6.2.5 Iteration 5: Interface design

Once the fundamental part of the system has been completed, it is necessary to implement a friendly interface that allows users to interact with the application easily and safely. The interface design will be based on how a fundamental user can interact and how a user with experience in the use of computer systems can manage them. In other words, the interface must be divided to provide different services to people who interact with the application, aiming to increase its security and not compromise its integrity. For example, a base user, a client, should not have access to the services of creating medicines, adding statuses, or assigning NFTs. In the same way, for example, a distributor should not be able to have access to a manufacturer's own shares, with all that this entails. In addition, the interface components must be simple and not entail a notable effort for the user. Otherwise, it will not be able to adapt to the environment, and it will not be trivial to carry out the most basic actions. Therefore, for the interface design, only three views have been used, which communicate directly to the user what he has to do at all times.

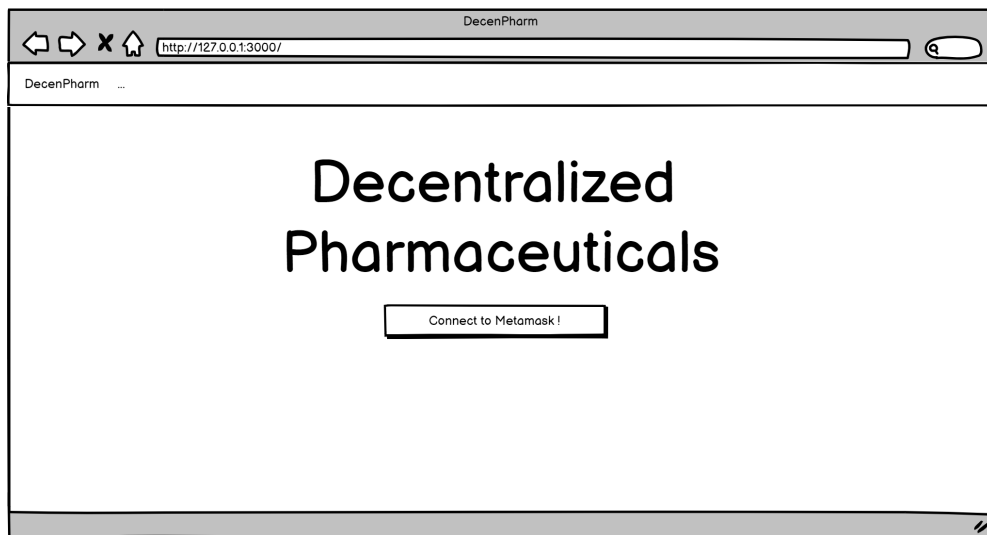


Figure 6.18: Mockup for the home page.

As it is a Blockchain-based application, the user must have extra ease when starting to perform functions since it is not a typical application where registration is based on mail or something similar is produced. In this case, the user needs to have a wallet that is assigned

to an address. This address will be the unique identification method for users, and they will have to connect to the application through an indicated button, granting the permissions that the application needs to be able to operate with the corresponding extension. Ideally, the application is designed to work with virtual wallets based on Metamask since it is the most widespread in terms of applications based on Smart Contracts, and its extension offers many simple options for the user to become familiar with the system. Therefore, the user will have to click on the button indicated in the figure 6.18, and the extension will begin to make the connection between the application and the Blockchain network (obviously, the user has to have access to the network in some way, either locally or remotely).

Once the application is connected to the Blockchain, each user is identified and filled in with information, so a registration window is displayed.

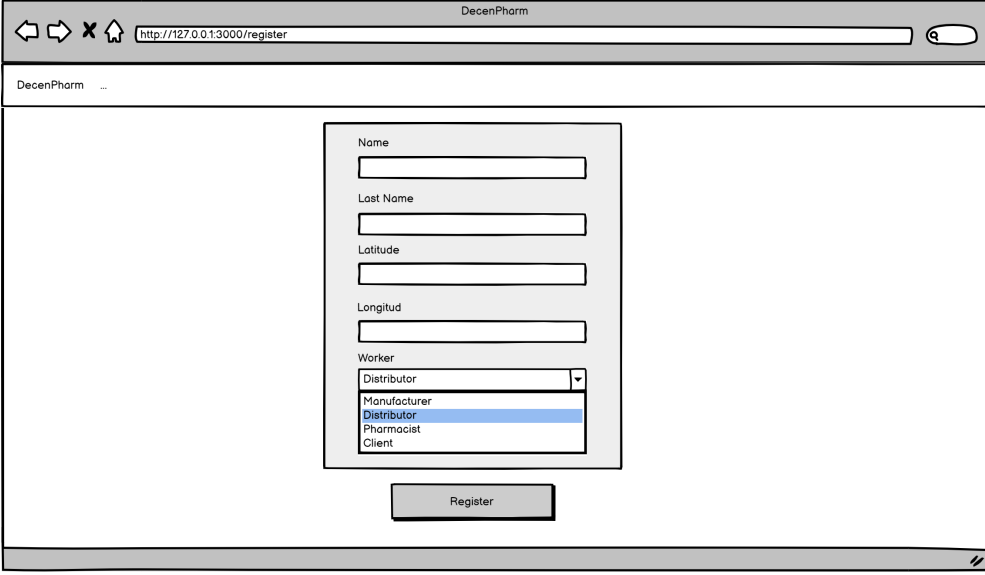
The image shows a web browser window titled "DecenPharm". The address bar contains "http://127.0.0.1:3000/register". The page content is a registration form with the following fields: "Name", "Last Name", "Latitude", "Longitude", and "Worker". The "Worker" field is a dropdown menu with options: "Distributor", "Manufacturer", "Distributor" (highlighted), "Pharmacist", and "Client". Below the form is a "Register" button.

Figure 6.19: Mockup for the register page.

As we can see (figure 6.19), basic user information is included in the registry, but also relevant data for drug tracking, such as the coordinates. Assuming normal user behavior, this information will be correct in most cases, however, if a user performs any illicit action in filling in data, it will not be very relevant since basic users do not intervene in product tracking. For this reason, and an improvement in usability, a drop-down form has been provided in which the type of worker can be chosen, although, in a business environment, this form would not really be accessible, and part of the code would have to be enabled for people authorized and restrict another part of it.

Finally, we see (figure 6.20) the design of the application's main page, which any user will

have access to and will be able to see the follow-up of the medicines together with all their information, as well as information from the workers or clients themselves.

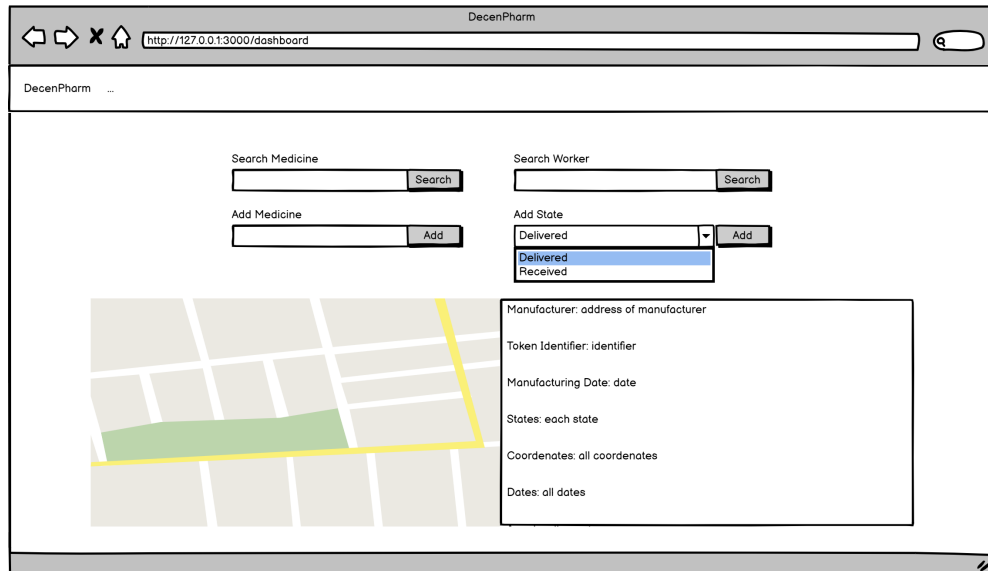


Figure 6.20: Mockup for the main page.

On this page, the user will be able to visualize the course of medicines through a map where each one of the steps that they carried out will be indicated with markers. In addition, on your right side, you will have a box where all the information corresponding to that medicine will be revealed: coordinates, states, people involved, dates, identifications, and an IPFS link with which you can corroborate the medical information in a decentralized storage system. However, this page would not be the same for all users, as discussed earlier in this iteration. Specifically, this page would be the one that manufacturers would see since they are the only ones that can perform the function of adding medicines to the system. Nevertheless, they are not the only the ones that can add states since pharmacists and distributors would also be able to do so. The clearest example would be clients, who would not have access to those two forms of adding medicine and adding status, but could simply search for drugs or workers. This information display layout has been chosen since both options (map and data) have a similar relevance when it comes to informing the user. Even so, the map has options to zoom in or out, so displaying information is not a problem. In addition, a column layout with enough space between the data has been chosen so that the user does not get tired when reading it and gets overwhelmed with all the information displayed. This way, they can easily redirect the view to specific data without conflicting with another type of information.

Through this iteration we have seen how the decisions made for the development of the interface design are, where simplicity for the user and the usability of the components that

make up the interface prevail. It is essential to add that this system is not made so that the user performs actions in a complex way, but that complex actions become trivial even for the most experienced users.

6.2.6 Iteration 6: Web development

During this iteration, the development carried out in the web interface implementation will be explained in some detail, where React has been used as a framework to speed up development. Throughout the process, the integration of the Blockchain itself with the interface and its joint action has been thought about. For this reason, there are some choices whose basis lies in that specification.

In order to develop the interface, the MVC (Model View Controller) software architecture pattern has been followed to better organize how the interface communicates with the model, the Blockchain. Since the model itself is not developed in this iteration, not much will be said about it, although it is interesting to mention that the model itself performs updates in the interface since it is the data itself that is manipulated during the course of actions in the application (figure 6.21).

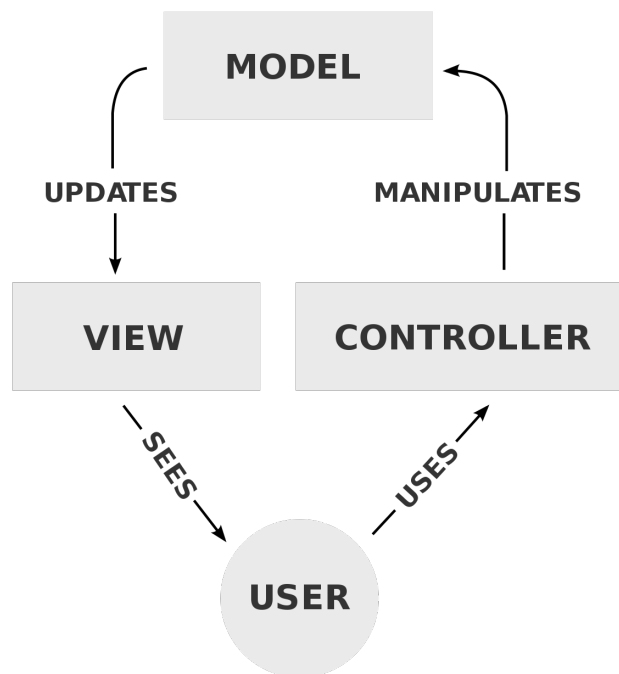


Figure 6.21: Model View Controller pattern graph [8].

- **Controller:** As we see in the figure 6.21 the controller is the one that users use and which manipulates the model, so in this application that will be the task of the React Hooks, that is, the only thing they will do is to develop the logic that the system will follow to connect React components (dashboards, navbars, ...) to the Smart Contracts and be able to execute their functions. For example, the *useConfiguration* Hook performs all the interactions of the application with the Smart Contracts thanks to the use of the Web3 library provided by JavaScript. This library allows us to indicate where the Blockchain is running from (being able to suggest other networks such as the main Ethereum network), in this case, we use Ganache to generate test networks. From this interaction, we obtain the ABI of the contracts, which will mainly give us all the information and be able to execute the functions that reside in the contracts.

A screenshot of a code editor with a dark background and light blue text. The code is as follows:

```
const web3 = new Web3('http://127.0.0.1:7545')  
  
const medChainData = MedChain.networks[5777]  
const medchain = new web3.eth.Contract(MedChain.abi, medChainData.address)
```

Figure 6.22: Function connection to the network.

Once the controller already has access to the network, we simply have to connect the functionalities of the interface with the Blockchain, for this, we will use functions that use this data and interact directly with the network. For example, when registering a user, we access that function and pass the indicated parameters, either by the context, by the input, or in another way. This way and thanks to the ABI, direct communication with the Blockchain is being carried out without intermediaries between requests. ABIs are a standard way of interacting with contracts in an Ethereum ecosystem, either from outside the Blockchain or by interacting between contracts [30]. Data is encoded depending on its type, and a subsequent scheme is required in order to be decoded. Basically, a mapping of types, functions, actions, and assignments to bytecode values that the Blockchain is capable of processing is performed. In this way, by completing the decoding efficiently, the Blockchain network is able to process and return information quickly and accurately.


```
const registerUser = async (
  account,
  agent,
  name,
  surname,
  latitude,
  longitude
) => {
  await medchain.methods
    .AddWorker([
      agent,
      account,
      name,
      surname,
      latitude.toString(),
      longitude.toString(),
    ])
    .send({ from: account, gas: 200000 })

  setAuth({
    address: account,
    agent: agent,
  })

  navigate('/dashboard')
}
```

Figure 6.23: Controller function to register an user.

- **View:** The realization of the view has been involved by the primary endpoints of the application, which are the main page (/), the registry (/register), and the dashboard (/dashboard). Each of those views has a component assigned, in addition to the other helpers to the main ones.

```
<Routes>
  <Route exact path="/" element={<HomePage />} </Route>
  <Route exact path="/register" element={<Register />} </Route>
  <Route exact path="/dashboard" element={<DashBoard />} </Route>
</Routes>
```

Figure 6.24: Routes for all the main components.

The main component on which users can perform actions interacting with the Blockchain

is "Dashboard". This component has the drug and worker search forms, as well as drug and status addition forms. In other words, this component renders the map on which the route of the medicine will be seen, as well as the display of information on the drug and the corresponding worker. However, the dashboard does not have actions visible to everyone, that is, in the case of pharmacists, they will not have access to the medicine addition form, although they will have access to the status addition form. This discrimination will be made using the *Visible* component of React, in which different roles will be assigned to user types and, thus, be able to divide the actions that each of those roles can do. However, from a security point of view, it is not the best option if you want an open deployment in which everyone can access the application. For extreme security, it would only be necessary to give access to pieces of code according to the role that the person acquires. For example, the application that the client downloads will not be the same one that a manufacturer has due to the fact that it has fewer features. This will allow users only to perform those actions that have been assigned to them. And if for some reason, you try to carry out an activity pretending to be someone else, it would be tremendously complicated since you would have to bypass your own virtual wallet and carry out the actions with the other person's address. For this type of action, Metamask itself has mnemonic coding systems that create seeds from a sequence of English words that are easy to transcribe but highly difficult to replicate [32]. However, to better visualize the options of each worker, it has been decided to use the "Visible" component and gradually change what each user can visualize according to the assigned role.

The screenshot shows the DecenPharm web application interface for a manufacturer. At the top, the browser address bar shows 'localhost:3000/dashboard'. The application header includes the DecenPharm logo and a 'Connected' status indicator. Below the header, the user is logged in as 'Connected as manufacturer'.

The main interface is divided into several sections:

- Search and Add Forms:**
 - A 'Search Medicine' form with a text input containing '1' and a 'Search Medicine' button.
 - A 'Search Worker' form with a text input containing 'ej: 0xe9cCC29CEba70C7A6F1468C68Dcb6800f' and a 'Search Worker' button.
 - An 'Add Medicine' form with a text input containing 'Ibuprofeno' and a 'Medicine that...' label, with an 'Add Medicine' button.
 - An 'Add State' form with a 'Token ID' input and a 'Delivered by manufacturer' dropdown menu, with an 'Add State' button.
- Map:** A map of Galicia, Spain, showing various cities and a red location pin.
- Medicine Information Table:**

Medicine Information	
Manufacturer	0x0dEEBb0a6F4C454AEaF83c64275c044208e47912
Identifier	1
Name	Ibuprofeno
Description	Medicine that...
Manufacturing date	8/27/2022, 2:48:48 PM
NFT Data	https://bafkreihp2wmvraillahkurirc3433kbqhre65ffxh5eioxf03ueyuo676q

Figure 6.25: Example of forms available for a manufacturer.

The screenshot shows the DecenPharm dashboard. At the top, it says "Connected as client". Below this, there are two search bars: "Search Medicine" with the value "1" and "Search Worker" with a long alphanumeric string. The main content area is split into two parts. On the left is a map of Galicia, Spain, with a red pin on the coast. On the right is a table titled "Medicine Information".

Medicine Information	
Manufacturer	0x0dEEBb0a6F4C454AEaf83c64275c044208e47912
Identifier	1
Name	Ibuprofeno
Description	Medicine that...
Manufacturing date	8/27/2022, 2:48:48 PM
NFT Data	https://bafkreihp2wmvraillahkurirc3433kbqhre65ffxh5eoi3fo3ueyuo676q
Number of States	1
States	Manufactured
Number of Dates	1

Figure 6.26: Example of forms available for a client.

We can see, thanks to the figures 6.25 and 6.26, depending on how the clients register in the application, they will see fewer forms or will be able to make more occasional changes. However, all search operations have been maintained for any type of user to facilitate the understanding of entered data and to carry out actions to add states to the drug having a reference to the data already added to the drug. Moreover, we can see an example of how a function is only visible for a specific role (figure 6.27).

```

<Visible auth={auth} role="USER">
  <form onSubmit={searchWorkerValidation}>
    <input
      placeholder="ej: 0xe9cCC29CEba70C7A6F1468C68Dcb68005f6A882D"
      className="shadow w-2/3 appearance-none border py-2 px-7 text-gray-700 leading-tight
      focus:outline-none focus:shadow-outline"
      onChange={(e) => setSelectedAddress(e.target.value)}
    ></input>
    <button
      type="submit"
      className="shadow w-1/3 bg-gray-800 hover:bg-gray-600 focus:shadow-outline focus:outline-none
      text-white font-bold py-2 px-4"
    >
      Search Worker
    </button>
  </form>
</Visible>

```

Figure 6.27: Example of Visible Component inside a role.

Finally, it is necessary to communicate to the user the status of the actions, that is, if they were successful, failed, or something unexpected happened. In most cases, the

actions of the components do not have an immediate response that can be offered to the user, therefore, we must integrate this functionality through another Hook, in this case, it will be *useNotification* (figure 6.28).



Figure 6.28: *useNotification* Hook for React Components.

Depending on the operation, the function that implements the Hook will be called to display a message with different colors, identifications, and objectives. It is important to grant the user to have all the information necessary to continue making new actions inside the application, that is, if a medicine is added it is essential to add in the notification the identifier that is related with the medicine, as we can see in figure 6.29.

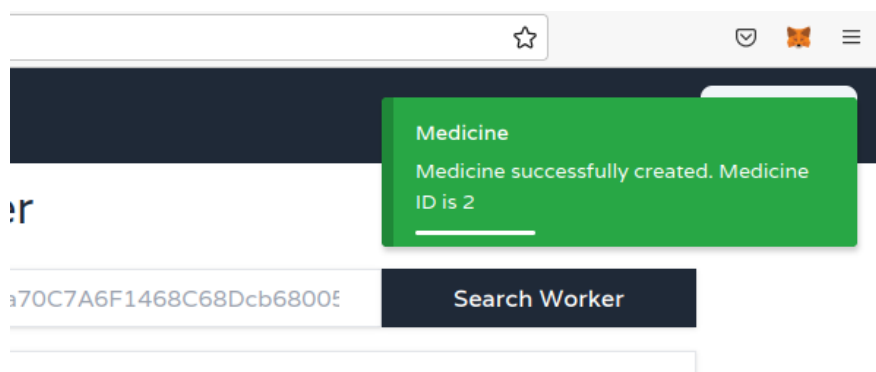


Figure 6.29: Successful notification for a new medicine added.

All in all, this section includes the main features of web interface development. The most important thing, and to highlight from the implementation, is how with the MVC pattern, the

code has been easily readable, as well as the importance of being reusable under the "Visible" component and not having to repeat it and complicate the application. The decisions made have been based on the usability and performance of the system to provide users with an environment that invites them to use it, and that does exactly what it has been designed for, not one more action, not less.

Chapter 7

Results

During the course of the entire project, different reasons have been exposed about why the application is based on a Blockchain environment and not in a centralized environment like any application present in the market network. Therefore, it is necessary to indicate the practical and theoretical results obtained with the tool in order to explain whether carrying out this application implies an improvement in some type of field. These results are based on a series of features that would be essential in any application that is 100% open to the public:

- **Security:** One of the objectives indicated in the introduction of the project [1](#), was the ability to be able to isolate each of the members from the actions of others or not be able to interfere in other actions outside their own account. This point is fulfilled satisfactorily since the application uses addresses to identify each user inside the system. That is, these addresses are a unique set of characters that cannot be replicated or spoofed by someone else, although if that were the case, as mentioned in the web development subsection [6.2.6](#), wallets themselves have mechanisms that substantially complicate the attempt to counterfeit an entire wallet. In addition, in terms of origin and identification, the addresses directly offer the possibility of fully knowing the first member of the chain, that is, if there were counterfeits whose drugs do not enter into the first steps of the supply chain, they would be easily controlled since they would not have the address of the manufacturer assigned. Moreover, it will be trivial to see in the NFT identifier how the information set does not correspond to any actual address or simply does not have data that can be related to any Blockchain address (if a real one was assigned), which would cause a complete rejection of the medicine and the person guilty of the facts could be quickly identified.

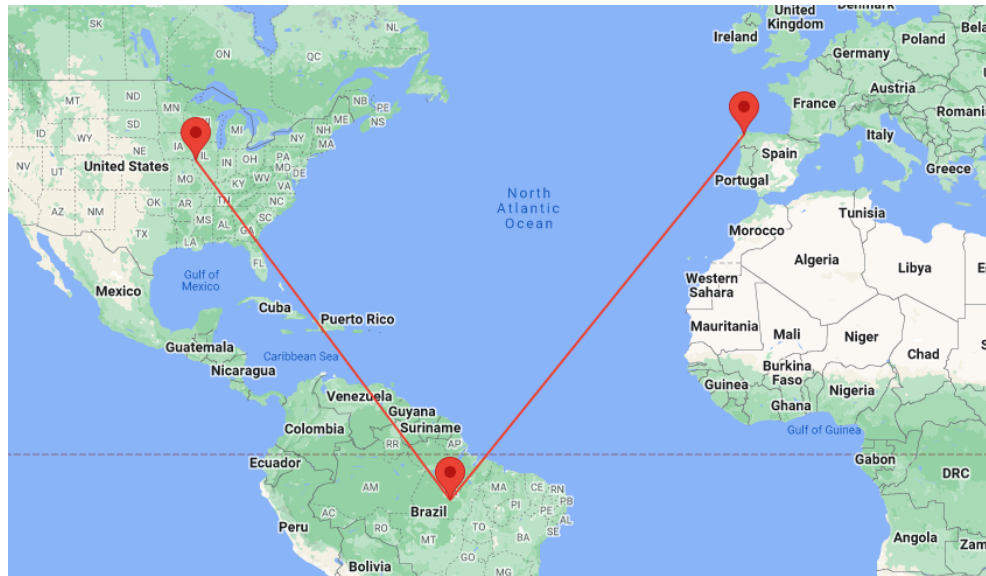


Figure 7.1: Medicine's path.

Name	Ibuprofeno
Description	Medicine that...
Manufacturing date	9/8/2022, 10:51:05 AM
NFT Data	ipfs://bafkreibcojz7acjgkhp07zflsrxr25zrakfck5n7p2q5t4kluviwqskhf5u
Number of States	6
States	Manufactured, Delivered by manufacturer, Received by distributor, Delivered by distributor, Received by pharmacy, Sold
Number of Dates	6
Dates	9/8/2022, 10:51:05 AM, 9/8/2022, 10:51:14 AM, 9/8/2022, 10:52:15 AM, 9/8/2022, 10:52:19 AM, 9/8/2022, 10:54:42 AM, 9/8/2022, 10:54:47 AM
Number of Coordinates	3
Coordinates	[40.42854255987023,-90.8382201790671], [-6.887100928925408,-52.56019249446572], [43.27416168395457,-8.245546931780334]

Figure 7.2: NFT and drug data.

On the other hand, as we saw in the Smart Contracts subsection 6.2.3, the use of the Blockchain implies an increase in the security of the information it stores. In a typical application, data is usually stored in databases, there may be several but, after all, they are centralized entities that have to carry out replication methods in order to keep the information safe against loss. However, within the Blockchain, each node that intervenes has all the information of each transaction carried out from the moment of its

deployment. In other words, not only can we know the information on medicines at any given time, but that information is immutable and cannot be changed at any time. For this reason, even if it is possible to enter the actions of the Blockchain externally, the information that already resides inside it cannot be changed (in addition, no function that implies the elimination of information is included in the actions of the application).

- **Integrity & Reliability:** This characteristic was essential when the project approach was carried out. During the State of the Art chapter 3, companies and governments discussed ways to maintain the integrity of information about products in a supply chain. However, they all have vulnerabilities that, while not trivial, substantially increase the chances of a product data spoofing attack. But, data integrity is something that this system can offer in each and every step of the process. This system substantially improves the centralized options mentioned because the information is shared entirely between each network node, maintaining a copy of the chain between them. In other words, while centralized options operate under a risk that, if it affects the central node, it will destroy the entire structure, with Blockchain, we can guarantee a 0% loss of information even with losses of 99% of the nodes that make up the network.

On the other hand, a significant improvement when it comes to achieving integrity in the information is to achieve reliability in terms of data offered. In alternatives (subsection 3.4.4), we saw how Blockchain provides high user reliability. However, this system offers a mechanism for associating NFTs with products that imply high reliability when assigning the identifier to medicines. In other words, these tokens are associated with the manufacturer, who would be responsible for the product; therefore, the end customer does not have any type of responsibility. Hence, since these tokens are not interchangeable (there is no function in the contracts that allows it), each identifier will be associated with a medicine and can never be part of another. In addition, since the manufacturer's address is associated with the product, it is easy to search against the system, and they cannot be falsified or exchanged. This simple search is something that must be done in a complex way in centralized environment systems that must use cryptographic or isolation methods to protect product identifiers, making the whole process more difficult and tedious for workers and customers.

- **Cost:** One of the most important improvements is the cost of maintaining a Blockchain-based system compared to other centralized systems (based on servers, databases, intermediate points, etc). Obviously, the cost was one of the most challenging objectives to achieve because a system of these characteristics at a business level requires a lot of maintenance and dedication. However, this application could be executed in

a Blockchain environment with thousands of nodes and would not reach the expense of centralized database anti-fraud systems. These environments require investment, transportation, quality, maintenance, advice, workers, servers, rental of host services, etc. However, this system does not need so much cost simply because most actions are not carried out manually but instead occur automatically. In addition, there should not be an excessive expenditure on computer machinery; only the nodes where the Blockchain is going to act are simply needed, we can talk about thousands of them, perhaps, but they do not require a high computing speed for the type of necessary transactions. That is, this system could offer all the functions that a standard centralized environment performs without the need for excessive expense in keeping the system active, renting space, or employees tied to preserving the environment functional.

- **Transparency:** As seen in chapter 3, all types of supply chains remain isolated from the end customer. In other words, they do not share any kind of information with users, and if they do share it, no one can be sure that the data is accurate, up-to-date, or even honest. However, this application makes the most substantial improvement of the entire project around transparency to its users. This system allows end customers to see all the information assigned to the medicine that workers and owners also see. In other words, users view the same information as the chain members. For instance, we are able to see in a medicine information related with all the workers involved in its processes (figure 7.4). We can assure those addresses are right if we see the figure 7.3 that shows us the first Ganache addresses. Those would represent public addresses that anybody can see or even search by it.

ADDRESS	BALANCE
0x7A67865Fa7cd29F34DdF042a749fCf91446ca3eE	100.00 ETH
ADDRESS	BALANCE
0x1fACD9E0cf18108212941E484CD48e7a6dCa1d06	100.00 ETH
ADDRESS	BALANCE
0x8d5639f5FD5e717aeDa5266662b8F13A516B79F8	100.00 ETH
ADDRESS	BALANCE
0x53C1145ddD1498f23214e5AD1bA28c5d70F55FdC	100.00 ETH

Figure 7.3: Ganache addresses.

0x7A67865Fa7cd29F34DdF042a749fCf91446ca3eE
manufacturer
Manufacturer
Description
[40.42854255987023, -90.8382201790671]
0x1fACD9E0cf18108212941E484CD48e7a6dCa1d06
distributor
Distributor
Description
[-6.887100928925408, -52.56019249446572]
0x53C1145ddD1498f23214e5AD1bA28c5d70F55FdC
pharmacist
Pharmacist
Description
[43.27416168395457, -8.245546931780334]

Figure 7.4: Workers medicine information.

In addition, since each drug has an assigned manufacturer address, they can also access that information. That is, they not only access product data but also the members themselves, so there is no type of information hidden from customers, and they are aware of what is happening in all processes at each moment. This is one of the most challenging improvements to incorporate and, thanks to the Blockchain, we can not only guarantee a secure environment that offers integrity to products but also the end customer can verify in each transaction what has happened, how it has happened, who has done it and what it has involved.

Worker Information	
Collaborator type	manufacturer
Address	0x7A67865Fa7cd29F34DdF042a749fCf91446ca3eE
Name	Manufacturer
Last Name	Description
Latitude	40.42854255987023
Longitud	-90.8382201790671

Figure 7.5: Worker data searched for its form.

These are some of the improvements achieved compared to traditional centralized supply chain systems. However, in order to clarify in a better way those features, a table would be build and it will sum up all features that this system improves compared to a traditional centralized system.

Features	Blockchain	Traditional
Contracts	Terms and clauses are written as computer codes (enforcing tariffs, trade policies, compliance agreements). Fully automated.	Terms and clauses are written in paper and signed by individuals.
Transaction and information data	Every data is encrypted with hash functions and stored permanently.	Modified, managed, and controlled by a single user called an administrator. Possibility of tampering or data loss.
Validation	Self-validation. Algorithms in Smart Contracts verify every piece of information stored. Outliners will be disputed.	Third-party validation. Manual entry must be validated to avoid discrepancies.
Node information	Multiple nodes. Decentralized as information is stored across different nodes which are independent of each other.	Single node. Centralized and the information is stored in a single database.
Transparency	All participants of the Blockchain system can access the information stored in the ledger.	Only privileged users can access and modify the information in the database.
Compliance Violations	Prevention. Any discrepancies will be immediately alerted to all users of the ledger.	Hard prevention. Only the several users has access to the database so violations are difficult to track.
Human error	Reduced. Any changes go through all the users of the system, and the outliers are identified by the inbuilt algorithms.	High. Data is manually entered by administrators and privileged users.
Failure risk	Low. Transparent, controlled, and tamper-proof, Delegates work and give credit to entities for contribution.	High. Single storage, data can get corrupted or erased. Entities' efforts go unnoticed with the failure of some other entity.

In this chapter, we have seen the results obtained and improvements compared to the currently proposed solutions. All these improvements are based on how the system performs under the same conditions as the other solutions. For this reason, we have verified how Blockchain-based environments offer a more outstanding guarantee of operation and

more transparency of information for the end user, an essential characteristic when it comes to keeping customers satisfied and being able to inform from a social and academic point of view. In addition, this system offers a great capacity to recover from errors, not only due to the decentralization of nodes that allows the majority of nodes to go down without apparent failure, but also due to the non-inclusion of the human factor within the system in terms of validation of transactions. The system feeds itself from the transactions, and the actions carried out by the nodes to recover from a fraudulent transaction or from fraud in the information chain stored in the blocks. For this reason, Blockchain-based environments offer great versatility and improvements to centralized environments that depend on a single central module.

Conclusion

ONCE the problem in question has been dealt with, we can see how the use of Blockchain is fascinating in this type of field. Even so, it is true that we are facing a relatively new world compared to traditional protocols for storage, communication, or data exchange. Due to this characteristic, it is expected that entering this technology causes rejection or even fear. However, as mentioned throughout this project, Blockchain-based environments are now booming for new banking, pharmaceutical, artistic, or any other kind of applications. Many fields today dominated by traditional protocols can be greatly benefited from the use of Blockchain as an addition to them, this concept is crucial to understand. Blockchain is not a protocol that implies a substitution of any previous technology. Yes, it is true that it significantly improves functionalities that, today, are controlled by obsolete actors, but it does not imply that Blockchain environments are the solution. When developing this type of project, the most sensible and interesting thing is to see how to provide users with a satisfactory and, above all, useful experience. For this reason, it is essential to connect these types of technologies and integrate them into environments where the functionalities help each other and can achieve new objectives by carrying out joint and hand-in-hand actions.

On the other hand, this specific project has been able to delve into Blockchain technology from an extreme depth, reaching bytecode levels in order to understand the flow and validation of transactions. That is to say, throughout the development of the tool, options have been investigated and studied to achieve the objectives in the best possible way, as well as trying to integrate the capabilities of the Blockchain environment with usability and help at the user level in the best possible way. In addition, as a result of the research carried out in pharmaceutical environments and because of the problems that exist at a global level in this field, it is vital to indicate that, although this system does not manage to fix all of them, it does try to achieve that, at least in the majority of cases, there is no fraud or scam at the supply chain level. All the steps carried out have offered a critical point of view to be able to

integrate this tool within a business context.

8.1 Future Work

All in all, from the first steps of the project, the design of the tool has been thought to be able to add new functionalities, for this reason, a modular-based architecture is followed throughout the implementation, so that new options could be added in a simple way. In other words, releasing this tool into the business world would not imply a headache for developers. However, various functionalities could be implemented in the future. Among them, we can highlight the integration of the system with other environments that are not compatible with EVM; that is, Smart Contracts or compatible code could be made and executed directly in other execution environments without the need to overwrite information or delete data. Within the business sector, it is a crucial factor since, in this way, the system's actions could be globalized and, for example, achieve a standard that can be executed anywhere in the world by any hardware.

On the other hand, new actions done by each worker could be integrated. For example, in the case of distributors or pharmacists, they could be indicated the possibility of carrying out actions to request the creation of medicines or even to edit them. However, this aspect should be consulted because a good theoretical base is needed to integrate this functionality without the entire system suffering consequences that imply a security flaw in the general structure.

A substantial improvement of the project would reside in the graphical interface. The one carried out right now has a fundamental but sufficient aspect for those users who are not familiar with the Blockchain environment and who, therefore, are visually indicated and notified of all their actions. Even so, in the case of user registration, more information could be added and would be helpful when it comes to searches, directly making a registration that implies a complete identification of the user. For example, identifiers such as DNI (verifying against a database) could help identify a person in case of fraud or malicious acts. Also, the system could cause bottlenecks in a business environment when entering data. Since the interface is based on forms, we would have to continuously write what we want to search for or add. For this reason, in industrial environments, technologies such as the QR could be added and store all the information on that specific medicine simply by scanning it.

In addition, at present, the business sector of information and communication technologies is immersed and interested in implementing traceability projects with the aim of being applied in various domain sectors. This tool offers direct and helpful functionalities for the

pharmaceutical industry that allow rapid deployment in any environment or field with minimal costs. Hence, its integration into the current pharmaceutical chain is an option subject to analysis that could offer numerous aid, reduction of expenses, and a guarantee to prevent the spread of medicines that do not fulfill their function or that imply a worsening in the state of patients.

Appendices

Glossary of terms

- **Blockchain:** it is a peer-to-peer, distributed ledger that is cryptographically secure, append-only, immutable (extremely hard to change), and updateable only via consensus or agreement among peers.
- **Smart Contract:** it is a secure and unstoppable computer program representing an agreement that is automatically executable and enforceable.
- **NFT:** it is a unique, non-interchangeable and indivisible token which represents an asset digitally on a Blockchain.
- **IPFS:** it is a protocol and network designed to create a content-addressable p2p method for storing and sharing hypermedia in a distributed file system.
- **DLT:** it is an electronic system or database to record information that is not executed by a single entity. This allows us to store and use data that can be decentralized and distributed both privately and publicly.
- **Supply Chain:** the planning and management activities that ensure the availability of needed products from the manufacturers to end users.
- **PBM:** worker in the pharmaceutical supply chain who negotiates with health plans and insurances to cover a list of drugs and display information on tier placement and coverage criteria that correspond with patient cost sharing for each drug.
- **IoT:** describes physical objects with sensors, processing power, software, and other technologies that connect and exchange data with other devices and systems via the Internet or other communication networks.
- **Decentralization:** is the process by which the activities of an organization, particularly those regarding planning and decision making, are distributed or delegated away from a central, authoritative location or group.

-
- **Solidity:** it is an object-oriented programming language for writing Smart Contracts. It is used to implement smart contracts on various Blockchain platforms, the most prominent being Ethereum.
 - **ROI:** it is a financial ratio that compares the profit or utility obtained in relation to the investment made, that is, it represents a tool to analyze the performance that the company has from the financial point of view.
 - **Ethereum:** it is an open source platform, which is used to execute Smart Contracts. It is programmable, which means that developers can use it in the creation of decentralized applications.
 - **EVM:** its function is to allow the execution of programs or smart contracts in order to deploy a series of added functionalities on said Blockchain so that users can enjoy them.
 - **Polkadot:** it is a relay mechanism that facilitates authenticated transactions from one Blockchain to another. Provides shared security that is the same for all members, regardless of the protocol they use.
 - **Solana:** it is an open source project implementing a new, high-performance, permissionless Blockchain.
 - **Hyperledger Fabric:** it is intended as a foundation for developing applications or solutions with a modular architecture. It allows components, such as consensus and membership services, to be plug-and-play. Its modular and versatile design satisfies a broad range of industry use cases.
 - **Tron:** it is a Blockchain platform that is dedicated to accelerating the decentralization of the Internet via blockchain technology and decentralized applications (DApps).
 - **Gas:** fee, or pricing value, required to successfully conduct a transaction or execute a contract on the Ethereum Blockchain platform.
 - **OpenZeppelin:** library for secure Smart Contract development. Build on a solid foundation of community-vetted code.
 - **CID:** it is a unique x bits identifier.
 - **DAG:** it is a directed graph that has no cycles; this means that for each vertex v , there is no direct path starting and ending at v .

- **DHT:** it is a type of hash table that stores key-value pairs and allows the value associated with a key to be queried. Data is stored in a distributed manner in a series of nodes (distributed systems), and provide an efficient search service which allows to find the value associated to a key.
- **Truffle:** it is a development environment, testing framework and asset pipeline for Blockchains using the Ethereum Virtual Machine (EVM).
- **Chai:** it is an assertion library, which can be paired with any Javascript testing framework. Chai has several interfaces: assert, expect and should, which allow the developer to choose the style that is most readable and comfortable when developing their tests.
- **Remix:** it is an Ethereum development environment that offers us the possibility of compiling and deploying contracts.
- **Token:** it is a reference (an identifier) that returns to sensitive data through a tokenization system. Mapping raw data to a token uses methods that make the tokens infeasible to reverse in the absence of the tokenization system.
- **Metamask:** it is a cryptocurrency software that is installed as a web browser extension. This is used to interact with the Ethereum Blockchain platform.
- **Web3:** it is an idea for a new iteration of the World Wide Web based on blockchain technology, incorporating concepts such as decentralization and token economy.
- **ABI:** it is the standard way to interact with contracts in the Ethereum ecosystem, both from outside the Blockchain and for contract-to-contract interaction.

Bibliography

- [1] S. Chopra and P. Meindl, *Supply Chain Management: Strategy, Planning and Operation*, 5th ed. Pearson, 2013.
- [2] Avalere. Follow the pill: Understanding the prescription drug supply chain. [Online]. Available: <https://avalere.com/insights/follow-the-pill-understanding-the-prescription-drug-supply-chainf>
- [3] D. F. W. Peter Behner, Dr. Marie-Lyn Hecht. Fighting counterfeit pharmaceuticals. [Online]. Available: <https://www.strategyand.pwc.com/gx/en/insights/2017/fighting-counterfeit-pharmaceuticals/fighting-counterfeit-pharmaceuticals.pdf>
- [4] J. P. B. Sci. Recent trends and future of pharmaceutical packaging technology. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3697200/>
- [5] D. Bansal. Anti-counterfeit technologies: A pharmaceutical industry perspective. [Online]. Available: <https://www.mdpi.com/2218-0532/81/1/1>
- [6] H. Schofield and L. Thasarathakumar. Blockchain, covid-19 and the pharmaceutical supply chain. [Online]. Available: <https://www.pharmexec.com/view/blockchain-covid-19-and-the-pharmaceutical-supply-chain>
- [7] C. Parizo. What are the 4 different types of blockchain technology? [Online]. Available: <https://www.techtarget.com/searchcio/feature/What-are-the-4-different-types-of-blockchain-technology>
- [8] Wikipedia. Model-view-controller. [Online]. Available: <https://en.wikipedia.org/wiki/Model-view-controller>
- [9] WHO. 1 in 10 medical products in developing countries is substandard or falsified. [Online]. Available: <https://www.who.int/news/item/28-11-2017-1-in-10-medical-products-in-developing-countries-is-substandard-or-falsified>

-
- [10] PharmTech. Why we need mass serialization — now! [Online]. Available: <https://www.pharmtech.com/view/why-we-need-mass-serialization-now>
- [11] I. Bashir, *Mastering Blockchain*, 3rd ed. Packt, 2020.
- [12] V. Buterin. Ethereum: A next-generation smart contract and decentralized application platform. [Online]. Available: https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf
- [13] C. Hill. Exploring structural and systemic improvements to promote effective and efficient pharmaceutical supply chain management for hiv/aids service delivery in nigeria. [Online]. Available: <https://www.proquest.com/docview/2239978684?pq-origsite=gscholar&fromopenview=true>
- [14] L. D. Fredendall and E. Hill, *Basics of Supply Chain Management*, 1st ed. The St. Lucie Press/APICS Series on Resource Management, 2001.
- [15] CSCMP. Cscmp certification collection. [Online]. Available: <https://books.google.es/books?hl=es&lr=&id=mCbnAwAAQBAJ&oi=fnd&pg=PT36&dq=cscmp+pharmaceutical+supply+chain&ots=sTZ1nIGuvL&sig=2jGuYS9TxkethLUZfdohdSVGKpg#v=onepage&q&f=false>
- [16] oodles. Improving authenticity verification in supply chain management with blockchain applications. [Online]. Available: <https://blockchain.oodles.io/blog/authenticity-verification-supply-blockchain-applications/>
- [17] xtelligen. Fundamentals of the pharmaceutical supply chain. [Online]. Available: <https://pharmanewsintel.com/news/fundamentals-of-the-pharmaceutical-supply-chain#:~:text=July%2001%2C%2020%20-%20The%20pharmaceutical,manufactured%20and%20delivered%20to%20patients.>
- [18] K. Karunamoorthi. The counterfeit anti-malarial is a crime against humanity: a systematic review of the scientific evidence. [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4064812/f>
- [19] Drugstore. Report: Majority of viagra bought online is fake (and possibly dangerous). [Online]. Available: <https://www.edrugstore.com/blog/erectile-dysfunction/viagra/report-majority-of-viagra-bought-online-is-fake-and-possibly-dangerous-m0917/>
- [20] G. L. Buckley GJ. Countering the problem of falsified and substandard drugs. [Online]. Available: <https://www.ncbi.nlm.nih.gov/books/NBK202526/>

- [21] J. Swiatek. Eli lilly intensifies efforts to stop fake pharmaceuticals. [Online]. Available: <https://eu.indystar.com/story/money/2014/04/06/eli-lilly-intensifies-efforts-stop-fake-pharmaceuticals/7327471/>
- [22] Anonymous. Tango attack: Cómo atacar (y proteger) dispositivos iot rfid. [Online]. Available: <https://www.hacking.land/2021/05/tango-attack-c-atacar-y-proteger.html>
- [23] S. P. McGrew. Hologram counterfeiting: Problems and solutions. [Online]. Available: http://www.nli-ltd.com/publications/hologram_counterfeiting.php
- [24] GS1, *GS1 healthcare reference*, 1st ed. GS1 Global Office, 2011.
- [25] C. Scardovi. Restructuring and innovation in banking. [Online]. Available: https://play.google.com/books/reader?id=uNM0DQAAQBAJ&pg=GBS.PA2.w.1.0.121_142&hl=es419
- [26] A. Alliance. Kanban. [Online]. Available: [https://www.agilealliance.org/glossary/kanban/#q=~\(infinite~false~filters~\(postType~\(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video\)~tags~\(~'kanban\)\)~searchTerm~'~sort~false~sortDirection~'asc~page~1\)](https://www.agilealliance.org/glossary/kanban/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'kanban))~searchTerm~'~sort~false~sortDirection~'asc~page~1))
- [27] H. J. Sun. About tron. [Online]. Available: <https://tron.network/about?lng=en>
- [28] H. Fabric. Hyperledger fabric docs. [Online]. Available: <https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html#smart-contracts>
- [29] OpenZeppelin. Openzeppelin docs. [Online]. Available: <https://docs.openzeppelin.com/contracts/4.x/>
- [30] Solidity. Solidity docs. [Online]. Available: <https://docs.soliditylang.org/en/v0.8.16/introduction-to-smart-contracts.html>
- [31] IPFS. Ipfs docs. [Online]. Available: <https://docs.ipfs.tech/concepts/what-is-ipfs/>
- [32] A. M. Antonopoulos, *Mastering Bitcoin*, 2nd ed. O'REILLY, 2017.

