GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Feasibility of private logistic micro-transactions in a distributed ledger

Author: Pablo Bonet Portugal

Director: György Dán

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título Feasibility of private logistic micro-transactions in a distributed ledger en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2019/20 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Pablo Bonet Portugal Fecha: 08/07/2020

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: György Dán Fecha: 08/07/2020

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Feasibility of private logistic micro-transactions in a distributed ledger

Author: Pablo Bonet Portugal

Director: György Dán

Madrid

FEASIBILITY OF PRIVATE LOGISTIC MICRO-TRANSACTIONS IN A DISTRIBUTED LEDGER

Author: Pablo Bonet Portugal

Supervisor: György Dán

Collaborating Entity: ICAI-Universidad Pontificia Comillas

ABSTRACT

Distributed ledgers are gaining acceptance in a variety of application areas, from energy to agriculture. In this project, the objective is to assess the feasibility of using a distributed ledger for storing and retrieving information about micro-transactions for low-quantity logistics applications. The project will consider the general use case of goods delivery and will design and evaluate a privacy-preserving data storage and retrieval model compatible with blockchain.

Keywords: Blockchain, Logistics, Privacy, Cybersecurity, QR Code

1. Introduction

The development of modern logistics has improved our lives as it has made it possible to build bridges between distant places. This growth has created a big challenge for everyone, as every time a transaction takes place, it must be constantly monitored in order to ensure it arrives at the correct destination, following the exact steps it should take.

In every logistic transaction, there are several entities involved which can be completely independent. This may create an issue in terms of trust as they may not rely on each other. For this reason, there is a need for a platform in the market of logistics that can manage and handle deliveries properly (continuously checking the stage of the delivery), using technologies that build an environment with a high degree of privacy and security. The project aims to assess the viability of storing information about logistic transactions in a distributed ledger or blockchain technology.

2. Project definition

The main purpose of this project is the creation of a platform for managing and tracking deliveries in an efficient way, making use of distributed ledger technology (DLT), or blockchain. In addition, some other technologies will be combined to ensure the highest degree of privacy and security. In this project, several different identities will participate in logistic transactions. A delivery plan includes all the steps between the producer and the client, whereas a handover is each of the intermediate steps between the initial and final entity. Fig. 1 shows the distinction between a delivery and a handover.

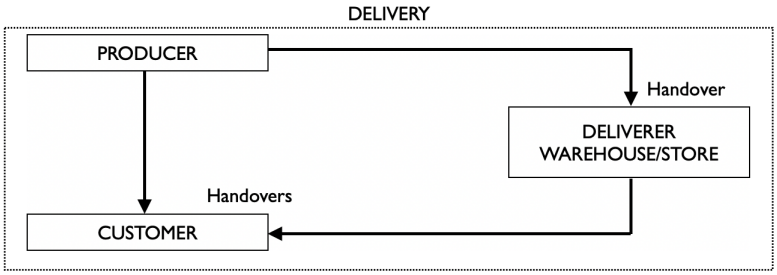


Figure 1: Logistic Chain

For using blockchain technology, HyperLedger Fabric is the platform selected, for many of its features [1], including the software development kit (SDK). The development of the project will use Java and SQL for database management, as well as Go for the chaincode needed in the blockchain network.

3. Description of the system

The functioning of the platform is based on the communication diagram presented in Fig. 2. In every handover, the first entity will generate a message m1 with information of the handover. Then, it will compute the hash H(m1)=SHA-256(m1), encrypting it with its private key: mO-R=(m1, [H(m1)]O), as seen in the diagram. This information will be encoded in a QR and then passed to the next entity in the delivery chain.

Then, the second step involves the following entity scanning the QR Code, verifying the signature and hash and sending a message m(R-DB) containing information regarding the whole handover to the blockchain network. Once this message is sent, the first entity will perform a query in the blockchain network to check if the parcel has correctly arrived at the

next stage of the delivery plan.

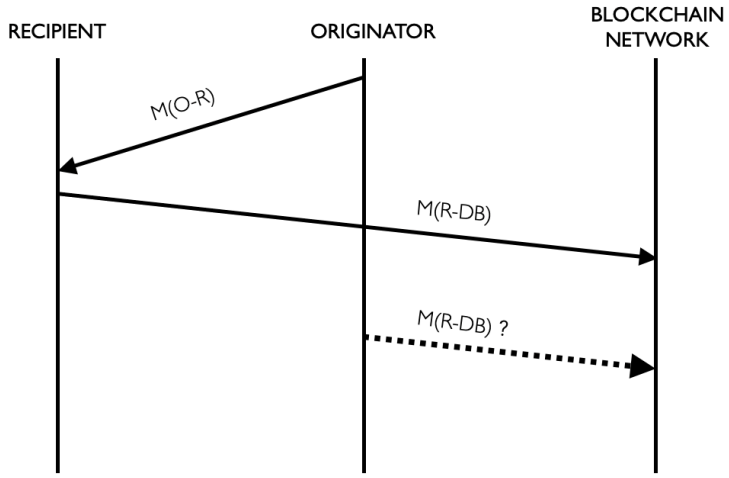


Figure 2: Communication Flow

When the process is completed successfully, the parcel can be sent to the next entity, passing through the delivery chain until the client receives it.

4. Results

As it is proved in this report, the platform is capable of handling deliveries and handovers, simulating real data for this purpose. Therefore, the main goal of the project is covered as it has been possible to create a software solution that is able to fit the needs in the logistics industry, especially the demand of an application that manages handovers enhancing privacy and security.

The use of blockchain has enormous benefits in many fields, including the market of logistic solutions [2]. However, the use of a distributed ledger to record transactions slightly increases the time to perform queries in this network, especially when the amount of transactions stored in the network is high. Fig 3 shows the time to query a transaction as a function of the number of transactions recorded in the network. As can be seen, there is an increase of 67% from 1 transaction to 30.000, which implies that if more transactions were recorded, time will continue to increase following a linear pattern.

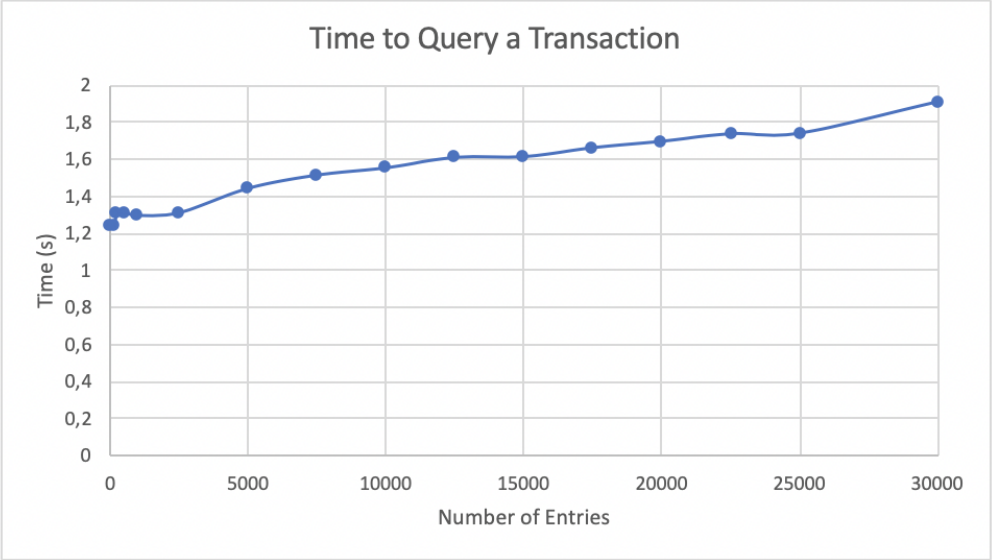


Figure 3: Network Performance

5. Conclusions

The use of blockchain technology is increasing in every sector and business. In the logistics industry, it represents an effective way to solve issues in a matter of trust between entities participating in a delivery chain [3]. For this reason, it is adequate to adapt current infrastructure to these advances.

With a platform like the one designed, the market of logistics has a solution for controlling transactions with privacy and security at the center, due to the fact that an optimal combination of technologies has been selected and integrated for its design and development.

6. References

[1] Cachin, C. (2016). Architecture of the Hyperledger Blockchain Fabric. Zurich: IBM Research.

[2] PwC. (2016). The future of the logistics industry . Shifting patterns, 5-9.

[3] Robinson, A. (n.d.). Blockchain Technology in Logistics: What Are the Implementation Challenges? Retrieved from Cerasis: https://cerasis.com/blockchain/

VIABILIDAD DE MICRO-TRANSACCIONES LOGÍSTICAS PRIVADAS EN UN REGISTRO DISTRIBUIDO

Autor: Pablo Bonet Portugal

Director: György Dán

Entidad Colaboradora: ICAI-Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Los registros distribuidos están ganando popularidad en multitud de áreas, como la energía o la agricultura. El objetivo de este proyecto es evaluar la viabilidad de utilizar un registro distribuido para almacenar y recuperar información sobre transacciones en una aplicación logística. El proyecto se desarrollará en torno a la entrega de bienes, diseñando y evaluando un modelo de almacenamiento y recuperación de datos que preserve la privacidad y sea compatible con blockchain.

Palabras Clave: Blockchain, Logística, Privacidad, Seguridad, Código QR

1. Introducción

El desarrollo de la industria logística ha mejorado nuestras vidas, conectando diferentes puntos del mundo alejados entre sí. Este crecimiento supone un gran desafío para todas las partes implicadas, ya que cada vez que realiza una transacción, ésta debe estar constantemente controlada para garantizar que llega a su destino, siguiendo la ruta adecuada.

En cada transacción logística hay varias entidades involucradas que pueden ser completamente independientes. Esto puede crear un problema en términos de confianza, ya que puede haber entidades que no confien plenamente en otras.

Por tanto, existe la necesidad de crear una plataforma en el mercado de la logística que pueda administrar y manejar las entregas adecuadamente (verificando continuamente el estado de la entrega), utilizando diversas tecnologías para crear un entorno con un alto grado de privacidad y seguridad. El objetivo del proyecto es evaluar la viabilidad de almacenar información sobre transacciones logísticas en registro distribuido o tecnología blockchain.

2. Definición del proyecto

El objetivo principal de este proyecto es la creación de una plataforma para administrar y rastrear entregas de manera eficiente, utilizando tecnologías de registro distribuído (DLT) o blockchain. Además, se combinarán otras tecnologías para garantizar un alto grado de privacidad y seguridad. En este proyecto, varias entidades diferentes participarán en transacciones logísticas. Un plan de entrega (delivery) incluye todos los pasos intermedios entre productor y cliente, mientras que una transacción (handover) es cada uno de los pasos intermedios entre la entidad inicial y la entidad final. La figura 1 muestra la distinción entre una plan entrega y un transacción.

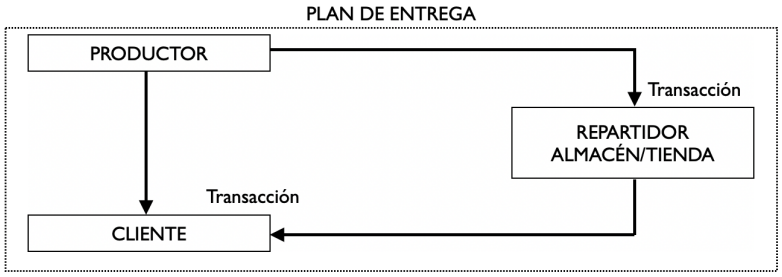


Figure 1: Cadena Logística

Para el uso de blockchain, se ha seleccionado la plataforma HyperLedger Fabric, por muchas de sus características [1], incluido el kit de desarrollo de software (SDK). El desarrollo del proyecto será mediante Java y SQL para la gestión de la base de datos, así como Go para el chaincode necesario en la red blockchain.

3. Description del sistema

El funcionamiento de la plataforma se basa en el diagrama de comunicación presentado en la figura 2. En cada transacción, la primera entidad generará un mensaje m1 con información de la misma. Luego, computará el hash H(m1)=SHA-256 (m1), encriptándolo con su clave privada: mO-R = (m1, [H(m1)]O), como se puede observar en el diagrama. Esta información se codificará en un QR y luego se pasará a la siguiente entidad en la cadena de entrega.

A continuación, el segundo paso involucra a la siguiente entidad, que debe escanear el Código QR y verificar la firma y el hash y enviando un mensaje m (R-DB) que contiene información sobre la transacción completa a la red blockchain. Una vez que se envía este mensaje, la

primera entidad realizará una consulta en la red blockchain para verificar si el paquete ha llegado correctamente a la siguiente etapa del plan de entrega.

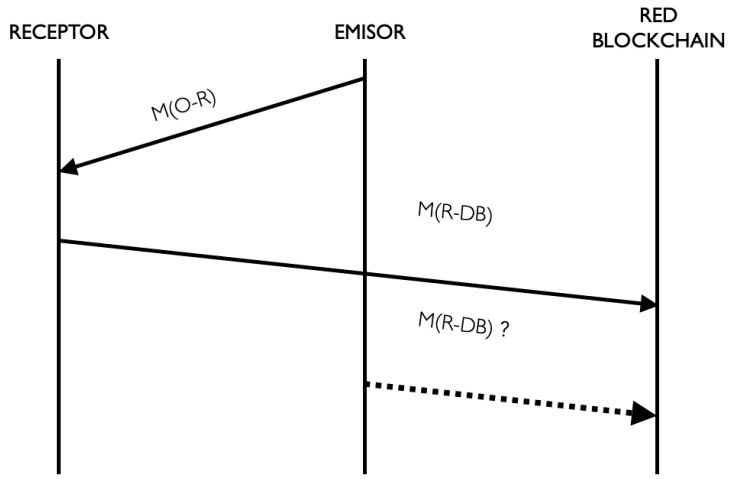


Figure 2: Ciclo de Comunicación

Cuando el proceso se ha completado con éxito, el paquete se puede enviar a la siguiente entidad, pasando por la cadena de entrega hasta que el cliente lo reciba.

4. Resultados

Tal y como demuestra esta memoria, la plataforma es capaz de manejar transacciones logísticas, simulando datos reales. Por tanto, el objetivo principal del proyecto se ha completado con éxito, ya que ha sido posible crear un software capaz de satisfacer las necesidades de la industria logística, especialmente la precisión de una aplicación que gestione transacciones de manera eficiente, teniendo en cuenta aspectos como la privacidad y la seguridad.

El uso de blockchain presenta claros beneficios en muchos campos, incluido el de las soluciones para la industria logística [2]. Sin embargo, el uso de un registro distribuido para registrar transacciones aumenta ligeramente el tiempo en realizar una consulta en esta red, especialmente cuando la cantidad de transacciones almacenadas es elevada. La figura 3 muestra el tiempo necesario para consultar una transacción en función del número de transacciones registradas en la red. Como se puede ver, hay un aumento del 67% de 1 transacción a 30.000, lo que implica que si se registran más transacciones, el tiempo continuar ía au-

mentando de manera lineal.

Figure 3: Rendimiento de la Red

5. Conclusiones

El uso de la tecnología blockchain está aumentando en todos los sectores y negocios. En la industria logística, supone una forma efectiva de resolver problemas en materia de confianza entre las entidades que participan en una entrega [3]. Por esta razón, es recomendable adaptar la infraestructura actual a estos avances.

Con una plataforma como la diseñada, el mercado de la industria logística tiene una solución para controlar las transacciones desde una perspectiva que considere la privacidad y seguridad como elementos determinantes, debido al hecho de que se han seleccionado e integrado diferentes tecnologías para el diseño y desarrollo de la plataforma.

6. Referencias

[1] Cachin, C. (2016). Architecture of the Hyperledger Blockchain Fabric. Zurich: IBM Research.

[2] PwC. (2016). The future of the logistics industry . Shifting patterns, 5-9.

[3] Robinson, A. (n.d.). Blockchain Technology in Logistics: What Are the Implementation Challenges? Retrieved from Cerasis: https://cerasis.com/blockchain/

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Globalization has brought an enormous amount of advantages in almost every sector of our life, from education to business through healthcare or energy. In the field of logistics, it has made it possible to connect almost every part of the world, with new e-commerce platforms such as Amazon, AliExpress, or eBay gaining market share over traditional retailing. These companies have transformed the way consumers behave as conventional distribution models are progressively deflating in favor of new ones, which constitute the main element of e-commerce [1]. E-commerce stands for electronic commerce and involves an exchange of products (goods and services) by electronic means, especially the internet. This type of commerce started functioning in the 70s, and it has continued to grow over the years. As shown in Fig.1.1, in the last decade, online sales have experimented a considerable growth compared to traditional or in-store sales, and everything points to this tendency maintaining for the future. In fact, due to the current Covid-19 pandemic, consumers have massively started to buy online as the restrictions and fear of the virus make this alternative desirable for everyone. Therefore, in the last months, the amount of deliveries has increased achieving records unseen before. Many experts believe that even when the pandemic is over, the propensity to buy online will remain [2].

The increase in this sort of purchase represents both an issue and a great opportunity in the area of logistic transactions. When a customer buys a product online, or when there is an exchange of goods between different companies, the transaction must be constantly monitored to make sure the item follows the correct path and arrives at the intended desti-

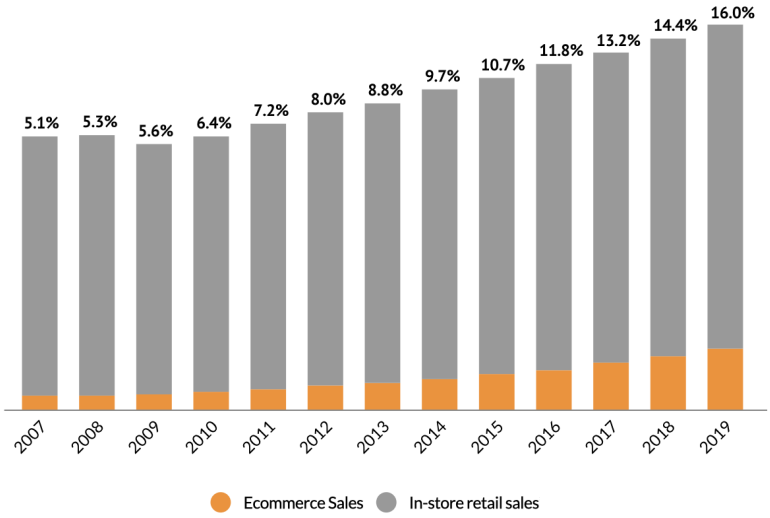


Figure 1.1: Growth of e-commerce (2007-2019) [3]

nation.

This may seem obvious, but the fact that both consumers and suppliers operate in a common environment is of extreme relevance for the market as it can prevent many problems from occurring or solve them in case they happen.

As a result of this, there is a need for effective management of deliveries, especially when the same client can purchase various products from different companies. These enterprises should concentrate all their efforts in making the customer experience as positive as possible since according to the last report of software and cloud solutions company Salesforce, 84% of clients consider experiences can have the same degree of importance as the product or service they are purchasing [4].

Furthermore, there can be several different entities involved in this process such as companies, deliverers, warehouses, stores, or recipients. These entities might not know each other beforehand and so they may have no trust in one another, and this can be a reason for conflict between them. In particular, from the perspective of consumers, since there can be various options for buying the same product, the decision will always depend on factors such as price, quality, or trust. In the case of similar conditions for a good or service, a customer will tend to buy from companies that are known to be trustworthy [5].

These are the main reasons why the market of logistic transactions needs solutions that are efficient, effective, and safe for governing these arrangements and tracking the paths products take. These solutions should be built and designed considering the principles of privacy and scalability for the points detailed in this section.

This project aims to create a solution for this issue, complying with international standards in the logistics industry and providing an answer to the matters of privacy, trust, and efficiency through a technological perspective. The main motivation for this development is the production of a platform that can be used for managing and tracking purposes of different deliveries with certain design principles at the core. Some of the most relevant ones are, as stated before, security, privacy, and efficiency.
Combining different technologies with this design postulates, a platform for managing deliveries will be built and used to simulate a real environment in which logistic transactions occur, evaluating the viability of storing information regarding these transactions.

Chapter 2

Description of Technologies

This project aims to create a solution for the logistics industry able to handle deliveries securely and systematically, enhancing privacy between the parties involved. In order to do so, an optimal combination of different technologies must be used. In this section, these technologies implemented will be explained in detail.

2.1 Cryptography

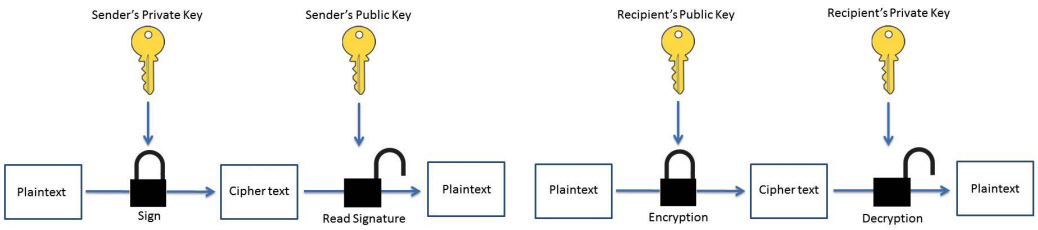
In the last decade, the amount of attacks has substantially increased, especially in terms of data breaches [6], which has made security crucial when designing software solutions. In fact, according to the World Economic Forum, cyberattacks can be considered as the fifth biggest threat to the world's stability [7].

Considering this, data security is vital for the development of this project, which is why it must comply with the C.I.A principles [8], which are:

- Confidentiality. Data can just be accessed by entities that are authorized to do it, but no unwanted access should be permitted. This would allow units involved in transactions to be able to read the information regarding their deliveries with the guarantee that no other party can do so.
- Integrity. Information regarding transactions must not be modified in transit, which represents that if no unit has to add any details, the recipient should see exactly what the originator sent.

- Availability. The resource, in this case, data, has to be accessible for the elements with access to it in an immutable way.

For this to be achieved, several different algorithms for various purposes will be used. Data containing information regarding transactions has to be hashed [9]. Hash functions are used to map data of arbitrary size with a fixed-size output so the input can't be retrieved from what the function returns. Then, digital signatures are implemented to verify authenticity. In each transaction, the sender can sign information with a private key so the recipient is able to verify the integrity of the message with the corresponding public key. Lastly, Public-key cryptography allows encryption of messages with a public key. The recipient can decrypt it using the private key, assuring confidentiality, authenticity, and non-repudiability of data. In Fig.2.1 digital signatures and asymmetric encryption are compared.



(a) Digitally Signing a Message

(b) Encrypting a Message

Figure 2.1: Digital Signature and Public Key Encyrption

For the purpose described above, some algorithms can be considered optimal, each of them with some features that make them unique and suitable. SHA-256 stands for 'Secure Hash Algorithm', developed by the National Security Agency (NSA) [10]. It is one of the most popular hash functions as it balances security with computational cost and it's efficient and collision-resistant. It is also used in several applications such as Bitcoin [11]. RSA is an asymmetric algorithm based on mathematical functions, in particular, the "factorization problem", which consists of factorizing the product of two large prime numbers. This algorithm is commonly used for encryption and decryption of information [12]. Table 2.1 summarizes the different purposes and algorithms.

Field	Algorithm Used
Hash	SHA-256
Digital Signatures	SHA-256 with RSA
Encryption	RSA

Table 2.1: Summary of algorithms.

2.2 Blockchain & Smart Contracts

2.2.1 Blockchain

In the last years, the popularity and use of blockchain technology has increased in many applications. Despite the fact that it would not be rare to know what blockchain is and how it works, this section will include a brief explanation to clarify some concepts.

The simplest approach to blockchain technology is understanding it as a database distributed through various computers where information can be recorded. This information is stored in blocks and the set of blocks form a chain, which is why this form of technology receives the name blockchain [13]. Nevertheless, the perspective that best helps understand the use of this technology in the project is blockchain as a Distributed Ledger Technology (DLT) where transactions are registered. Each of the blocks mentioned contains a certain amount of transactions so when a new transaction takes place, it is recorded and every participant in the network can see it.

Blockchain has gained notoriety in the field of cryptocurrencies, especially because of bitcoin [14], the most popular among them, but it used in several other spaces due to some particular aspects that make this technology extremely useful [15]:

- Security. Blockchain technology is secure as all the transactions recorded are encrypted individually.
- Anonymity. The identities of all the participants remain anonymous (it is not possible to relate any information to who recorded it) or at least pseudonymous (entities use

pseudonyms to hide their real identity).

- Unanimity. All the participants have to agree to the plausibility of all records.
- Immutability. Once the transactions have been recorded, they remain unaltered, and is not possible to modify them.
- Consensus. It constitutes a key aspect of blockchain technology. Since blockchain database is decentralized, all participants must work verifying in authentication transactions taking place in the network.

Blockchain networks can be classified attending to different criteria [16], but for this project, the most relevant distinction is between permissioned and permissionless blockchain networks [17]. In a permissioned blockchain, a previous agreement is required before the network can start operating and transactions can be recorded. On the other hand, if the blockchain network is permissionless, anyone is free to participate in the process. Both permissioned and permissionless networks share the basic characteristics of blockchain such as being distributed ledgers, based on consensus or immutable. Permissioned blockchains are better in terms of efficiency, and scalability and present a higher degree of security, transparency, and anonymity, which is why they constitute a robust and resilient solution for blockchain applications.

2.2.2 Smart Contracts & Chaincode

A blockchain network is formed by several different peer nodes, where the ledger as well as smart contracts are hosted. Smart contracts define the terms of the agreement between the parties involved in a transaction. In the frame of blockchain, these contracts are developed through chaincode, which is code written to handle the business logic defined and agreed on [18].

As all these concepts may seem confusing, the diagram in Fig.2.2 contains a schematic representation of them. In it, a blockchain network is represented. There are three different peer nodes (P1, P2 & P3), and the ledger (L) as well as the smart contract (S) is instantiated and deployed in the three of them.

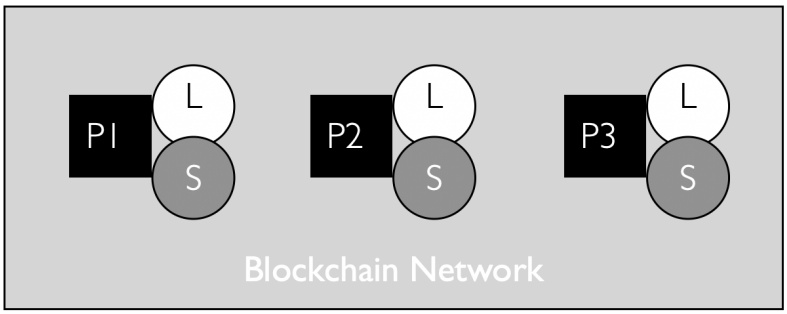


Figure 2.2: Example of blockchain network

2.3 HyperLedger Fabric

Once the theoretical frame for blockchain has been set, the practical approach to it in the project must be explained. In order to implement the blockchain network, Hyperledger Fabric has been the platform selected. This decision is based on some of the features the Hyperledger Fabric offers, that will be detailed in the following lines[19].

Hyperledger Fabric is a permissioned distributed ledger technology platform developed by the Linux Foundation with the collaboration of IBM and Digital Asset. It is an infrastructure for permissioned blockchain network distinguished from other similar platforms for its configurable modular architecture and the possibility to use several different programming languages for development and chaincode. As presented in the previous section, Hyperledger Fabric operates as a permissioned network. In this network, there are several nodes in which the chaincode is executed to perform actions such as recording transactions in the ledger or querying information relative to the transactions.

One of the reasons why this platform was chosen is the development kit available. Hyperledger Fabric SDK is in charge of managing the interaction between the user and the network, as it governs the channel's and chaincode's lifecycle as well as executing the last one. The diagram presented in Fig.2.3 is a visual representation of the flow of execution in Hyperledger fabric. This sequence can be summarized in the following steps:

1. Generation of artifacts. Artifacts are the files needed to configure the channel for the blockchain network. These include both the file with the details of the channel trans-

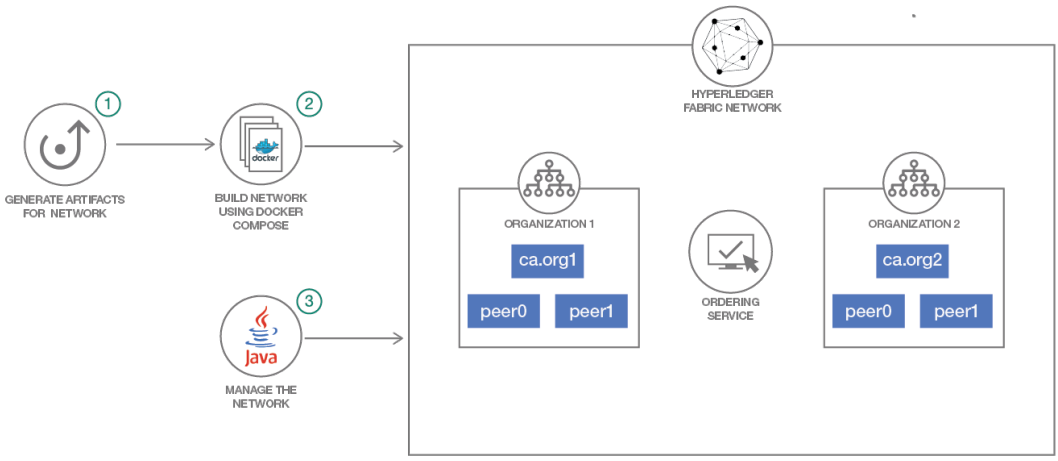


Figure 2.3: Flow in Hyperledger Fabric execution

actions as well as the genesis block (the one corresponding to the first block in the chain, which is in charge of initializing the whole blockchain).

2. Building the blockchain network. In order to use this network, it has to be built in advance. For this purpose, as well as the artifacts already explained, docker-compose will be used.

3. Managing the blockchain network. Once the network is built and functioning, it can be used for the purpose of the project itself. This includes deploying and instantiating the chaincode, registering users, recording transactions, and performing queries.

2.4 Docker Compose

Docker is a platform as a service (PaaS) designed to run applications packaged into containers, which run isolated from one another [20]. Docker standardizes the way code is run in a fast and efficient way, which is why it's frequently used when designing solutions based on Hyperledger Fabric. One of the most useful tools Docker offers is Docker Compose, used for defining applications in a Docker environment comprising several different packages. Despite running in isolation, Docker Compose is in charge of making all these containers work together.

Docker Compose allows applications to run in various environments. This makes Docker

Compose especially useful for scalability purposes, as it makes it possible to run applications from a single server to a Swarm cluster (a group of computers running Docker simultaneously) [21].

The way this tool operates in the frame of Hyperledger Fabric is as follows [22]. Firstly chaincode is distributed through the blockchain network. Peers then create an image that can be executed from the chaincode, and Docker creates a container through that image to send the chaincode. Then this container will create an image for the built chaincode. Now the peer is ready to use that image for executing the chaincode, starting a new container from the image of the chaincode, using a provided IP address, or a DNS for successfully connecting to the peer.

Chapter 3

State of the Art

3.1 Logistic Management

The growth in deliveries represents a challenge for companies, as their control is crucial. Consequently, there has been an increase in solutions for these firms to manage logistic transactions. Some companies use their tracking systems since they work exclusively with their own products or they control all the elements and participants in the delivery chain. The scope of this project is more ambitious and plans to go one step further. The idea is to create a platform for managing, controlling, and tracking these deliveries, where different entities completely unknown to each other can participate without having concerns about privacy or trust matters. For this reason, the main feature of this solution is the use of blockchain technology and in particular of the platform Hyperledger Fabric.

As can be imagined, there are thousands of platforms that satisfy the basic objective of the project, managing and tracking logistic transactions [23], and each year, the number of solutions for this purpose continues to grow. Some of them, along with the greatest benefits they offer are detailed in this section:

- Deliforce. Probably the most popular software solution for this matter worldwide. It stands out for the optimization of routes in the delivery process, constantly informing the customer and efficient duty allocation for deliverers. The analytics obtained from previous transactions ensure the platform constantly improves the service provided.

- Onfleet. The main characteristic of Onfleet is how automatized the process is. Notifications via SMS and tracking links are created by the system automatically and sent to the parts involved immediately, which makes the process faster and easier.

- Bringg. There are some distinctive features that make Onfleet an interesting solution, such as Crowdsourcing or Multiple Fleets. In fact, it has one the highest satisfaction rates for platforms of this kind, and top companies (Coca-Cola, Leroy Merlin, Walmart ...) choose it for their deliveries and pick-ups.

- Tookan. Last-mile delivery tracking as well as tracing deliverers in real-time, make Tookan a powerful alternative too. It also has powerful analytics to improve routes and accelerate deliveries.

It is important to mention the fact that these are real platforms already prepared to function for actual deliveries, whereas the purpose of this project is to create a platform that serves the tracking goal, with data that simulates real deliveries. They should not be seen as alternatives but as the current state of platforms in this matter.

3.2 TradeLens

After describing the leading existing platforms for logistic transactions, there is a project that must be mentioned due to the fact that it shares some important characteristics with the one developed and described here. TradeLens is a platform launched jointly by multinational technological company IBM and Maersk, the world's leading container ship operator. TradeLens is based on IBM Blockchain Platform, which is built on HyperLedger Fabric, the same blockchain platform as the project developed. Fig.3.1 shows the architecture of TradeLens including all the elements (Blockchain Netowrk, APIs ...).

TradeLens is a digital shipment platform that can be used to save time and money for companies and individuals in the supply chain, as well as reducing friction between these entities. This platform is based on the principles of collaboration, transparency, and efficiency. It provides control and management of logistic information for the different elements involved (carriers, ports, authorities, financial service providers ...).

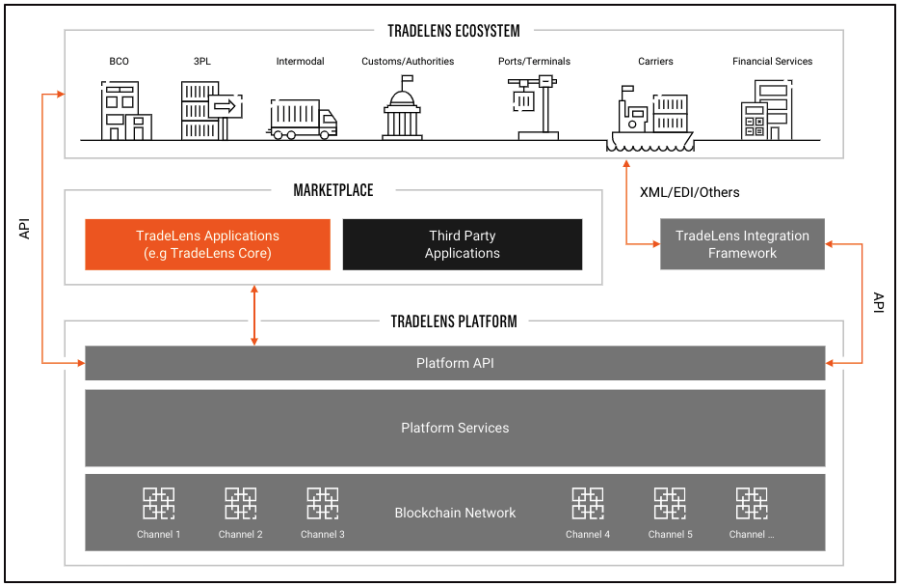


Figure 3.1: Platform Architecture

It operates sharing documents with data regarding transactions between the different entities involved. Its sophisticated permissioning model makes it easy to control views and updates information at each stage of the supply chain. This leads to one of the biggest benefits of TradeLens. In the event of an unexpected change in the destination of a delivery, parties are allowed to see immediately the new information so that everything is successful [24].

As well as with the platforms described earlier, TradeLens is a solution provided by international top companies (IBM and Maersk), which makes it completely legit and respected by everyone in the logistics industry. It enhances privacy between the participants of a transaction by using blockchain technology to store information regarding deliveries, which is the reason why it deserves to be mentioned for the similarities with the project developed. However, compared to the project at issue, TradeLens is more oriented to international shipment (partly because of Maersk), involving authorities, customs, and other entities that in the framework of the solution developed are of less importance.

Chapter 4

Project Definition

4.1 Justification

Section State of the Art explores several different platforms available in the market, from conventional logistic solutions to a more developed approach using blockchain technology, TradeLens. Despite the similarities TradeLens may have with this project, there are clear differences. TradeLens is a real platform created by IBM and Maersk, oriented to international commerce and shipments, whereas this project aims to create a simulation of a platform for deliveries on a lower scale. However, this doesn't mean the platform designed doesn't intend to be real as it could perfectly be extrapolated to the real market. After analyzing the previous alternatives, none of them has the right combination of technologies to generate the desired output. As privacy and security are at the center of the design process, the proposed solution integrates the technologies already mentioned in an attempt to make the optimal platform for logistic purposes as well as a safe and private environment for all entities involved so their activities can be developed without having any concern. The cryptographic tools ensure data is protected in the transition from one entity to another, whereas the use of a distributed ledger provides an answer to the trust issue between different entities [25].

4.2 Objectives

The main purpose of the project is the creation of the platform in the terms described. However, there are several objectives that are pursued with this project. For greater clarity, the main goals will be covered and classified into two different groups:

4.2.1 General Objectives

In the first category, the purposes that are of greater scope will be detailed.

- Solving the Problem. As it was presented in previous sections, despite the fact that there are several platforms that serve the purpose of managing, tracking or monitoring parcels in a delivery, there is no solution in the market that combines the technologies used in this one.

- Distributed Ledger. One of the technologies mentioned in the previous objective, and perhaps the one that makes this solution so unique, is the use of a distributed ledger. As stated in the title, this project wills to assess the feasibility of using this distributed ledger for recording information on transactions and then retrieving it. This is mainly why blockchain technology is a major part of the project.

- Security & Privacy. Section Cryptography emphasizes on the different techniques that have been implemented to assure data is safe. Furthermore, the use of blockchain technology and especially of the platform Hyperledger Fabric guarantees a high degree of privacy which is enormously beneficial for the users and companies using this platform.

4.2.2 Technical Objectives

Apart from the goals that have been described above, there are some others that have to do with the learning outcome of this project. They belong to more academic fields as they represent some acquirement of knowledge.

- Programming. The first technical objective, which is the main one, is related to programming skills. As it was explained in previous sections, one of the main reasons

to select Hyperledger Fabric as the platform to use for blockchain technology was the software development kit (SDK) available, and the possibility to use several different programming languages for various purposes. Table 4.1 summarizes the different languages used in the project. In this context, General Development includes the SDK mentioned as well as security functions, QR creation and reading, data generation ...

Field	**Language**
General Development	Java
Database Management	SQL
Chaincode	Go

Table 4.1: Programming Languages.

- Technologies. As it has been mentioned several times, many different technologies must be combined in order to achieve the solution desired. This fusion is done to create some sort of synergy so that these technologies together will result in the most advantageous output possible.

4.3 Methodology

Since the project presents a solution framed in the logistics field, it has to comply with certain standards of this industry. For this purpose, the standards developed and maintained by GS1[1] will be used [26]. In order to understand the sequence and functioning of this project, a distinction in terms of notation must be made:

- *Delivery.* A delivery is the general logistic process, which includes from the moment the parcel leaves the initial destination (producer) until it reaches its final destination (customer), including all the stages in-between (warehouses, stores, deliverers...).

[1]GS1 is a non-profit organization in charge of the development and support of global standards of industrial and logistic communication worldwide. It is formed by 114 local members and more than 1.5 million companies. One of its most popular development is the barcode, which along with the rest of GS1 contributions are known for their efficiency and safety

- *Handover*. A handover is each one of those halfway transactions performed within the delivery chain. A set of handovers constitute a delivery.

It is important to identify every element of the process, from entities to handovers and delivery plans. Each one of them has a unique identifier with a fixed structure. Table 4.2 contains the name of each identifier, which will later be explained.

Element	**Identifier**
Entity	GCP
Delivery	SSCC
Handover	PHO

Table 4.2: Logistic Identifiers.

Each entity in the logistic chain (producers, deliverers, stores, consumers ...) has a unique identifier, GS1 company prefix (GCP). A GCP consists of a country code followed by some digits, in total up to 9 digits. A delivery, and therefore the parcel that is being transported, is identified by a parcel identifier or GS1 Serial Shipping Container Code (SSCC). This identifier consists of 18 digits, with the following structure: N FFFFFFFFF AAAAAAA C. N is the extension digit, with a random value (0-9). It is followed by the GCP of the producer of the parcel, and then 7 digits that identify the parcel itself. The last digit (C), is the check digit, for verification purposes. Each handover within this delivery plan will have another unique identifier, a Planned Handover Identifier (PHO), comprised of 9 digits. Therefore, a handover or transaction contains the following information:

- Planned Handover Identifier (PHO).
- Parcel Identifier (SSCC).
- Originator (GCP).
- Recipient (GCP).
- Date (From, To).

- Location (Latitude, Longitude).

Whereas a delivery plan contains:

- Parcel Identifier (SSCC).
- Producer (GCP).
- Customer (GCP).
- Date (From, To).
- Location (Latitude, Longitude).

For communication purposes, information on deliveries and handovers will be encoded in QR codes[2], which in a real environment would be printed and attached to the parcel at issue. An actual handover takes place between an originator (O) and a recipient (R) in the delivery chain (going through several different intermediaries) and is based on the information detailed above of a planned handover. The steps of a handover are as follows.

1. **Originator (GCPO)**: The originator scans the QR on the parcel and checks that the SSCC contained matches the one in the handover. Then creates message m1=(PHO, SSCC, GCPO, DateTime, Location, RO), where R0 is a random number (nonce) generated by the originator. Originator creates the hash of this message, H(m1)=SHA-256(m1), and digitally signs it using its private key.

2. **Recipient (GCPR)**: The recipient will receive the QR where (m1, [H(m1)]) is encoded and digitally signed by the originator. This entity has to validate the digital signature using the public key of Originator. Again it will scan QR and check that both the SSCC and the PHO are the ones expected. The recipient then creates message m2=(PHO, SSCC, GCPO,GCPR, DateTime, Location) as well as H(m2)=SHA-256(m2). Everything is encrypted with the public key and sent to the blockchain

[2]QR Codes or "Quick Response" Codes were created in 1994 to develop a new generation of standards able to improve barcodes. In comparison with them, QR Codes are faster to read and have greater storage capacity

network.

3. **Originator (GCPO)**: The originator must perform a query in the blockchain network with the PHO of the handover and retrieve from it the whole message, which contains the information of the whole handover encrypted. Then it decrypts this message using the private key of the Recipient to check if this second entity received the message.

If the steps above are successfully completed, then the transaction has been performed without drawbacks. Therefore, the parcel can be sent to the next entity so it can continue along the delivery chain until it reaches the intended destination and the delivery plan is accomplished.

4.4 Planning & Economic Estimation

4.4.1 Planning

This project has been developed in several different stages. Table 4.3 contains a summary of the weeks and blocks developed. During the first week, the project was described, understanding the concepts of the logistic framework and the technological approach (combining blockchain, QR Codes, cryptography ...).
The following weeks were used for the development of the main parts of the project, which are the ones related to data and logistics. As data simulates real deliveries and handovers, it has to be built in a consistent and robust way despite being randomly created. All the identifiers have to be generated with the logical sequence of the delivery chain. The delivery plan as well as the handovers that are part of it have to be resilient and congruous. As stated in Methodology, information related to this messages is encoded in QR codes, therefore the development of code to generate and read this type of code, including the use of libraries was also done during these weeks.

The subsequent weeks were used for the development and integration of technologies, in particular the cryptographic matters and the blockchain network: designing functions for hashing, digitally signing or encrypting data as well as learning how the Hyperledger Fabric

Week	Objective
Week 1	General Description of the Project
Week 2 - 3	QR Development
Week 3 - 4	Data Generation
Weeks 5 - 7	Logistics Development
Weeks 10 - 11	Cryptographic Development
Weeks 12 - 14	Blockchain Network
Weeks 15 - 16	Statistics Obtention
Weeks > 17	Report

Table 4.3: Project Scheduling.

SDK works and how to integrate it with the rest of the project so that the blockchain network could be used.

The last weeks of the project were destined to obtain certain analytics related to the performance of the network as well as writing this report.

4.4.2 Economic Estimation

The reasons why a distributed ledger technology implementation has been selected for this project have already been discussed. Blockchain is, in general terms, secure and easy to use, but it also has a great benefit: the economic aspect. HyperLedger Fabric, the platform selected for the development of the blockchain network, is part of the HyperLedger project, a set of open-source tools for the development and maintenance of blockchain networks, especially in business environments. The fact that is open-source software means it is ready to be used and modified for the intended purpose without any associated monetary cost [27]. However, there are some costs that must be considered if a real implementation of this platform is carried out.

Firstly, since HyperLedger Fabric is an open-source software platform, there is no technical support or warranty. In the case of proprietary software this would not be a problem, but for this matter, there should be some external support considered for the blockchain

network in case it fails or there is any setback. There must also be technical support for general IT purposes as well as maintenance of the infrastructure involved. This project presents the platform created and a simulation of its use, but the reality is far more complex than it. For a real implementation in which actual deliveries are managed by means of this software solution, there must be an investment in infrastructure that is not needed for the project right now.

Chapter 5

Development

5.1 General Aspects

The purpose of this project is the creation of a platform to manage and track parcels in a delivery chain in an efficient considering privacy and security. In this context, producers will send the parcel to consumers, going through intermediate entities, which can be deliverers, warehouse stores, distribution centers...

The diagram in Fig.5.1 details a logistic chain with all the different entities that may be involved. It also clarifies the distinction made before between a delivery and a handover. The general process will be now explained in detail:

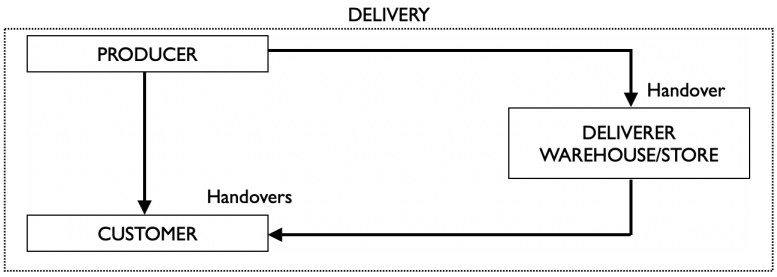


Figure 5.1: Logistic Chain

Firstly, random data will be generated in a consistent way to simulate a delivery, in particular, a delivery plan with a set of handovers that goes randomly from 1 to 10. This information will be stored in three different centralized databases. The summary of the

databases used along with the information stored in each one can be found in Table 5.1.

Database	**Information Contained**
Deliveries	Producer, Customer, SSCC, Location and Time
Handovers	PHO, SSCC, Originator, Recipient, Location and Time
Users	GCP and KeyPair (Public Key and Private Key) of each entity (also called user)

Table 5.1: Databases.

Once the data is correctly generated, the first entity involved in the transaction will enter its GCP as identification. If the GCP corresponds to an entity with incoming or outgoing handovers from a delivery plan (which is checked in the *Handovers* database), the user will proceed to the next step. Otherwise, a message will be displayed stating "There are no deliveries planned today".
If access is correct, a menu with the following options will be displayed:

1. Show my Handovers
2. Create QR For New Delivery
3. New Handover Message
4. Scan QR for Incoming Handover and Send Message
5. Search for Message
6. Show All BlockChain Messages

The first option has merely informative purposes as it will just display the incoming and outgoing handovers for this entity. The system will take from *Handovers* all the handovers that contain the GCP of this party and will store them in two different Arraylists (incoming and outgoing) depending on whether the entity is the sender or the recipient of the handover at issue. Then, after selecting this first option, the information regarding those handovers will be printed.

As all the information is encoded in QR codes, the following options deal with this matter. The first time the parcel is sent from the producer to the next participant, it should include a QR with information of the whole delivery plan so it can be passed on the chain. The second option is in charge of this function. Once it is selected, the system will ask for the SSCC, the unique identifier for the delivery. Again, if the SSCC corresponds to a delivery in *Deliveries*, the process will be completed successfully and a QR code will be shown on the screen. If not, an error message will be displayed.

The next 3 options constitute the main part of the development of the project as they represent the communication between entities. The first one is similar to the previous one but applied to handovers. Once this option is selected by the sender of a transaction, the platform will request a PHO (identifier for a handover) and look for it in *Handovers*. If such handover exists and is part of the outgoing handovers of the entity, the QR code will be created and displayed. This is done selecting the handover from the database and iterating the ArrayList of outgoing deliveries to check if the information of the handover is included there.

This QR would be on the parcel sent. Therefore, once it has arrived at the next stage of the chain it has to be scanned, which is function 4. When this option is selected, the QR code generated in the previous stage will be scanned, retrieving the information encoded on it. Then, the correctness of this information will be checked with the databases and the ArrayList of incoming deliveries. If every check is correct, the recipient of this transaction will then send a message to the blockchain network with information of the transaction. The details of these messages will be clarified in section 5.2 Communication Protocol.

When this last message is sent it means the parcel has arrived at the recipient of this particular handover. Therefore, the originator has to able to check this is correct. Function 5 allows this first entity to query for a message in the blockchain network by introducing the PHO, identifier of the handover which will also be used as an index in the network. There is a last option available that displays all the messages stored in the blockchain network. It can be understood as a way of controlling or checking the history of transactions performed in the network.

5.2 Communication Protocol

In the previous section, General Aspects, the functions of the platform were explained in a generic manner. Now this part will cover the details of the communication protocol so it is easier to understand how the transmission of data works in this environment. An schematic diagram of this protocol is shown in Fig. 5.2.

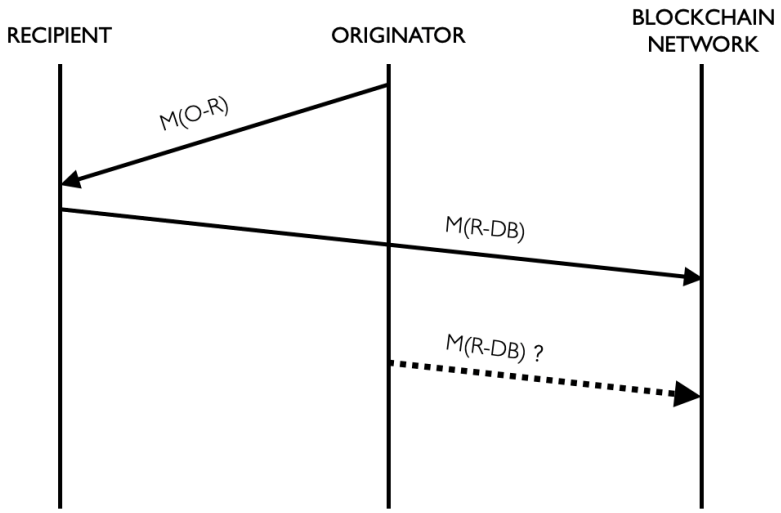


Figure 5.2: Communication Flow

This process can be summarized in three steps, which correspond to functions 3, 4 & 5 in 5.1 General Aspects as well as the ones mentioned in 4.3 Methodology.

1. The originator has already identified itself by means of GCP, selected option 3, and entered PHO of the handover. The system will then generate a message m1=(PHO, SSCC, GCPO, DateTimeO, LocationO, RO), being R0 a nonce generated by this entity. Once the message is created, the originator will compute the hash H(m1)=SHA-256(m1), encrypting it with its private key: mO-R=(m1, [H(m1)]O) (As seen in Fig. 5.2). This information (m0-R) with the message and its hash will be encoded in a QR generated by this identity, which will then be sent to the recipient of this transaction, the next entity in the delivery plan.

2. The second stage is from the perspective of the recipient of this transaction in particular. Once the QR is received, it will be scanned at the destination of arrival by selecting option 4. Retrieving the public key of the originator from *Users*, using the GCP of the first entity as an index, the recipient can extract the hash from the QR code. As the message is already encoded, this entity will compute again the hash of the message H(m1)=SHA-256(m1) and compare it with the one found in the QR and decrypted, checking if the message is in fact coming from the expected originator.
   If the comparison results in a positive output and the computed and encoded hashes match, the recipient will send a message m(R-DB)=(PHO, SSCC, GCPO,GCPR, DateTimeO, LocationO, DateTimeR, LocationR), which contains the whole information of the handover to the blockchain network, encrypting it with its public key.

3. The last part of the communication process relies again on the originator of the transaction by selecting option 5. It will allow this entity to perform a query in the blockchain network with and index, in this case, the PHO of the handover that was sent to the network in the previous step. From *Users*, this first entity will take the private key of the second one and then decrypt the message that was in the network. If everything is correct the communication process is over and the parcel can continue its way.

This process will be repeated each time there is a transaction or handover between two different entities, from the producer until the parcel has reached the final destination, which is the customer. Therefore, it is possible to check in which step the parcel is and monitor it to ensure it arrives at the intended entity.

Chapter 6

Results

6.1 The Platform

This project aims to create an innovative platform for the management of handovers and deliveries. Therefore, the most important result obtained is the design and development of this software solution as well as the data that makes it possible to simulate these transactions in a real way. This section of the project will detail how this platform is functioning correctly and serves the purpose. All the code used can be found in Appendix C, Code.

The first figure, Fig.6.1 is an output taken from the terminal. It shows how the network is successfully built and working, the channel created and chaincode is deployed and instantiated. For greater clearance, the parts in which messages confirm this is correct have been framed in green. Once the blockchain network is working, the data has to be generated to simulate a delivery plan and subsequent handovers.
To illustrate how this platform works, some data will be generated and used to simulate the whole process. Fig.6.2 shows information regarding a delivery plan and the corresponding handovers. As can be seen, data is built in a consistent manner, as field SSCC (identifier for the delivery plan) is the same for the delivery and all the handovers, the producer is the sender of the first transaction and the customer is the recipient of the last one. Information regarding time and location is maintained the same to simplify the process.

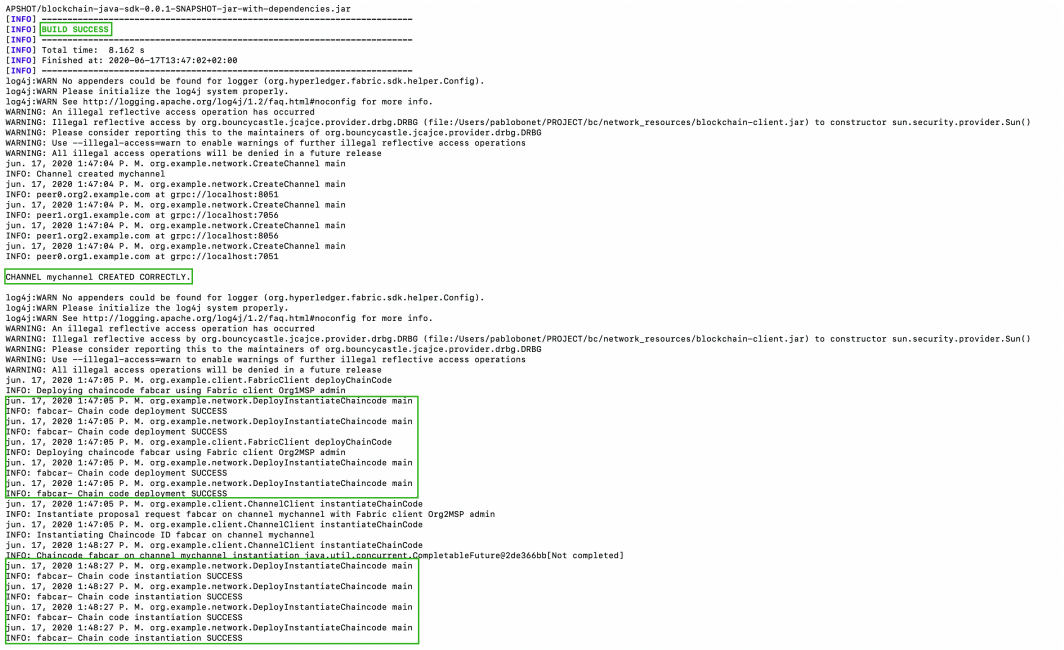


Figure 6.1: Blockchain Network & Chaincode

For this example, the first handover will be used as a reference. The first step implies the sender generating the first message. Fig.6.3 contains an output of the terminal in which the identification of the first entity as well as sending the first message takes place. The next one, Fig.6.4 shows how the QR is scanned and the message retrieved from it is verified correctly. Then, the encrypted message to the blockchain network is sent. The last figure of this section, 6.5 represents the last step in the communication process, showing how the originator performs a query in the network and finds the message with the specified PHO. Once the message is found, the first entity has to decrypt it, and as it is displayed, the message is decrypted and the information contained is the same as the one originally sent. This represents the process that has been described several times in this report, but in a real way. As can be seen, the platform fulfills its objectives as it has successfully managed a handover within a delivery plan, ensuring the parcel has arrived at the correct recipient.

```
DELIVERY INFORMATION
====================
SSCC: 171764135877911193
PRODUCER: 717641358
CUSTOMER: 59950234
DELIVERY PERIOD: 2020-06-22 - 2020-06-25
LOCATION: 59.33459 , 18.06324

[
HANDOVER INFORMATION
====================
PHO: 732544651
SSCC: 171764135877911193
ORIGINATOR: 717641358
RECIPIENT: 527474053
DELIVERY PERIOD: 2020-06-22 - 2020-06-25
LOCATION: 59.33459 , 18.06324,
HANDOVER INFORMATION
====================
PHO: 918196183
SSCC: 171764135877911193
ORIGINATOR: 527474053
RECIPIENT: 556121780
DELIVERY PERIOD: 2020-06-22 - 2020-06-25
LOCATION: 59.33459 , 18.06324,
HANDOVER INFORMATION
====================
PHO: 907452359
SSCC: 171764135877911193
ORIGINATOR: 556121780
RECIPIENT: 59950234
DELIVERY PERIOD: 2020-06-22 - 2020-06-25
LOCATION: 59.33459 , 18.06324]
```

Figure 6.2: Data Generated

```
WELCOME

Introduce your GCP:

717641358


HANDOVERS MANAGEMENT

Select One Option:

1. Show my Handovers
2. Create QR For New Delivery
3. New Handover Message
4. Scan QR for Incoming Handover and Send Message
5. Search for Message
6. Show All BlockChain Messages
7. Exit
3
Introduce the PHO:
732544651
```

(a) First Message

(b) QR Code

Figure 6.3: Sending the First Message

```
Introduce your GCP:

527474053


HANDOVERS MANAGEMENT

Select One Option:

1. Show my Handovers
2. Create QR For New Delivery
3. New Handover Message
4. Scan QR for Incoming Handover and Send Message
5. Search for Message
6. Show All BlockChain Messages
7. Exit
4

MESSAGE VERIFIED CORRECTLY.

HANDOVER INFORMATION
====================
PHO: 732544651
SSCC: 171764135877911193
ORIGINATOR: 717641358
RECIPIENT: 527474053
DELIVERY PERIOD: 2020-06-22 - 2020-06-25
LOCATION: 59.33459 , 18.06324
log4j:WARN No appenders could be found for logger (org.hyperledger.fabric.sdk.helper.Config).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.bouncycastle.jcajce.provider.drbg.DRBG (file:/Users/pablobonet/bc/java/lib/blockchain-client.jar)
WARNING: Please consider reporting this to the maintainers of org.bouncycastle.jcajce.provider.drbg.DRBG
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
jun. 22, 2020 10:09:11 A. M. org.example.client.ChannelClient sendTransactionProposal
INFO: java.util.concurrent.CompletableFuture@4ba6ec50[Not completed]

THE MESSAGE HAS BEEN SUCCESFULLY SENT
```

Figure 6.4: Scanning QR & Sending Message

```
Select One Option:

1. Show my Handovers
2. Create QR For New Delivery
3. New Handover Message
4. Scan QR for Incoming Handover and Send Message
5. Search for Message
6. Show All BlockChain Messages
7. Exit
5
Introduce the PHO of the Message to Query:
732544651
log4j:WARN No appenders could be found for logger (org.hyperledger.fabric.sdk.helper.Config).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.bouncycastle.jcajce.provider.drbg.DRBG (file:/Users/pablobonet/bc/java/lib/blockchain-client.jar)
WARNING: Please consider reporting this to the maintainers of org.bouncycastle.jcajce.provider.drbg.DRBG
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
jun. 22, 2020 10:09:40 A. M. org.example.client.ChannelClient queryByChainCode
INFO: Querying queryMessage on channel mychannel


MESSAGE WITH PHO 732544651 FOUND.


DECRYPTING MESSAGE...


DECRYPTED MESSAGE:

PHO: 732544651
SSCC: 171764135877911193
Originator: 717641358
Recipient: 527474053
From: 2020-06-22
To: 2020-06-22
Longitude: 18.06324
Latitude: 59.33459
```

Figure 6.5: Query in Blockchain Network

6.2 Performance Analysis

The use of blockchain technology brings enormous benefits to this project, especially in terms of privacy, traceability, security ... [28]. However, this implies a slight reduction of efficiency, especially when the amount of transactions logged in the blockchain network is high. Figure 6.6 represents the time the system takes to perform a single query in the blockchain network as a function of the number of transactions stored in it. Due to some technical limitations, the maximum amount of transactions that have been stored are 30.000. Increasing this number implied losing some functionalities so it was decided to measure up to this benchmark. It may seem the increase in time is modest, but it is rather significant considering it goes from 1,2 seconds to almost 2 seconds, which represents an increase of 67%. The rise in time follows an almost linear pattern, which leads to thinking that in an unlimited environment, storing more transactions would generate a greater increase time that would also be linear.

To obtain these results, several different measurements have been performed. In particular, for each amount of transactions, the time has been computed three times and a mean value has been obtained. Appendix Time Measurements contains this computations.

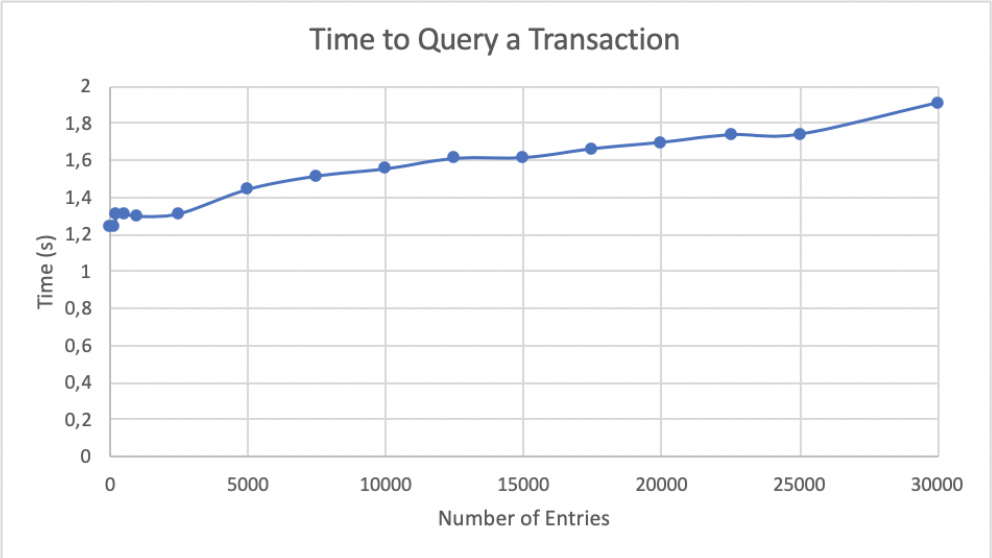


Figure 6.6: Network Performance

Chapter 7

Conclusions and Future Development

7.1 Conclusions

The logistics industry has always had issues in coordination and trust, partly due to the different entities that may be involved in a transaction. The use of different technologies and in particular blockchain has helped this field, especially in terms of security and efficiency [29]. There can be certain challenges when implementing new technologies in businesses that have been operating for many years without them. Blockchain technology must be integrated with the existing management systems, which is not always an easy task [30]. However, blockchain technology is just starting to spread through all sectors, including logistics. In fact, according to a survey conducted by the World Economic Forum, by 2027 around 10% of the world's global GDP will be stored in blockchain [31].

This means every company in every field should at least consider using blockchain for some purpose. Fig. 7.1 shows the feasibility and impact of investing in blockchain solutions in the fields of transport and logistics. As it can be seen, investing in blockchain technology for shipping and management of assets (transactions) is feasible and can have a relatively high impact. This is significant due to the fact that this impact is even higher than in other sectors such as the automotive industry, manufacture or retail [32].

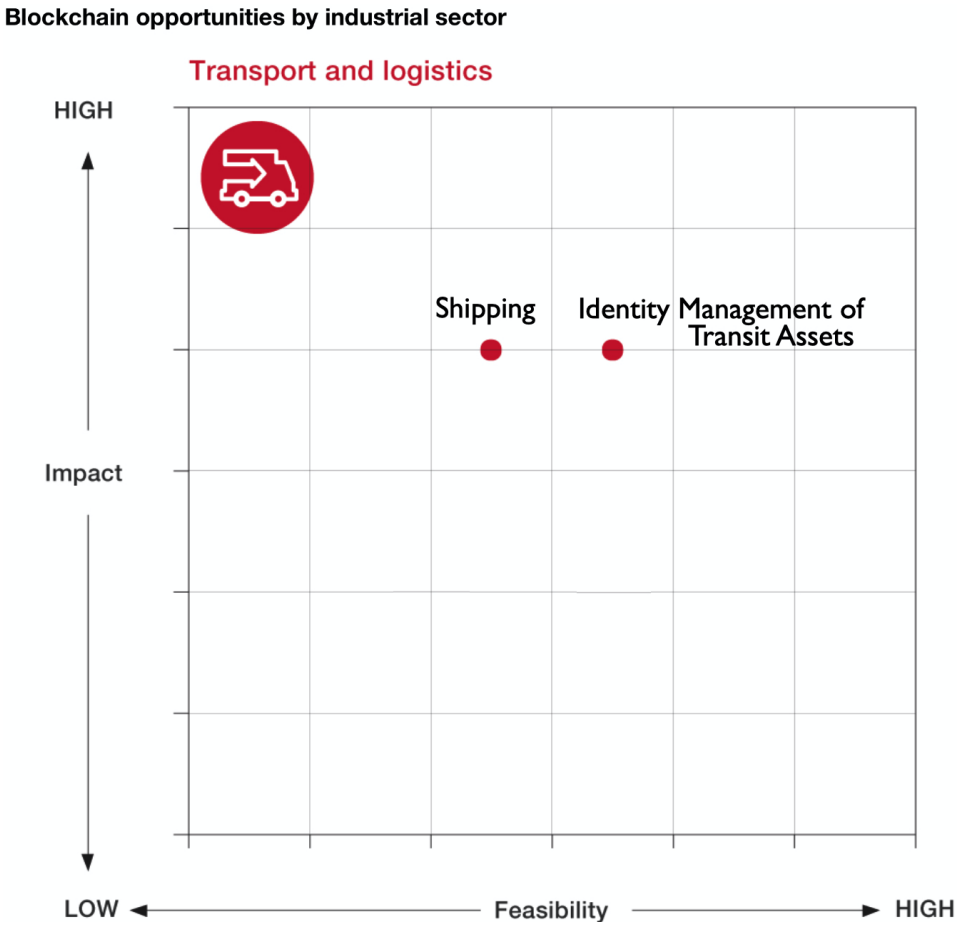


Figure 7.1: Blockchain & Logistics [32]

This project serves the intended purpose as it was designed for the management and tracking of handovers and deliveries within a delivery chain and it has proved to fulfill this task successfully. It also intended to use a distributed ledger for it and assess the feasibility of storing transactions there, which has been also completed.

The contribution of blockchain technology as well as cryptograpahy, has made the project a robust solution for the logistics sector and has proved this sort of techniques have a place in designing real modern software solutions.

7.2 Future Devlopment

The objectives presented for this project are fully covered as this platform is able to manage and track handovers and deliveries by means of a solution that integrates different technologies, especially a distributed ledger (blockchain technology). However, there are some aspects that despite being out of the scope of the project, could be included for further versions.

7.2.1 User Interface

The project has achieved its purpose of creating the platform and simulating in a real manner how it would function in a logistic environment. Nevertheless, there is no current application that doesn't have a user-friendly interface, as it is much easier to use it and clients usually prefer software solutions that are visually pleasurable.

The current version of this platform presents a basic interface based on the computer's terminal and does not use any external library to create a window in which the user can perform different actions. The reason for not developing this visual part is because the aim of the project is to assess the feasibility of using the distributed ledger for storing information about logistic transactions and performing queries on the network, which has nothing to do with the aesthetic part. Developing this would imply a heavier application, which could also slow the process and then backfire the whole project.

7.2.2 Prices

As every existing transaction, a delivery has and economic exchange implicit. For this version of the project, working with data without prices is enough, as it has been proved the main objective is covered. Further versions may include some price tag in every handover or delivery plan to make the transaction even more realistic. In fact, there has been developed a function to obtain average and maximum prices (using PHO as temporary input). The problem is that the time it takes to calculate any of them is extremely high for an unnecessary feature, so it was decided to leave it out.

7.2.3 Multicast Encryption

Privacy and security are two fundamental aspects of this software solution. This is the reason why public-key encryption is used. In the context of this project, every transaction has one sender and recipient, which is why this sort of cryptographic tool was selected as optimal. However, under other circumstances, transactions may occur involving several different entities at the same time. In the event of such conditions, the solution proposed is multicast encryption. Multicast Encryption allows the sender to address data (information about the transaction) to various recipients simultaneously [33]. Despite being a good solution for certain environments, it has some downsides. It is not always easy to guarantee that only the intended recipients will receive and decrypt the message, due to the fact that multicast groups are usually dynamic, and members are constantly changing.
Multicast Encryption can be an adequate solution if the framework allows it, but for the case of this particular project, traditional asymmetric cryptography is one step ahead as it guarantees only the recipient of each transaction is able to read the message decrypting it.

These three items are some examples of ideas for a future improved development of a platform that would better fit the logistics industry needs. Developing them over the foundations of the current project would make this a very powerful tool and an ideal solution for the market.

Bibliography

[1] Bhat, Dr. Shahid & Kansana, Keshav & Majid, Jenifur. (2016). A Review Paper on E-Commerce.

[2] Columbus, L. (2020). How COVID-19 Is Transforming E-Commerce. Forbes.

[3] Ali, F. (2020, March 3). A decade in review: Ecommerce sales vs. retail sales 2007 2019. Retrieved from Digital Commerce 360: https://www.digitalcommerce360.com/article/e-commerce-sales-retail-sales-ten-year-review/

[4] Donegan, C. (2019). State of the Connected Customer. San Francisco: Salesforce.

[5] Mitchell, V. (2018). Why customer trust is more vital to brand survival than it's ever been. CMO.

[6] McCarthy, N. (2014). Chart: The Biggest Data Breaches in U.S. History. Forbes.

[7] Joe Myers, K. W. (2019). These are the biggest risks facing our world in 2019. World Economic Forum.

[8] Nweke, L. O. (2017). Using the CIA and AAA Models to Explain Cybersecurity Activities. PM World Journal, Vol. VI, Issue XII – December 2017.

[9] Edem, Swathi & Vivek, G. & Rani, G.. (2016). Role of Hash Function in Cryptography. 10-13.

[10] Wagner, L. (2018, June 29). (Very) Basic Intro to Hash Functions (SHA-256, MD-5, etc). Retrieved from Good Audience: https://blog.goodaudience.com/very-basic-intro-to-hash-functions-sha-256-md-5-etc-ed721622ff8

[11] Rhodes, D. (2020, April 27). SHA-256: An Overview & Guide of the SHA-256 Algorithm. Retrieved from Komodo: https://komodoplatform.com/sha-256-algorithm/

[12] Milanov, E. (2009). The RSA Algorithm . Washington: University of Washington.

[13] Drescher, D. (2017). Blockchain Basics: A Non-Technical Introduction in 25 Steps. Apress.

[14] Vujicic, Dejan & Jagodic, Dijana & Randić, Sinisa. (2018). Blockchain technology, bitcoin, and Ethereum: A brief overview. 1-6.

[15] Euromoney Learning. (2020). What is blockchain? Retrieved from Euromoney: https://www.euromoney.com/learning/blockchain-explained/what-is-blockchain

[16] Tasca, Paolo & Tessone, Claudio. (2019). A Taxonomy of Blockchain Technologies: Principles of Identification and Classification. Ledger. 4.

[17] Sharma, T. K. (2019, November 13). PERMISSIONED AND PERMISSIONLESS BLOCKCHAINS: A COMPREHENSIVE GUIDE. Retrieved from Blockchain Council: https://www.blockchain-council.org/blockchain/permissioned-and-permissionless-blockchains-a-comprehensive-guide/

[18] Gadgilwar, C. (2019, December 26). Overview of Smart Contracts and Chaincode in HyperLedger Fabric Blockchain. Retrieved from Medium: https://medium.com/@chetan.gadgilwar/overview-of-smart-contracts-and-chaincode-in-hyperledger-fabric-blockchain-e09ef0d7faab

[19] Cachin, C. (2016). Architecture of the Hyperledger Blockchain Fabric. Zurich: IBM Research.

[20] Babak Bashari Rad, H. J. (2017, March). An Introduction to Docker and Analysis of its Performance. International Journal of Computer Science and Network Security, pp. 228-235.

[21] Church, M., Ruiz, M., Seifert, A., & Marshall, T. (n.d.). Docker Swarm Reference Architecture: Exploring Scalable, Portable Docker Container Networks. Retrieved from Docker Success Center: https://success.docker.com/article/networking

[22] Rilee, K. (22, February 2018). Simpler setup for Hyperledger Fabric on Kubernetes using Docker-in-Docker. Retrieved from Medium: https://medium.com/kokster/simpler-setup-for-hyperledger-fabric-on-kubernetes-using-docker-in-docker-8346f70fbe80

[23] Cm, S. (2019, June 21). Top 10 delivery management software in the World (June 2019) . Retrieved from Medium: https://medium.com/@saran_51307/top-10-delivery-management-software-in-the-world-june-2019-5f7715152c32

[24] Maersk. (2019, September 20). A game changer for global trade. Retrieved from Maersk: https://www.maersk.com/news/articles/2019/09/20/a-game-changer-for-global-trade

[25] Fremont, V., & Jonathan, G. M. (2018). Can Blockchain Technology Solve Trust Issues in Industrial Networks? Stockholm: Gävle University.

[26] Payne, J. (2018, April 17). Supply Chain Insight: Improve Accuracy and Visibility with GS1 Standards. Inbound Logistics.

[27] Zhao, W. (2012). Beliefs And Misbeliefs About Open Source Software. Forbes.

[28] Koksal, I. (2019). The Benefits Of Applying Blockchain Technology In Any Industry. Forbes.

[29] PwC. (2016). The future of the logistics industry . Shifting patterns, 5-9.

[30] Robinson, A. (n.d.). Blockchain Technology in Logistics: What Are the Implementation Challenges? Retrieved from Cerasis: https://cerasis.com/blockchain/

[31] Deep shift: Technology tipping points and societal impact, World Economic Forum, September 2015, weforum.org.

[32] Carson, B., Romanelli, G., Walsh, P., & Zhumaev, A. (2018, June 19). Blockchain beyond the hype: What is the strategic business value? Retrieved from McKinsey Digital: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/blockchain-beyond-the-hype-what-is-the-strategic-business-value#

[33] Brooks, R. & Pillai, Brijesh & Pirretti, Matthew & Weigle, Michele. (2009). Multicast Encryption Infrastructure for Security in Sensor Networks. IJDSN. 5. 139-157.

[34] Kroll, C., Warchold, A. & Pradhan, P. (2019) Sustainable Development Goals (SDGs): Are we successful in turning trade-offs into synergies?. Palgrave Commun 5, 140.

[35] United Nations. (2018). Sustainable Development Goals. Retrieved from UN: https://sustainabledevelopment.un.org/?menu=1300

[36] United Nations. (2020). Promote inclusive and sustainable economic growth, employment and decent work for all. Retrieved from United Nations: https://www.un.org/sustainabledevelopment/economic-growth/

[37] United Nations. (2020). Build resilient infrastructure, promote sustainable industrialization and foster innovation. Retrieved from United Nations: https://www.un.org/sustainabledevelopment/infrastructure-industrialization/

Appendix A

Alignment with SDG

In 2015 resolution 70/1 of United Nations established 17 global goals for the future as a part of 2030 Agenda [34]. The aim is to achieve a better future for the world in fields such as equality, education, health, climate or justice among many others. Fig. A.1 contains the official sign with the 17 objectives.

Figure A.1: Sustainable Development Goals [35]

This project complies with some of these 17 objectives. This Appendix will describe the goal and state of the objectives covered as well as how can these goals be aligned with the project developed.

Decent Work and Economic Growth (8)

The number of unemployed people keeps decreasing every year. This enhances global economic growth, which gives many people the chance to have a brighter and better future. However, there is still a long journey until decent work is achieved everywhere and for everyone. Despite the fact that the world is recovering from the 2008 financial crisis, it is uncertain how deep the Covid-19 recession will affect worldwide [36].

As it was mentioned in previous sections of this report, the pandemic has lead to an increase in deliveries due to massive online buying. This is the reason why this platform is a great opportunity to create decent jobs in various fields, from software designing and IT maintenance to deliverers and carriers.

Industry, Innovation and Infrastructure (9)

Financial, economic and industrial infrastructure improves every year. However, there is still a significant difference between developed countries and the rest of the world in this matter. In order to support this growth and spread it to all the nations in the world, transportation services are extremely important, as most of the merchandise existing in the world is moving between different areas [37]. Therefore, with this solution is a great opportunity to improve infrastructure in the industry of logistics as it clearly brings benefits in technological matters and integrates blockchain with other technologies making it an original and genuine solution.

Responsible Consumption and Production (12)

It is of extreme importance to change the current model of consumption, as it is based on mass production and does not take into consideration the impact this may have on our health or the environment. The available resources should be used responsively to ensure there will be more available in the future. This project can contribute to this goal by connecting different entities that are committed to it. Establishing partnerships between conscious organizations and consumers is the first step to change the production model to a better and more responsible one.

Appendix B

Time Measurements

Entries	**Time (ms)**	2500	1371	15000	1634
1	1293	2500	1347	15000	1595
1	1289	2500	1213	17500	1749
1	1138	5000	1513	17500	1692
100	1162	5000	1452	17500	1544
100	1259	5000	1452	20000	1757
100	1295	7500	1581	20000	1677
250	1304	7500	1483	22500	1802
250	1305	7500	1476	22500	1802
250	1335	10000	1528	22500	1673
500	1256	10000	1475	25000	1697
500	1318	10000	1655	25000	1732
500	1352	12500	1598	25000	1792
1000	1395	12500	1610	30000	1927
1000	1230	12500	1620	30000	1991
1000	1258	15000	1610	30000	1813

Table B.1: Time Measurements.

Table B.1, contains the original measurements of time as function of the number of trans-

actions stored in the network. As it can be seen, there are three different measures for each number of transactions stored, with the purpose of making it more accurate. To represent it in the figure shown in section Performance Analysis, the average of this measures has to be calculated. This mean values can be seen below in Table B.2.

Entries	**Time (ms)**
1	1240
100	1238,666667
250	1314,666667
500	1308,666667
1000	1294,333333
2500	1310,333333
5000	1442
7500	1513,333333
10000	1552,666667
12500	1609,333333
15000	1613
17500	1661,666667
20000	1696,333333
22500	1737,333333
25000	1740,333333
30000	1910,333333

Table B.2: Average Time Measurements.

Appendix C

Code

The following pages include the code that has been used to develop the platform. It will just include the code that has been written specifically for this purpose, but not other classes such as the Hyperledger Fabric SDK, as it can be obtained at the project's website or GithHub and including it would make this report extremely extensive.
These classes have been organized in different categories (coinciding with the packages), that can be seen in Table C.1

Category/Package	**Classes**
Crypto	Crypto.java
Data	DataGenerator.java
DB	DAO.java
Delivery	Delivery.java / Entity.java Handover.java / Message.java
QR	ReadQR.java / WindowQR.java
Chaincode	Chaincode.go
App	AppDelivery.java

Table C.1: Organization of Code.

```java
package crypto;

import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature;
import java.util.Base64;
import javax.crypto.Cipher;
import static java.nio.charset.StandardCharsets.UTF_8;
import java.math.BigInteger;
import java.nio.charset.StandardCharsets;

public class Crypto {

 public static KeyPair generateKeyPair() throws Exception
 {
    KeyPairGenerator generator = KeyPairGenerator.getInstance("RSA");
    generator.initialize(2048);
    KeyPair pair = generator.generateKeyPair();
    return pair;
 }

 public static String sign(String plainText, PrivateKey privateKey) throws Exception
 {
    Signature privateSignature = Signature.getInstance("SHA256withRSA");
    privateSignature.initSign(privateKey);
    privateSignature.update(plainText.getBytes(UTF_8));

    byte[] signature = privateSignature.sign();

    return Base64.getEncoder().encodeToString(signature);
 }

 public static boolean verify(String plainText, String signature, PublicKey publicKey) throws Exception
 {
    Signature publicSignature = Signature.getInstance("SHA256withRSA");
    publicSignature.initVerify(publicKey);
    publicSignature.update(plainText.getBytes(UTF_8));

    byte[] signatureBytes = Base64.getDecoder().decode(signature);

    return publicSignature.verify(signatureBytes);
 }

 public static byte[] getSHA(String input) throws NoSuchAlgorithmException
 {
  MessageDigest md = MessageDigest.getInstance("SHA-256");
  return md.digest(input.getBytes(StandardCharsets.UTF_8));
 }

 public static String toHexString(byte[] hash)
 {
  BigInteger number = new BigInteger(1, hash);
  StringBuilder hexString = new StringBuilder(number.toString(16));
  while (hexString.length() < 32)
  {
   hexString.insert(0, '0');
  }

  return hexString.toString();
 }
```

```
public static String encrypt(String plainText, PublicKey publicKey) throws Exception
{
 Cipher encryptCipher = Cipher.getInstance("RSA");
        encryptCipher.init(Cipher.ENCRYPT_MODE, publicKey);
 byte[] cipherText = encryptCipher.doFinal(plainText.getBytes(UTF_8));
 return Base64.getEncoder().encodeToString(cipherText);
}


public static String decrypt(String cipherText, PrivateKey privateKey) throws Exception
{
 byte[] bytes = Base64.getDecoder().decode(cipherText);
 Cipher decriptCipher = Cipher.getInstance("RSA");
        decriptCipher.init(Cipher.DECRYPT_MODE, privateKey);
 return new String(decriptCipher.doFinal(bytes), UTF_8);
}

}
```

```
package data;

import java.util.Random;
import org.example.chaincode.invocation.InvokeChaincode;
import db.DAO;
import delivery.Delivery;
import delivery.Entity;
import delivery.Handover;

public class DataGenerator {

 public static void insertDeliveryHandover() throws Exception
 {
  Delivery d = new Delivery();
  d.generateDelivery();
  DAO.includeDelivery(d);
  Random rand = new Random();
  int n = rand.nextInt(10);
  if(n==0)
  {
   n=1;
  }
  if(n==1)
  {
   Handover h = new Handover();
   h.generateHandover();
   d.setAttributes(h);
   d.setOriginator(h);
   d.setRecipient(h);
   d.addHandover(h);
   DAO.includeHandover(h);
   DAO.includeUser(h.getOriginator());
   DAO.includeUser(h.getRecipient());
  }
  else
  { Entity nextorg = null;
   for(int i =0;i<n;i++)
   {
    Handover h = new Handover();
    h.generateHandover();
    d.setAttributes(h);
    h.setOriginator(nextorg);
    d.addHandover(h);

    if(i==0)
    {
     h.setOriginator(d.getProducer());
     DAO.includeUser(d.getProducer());
    }

    if(i==(n-1))
    {
     h.setRecipient(d.getCustomer());
    }
    nextorg = h.getRecipient();
    DAO.includeHandover(h);
    DAO.includeUser(nextorg);
   }
  }
  System.out.println(d);
 }

}
```

```
package db;

import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDate;
import java.util.ArrayList;
import delivery.Delivery;
import delivery.Entity;
import delivery.Handover;

public class DAO
{
 public static void includeDelivery(Delivery d)
 {
  Connection con = null;
  PreparedStatement pstmt = null;

  try
  {
   con = DriverManager.getConnection("jdbc:derby:/Users/pablobonet/MyDB");
   pstmt = con.prepareStatement("INSERT INTO DELIVERIES VALUES (?,?,?,?,?,?,?)");
   pstmt.setLong(1,d.getSSCC());
   pstmt.setInt(2,d.getProducer().getGCP());
   pstmt.setInt(3,d.getCustomer().getGCP());
   pstmt.setDate(4,Date.valueOf(d.getFrom()));
   pstmt.setDate(5, Date.valueOf(d.getTo()));
   pstmt.setFloat(6,d.getLatitude());
   pstmt.setFloat(7,d.getLongitude());

   pstmt.executeUpdate();
  }
  catch(SQLException e)
  {
   e.printStackTrace();
  }
  finally
  {
   try
   {
    if(pstmt!=null)
    {
     pstmt.close();
    }
   }
   catch(SQLException e) {}
   try
   {
    if(con!=null)
    {
     con.close();
    }
   }
   catch(SQLException e) {}
  }
 }

 public static void includeHandover(Handover h)
 {
  Connection con = null;
  PreparedStatement pstmt = null;

  try
  {
   con = DriverManager.getConnection("jdbc:derby:/Users/pablobonet/MyDB");
   pstmt = con.prepareStatement("INSERT INTO HANDOVERS VALUES (?,?,?,?,?,?,?,?)");
   pstmt.setLong(1,h.getPHO());
   pstmt.setLong(2,h.getSSCC());
   pstmt.setInt(3,h.getOriginator().getGCP());
   pstmt.setInt(4,h.getRecipient().getGCP());
   pstmt.setDate(5,Date.valueOf(h.getFrom()));
   pstmt.setDate(6, Date.valueOf(h.getTo()));
   pstmt.setFloat(7,h.getLatitude());
   pstmt.setFloat(8,h.getLongitude());

   pstmt.executeUpdate();
  }
  catch(SQLException e)
  {
   e.printStackTrace();
  }
  finally
  {
   try
   {
    if(pstmt!=null)
    {
     pstmt.close();
    }
   }
   catch(SQLException e) {}
   try
   {
    if(con!=null)
```

```
  {
   con.close();
  }
 }
 catch(SQLException e) {}
}
}

public static void includeUser(Entity ent) throws InvalidKeySpecException, NoSuchAlgorithmException
{
 Connection con = null;
 PreparedStatement pstmt = null;

 try
 {
  con = DriverManager.getConnection("jdbc:derby:/Users/pablobonet/MyDB");
  pstmt = con.prepareStatement("INSERT INTO USERS VALUES (?,?,?)");
  pstmt.setInt(1,ent.getGCP());
  pstmt.setBytes(2,ent.getPublicBytes());
  pstmt.setBytes(3,ent.getPrivateBytes());
  pstmt.executeUpdate();

 }
 catch(SQLException e)
 {
  e.printStackTrace();
 }
 finally
 {
  try
  {
   if(pstmt!=null)
   {
    pstmt.close();
   }
  }
  catch(SQLException e) {}
  try
  {
   if(con!=null)
   {
    con.close();
   }
  }
  catch(SQLException e) {}
 }
}

public static void updateUser(Entity ent) throws InvalidKeySpecException, NoSuchAlgorithmException
{
 Connection con = null;
 PreparedStatement pstmt = null;

 try
 {
  con = DriverManager.getConnection("jdbc:derby:/Users/pablobonet/MyDB");
  pstmt = con.prepareStatement("UPDATE USERS SET PBKEY = ? , PRKEY = ? WHERE GCP = ?");
  pstmt.setBytes(1,ent.getPublicBytes());
  pstmt.setBytes(2,ent.getPrivateBytes());
  pstmt.setInt(3,ent.getGCP());

  pstmt.executeUpdate();

 }
 catch(SQLException e)
 {
  e.printStackTrace();
 }
 finally
 {
  try
  {
   if(pstmt!=null)
   {
    pstmt.close();
   }
  }
  catch(SQLException e) {}
  try
  {
   if(con!=null)
   {
    con.close();
   }
  }
  catch(SQLException e) {}
 }
}

public static Entity selectUser(int GCP) throws InvalidKeySpecException, NoSuchAlgorithmException
{
 Connection con = null;
 PreparedStatement pstmt = null;
 ResultSet rs = null;
 Entity user = null;

 try
 {
  con = DriverManager.getConnection("jdbc:derby:/Users/pablobonet/MyDB");
```

```
 pstmt = con.prepareStatement("SELECT * FROM USERS WHERE GCP = ?");
 pstmt.setInt(1,GCP);
 rs = pstmt.executeQuery();
 rs.next();
 user = new Entity(rs.getInt(1),rs.getBytes(2),rs.getBytes(3));
}
catch(SQLException e)
{
 e.printStackTrace();
}
finally
{
 try
 {
  if(pstmt!=null)
  {
   pstmt.close();
  }
 }
 catch(SQLException e) {}
 try
 {
  if(con!=null)
  {
   con.close();
  }
 }
 catch(SQLException e) {}
}
return user;
}

public static ArrayList<Handover> myOutgoingHandovers(int GCP) throws Exception
{
 Connection con = null;
 PreparedStatement pstmt = null;
 ResultSet rs = null;
 ArrayList<Handover> handovers = new ArrayList<Handover>();
 LocalDate ld = LocalDate.now();
 Date today =  Date.valueOf(ld);

 try
 {
  con = DriverManager.getConnection("jdbc:derby:/Users/pablobonet/MyDB");
  pstmt = con.prepareStatement("SELECT * FROM HANDOVERS WHERE ORIGINATOR = ? AND FROMD = ?");
  pstmt.setInt(1,GCP);
  pstmt.setDate(2,today);
  rs = pstmt.executeQuery();
  while(rs.next())
  {
   handovers.add(new
Handover(rs.getInt(1),rs.getLong(2),rs.getInt(3),rs.getInt(4),Date.valueOf((rs.getDate(5)).toString()).toLocalDate(),Date.valueOf((rs.getDate(6)).toString()).toLocalDate(),rs.getFloat(7),rs.getFloat(8)));

  }
 }
 catch(SQLException e)
 {
  e.printStackTrace();
 }
 finally
 {
  try
  {
   if(pstmt!=null)
   {
    pstmt.close();
   }
  }
  catch(SQLException e) {}
  try
  {
   if(con!=null)
   {
    con.close();
   }
  }
  catch(SQLException e) {}
 }
 return handovers;
}

public static ArrayList<Handover> myIncomingHandovers(int GCP) throws Exception
{
 Connection con = null;
 PreparedStatement pstmt = null;
 ResultSet rs = null;
 ArrayList<Handover> handovers = new ArrayList<Handover>();
 LocalDate ld = LocalDate.now();
 Date today =  Date.valueOf(ld);

 try
 {
  con = DriverManager.getConnection("jdbc:derby:/Users/pablobonet/MyDB");
  pstmt = con.prepareStatement("SELECT * FROM HANDOVERS WHERE RECIPIENT = ? AND FROMD = ?");
  pstmt.setInt(1,GCP);
  pstmt.setDate(2,today);
  rs = pstmt.executeQuery();

  while(rs.next())
```

```
  {
   handovers.add(new
Handover(rs.getInt(1),rs.getLong(2),rs.getInt(3),rs.getInt(4),Date.valueOf((rs.getDate(5)).toString()).toLocalDate(),Date.valueOf((rs.getDate(6)).toString()).toLocalDate(),rs.getFloat(7),rs.getFloat(8)));

  }
 }
 catch(SQLException e)
 {
  e.printStackTrace();
 }
 finally
 {
  try
  {
   if(pstmt!=null)
   {
    pstmt.close();
   }
  }
  catch(SQLException e) {}
  try
  {
   if(con!=null)
   {
    con.close();
   }
  }
  catch(SQLException e) {}
 }
 return handovers;
}

 public static Handover searchHandover(int pho) throws Exception
 {
  Connection con = null;
  PreparedStatement pstmt = null;
  ResultSet rs = null;
  Handover handover = null;

  try
  {
   con = DriverManager.getConnection("jdbc:derby:/Users/pablobonet/MyDB");
   pstmt = con.prepareStatement("SELECT * FROM HANDOVERS WHERE PHO = ?");
   pstmt.setInt(1,pho);
   rs = pstmt.executeQuery();
   rs.next();
   handover = new
Handover(rs.getInt(1),rs.getLong(2),rs.getInt(3),rs.getInt(4),Date.valueOf((rs.getDate(5)).toString()).toLocalDate(),Date.valueOf((rs.getDate(6)).toString()).toLocalDate(),rs.getFloat(7),rs.getFloat(8));

  }
  catch(SQLException e)
  {
   e.printStackTrace();
  }
  finally
  {
   try
   {
    if(pstmt!=null)
    {
     pstmt.close();
    }
   }
   catch(SQLException e) {}
   try
   {
    if(con!=null)
    {
     con.close();
    }
   }
   catch(SQLException e) {}
  }
  return handover;
 }


 public static Delivery searchDelivery(long sscc) throws Exception
 {
  Connection con = null;
  PreparedStatement pstmt = null;
  ResultSet rs = null;
  Delivery d = null;

  try
  {
   con = DriverManager.getConnection("jdbc:derby:/Users/pablobonet/MyDB");
   pstmt = con.prepareStatement("SELECT * FROM DELIVERIES WHERE SSCC = ?");
   pstmt.setLong(1,sscc);
   rs = pstmt.executeQuery();
   rs.next();
   d = new Delivery(rs.getLong(1),rs.getInt(2),rs.getInt(3),Date.valueOf((rs.getDate(4)).toString()).toLocalDate(),Date.valueOf((rs.getDate(5)).toString()).toLocalDate(),rs.getFloat(6),rs.getFloat(7));
  }
  catch(SQLException e)
  {
   e.printStackTrace();
  }
  finally
  {
```

```
   try
   {
    if(pstmt!=null)
    {
     pstmt.close();
    }
   }
   catch(SQLException e) {}
   try
   {
    if(con!=null)
    {
     con.close();
    }
   }
   catch(SQLException e) {}
  }
  return d;
 }

 public static ArrayList<Handover> searchHandovers(long sscc) throws Exception
 {
  Connection con = null;
  PreparedStatement pstmt = null;
  ResultSet rs = null;
  ArrayList<Handover> hs = new ArrayList<Handover>();

  try
  {
   con = DriverManager.getConnection("jdbc:derby:/Users/pablobonet/MyDB");
   pstmt = con.prepareStatement("SELECT * FROM HANDOVERS WHERE SSCC = ?");
   pstmt.setLong(1,sscc);
   rs = pstmt.executeQuery();


   while(rs.next())
   {
    hs.add(new
Handover(rs.getInt(1),rs.getLong(2),rs.getInt(3),rs.getInt(4),Date.valueOf((rs.getDate(5)).toString()).toLocalDate(),Date.valueOf((rs.getDate(6)).toString()).toLocalDate(),rs.getFloat(7),rs.getFloat(8)));

   }

  }
  catch(SQLException e)
  {
   e.printStackTrace();
  }
  finally
  {
   try
   {
    if(pstmt!=null)
    {
     pstmt.close();
    }
   }
   catch(SQLException e) {}
   try
   {
    if(con!=null)
    {
     con.close();
    }
   }
   catch(SQLException e) {}
  }
  return hs;
 }

}
```

```java
package delivery;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Random;

public class Delivery {

 Random rand = new Random();
 LocalDate today = LocalDate.now();

 long SSCC;
 Entity producer;
 Entity customer;
 LocalDate from;
 LocalDate to;
 float latitude;
 float longitude;
 ArrayList<Handover> handovers = new ArrayList<Handover>();


 public Delivery() throws Exception
 {
  producer = new Entity();
  customer = new Entity();
 }

 public Delivery(long SSCC, Entity producer, Entity customer , LocalDate from, LocalDate to, float latitude, float longitude,
ArrayList<Handover> handovers)
 {
  this.SSCC=SSCC;
  this.producer=producer;
  this.customer=customer;
  this.from=from;
  this.to=to;
  this.latitude=latitude;
  this.longitude=longitude;
  this.handovers=handovers;
 }

 public Delivery(long SSCC,int producer,int customer,LocalDate from,LocalDate to,float latitude,float longitude) throws
Exception
 {
  this.SSCC=SSCC;
  Entity p = new Entity();
  p.setGCP(producer);
  this.producer=p;
  Entity c = new Entity();
  c.setGCP(customer);
  this.customer=c;
  this.from=from;
  this.to=to;
  this.latitude=latitude;
  this.longitude=longitude;
 }


 public void generateDelivery()
 {
  this.producer.setGCP(rand.nextInt(1000000000));
  this.SSCC=generateSSCC();
  this.customer.setGCP(rand.nextInt(1000000000));
  this.from=today;
  this.to=today.plusDays(rand.nextInt(15));;
```

```
 this.latitude=(float)59.334591;
 this.longitude=(float)18.063240;
}


public long getSSCC()
{
 return SSCC;
}

public Entity getProducer()
{
 return producer;
}

public Entity getCustomer()
{
 return customer;
}

public int getProducerGPG()
{
 return producer.getGCP();
}

public LocalDate getFrom()
{
 return from;
}
public LocalDate getTo()
{
 return to;
}
public float getLatitude()
{
 return latitude;
}
public float getLongitude()
{
 return longitude;
}
public long generateSSCC()
{
 long n = rand.nextInt(10);
 int f = getProducerGPG();
 int a = rand.nextInt(10000000);
 long res1 = Long.valueOf(String.valueOf(n) + String.valueOf(f) + String.valueOf(a));
 int c= Math.abs((int)res1);
 do
 {
  c = (int) (c/(rand.nextInt((int)c)+1));

 }while(c>9);
 long res = Long.valueOf(String.valueOf(res1) + String.valueOf(c));
 return res;
}

public void addHandover(Handover h)
{
 handovers.add(h);
}

public void setHandovers(ArrayList<Handover> h)
{
 this.handovers=h;
```

```
 }


 public void setAttributes(Handover h)
 {
  h.setSSCC(SSCC);
  h.setFrom(from);
  h.setTo(to);
  h.setLatitude(latitude);
  h.setLongitude(longitude);
 }

 public void setOriginator(Handover h)
 {
  h.setOriginator(producer);
 }

 public void setRecipient(Handover h)
 {
  h.setRecipient(customer);
 }

 public void getHandovers()
 {
  Iterator<Handover> it = handovers.iterator();
  while(it.hasNext())
  {
   Handover h = (Handover)it.next();
   System.out.println(h);
  }
 }

 @Override
 public String toString()
 {
  return "\nDELIVERY INFORMATION" + "\n====================" +
    "\nSSCC: " + SSCC +
    "\nPRODUCER: " + producer +
    "\nCUSTOMER: " + customer +
    "\nDELIVERY PERIOD: " + from + " - " + to +
    "\nLOCATION: " + latitude + " , " + longitude +
    "\n\n" + handovers.toString();

 }

}
```

```java
package delivery;

import java.security.KeyFactory;
import java.security.KeyPair;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;

import crypto.Crypto;

public class Entity
{
 int GCP;
 PublicKey pbk;
 PrivateKey prk;

 public Entity() throws Exception
 {
  KeyPair kp = Crypto.generateKeyPair();
  this.pbk=kp.getPublic();
  this.prk=kp.getPrivate();
 }


 public Entity(int GCP, byte[] publickey, byte[] privatekey) throws InvalidKeySpecException, NoSuchAlgorithmException
 {
  this.GCP=GCP;
  X509EncodedKeySpec spec = new X509EncodedKeySpec(publickey);
  KeyFactory kf = KeyFactory.getInstance("RSA");
  PublicKey publicKey = kf.generatePublic(spec);
  this.pbk=publicKey;

  PKCS8EncodedKeySpec spec1 = new PKCS8EncodedKeySpec(privatekey);;
  KeyFactory kf1 = KeyFactory.getInstance("RSA");
  PrivateKey privateKey = kf1.generatePrivate(spec1);
  this.prk=privateKey;
 }

 public void setGCP(int gcp)
 {
  this.GCP=gcp;
 }

 public int getGCP()
 {
  return GCP;
 }

 public PublicKey getPublicKey()
 {
  return pbk;
 }

 public PrivateKey getPrivateKey()
 {
  return prk;
 }

 public byte[] getPublicBytes()
 {
  return pbk.getEncoded();
 }
```

```
public byte[] getPrivateBytes()
{
 return prk.getEncoded();
}


@Override
public String toString()
{
 return "" + GCP;
}
}
```

```java
package delivery;

import java.time.LocalDate;
import java.util.Random;

public class Handover{

 Random rand = new Random();
 LocalDate today = LocalDate.now();

 public int PHO;
 public long SSCC;
 public Entity originator;
 public Entity recipient;
 public LocalDate from;
 public LocalDate to;
 public float latitude;
 public float longitude;

 public Handover() throws Exception
 {
  originator = new Entity();
  recipient = new Entity();
 }

 public Handover(int PHO,long SSCC,Entity originator,Entity recipient,LocalDate from,LocalDate to,float latitude,float longitude)
 {
  this.PHO=PHO;
  this.SSCC=SSCC;
  this.originator=originator;
  this.recipient=recipient;
  this.from=from;
  this.to=to;
  this.latitude=latitude;
  this.longitude=longitude;
 }

 public Handover(int PHO,long SSCC,int originator,int recipient,LocalDate from,LocalDate to,float latitude,float longitude)
throws Exception
 {
  this.PHO=PHO;
  this.SSCC=SSCC;
  Entity orig = new Entity();
  orig.setGCP(originator);
  this.originator=orig;
  Entity rec = new Entity();
  rec.setGCP(recipient);
  this.recipient=rec;
  this.from=from;
  this.to=to;
  this.latitude=latitude;
  this.longitude=longitude;
 }



 public void generateHandover()
 {
  this.PHO=rand.nextInt(1000000000);
  this.originator.setGCP(rand.nextInt(1000000000));
  this.recipient.setGCP(rand.nextInt(1000000000));
 }

 public void setOriginator(Entity org)
```

```
{
 this.originator=org;
}

public void setRecipient(Entity rec)
{
 this.recipient=rec;
}

public void setSSCC(long sscc)
{
 this.SSCC=sscc;
}

public void setFrom(LocalDate from)
{
 this.from=from;
}

public void setTo(LocalDate to)
{
 this.to=to;
}

public void setLatitude(float latitude)
{
 this.latitude=latitude;
}

public void setLongitude(float longitude)
{
 this.longitude=longitude;
}


public int getPHO()
{
 return PHO;
}
public long getSSCC()
{
 return SSCC;
}
public Entity getOriginator()
{
 return originator;
}

public Entity getRecipient()
{
 return recipient;
}

public LocalDate getFrom()
{
 return from;
}
public LocalDate getTo()
{
 return to;
}
public float getLatitude()
{
 return latitude;
}
```

```
 public float getLongitude()
 {
  return longitude;
 }

 @Override
 public String toString()
 {
  return "\nHANDOVER INFORMATION" + "\n====================" +
    "\nPHO: " + PHO +
    "\nSSCC: " + SSCC +
    "\nORIGINATOR: " + originator +
    "\nRECIPIENT: " + recipient +
    "\nDELIVERY PERIOD: " + from + " - " + to +
    "\nLOCATION: " + latitude + " , " + longitude ;
 }

}
```

```java
package delivery;

import java.security.NoSuchAlgorithmException;
import java.time.LocalDate;
import java.util.Random;

import crypto.Crypto;

public class Message
{
 int PHO;
 long SSCC;
 Entity originator;
 LocalDate from;
 float latitude;
 float longitude;
 long nonce;
 String hashsigned;


 public Message(int PHO, long SSCC, Entity originator, LocalDate from, float latitude,float longitude) throws
NoSuchAlgorithmException, Exception
 {
  this.PHO=PHO;
  this.SSCC=SSCC;
  this.originator=originator;
  this.from=from;
  this.latitude=latitude;
  this.longitude=longitude;
  this.nonce=generateNonce();
  this.hashsigned=Crypto.sign(Crypto.toHexString(Crypto.getSHA(createPartialMessage())) , originator.getPrivateKey());
 }

 public long generateNonce()
 {
  Random random = new Random();
  long nonce = random.nextLong();
  return nonce;
 }

 public String createPartialMessage()
 {
  String message = String.valueOf(PHO) + "," + String.valueOf(SSCC) + "A" + String.valueOf(originator.getGCP()) + "Z" +
String.valueOf(from.toString()) + String.valueOf(latitude) + "," + String.valueOf(longitude) + "N" + String.valueOf(nonce);
  return message;
 }

 public String createMessage()
 {
  String message = createPartialMessage() +"|"+ String.valueOf(hashsigned);
  return message;
 }

 public String getPHO()
 {
  return String.valueOf(PHO);
 }
}
```

```java
package qr;

import java.io.FileInputStream;
import java.io.IOException;
import javax.imageio.ImageIO;
import com.google.zxing.BinaryBitmap;
import com.google.zxing.MultiFormatReader;
import com.google.zxing.NotFoundException;
import com.google.zxing.Result;
import com.google.zxing.client.j2se.BufferedImageLuminanceSource;
import com.google.zxing.common.HybridBinarizer;

public class ReadQR {
  public static String readQR(){
   Result r = null;
        try {
           BinaryBitmap binaryBitmap = new BinaryBitmap(new HybridBinarizer(
                 new BufferedImageLuminanceSource(
                       ImageIO.read(new FileInputStream("/Users/pablobonet/Documents/qr2.png")))));
            r = new MultiFormatReader().decode(binaryBitmap);
        } catch (IOException e) {
           e.printStackTrace();
        } catch (NotFoundException e) {
         e.printStackTrace();
        }

        return r.toString();
  }


}
```

```java
package qr;

import com.google.zxing.BarcodeFormat;
import com.google.zxing.Writer;
import com.google.zxing.WriterException;
import com.google.zxing.common.BitMatrix;
import com.google.zxing.qrcode.QRCodeWriter;
import java.io.File;
import java.io.IOException;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;


@SuppressWarnings("serial")
public class WindowQR extends JFrame {

   public WindowQR(String s, int i) throws WriterException
   {
    BufferedImage image = createQR(s, 250);

      ImageIcon icon = new ImageIcon(image);
      JLabel label = new JLabel("");

      String filePath = "/Users/pablobonet/Documents/qr" + i + ".png";
      File myFile = new File(filePath);
      String fileType = "png";
      try
      {
       ImageIO.write(image, fileType, myFile);
      }
      catch(IOException e)
      {
       e.printStackTrace();
      }


      label.setIcon(icon);

      this.setIconImage(image);
      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      this.setTitle("QR");
      this.getContentPane().add(label);
      this.pack();
   }

   public BufferedImage createQR(String PHO, int size) throws WriterException
   {
      BitMatrix matrix;
      Writer wr = new QRCodeWriter();

      matrix = wr.encode(PHO, BarcodeFormat.QR_CODE, size,size);

      BufferedImage img = new BufferedImage(size, size, BufferedImage.TYPE_INT_RGB);

      for(int y = 0; y < size; y++) {
         for(int x = 0; x < size; x++) {
            int grayValue = (matrix.get(x, y) ? 0 : 1) & 0xff;
            img.setRGB(x, y, (grayValue == 0 ? 0 : 0xFFFFFF));
         }
      }
      return img;
   }
```

```go
package main

import (
 "bytes"
 "encoding/json"
 "fmt"
 "strconv"
 "time"

 "github.com/hyperledger/fabric/core/chaincode/shim"
 sc "github.com/hyperledger/fabric/protos/peer"
)

type SmartContract struct {
}

type Message struct {
 PHO       string `json:"PHO"`
 SSCC      string `json:"SSCC"`
 Originator string `json:"Originator"`
 Recipient  string `json:"Recipient"`
 From      string `json:"From"`
 To        string `json:"To"`
 Latitude   string `json:"Latitude"`
 Longitude  string `json:"Longitude"`
}

func (s *SmartContract) Init(APIstub shim.ChaincodeStubInterface) sc.Response {
 return shim.Success(nil)
}

func (s *SmartContract) Invoke(APIstub shim.ChaincodeStubInterface) sc.Response {

 function, args := APIstub.GetFunctionAndParameters()

 if function == "queryMessage" {
  return s.queryMessage(APIstub, args)
 } else if function == "initLedger" {
  return s.initLedger(APIstub)
 } else if function == "createMessage" {
  return s.createMessage(APIstub, args)
 } else if function == "queryAllMessages" {
  return s.queryAllMessages(APIstub)
 } else if function == "getTransactionWithKey" {
  return s.getTransactionWithKey(APIstub, args)
 }

 return shim.Error("Invalid Smart Contract function name.")
}

func (s *SmartContract) queryMessage(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {

 if len(args) != 1 {
  return shim.Error("Incorrect number of arguments. Expecting 1")
 }

 MessageAsBytes, _ := APIstub.GetState(args[0])
 return shim.Success(MessageAsBytes)
}

func (s *SmartContract) initLedger(APIstub shim.ChaincodeStubInterface) sc.Response {
 Messages := []Message{
  Message{PHO: "1", SSCC: "1", Originator: "1", Recipient: "1", From: "1", To: "1", Latitude: "1", Longitude: "1"},
  Message{PHO: "2", SSCC: "2", Originator: "2", Recipient: "2", From: "2", To: "2", Latitude: "2", Longitude: "2"},
 }
```

```
 i := 0
 for i < len(Messages) {
  fmt.Println("i is ", i)
  MessageAsBytes, _ := json.Marshal(Messages[i])
  APIstub.PutState("Message"+strconv.Itoa(i), MessageAsBytes)
  fmt.Println("Added", Messages[i])
  i = i + 1
 }

 return shim.Success(nil)
}

func (s *SmartContract) createMessage(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {

 if len(args) != 9 {
  return shim.Error("Incorrect number of arguments. Expecting 9")
 }

 var Message = Message{PHO: args[1], SSCC: args[2], Originator: args[3], Recipient: args[4], From: args[5], To: args[6],
Latitude: args[7], Longitude: args[8]}

 MessageAsBytes, _ := json.Marshal(Message)
 APIstub.PutState(args[0], MessageAsBytes)

 return shim.Success(nil)
}

func (s *SmartContract) queryAllMessages(APIstub shim.ChaincodeStubInterface) sc.Response {

 startKey := "000000000"
 endKey := "999999999"

 resultsIterator, err := APIstub.GetStateByRange(startKey, endKey)
 if err != nil {
  return shim.Error(err.Error())
 }
 defer resultsIterator.Close()

 var buffer bytes.Buffer
 buffer.WriteString("[")

 bArrayMemberAlreadyWritten := false
 for resultsIterator.HasNext() {
  queryResponse, err := resultsIterator.Next()
  if err != nil {
   return shim.Error(err.Error())
  }

  if bArrayMemberAlreadyWritten == true {
   buffer.WriteString(",")
  }
  buffer.WriteString("{\"Key\":")
  buffer.WriteString("\"")
  buffer.WriteString(queryResponse.Key)
  buffer.WriteString("\"")

  buffer.WriteString(", \"Record\":")
  buffer.WriteString(string(queryResponse.Value))
  buffer.WriteString("}")
  buffer.WriteString("\n")
  bArrayMemberAlreadyWritten = true
 }
 buffer.WriteString("]")

 fmt.Printf("- queryAllMessages:\n%s\n", buffer.String())
```

```
 return shim.Success(buffer.Bytes())
}

func (t *SmartContract) getTransactionWithKey(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {

 if len(args) < 1 {
  return shim.Error("Incorrect number of arguments. Expecting 1")
 }

 ufoId := args[0]
 resultsIterator, err := APIstub.GetHistoryForKey(ufoId)
 if err != nil {
  return shim.Error(err.Error())
 }
 defer resultsIterator.Close()

 var buffer bytes.Buffer
 buffer.WriteString("[")

 bArrayMemberAlreadyWritten := false
 for resultsIterator.HasNext() {
  response, err := resultsIterator.Next()
  if err != nil {
   return shim.Error(err.Error())
  }
  if bArrayMemberAlreadyWritten == true {
   buffer.WriteString(",")
  }
  buffer.WriteString("{\"TxId\":")
  buffer.WriteString("\"")
  buffer.WriteString(response.TxId)
  buffer.WriteString("\"")

  buffer.WriteString(", \"Value\":")
  if response.IsDelete {
   buffer.WriteString("null")
  } else {
   buffer.WriteString(string(response.Value))
  }

  buffer.WriteString(", \"Timestamp\":")
  buffer.WriteString("\"")
  buffer.WriteString(time.Unix(response.Timestamp.Seconds, int64(response.Timestamp.Nanos)).String())
  buffer.WriteString("\"")

  buffer.WriteString(", \"IsDelete\":")
  buffer.WriteString("\"")
  buffer.WriteString(strconv.FormatBool(response.IsDelete))
  buffer.WriteString("\"")

  buffer.WriteString("}")
  bArrayMemberAlreadyWritten = true
 }
 buffer.WriteString("]")

 fmt.Printf("- History returning:\n%s\n", buffer.String())
 return shim.Success(buffer.Bytes())
}

func main() {

 err := shim.Start(new(SmartContract))
 if err != nil {
  fmt.Printf("Error creating new Smart Contract: %s", err)
 }
}
```

```java
package delivery;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Iterator;
import org.example.chaincode.invocation.InvokeChaincode;
import org.example.chaincode.invocation.QueryChaincode;
import crypto.Crypto;
import db.DAO;
import qr.ReadQR;
import qr.WindowQR;
@SuppressWarnings("all")
public class AppDelivery {

 public static void main(String[] args) throws Exception
 {

 System.out.println("WELCOME\n");
      System.out.println("Introduce your GCP:\n");
 int myGCP;
 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
 myGCP = Integer.parseInt(br.readLine());
 ArrayList<Handover> deloutgoing = DAO.myOutgoingHandovers(myGCP);
 ArrayList<Handover> delincoming = DAO.myIncomingHandovers(myGCP);
 Entity me = new Entity();
 me.setGCP(myGCP);
 DAO.updateUser(me);


 if(deloutgoing.isEmpty() && delincoming.isEmpty())
 {
  System.out.println("You have no deliveries planned today");
 }

 else
 {
  int option;
  System.out.println("\n\nHANDOVERS MANAGEMENT\n");
     System.out.println("Select One Option:\n");
     System.out.println("1. Show my Handovers");
     System.out.println("2. Create QR For New Delivery");
     System.out.println("3. New Handover Message");
     System.out.println("4. Scan QR for Incoming Handover and Send Message");
     System.out.println("5. Search for Message");
     System.out.println("6. Show All BlockChain Messages");
     %FUNCTIONS 7 & 8 FOR FUTURE DEVELOPMENT
     System.out.println("7. Calculate Average Price");
     System.out.println("8. Calculate Maximum Price");
     System.out.println("9. Exit");

  BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));
  option = Integer.parseInt(br1.readLine());

  switch (option) {
   case 1:

    System.out.println("INCOMING HANDOVERS\n");
    System.out.println(delincoming.toString());
    System.out.println("\nOUTGOING HANDOVERS\n");
    System.out.println(deloutgoing.toString());
    break;

   case 2:
```

```
System.out.println("Introduce the SSCC: ");
BufferedReader br2 = new BufferedReader(new InputStreamReader(System.in));
long sscc = Long.parseLong(br2.readLine());
Delivery d = DAO.searchDelivery(sscc);
ArrayList<Handover> hs = DAO.searchHandovers(sscc);
d.setHandovers(hs);
WindowQR w = new WindowQR(d.toString(),1);
w.setVisible(true);
System.out.println(d.toString());
break;

case 3:
System.out.println("Introduce the PHO: ");
BufferedReader br3 = new BufferedReader(new InputStreamReader(System.in));
int outpho = Integer.parseInt(br3.readLine());
Iterator<Handover> it = deloutgoing.iterator();
Handover out = null;

while(it.hasNext())
{
 Handover aux = (Handover)it.next();
 if(aux.getPHO()==outpho)
 {
  out=aux;
 }
}


if(out==null)
{
 System.out.println("\nERROR, THERE IS NO HANDOVER WITH PHO: " + outpho);
}
else
{
 Message m =new Message(out.getPHO(),out.getSSCC(),me,out.getFrom(),out.getLatitude(),out.getLongitude());
 WindowQR wout = new WindowQR(m.createMessage(),2);
 wout.setVisible(true);
}
break;

case 4:

String messageread = ReadQR.readQR();

int i1 = messageread.indexOf('A');
int i2 = messageread.indexOf('Z');
String orig = messageread.substring(i1+1, i2);

Entity originator = (Entity)DAO.selectUser(Integer.parseInt(orig));

int i3 = messageread.indexOf('|');
String ms = messageread.substring(0,i3);
String msh = Crypto.toHexString(Crypto.getSHA(ms));
String hashtover = messageread.substring(i3+1);

int i4=messageread.indexOf(',');
String phoin = messageread.substring(0,i4);

int i5 = messageread.indexOf('N');
String prevnonce = messageread.substring(i5+1,i3);


Iterator<Handover> it1 = delincoming.iterator();

while(it1.hasNext())
{
```

```
  Handover aux1 = (Handover)it1.next();
  if(aux1.getPHO()==Integer.parseInt(phoin))
  {
   Boolean ver = Crypto.verify(msh, hashtover, originator.getPublicKey());
   if(ver.equals(true))
   {
    System.out.println("\nTHE MESSAGE HAS BEEN SUCCESSFULLY VERIFIED");
    System.out.println(aux1.toString());
    InvokeChaincode.InvChainCode(aux1, me.getPublicKey());

   }
   else
   {
    System.out.println("\nTHE MESSAGE CAN'T BE VERIFIED");

   }
  }
  else
  {
   System.out.println("\nTHERE ARE NO INCOMING HANDOVERS");

  }
 }
break;

case 5:

 System.out.println("Introduce the PHO of the Message to Query: ");
 BufferedReader br4 = new BufferedReader(new InputStreamReader(System.in));
 int phoquery = Integer.parseInt(br4.readLine());
 Handover h = DAO.searchHandover(phoquery);
 Entity destination = DAO.selectUser(h.getRecipient().getGCP());
 String[] res = {String.valueOf(phoquery)};
 try
 {
  String s = QueryChaincode.QueryMessage(res);

  int i6 = s.indexOf("ARG1");
  int i7 = s.indexOf("ARG11");
  String phoe = s.substring(i6+4, i7);
  //String phod = Crypto.decrypt(phoe,destination.getPrivateKey());

  int i8 = s.indexOf("ARG2");
  int i9 = s.indexOf("ARG22");
  String sscce = s.substring(i8+4, i9);
  String ssccd = Crypto.decrypt(sscce,destination.getPrivateKey());

  int i10 = s.indexOf("ARG3");
  int i11 = s.indexOf("ARG33");
  String orige = s.substring(i10+4, i11);
  String origd = Crypto.decrypt(orige,destination.getPrivateKey());

  int i12 = s.indexOf("ARG4");
  int i13 = s.indexOf("ARG44");
  String deste = s.substring(i12+4, i13);
  String destd = Crypto.decrypt(deste,destination.getPrivateKey());


  int i14 = s.indexOf("ARG5");
  int i15 = s.indexOf("ARG55");
  String frome = s.substring(i14+4, i15);
  String fromd = Crypto.decrypt(frome,destination.getPrivateKey());

  int i16 = s.indexOf("ARG6");
  int i17 = s.indexOf("ARG66");
  String toe = s.substring(i14+4, i15);
  String tod = Crypto.decrypt(toe,destination.getPrivateKey());
```

```
 int i18 = s.indexOf("ARG7");
 int i19 = s.indexOf("ARG77");
 String longe = s.substring(i18+4, i19);
 String longd = Crypto.decrypt(longe,destination.getPrivateKey());

 int i20 = s.indexOf("ARG8");
 int i21 = s.indexOf("ARG88");
 String late = s.substring(i20+4, i21);
 String lated = Crypto.decrypt(late,destination.getPrivateKey());

 System.out.println("\n");
 System.out.println("MESSAGE WITH PHO "+ phoquery + " FOUND.");
 System.out.println("\n");
 System.out.println("DECRYPTING MESSAGE...");
 System.out.println("\n");



 System.out.println("DECRYPTED MESSAGE:");
 System.out.println("\nPHO: " + phoe);
 System.out.println("SSCC: " + ssccd);
 System.out.println("Originator: " + origd);
 System.out.println("Recipient: " + destd);
 System.out.println("From: " + fromd);
 System.out.println("To: " + tod);
 System.out.println("Longitude: " + longd);
 System.out.println("Latitude: " + lated);

}
 catch(Exception e)
 {
  e.printStackTrace();
 }
 break;

case 6:
 String s = QueryChaincode.QueryAllMessages();
 System.out.println(s);

 break;
case 7:

 int i = 0;
 int avg = 0;
 String price = QueryChaincode.QueryAllMessages();

 do

 {
  int i22 = price.indexOf("ARG1");
  int i23 = price.indexOf("ARG11");
  String phoe = price.substring(i22+4, i23);
  int phoei = Integer.parseInt(phoe);
  avg += phoei;
  i++;
  price = price.substring(i23+4);
 }while(price.contains("ARG1"));
 avg = (Integer) avg/i;


 System.out.println("\nAverage Price: " + avg);

 break;

case 8:
```

```
    int max = 0;
    String price1 = QueryChaincode.QueryAllMessages();

    do

    {
     int i24 = price1.indexOf("ARG1");
     int i25 = price1.indexOf("ARG11");
     String phoe = price1.substring(i24+4, i25);
     int phoei = Integer.parseInt(phoe);
     if(phoei>max)
     {
      max = phoei;
     }
     price1 = price1.substring(i25+4);
    }while(price1.contains("ARG1"));


    System.out.println("\nMaximum Price: " + max);


    break;


   default:
    System.out.println("Invalid Option.");
   }
  }
 }
}
```