Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació

# BLOCKCHAIN-BASED DIGITAL DATA CERTIFICATION AND TRACEABILITY SYSTEM

Memòria Final

VÍCTOR BARROSO CORCHERO

TUTOR: LÉONARD JANER GARCÍA

2021-2022

## Abstract

The goal of this project is to develop a solution to reduce Social Engineering cyberattacks using the blockchain technology. To achieve this goal, different blockchain technologies are studied with which to develop the solution, also the advantages and disadvantages of using such technology. In the development process a Smart Contract, that is deployed in a private Ethereum blockchain, and a web application, which interacts with the Smart Contract,  are developed

## Resum

L'objectiu d'aquest projecte és desenvolupar una solució per reduir els ciberatacs d’enginyeria social amb tecnologia blockchain. Per assolir aquest objectiu s'estudien diferents tecnologies blockchain per desenvolupar la solució i els avantatges i inconvenients d'usar aquesta tecnologia. Durant el procés de desenvolupament es crea aplicació web que interactua amb un Smart Contract desenvolupat en una blockchain d'Ethereum privada a través d'una API.

## Resumen

El objetivo de este proyecto es desarrollar una solución para reducir los ciberataques de phishing con tecnología blockchain. Para lograr este objetivo se estudian diferentes tecnologías blockchain con las que desarrollar la solución y las ventajas e inconvenientes de usar tal tecnología. Durante el proceso de desarrollo se crea aplicación web que interactúa con un Smart Contract desarrollado en una blockchain de Ethereum privada a través de una API.

## Table of contents

# Index of figures

# Index of tables

# 1 Project object

Over the last few years cyberattacks have become more and more sophisticated and common, and that is the case of phishing attacks, they can reach a sophistication level where you would hardly know if the email, or document, you are reading is a phishing or if it is not. That makes phishing a dangerous risk with a high impact, even more if an organization team is not aware of these cyberattacks.

This project aims to create a blockchain-based application that allows people or organizations to sign digital files and store the signatures in the blockchain so the receiver can validate that the file they received was truly sent by the person who the sender says to be and trace the versions of that file.

Lastly, this project has been chosen because the result of it is a product that helps to avoid cyberattacks, also because the solution is being developed using the blockchain technology, which provides scalability and security to the final product.

# 2 Previous study

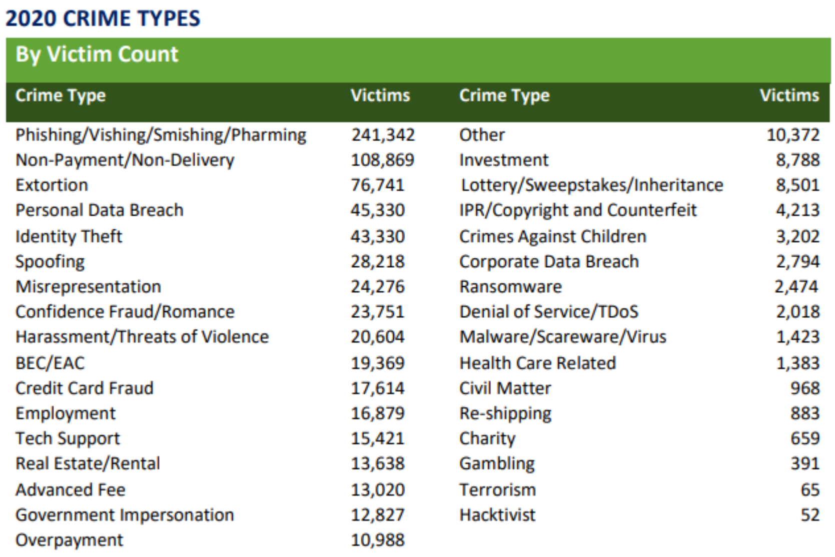
## 2.1 Context

Along the past decade cybercrime has risen up, and even more these past five years. The evolution of technology and its high availability for companies and people made more criminals interested in this world of cybercrime.

Due to COVID the cybercrime has grown exponentially since people work from home and cyber criminals are taking advantage of the vulnerabilities that the situation entails.

In agreement with Allot Ltd. [1], a high-tech company that manufactures telecommunications equipment, before the pandemic hit Europe, in January 2020 the 5% of blocked cyberthreats by them were phishing, by April phishing rose to 56% of all blocked threats, more than 1.100 million blocked cyberthreats [2].

According to the IC3 (Internet Crime Complaint Center) [3], phishing was the most common type of cyberattack in 2020 in USA and there were 241.342 complaints with adjusted losses of over 54$ million, as seen in the Figure 1 [4].



2020 CRIME TYPES

By Victim Count

| Crime Type | Victims | Crime Type | Victims |
|---|---|---|---|
| Phishing/Vishing/Smishing/Pharming | 241,342 | Other | 10,372 |
| Non-Payment/Non-Delivery | 108,869 | Investment | 8,788 |
| Extortion | 76,741 | Lottery/Sweepstakes/Inheritance | 8,501 |
| Personal Data Breach | 45,330 | IPR/Copyright and Counterfeit | 4,213 |
| Identity Theft | 43,330 | Crimes Against Children | 3,202 |
| Spoofing | 28,218 | Corporate Data Breach | 2,794 |
| Misrepresentation | 24,276 | Ransomware | 2,474 |
| Confidence Fraud/Romance | 23,751 | Denial of Service/TDoS | 2,018 |
| Harassment/Threats of Violence | 20,604 | Malware/Scareware/Virus | 1,423 |
| BEC/EAC | 19,369 | Health Care Related | 1,383 |
| Credit Card Fraud | 17,614 | Civil Matter | 968 |
| Employment | 16,879 | Re-shipping | 883 |
| Tech Support | 15,421 | Charity | 659 |
| Real Estate/Rental | 13,638 | Gambling | 391 |
| Advanced Fee | 13,020 | Terrorism | 65 |
| Government Impersonation | 12,827 | Hacktivist | 52 |
| Overpayment | 10,988 | | |

**Figure 1: Top crime types in USA in 2020. Source: 2020 Internet Crime Report, IC3**

On top of that compared to the top 5 crime types through the years, like is done in the Figure 2, it can be seen how phishing increased above all. It is true that in the first graph

IC3 groups by phishing, vishing, smishing and pharming and that might be the reason they count as the most common cybercrime, but in fact they are grouped because those cybercrimes use similar methods of social engineering[1] but different communication channels, we will now explain the differences between them:

- Phishing: The attacker sends a fraudulent message designed to trick a human victim into revealing sensitive information to the attacker or to deploy malicious software on the victim’s infrastructure.
- Vishing: Also called voice phishing, is the use of telephony to conduct phishing attacks. This method is typically used to steal sensitive personal information, mostly credit card numbers.
- Smishing: Also called SMS phishing, the attacker use cell phone text messages where they invite the victim to click on a link, call a phone number or contact and email address. Then the victim is asked to share their private data.
- Pharming: Cyberattack where the attacker redirects the traffic of the victim to a website controlled by the attacker. Then the attacker can steal private information of the victim or install malicious software.

[1] Psychological manipulation of people into performing actions or divulging confidential information.

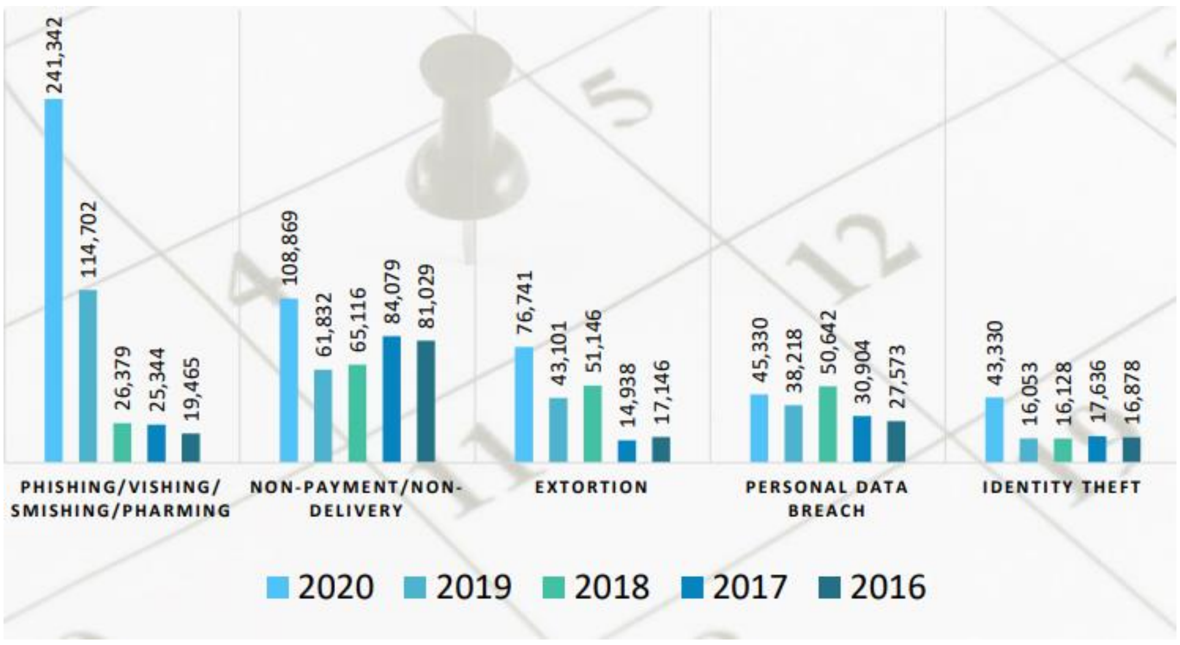


**Figure 2: Top 5 Crime Type Comparison. Source: 2020 Internet Crime Report, IC3**

Furthermore, as claimed by Europol [5] in their 2021 report of Internet Organised Crime Threat Assessment (IOCTA) [6], phishing and social engineering remain vectors for payment fraud, increasing in both volume and sophistication. Also, CEO Fraud[2] and Business email compromise[3] (BEC[4]) are the top threats in the area of non-cash payment fraud which can cause devastating losses to companies.

What all the cyberattacks mentioned earlier have in common is that they share the same entry point, the email.

Most of the time phishing can be easily detected, but as mentioned earlier it is becoming more and more sophisticated and just by clicking one URL or downloading any malicious file a person can compromise one whole company.

[2] Type of social engineering where the attacker targets a business to defraud the company. CEO Fraud is one of the many BEC scams.
[3] Type of social engineering in which cybercriminals spoof company email accounts and impersonate executives to try and fool an employee, usually someone from Human Resources or accounting, into executing unauthorized wire transfers, or sending out confidential tax information.
[4] Business Email Compromise

Phishing is a threat with high impact that can lead to other threats with major impact. Also, in 2021 phishing was the main vector[5], along with social engineering, for payment fraud in Europe [6].

Most of the people believes the veracity of a document by how it looks and cybersecurity software proves the document is harmless to the computer by opening the file and controlling its behavior and by looking for malicious code. But what happens if the document is an invoice sent by an organization or company email that looks 100% real and is completely harmless? The only way to verify its authenticity is to contact the sender, and companies cannot do that every time they send or receive an invoice, contract or any other kind of legal document[6].

One of the most significant problems that comes with phishing is the invoice fraud, which is a cyberattack where the attacker approaches a business pretending to represent a supplier, service provider or creditor and requesting the change of bank details for future invoices.

With all being said, this project aims to help companies to prevent these devastating cyberattacks which can cost up to hundreds of thousands of euros. The developed solution focuses on validating the author and integrity of a sent file. The target audience of the project are companies, but it is also be possible to use the solution for personal purposes.

In conclusion, validating the integrity and the author of any kind of file is a high responsibility job. In a company the employees who works with legal documents, such as accountants, can make the company lose up to hundreds of thousands of euros if they fall for a scam. To protect them from this kind of phishing the solution has to ensure that either no one can access the database of the application nor change the records, while bringing high security and availability, which can be easily achieved thanks to the blockchain technology, which we will talk about this later.

[5] Method or pathway used by a hacker to access or penetrate the target system.

[6] Document that states some contractual relationship or grants some rights. Some examples are: invoices, purchase agreements, employment contracts, etc.

## 2.2 Background

Nowadays there are possible solutions to the exposed problem, like e-signing platforms that helps users to sign and track versions of legal documents. This section exposes some of these solutions, their advantages and disadvantages and why a blockchain-based solution is better for solving the problem. Nevertheless, the developed solution focuses on validating any kind of file, not just legal documents.

### 2.2.1 E-Signing

E-signing, or electronic signature, is a personal electronic signature that can be used to sign documents and can have the same legal standing as a handwritten as long as it satisfies the requirements of the specific regulation under which it was created. The regulation used in the EU is called eIDAS [7], which stands for electronic Identification, Authentication and trust Services.

A digital signature that has met the requirements set forth under the eIDAS-regulation is called Advanced Electronic Signature (AdES) [8], the requirements are the following:

- The signatory can be uniquely identified and linked to the signature.
- The signatory must have the sole control of the private key used to create the electronic signature.
- If the data is tampered with after the message has been signed, the signature must identify that this has happened.
- Invalidating the signature in the event its accompanying data has changed.

For someone to use this solution they need to use an electronic signature software, and this comes with its advantages and disadvantages. The advantages are the following:

- Contracts are easily written, completed and signed by all interested parties in a little amount of time.
- Digitally signed documents can easily be tracked and located in a short amount of time.
- Signing an electronic document digitally identifies the signer as the signatory and that cannot be later denied.
- No one can forge the digital signature of someone or impersonate.

- Security: The use of digital signatures and electronic documents reduces risks of documents being intercepted, read, destroyed or altered while in transit.
- This method is cheap for signing legal documents.

On the other hand, there are some disadvantages which can complicate their use:

- The sender and the recipient may have to buy digital certificates at a cost from trusted certificate authorities[7] in order to use digital signatures.
- The sender and the recipient have to use verification software to work with digital certificates.
- Most of the different digital signature standards are incompatible between them.
- You delegate the security issue to a third-party.
- Some electronic signature vendors require the user to store their documents their servers.

To sum up, e-signing requires the sender and recipient to use an e-signing software in order to sign and validate files, furthermore they both need to use the same standard since not all of them are compatible between each other. Also, choosing this option means that the company has to delegate the security issue to a third-party, in addition some vendors make users store their documents in the servers of the vendor.

### 2.2.2 File encryption

File encryption consists in encrypting a file by using asymmetric cryptography, which is a system that uses pairs of keys that consists of a public key that can be known by others and a private key which should only be known by the owner. These keys are generated using cryptographic algorithms which are based on mathematical problems termed one-way functions, this means that they are operations that cannot be reversed.

[7] Certificate Authority issues an authenticated encryption for the digital certificate.

In this system the sender must know the public key of the receiver before sending any file or email. As shown in the Figure 3, Bob wants to send a message to Alice, so they use Alice's public key to encrypt it so Alice can decrypt the message with their private key.

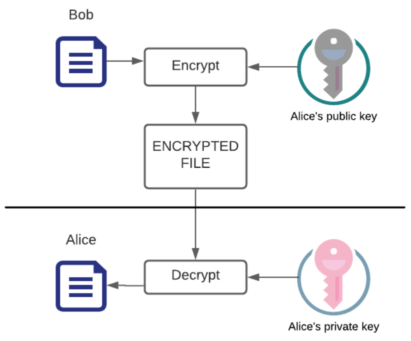


**Figure 3, Asymmetric cryptography. Source: Víctor Barroso Corchero**

Furthermore, for this solution to work on a large scale, like in a company, the sender and receiver should use an email client that lets them encrypt emails and files. Also, for verifying the sender the email should be signed with the sender's private key.

Email encryption can be achieved with different standards, the most common is OpenPGP [9], this method combines symmetric and asymmetric cryptography.

Symmetric-key algorithms are algorithms that use the same cryptographic keys for encrypting and decrypting. The keys are not identical, they represent a shared secret between two or more parties that can be used to maintain a private information link. In PGP encryption each public key is linked to a username or an e-mail address.

PGP encryption use a combination of hashing, data compression, symmetric-key cryptography and public-key cryptography, the Figure 4 shows how it works.

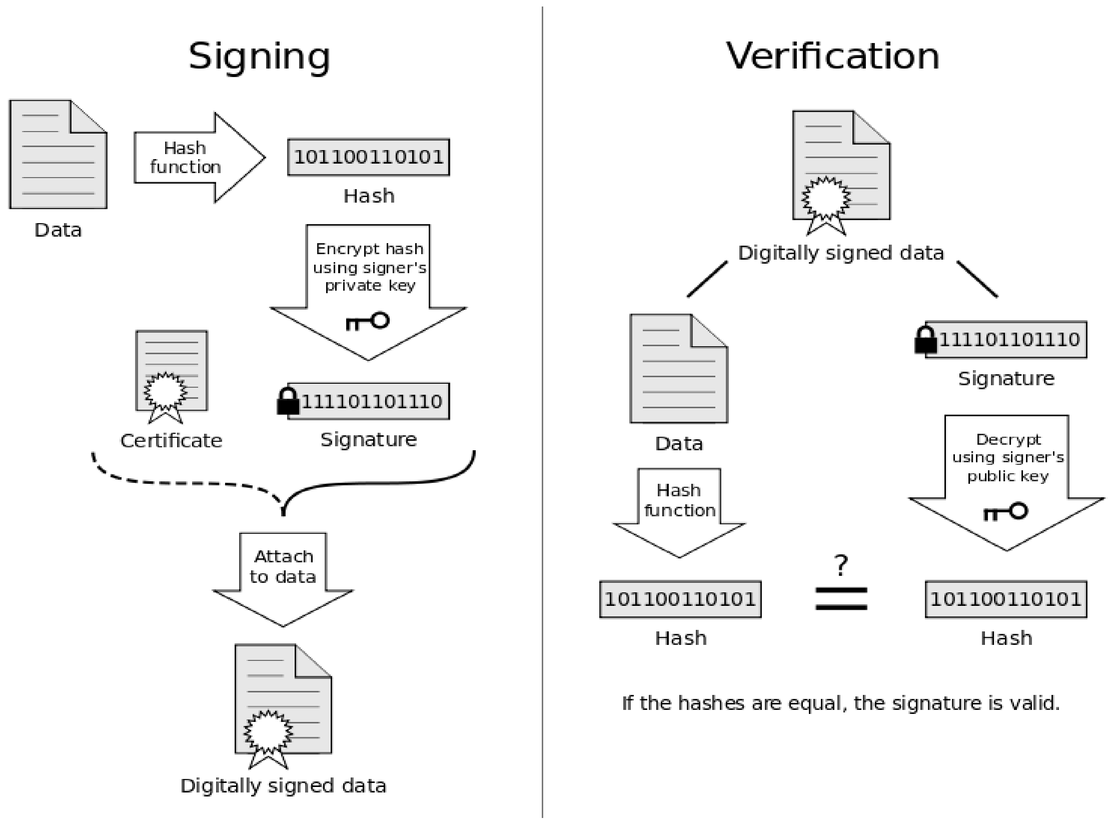


**Figure 4, How PGP encryption works visually. Source: blog.mailfence.com/es/buenas-practicas-de-firmado-digital-con-openpgp/**

Some advantages of using this method are:

- This method guarantees that the message has not been altered in transit.
- The receiver can verify the identity of the sender.
- The sender cannot deny sending a message. Only the owner of the private key can sign a message.
- This is a secure method.

And the disadvantages are:

- This method is not very user friendly. It gives an unnecessary complexity to file sharing inside the organization.
- Like e-signing, this method requires that both, the sender and receiver, can sign, encrypt and decrypt emails using an email client or software.

To conclude, email encryption can be done using different algorithms and standards, the most common is OpenPGP, also although this method brings security, non-repudiation, and authenticity it has a huge problem, the fact that this method is not user friendly, file sharing would be complex and it requires a software that can encrypt, decrypt and sign emails and files.

## 2.3 Blockchain

A blockchain is a distributed database of records or public ledger of all transaction or digital events that have been executed and shared among participating parties. Each transaction in the ledger is verified by the consensus mechanism of the blockchain and can never be deleted. The blockchain contains a certain and verifiable record of every single transaction ever made. The fact that the blockchain establishes a system of distributed consensus allows participants to know for certain that an event happened by creating an irrefutable record in a public ledger. [10]

Before going any further, there are some concepts that need to be explained. First the word blockchain is composed of two words, “block” which refers to where the data is stored and “chain” which refers to the fact that each block cryptographically references its parent.

In the Figure 5 we have a representation of what a blockchain is. The fields are:

- Block: The number of the block.
- Nonce: Random number used to calculate the hash. This is done since some blockchains requires the hash to follow a pattern, like in this image where the first three numbers of the hash are 0.
- Data: The data that is being added to the blockchain, it could be a group of transactions. The stored data is protected by the blockchain.
- Previous block hash: The first block of a blockchain, also called genesis block, has a hash full of zeros. The next blocks have the hash of their parent, this is how blocks are connected to each other.
- Hash: The hash of the information of the block, it includes all the fields mentioned above. If anything from this block changes the hash of the current and the next block changes, meaning that if someone wanted to change any block from the blockchain they should compute the hash of the block they changed and the consecutive ones, including the blocks that are being added in the meantime.

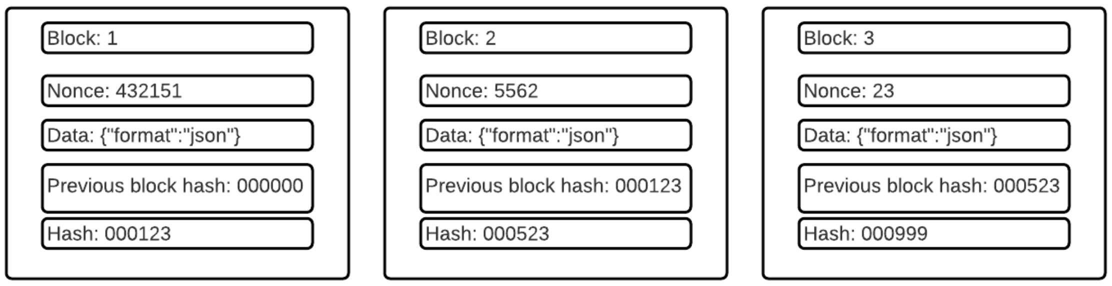


**Figure 5: Blockchain representation. Source: Víctor Barroso Corchero**

### 2.3.1 Versions of blockchain

The blockchain was first proposed as a research project in 1991 [11] and its first application in use was Bitcoin in 2009, which used the Distributed Ledger Technology (DLT), a technology that is explained later.

As described in Geeks for Geeks [12], a computer science portal, blockchain has three versions [13], each of one introducing a new revolutionary feature.

The blockchain 1.0 used distributed ledger technology and its first application was based on cryptocurrencies. This allowed Financial Transaction based on blockchain technology. As mentioned earlier, the first application was Bitcoin.

Moreover, the version 1.0 had problems of Scalability of Network, which refers to the capability of the blockchain to handle growing amount of work, also the mining of Bitcoin was wasteful, not environmentally friendly.

The mining of cryptocurrencies is the process of creating new coins while blocks of transactions are created. To create a block the miner has to solve a cryptographical puzzle and in Bitcoin involves a high computational power because the difficulty of that puzzle can vary depending on the amount of miners mining.

For a puzzle to be completed the miners have to guess by brute force the hash that follows a pattern. This is done by computing the hash of the block changing the Nonce field.

Blockchain 2.0 [13] introduced Smart Contracts[8] and solved the problem of scalability. Smart Contracts are digital contracts stored on a blockchain that are automatically executed when their terms and conditions are met. An advantage of this technology is that the since the blockchain is nearly impossible to hack it protects the integrity of those contracts. On top of that, Smart Contracts reduce the cost of verification and allow transparency contract definition.

The last version of blockchain is the 3.0, which introduced Decentralized Apps [14], mostly known as DApps[9]. A DApp is like a conventional application, the frontend can be done in any programming language and can make calls to its backend, which is running on decentralized Peer-To-Peer Network for using decentralized storage and communication.

### 2.3.2 Centralized, decentralized and distributed system

In the blockchain world terms like decentralized, distributed and centralized systems are mentioned very often. This section explains the differences between them, the advantages and disadvantages.

First of all, centralization and decentralization refer to levels of control. In a centralized system is one entity who has the control whereas a decentralized system shares the control among a number of independent entities.

On the other hand, distribution refers to differences of location. In a distributed system parts of the system are located in separate location, while in a non-distributed system all the parts of the system are in the same physical location.

In addition, as shown in Figure 6 centralized systems are supported by one node and if that single node fails the entire system falls down making impossible to continue with the

[8] Self-executing program that are stored on a blockchain that run when predetermined conditions are met. Smart contracts are more often used to automate the execution of an agreement made between two or more participants without the involvement of any intermediary.

[9] Decentralized App

operations where the system is needed. On the contrary, decentralized and distributed systems have more than one node making it more reliable, also when a node fails the entire system can keep working.

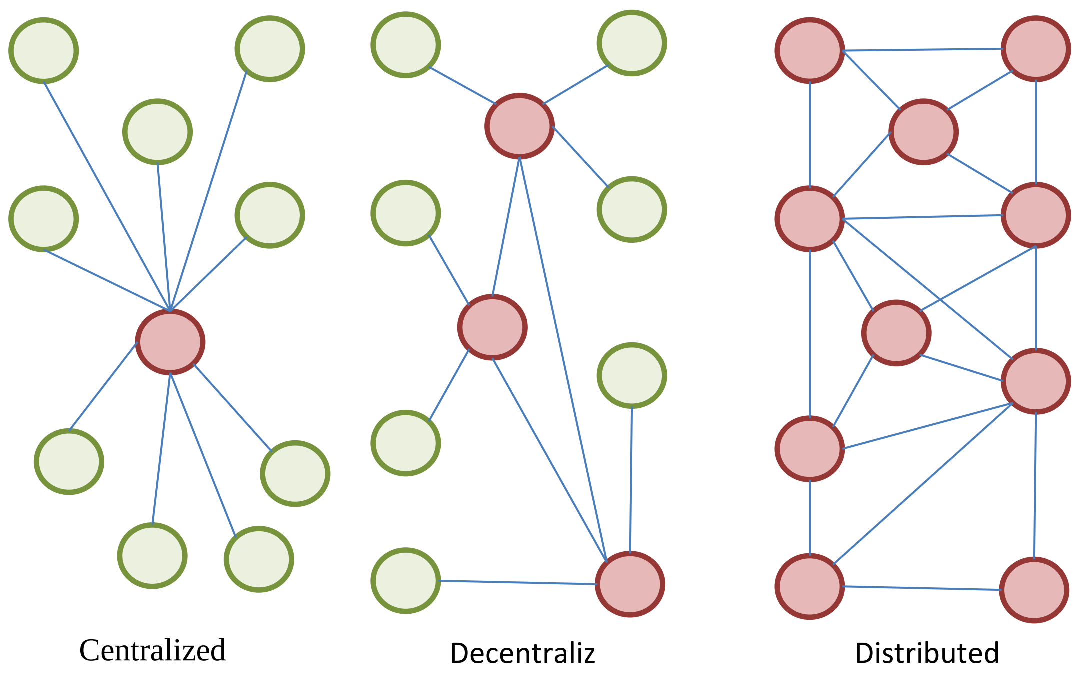


**Figure 6: Centralized, Decentralized and Distributed systems representation. Source: Víctor Barroso Corchero**

## 2.4 State of art

In order to make this project and to chose the best blockchain for developing DApps they need to be investigated to know what features brings each one of them, the programming language they use and the limitations. The solution will be a DApp.

### 2.4.1 Popular blockchains for developing DApps

This point exposes the current most popular blockchains for developing DApps, which consensus algorithm they use, what are their advantages and disadvantages and their environmental impact.

#### *2.4.1.1 Ethereum*

Ethereum [15] is the blockchain that introduced smart contracts and it is one of the most popular blockchains for developing smart contracts and DApps since then. Ethereum provides their users with:

- **Solidity,** a JavaScript-like language used for developing smart contracts.
- **EVM (Ethereum Virtual Machine),** which is the software platform used to create DApps on Ethereum.

Ethereum is planning a major update for 2022 [16], which includes the change of the consensus algorithm from Proof of Work to Proof of Stake. Ethereum is doing so because they want to be more compromised with the environment and to upgrade their features. Also, even though this is the most popular blockchain framework, nowadays there are better solutions for developing DApps. The reasons are the following:

- Ethereum handles low Transactions per Second[10] (TPS). The average is around 14 TPS and some blockchains we will see further can handle thousands. The reason Ethereum handles low transactions has to do because of the consensus algorithm used.
- The actual consensus mechanism Ethereum uses is Proof of Work, which as mentioned earlier in the report is dangerous for the environment and slows down their TPS.
- High transaction fees[11]. The average transaction fee is 14€ (February 1st, 2022)

To sum up, Ethereum has great tools for developing Smart Contracts and DApps, but its negatives points are stronger than its positive. Also, right now Ethereum is harmful to the environment.

[10] The number of transactions a blockchain can handle per second. This number determines the speed of a blockchain and may vary through time.
[11] The cost of making a transaction in a blockchain, the cost may vary through time due to multiple factors like the amount of transactions that needs to be verified.

#### *2.4.1.2 Solana*

Solana [17] claims to be the fastest blockchain in the world, at the time of today (February 2nd, 2022) it has an average of 4.780TPS also, as Anatoly Yakovenko, the CEO of Solana, claims in their whitepaper[12], Solana can get up to 710.000 TPS [18]. Moreover, Solana introduced a new consensus mechanism, the Proof of History.

In the blockchain Solana users can develop Programs, which are Smart Contracts in Ethereum, and DApps [19]. The development of Smart Contracts is done in the programming languages Rust, C and C++. Furthermore, the base of app development on Solana is the JSON RPC API, which is a layer of communication that allows developers to interact with the blockchain.

One of the main advantages of Solana, apart from its high TPS, is the cost of its transaction fee, which is 0,00022€, a really low number.

To sum up, Solana introduced the Proof of History, a consensus mechanism that speeds up transactions, also, the Solana network handles and average of 4.780 TPS being able to scale and handle up to 710.000 TPS. Also, the Smart Contracts are programmed in Rust, C and C++ and apps use JSON RPC API for communicating with the blockchain.

#### *2.4.1.3 EOSIO*

As mentioned in a Cryptopedia article [20], EOSIO [21] uses its own consensus mechanism invented by the CTO of Block.one [22] Dan Larimer, which is DPoS[13], a variation of Proof of Stake, with a Byzantine Fault Tolerance[14] model. EOS blockchain nowadays handles 4000 TPS.

[12] Whitepapers are documents that explain the purpose and technology behind a project.
[13] Delegated Proof of Stake
[14] Resiliency of a fault-tolerant computing system, particularly distributed computing systems.

In addition to this, EOSIO provides the EOSIO Contract Development Toolkit, a ToolChain[15] built to compile and optimize EOSIO smart contracts, also the standard libraries they provide are for both C and C++ programming languages.

Furthermore, this blockchain also provides a Command Line Interface[16] (CLI), which is called Cleos, with this CLI developers can access to specific developer tools for reading data from the blockchain history, sending new transactions, and to test and deploy smart contracts. Also EOSIO does not have transactions fees.

[15] Set of software programming tools designed to simplify complex software development taks.

[16] Command Line Interfaces connects a user to a computer program or operating system by typing in commands in text format.

# 3 Objectives and scope

## 3.1 Project objectives

The project objectives are:

- To develop a web application that can generate a digital signature of a file and store it into the blockchain. Also, the solution should be able to validate who the owner of a file is just by its digital signature.
- To develop a web application that can is supported by the blockchain technology.
- To add a layer of protection to companies by offering them a solution that can protect them from cyberattacks like phishing.

## 3.2 Product objectives

The product aspires to help companies to ensure the documents they receive are not fake, and its objectives are:

- To offer a method to validate files without having to trust on a third party.
- To offer a method to keep track of a file’s changes by its digital signature.
- To validate a file’s owner by its digital signature.
- To ensure the privacy, integrity and immutability of the data.
- To have a user-friendly interface.

## 3.3 Customer objectives

This project is developed with a customer, ESED [23] a cybersecurity company, and their objectives are:

- To offer a solution to companies for validating digital files using the blockchain technology.
- To offer a solution for complimenting cybersecurity inside companies for detecting cyberattacks, such as CEO fraud, which we mentioned earlier.

# 4 Methodology

The blockchain technology is a recent one and is booming, that means that technical documentation is not abundant, but on the other hand, the community around the blockchain technology grows fast and is active. Due to the fact that investors are very interested on cryptocurrencies there are a lot of web pages where they explain how blockchains work and the meaning of technical words.

Besides that, information about cybercrime can be found in reports of legal institutions like Interpol or FBI, also in reports of IT companies specialized in cybersecurity.

The development methodology used for this project is Waterfall [24]. This methodology is a rigid linear model that consists of sequential phases. Each phase starts when the previous one is completed. Furthermore, as shown in the Figure 7, we can see the phases this project has:

1. Requirements: Define functional and non-functional requirements.
2. Research: Find information about the chosen technology, make a previous study of the project field.
3. Design: Create the BPMN 2.0[17] of the solution processes. From the processes the tasks needed to develop the solution can be extracted.
4. Development & testing: In Waterfall these two tasks are usually separated, first comes the development and then the testing, in this project the testing and development are done at the same time, doing this benefits the project execution since errors are detected earlier.
5. Deployment: Once the development and testing are done the solution is deployed to production phase, where it is available for the public use.
6. Documentation: In this step we document everything we have done in this project.
7. Maintenance: This step involves the future steps of the project.

[17] The acronym stands for Business Process Model and Notation, which is a graphical representation for specifying business processes in a business process model.

Furthermore, the academical tutor and the project manager are meeting every two weeks since the start of the project until the end of the same. The communication channels used are e-mail and WhatsApp [25], also before each meeting the project manager sends the meeting minutes with the topics that are being discussed, the project manager sends the meeting minutes with all the discussed things after the meeting is done as well.

The project manager follows the same methodology with the technical tutor, the meetings are done every two weeks too. The communication is done via Google Chat [26].

Lastly, the tool used for organizing the project is ClickUp [27], an application made for managing people and projects, the application also has interesting features such as file sharing and screen recorder.

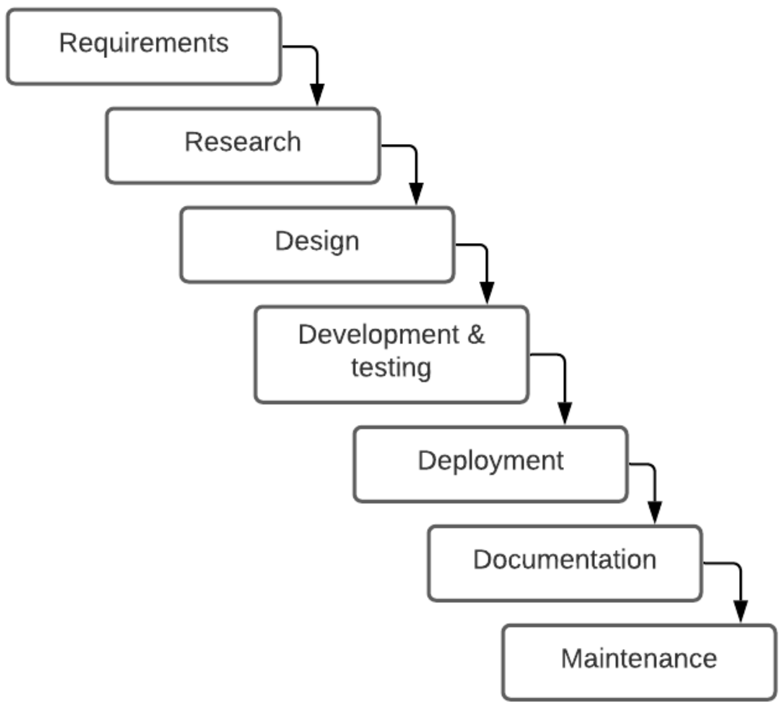


**Figure 7: Waterfall Methodology Life Cycle. Source: Víctor Barroso Corchero**

# 5 Development

## 5.1 Definition of functional and technological requirements

### 5.1.1 Functional requirements

The functional requirements of the developed solution:

- To make transactions into a blockchain.
- To generate a digital signature of a file.
- To add metadata to the file. The metadata[18] information should include:
  - File Name: The name of the file.
  - Owner: The account which makes the transaction.
  - Company name: The name of the company
  - Transaction Date: The date when the transaction is made.
  - Expiry Date: This field can be null.
  - Version: Number of the version.
  - UID: Unique identifier.
  - Hash: The signature of the file.
- To modify the metadata of a file. All the fields but the UID can be modified.
- To have a version control of the metadata of the file where:
  - If the user is consulting an older version of a file the application should tell them.
  - If the user is consulting a deprecated document the application should tell them.
- To let authors to revoke an uploaded file signature.
- To be able to extract information about a file by a file signature.
- To contain a user-friendly interface.
- To be as transparent as possible to the end-user.

[18] Data that provides information about other data.

### 5.1.2 Technological requirements

What is need for developing the application is:

- Cloud server for hosting a web application. The minimum specs for the web server are:
    - RAM: 4GB DDR4
    - Processor: 2 x 1,6 GHz CPU
    - Storage:  40GB HDD

  The price of a server with these specs in Amazon Web Services [28] (AWS), subsidiary of Amazon providing on-demand cloud computing platforms and APIs, is 0,052€/h. The development starts on March 3rd and the web server should be up until July 16th, the total price for the development time is 1,248€/day * 136 days = 169,728€.
- Computer for developing the application, it has the next specs:
    - RAM: 16 GB DDR4
    - Storage: 256 GB SSD
    - Processor:
        - Cores: 4
        - Logical processors: 4
        - Sockets: 1
        - Speed: 4 GHz
- Peripheral and their needed specs, in case of needing them:
    - Screen:
        - Resolution: 1980x1080 Full HD
        - Refreshment rate: 60HZs
- Blockchain platform:
    - Technology: Ethereum
    - Minimum number of nodes: 2
    - Node memory: 1GB

To sum up, the Table 1 shows the costs of the platform for the development.

| Technological requirements | | | | |
|---|---|---|---|---|
| Technological requirement | Calculation | Price (€) | Amortization (€) | Cost (€) |
| Cloud server | 1,248€/day*136days | 169,728 | - | 169,728 |
| Computer | https://www.pccomponentes.com/lenovo-ideapad-5-14itl05-intel-core-i7-1165g7-16-gb-512gb-ssd-14 | 849 | 212 | 159 |
| Screen | - | 100 | 25 | 19 |
| Blockchain Platform | Free trial | 0 | 0 | 0 |
| Total | | | | 347,67 € |

**Table 1: Summary of technological costs. Source: Víctor Barroso Corchero**

## 5.2 Technology comparison

To begin with, the chosen technology with which the project is developed has been chosen by making a comparison, as shown in the **Table 2,** between different blockchains, it is also shown a comparison between public and private blockchain.

| Solution | Positive aspects | Negative aspects |
|---|---|---|
| Blockchain as a Service[19](BaaS) – Kaleido [29] | - Customizable blockchain.<br>- Plug-and-play technology.<br>- It offers a platform to monitor your network health.<br>- Control all over the blockchain.<br>- Maintenance cost can be predicted. | - Maintenance cost can get high in production, minimum of 0,49€/h per node. A blockchain needs a minimum of 2 nodes to operate that means 0,98€/h which is 23,52€/day. Also, this is the price for the smallest node.<br>- You delegate the securities issues to Kaleido, a third party.<br>- Needs maintenance. You manage the blockchain. |
| Public blockchain | - Technological platform for storing the data is provided by the blockchain.<br>- High security.<br>- Transparency. | - For each transaction a user makes into the blockchain they have to pay a transaction fee, depending on the blockchain that fee can be more or less expensive.<br>- For each time you deploy a smart contract you have to pay.<br>- Hard to calculate the cost of maintaining the DApp, each blockchain have different costs. |
| Solana | - Transactions fees are really low: 0,00022€.<br>- Transaction fees doesn't change.<br>- A very high number of Transactions Per Second (TPS), the blockchain handles an average of 4.780 TPS.<br>- Founders claims to handle up to 710.000 TPS with their current nodes. | - Deploying a Smart Contract requires you to pay a fee that can be very high and is not stable.<br>- Smart contracts last for a period of time. |
| EOSIO | - Handles an average of 4000 TPS<br>- Free transaction fees | - The developer pays for the used resources of their Smart Contract, such as RAM, CPU and Network Bandwidth. The maintenance cost is hard to calculate. |

**Table 2: Comparison between technologies Src: Víctor Barroso Corchero**

[19] Third-party cloud-based infrastructure and management for companies building and operating blockchain apps [31]

#### *5.2.1.1 Conclusion*

To conclude with, all the options are viable, using a BaaS solution, like Kaleido, could be interesting since the developers can manage their own blockchain. In addition, this solution might be the most expensive in the long term. Also, for this option to scale the developers must spend more money to maintain more nodes. Moreover, in a BaaS solution the blockchain can be customized if needed.

On the other hand, I believe that using a public blockchain simplifies the maintenance of the DApp. Also, since all the stored data should be public it is not a matter of importance to worry about the privacy of the data.

Lastly, in a BaaS the developer pays for the blockchain itself, in the Kaleido platform users pay per hour and node, on the contrary, choosing a public blockchain means that the developer pays for each time they deploy a smart contract and/or for the resources you use, on the other hand public blockchains have devnets, which is a copy of the real blockchain but with fake tokens used for developing without any cost.

To sum up, I believe that the best option is to develop the solution using a BaaS, like Kaleido, the reasons are the following:

- In a BaaS solution you can calculate the cost of development and production, while the prices in a public blockchain can fluctuate and are impossible to calculate.
- In a BaaS solution you can deploy as many Smart Contracts as you want, in a public blockchain you have to be more careful since you pay for each deployment and/or for the resources used. Also, since we don’t have experience developing DApps it is expected to make mistakes when the DApp is in production.

## 5.3 Blockchain as a Service – Kaleido

As explained earlier this project is using a private blockchain created using the Kaleido platform. This platform simplifies the communication between apps and lets the developer customize the blockchain at their will, like changing the consensus mechanism, transaction fees and also, since the developer owns the blockchain, Smart Contract deployment is free.

The configured Blockchain in the Kaleido platform built for this project uses the technology of Ethereum, it has 3 nodes operating, which are the ones included in the Starter version of this product.

Kaleido also offers a way for developers to manage accounts, it lets developers synchronize their environment, which is the blockchain itself, with a Cloud-hosted Hardware Security Module service [20](Cloud HSM), for managing new accounts for new users.

Furthermore, as shown in the Figure 8, Kaleido has robust processes implemented to ensure the proper protection of critical key materials, secure virtual networking, strong authentication schemes, highly available runtime components, firewalls and logical isolation, encrypted transport, cross-cloud tunneling and disaster recovery.

The most important features of the Kaleido infrastructure shown in Figure 8 are the following:

- **Network layer**:
    - It contains a default DDoS protection.
    - Permits secure TLS connection.
    - Serves a CA-signed and validated certificate for identity and validation
- **Application layer**:
    - Strongly generated username and password to authenticate access to the ingress.
    - Sensitive information, such as passwords or username secrets, does not persist in plaintext, rather a salted hash [21]is kept and used for verification.
    - Communication uses secure protocols.
    - Verified calls are granted access to the isolated virtual network encapsulating the blockchain layer.
- **Environments and Nodes**:
    - Logical isolation per environment
    - Resources, like credentials, are unusable between environments since they are confined solely to their environment.
    - Option for secure integration with user-controlled native cloud services such as:
        - File system and encryption
        - Log streaming
        - File system backups
        - Private connections
- **Storage**:
    - Scalable file system.

[20] Service that allows you to host encryption keys and perform cryptographic operations.
[21] When random data is used as an additional input to a hash function that hashes a password

- Native cloud file systems come with default High Availability[22] safeguard built in.
- File systems are mounted and provisioned to each node ensuring data isolation in a shared virtual environment.

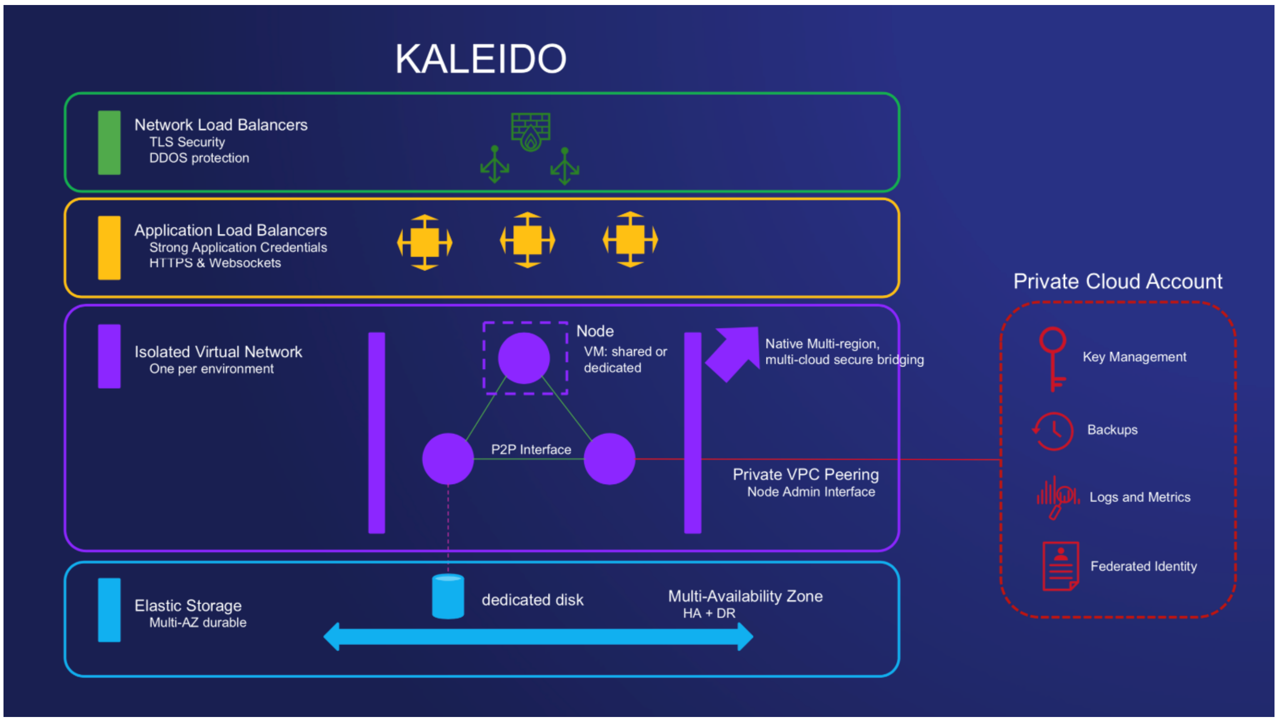


**Figure 8: Kaleido architecture Src: Kaleido (https://docs.kaleido.io/kaleido-platform/architecture-reference/)**

### 5.3.1 DApp architecture

The typical architecture of a DApp is like the one shown on the Figure 9**,** the user connects to the DApp via a server, but when interacting with the blockchain it uses a JavaScript library called web3.js [30] to interact with a local or remote Ethereum node using HTTP, IPC or WebSocket.

[22] Component of a technology system that eliminates single points of failure to ensure continuous operations or uptime for an extended period

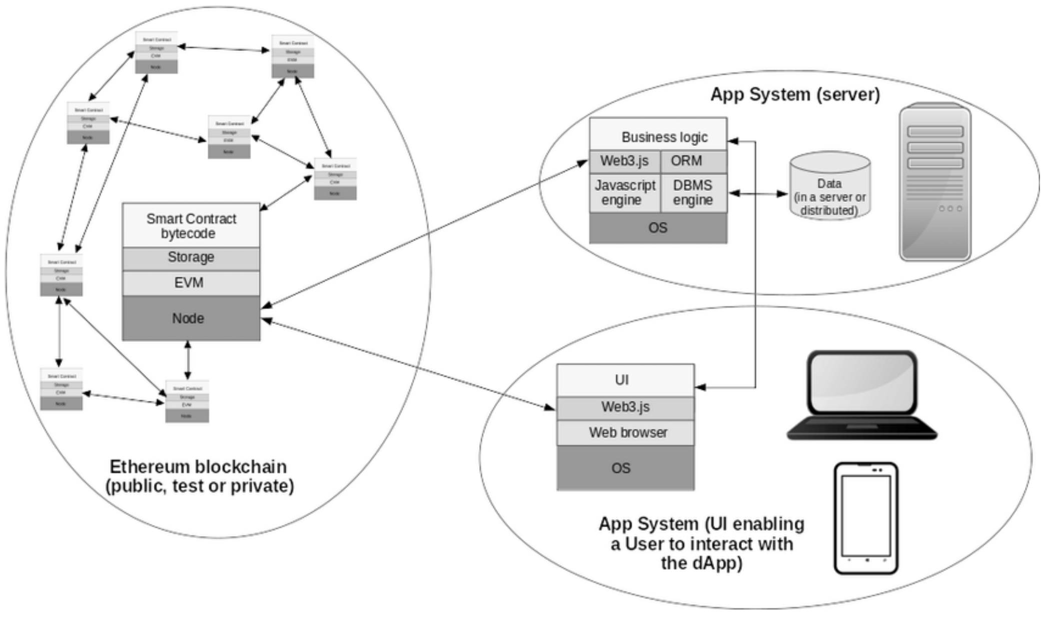

Smart Contract bytecode
Storage
EVM
Node
Ethereum blockchain (public, test or private)
App System (server)
Business logic
Web3.js
ORM
Javascript engine
DBMS engine
OS
Data (in a server or distributed)
UI
Web3.js
Web browser
OS
App System (UI enabling a User to interact with the dApp)

**Figure 9: Typical architecture of an Ethereum DApp Src: Lodovica Marchesi (https://www.researchgate.net/figure/A-typical-architecture-of-an-Ethereum-dApp-application-The-App-System-is-shown-on-the_fig2_338064306)**

On the other hand, with the private blockchain of Kaleido this scheme changes a bit, the scheme should stay the same, but instead of using Web3.js it is required to make an API call to the Kaleido platform, which afterwards makes the required operation on the blockchain.

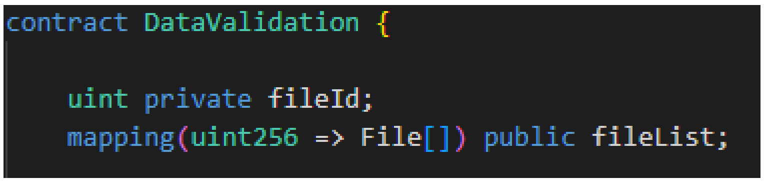
## 5.4 Smart Contract

To begin with, the Smart Contract has been developed in Solidity [23] and has been deployed into the Kaleido environment, which provides an API for executing the Smart Contract functions and make transactions into the blockchain.

The Smart Contract variables are the ones shown in the Figure 10:

- fileId: The variable keeps track of the created file objects and starts with a value of 0.
- fileList: The variable stores all the file objects with their versions.

[23] Object-oriented programming language for implementing smart contracts on various blockchains. Programs in Solidity run on Ethereum Virtual Machine (EVM)



```
contract DataValidation {

    uint private fileId;
    mapping(uint256 => File[]) public fileList;
```

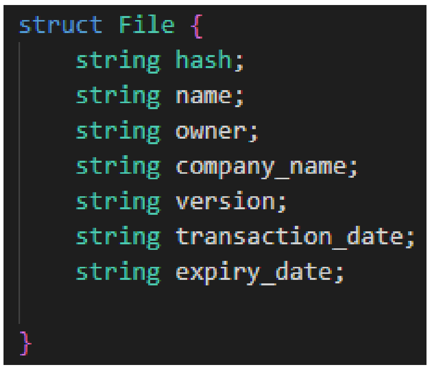
**Figure 10: Smart contract variables Src: Victor Barroso Corchero**

### 5.4.1 Methods

This section discusses the main methods of the Smart Contract mentioned earlier, DataValidation, and their functionalities.

The methods are the following:

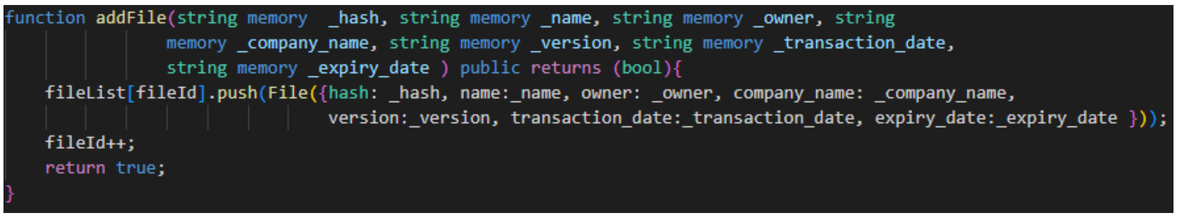
1. **Add a file:** The parameters of this function are the object variables of a File (Figure 11). As shown in the Figure 12, the function adds into the fileList variable a new File object in the position of the current value of fileId, then increases the variable value by one. Lastly, it returns a true meaning that the operation was done successfully.



```
struct File {
    string hash;
    string name;
    string owner;
    string company_name;
    string version;
    string transaction_date;
    string expiry_date;

}
```

**Figure 11: Smart contract File object Src: Victor Barroso Corchero**

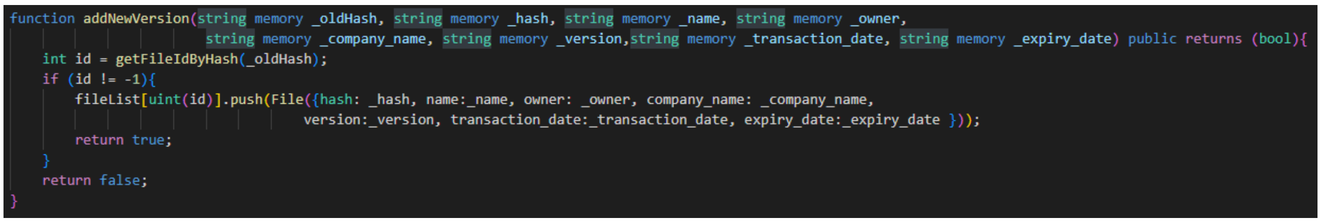


```
function addFile(string memory  _hash, string memory _name, string memory _owner, string
                memory _company_name, string memory _version, string memory _transaction_date,
                string memory _expiry_date ) public returns (bool){
    fileList[fileId].push(File({hash: _hash, name:_name, owner: _owner, company_name: _company_name,
                            version:_version, transaction_date:_transaction_date, expiry_date:_expiry_date }));
    fileId++;
    return true;
}
```

**Figure 12: Smart contract addFile function Src: Victor Barroso Corchero**

2. **Add a new version of a file:** With this function a new version of an old file is added. As shown in the Figure 13, the parameters of the function are the file object variables and the old hash, corresponding to the file the user wants to update. For finding the file it uses the function getFileIdByHash (Figure 14), if the function finds the id it

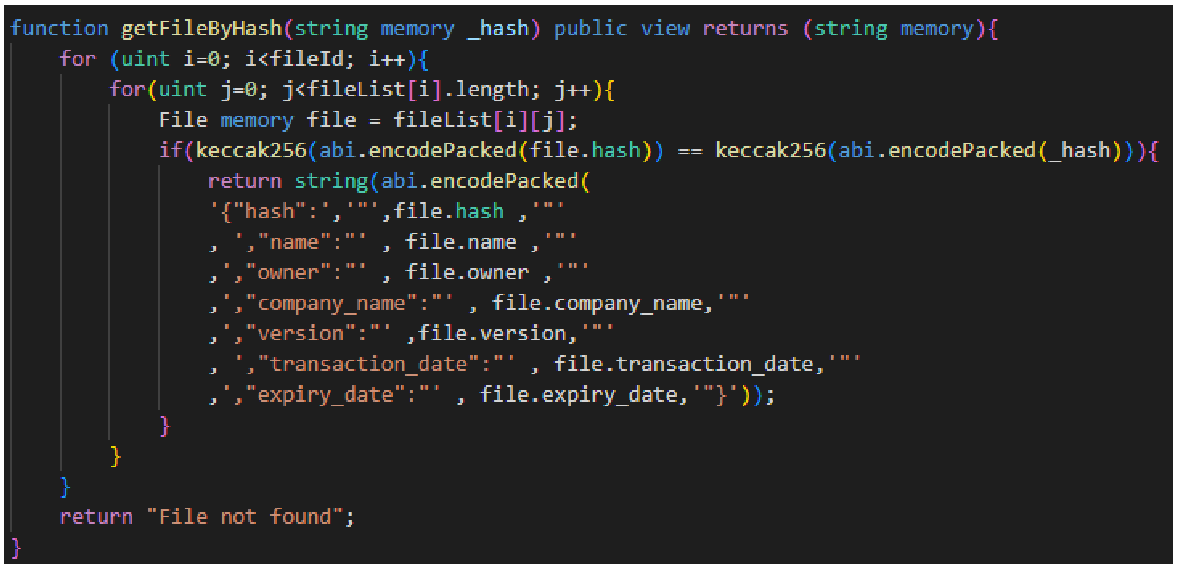
adds the file to the array of files with the same Id and returns a true, if not it returns a false and doesn't change anything.



```
function addNewVersion(string memory _oldHash, string memory _hash, string memory _name, string memory _owner,
                        string memory _company_name, string memory _version,string memory _transaction_date, string memory _expiry_date) public returns (bool){
    int id = getFileIdByHash(_oldHash);
    if (id != -1){
        fileList[uint(id)].push(File({hash: _hash, name:_name, owner: _owner, company_name: _company_name,
                                version:_version, transaction_date:_transaction_date, expiry_date:_expiry_date }));
        return true;
    }
    return false;
}
```

**Figure 13: Smart contract addNewVersion function Src: Victor Barroso Corchero**

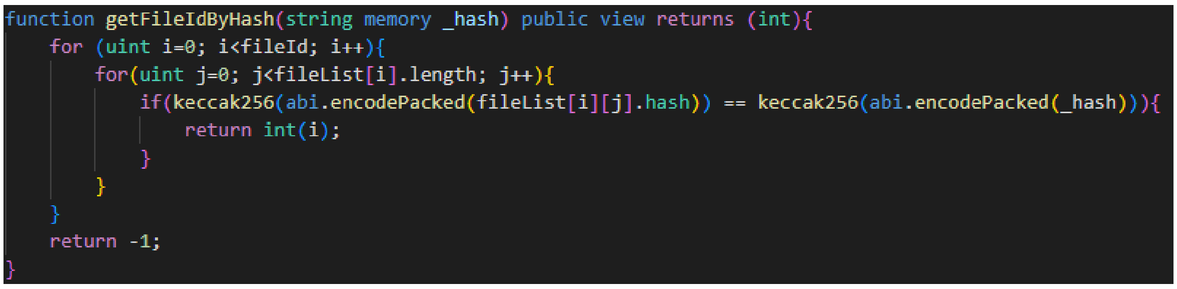
3. **Get a file by its hash:** This function retrieves the information of a file within the blockchain by its hash. As shown in the Figure 14, the code iterates the map and array of the files, for each file it compares the hash of the file the user wants to check, that it is given by the only parameter, with the hash of the file object. If the function finds the file it returns the object with a JSON format, otherwise it returns a message saying *File not found.*



```
function getFileByHash(string memory _hash) public view returns (string memory){
    for (uint i=0; i<fileId; i++){
        for(uint j=0; j<fileList[i].length; j++){
            File memory file = fileList[i][j];
            if(keccak256(abi.encodePacked(file.hash)) == keccak256(abi.encodePacked(_hash))){
                return string(abi.encodePacked(
                '{"hash":','"',file.hash ,'"'
                , ',"name":"' , file.name ,'"'
                ,',"owner":"' , file.owner ,'"'
                ,',"company_name":"' , file.company_name,'"'
                ,',"version":"' ,file.version,'"'
                , ',"transaction_date":"' , file.transaction_date,'"'
                ,',"expiry_date":"' , file.expiry_date,'"}'));
            }
        }
    }
    return "File not found";
}
```

**Figure 14: Smart contract getfileByHash function Src: Victor Barroso Corchero**

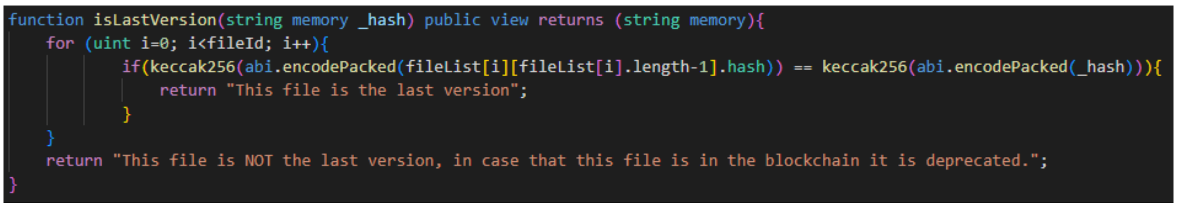
4. **Get the id of the file:** Given a hash this function returns the id of the file. As shown in the Figure 15, the function iterates through the map and array of files trying to match the hash given by the parameter and the hash of the current file. If the function finds a match, then it returns the id of the file, otherwise it returns a negative value.



```
function getFileIdByHash(string memory _hash) public view returns (int){
    for (uint i=0; i<fileId; i++){
        for(uint j=0; j<fileList[i].length; j++){
            if(keccak256(abi.encodePacked(fileList[i][j].hash)) == keccak256(abi.encodePacked(_hash))){
                return int(i);
            }
        }
    }
    return -1;
}
```

**Figure 15: Smart contract getfileIdByHash function Src: Victor Barroso Corchero**

5. **Is last version:** This function finds if the hash of the file is the corresponds to the last version. As shown in the Figure 16, this function iterates through the map and compares the last file added to each array with the one given as a parameter, if it finds a match it returns a message saying that the file is the last version, otherwise it returns a warning message meaning that if the file was uploaded into the blockchain, is not the last version.



```
function isLastVersion(string memory _hash) public view returns (string memory){
    for (uint i=0; i<fileId; i++){
            if(keccak256(abi.encodePacked(fileList[i][fileList[i].length-1].hash)) == keccak256(abi.encodePacked(_hash))){
                return "This file is the last version";
            }
    }
    return "This file is NOT the last version, in case that this file is in the blockchain it is deprecated.";
}
```

**Figure 16: Smart contract isLastVersion function Src: Victor Barroso Corchero**

### 5.4.2 API[24]

As mentioned earlier in the document, the connection between the Web Application and the blockchain will be made through the Kaleido API. This section discusses the most relevant parameters to take in mind when using the API.

The Kaleido platform generates two API methods for each function of the Smart Contract, a POST and a GET function. Each function requires some common parameters, apart from the ones that the functions of the Smart Contract use.

[24] Application programming interface (API). Set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service

These parameters are:

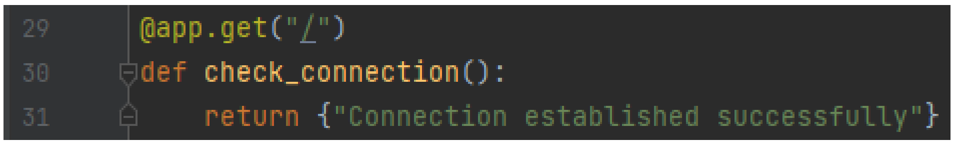
- **kId-from:** Required parameter, the address that makes the transaction.
- **kId-gasprice**: Optional parameter, gas price offered for the transaction.
- **kId-gas:** Gas to send with the transaction, this value is auto-calculated if sent empty.
- **kId-id:** Optional parameter, set an unique ID for this request.

### 5.4.3 Hash calculator

One of the parameters of all the Smart Contract functions is the hash of the file, which is calculated using a python API that only contains two functions. The API is developed with the python library FastAPI.

The methods are the following:

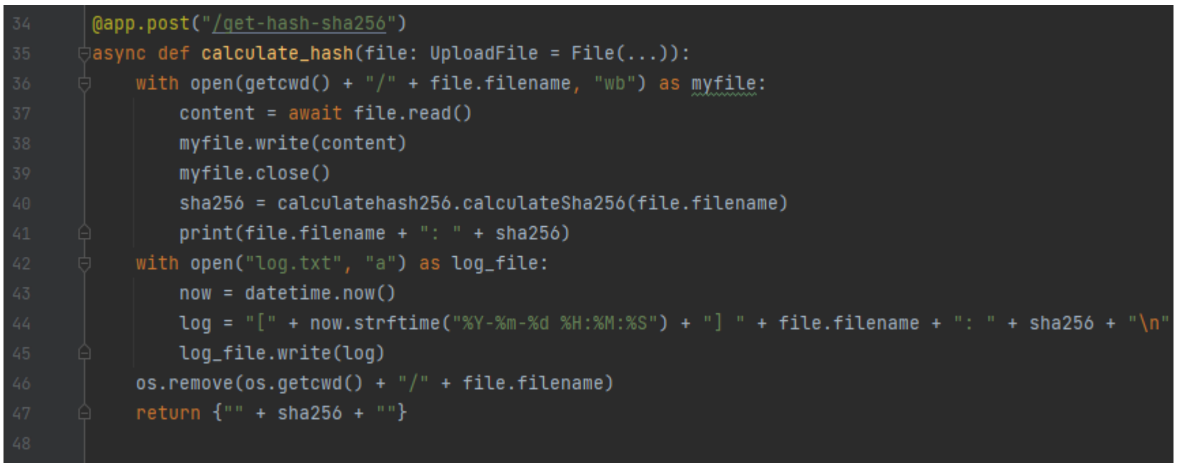
- GET("/") check_connection(): The purpose of this method is to check if the connection between the application is made correctly. Further, as shown in the Figure 17, the code only returns a string that notifies the user that the connection works.



```python
@app.get("/")
def check_connection():
    return {"Connection established successfully"}
```

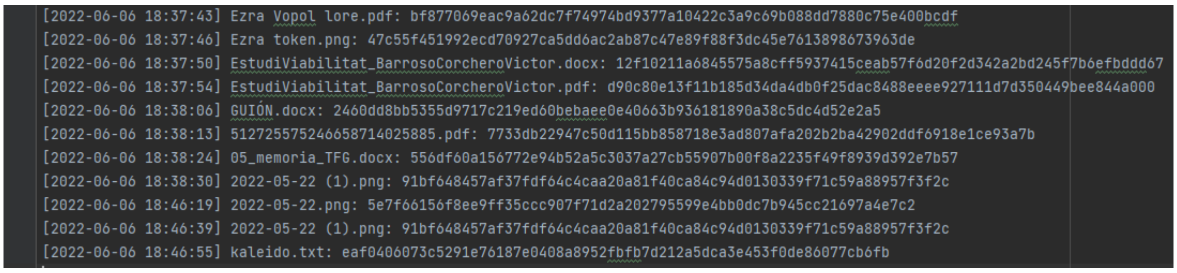
**Figure 17: Python API check connection function API Src: Victor Barroso Corchero**

- POST("/get-hash-sha256"): This method returns the hash of the file using the cryptographic hash function sha256.
  Moreover, in the Figure 18 the code can be seen, the API function requires a file as a parameter. First it stores the file temporarily which is later on sent as a parameter for the function calculateSha256 that returns the hash of the file.
  Then function stores in the log file log.txt a record of the transaction with the next format, as seen in the Figure 19:
  [date + time] file name : hash in sha256 format
  Lastly, the function removes the file from the file system and returns the hash of the file.



```python
@app.post("/get-hash-sha256")
async def calculate_hash(file: UploadFile = File(...)):
    with open(getcwd() + "/" + file.filename, "wb") as myfile:
        content = await file.read()
        myfile.write(content)
        myfile.close()
        sha256 = calculatehash256.calculateSha256(file.filename)
        print(file.filename + ": " + sha256)
    with open("log.txt", "a") as log_file:
        now = datetime.now()
        log = "[" + now.strftime("%Y-%m-%d %H:%M:%S") + "] " + file.filename + ": " + sha256 + "\n"
        log_file.write(log)
    os.remove(os.getcwd() + "/" + file.filename)
    return {"" + sha256 + ""}
```

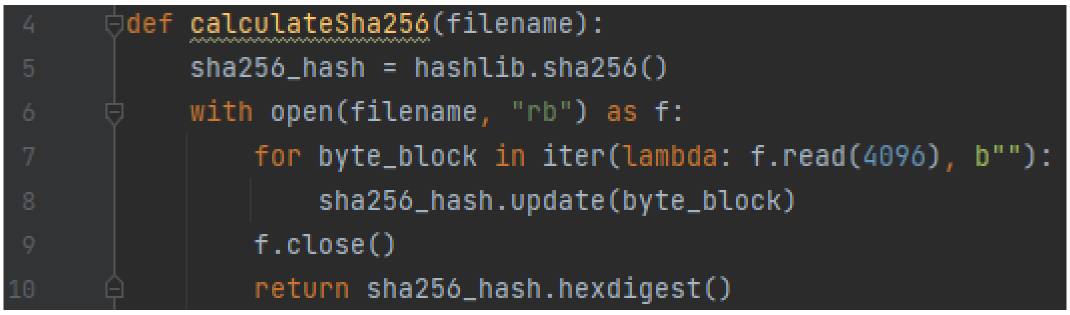
**Figure 18: Python API get-hash-sha256 function. Src: Victor Barroso Corchero**



```
[2022-06-06 18:37:43] Ezra Vopol lore.pdf: bf877069eac9a62dc7f74974bd9377a10422c3a9c69b088dd7880c75e400bcdf
[2022-06-06 18:37:46] Ezra token.png: 47c55f451992ecd70927ca5dd6ac2ab87c47e89f88f3dc45e7613898673963de
[2022-06-06 18:37:50] EstudiViabilitat_BarrosoCorcheroVictor.docx: 12f10211a6845575a8cff5937415ceab57f6d20f2d342a2bd245f7b6efbddd67
[2022-06-06 18:37:54] EstudiViabilitat_BarrosoCorcheroVictor.pdf: d90c80e13f11b185d34da4db0f25dac8488eeee927111d7d350449bee844a000
[2022-06-06 18:38:06] GUIÓN.docx: 2460dd8bb5355d9717c219ed60bebaee0e40663b936181890a38c5dc4d52e2a5
[2022-06-06 18:38:13] 5127255752466587140258885.pdf: 7733db22947c50d115bb858718e3ad807afa202b2ba42902ddf6918e1ce93a7b
[2022-06-06 18:38:24] 05_memoria_TFG.docx: 556df60a156772e94b52a5c3037a27cb55907b00f8a2235f49f8939d392e7b57
[2022-06-06 18:38:30] 2022-05-22 (1).png: 91bf648457af37fdf64c4caa20a81f40ca84c94d0130339f71c59a88957f3f2c
[2022-06-06 18:46:19] 2022-05-22.png: 5e7f66156f8ee9ff35ccc907f71d2a202795599e4bb0dc7b945cc21697a4e7c2
[2022-06-06 18:46:39] 2022-05-22 (1).png: 91bf648457af37fdf64c4caa20a81f40ca84c94d0130339f71c59a88957f3f2c
[2022-06-06 18:46:55] kaleido.txt: eaf0406073c5291e76187e0408a8952fbfb7d212a5dca3e453f0de86077cb6fb
```

**Figure 19: log.txt generated by the API Src: Victor Barroso Corchero**

Besides the API method, the Figure 20 shows the code of the calculateSha256 function, which calculates the hash of the given file. This function takes the filename as a parameter, used for finding the file and calculating the hash, then for doing this operation it reads and updates the hash string value in blocks of 4096 bytes. The reason the function reads the file in blocks of 4096 bytes is to handle larger size files.



```python
def calculateSha256(filename):
    sha256_hash = hashlib.sha256()
    with open(filename, "rb") as f:
        for byte_block in iter(lambda: f.read(4096), b""):
            sha256_hash.update(byte_block)
        f.close()
        return sha256_hash.hexdigest()
```

**Figure 20: Hash sha256 calculator Src: Victor Barroso Corchero**

## 5.5 Front-end development

The development of the user interface is addressed in this point. The chosen technology for doing the development is React [31], which is a free and open-source front-end JavaScript library, based on UI components.

The webpage is a prototype filling the text gaps with Lorem Ipsum[25], in addition, there are four forms, each of those used for a different requirement and explained later in this section.

Moreover, the style of the webpage is done with the help of TailWin CSS [32], which is an open-source CSS framework used for web design. With the help of this framework the webpage is responsive, meaning that it has a good visual aspect regardless of the device from which the application is being used.

Furthermore, the API calls have been done with the Axios [33], a react package used as HTTP client.

In the Figure 21 it can be appreciated how the UI[26] adapts to different resolutions. From left to right, the first resolution could represent how the application is seen by a mobile phone, the second one by a tablet and the third one by a laptop.

[25] Placeholder text commonly used to demonstrate the visual form of a document without relying on meaningful content

[26] User Interface

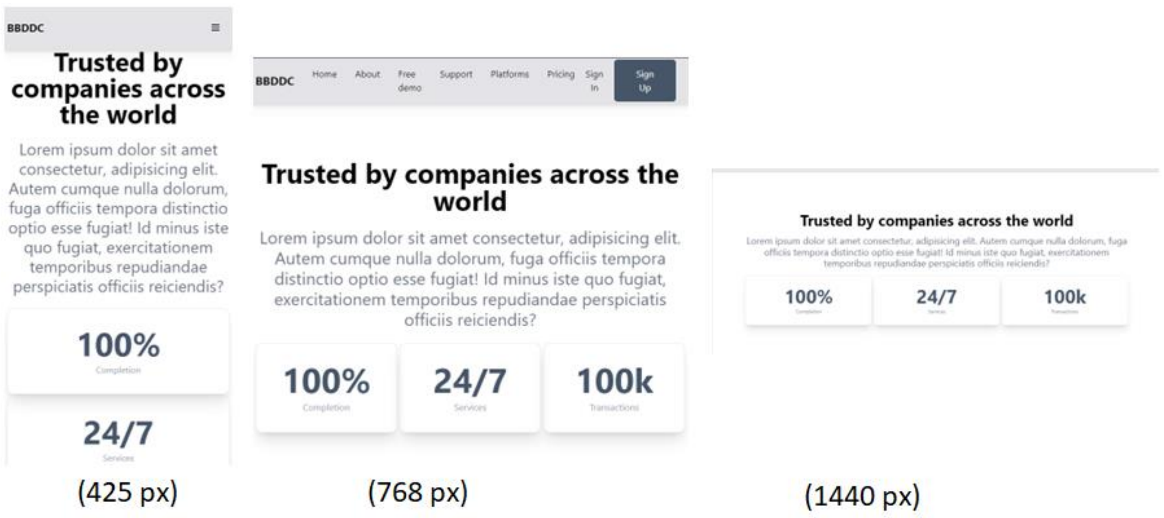
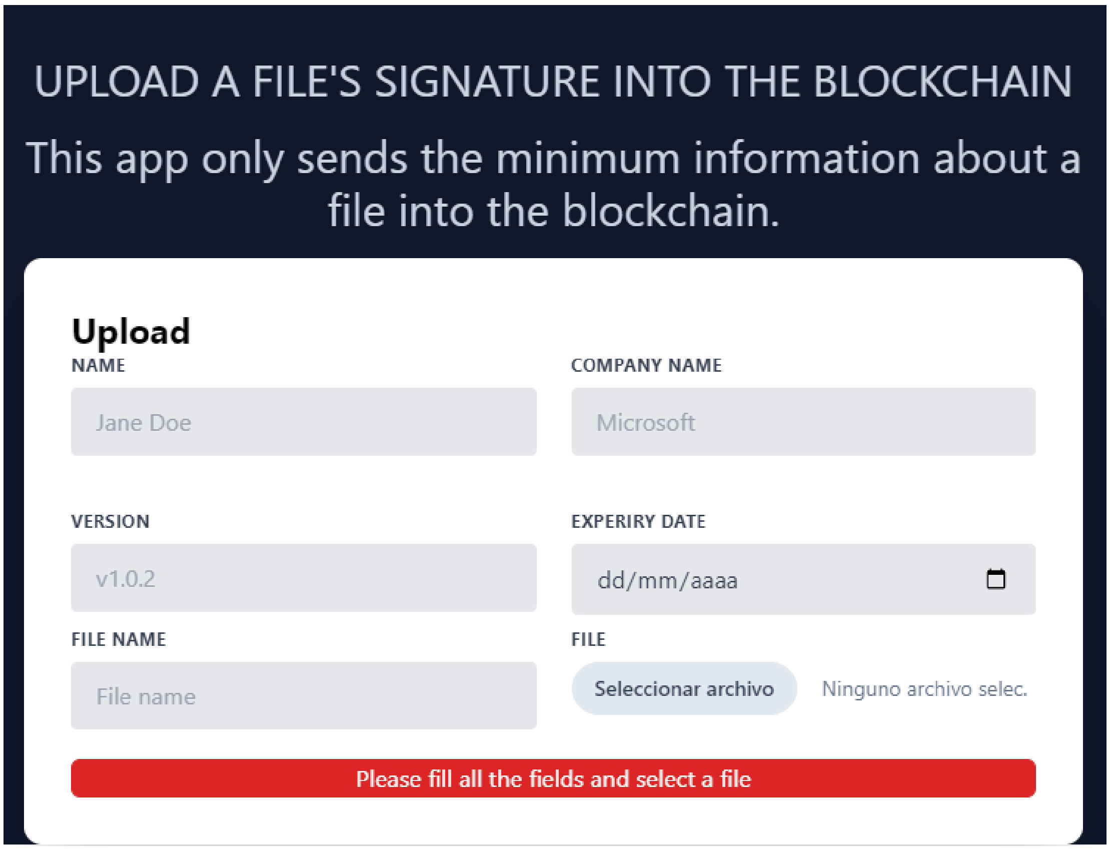

BBDDC
≡
Trusted by companies across the world
Lorem ipsum dolor sit amet consectetur, adipisicing elit. Autem cumque nulla dolorum, fuga officiis tempora distinctio optio esse fugiat! Id minus iste quo fugiat, exercitationem temporibus repudiandae perspiciatis officiis reiciendis?
100%
Completion
24/7
Services
100k
Transactions
Home
About
Free demo
Support
Platforms
Pricing
Sign In
Sign Up
(425 px)
(768 px)
(1440 px)

**Figure 21: Web application responsiveness Src: Victor Barroso Corchero**

### 5.5.1 Upload a file signature into the blockchain

This is the first form the web application shows, all fields must be fulfilled for uploading the file's information into the blockchain, as shown in the Figure 22.

When the user selects a file, not just in this form but in any other, the application makes an API call to the python API shown in Figure 18 to retrieve the hash. Afterwards, when all fields have been filled and the hash has been calculated the user is able to click the button for uploading the file's information into the blockchain.


UPLOAD A FILE'S SIGNATURE INTO THE BLOCKCHAIN

This app only sends the minimum information about a file into the blockchain.

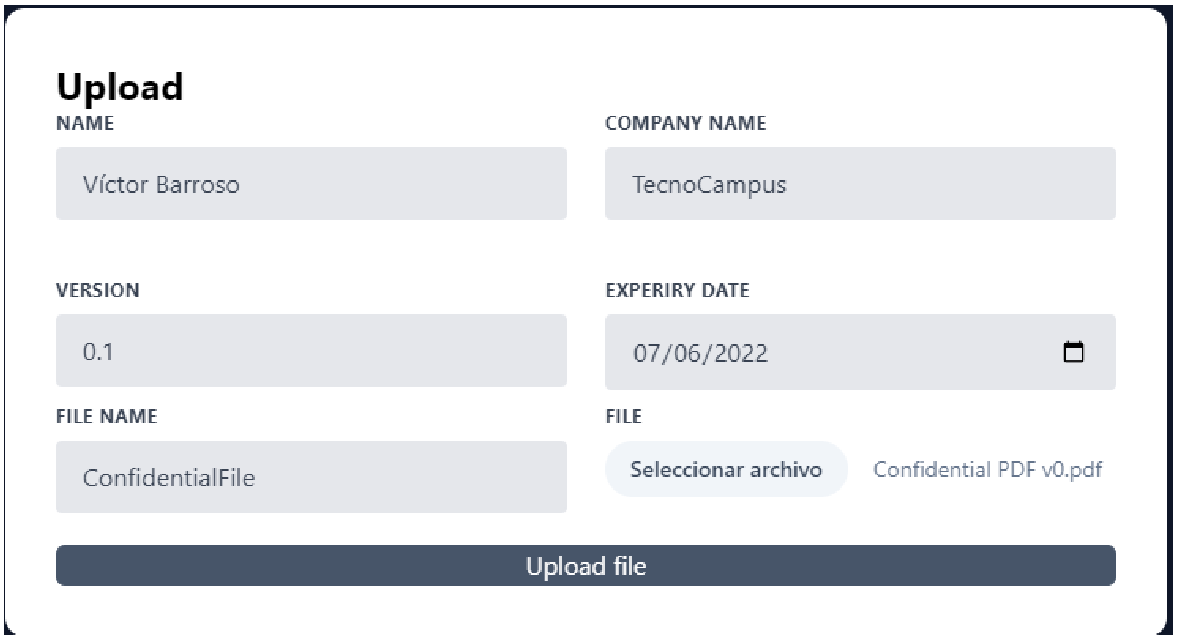
Upload

| | |
|---|---|
| NAME | COMPANY NAME |
| Jane Doe | Microsoft |
| VERSION | EXPERIRY DATE |
| v1.0.2 | dd/mm/aaaa |
| FILE NAME | FILE |
| File name | Seleccionar archivo Ninguno archivo selec. |

Please fill all the fields and select a file

**Figure 22: Web App, Upload file's signature, empty form Src: Víctor Barroso Corchero**

In the Figure 23 it is shown an example of a completed form, notice that the button's color and text has changed. Later on, when the user clicks the button two floating notifications pops up, in the Figure 24 the notification pops up when the transaction is being processed, and in the Figure 25 it pop ups when the transaction is completed.

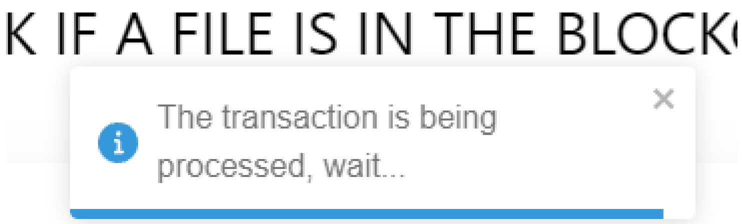


Upload

| | |
|---|---|
| NAME | COMPANY NAME |
| Víctor Barroso | TecnoCampus |
| VERSION | EXPERIRY DATE |
| 0.1 | 07/06/2022 |
| FILE NAME | FILE |
| ConfidentialFile | Seleccionar archivo Confidential PDF v0.pdf |

Upload file

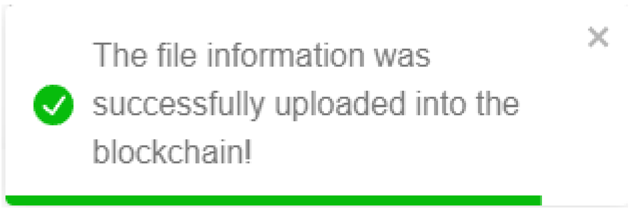
Figure 23: Web App, Upload file's signature, complete form Src: Víctor Barroso Corchero



K IF A FILE IS IN THE BLOCK

The transaction is being processed, wait...

Figure 24: Web App, Upload file's signature waiting notification

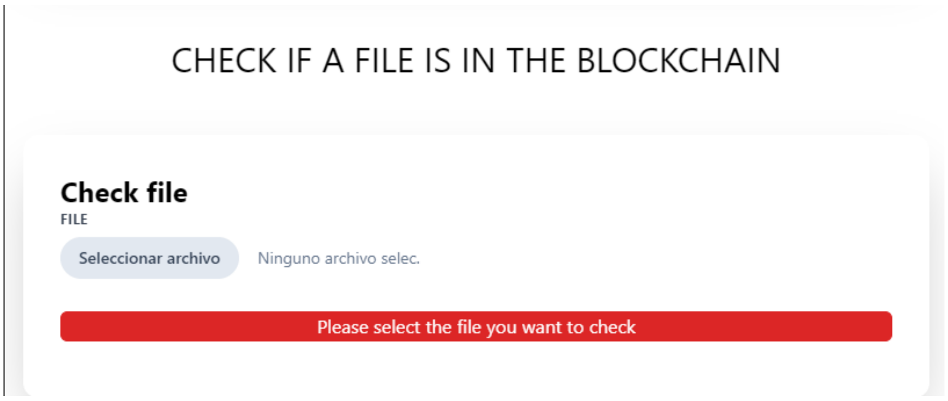


The file information was successfully uploaded into the blockchain!

Figure 25: Web App, Upload file's signature toast success notification

### 5.5.2 Check if a file is in the blockchain

The form shown in the Figure 26 is used for checking if the file information is in the blockchain or if it is not. As mentioned earlier, the button is disabled until the user selects a file.



CHECK IF A FILE IS IN THE BLOCKCHAIN

**Check file**

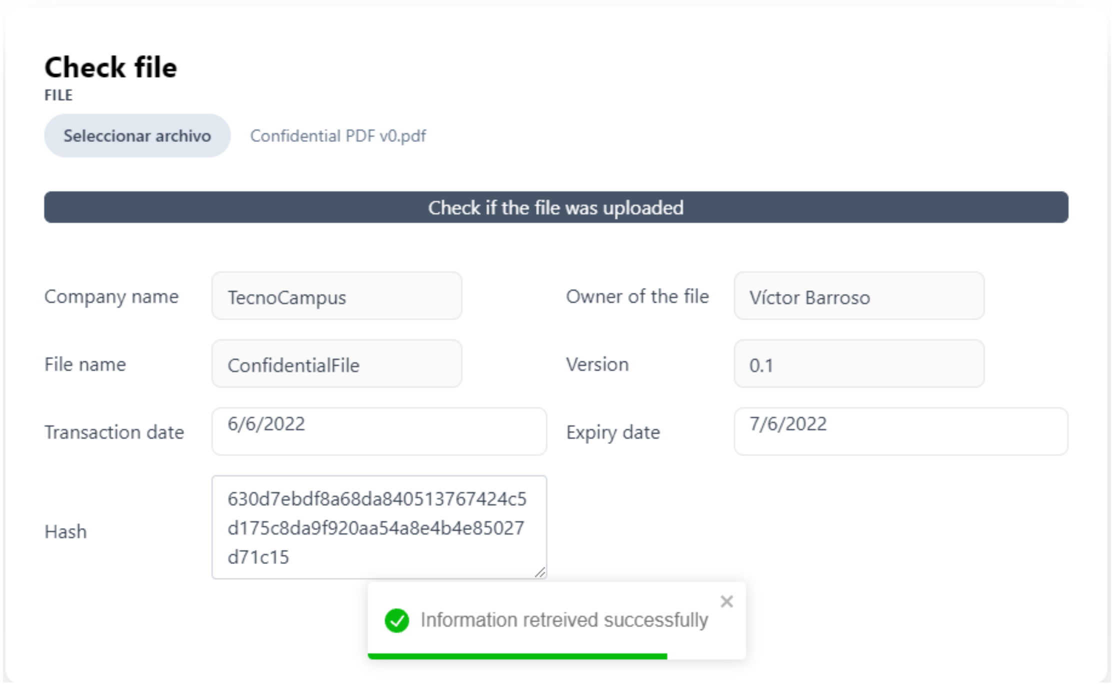
FILE

Seleccionar archivo Ninguno archivo selec.

Please select the file you want to check

**Figure 26: Web App, Check file, no file selected. Src: Víctor Barroso Corchero**

Moreover, in the Figure 27 it is shown how the information is seen when the user checks a file that is already in the blockchain. It can be seen that the fields' values are the same as the ones shown in the Figure 23. On the other hand, if the file is not found it shows a notification as seen in the Figure 28.



**Check file**

FILE

Seleccionar archivo Confidential PDF v0.pdf

Check if the file was uploaded

| | | | |
|---|---|---|---|
| Company name | TecnoCampus | Owner of the file | Víctor Barroso |
| File name | ConfidentialFile | Version | 0.1 |
| Transaction date | 6/6/2022 | Expiry date | 7/6/2022 |
| Hash | 630d7ebdf8a68da840513767424c5d175c8da9f920aa54a8e4b4e85027d71c15 | | |

Information retreived successfully

**Figure 27: Web App, Check file, file information retrieved. Src: Víctor Barroso Corchero**

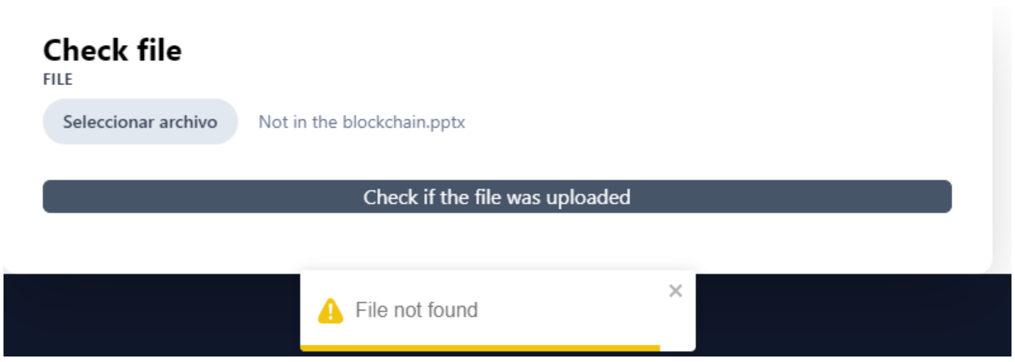


**Figure 28: Web App, Check file, file not found. Src: Víctor Barroso Corchero**

### 5.5.3 Check if a file is the last version

The form shown in the Figure 29 is used to check if the file is uploaded into the blockchain and corresponds to the last version. When the user selects a file and clicks on the button if the file is the last version it shows the notification shown in the Figure 30, otherwise it shows the notification shown in the Figure 31.

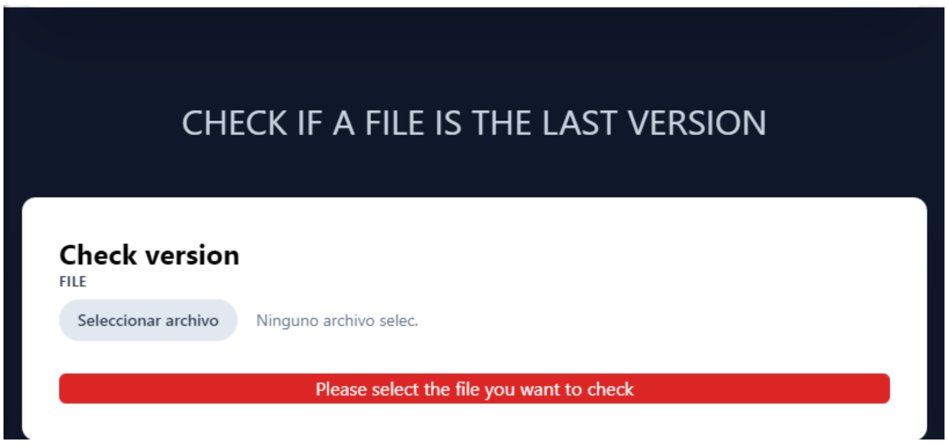

CHECK IF A FILE IS THE LAST VERSION
Check version
FILE
Seleccionar archivo
Ninguno archivo selec.
Please select the file you want to check

**Figure 29: Web App, Check file version, no file selected. Src: Víctor Barroso Corchero**

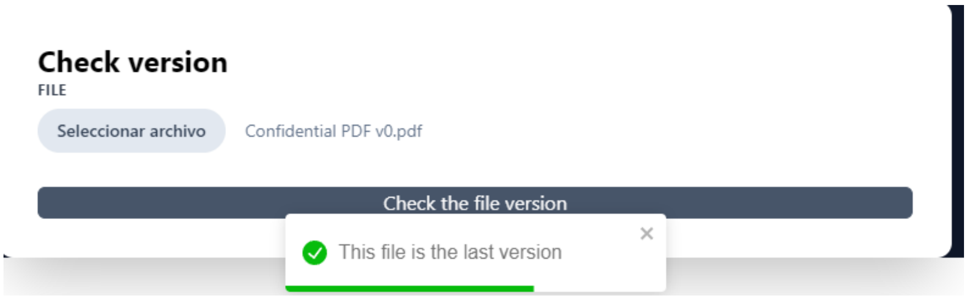
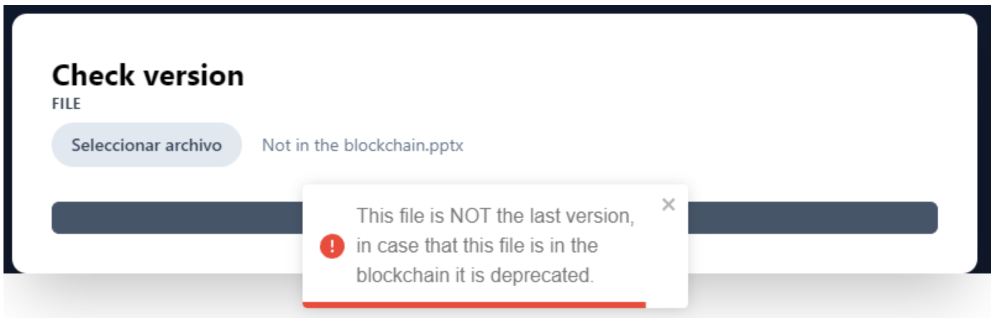


**Figure 30: Web App, Check file version, file is last version. Src: Víctor Barroso Corchero**



Not in the blockchain.pptx

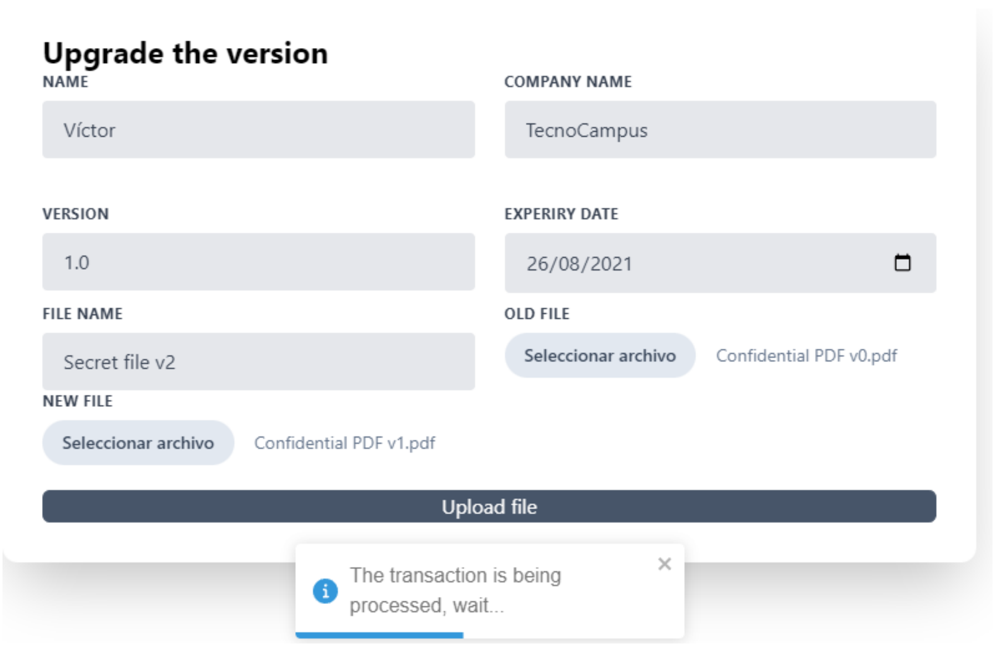
This file is NOT the last version, in case that this file is in the blockchain it is deprecated.

**Figure 31: Web App, Check file version, the selected file is not the last version / does not exist in the blockchain. Src: Víctor Barroso Corchero**

### 5.5.4 Upgrade the version of a previous uploaded file

This is the last form and is the one used to upgrade the version of a file that is currently on the blockchain, as seen in the Figure 32 the form is like the one shown in the Figure 22, corresponding to the upload file form, but has an extra field, the old file. To upgrade the file the user has to select the old file version and the new.

In the Figure 33 shows the notification alert when the transaction is completed. As an example, since the Figure 32 is upgrading the version of the file that was uploaded in the Figure 22, the file of the Figure 22 should not be the last version of the file, as shown in the Figure 34. Also, in the Figure 35 it can be appreciated that the new version is correctly detected by the blockchain.



| | |
|---|---|
| NAME | Víctor |
| COMPANY NAME | TecnoCampus |
| VERSION | 1.0 |
| EXPERIRY DATE | 26/08/2021 |
| FILE NAME | Secret file v2 |
| OLD FILE | Confidential PDF v0.pdf |
| NEW FILE | Confidential PDF v1.pdf |

The transaction is being processed, wait...

**Figure 32: Web App, upgrade file version, completed form. Src: Víctor Barroso Corchero**

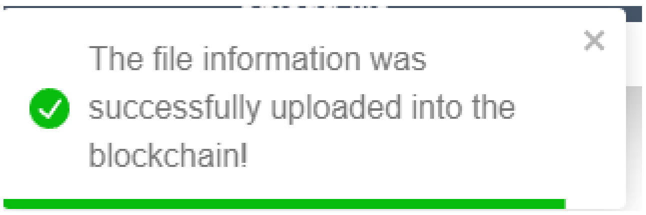
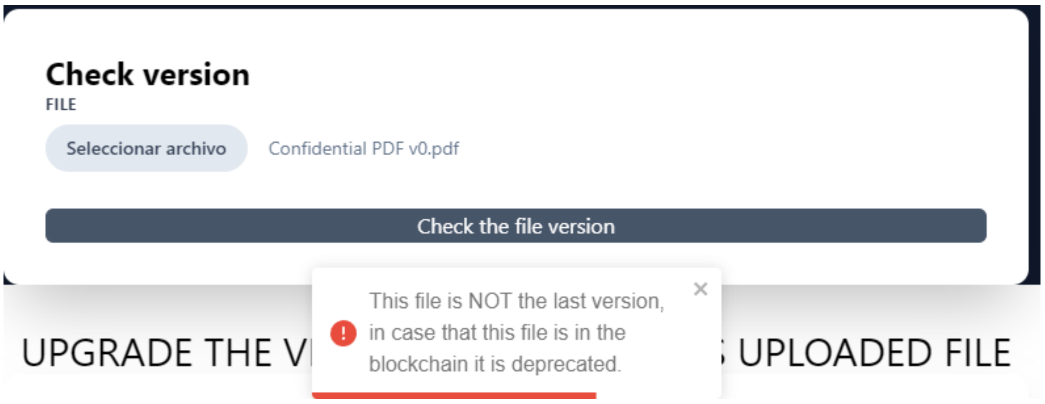
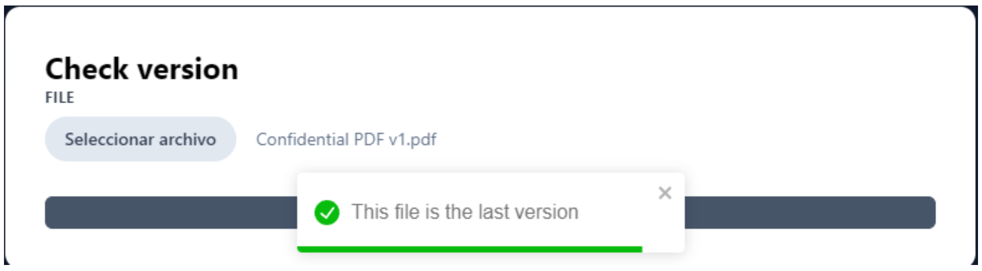

The file information was successfully uploaded into the blockchain!

Figure 33: Web App, upgrade file version, success toast. Src: Víctor Barroso Corchero



This file is NOT the last version, in case that this file is in the blockchain it is deprecated.

Figure 34: Web App, Check file version after upgrading version, file is not the last version. Src: Víctor Barroso Corchero



This file is the last version

Figure 35: Web App, Check file version after upgrading version, file is last version. Src: Víctor Barroso Corchero

# 6 Conclusion

This project studies potential cybercrimes that companies are exposed to, also how the amount of cyberattacks and cybercriminals have been growing over the past few years and the impact of this.

Moreover, this project focuses its objectives on reducing the probability that a company suffers from Social Engineering cyberattacks, such as phishing or CEO fraud. This has been achieved thanks to the blockchain technology, which has also been studied in this project along the most popular blockchain technologies.

The developed prototype consists of three main parts:

- **Blockchain environment**: The blockchain used for this project is private and is managed through the Kaleido platform. The blockchain uses the Ethereum technology, also the Smart Contract is developed with the programming language Solidity
- **Python API**: This API is used for calculating the hash of the files using the sha256 function and storing the log of calculated hashes.
- **Web application**: The web application is developed with React, interacts with the blockchain through the Kaleido API, from which the Smart Contract functions are called, and uses the Python API as a support for getting the hash of the files.

To conclude with, in recent years cybercrime has grown exponentially and tools like the one developed are needed to keep companies and individuals safe. Furthermore, with the help of the blockchain technology the solution ensures the immutability of the stored data and a control version, leading to bringing a transparent tool which keeps tracks of the transactions and maintains the integrity and immutability of the stored data.

The developed application can be called a Decentralized Application, or DApp, this application inserts the signature of a file and some metadata of it, so it can identify the transaction later on, with the purpose of ease the work of validating any kind of digital file.

# 7 Possible next steps

This point explains what should be done next in order to complete and improve the prototype and acquire a potential tool for preventing Social Engineering cyberattacks.

First of all, there is a functionality that couldn't be developed since the free trial of the chosen platform had limitation. This feature is the user and role control, which in the Kaleido platform is done through the Cloud Hardware Security Module Wallets, explained in the point 5.3 of this project.

Furthermore, more applications could be developed to facilitate and speed up the process of checking a file. And some examples could be:

- Web browser plugin: A plugin that enables you to drop a file into a pop-up without the need of using the web application.
- Email client plugin: A plugin that every time a message is received checks the veracity of the files attached to it and alerts the user if they are not in the blockchain.
- New features of the web application:
    - Screen with all the files uploaded for the current user.
    - Access to all the blockchain records with filters.

Lastly, the Smart Contract could be updated and add more functionalities such as increasing the fields a file should have, like the name of the original file with the extension. Then all the data from the blockchain could be used to create reports.

# 8 Bibliography

[1] "Allot," Allot Ltd., [Online]. Available: https://www.allot.com/. [Accessed 13 January 2022].

[2] Allot Ltd., "Allot Ltd.," 2021. [Online]. Available: https://www.allot.com/resources/2020-Cyber-Threat-Report.pdf. [Accessed 13 January 2022].

[3] "IC3," [Online]. Available: https://www.ic3.gov/. [Accessed 13 January 2022].

[4] P. Abbate, "IC3," 2021. [Online]. Available: https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf. [Accessed 13 January 2022].

[5] "Europol," [Online]. Available: https://www.europol.europa.eu/about-europol. [Accessed 17 January 2022].

[6] Europol, "Internet Organised Crime Threat Assessment (IOCTA) 2021," Publications Office of the European Union, Luxembourg, 2021.

[7] D. M. Turner, "Digital Strategy," European Commission, [Online]. Available: https://digital-strategy.ec.europa.eu/en/policies/eidas-regulation. [Accessed 30 January 2022].

[8] W. Vercruysse, "European Commission," 3 August 2020. [Online]. Available: ec.europa.eu/cefdigital/wiki/display/ESIGKB/What+are+the+levels%2C+simple%2C+advanced+and+qualified+of+electronic+signatures. [Accessed 30 January 2022].

[9] "OpenPGP," [Online]. Available: https://www.openpgp.org/. [Accessed 10 January 2022].

[10] M. Crosby, Nachiappan, P. Pattanayak, S. Verma and V. Kalyanaraman, "Sutardja Center for Entrepreneurship & Technology Technical Report. Url: https://scet.berkeley.edu/wp-content/uploads/BlockchainPaper.pdf".

[11] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," 1991.

[12] "Geeks for Geeks," [Online]. Available: https://www.geeksforgeeks.org/about/?ref=footer. [Accessed 25 January 2022].

[13] jagroopofficial, "Geeks for Geeks," 7 September 2021. [Online]. Available: https://www.geeksforgeeks.org/different-version-of-blockchain/. [Accessed 25 January 2022].

[14] J. Frankenfield, "Investopedia," 16 November 2021. [Online]. Available: https://www.investopedia.com/terms/d/decentralized-applications-dapps.asp. [Accessed 31 January 2022].

[15] Ethereum, "ethereum.org," Ethereum, 17 December 2021. [Online]. Available: https://ethereum.org/en/developers/docs/intro-to-ethereum/#what-is-a-blockchain. [Accessed 11 January 2022].

[16] ethereum.org, "Ethereum," 24 January 2022. [Online]. Available: https://blog.ethereum.org/2022/01/24/the-great-eth2-renaming/. [Accessed 1 February 2022].

[17] Solana, "Solana," [Online]. Available: https://solana.com/. [Accessed 1 February 2022].

[18] A. Yakovenko, "Solana: A new architecture for a high performance blockchain v0.8.13," 1 February 2022. [Online]. Available: https://solana.com/solana-whitepaper.pdf.

[19] C. Barker, "Solana," 20 August 2021. [Online]. Available: https://solana.com/news/getting-started-with-solana-development. [Accessed 2 February 2022].

[20] C. Staff, "Cryptopedia," Gemini Trust Company, 3 December 2021. [Online]. Available: https://www.gemini.com/cryptopedia/eos-blockchain-architecture-eosio. [Accessed 3 February 2022].

[21] "EOSIO," Block.one, [Online]. Available: https://eos.io/. [Accessed 2 February 2022].

[22] "Block.one," [Online]. Available: https://b1.com/. [Accessed 2 February 2022].

[23] "ESED," [Online]. Available: https://www.esedsl.com/. [Accessed 9 February 2022].

[24] S. Balaji and M. Murugaiyan, "WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC.," 2012.

[25] "WhatsApp," Meta, [Online]. Available: https://www.whatsapp.com/. [Accessed 9 February 2022].

[26] "Google Workspace," Google, [Online]. Available: https://workspace.google.com/intl/es/products/chat/. [Accessed 8 February 2022].

[27] "ClickUp," [Online]. Available: https://clickup.com/. [Accessed 8 February 2022].

[28] "Amazon Web Services," Amazon, [Online]. Available: https://aws.amazon.com/. [Accessed 7 February 2022].

[29] Kaleido, Inc., "Kaleido," [Online]. Available: https://www.kaleido.io/. [Accessed 20 April 2022].

[30] "Web3js," [Online]. Available: https://web3js.readthedocs.io/en/v1.7.3/#. [Accessed 19 April 2022].

[31] Meta Open Source, "React," [Online]. Available: reactjs.org. [Accessed 1 6 2022].

[32] Tailwind CSS, "tailwindcss," [Online]. Available: tailwindcss.com/. [Accessed 6 6 2022].

[33] Axios, "Axios," [Online]. Available: axios-http.com. [Accessed 3 6 2022].

[34] J. Frankenfield, "Blockchain-as-a-Service (BaaS)," 2021.