ΜΗΤΡΟΠΟΛΙΤΙΚΟ ΚΟΛΛΕΓΙΟ

University of East London

SCHOOL OF ARCHITECTURE, COMPUTING & ENGINEERING

MSc Information Security & Digital Forensics

Master's thesis entitled:

Research on the synergy of private Blockchains and Chain of Custody applications in digital forensics: Development of private Ethereum Blockchain and Chain of Custody application

THESIS EDITING

BANDIS CHRISTOS
U2121211

SUPERVISOR

Dr. LIAMBAS CHRISTOS

INSTITUTION

METROPOLITAN COLLEGE CAMPUS
AMAROUSIOU

August, 2023

Acknowledgements

I would like to express my sincere thanks to all those who contributed to the completion of this thesis. The journey of researching and creating this project has been significant and without the support and assistance of the following individuals, this achievement would not have been possible.

I would like to thank my supervisor, Dr. Liambas Christos, for his consistent guidance, trust and motivation throughout the whole duration of this thesis. Your invaluable insights and constructive criticism have greatly enriched the structure and depth of this study.

Finally, I would like to express my grateful thanks to my family and friends who have supported me and encouraged me during this journey.

Abstract

The rapidly evolving nature of technology and the widespread dissemination of digital evidence have created an urgent need to intensively review and strengthen digital forensic practices. Developments in the IT landscape over the past two decades have made the collection, preservation and analysis of digital evidence a valuable tool for resolving cases and administering justice to the parties involved. Therefore, protecting this evidence from any form of tampering is of paramount importance. In this context, the integration of private blockchains and chain of custody applications is a promising avenue to address the challenges related to maintaining the integrity and authenticity of digital evidence.

A blockchain is a digitally distributed ledger of transactions signed through a cryptographic function in chronological order. Its content is classified into a block and is characterized by the transparency it provides to all participants of the network (Sathyaprakasan et al., Therefore, taking advantage of the aforementioned transparency, but also the inherent security it offers and utilizing the “Ethereum” blockchain as the fundamental development platform, a private blockchain network was designed and developed, aiming to meet the particular requirements of digital forensic investigations. More specifically, the private network provides a decentralized platform for participants, who have the ability to securely record, verify and monitor the traffic of forensic investigations. This implementation aims to ensure enhanced privacy, access control and scalability, while ensuring a high level of data integrity.

At the same time, a decentralized chain of custody application was developed, aiming at the secure recording of traffic and management of digital evidence. The application takes full advantage of the capabilities of the private blockchain, allowing authorized users to easily access - in real time - information about the ownership of evidence, its transfer history and its integrity, reducing risks such as tampering with evidence and unauthorized access.

Keywords

Digital Evidence, Blockchain, Chain of Custody, Ethereum, Private Network, Decentralized Application

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1. Research context

The field of digital forensics has experienced remarkable growth and visibility in recent years, largely attributed to rapid developments in technology and the widespread use of electronic devices in various aspects of modern life. The increasing reliance of legal processes on digital evidence and the ever-expanding range of cybercrimes lead to new challenges and complexities for digital forensics investigators. Traditional methods of handling digital evidence often face limitations regarding ensuring the integrity, verifiability and traceability of data, raising concerns about the reliability of findings presented in court.

In digital forensics, evidence follows a hierarchy, passing through different levels, starting with the first responder and reaching the higher authorities responsible for managing cybercrime investigations. During this transmission of digital evidence, there is always a high risk of its integrity being compromised, making it inherently vulnerable to various incidents of tampering and forgery. The presence of such incidents is usually due to the continuous technological development and lack of knowledge and expertise at the level of investigators where, in the process of investigating a particular cybercrime case, the collection, storage and analysis of digital forensic evidence is carried out (Lone and Mir, 2019).

In the field of digital forensics, ensuring the integrity and authenticity of evidence is vital to investigative and legal processes. In a legal context, verifying authenticity involves proving that the digital evidence originates from its purported source, has not been altered since it was acquired, and relevant details, such as the apparent date of the record, are accurate (Ćosić and Bača, 2010). Preserving the provenance information of digital evidence is a particularly challenging process, given the ease with which it can be copied, altered or destroyed, as the number of parties who can edit it, increases the risk of alteration, either intentional or unintentional.

The solution to the above problem is provided by the maintenance of a strong chain of custody, which records the chronological history and control of evidence from the moment it is obtained until it is presented to the court (Giova, 2011). A chain of custody includes details of who processed the digital evidence, when, where how and usually provides a way of verifying its integrity. The most common way of verifying the integrity of digital evidence is the use of a cryptographic hashing function, which

allows the stakeholder to verify that the evidence has not been tampered with or processed in any way after it was received (Rasjid et al., 2017).

1.2. Existing Problem

Traditional methods of maintaining a chain of custody are prone to challenges such as data corruption, lack of transparency and reliance on one or more central authorities. The database systems involved in these methods are also unable to maintain the integrity, originality and confidentiality of the evidence collected, as well as the history of events that occurred in a particular sequence during the collection, transfer, storage, analysis and interpretation of evidence to solve a case (Rochmadi and Heksaputra, 2019).

To address these challenges, several researchers and practitioners have turned to blockchain technology as a promising solution. The blockchain, as a distributed ledger, provides multiple benefits that make it ideal for ensuring the integrity and transparency of a chain of custody, since by design, it guarantees transparency, authenticity, security and auditability, offering the possibility of creating an immutable record of all actions and movements related to digital evidence. (Xu, Chen and Kou, 2019).

Utilizing blockchain technology, every action taken by a forensic investigator when processing digital evidence, can be recorded and stored securely. This creates a clear and permanent audit trail, where every attempt to transfer and access data is recorded in an automated way, thanks to “smart” contracts (Guo et al., 2022). These smart contracts, developed and executed on blockchain platforms such as Ethereum, ensure that the chain of custody is maintained and enforced according to predefined rules and protocols.

The use of blockchain technology, in combination with the chain of custody, offers many advantages. It provides a secure and decentralized means of monitoring the movement and management of digital evidence, reducing the risks of data tampering and unauthorized access. In addition, it improves transparency by allowing competent parties to easily monitor and verify the history of evidence, reducing the need for reliance on central authorities.

This paper explores the combined potential of private blockchain and chain of custody applications in the field of digital forensics. The study includes comprehensive

research on blockchain technology, digital forensics, the contribution of private blockchains to it, and the ways that private blockchains can contribute to the maintenance of a chain of custody, leading to the development of a private blockchain based on the Ethereum platform and a decentralized chain of custody application.

1.3. Objectives of the thesis

The rapidly evolving technological landscape and the pervasive presence of digital evidence have necessitated the application of robust and secure methodologies in the field of digital forensics. As the volume and complexity of digital data continues to increase, the need for enhanced verifiability, traceability and integrity of digital evidence has become a primary concern for forensic investigators and, by extension, law enforcement agencies.

The rise of blockchain technology, known for its enhanced privacy, data integrity and its - tamper-resistant - nature, combined with the concept of chain of custody, which ensures the secure tracking and management of digital evidence, presents a promising avenue for addressing the shortcomings of traditional digital forensics methods.

In response to the aforementioned challenges, this thesis, utilizing the robustness of blockchain technology and more specifically, the blockchain platform “Ethereum”, attempts to explore the potential synergies between private blockchain and chain of custody applications, with the aim of developing an innovative and reliable solution for digital forensic investigations. This solution includes the creation of a private blockchain, tailored to the specific requirements of digital forensic applications, as well as the development of a decentralized chain of custody application designed to record and monitor the movement and management of digital evidence throughout the investigation process. The integration of these two elements seeks to confer a higher degree of confidence in digital forensic findings, enhancing the admissibility and reliability of digital evidence in legal proceedings. Through an empirical evaluation of the developed solution, this study aims to contribute to the advancement of digital forensic practices by promoting trust and transparency in the ever-evolving world of digital technology.

Chapter 2: The Blockchain

Blockchain is a fast-growing distributed ledger technology that has expanded from initially decentralized financial transactions, to more "real world" applications such as commerce, supply chain services, resource allocation services, transportation of goods, "IoT" etc. (Chen and Liang, 2022). Furthermore, blockchain technology has started to have an impact on the legal system and particularly on the rules of evidence. These application scenarios have brought new challenges to the original blockchain technology, prompting developers and researchers to continuously explore innovative solutions to address issues of scalability, security, privacy and interoperability. In order to realize a full understanding of this particular technology, it is necessary to provide a definition of it. Blockchain can be defined as a distributed ledger technology that can permanently and securely record transactions between parties. This decentralized ledger is maintained through various nodes connected to each other through networks, which are responsible for communicating and recording transactions (Umoren et al., 2022). Over the years, blockchain technology has evolved and has paved the way for integrating or combining decentralized applications with other technologies.

With the methods of data storage, exchange and synchronization used in a network of remote computers, the decentralized and "untrusted" blockchain can effectively solve the problems of data loss and forgery in a centralized storage system, thus reducing the cost of information preservation and trust and providing a more reliable method for the forensic examination of electronic evidence. Blockchain technology has revolutionized data security with its secure storage mechanism, which involves recording transactions using immutable cryptographic signatures (Umoren et al., 2022).

Simply put, the blockchain is a series of connected data structures called "blocks" that contain or track everything that happens on any distributed systems on a "peer to peer" network. Each block is connected to the previous one through a special pointer called a "hash pointer", forming a chain (Figure 1), resulting in a system whose data is not subject to any processing (Nakamoto, 2008).

Blockchain, by definition, guarantees the transparency, authenticity, security and auditability of data and, therefore, becomes as the most suitable for maintaining the chain of custody of criminal investigations (Lone and Mir, 2019).

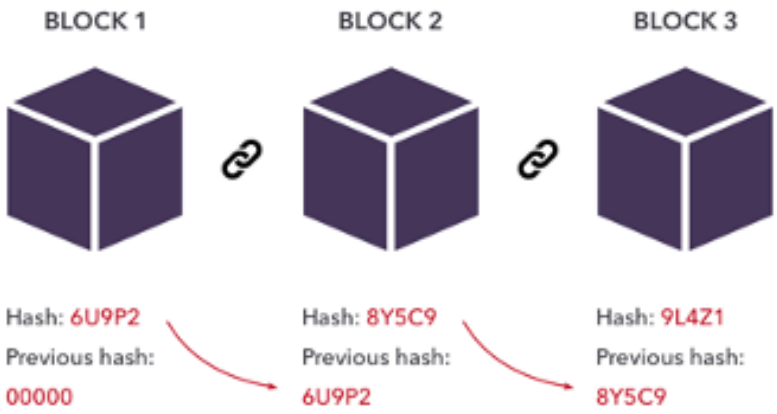

Figure 1 - Connection of blocks in a blockchain

2.1. Blockchain Structure and Types

The blockchain is a data structure that allows the creation of a digital ledger to record and store transactions, which are shared by all participating parties through a distributed computer network, using cryptography to protect them, creating an unbreakable audit trail (Lone, 2017). In contrast, existing systems follow the client-server architecture and store their data entirely at a target point. The result of this is the creation of vulnerabilities, leading to instances of data loss, which in turn may cause failure or malicious activity risks.

A blockchain consists of multiple blocks and uses the hash function of each block to connect it to the previous one. Thus, a block is like a page of a ledger or a record book. Each time a block "completes", it gives way to the next block and so on. A block is therefore a permanent storage location for records that, once registered, cannot be modified or removed. These blocks are placed "on top of each other", with the "Genesis" block being the foundation. The "Genesis" block, also known as the block 0, is the first block, upon which the additional blocks on a blockchain are created. It is essentially the "ancestor" to which every other block can trace its origin, since each block refers to the one that precedes it (Tardi, 2021) (Figure 2).

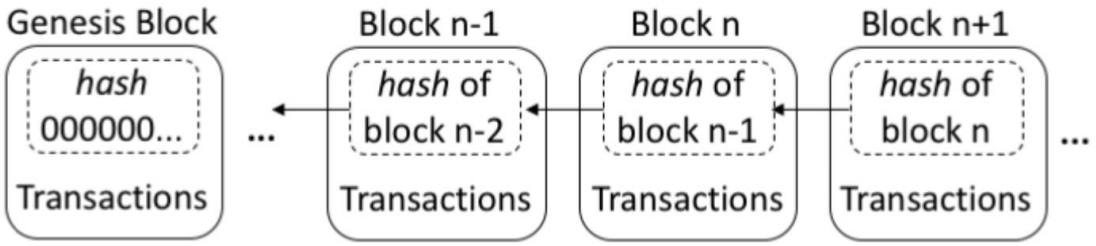

Figure 2 - How the genesis block is connected to the other blocks (Oliveira et al., 2019)

At its core, a blockchain consists of three basic elements: the blocks, which, as mentioned above, store the data; the chain, which links these blocks together via the hash function; and the consensus mechanism, which allows multiple participants to reach an agreement on the validity of new transactions and maintain the integrity of the network. In more detail, the main features of a blockchain are as follows:

- **Cryptographic Hash Function**

Hash functions are one of the most important elements of blockchain technology. They are algorithms used in cryptography to convert existing data into fixed-length encrypted data (Kahate, 2017) (Figure 3). However, any, change in them will produce a result completely different from the original one.

Hashing algorithms such as "SHA-256" (Secure Hash Algorithm-256) or "Scrypt" are commonly used by blockchain because they are easy to verify but hard to forge, thus allowing the creation of digital signatures that blockchain users need to authenticate themselves or their transactions (Fernandez-Carames and Fraga-Lamas, 2020). Hash functions are also used by blockchains to connect their blocks. The blocks are linked together in chronological order, each containing the hash of the previous block.

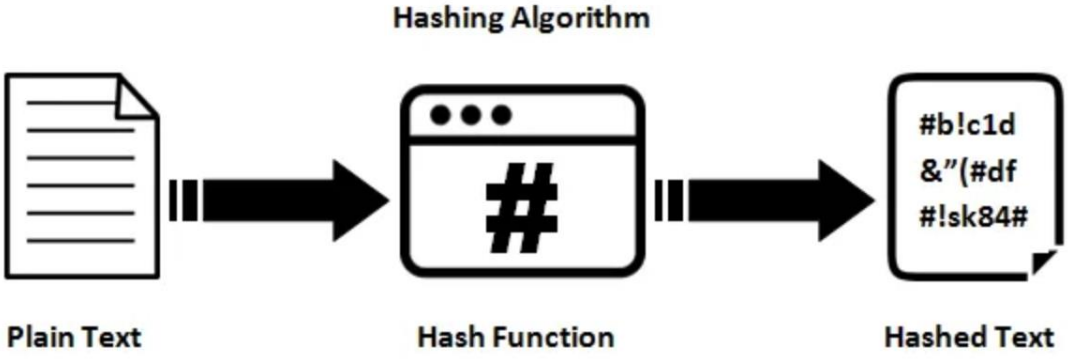

Figure 3 - How a hash algorithm (SHA-256) works (Anand, 2020)

- **Asymmetric Cryptography / Public Key Cryptography**

A blockchain typically, takes advantage of public key encryption to secure transactions between parties, verifying transactions through digital signatures. Each public key is accompanied by a private key while, the encryption and decryption of data is carried out using both. Thus, when the algorithm responsible for the signature process is secure, it is undeniable that only the person with that private key could have created that signature (Fernandez-Carames and Fraga-Lamas, 2020). This form of cryptography creates a trust factor between users, providing a mechanism that can validate the integrity as well as the authenticity of public transactions.

Public key encryption is also essential for the operation of the "wallets". In a blockchain each user has a wallet associated with at least one public address (usually the hash of their public key) and a private key that the user needs to sign transactions (Fernandez-Carames and Fraga-Lamas, 2020).

- **Transactions**

A transaction is defined as an interaction between two individuals, organizations, etc. For example, the transfer of cryptocurrency between Bitcoin holders represents a transaction (Kahate, 2017). Each block may contain more than one or even zero transactions. In several blockchains, continuous creation of new blocks is required, regardless of whether they contain zero transactions, to ensure that the network is protected from any tampering that any malicious users may cause.

- **Distributed Ledger**

The distributed ledger technology refers to a new and rapidly evolving approach to capturing and sharing data across multiple repositories (ledgers), each of which has exactly the same files and is maintained and controlled collectively by a distributed network of server-computers called "nodes" (World Bank, 2017) (Figure 4). Blockchain technology, a particular form of distributed ledger technology, uses cryptographic and algorithmic methods to create and verify an ever-expanding data structure, which takes the form of a chain and acts as a ledger.

The process of adding a record to the blockchain is initiated by one of the members (nodes), which creates a new block of data, containing the history of certain

transactions. Then, the information about this new data block is communicated to the entire network, containing encrypted data so that the transaction details are not made public, and then, the validity of the block is determined by the participants who have the corresponding rights, according to the predefined consent mechanism. Only after validation, the set of participants adds the new block to their respective ledgers. Through this mechanism, any change to the ledger is replicated across the entire network and each network member has a complete, identical copy of the entire ledger at all times (World Bank, 2017).

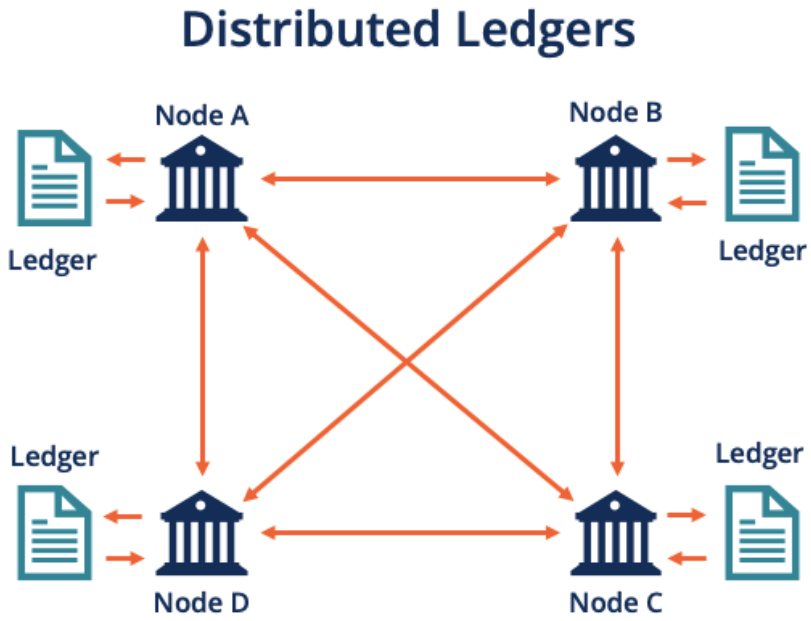

Figure 4 - Representation of a distributed ledger (Majumder, 2022)

- **Consensus Mechanism**

A node is a computing entity capable of performing operations on the blockchain. It is common to distinguish between "normal" nodes of a blockchain, which only interact with the blockchain by executing transactions, and "full" nodes, which maintain a copy of the blockchain and contribute to it by validating transactions. A "miner" is a third type of node present in many blockchains, which, following the respective consensus protocol, contributes to the validation of transactions (Wang et al, 2019). As blockchain technology evolves, a plethora of consensus mechanisms, each aiming at a different purpose, with some of the most popular ones being "Proof-of-Work (PoW)", "Proof-of-Stake (PoS)" and "Proof-of-Authority (PoA)":

- **Proof-of-Work (PoW): I**t is the first consensus mechanism implemented in a blockchain network. In the "PoW" protocol, miners compete within the network to solve complex computational puzzles. When one miner finds the solution to the puzzle, he will transmit the newly created block to the network and then all other miners will check the solution and verify if it is correct (Castor, 2017).
- **Proof-of-Stake (PoS):** Commonly used instead of the Proof-of-Work protocol. In this protocol, the party concerned can validate the block according to the number of cryptocurrencies it holds. The more cryptocurrencies a validator or a miner has, the more mining "power" and therefore the more chances he has to create the block (Castor, 2017).
- **Proof-of-Authority (PoA):** Is a consensus algorithm that provides an efficient solution for private blockchains. The term was coined in 2017 by Gavin Wood, co-founder of the blockchain "Ethereum". The operation of the algorithm is based on trust in the identity of the validator. In "Proof-of-Authority", nodes gain the right to create new blocks by passing a strict verification process, submitted by the system moderators. These moderators are pre-approved participants who check the blocks and transactions. As a result, "PoA" blockchains are only protected by trusted nodes validation (Antolin, 2022).

In general, a blockchain can be permissionless or permissioned, with fundamental differences existing between the two. Bitcoin and Ethereum are the most prominent examples of permissionless blockchains, where participants in the network can join or leave the network at will, without being pre-approved or controlled by any entity. All that is needed to join the network and add transactions to the ledger is a computer with the appropriate software. There is no central owner, and identical copies of the ledger are distributed to all participants in the network (World Bank, 2017).

In permissioned blockchains, members are pre-selected by the blockchain owner or administrator, who controls access to the network and sets its rules. This resolves several concerns that governments and regulators have about permissionless blockchains, such as authenticating network members and recording the legal ownership of the ledger (World Bank, 2017). On the counterpart, negates a key advantage of permissionless blockchains: the ability to operate without the need for any single entity to play a coordinating role, which necessarily requires all participants to trust that entity.

Permissioned blockchains, in which network access is regulated, usually do not require the energy-intensive Proof-of-Work consensus algorithm to verify transactions but rely on different protocols to establish consensus between members, such as Proof-of-Authority. In permissionless blockchains, which do not regulate access to the network, there is no trust requirement between participants and, therefore, a complex Proof-or-Work algorithm is used to create consensus between users (World Bank, 2017).

In fact, blockchains are divided into four (4) categories: public, permissioned, private and consortium. These categories can be divided into "Public / Private" (in terms of access) and "Permissioned / Permissionless" (in terms of roles) blockchains. Ripple, for example, is a permissioned blockchain, but the data is validated by all network participants, therefore their system can be classified as "Public Permissioned" (World Bank, 2017). On the other hand, a permissioned blockchain, where data is only validated by a set of participants, is considered "Private permissioned". Finally, a consortium blockchain is a type of blockchain that combines the characteristics of public and private blockchains. It is typically used within companies or business groups that share the same goal or set of goals related to the use of blockchain technology and collaborate to manage a common blockchain network (Gondek, n.d.).

2.2. Ethereum Blockchain

Ethereum, announced in 2014 and released in 2015, aims to create a universal blockchain-based application platform. It incorporates a "Turing-Complete" programming language (a complete language that allows programmers to solve any computational problem), enabling users to "write" smart contracts and decentralized applications where they define their own rules regarding ownership (Buterin, 2013).

Ethereum is an open-source, decentralized computing infrastructure, often described as "the global computer" (Antonopoulos and Wood, 2018), on which programs called "smart contracts" are executed. It uses a native cryptocurrency called "Ether" as a unit of measurement and cost-constrained resource for executing smart contracts in a virtual machine called "Ethereum Virtual Machine" (EVM) (Tikhomirov, 2018) (Figure 5). Furthermore, it allows developers to develop decentralized applications with built-in financial functions.

Ethereum's currency unit, the ether, is also identified as “ETH” or by the symbols “X” and “♦” (e.g. 1 Ether or 1 ETH or X1 or ♦1). The ether is subdivided into smaller units, the smallest of which is called “wei”. One ether corresponds to 10^{18} or 1,000,000,000,000,000,000 wei (Antonopoulos and Wood, 2018).

Ethereum incorporates a pricing mechanism whereby each computational step in the “EVM” is priced in units of “gas” (Wood, 2014). The correspondence of a unit of gas to ether is determined by its market price. For each transaction, the sender specifies the maximum amount of gas the computational step is expected to consume (gas limit) and the price the sender wishes to pay per unit of gas (gas price). The final transaction amount is equal to the “gas limit” multiplied by the “gas price” at the given time.

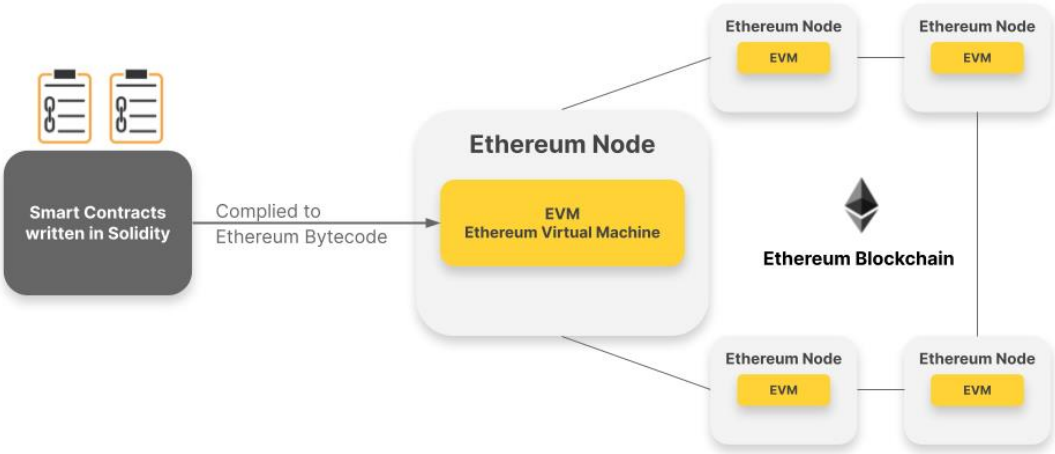

Figure 5 - The Ethereum Virtual Machine (EVM) (coin98.net, 2022)

2.3. Smart contracts and decentralized applications

The term smart contract has been used over the years to describe a variety of different things. In the 1990s, cryptographer Nick Szabo coined the term and defined it as “a set of promises, defined in digital form, including protocols, within which parties execute other promises” (Antonopoulos and Wood, 2018). In the context of Ethereum, the term is actually a bit of a misnomer, since smart contracts are neither smart nor legal contracts, but rather immutable computer programs that are executed deterministically within an “EVM” as part of the Ethereum network protocol (Antonopoulos and Wood, 2018). In other words, a smart contract is a piece of code stored on the blockchain that can be executed independently, with the purpose of automating certain tasks (Christidis and Devetsikiotis, 2016).

Smart contracts are often equated with software applications. Instead, they take more the form of classes in object-oriented programming. When developers talk about "smart contract development", they usually refer to the practice of writing code in the programming language "Solidity" (Figure 6), which will then be executed on the Ethereum network.

```
1   // SPDX-License-Identifier: MIT
2   pragma solidity ^0.8.19;
3
4   contract Migrations {
5     address public owner;
6     uint public last_completed_migration;
7
8     constructor() public {
9       owner = msg.sender;
10    }
```

Figure 6 - Example of code in "Solidity" programming language

Developers usually develop smart contracts in high-level languages aimed at "EVM", the most popular being "Solidity". Nominally, some of the other programming languages are "Serpent" (discontinued in 2017), "Vyper" and "LLL".

A major issue with the architecture of smart contracts is the inability to process their code once they are deployed on the blockchain. The only way to delete a smart contract is if it is programmed with an accessible "SELFDESTRUCT" "opcode" (machine language instruction for a specific operation), which will remove it completely from the blockchain.

A Decentralized Application (dApp) is an application that is based on a decentralized network and combines one or more smart contracts with a user interface (frontend) for its operation. The main difference with a centralized application is that the backend code of the latter runs on centralized servers (Ethereum.org, 2023).

There are many advantages to creating a dApp that a typical centralized architecture cannot offer (Antonopoulos and Wood, 2018):

- **Durability**

Because the business logic (the code) "behind" a decentralized application is controlled by a smart contract, the "backend" code of the dApp will be distributed and managed on a blockchain platform. Unlike an application deployed on a centralized

server, no downtime is foreseen for a dApp, resulting in it continuing to be available as long as the blockchain platform in question continues to operate.

- **Transparency**

The "on-chain" nature of a dApp allows everyone to inspect its code and be more confident in its operation. Any interaction with the dApp will be permanently stored on the blockchain.

- **Resistance to censorship**

As long as a user has access to an Ethereum node (running one if necessary), they will always be able to interact with a dApp without interference from any central authority. No service provider, or even the owner of the smart contract, can change the code once it is deployed on the network.

In a dApp, smart contracts are used to store the business logic of the application. In this way, in simple terms, a smart contract works identically to a server-side component of a regular application. One of their main differences is that any computation performed on a smart contract is characterized by high cost, and so it is important to keep it as simple as possible. Therefore, it is important to determine which aspects of the application need a reliable and decentralized execution platform, and which do not, so that appropriate steps can be taken when writing the code.

In contrast to the business logic of dApps, which requires a programmer to understand how the "EVM" works, but also new programming languages such as "Solidity", for the user interface of a dApp standard web technologies can be used (HTML, CSS, JavaScript etc.) and libraries, such as "Web3.js", to connect the "frontend" of the application, via a " JavaScript API ", to its "backend" - the blockchain. This allows a "traditional" web developer to use familiar tools, libraries and frameworks. Interactions with Ethereum, such as signing messages, sending transactions and key management, are often done through the web browser, with the help of a "wallet" extension, such as "Metamask ".

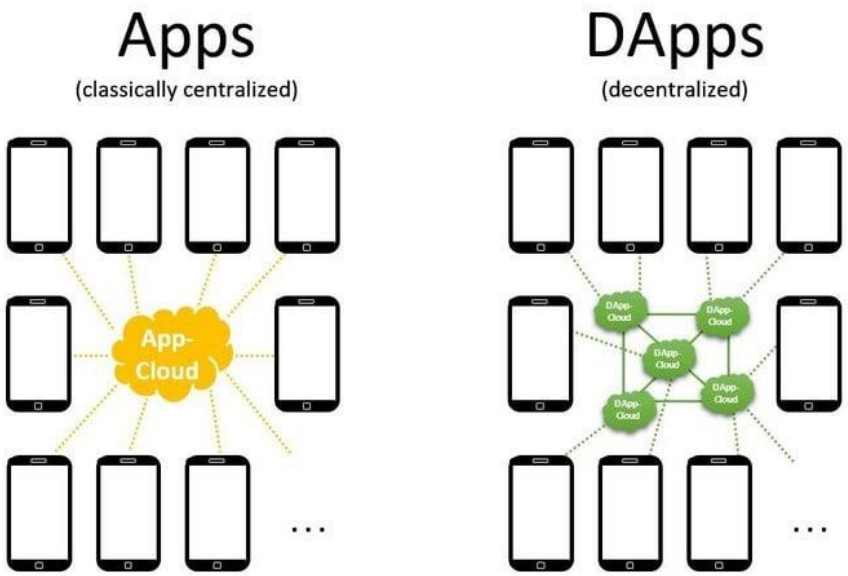

Figure 7 - Comparison of centralized and decentralized applications (Goldmann, 2019)

Chapter 3: Digital forensics

Digital forensics represents a growing field of information technology, which falls within the broader context of information security. In particular, it is related to the response to security incidents through common practices and tools that enable the analysis and interpretation of digital evidence (Mavridis, 2015). It is the practice of collecting, analyzing and reporting digital data in a legally acceptable manner through specialized recovery, authentication and analysis techniques, when a case involves issues related to the representation of computer use, the examination of data residues and the identification of data through technical analysis

Similarly to many other forensic sciences, digital forensics encompasses the use of sophisticated technological tools and procedures that must be followed to ensure the integrity of evidence and the accuracy of results related to computer processing (Prasad and Pandey, 2016). The digital forensics investigation process usually follows predefined processes to extract information and write a structured evidence report (Khan et al., 2021):

- **Collection:** Acquisition of digital content related to the case under investigation from various multimedia devices.
- **Identification:** Identify the ownership and potential sources of the most important information and data.
- **Examination and analysis:** Conduct a detailed systematic examination of evidence related to the crime and draw substantive conclusions.
- **Report:** Writing a comprehensive report based on the evidence examined, using appropriate forensic techniques. The resulting report is presented to the court as substantive evidence against the case under investigation.
- **Conservation:** Electronic preservation of collected evidence, protecting the media devices, logs associated with the device and its users and the folder of all relevant evidence in a secure manner.

The primary goal of digital forensics is to meticulously collect the information without modifying the original content and to conduct thorough research to validate the

findings with concrete evidence that excludes any ambiguous interpretations. The collection and research procedures may include methods such as (Mavridis, 2015):

- Extensive research to collect data from the information system and from a variety of online resources.
- Retrieve modified or deleted data.
- Recovering passwords of unauthorized users, in order to carry out or hide their activities.
- Documentation of the actions and operations of unauthorized users.
- Detailed examination of collected evidence.
- Compilation of a detailed analysis of the results of the research process, with a view to using it as the basis of the findings that addresses to the relevant expert.

3.1. History of forensics

Determining the exact conduct of the first digital forensics survey faces difficulties. However, a consensus among experts acknowledges that the field began its development about three decades ago (Wheelbarger, 2009). Its origins can be traced to the United States, particularly when law enforcement and military investigators noticed an increase in technologically skilled criminals. Government personnel tasked with safeguarding sensitive, confidential and classified data conducted forensic studies in response to potential security breaches, not only to investigate a particular breach but also to gather information to prevent possible future breaches (Wheelbarger, 2009). Over time, the fields of information security and digital forensics gradually began to intertwine. The major events in the history of forensics are summarized in the table below (Prasad and Pandey, 2016):

Year	Event
1835	Henry Goddard of Scotland Yard became the first person to use physical analysis to link a bullet to the murder weapon.
1836	James Marsh developed a chemical test to detect arsenic, which was used during a murder trial.
1892	Sir Francis Galton established the first system of designating fingerprints as confidential data.

1920	American doctor Calvin Goddard created the "comparison microscope" to help determine which bullets came from which cartridges.
1930	Karl Landsteiner won the Nobel Prize for his classification of human blood into its various groups.
1970	Aerospace Corporation in California has developed a method for detecting gunshot residue using scanning electron microscopes
1984	The "Magnetic Media" program of the "FBI" was created, later renamed the "Computer Analysis and Response Team" (CART) and is believed to be the forerunner of digital forensics.
1988	The International Association of Computer Investigative Specialists (IACIS) was founded.
1995	The "International Organization on Computer Evidence" (IOCE) was established.
1997	The G8 members stated that "law enforcement personnel must be trained and equipped to deal with high-tech crimes".
1998	• The G8 commissioned the IICE to establish international principles, guidelines and procedures on digital evidence. • The 1st Symposium on Forensic Sciences of "INTERPOL" took place.
2000	The FBI's first Regional Digital Forensics Laboratory was established.

Table 1 - Major events in the history of forensics (Prasad and Pandey, 2016)

The field of digital forensic science is one of the many branches of forensic science, mainly applied in conjunction with civil trials or criminal investigations. Gradually, the field has undergone significant growth over the past few decades, continuing to evolve to the present day. Both government agencies and private companies have adopted this path, involving in-house information security and digital forensics experts or contracting with specialized professionals or companies when necessary. In particular, the private legal sector has also recognized the importance of digital forensics analyses in civil litigation, noting a notable increase in interest in the field of digital investigations.

3.2. Forms and aim of digital forensics

Digital forensics is typically associated with criminal investigation processes, often focusing on various forms of digital crime. These types of crimes are usually divided, into two distinct categories: digital crime and crime involving one or more computers (Prasad and Pandey, 2016):

- **Digital crime:** The term refers to illegal actions carried out exclusively through computers, such as cyberbullying or spam (spamming). In addition to the newly emerging crimes facilitated by the digital age, it also includes conventional crimes carried out exclusively on computers, exemplified by cases such as child pornography.
- **Crime involving one or more computers:** This is a crime that takes place in the "real world" but is facilitated using computers. A classic example of this type of crime is fraud, where computers often serve as a means of communication between fraudsters, recording (or planning) activities or creating false documents.

Computers can be a "crime scene" in activities such as hacking or denial of service (DoS) attacks or by containing critical evidence such as emails, internet history, documents or other relevant files linked to crimes such as murder, kidnapping, fraud and drug trafficking. An investigator is not only interested in the content of emails, documents and records, but also in the metadata associated with them. Through digital forensic analysis, details such as the first appearance of the document on a device, the most recent date it was processed, the most recent date it was saved or printed, and the user who performed these actions can be revealed. In a broader sense, the goal of digital forensics is to provide guidelines for (Prasad and Pandey, 2016):

- Adherence to the initial response protocol and access to the victim's computer after the incident.
- Configuring protocols at a suspected crime scene to ensure that digital evidence obtained is not corrupted.
- Data acquisition and copying.
- Retrieval of deleted files and "partitions" from digital media for the extraction and validation of evidence.
- Analyzing digital media to preserve evidence, analyzing logs, drawing conclusions, investigating network traffic and logs to establish correlations with the event, examining cyber-attacks and monitoring emails to investigate related crimes.

- Writing a comprehensive forensic report detailing the entire investigation process.
- Preservation of evidence by maintaining a chain of custody.
- Implement rigorous procedures to ensure that the results of forensic investigations will stand up to the rigorous scrutiny of the court.
- Presentation of the results of forensic investigations to the court.

Chapter 4: Blockchain in Digital Forensics

Many forensic experts are increasingly using blockchain technology, taking advantage of its decentralized and distributed architecture, which enhances resilience against a range of malicious attacks, usually targeting centralized systems (Xiong and Du, 2019). By also leveraging encryption and hash functions, combined with the integration of intrusion detection and prevention systems, firewalls, and anti-disclosure tools and policies, the defense of decentralized nodes is significantly increased. In digital forensics, blockchain can also play a central role, encrypting and storing evidence and case-related transactions in an immutable ledger, facilitating the conduct of forensic investigations.

Blockchain technology has been widely adopted and used in an extensive range of security applications, offering immutable distributed ledger solutions for data security and protection. This is achieved by the property that each block carries the encrypted hash value of the previous in sequence, stored block. This allows the storage of hashed encrypted data, thus ensuring its integrity and preventing unauthorized interference, as any modification must be agreed, signed and approved by all authorized participants in a permissioned blockchain network, using predefined consensus algorithms (Ahmad et al., 2020). Moreover, the business logic (code) is governed and efficiently regulated through smart contracts, implementing a decentralized digital forensics research application. This helps to achieve secure, transparent and immutable digital investigations, making them resistant to forgery and counterfeiting.

Blockchain technology shows a remarkable correlation with the “CIA” security triad (Confidentiality, Integrity, and Availability) (Gupta, 2018). The decentralized and encrypted nature of blockchain ensures the confidentiality of sensitive data by providing access only to authorized parties through cryptographic keys. The “one-way” property of the hash function ensures that retrieving data from the hash result is logically impossible. Moreover, predicting the original data from the hash result proves difficult, as even small changes in the source text lead to significant differences. The distribution feature of the network guarantees availability by eliminating single points of failure and outages. Even if one node is disabled, the information remains accessible to the others, as they all keep an exact copy of the ledger, ensuring continuous updates. The integration of the above security concepts (CIA) into the core of blockchain

technology makes it an extremely favorable option for maintaining and tracking a chain of custody of digital forensics.

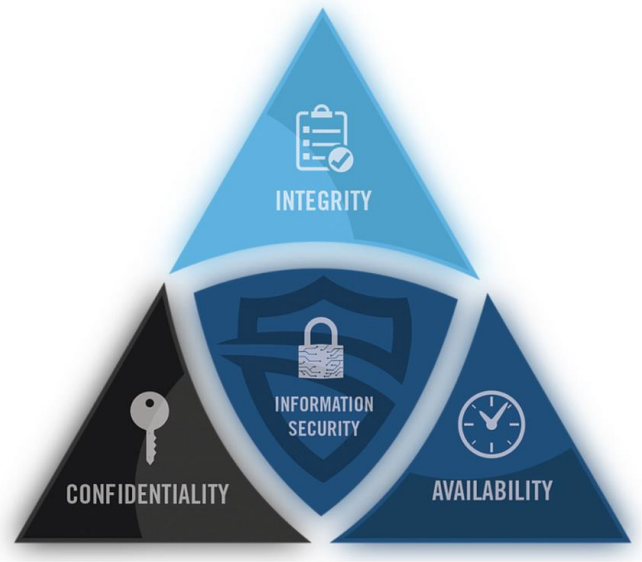

Figure 8 - The "CIA" security triad (Oliveira et al., 2020)

4.1. Understanding the synergy of private Blockchain and Chain of Custody applications

Digital forensic investigations involve a systematic and methodological approach to identifying, preserving, collecting, analyzing and presenting digital evidence in a manner acceptable to a court of law. A prerequisite for achieving this objective is the establishment and maintenance of a chain of custody throughout the investigation process. The chain of custody is a process of recording, documenting and maintaining the chronological order and complete history of handling and preserving the digital evidence for the case under investigation so that it is admissible in court (Khan et al., 2021). Safeguarding the integrity of the chain of custody against any alterations or tampering is of paramount importance. Its ultimate objective is to prove that the evidence collected is true, factual and relevant to the crime under investigation.

Evidence obtained from professionals is considered reliable by the courts. However, in cases of doubt, a more extensive examination is carried out to verify the authenticity and integrity of the reports. The hash value of the digital files, the location of the crime scene and the names of the investigators are not sufficient for the evidence to be accepted by the courts. Additional data are now required, such as the digital signature of each item, the exact processing locations of each digital evidence item, the

identification of all individuals involved in the forensic investigation and the individuals who maintained access to the evidence, as well as a log of all transactions conducted (Lone and Mir, 2019). Furthermore, forensic investigators are heavily dependent on automated forensic tools, therefore the reliability of the investigation results depends on the accuracy of these tools.

Without the implementation of a secure tampering prediction mechanism, there is a potential risk that digital evidence and the associated hash value could be tampered with by malicious individuals. The information controller can only verify that the evidence remains unchanged from the moment the hash value was calculated. By making use of blockchain technology, since it consists of consecutive blocks, which are linked together in chronological order, any tampering attempt becomes obvious, as the data stored on it cannot be modified or deleted.

The public blockchains are decentralized, allowing anyone to access them and the stored data. In contrast, a private blockchain allows access exclusively to trusted entities, making it ideal for use in criminal cases and other situations involving sensitive information. An additional level of security can be achieved by periodically logging the state of the private blockchain to a public one, ensuring the integrity of the ledger itself (Lu, 2019). The data registered on a public blockchain cannot be modified, as mentioned above, as it is decentralized and therefore all participants can verify its authenticity. Thus, public blockchains can perform a remarkable role in cases where a more robust chain of custody is required.

In a “traditional” chain of custody, the only option regarding the preservation of evidence is to store it in a centralized location, increasing the risk of a data breach due to the security gaps that exist in its handling. Such an arrangement lacks robust measures to counter cyber-attacks, making it vulnerable to tampering or forgery of evidence (Khan et al., 2021). The main issue facing methods of maintaining a chain of custody is the documentation and proper recording of stakeholder interactions with evidence. Furthermore, when multiple parties have access to the evidence, there is a risk of tampering with it.

By exploiting the capabilities of a blockchain, especially the hash function and encryption, it is ensured that once evidence is hashed and stored in a block, tampering becomes impossible, streamlining the verification process as only authorized parties are allowed to interact and make changes to the evidence. It also provides transparency to all participants, who can verify transactions through their public addresses (Lone and

Mir, 2019). All the above lead to the conclusion that blockchain technology becomes an ideal solution for maintaining the integrity and immutability of data, which is a paramount challenge when implementing a chain of custody.

4.2. Literature Review

The number of publications that have emerged in recent years, regarding the benefits of blockchain technology in digital forensics and more specifically, in improving the chain of custody process, highlights the breadth of research and the commitment of the research community to explore this area. As technology continues to develop and blockchain developments proliferate, it is expected that literature will continue to expand, providing further insights and innovations in this ever-evolving field.

This literature review aims to investigate existing research, and studies related to private blockchain networks and their relation to chain of custody applications. By examining the current state of research, this review will provide a comprehensive understanding of the topic and identify the research gaps that this thesis aims to address.

Private blockchain networks have proven to be a promising solution for organizations seeking improved security and privacy. The study by Xu et al. (2017) highlights the advantages of private blockchains, such as increased transaction speed, scalability, and oversight of network participants. These networks offer controlled access, allowing organizations to maintain confidentiality while leveraging the immutable and transparent nature of the blockchain.

The integration of private blockchain networks into chain of custody applications has attracted a lot of interest as a means of enhancing the monitoring and security of assets. Liu et al. (2020) propose frameworks for integrating blockchain technology into supply chain and asset management systems. These studies highlight the benefits of using a private blockchain network to enhance the transparency and verifiability of a chain of custody. Lone et al. (2019) advocate a blockchain-based chain of custody for digital forensics purposes to provide integrity and resilience against breaches of digital evidence. Their operating model includes four actions regarding evidence: creation, transfer, deletion and display of evidence. Jeong et al. (2020) present a solution for designing and implementing a permissioned forensic investigation model using “Hyperledger Fabric” to promote the reliability and authenticity of data in forensic cases.

The body of literature on the benefits of blockchain technology in digital forensics, and more specifically, in improving the chain of custody process, offers valuable insights into how blockchain technology can enhance the integrity and security of digital forensic evidence. Research in this area demonstrates the potential of blockchain in creating an immutable and transparent chain of custody, thereby addressing the challenges associated with evidence tampering and data manipulation in digital forensic investigations.

It is important to note that the use of blockchain in the chain of custody process can bring significant value to society. By providing an immutable record of all interactions with digital evidence, blockchain technology ensures its integrity, authenticity and traceability throughout its lifecycle. This enhances trust in the criminal justice system and the admissibility of digital evidence in court, ultimately leading to more reliable and fairer outcomes in legal proceedings.

Despite the growing interest in blockchain networks and chain of custody applications, existing research focusing specifically on the development of a private blockchain network and a corresponding chain of custody application is incomplete. Although existing studies offer valuable insights, they often lack comprehensive exploration of the technical implementation and practical implications of such a system. This research gap provides a clear motivation for this thesis, whose goal is to fill this by developing a private blockchain network and a user-friendly chain of custody application.

Chapter 5: Methodology

For the successful fulfillment of this thesis, a qualitative research design was applied that includes a thorough literature review in order to create a strong knowledge base in the areas of private blockchain, chain of custody applications and digital forensics. Then, taking advantage of the principles of the "Agile" methodology, the process of gradual implementation of the practical part of the thesis was applied, starting from the design and development of the private Ethereum Blockchain. Then, the decentralized chain of custody application was designed and developed, which was then integrated into the private blockchain infrastructure.

To address the complex problems that arose during the application development process, the management framework "Scrum" was used, which includes a set of tools and roles aimed at facilitating the organization and management of projects that implement the "Agile" methodology (Scrum.org, n.d.). The diagrams, reports and tables that follow were created using the "Visual Paradigm" application:

Product Owner Report

Member	Christos Bandis
Responsibilities	• Defining the vision of the project. • Configuration of epics (defined requirements). • Planning, defining and prioritizing user stories (the sub-tasks generated by epics). • Creating and updating the release plan. • Monitoring of the prioritized product backlog.

Table 2 - Project Owner Report

Scrum Master Report

Member	Christos Bandis
Responsibilities	• Supports the configuration of epics. • Coordinate the preparation of the traffic plan. • Contributes to the maintenance of the recorded deterrents. • Ensures that issues affecting development are identified and resolved.

Table 3 - Scrum Master Report

Scrum Team Report

Member	Christos Bandis
Responsibilities	• Commitment that user stories will be done within a sprint (recurring fixed time frame). • Identify risks and implement mitigation actions to reduce potential errors. • Full development of the product or service. • Development of the private Blockchain network. • Writing the code of smart contracts. • Design and development of the user interface of the application. • Continuous testing of the functions of the application.

Table 4 - Scrum Team Report

Project Charter

A Project Charter represents a formal, usually short document that identifies the project in its entirety (Wrike, 2019).

1. Project Vision

The project envisions a future where the integration of private blockchain and chain of custody applications transforms the landscape of digital forensics. By seamlessly combining the security and transparency of the private Ethereum Blockchain with the meticulous monitoring capabilities offered by chain of custody applications, the project aims to create a new paradigm of trust in digital evidence management. This vision includes streamlined forensic processes and an increased level of trust in the reliability and authenticity of digital evidence presented in legal contexts.

2. Project Mission

Through the development of a private Ethereum Blockchain and a decentralized chain of custody application, the project aims to enhance the integrity, traceability and security of digital evidence throughout its lifecycle. By leveraging the immutability and transparency of blockchain technology, the project seeks to help promote strong digital forensic practices, enhancing the reliability of evidence in legal proceedings and strengthening procedural cooperation between law enforcement agencies.

3. Project Success Criteria

The success of the project will be evaluated based on a set of predefined criteria. These include the successful development of a private Ethereum Blockchain, and a decentralized Chain of Custody application. The achievements of the project will be evident through proven improvements in the integrity, transparency and security of digital evidence management. In addition, the successful adaptation of the developed solutions to the practical workflows of digital forensics professionals and law enforcement agencies will be a key indicator of success. Finally, the success of the project will be reflected in its contribution to promoting the reliability of digital evidence in legal proceedings, promoting collaborative efforts between stakeholders and enhancing the overall effectiveness of digital forensic practices.

Use Case Report

1. Use Case Diagram

The following Use Case Diagrams show, by way of example, the cases of creating (opening) a new case (Figure 9) and managing evidence (Figure 10).

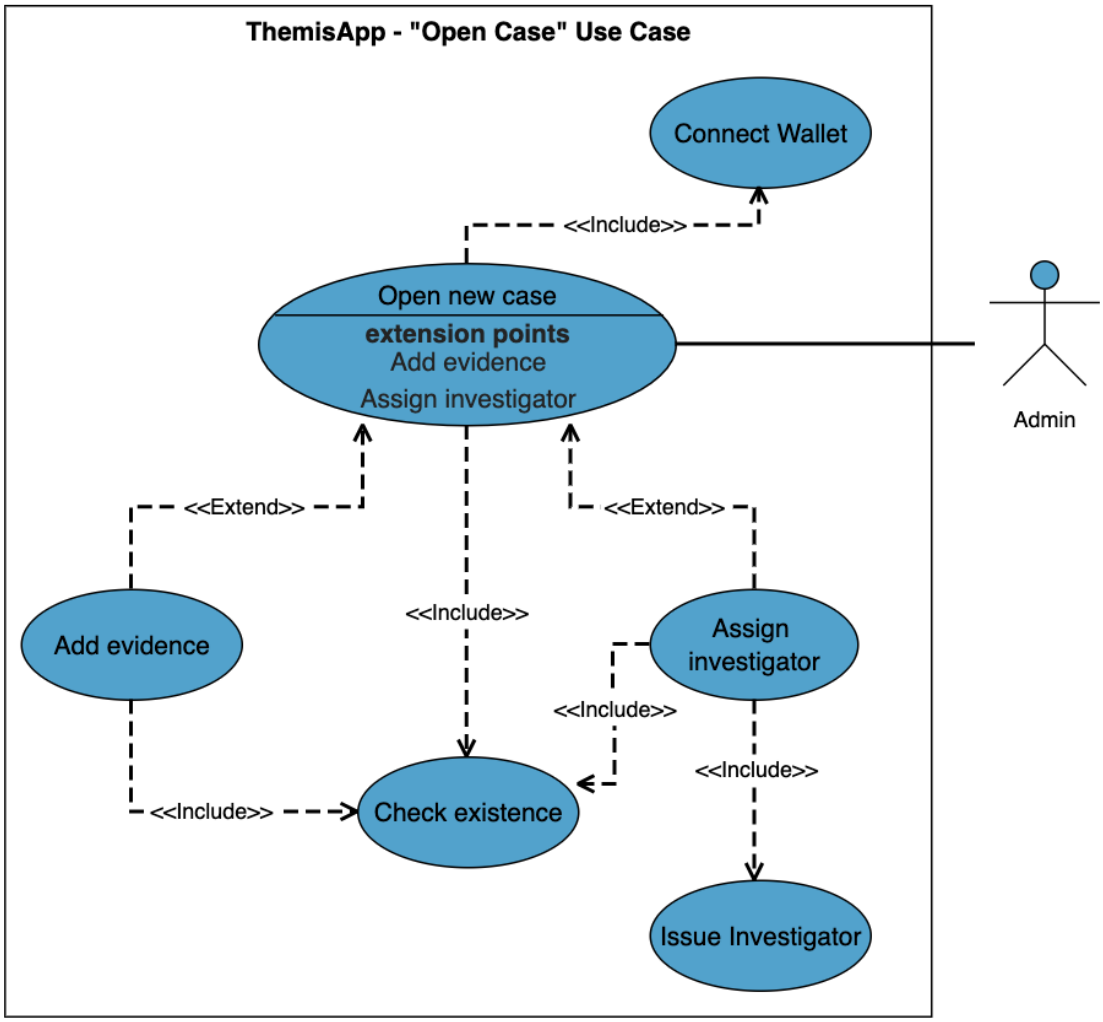

Figure 9 - Use case diagrams for creating (opening) a new case

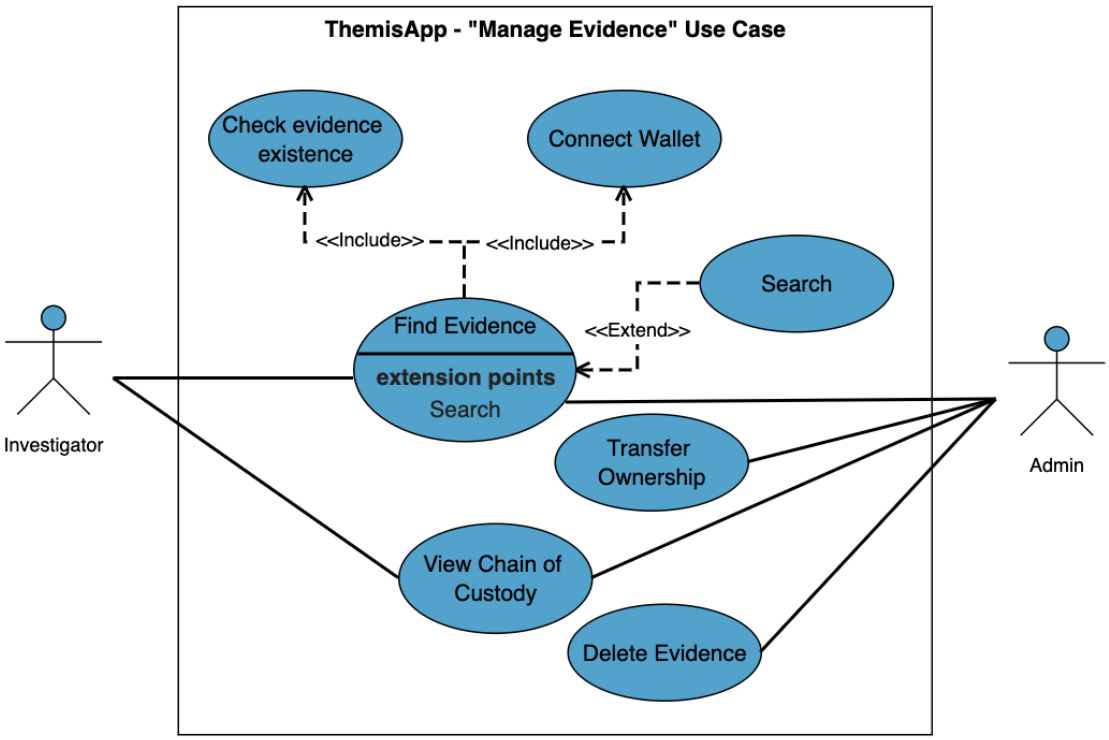

Figure 10 - Evidence management use case diagrams

2. Prioritized Use Cases

Use case prioritization is an approach through which organizations can identify potential use case scenarios, assess the business relevance arising from them and organize them in order of influence on their strategic objectives (Mishra, 2021). In the table below, the "Priority" column indicates the importance of the scenarios in delivering substantial and immediate business benefits. The "Size" column includes a subjective assessment of the resources required to support each scenario, while the "Complexity" column includes a subjective measure of the complexity associated with supporting each scenario.

Name	Description	Priority	Size	Complexity
Connect Wallet	The user connects to the app via his/her digital "wallet".	Must	Very Small	Low
Find Evidence	The user (investigator/admin)	Should	Medium	Medium

	manages the evidence they want.			
View Chain of Custody	The user (investigator/admin) gains access to the chain of custody of the evidence.	Could	Medium	Medium
Add evidence	The admin adds one or more pieces of evidence to a case.	Must	Medium	Medium
Transfer ownership	Admin transfers ownership of the evidence to another investigator.	Could	Small	Low
Close case	The admin closes and deletes a case.	Must	Medium	Low
Check evidence existence	The admin checks for the existence of a proof.	Must	Medium	Medium
Check existence	The admin checks if there is a registered investigator with the same address or if this investigator already exists in the case.	Must	Medium	Low
Remove investigator	The admin removes an investigator from the blockchain or from a case.	Must	Small	Medium
Delete evidence	Admin deletes evidence from a case.	Should	Medium	Medium
Issue investigator	The admin creates an identity for a new investigator.	Must	Medium	Low
Open new case	Admin opens a new case.	Must	Medium	Low

Assign case to investigator	The admin assigns the case to one or more investigators.	Must	Medium	Medium
Read case description	Show the description of a case.	Should	Small	Low
View active cases	Show the list of cases assigned to the connected investigator/admin.	Should	Medium	Low
Manage investigators	Display the list of registered investigators.	Should	Small	Low

Table 5 - Use Case Prioritization (Use Case Prioritization)

Epics Report

Epics represent an increased level of functionality or generally described requirements and can be broken down into more manageable units called "User Stories" (Sinha, 2023). In the table below, the "Priority" column indicates the degree of importance of epics in terms of delivering substantial and immediate business benefits, while the "Risk" column indicates the level of uncertainty regarding the successful completion of an epic.

Name	Description	Parent Use Case	Priority	Risk
Login to the application	Display a dialog box for the investigator or admin to log in via their "wallet" application.	Connect Wallet	Must	Low
Adding evidence to a case	Display a window to add evidence to a case.	Add Evidence	Must	High
View case description	Show case description window.	Read case description	Should	Low

Transfer of ownership of evidence	Display a transfer of ownership window for evidence.	Transfer ownership	Must	High
Case Closure/Registration	Display a case deletion confirmation window.	Close case	Should	Medium
Registration of a new investigator	Display a transaction window to register a new investigator.	Issue investigator	Must	High
Registration of a new case	Display a transaction window to register a new case.	Open new case	Must	High
Error display	Notify the user in case of an error during the execution of the application or blockchain transactions.	Show error	Should	Low
Show active cases	Show list of active cases of connected investigator/admin.	View active cases	Should	Medium
Show chain of custody	Show the chain of custody of an item of evidence.	View Chain of Custody	Must	High
Management of registered investigators	Display a list of registered investigators.	Manage investigators	Must	Medium
Management of evidence	Search for evidence based on a case.	Find evidence	Must	High
Deletion of evidence	Deletion of the evidence from the case.	Delete evidence	Should	Medium

Removing an investigator	Display a window to enter a investigator address to be removed from the blockchain or case.	Remove investigator	Must	Low
Assignment of a case to an investigator	Display a window to enter an investigator's address to assign the case.	Assign case to investigator	Should	Medium
Check for the existence of an investigator/evidence	Check if the investigator is registered in the blockchain or in the case / Check if the evidence already exists in the case.	Check existence	Must	Medium

Table 6 - Reference of "Epics"

Product Backlog

A Product Backlog is a dynamic, organized list that includes all the elements necessary to improve a project (Scrum.org, n.d.).

1. User Story Map

User Story Mapping is a simple design technique that describes the - expected by the "Scrum" team - steps that users follow to achieve their goals when using a digital product (Kaley, 2021).

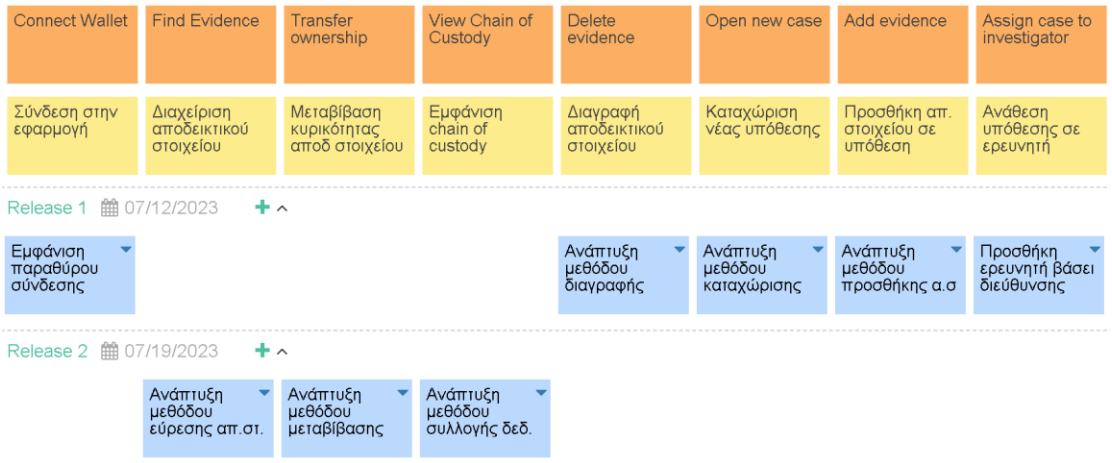

Figure 11 - User Story Map (1/2)

Figure 12 - User Story Map (2/2)

2. Prioritized User Stories

The hierarchy of User Stories follows the collective convention among the Scrum team members regarding the organization of the project features. This organization starts with the highest priority features, ensuring the rapid release of the project to market (CardBoard, 2020).

Each User Story is accompanied by relevant Acceptance Criteria, which serve as reference points for its fulfilment. These criteria provide clarity to the team about the expected outcomes of a user story, eliminating uncertainties from the requirements and helping to formulate clear expectations. Story points are numerical representations indicating the estimated “size” of a user story. The assessment of the size of a user story considers factors such as risk, effort required and level of complexity.

Name	Description	Epic	Status	Acceptance criteria	Story Points	Priority	Risk
Show login window	As a user, I want to be able to log in to the app directly and easily.	Login to the application	Approved	When visiting the app, you will the login window should automatically appear in the application.	1	Must	Low
Development of a method for finding evidence	As a user, I want to search for the evidence I am interested in, based on the case number to which it belongs.	Management of evidence	Approved	Develop a method to search by case number and display the evidence belonging to it.	3	Must	High
Developing a method for transferring ownership of evidence	As an admin, I wish to transfer ownership of one or more pieces of evidence to other investigators.	Transfer of ownership of evidence	Approved	Develop a method whereby ownership of one or more pieces of evidence is transferred using the addresses of the investigators.	4	Must	High
Developing a method for collecting evidence data	As a user, I wish to check the chain of custody of one or more pieces of evidence.	Show chain of custody	Approved	Development of a method to project the chain of custody of the requested evidence.	4	Must	High
Development of a method for the	As an admin, I wish to have the option to delete	Deletion of evidence	Approved	Developing a method to delete one or more pieces of	3	Must	Medium

deletion of evidence	one or more pieces of evidence.			evidence from a case.			
Development of a method for the registration of new Case	As an admin, I wish to be able to register a new case.	Registration of a new case	Approved	Development of a method for registering a new case on the blockchain.	4	Must	High
Development of a method for adding evidence	As an admin, I want to be able to add one or more pieces of evidence to a case.	Adding evidence to a case	Approved	Develop a method of adding one or more pieces of evidence to a case.	4	Must	High
Adding an investigator by address	As an admin, I want to be able to add an investigator to a case based on his/her wallet address.	Assignment of a case to an investigator	Approved	Show the option to add an investigator when managing a case.	3	Must	High
Search in memory of smart contract	As an admin, I like to check the existence of an investigator or evidence before taking any action.	Check for the existence of an investigator/evidence	Approved	Check the investigator's address or the hash of the evidence before any action is taken.	3	Must	Medium
Development of a method for registering new	As an admin, I wish to have the ability to add a new investigator	Registration of a new investigator	Approved	Development of a method to register a new investigator on the blockchain.	4	Must	High

investigator	to the blockchain.						
Development of a method for collecting active cases by investigator	As a user, I wish to check the active cases assigned to me.	Show active cases	Approved	Develop a method for displaying active cases by investigator.	3	Should	Medium
Show case description window	As a user, I wish to view the description of a case.	View case description	Approved	Display an option to view the description of a case.	2	Should	Low
Use of method alert of Javascript	As a user, I wish to be informed of any error that occurs while using the application or while performing any transaction with the blockchain.	Error display	Approved	Display a message for any kind of error that occurs.	1	Should	Low
Development of a case deletion method	As an admin, I wish to have the ability to delete any case.	Case Closure/Registration	Approved	Show the option to delete a case when managing cases.	3	Should	Medium
Development of an investigator abstraction method	As an admin, I wish to have the ability to remove an investigator from a case or	Removing an investigator	Approved	Display an option to remove an investigator when managing the available investigators or	3	Must	Medium

	from the blockchain.			when managing a case.			

Table 7 - User Stories Prioritization

Release Plan

A release plan provides the Scrum team with a comprehensive view of the planned releases of the project and the associated delivery schedule. This ensures compliance with the project owner's expectations and allows for more efficient development.

1. Project Deliverables

A deliverable refers to a tangible or intangible product or service intended for development as part of a project.

Deliverable	Description	planned release date	Priority	Status	Owner
Private Ethereum Blockchain and Decentralized Chain of Custody Application	Private blockchain and decentralized chain of custody model for digital forensics evidence management.	2023-06-25	High	Done	Christos Bandis

Table 8 - Project Deliverables

2. Release Configuration

Release	Description	planned release date
Release 1	Fully functional private Ethereum Blockchain, under the name “ThemisChain”, deployed on the cloud platform “AWS”.	2023-07-05

Release 2	Fully functional decentralized chain of custody application, based on the private network "ThemisChain", according to the model presented in the deliverable.	2023-07-12
Release 3	Enriching the application with new functions and renaming it to "ThemisApp".	2023-07-19

Table 9 - Project Releases

5.1. Modelling

Application modeling is a critical step in the software development lifecycle, serving as a bridge between conception and implementation. "UML" (Unified Modeling Language) diagrams provide a standardized and visual way of representing the various aspects of an application's structure, behavior, and interactions (Lucidchart, 2019). Through UML diagrams, developers can create a solid foundation for creating successful and user-friendly applications. Each type of UML diagram plays a unique role in ensuring an integrated and well-organized application development process.

1. Class Diagram

The following class diagram (Figure 13) shows the structure of the application - classes, the necessary variables and methods, as well as the relationship between them along with their completeness ratio. It should be noted that, a fundamental feature of the "Solidity" programming language, used for the development of smart contracts, is that it is not necessary to implement "getter" methods (a method that returns the value of a variable), as they are created automatically along with the creation of a variable.

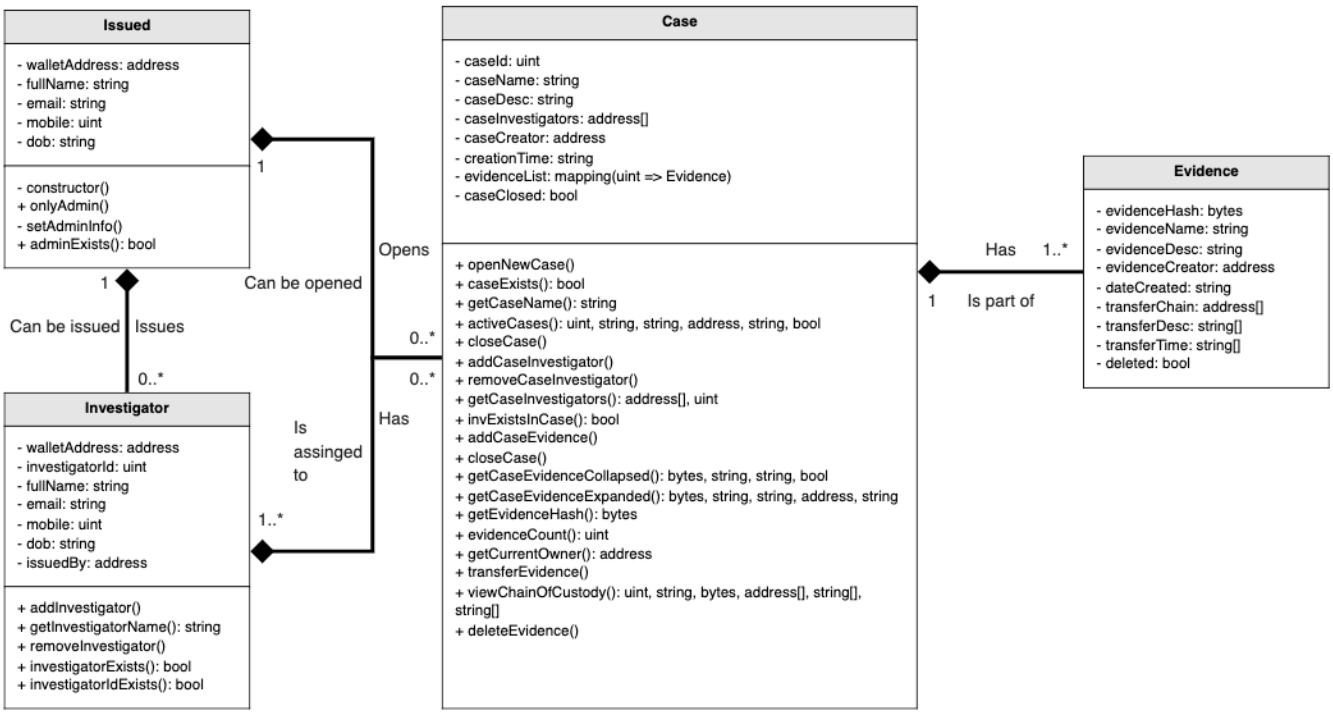

Figure 13 - Class Diagram

2. Object Diagram

Everything mentioned in the above paragraph is also visible in the object diagram in Figure 14, which incorporates a full use case with real data, verifying the integrity of the class diagram.

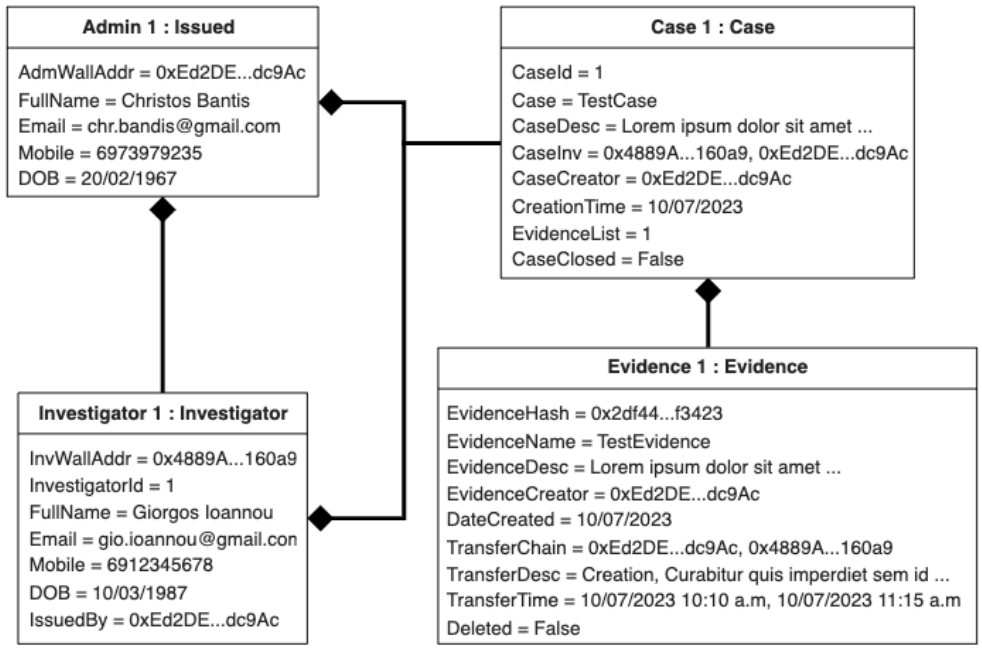

Figure 14 - Object Diagram

3. Flowcharts

The flowcharts that follow represent the simplicity of the sequence of steps and decisions taken in carrying out the processes they depict. More specifically, the flowchart in Figure 15 visualizes the flow of data and controls during the creation of a new case, while the flowcharts in Figures 16 and 17 relate to the registration of a new investigator and the addition of an evidence respectively.

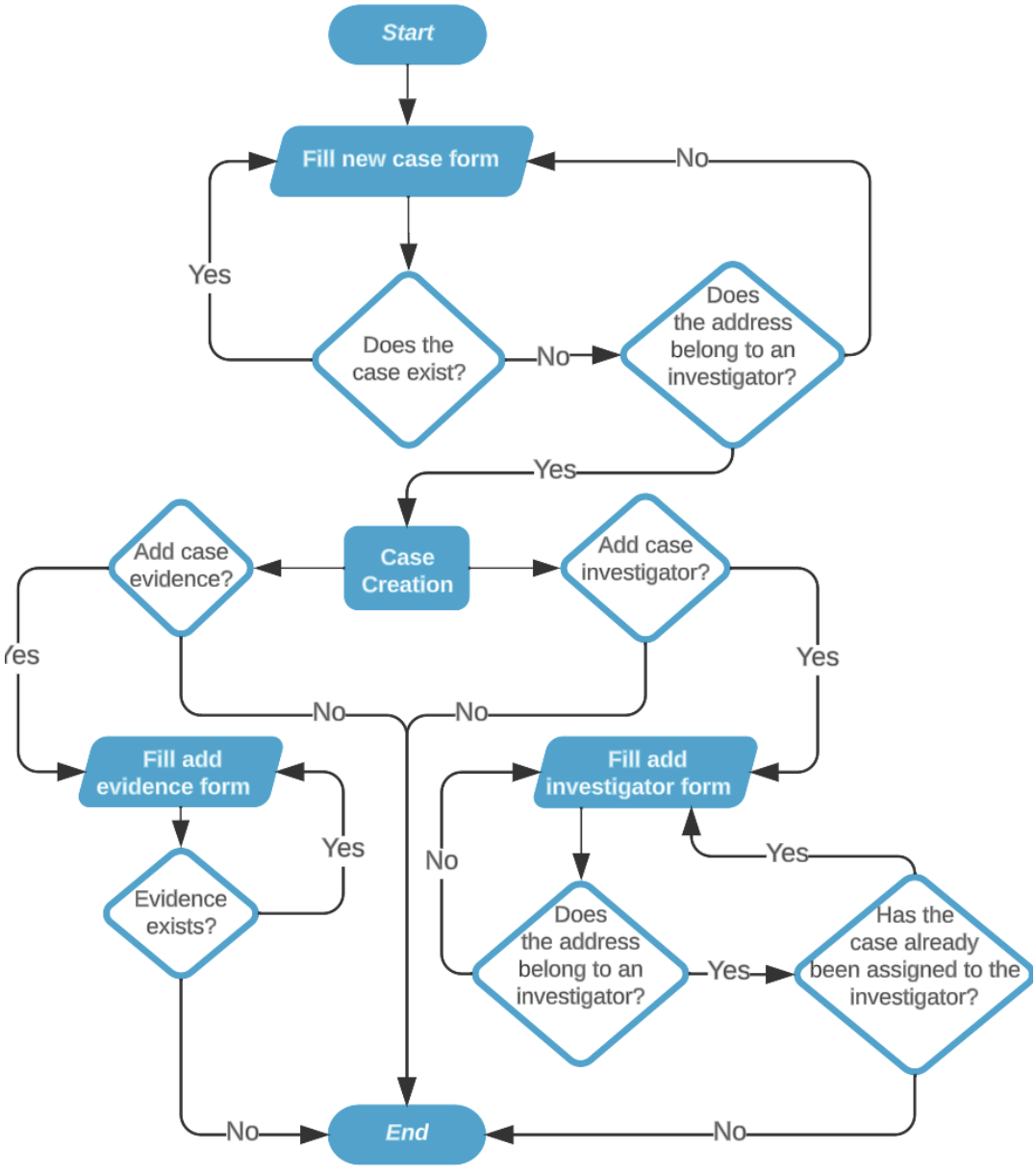

Figure 15 - Open new case Flowchart

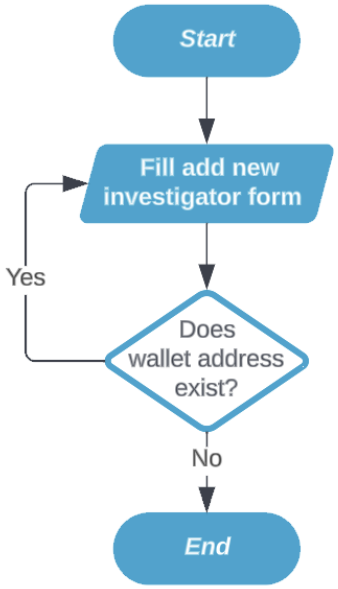

Figure 16 - Add new investigator Flowchart

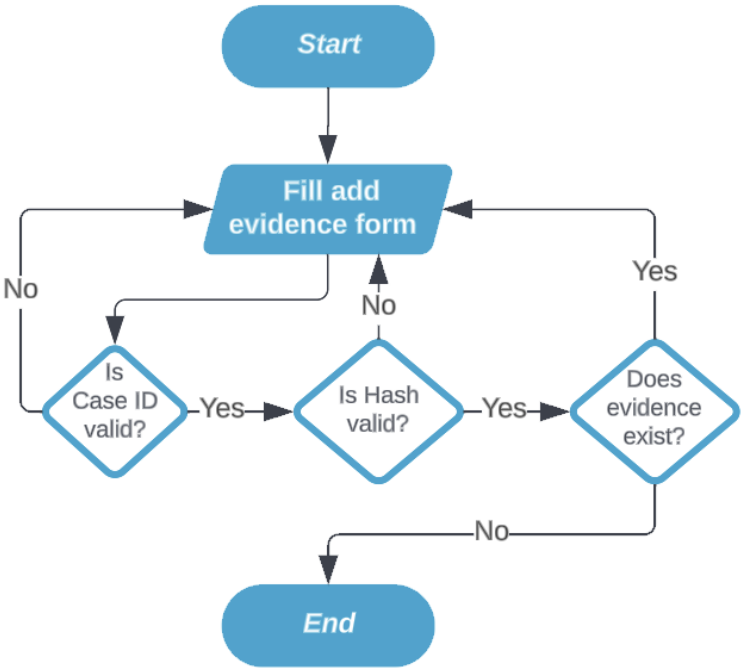

Figure 17 - Add evidence Flowchart

4. Activity Diagram (Activity Diagram)

The activity diagram in Figure 18 illustrates, by way of example, how the activities and actions performed during the process of managing the cases recorded in the blockchain are linked.

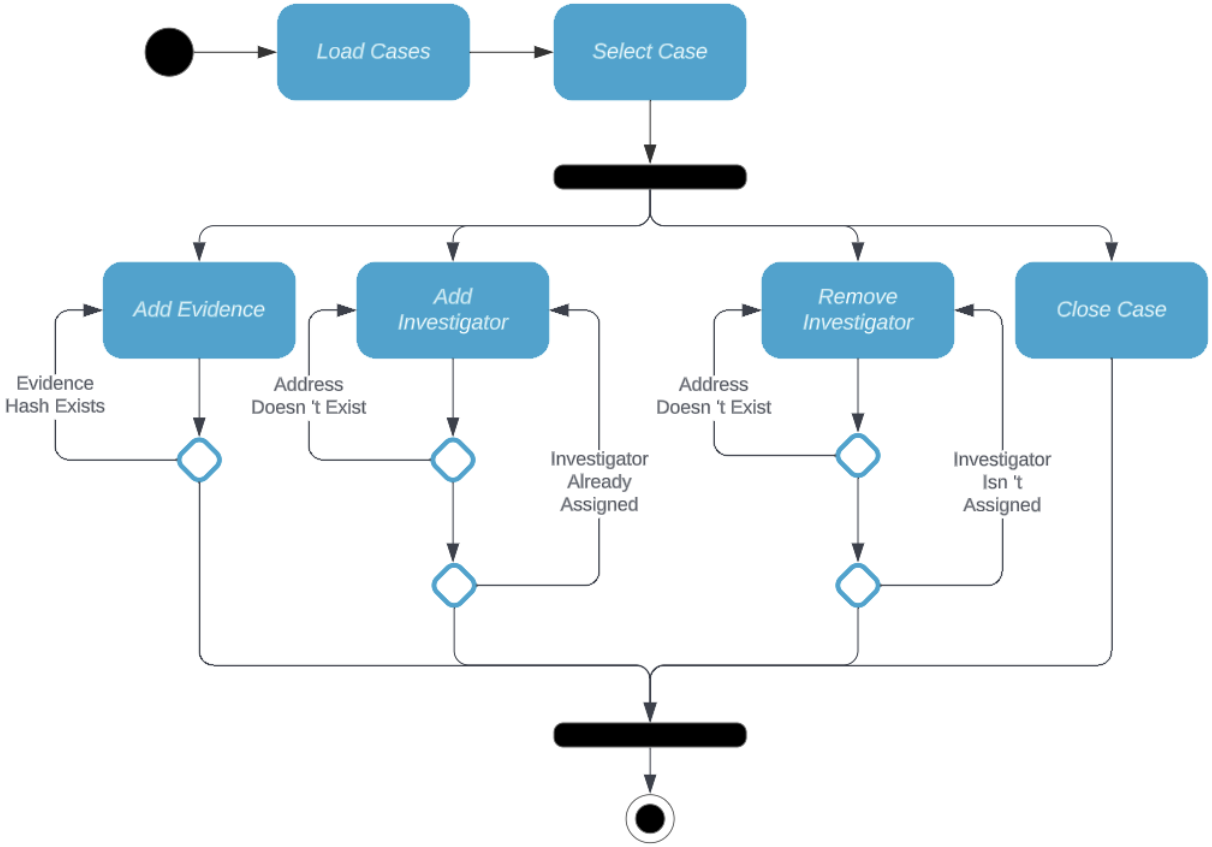

Figure 18 - Manage cases Activity Diagram

5. Sequence Diagram

The following sequence diagram (Figure 19) provides a visual representation of the process of viewing the active cases of the connected investigator and serves to understand the flow of messages and the interaction between the objects that form the background of the application. Similarly to the aforementioned process, the present diagram characterizes the sum of use cases of the application.

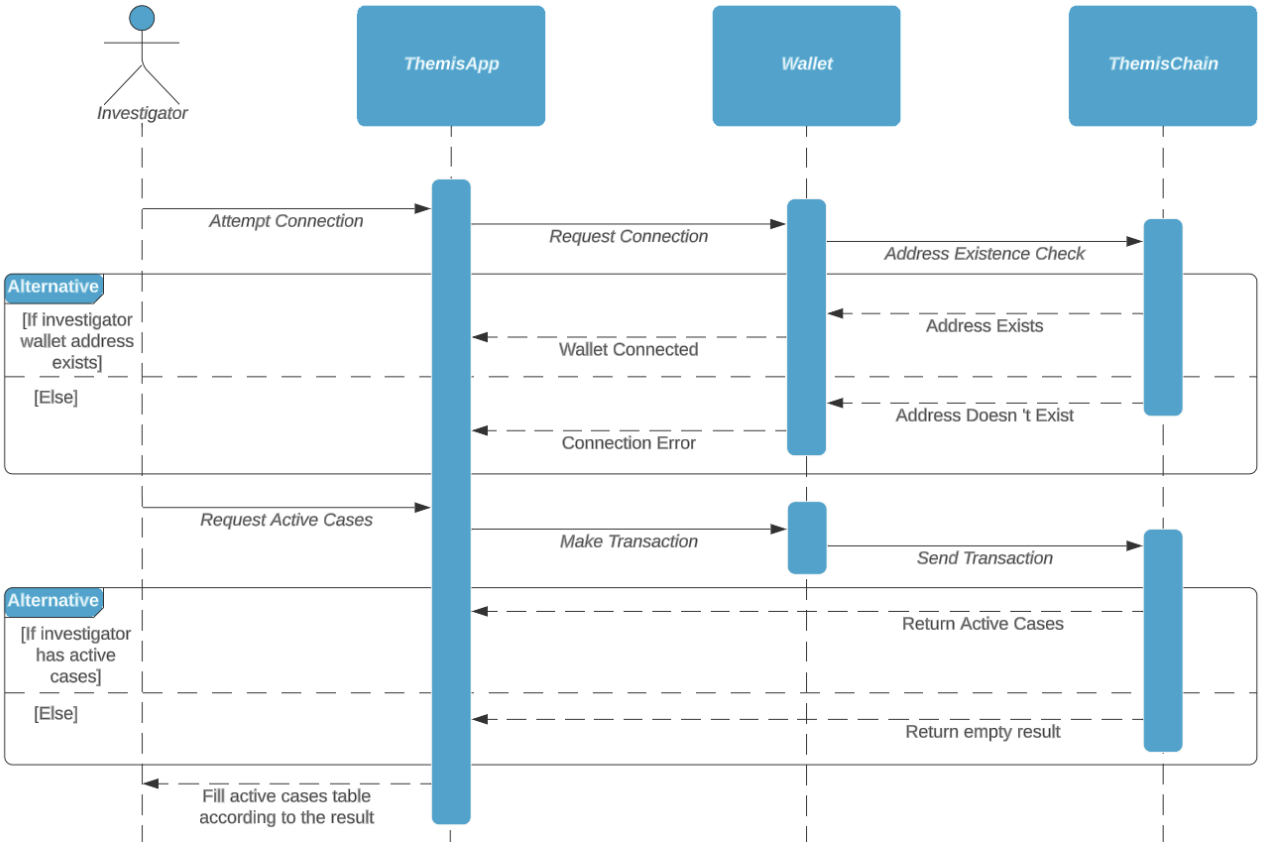

Figure 19 - Active cases Sequence Diagram

5.2. Design

This subchapter presents a combination of drafts and mock-ups, in order to capture the pre-designed layout, navigation and overall aesthetics of the application. The mockups present a static representation of some of the major user interfaces of the application, which in order are: Search for Evidence (Figure 20), Create a New Case (Figure 21), Case Management (Figure 22) and Investigator Profile (Figure 23). In this way, key design elements such as content layout, buttons and text placement are highlighted. Next, the wireframe on Figure 24 depicts the final appearance and structure of the application in the use case “Viewing the chain of custody of an evidence item”, focusing on the layout of the elements and the flow between user environments across all types of devices, in this case, a smartphone.

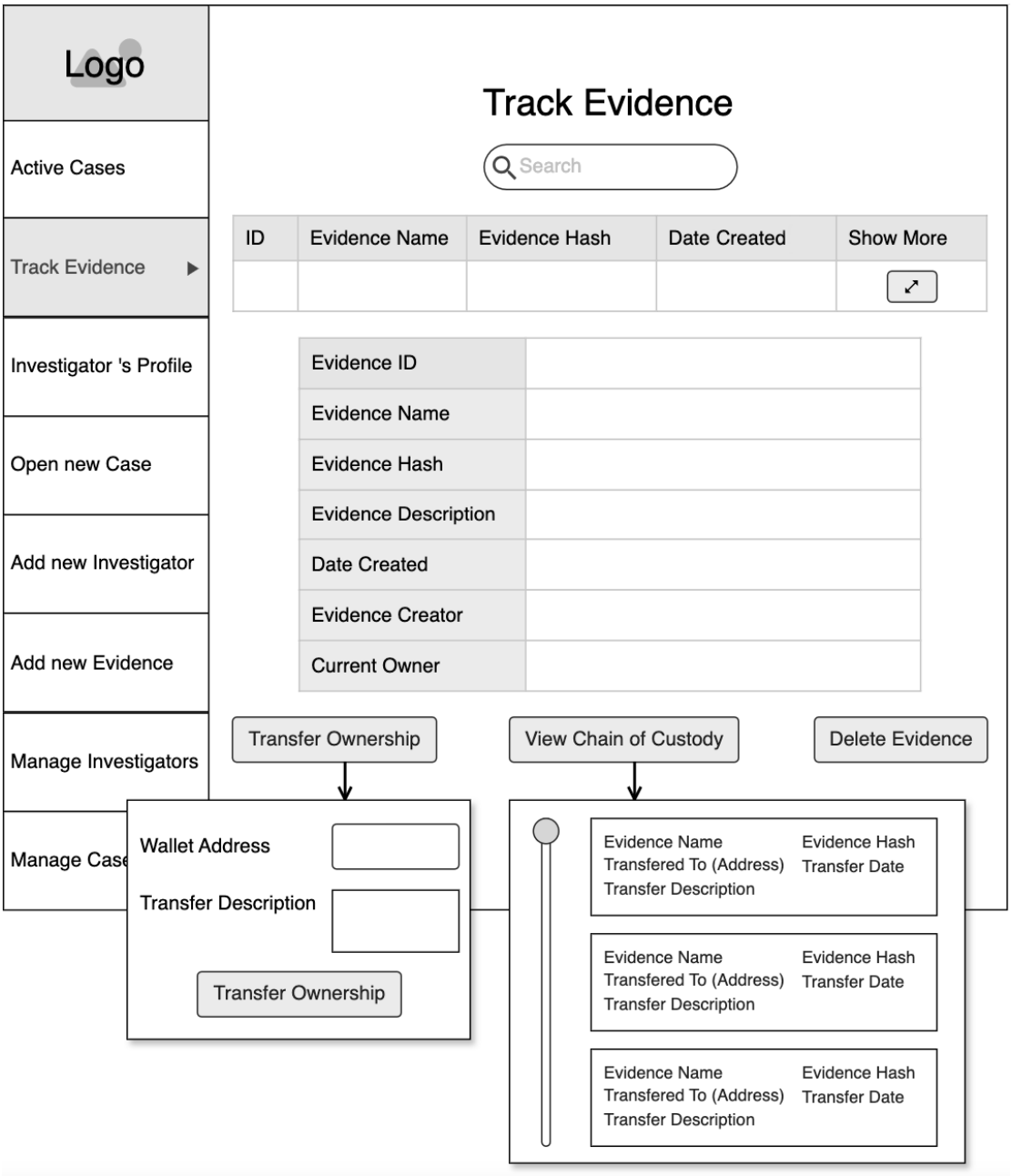

Figure 20 - Track Evidence Mockup

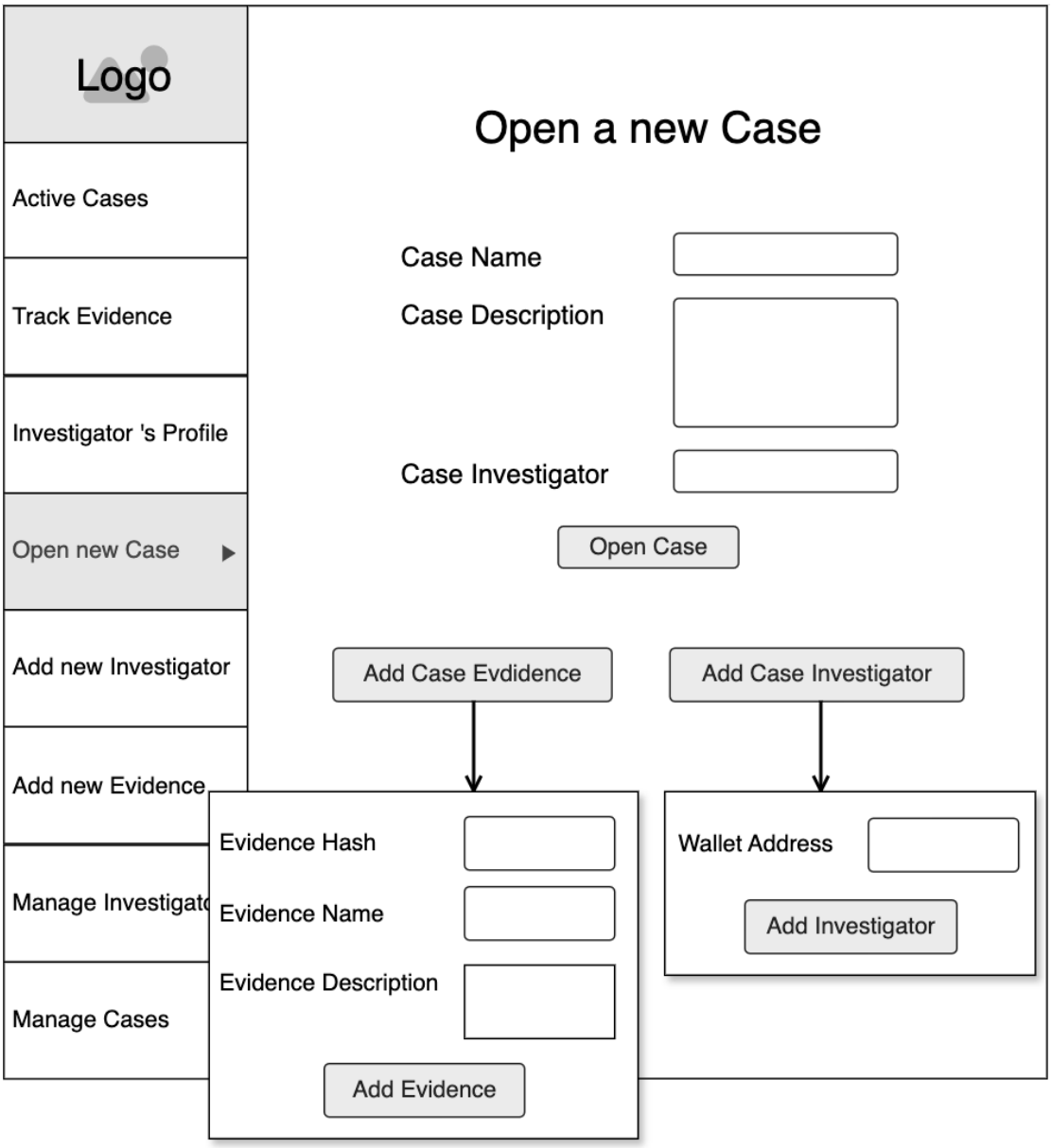

Figure 21 - Open a new Case Mockup

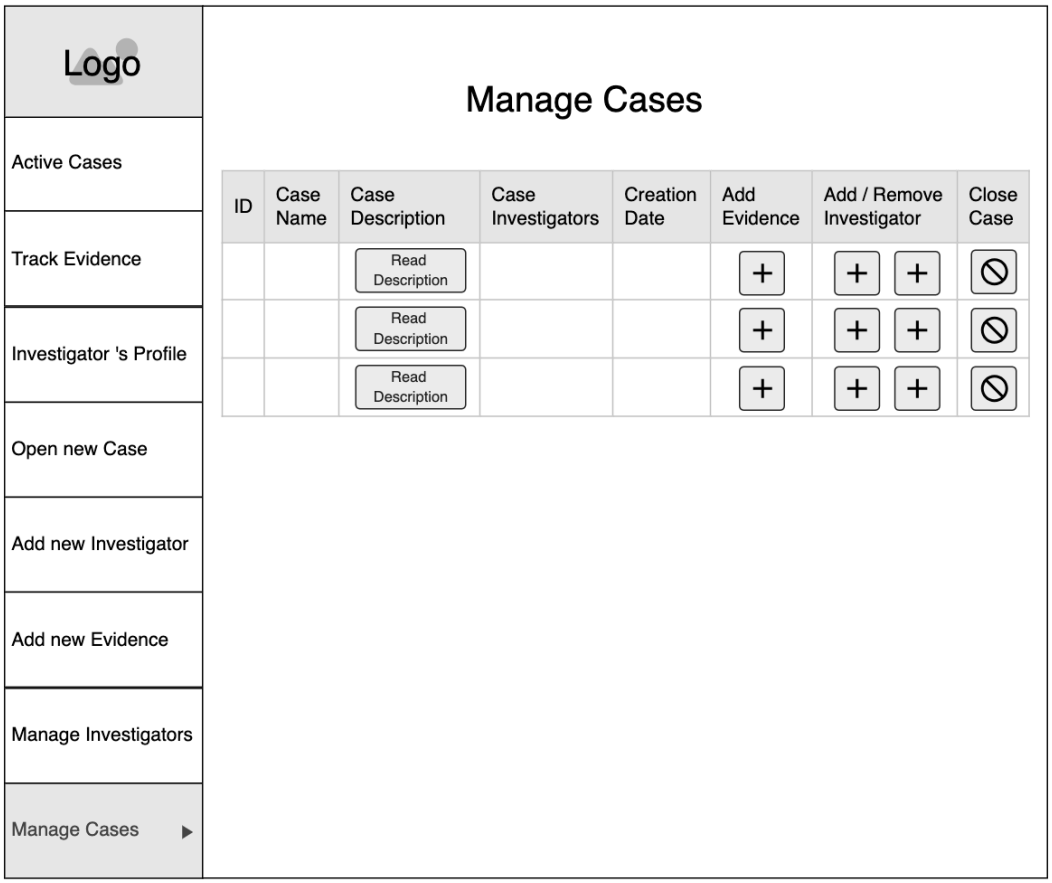

Figure 22 - Manage Cases Mockup

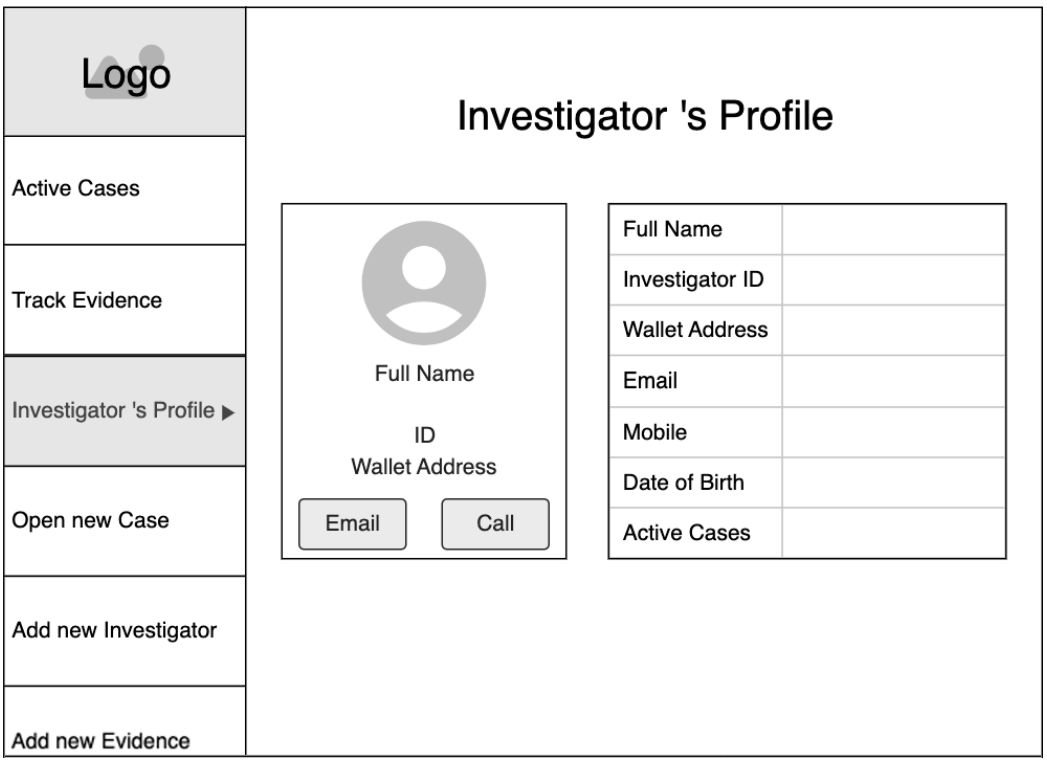

Figure 23 - Investigator 's Profile Mockup

Figure 24 - View Chain of Custody Wireframe

Chapter 6: Development of the private Ethereum Blockchain

As discussed in Chapter 2, a blockchain is a decentralized ledger that connects an ever-expanding block sequence via cryptographic hashes. These blocks contain a certain number of transactions between users and the creation of each of them is attributed to a selected miner based on the consensus algorithm. Also, the information contained in a blockchain is available to everyone, eliminating the need for third-party intervention and promoting direct communication between any two nodes, bypassing intermediate costs.

The choice of Ethereum as the development platform on which the chain of custody application will be based entails several compelling reasons. Ethereum, known for its smart contract capabilities and its decentralized nature, offers an ideal environment for creating secure and transparent applications. The integration of Ethereum's blockchain technology guarantees the immutability of data and increases its integrity, two key aspects in the field of digital forensics. Furthermore, Ethereum provides the flexibility to customize many key features of the protocol, including the choice of consensus algorithms and the creation of customized public or private, permissioned or permissionless blockchain networks.

For the needs of this project, an Ethereum private network, “ThemisChain”, was deployed on the “AWS” (Amazon Web Services) infrastructure, providing a solution that allows maintaining control of data while reducing financial costs and enhancing the speed of transactions. Although a private blockchain does not offer a completely decentralized structure, as is the case with a public counterpart, it can still inherit key features such as traceability, integrity and confidentiality and allows the use of a distributed ledger between all parties involved, ensuring the synchronization of their records. The blockchain infrastructure was implemented through “Geth”, a popular Ethereum implementation that can turn any computer into an Ethereum blockchain node. Geth provides the ability to create a private network and allows for the customization of all blockchain components, as well as the consensus mechanism used. In the context of the private and permissioned blockchain developed, the “PoA” (Proof-of-Authority) consensus algorithm was adopted. The network consists of three (3) nodes with the status of “validator” (the equivalent of “miner” in “PoW”), which, due to the consensus algorithm, are the only ones responsible for checking and validating

the transactions that take place. However, more "validators" are likely to be added in the future. The cost of transactions is set to be zero, because the blockchain, and by extension the chain of custody application, will be used by pre-defined trusted entities for security and evidence retention purposes, which means that the size of the data will be many "MB" (Megabytes) and therefore will be characterized by high costs.

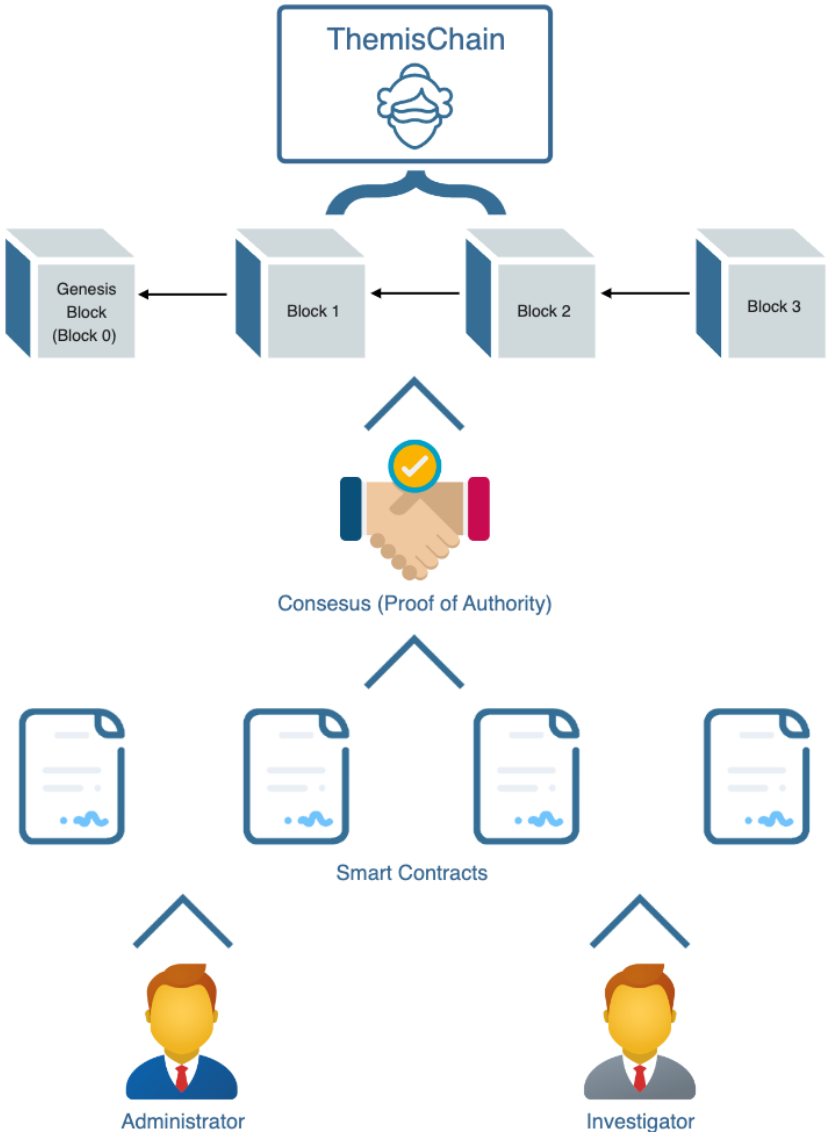

Figure 25 - Simplified architecture of the "ThemisChain" network

6.1. Network configuration

Below the steps followed for the configuration of the network will be analyzed, which will be divided into four (4) sections: 1. Configuring the Virtual Machines, 2. Installing Geth, 3. Creating the blockchain, 4. Adding the nodes' accounts to the "Metamask" wallet. For a better understanding of the content of the current subchapter, along with the third-person singular, the first-person plural will be used.

1. Configuration of the Virtual Machine

The first step to configure the virtual machine is to create an account on the "AWS" website (https://aws.amazon.com/) (Figure 26), which, as mentioned above, will host the blockchain. Then, through the "Services" menu, navigate to the "EC2/Instances" path and click the "Launch Instances" button (Figure 27).

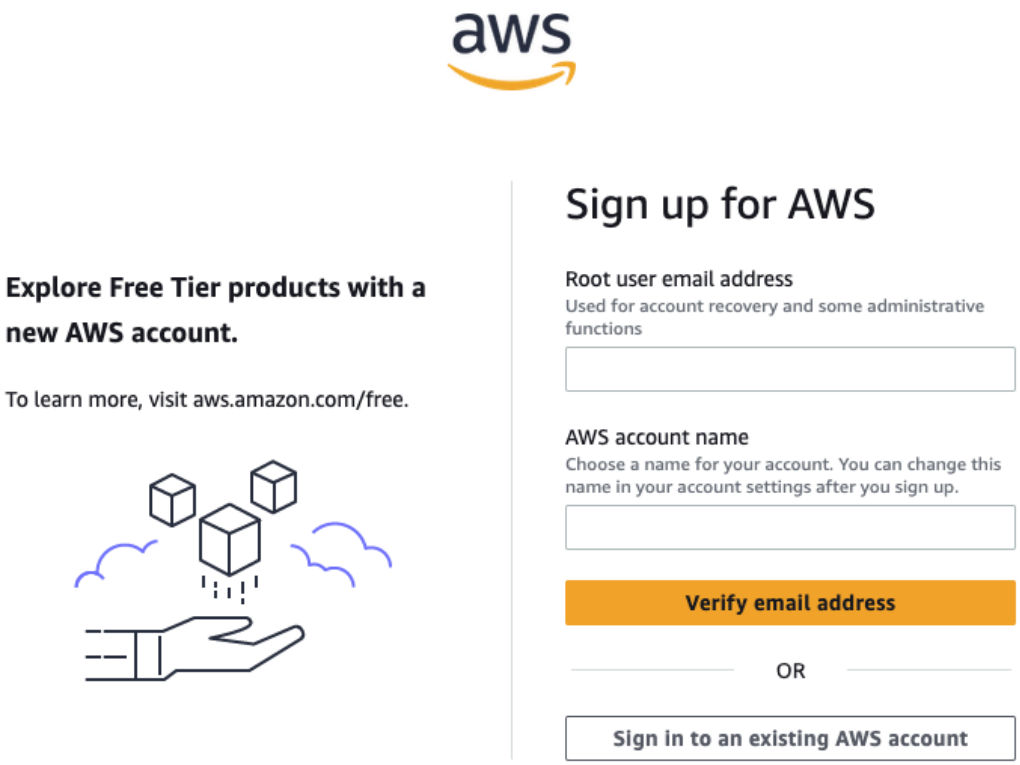

Figure 26 - Create an account in "AWS"

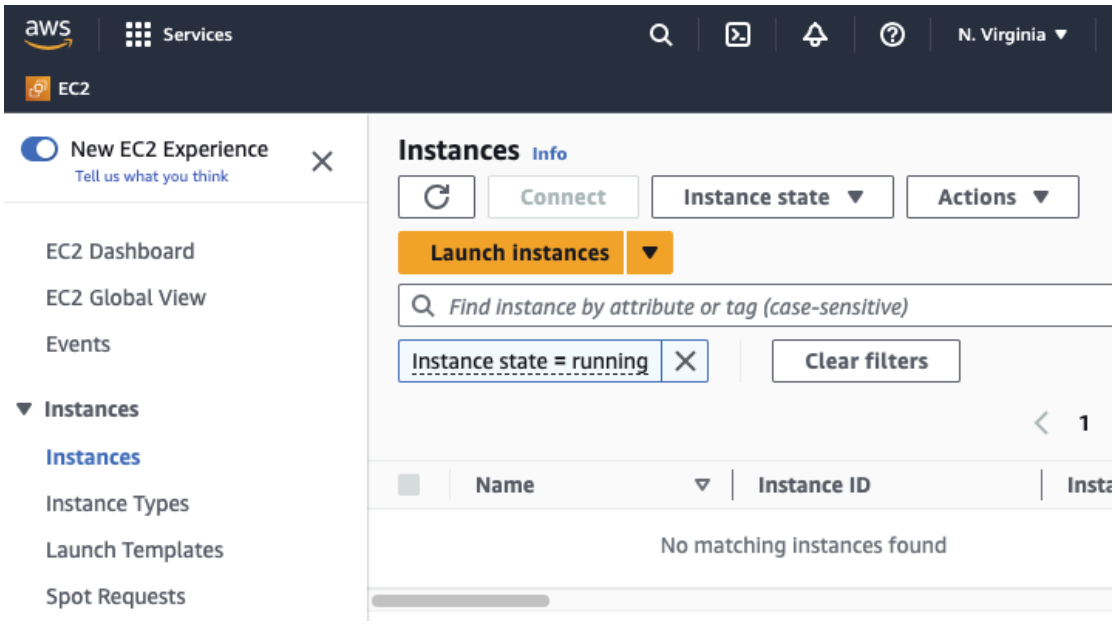

Figure 27 - Create virtual machine button

It is important, before starting the process, to create a "Key pair", which serves to connect securely to the virtual machine, through the option found in the virtual machine creation form (Figures 28 and 29).

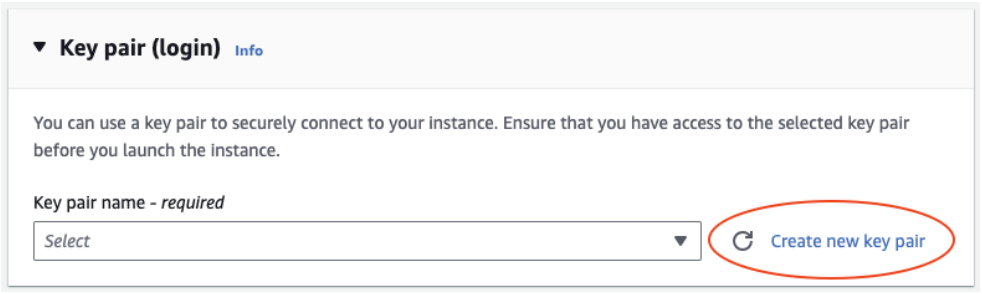

Figure 28 - Selecting the creation of a "Key pair"

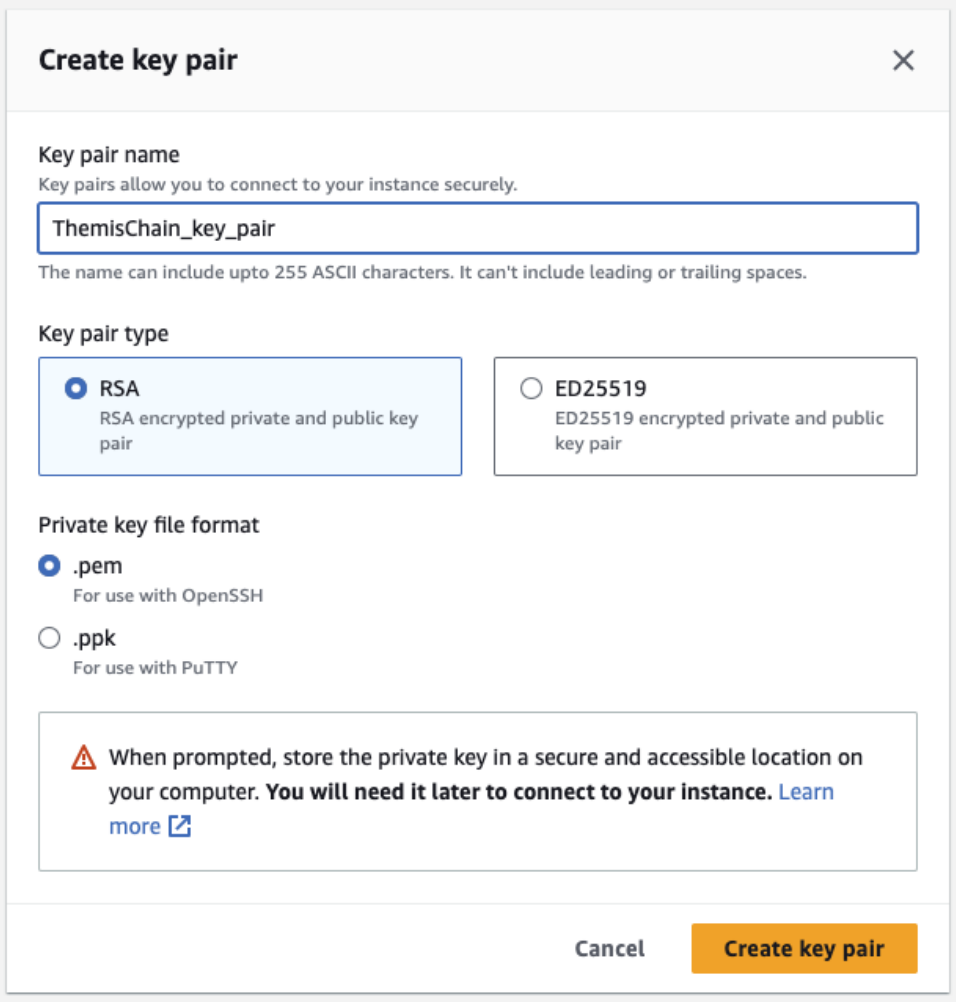

Figure 29 - Form for creating a "Key pair"

Once the "Key pair" is created and we save the file with the ".pem" extension that is generated, we can continue the process by filling in the form in Figure 30. As shown in the figure, the name of the virtual machine is "ThemisChain - Node0", as it is the first node of the blockchain, the operating system is "Ubuntu Server 22.04 LTS" and it runs on a system with a single-core processor and 1GiB (Gibibyte - 1GiB ≈ 1.074GB) of Ram memory.

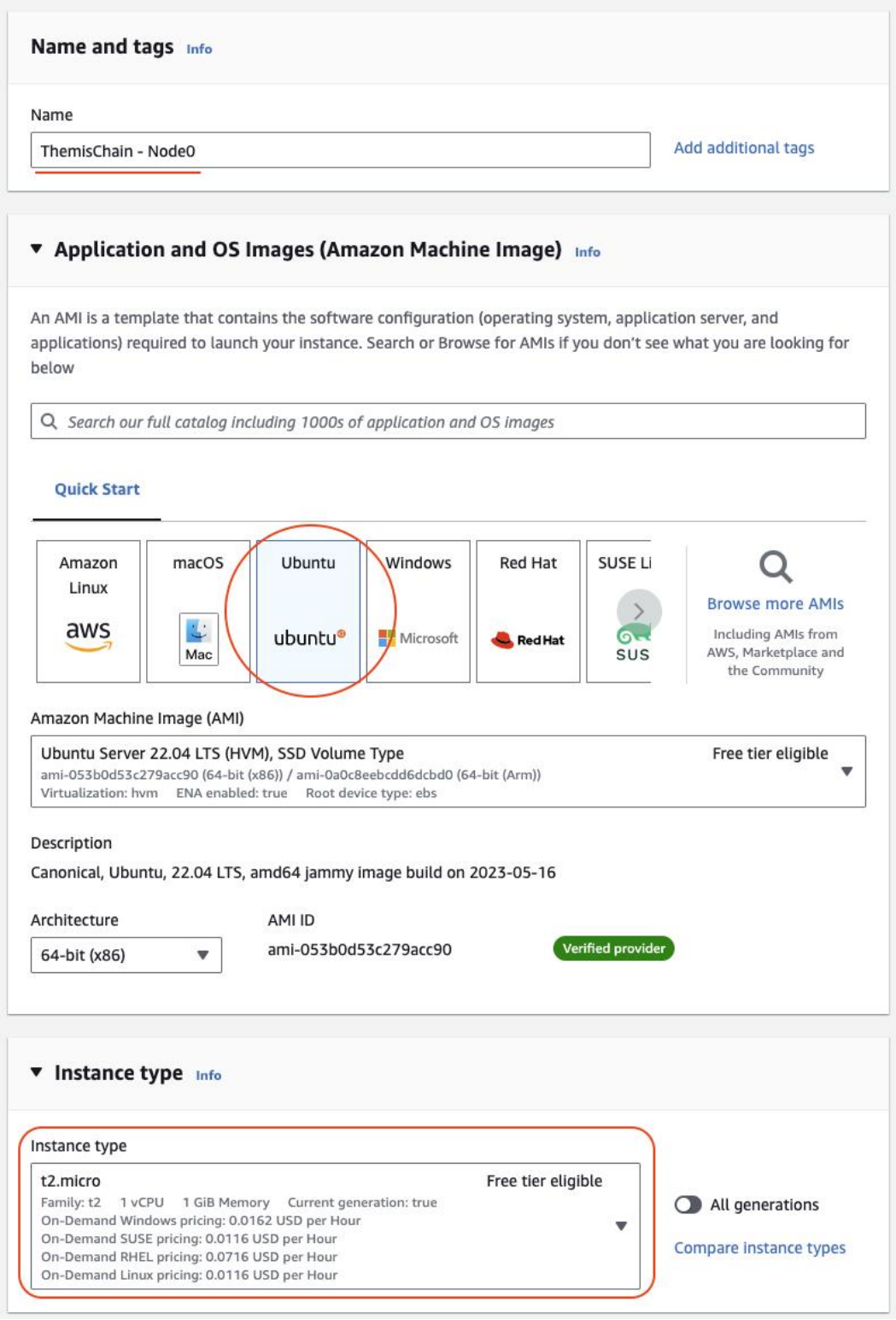

Figure 30 - Virtual machine creation form

Then, through the form, it is necessary to define a "Security Group", which is a set of rules that control the traffic of the virtual machine or, more simply, who can access it. Figures 31 and 32, list the rules applied to the blockchain.

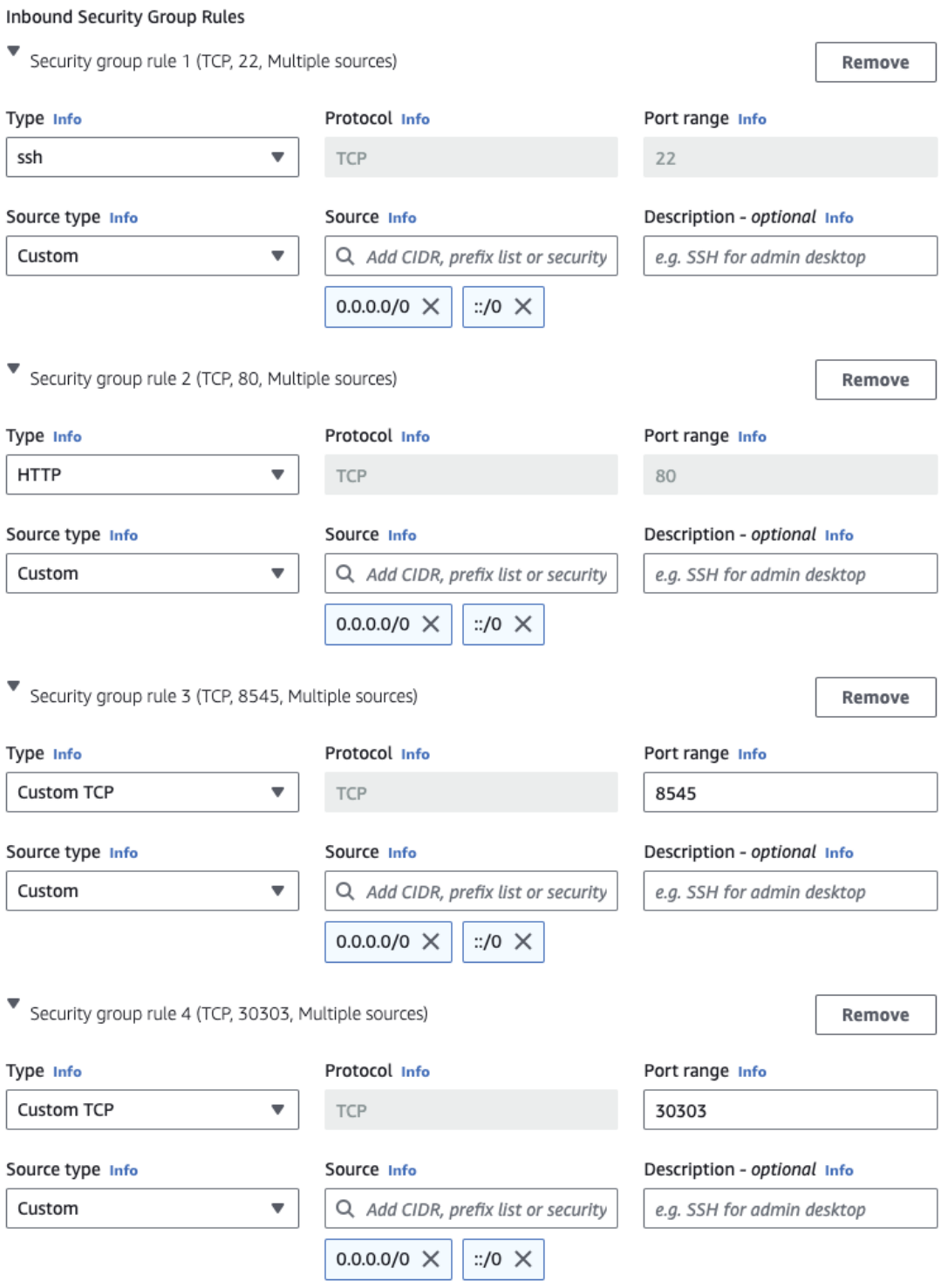

Figure 31 - Rules of the "Security Group" (1/2)

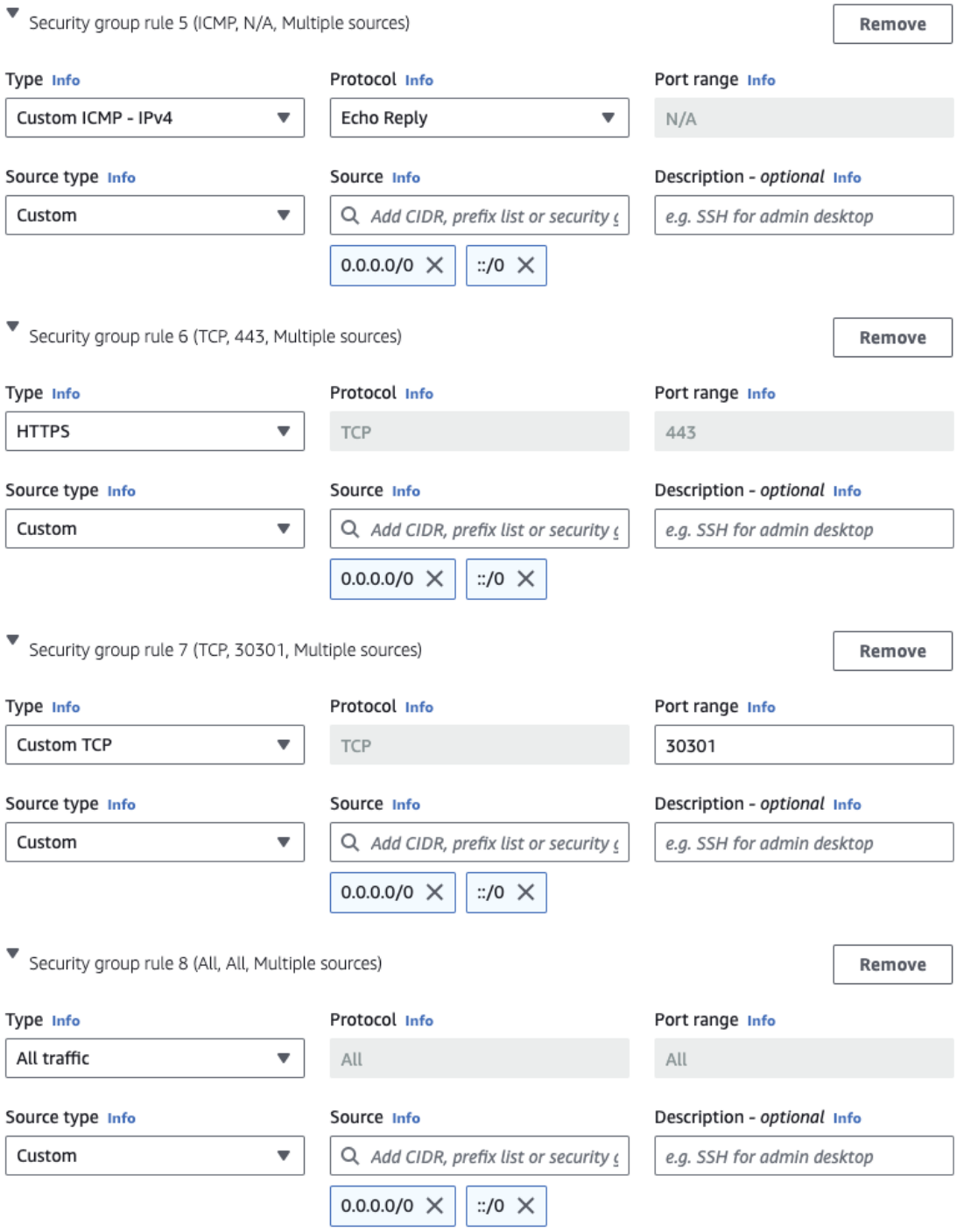

Figure 32 - Rules of the "Security Group" (2/2)

As can be seen in the above images, all rules were set to be accessed by all IP addresses (due to 0.0.0.0/0 and ::/0), because blockchain was created and intended for the current thesis and not for any other purpose. Therefore, there is no question of

maintaining data security rules in the real world. Also, in addition to the predefined ports, some that have been added manually are distinguished:

- **8545:** The port of the blockchain.
- **30303, 30301:** Ports that allow nodes to communicate with each other.

Completing the process, we are asked to select the capacity of the system, which in this case is 16GiB (Figure 33).

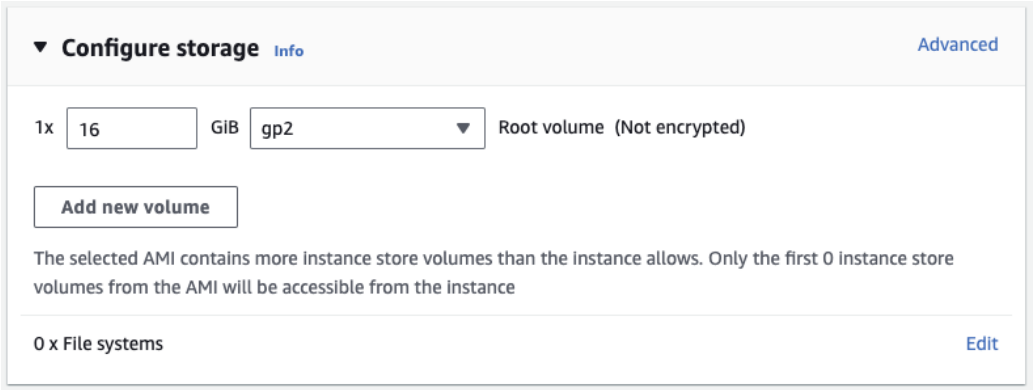

Figure 33 - Virtual machine capacity selection

After we create the virtual machine, what needs to be configured is its public IP address, because after each reboot it will change, which means that it will not be possible to connect to the blockchain. This is achieved by creating an "Elastic IP Address" and then assigning it to the virtual machine. By going to the "Elastic IPs" link on the left of the page (Figure 34) and clicking on the "Allocate Elastic Ip address" button, the IP address is created and the "Associate this Elastic IP address" button appears at the top of the screen (Figure 35). This button takes us to the "Elastic IP" address assignment form (Figure 36), where we select the virtual machine that we want and complete the process.

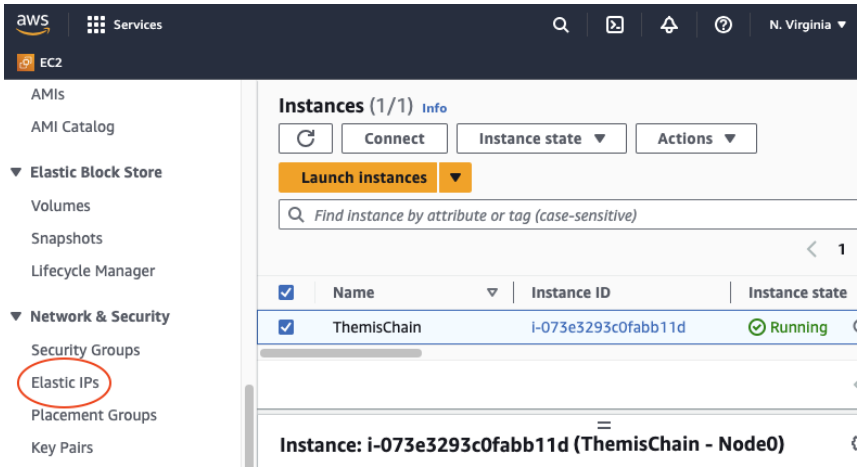

Figure 34 - Location of the "Elastic IPs" link

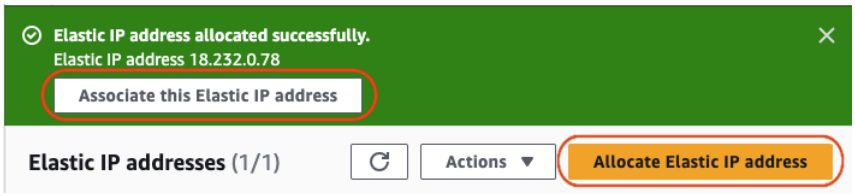

Figure 35 - The "Allocate Elastic Ip address" and "Associate this Elastic IP address" buttons

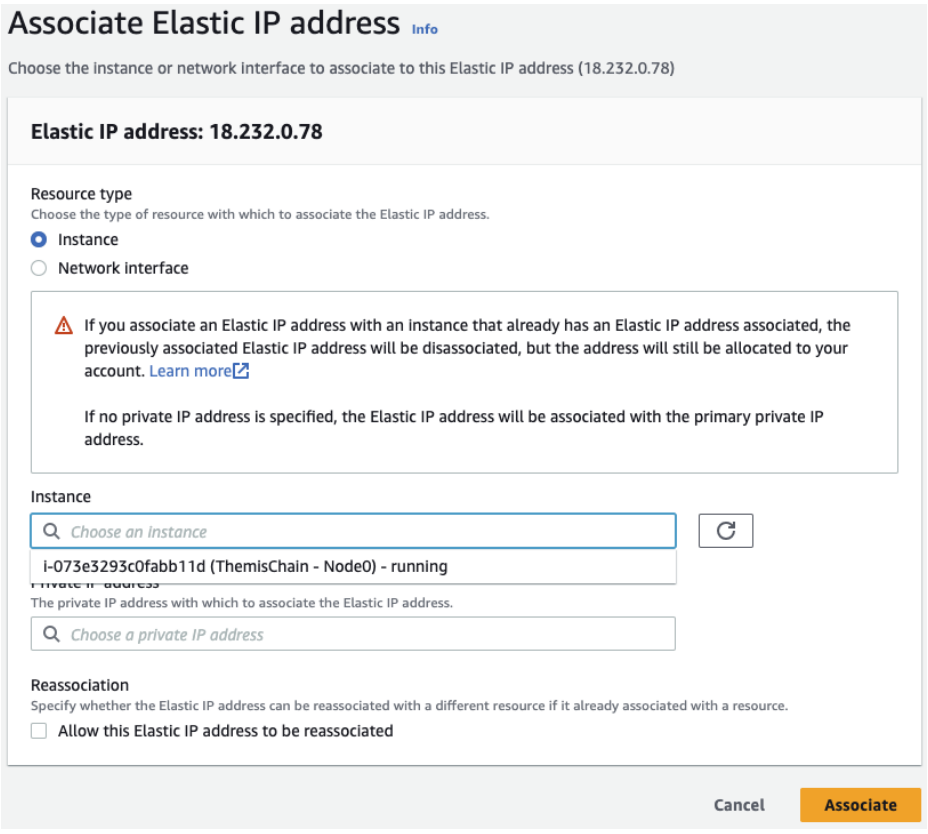

Figure 36 - The "Elastic IP" address assignment form

The final form of the virtual machine is as follows:

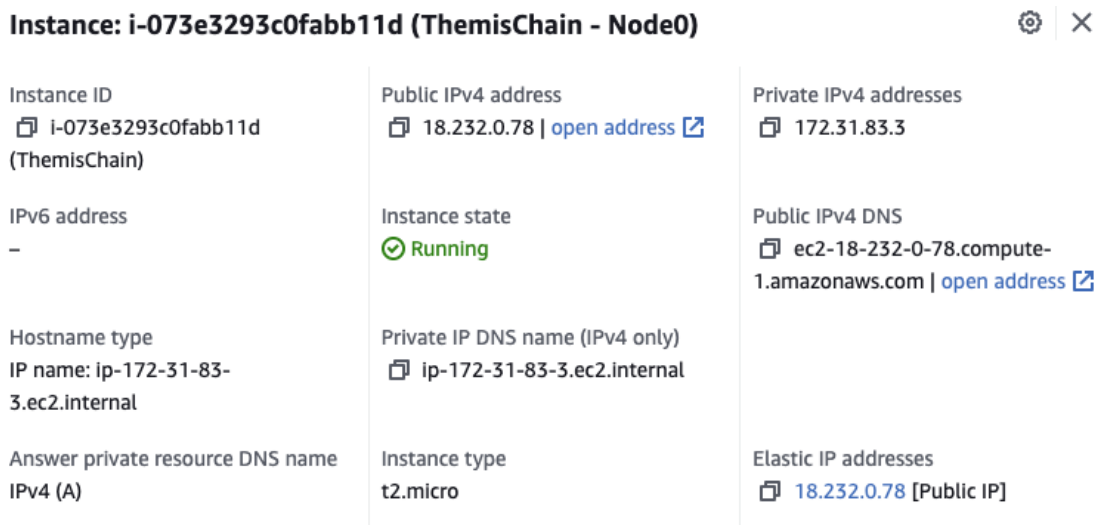

Figure 37 - The final form of the virtual machine

2. Installing Geth

The first step in installing Geth is to connect to the virtual machine. By pressing the "Connect" button located above the virtual machine, we go to the "SSH client" tab, which contains some instructions (Figure 38). After following them, we copy the command at the bottom (*ssh -i "ThemisChain_key_pair.pem"* ubuntu@ec2-18-232-0-78.compute-1.amazonaws.com) and execute it in a terminal, inside the folder where we saved the file with the ".pem" extension (see page 62) (Figures 39 and 40).

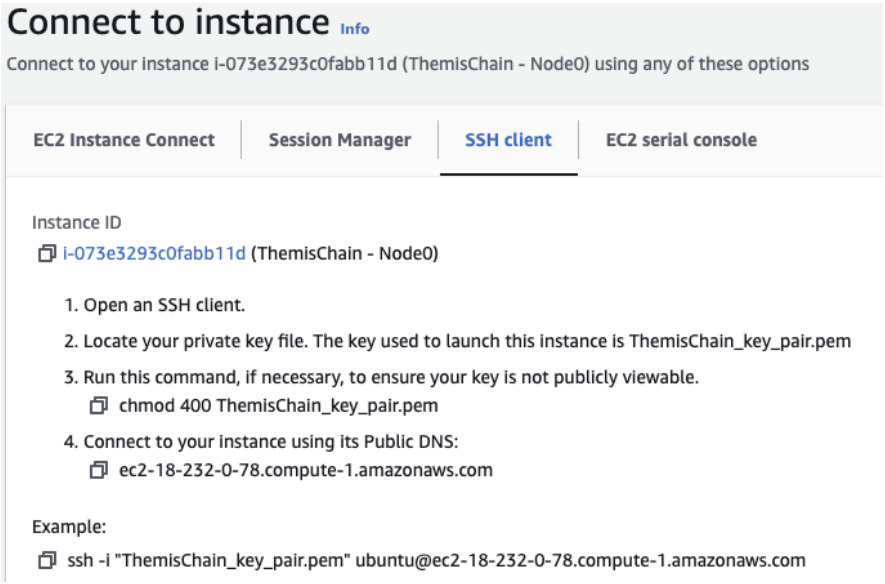

Figure 38 - Instructions for connecting to the virtual machine

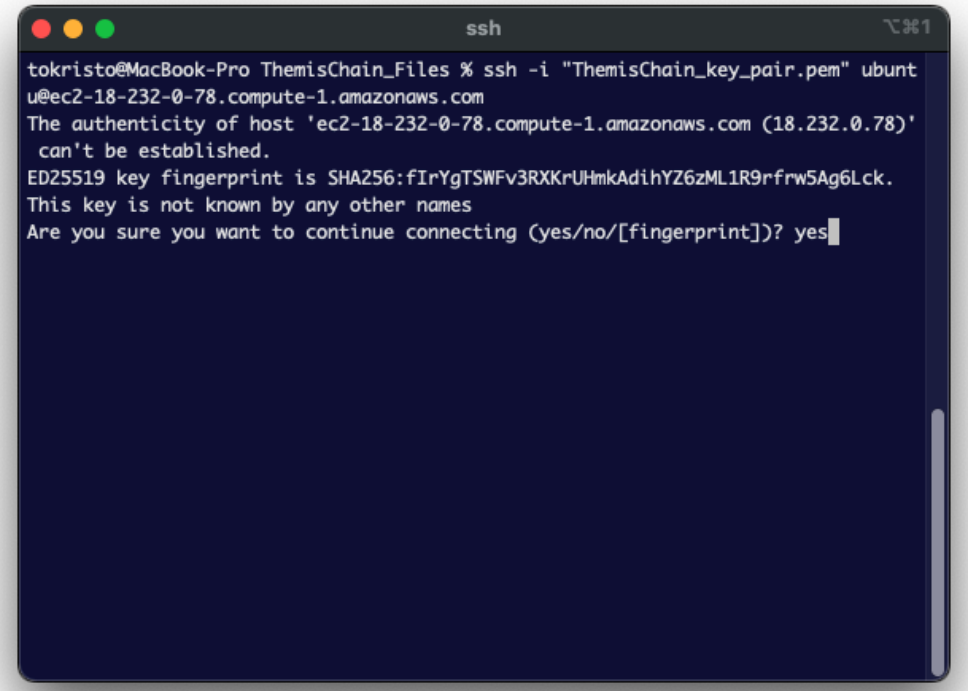

Figure 39 - Connecting to the virtual machine via terminal (1/2)

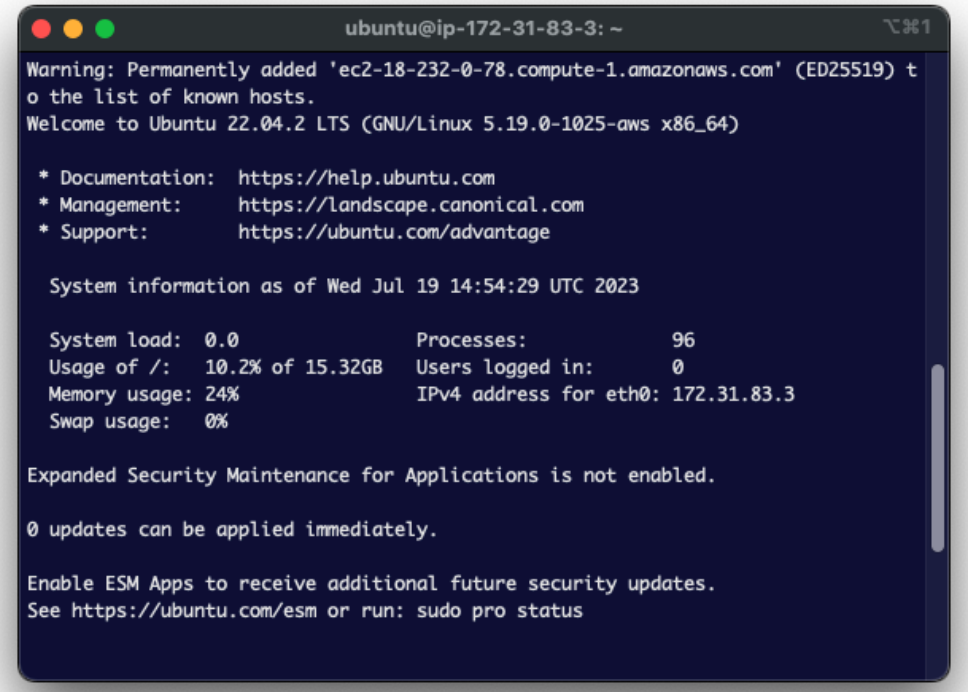

Figure 40 - Connecting to the virtual machine via terminal (2/2)

Next, we go to https://gethstore.blob.core.windows.net/builds/geth-alltools-linux-amd64-1.9.18-f5382591.tar.gz to download the compressed file containing version 1.9.18 of Geth, but also some accompanying tools that will be used later. Once the download is complete and placed in the folder where the file with the ".pem" extension is located, we execute the following command to upload it to the virtual machine:

scp -i "ThemisChain_key_pair.pem" geth-alltools-linux-amd64-1.9.18-f5382591.tar.gz ubuntu@ec2-18-232-0-78.compute-1.amazonaws.com:~ (Figure 41)

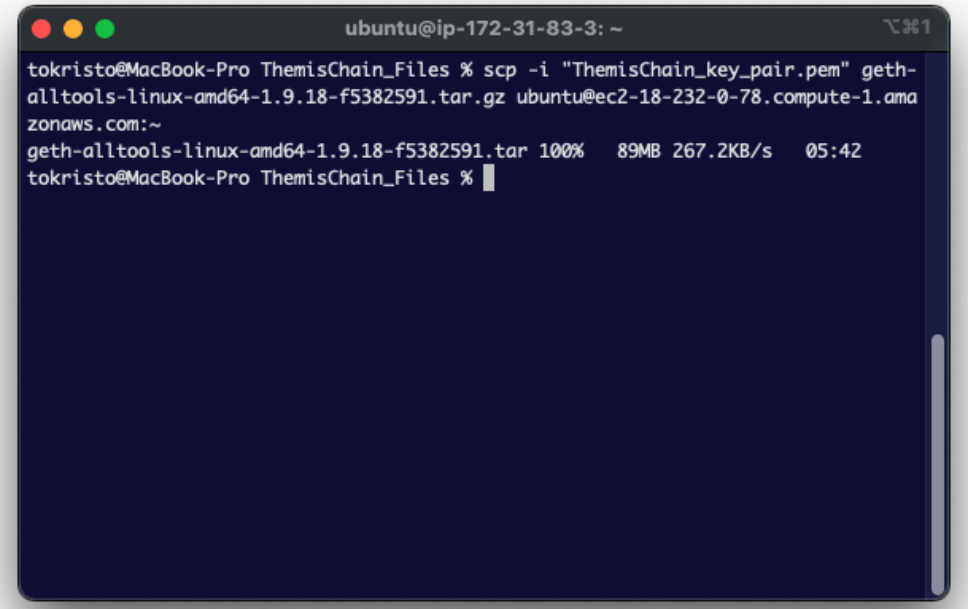

Figure 41 - Uploading the compressed Geth file to the virtual machine

Then, after obtaining administrator privileges with "*sudo -s*", we follow a series of commands to: 1. Unzip the file we downloaded, 2. Grant permission to the Geth file to run, and 3. Move all the files to the "/usr/local/bin" system path.

1a. tar -xvf geth-alltools-linux-amd64-1.9.18-f5382591.tar.gz

1b. cd geth-alltools-linux-amd64-1.9.18-f5382591

2. chmod +x geth

*3. cp * /usr/local/bin/*

Finally, by running the command "geth version" we can check if the above process was completed successfully (Figure 42).

Figure 42 - The "geth version" command

3. Creation of the blockchain

After logging into the virtual machine (see page 68), we need to create a node account in geth. The steps to create such an account are as follows (after obtaining administrator privileges with the "*sudo -s*" command) (Figures 43 and 44): 1. Go to the path "/opt", which is used to install software packages on "Linux" operating systems 2. Create a folder containing the necessary blockchain files, 3. Create a folder for the node files, 4. Creation of account, where we are asked to enter a code, which is "12345" similarly for all three nodes created.

1. cd /opt

2a. mkdir ThemisChain

2b. cd ThemisChain

3a. mkdir node0

3b. cd node0

4. geth --datadir "./data" account new

Figure 43 - Steps to create a node account in "geth" (1/2)

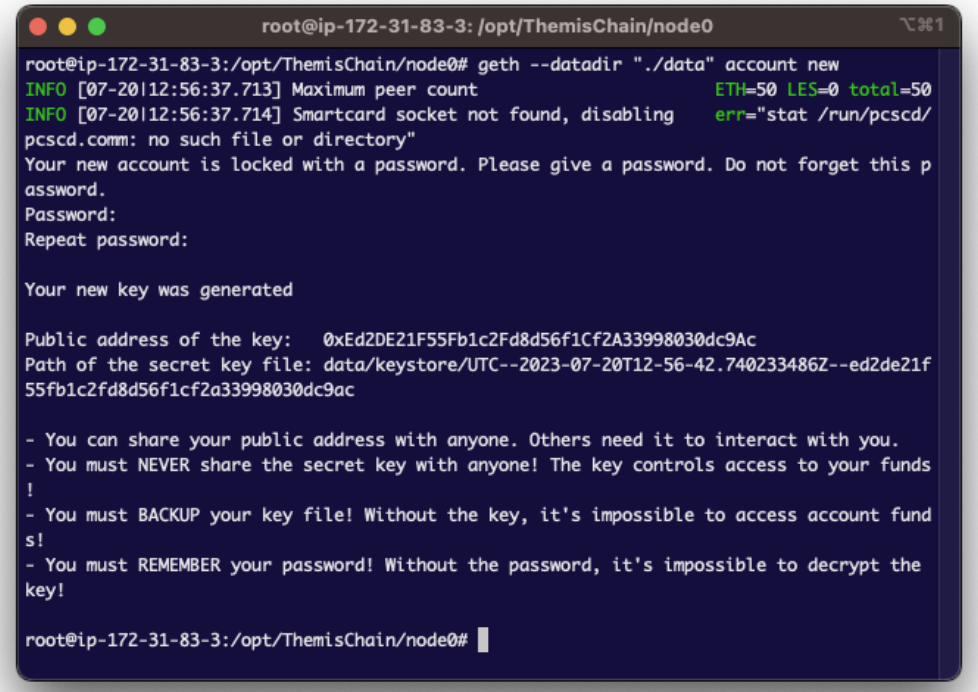

Figure 44 - Steps to create a node account in "geth" (2/2)

From the above image it is important to preserve the "Public address of the key", in this case "0xEd2DE21F55Fb1c2Fd8d56f1Cf2A33998030dc9Ac", because it will be needed later.

Before proceeding to the creation of the genesis block (see page 14), it is necessary to perform the steps of the above two procedures (1. Configuring the Virtual Machine and 2. Installing Geth), as well as the steps presented so far in this procedure for the remaining two nodes of the blockchain, as the creation of a PoA (proof of authority) blockchain requires the existence of at least one node (in this case all three).

To create the genesis file, used to develop the genesis block, the "puppeth" tool of geth (Figure 45) was used, which asks the user to type only the data needed, greatly facilitating the process. These data, in the order depicted in Figure 46 are:

- Create a new genesis file.
- Selection of the "Proof of Authority" consensus algorithm.
- Set the new block creation time to five (5) seconds.
- Set the accounts that can validate a transaction (validators) (see page 58).
- Setting the accounts to be credited with the cryptocurrency "ETH".
- Set the ID of the blockchain.

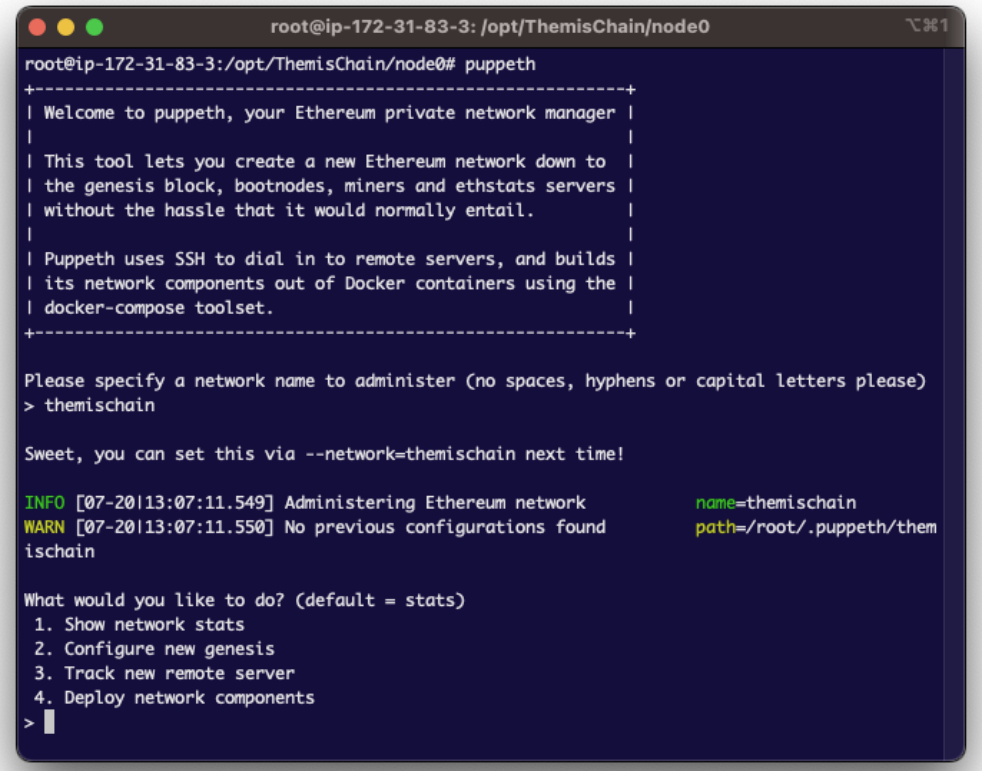

Figure 45 - The "puppeth" tool

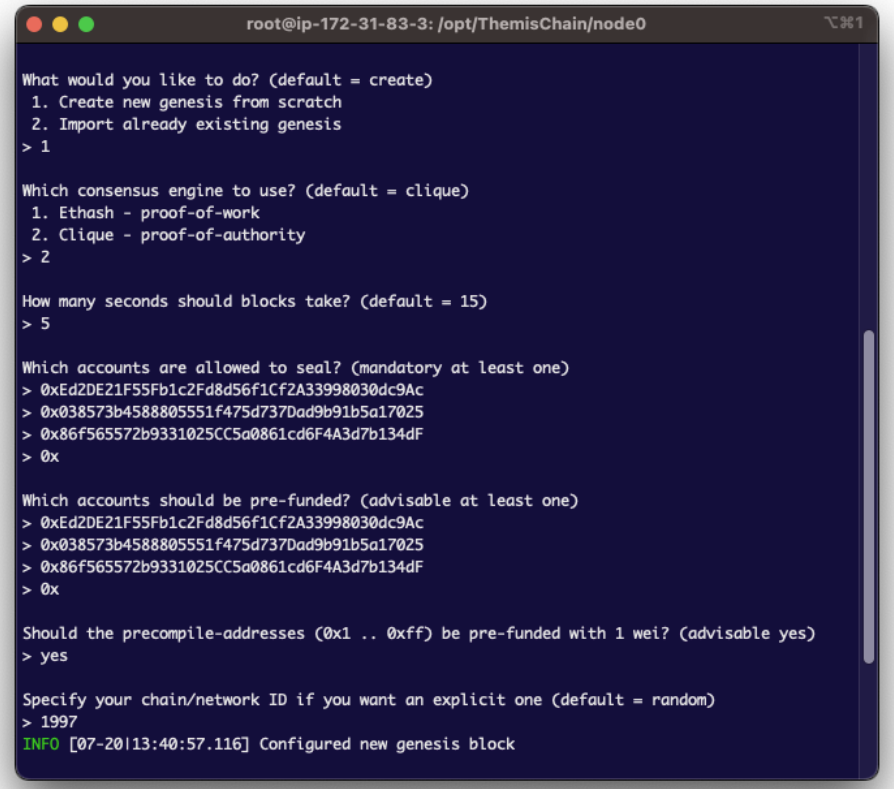

Figure 46 - Steps to create the "genesis" file

Then, we extract the generated files (Figure 47) and delete the file "themischain-harmony.json", because it will not be needed later (Figure 48).

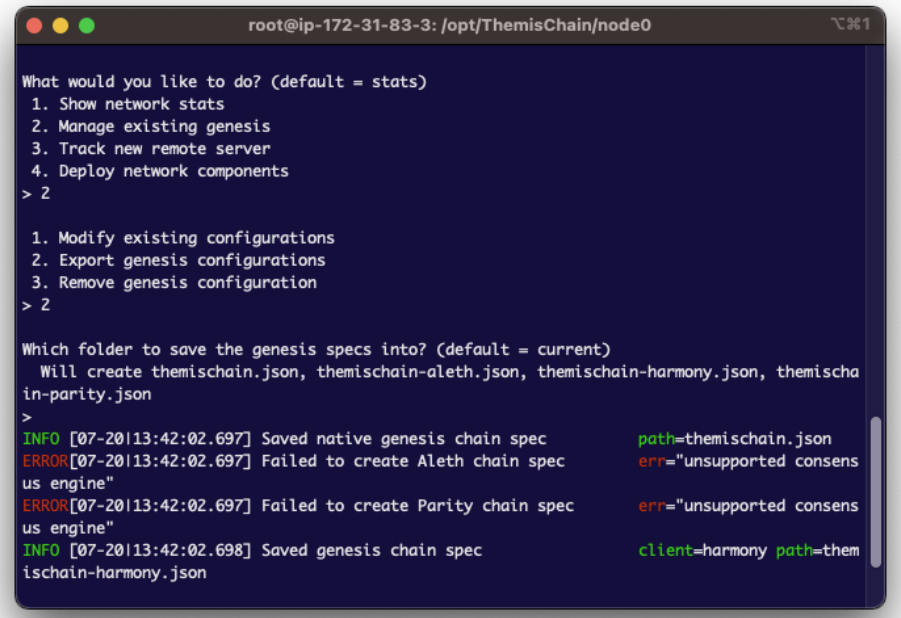

Figure 47 - Exporting the generated files from the "genesis" file creation process

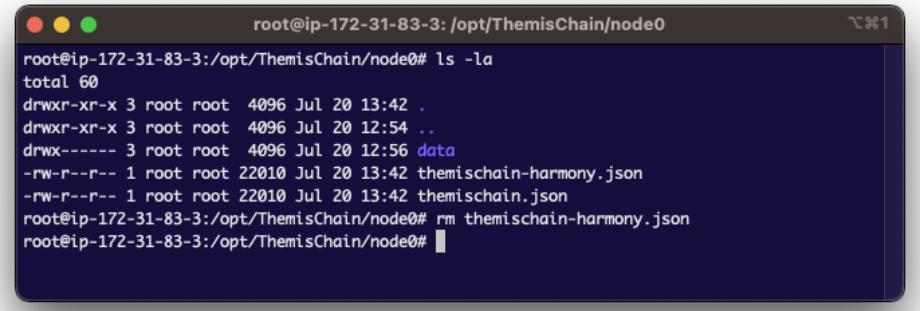

Figure 48 - Delete the file "themischain-harmony.json"

Then, the file "themischain.json" was edited-minimized using the application "Visual Studio Code" and renamed to "genesis.json". In order to save space, the content of the "genesis.json" file is presented in subsection 9.4.

The last step is to initialize the node, based on the "genesis.json" file with the command "*geth --datadir "./data" account new*", in order to create the genesis block (Figure 49). For the other nodes the procedure remains the same, but there is no need to create a new genesis file, since the "genesis.json" file created above is the one on which their initialization will be based.

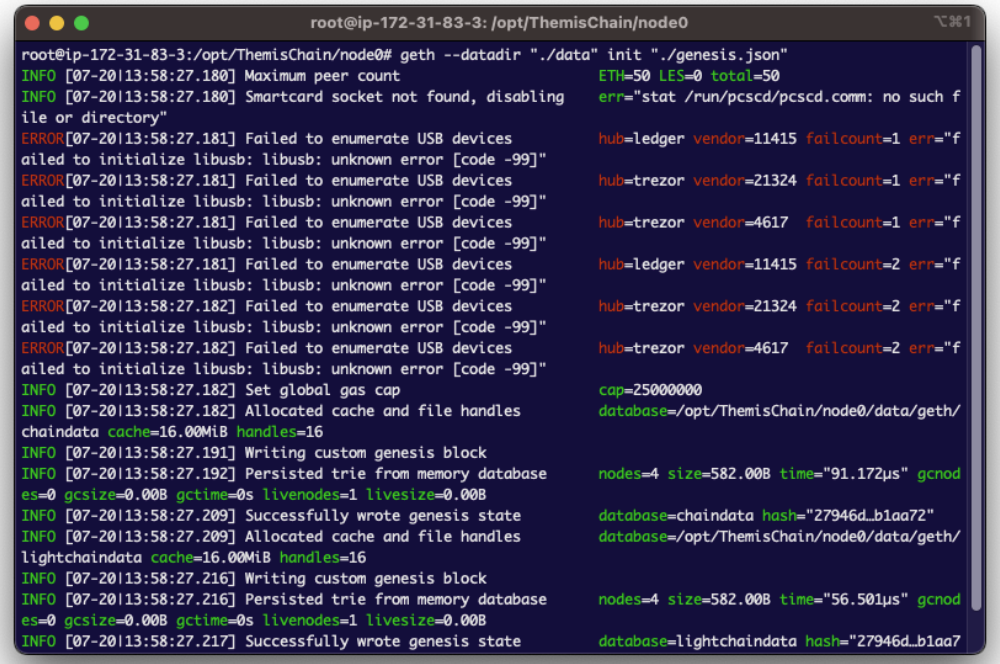

Figure 49 - Initialization of the blockchain based on the "genesis.json" file

3.1. Connecting nodes to each other

Once all three nodes have been created, the next step is to connect them to form the final form of the blockchain. First, we need to attach to each node via "Geth Javascript Console" by typing the command "*geth attach (node IP address)*" (Figure 50). As shown in the image, the node is not attached to the others (*admin.peers -> []*). The element needed to make the attachment is the identifier of each node (enode), which is displayed via the command "*admin.nodeInfo.enode*". Therefore, having the "enodes" of the nodes, we connect to the Geth Javascript Console of each node and execute the command "*admin.addPeer(enode)*" for those nodes remaining (Figure 51).

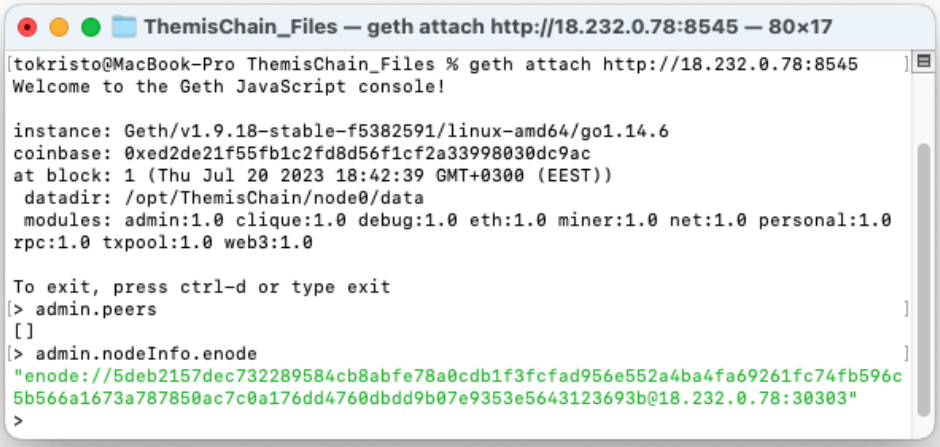

Figure 50 - The "Geth Javascript Console"

Figure 51 - The admin.addPeer() command

Then, we go to the "data" subfolder of each node (e.g., /opt/ThemisChain/node0/data) and create the file "static-nodes.json", which serves to automatically reconnect the nodes, even if one of them stops working for a period of time (for space saving reasons, the contents of the "static-nodes.json" file are listed in subsection 9.5).

3.2. Creation of "geth.service"

The way the blockchain is structured so far, every time the virtual machines are restarted, the blockchain would have to be restarted, which would make it unsuitable for the purposes for which it was developed. This is addressed by creating a service in the operating system of each virtual machine, which will be executed each time the system is booted. The steps to create this service are as follows (after obtaining administrative privileges with the "*sudo -s*" command): 1. Go to the system folder path "/etc/systemd/system", 2. Create the service file "geth.service" (to save space, the contents of the "geth.service" file are listed in subsection 9.6), 3. Grant permission to run the "geth.service" file, 4. Reload the contents of the folder, 5. Enable the service, and 6. Start the service.

1. cd /etc/systemd/system

2. *nano geth.service*
3. *chmod +x geth.service*
4. *systemctl daemon-reload*
5. *systemctl enable geth.service*
6. *systemctl start geth.service*

Also, we can check the status of the service via the command "*systemctl status geth.service*" (Figure 52).

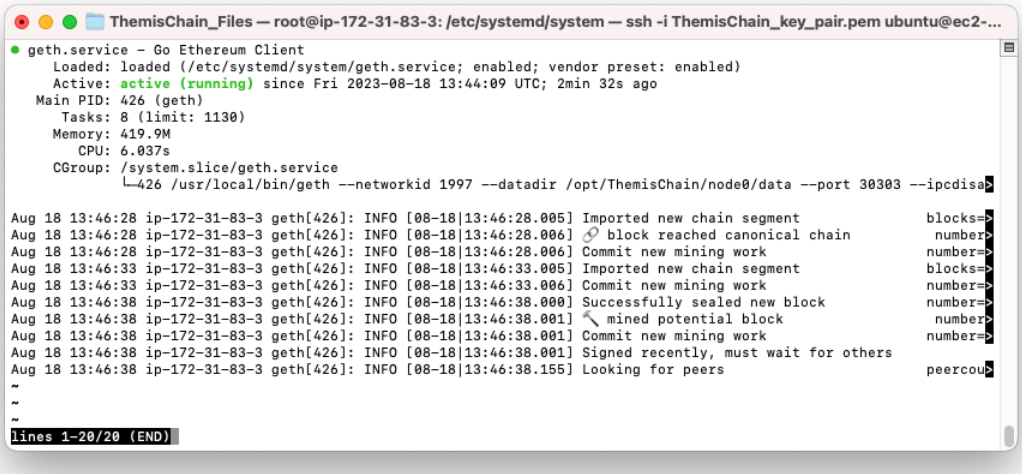

Figure 52 - The command "systemctl status geth.service"

4. Adding node accounts to the "Metamask" wallet

The addition of the node accounts to a wallet is necessary to process transactions when using the application. To start, the private key of the accounts should be extracted from the node records. By going to the "keystore" subfolder of each node (e.g. /opt/ThemisChain/node0/data/keystore), we find the file containing the private key, which we open using the "cat" command (Figure 53).

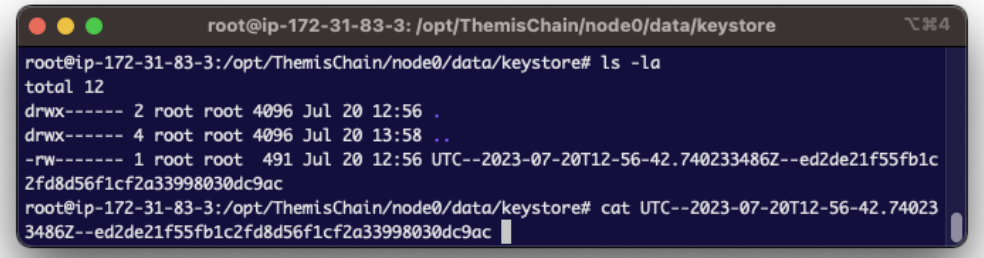

Figure 53 - The file containing the private key of the node account

After copying the contents of the file and saving it in a new one with the extension ".json" (e.g. node0account.json, see subchapter 9.7), we click on the "Import Account" option of the Metamask window (Figure 54) and then select the file we created above and enter the code we entered when creating each node account (see page 72) (Figure 55).

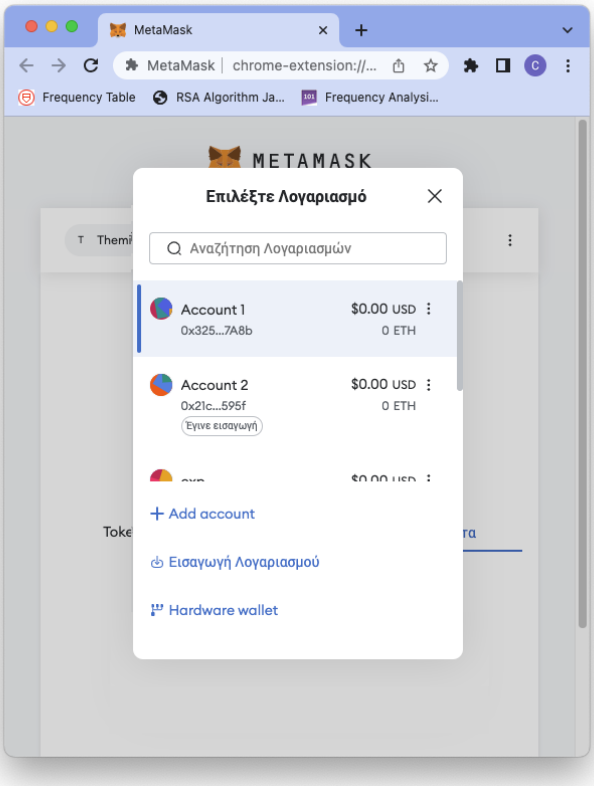

Figure 54 - Account management window in Metamask

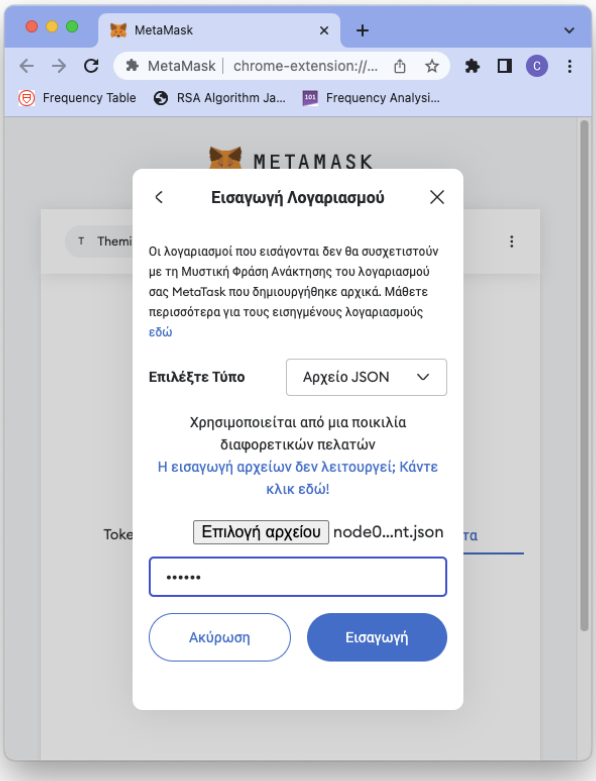

Figure 55 - Account import window in Metamask

Chapter 7: Development of the Chain of Custody application

This thesis presents “Themis”, an evidence management system based on a native private Ethereum blockchain. This solution aims to enhance the control and oversight of chain of custody of evidence in the context of digital forensics, offering features such as transparency, immutability and verifiability.

The model of this project (ThemisChain & ThemisApp) encompasses two roles: the investigator and the administrator. The tasks of an investigator are to collect evidence and calculate their hash, which is then passed offline to an admin to enter it into the blockchain. In this way, any sensitive information remains secure and unaltered. An admin is essentially an investigator with validator rights on the blockchain (see Chapter 6). An admin's duties include those of an investigator, along with adding a new investigator, adding a new case and its evidence, transferring ownership of an evidence between investigators, and deleting all the aforementioned data from the blockchain. In summary, the role of the admin is one of having full control over the blockchain and the data stored into it.

The process of creating and then managing the chain of custody starts with the admin adding the evidence to the blockchain. The uniqueness of the evidence remains intact as it is maintained between all participants in the network. Given the possible inclusion of sensitive data in the evidence, the actual evidence is protected from exposure to the application and thus the blockchain, hence only its hash values are stored. Each administrator is able to check the integrity of the data on the blockchain by calculating a hash value of each piece of evidence and comparing it with the corresponding value stored on the blockchain. This process guarantees the assurance of integrity and auditability.

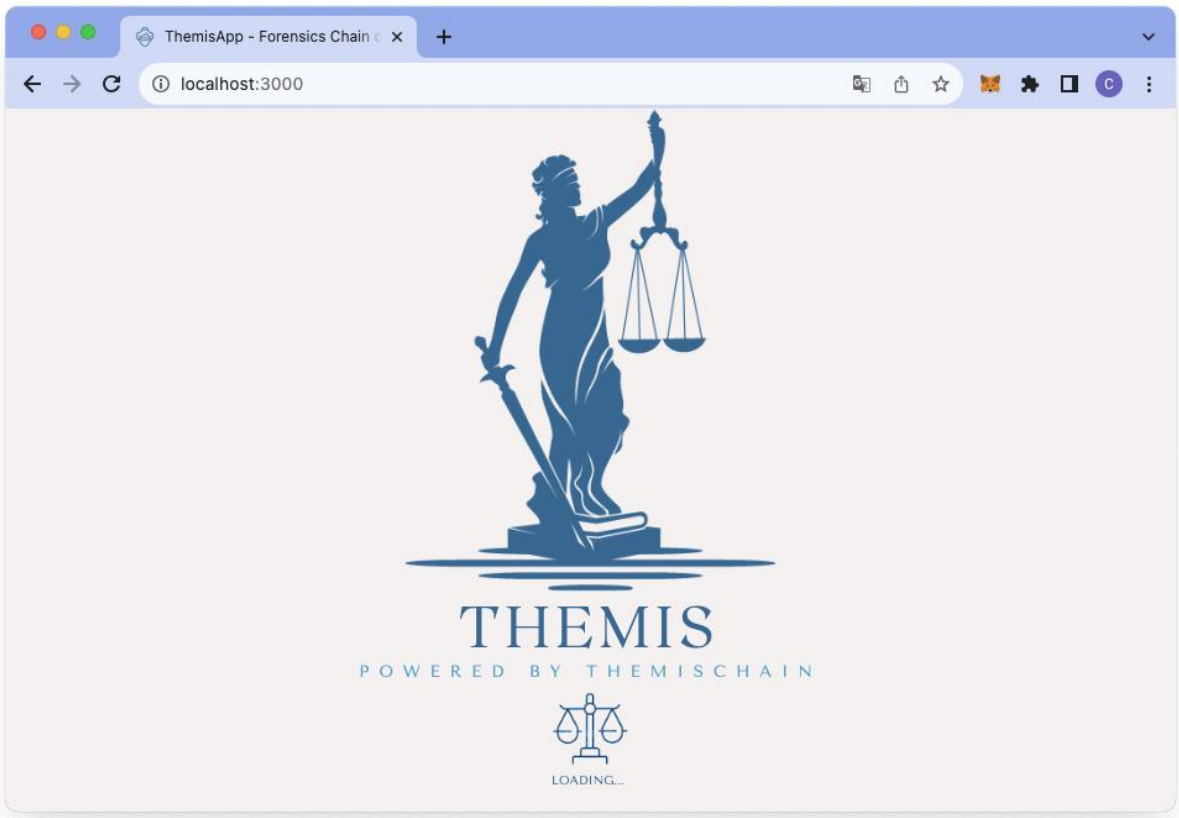

Figure 56 - The home page of the "Themis" application

7.1. Structure analysis

This chapter explores the architectural and organizational dynamics associated with the development of the decentralized application "Themis" and its seamless interaction with the native private Ethereum blockchain "ThemisChain". More specifically, the structural framework of the application is analyzed, revealing its role in preserving, tracking and managing digital evidence throughout the investigation process.

Through the application, details such as existing investigators and their details, characteristics of a case, such as the name, the investigators in charge, etc., and the necessary information for each piece of evidence are recorded on the blockchain. During a digital forensics' investigation, any transfer of evidence is automatically recorded on the blockchain through smart contracts, including basic information such as the address of the recipient and the exact date and time of the transfer.

The main features of the application concerning evidence can be analyzed below:

- **Generating evidence**

The role of the admin is the sole responsible for adding the evidence to the blockchain, enhancing the security and reliability of the digital forensic process. To add a piece of evidence, the admin must enter the required details, namely its hash generated through the SHA-256 algorithm, the name of the evidence and a brief description about it. Once the evidence is added, then the admin can transfer the ownership of it to the investigator who found it.

- **Transfer of ownership of evidence**

By facilitating the seamless collaboration and information flow between the stakeholders involved, the proposed system supports the secure transfer of evidence between investigators. It is also a function that only the admin role is capable of performing. In order to transfer evidence, the admin is asked to enter the address of the recipient and a short description stating the purpose of the transfer. It then updates the corresponding smart contract and adds the event to the chain of custody of the evidence.

- **Access to evidence**

Access to the evidence in question is available to investigators assigned to the case to which the evidence belongs, whether or not they are involved in the chain of custody of the evidence. However, none of them has the ability to make any changes.

Similarly to the above, an important feature of the application is that all parties involved (investigators) directly related to the cases under investigation are registered on the private blockchain network by the predefined admin-authenticators. These verifiers take on the role of supervision and coordination throughout an investigation process, as well as the proper functioning of the system in general.

The main design objectives that the “Themis” application seeks to achieve are:

- **Confidentiality:** Protecting data confidentiality by limiting the fact that an investigator cannot gain access to cases and evidence that are outside his or her jurisdiction.
- **Access control:** Only investigators registered by the default administrators have access to the application.

- **Integrity and Controllability:** Guarantee the integrity and auditability of data on the blockchain, to prevent tampering and facilitate its control.
- **Traceability:** Tracing the source of the evidence, to the extent that the evidence was added to the case under investigation.
- **Efficiency:** By default, the transaction costs are eliminated, so as not to burden the investigators using the application, as the main concern of the system is to ensure that it works properly, giving the best return on digital forensics data management.
- **Easy-to-use user interface:** In the field of digital forensics, where accuracy and efficiency are paramount, a user interface that is easy to manage improves user engagement, minimizes user familiarization time and reduces errors. It also ensures that users can navigate the application effortlessly, leading to faster task completion and fewer errors.

As a consequence of the above application objectives, the following "functional" and "non-functional" requirements are defined:

- **Functional requirements**
    - **User authentication and authorization:** A secure user login should be performed using authentication mechanisms. Users should be subject to rights-based access control to perform tasks according to their roles.
    - **Creating and recording evidence:** Only the administrator should be allowed to create and record evidence on the blockchain.
    - **Transfer of evidence:** The application should facilitate the secure transfer of evidence between investigators.
    - **Immutability:** All evidence and transactions recorded on the blockchain must be immutable and inviolable.
    - **Monitoring the chain of custody:** The application should, through smart contracts, automatically monitor the chain of custody for each piece of evidence, recording the history of transfers, participants and timestamps.

- **Non-functional requirements**

- **Security and privacy:** The application should ensure the confidentiality and privacy of evidence and investigators' data through encryption mechanisms and access controls.
- **Scalability:** Themis system (ThemisChain & ThemisApp) should be scalable to accommodate an increasing number of investigators, evidence and transactions without impacting performance.
- **Reliability:** Themis system should maintain high availability and reliability, minimizing potential downtime and ensuring uninterrupted access to case data and evidence.
- **Data integrity:** Themis system should ensure the ongoing integrity of evidence and transactions, protecting them from data corruption or falsification.

At the code level, the application communicates with the nodes of the private blockchain using a “library” of the programming language “Javascript”, while the “Solidity” programming language was used to write the smart contracts. The “Node.js” runtime environment played a central role in the execution of the code and provided sub-applications that streamlined the required processes.

In a more detailed context, the connection to the blockchain is achieved through a “REST API”, which returns a “JSON” (JavaScript Object Notation) file that contains all the elements that make up a smart contract. The development of modern decentralized applications often involves the use of specific programming libraries containing ready-made functions and methods, facilitating communication with the blockchain and in particular the encoding and decoding of “JSON” files. During the development of the “Themis” application, the library “Web3.js” of the Javascript programming language was utilized for this purpose.

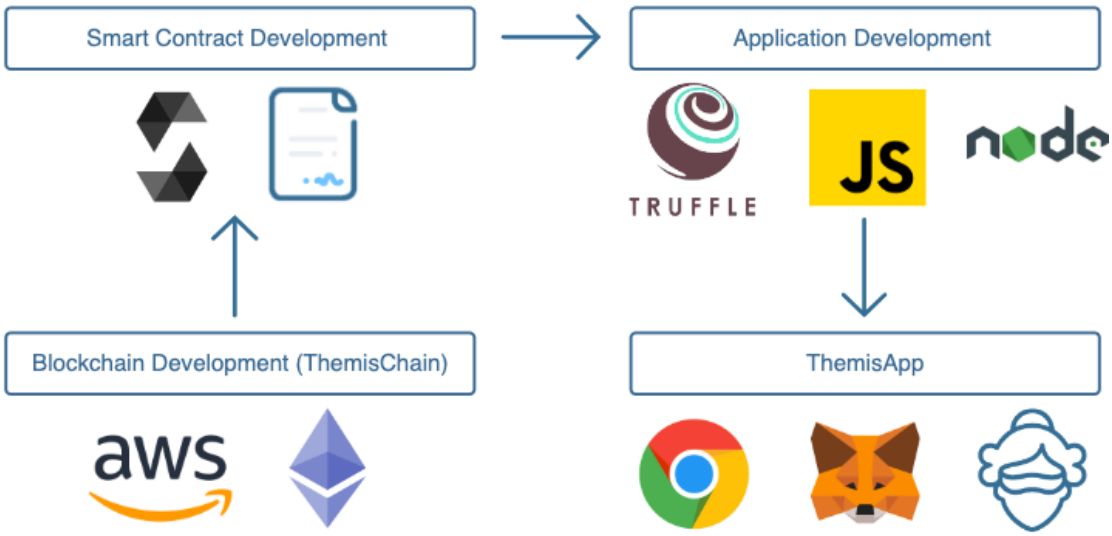

Figure 57 - Simplified architecture of the "Themis" application

7.2. Technical parts

The technologies and applications used during the development of the "Themis" system (ThemisChain & ThemisApp) are:

- **AWS (Amazon Web Services):** Cloud platform that hosts the private blockchain network.
- **Geth (Go Ethereum):** Command-line interface that allows running full Ethereum nodes, mining the Ethereum cryptocurrency and executing smart tokens.
- **Ganache:** Application that provides a local Blockchain for developing applications targeted at the Ethereum Blockchain.
- **Truffle:** Suite of tools for developing "Web3" applications.
- **Remix IDE:** Open source tool specialized in writing and developing smart contracts in the "Solidity" programming language.
- **Solidity:** Object-oriented programming language for developing smart contracts on the Ethereum blockchain.
- **Javascript:** Object-oriented programming language for enriching and enhancing the dynamics of the graphical environment of the application.
- **Web3.js:** Collection of Javascript libraries that allow the development of "Web3" applications and interaction with an Ethereum node.
- **Bootstrap:** Open source front-end framework used to create modern websites and web applications.

- **Node.js:** Software development platform that allows Javascript code to be executed outside of a web browser environment.
- **Chai:** Behavioral-Driven Development / Test-Driven Development (BDD/TDD) library for "Node.js", which can be combined with any Javascript test framework.
- **Lite-server:** Node.js module that acts solely as a server for the development of an application.
- **Metamask:** Software wallet in the form of a web browser extension, used to interact with blockchain.
- **Visual Studio Code:** Code editing application.
- **Visual Paradigm:** Modeling and project management application.
- **Ubuntu:** Operating system on which the private blockchain network was developed and runs.
- **macOS:** Operating system on which the app was developed.

7.2.1. Installation guide for necessary applications (Windows)

Below are instructions for installing the necessary technical parts to run the application code on a "Windows 10" operating system. Specifically, the installation of Node.js, its modules and the "Metamask" extension in the "Google Chrome" browser will be explained. For a better understanding of the current and the following subchapter, the third person singular will be used along with the first person plural.

1. Node.js

The Node.js installation file is downloaded from the official website via the link: https://nodejs.org/en/download/ (Figure 58). For the steps of this guide, and for the successful execution of the "Themis" application, version "18.17.1" was downloaded.

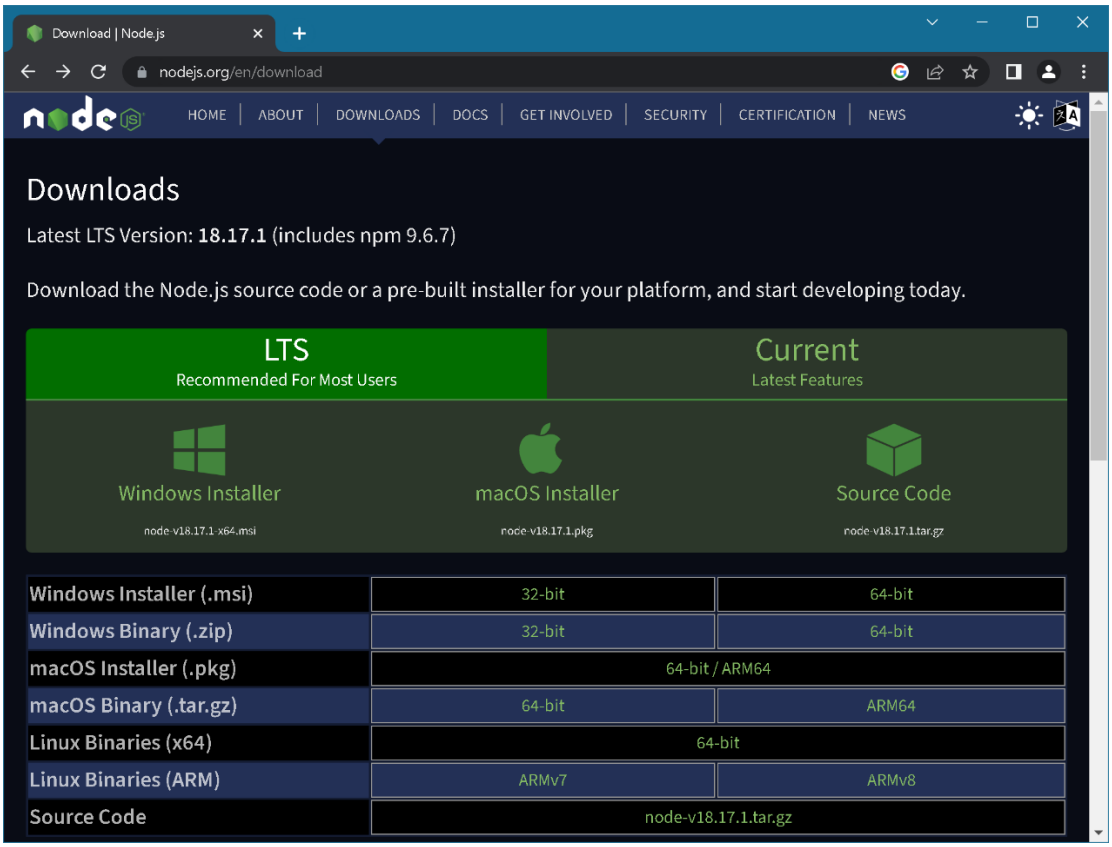

Figure 58 - Download site of "Node.js"

After the download is complete, we run the downloaded file and proceed with the installation process until we reach the step in Figure 59, where the checkbox selection is necessary to install additional basic tools.

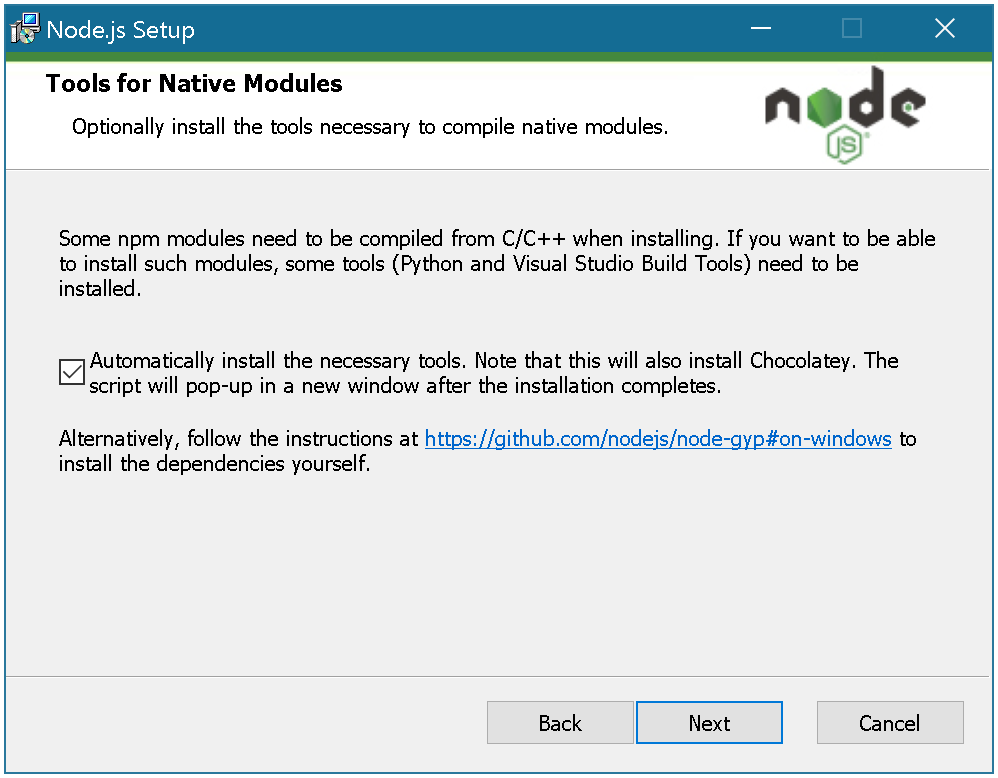

Figure 59 - Required option when installing "Node.js"

With this checkbox selection, two consecutive command line windows (CMDs) will be displayed, prompting the user to press any key to continue the process (Figures 60 and 61). These windows will be replaced by a "Windows PowerShell" window, which, upon completing the automated tool installation process, will look like the one in Figure 62.

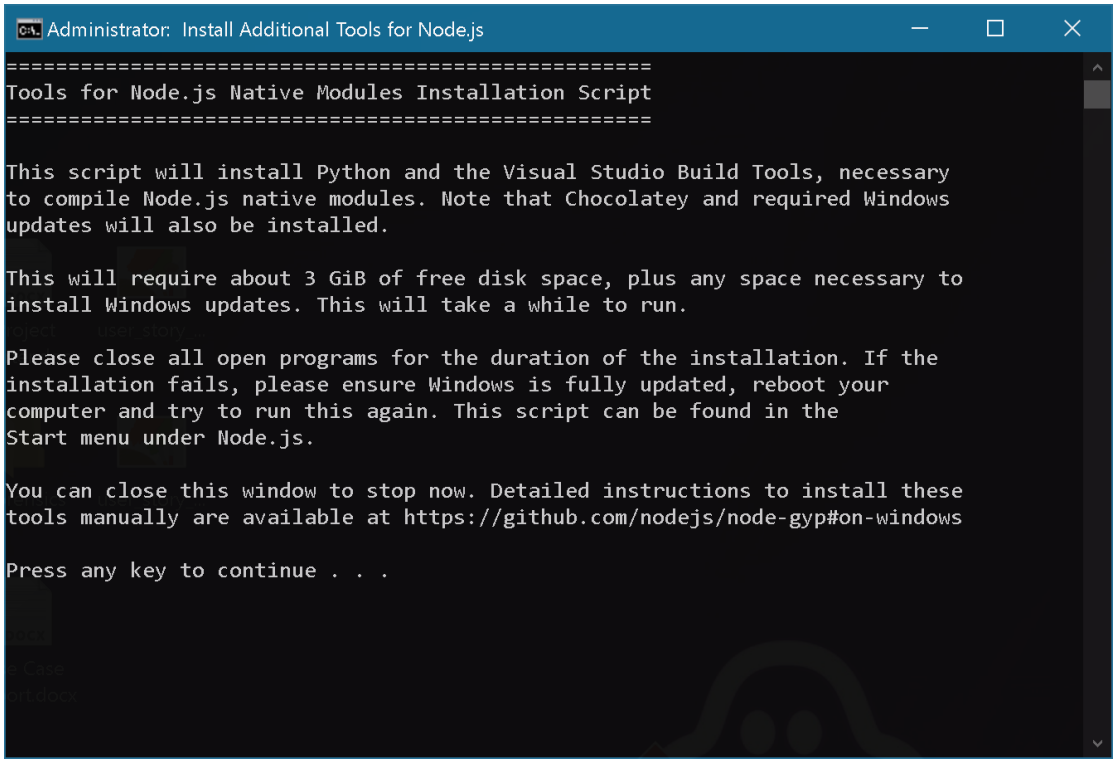

Figure 60 - Installation window for additional "Node.js" tools (1/2)

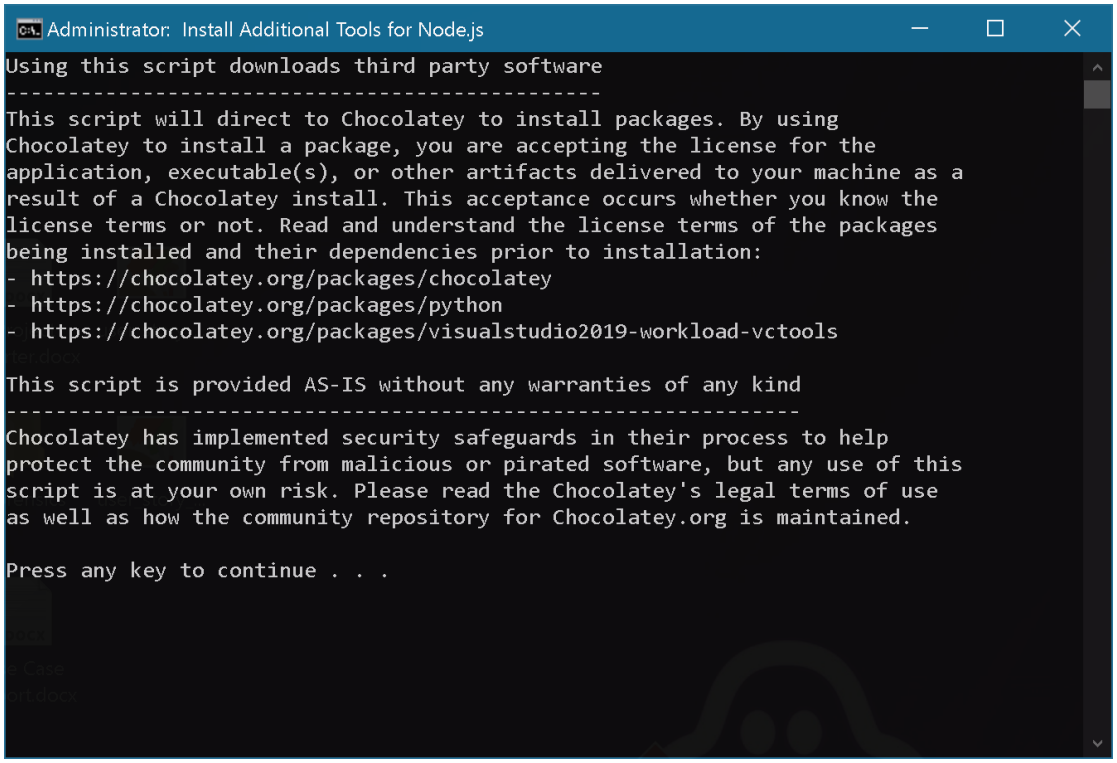

Figure 61 - Installation window for additional Node.js tools (2/2)

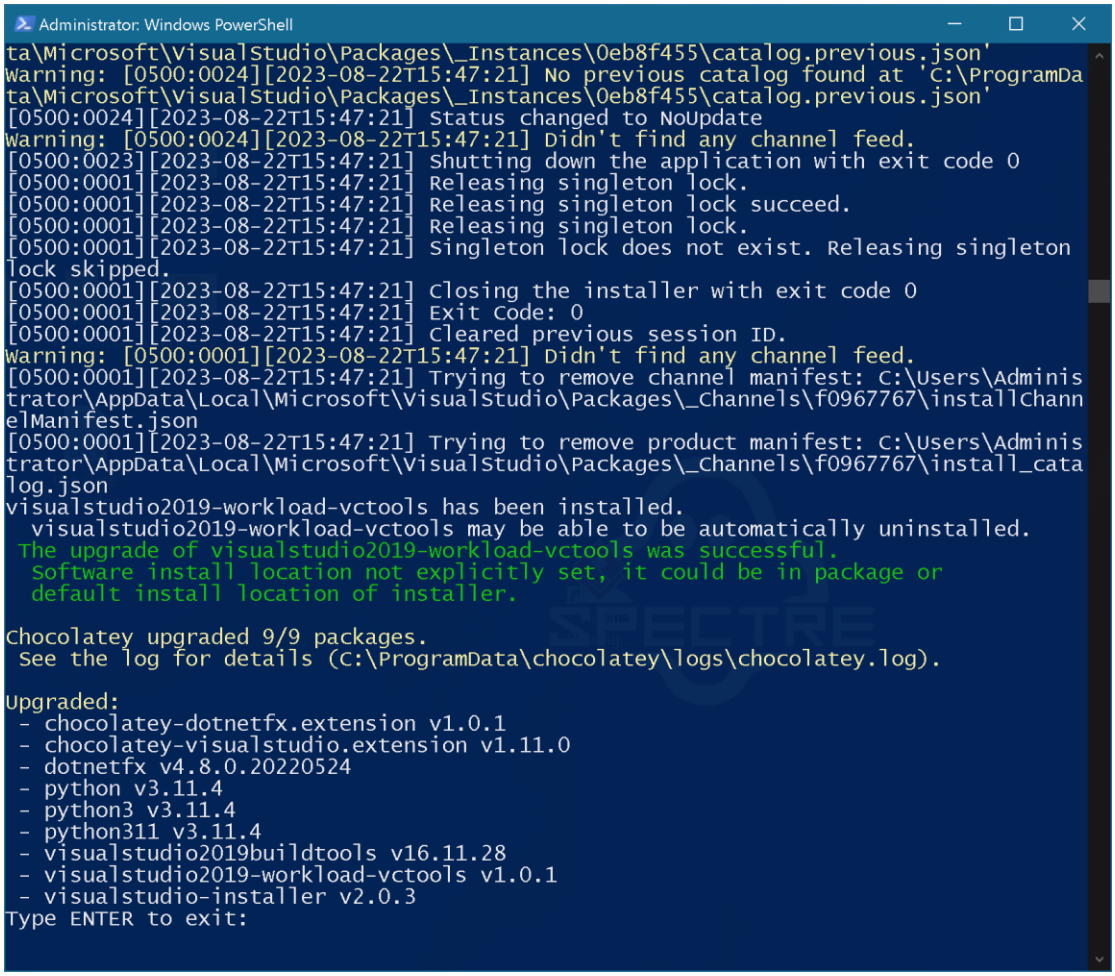

Figure 62 - Completion of the additional Node.js tools installation

2. Metamask

To install Metamask we need to create a wallet account, add the private network "ThemisChain" and finally, enter the account of the first node of the network, through its private key.

To get started, we download Metamask either through its official website: https://metamask.io/download/ (Figure 63), or through the "Google Chrome" browser web store by visiting: https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn (Figure 64).

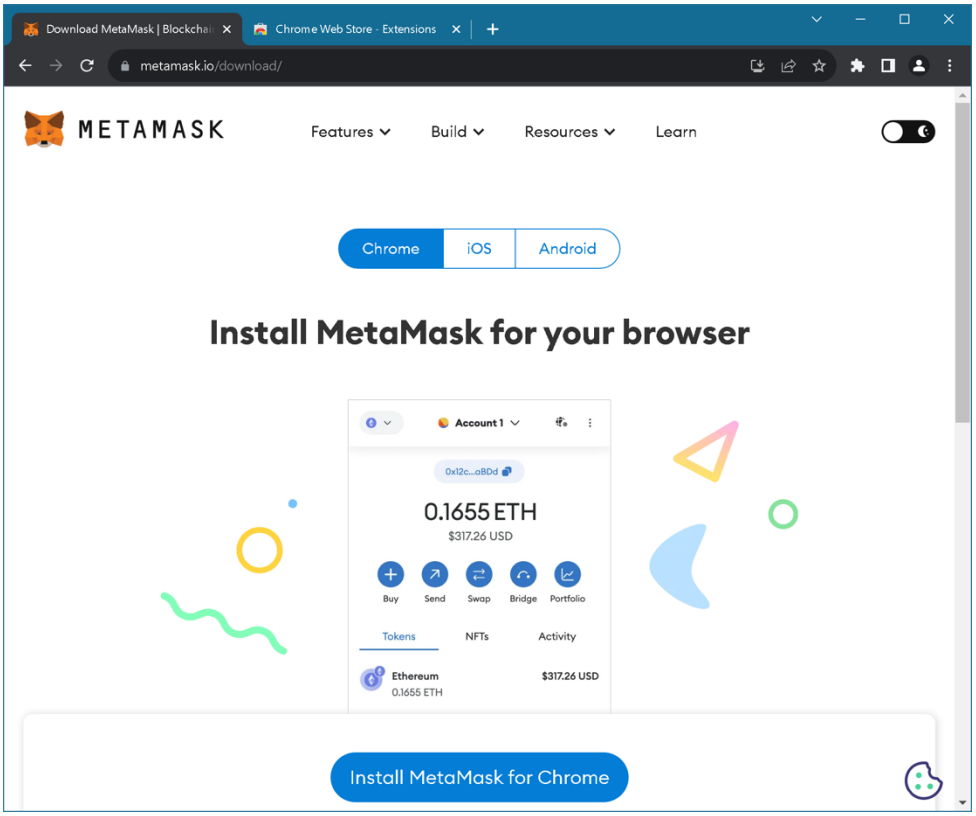

Figure 63 - Official website of Metamask

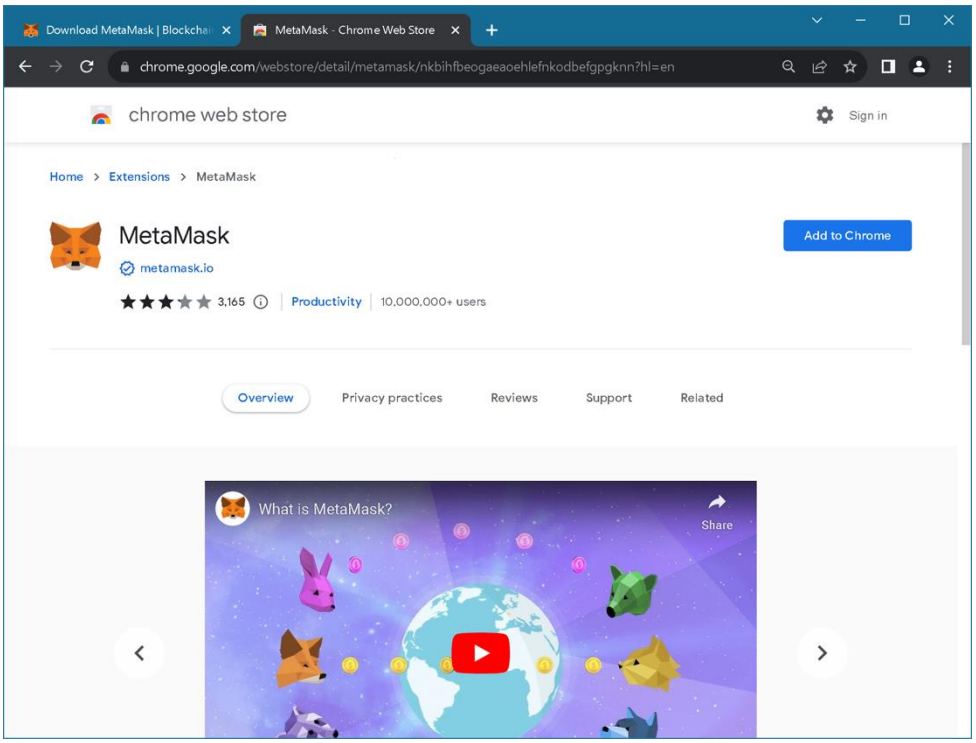

Figure 64 - Metamask in the "Google Chrome" web store

Once the download and installation of Metamask is complete, we choose to create a new wallet account (Figure 65) and enter the desired password (Figure 66).

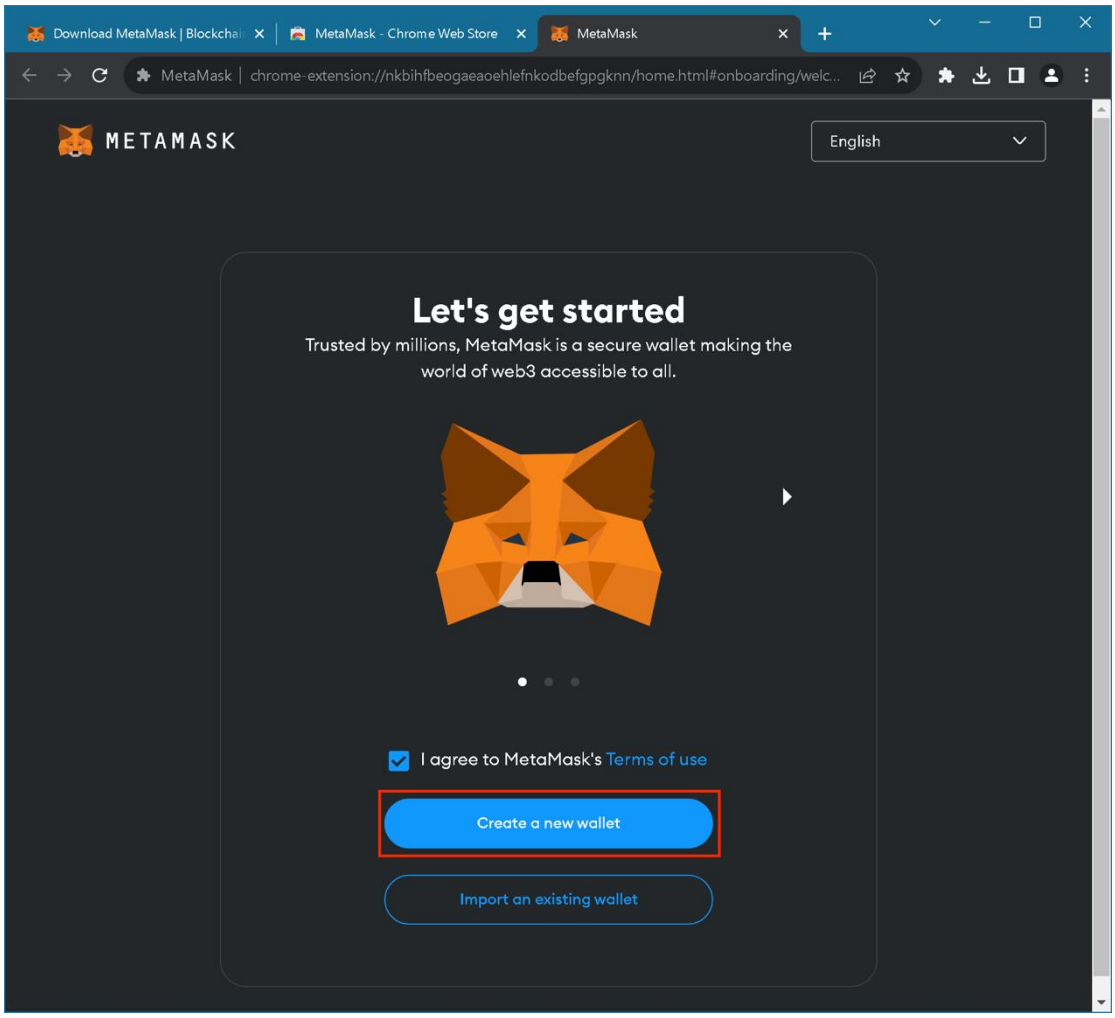

Figure 65 - Option to create a new Metamask wallet account

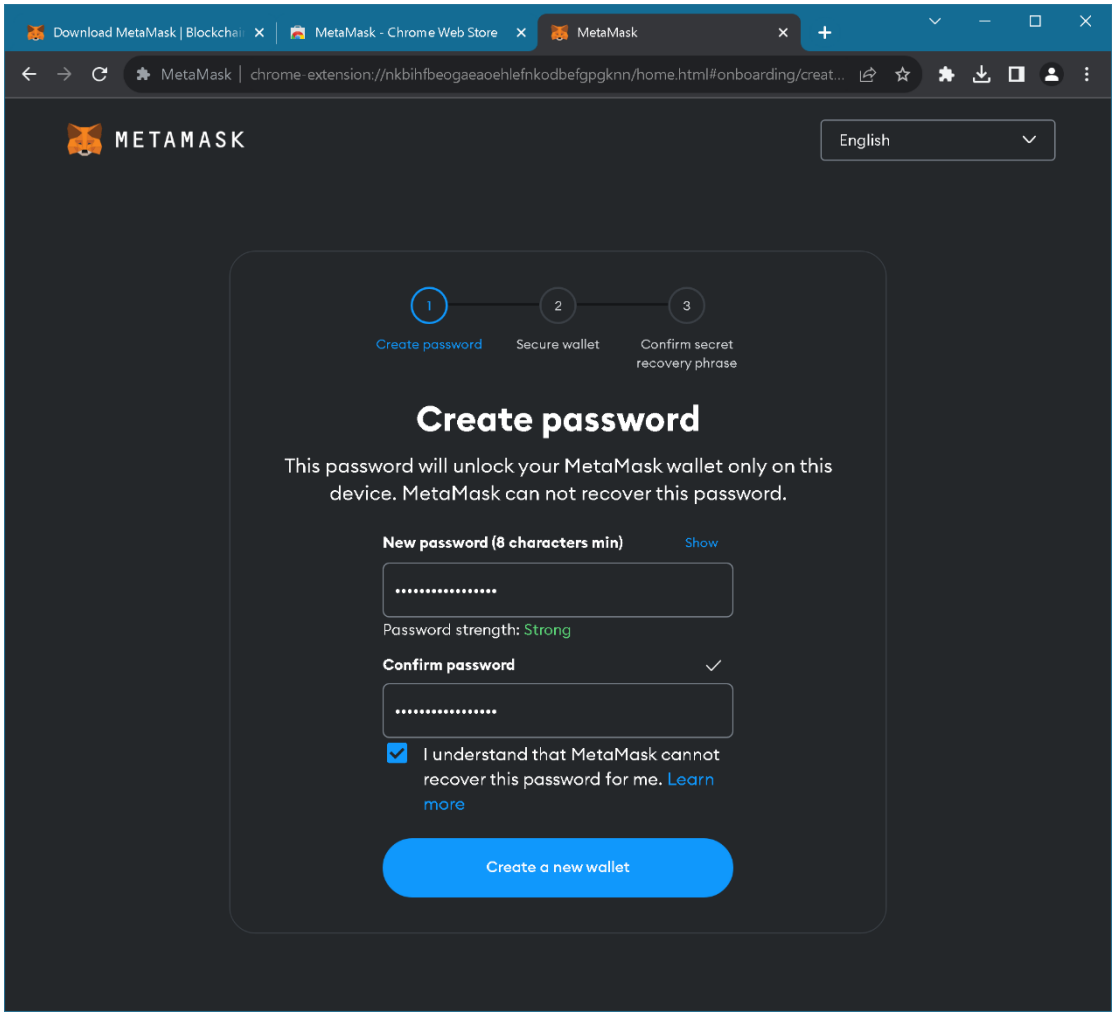

Figure 66 - Entering the password of a new Metamask wallet account

In the next step, we are asked to secure our account using a unique secret phrase. However, as this wizard was created for demonstration purposes, we skip this step (Figure 67).

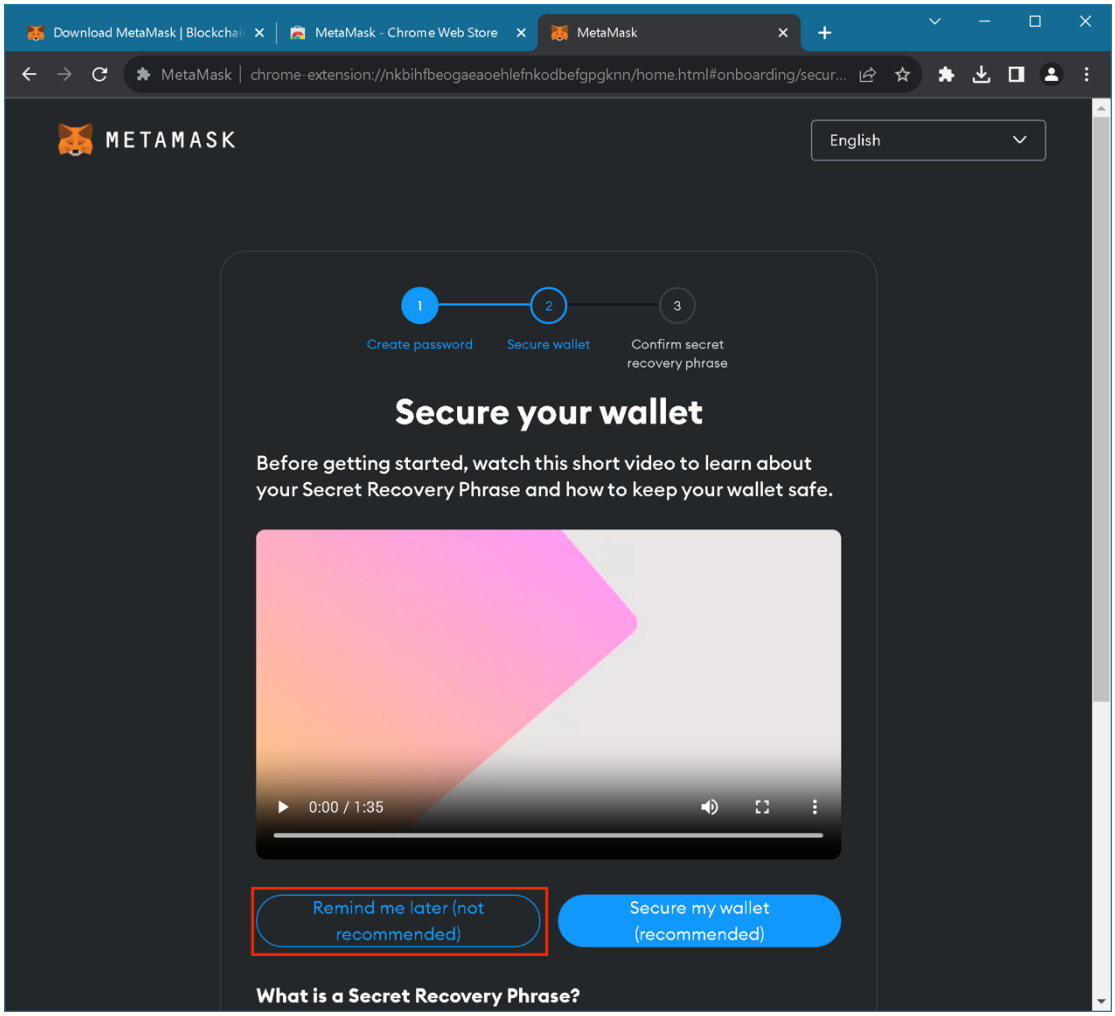

Figure 67 - Securing the account using a unique secret phrase

After completing the procedure, the home page of our wallet appears, where we go to and select “Settings” from the menu (Figure 68). From there, we select the “Networks” tab on the left, click on the “Add a network” button (Figure 69) and then click on the “Add a network manually” option (Figure 70).

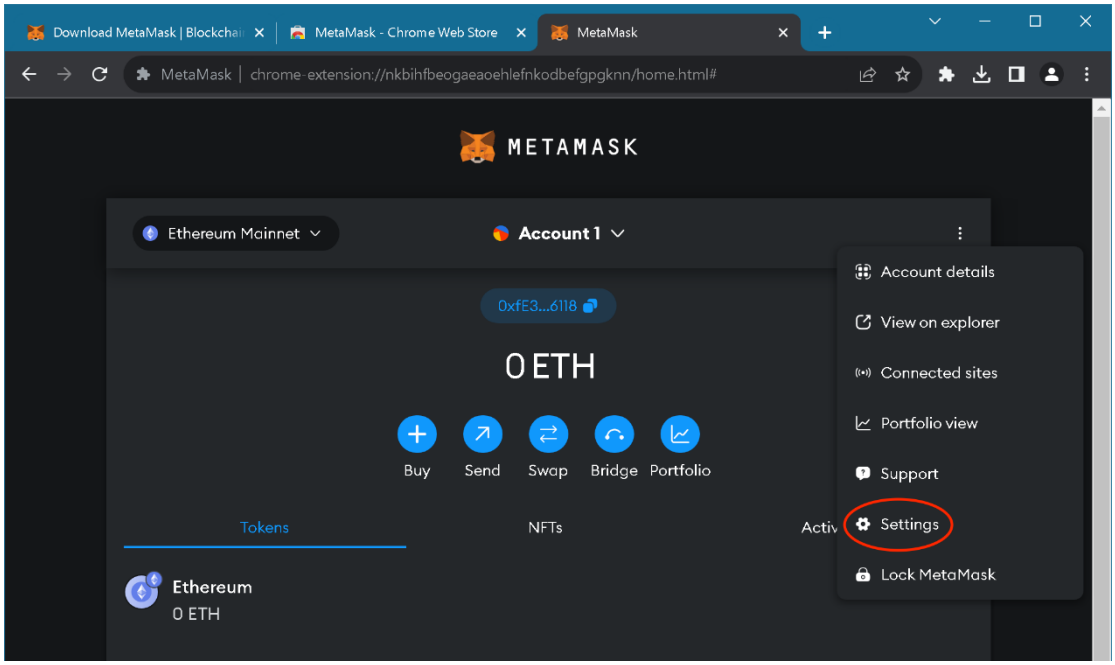

Figure 68 - The "Settings" option of the Metamask menu

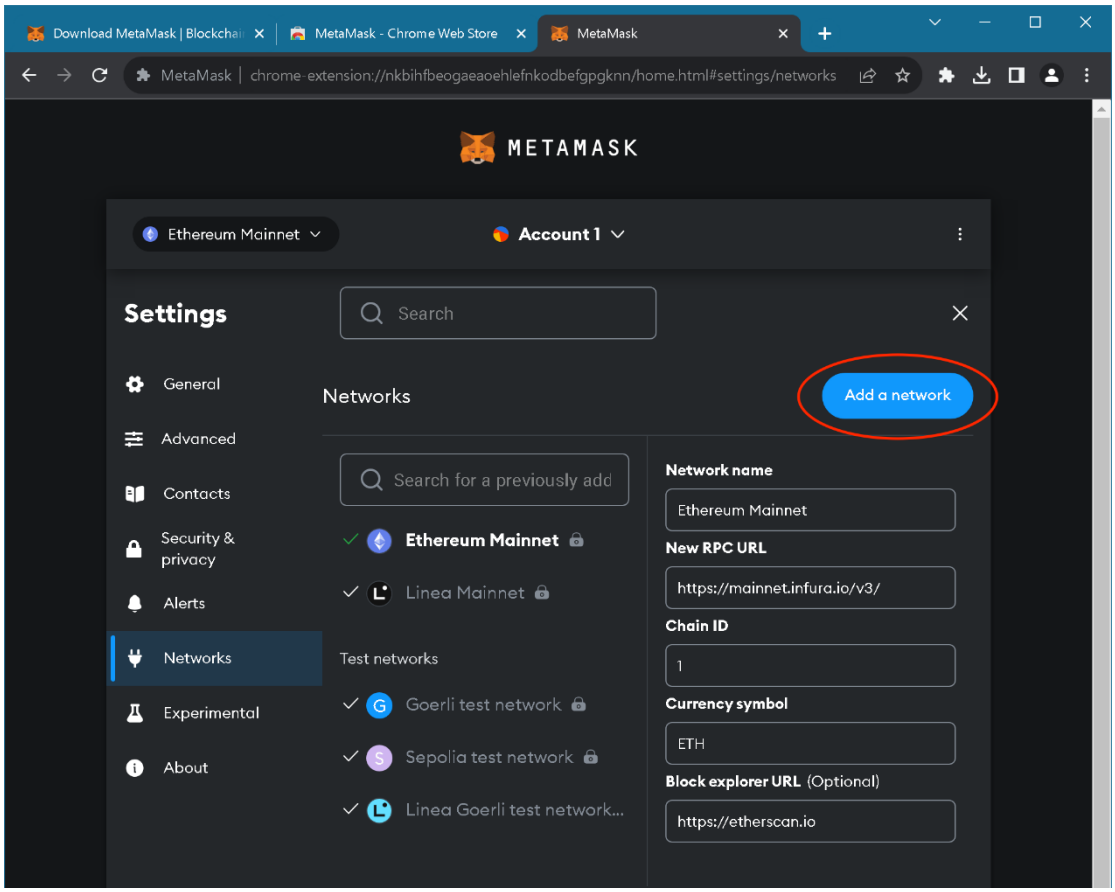

Figure 69 - The "Add network" button

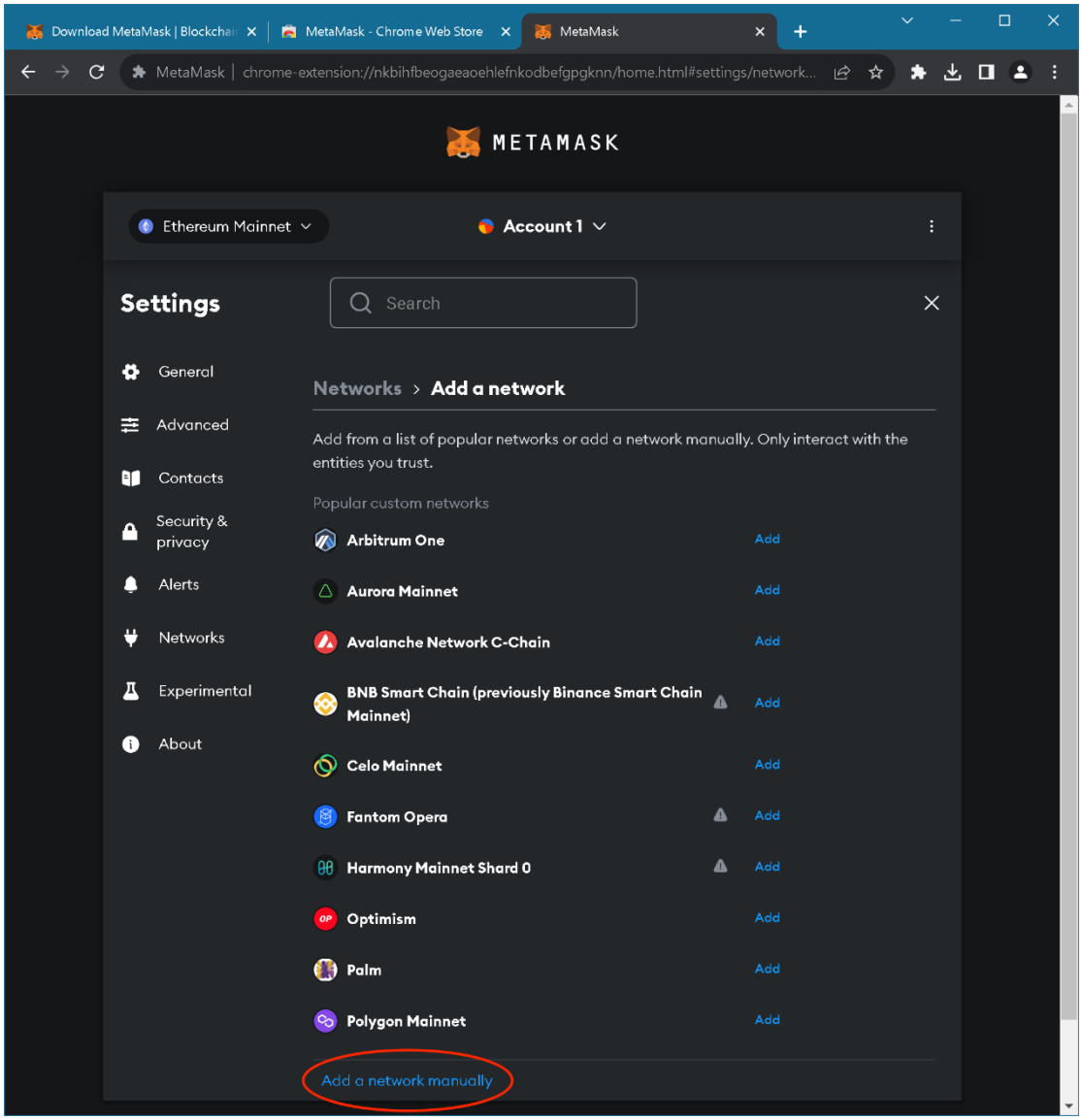

Figure 70 - The "Add network manually" option

Next, we enter the following network details "ThemisChain" (Figure 71):

- **Network name:** ThemisChain
- **New RPC URL:** http://18.232.0.78:8545
- **Chain ID:** 1997
- **Currency symbol:** ETH

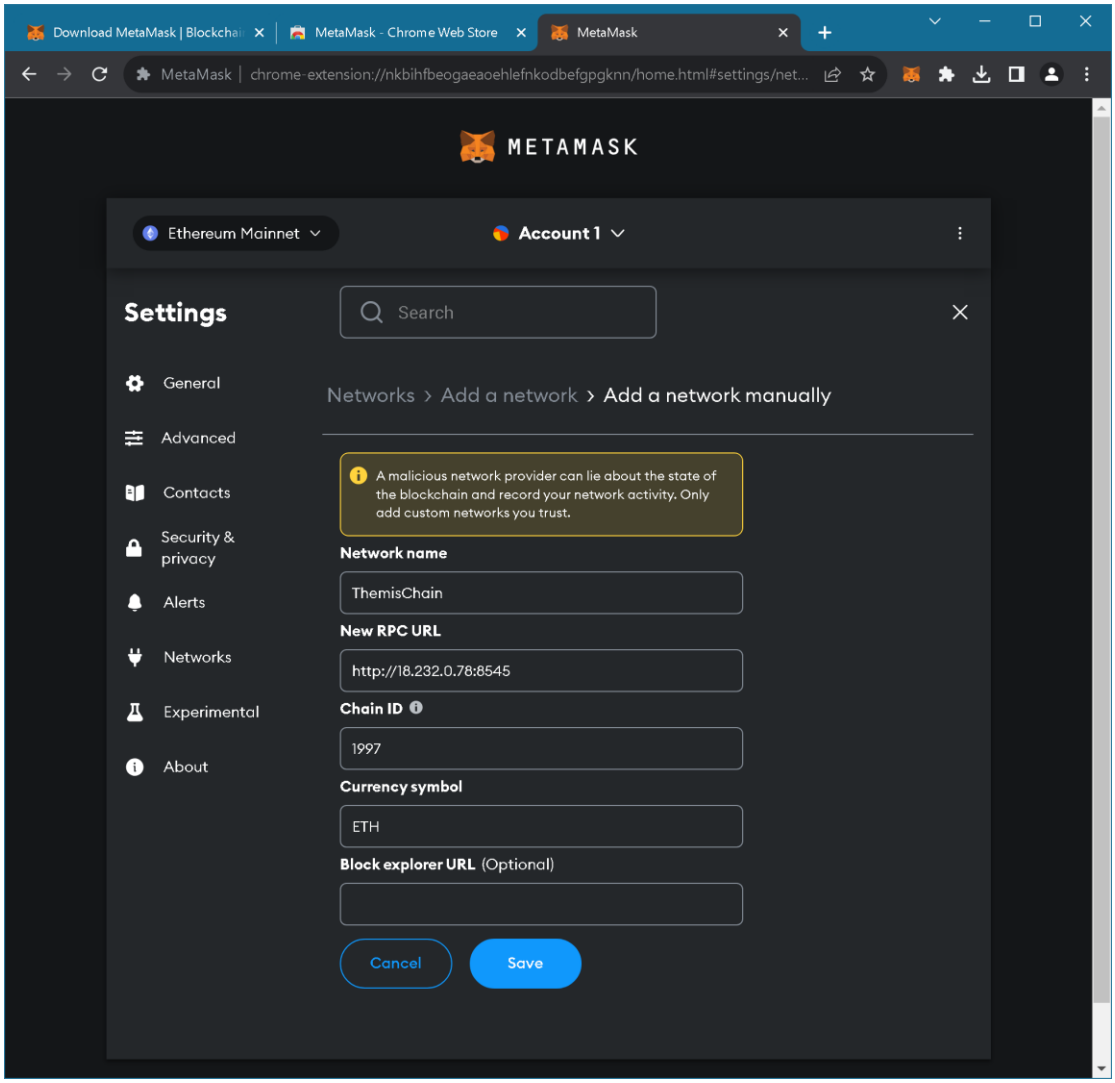

Figure 71 - The details of the "ThemisChain" network

To import the account of the first node in the network into Metamask, follow the instructions mentioned above in subchapter 6.1 (see page 79). If the connection to the network and the import of the account was successful, then the account and its balance in "ETH" will be displayed in Metamask (Figure 72).

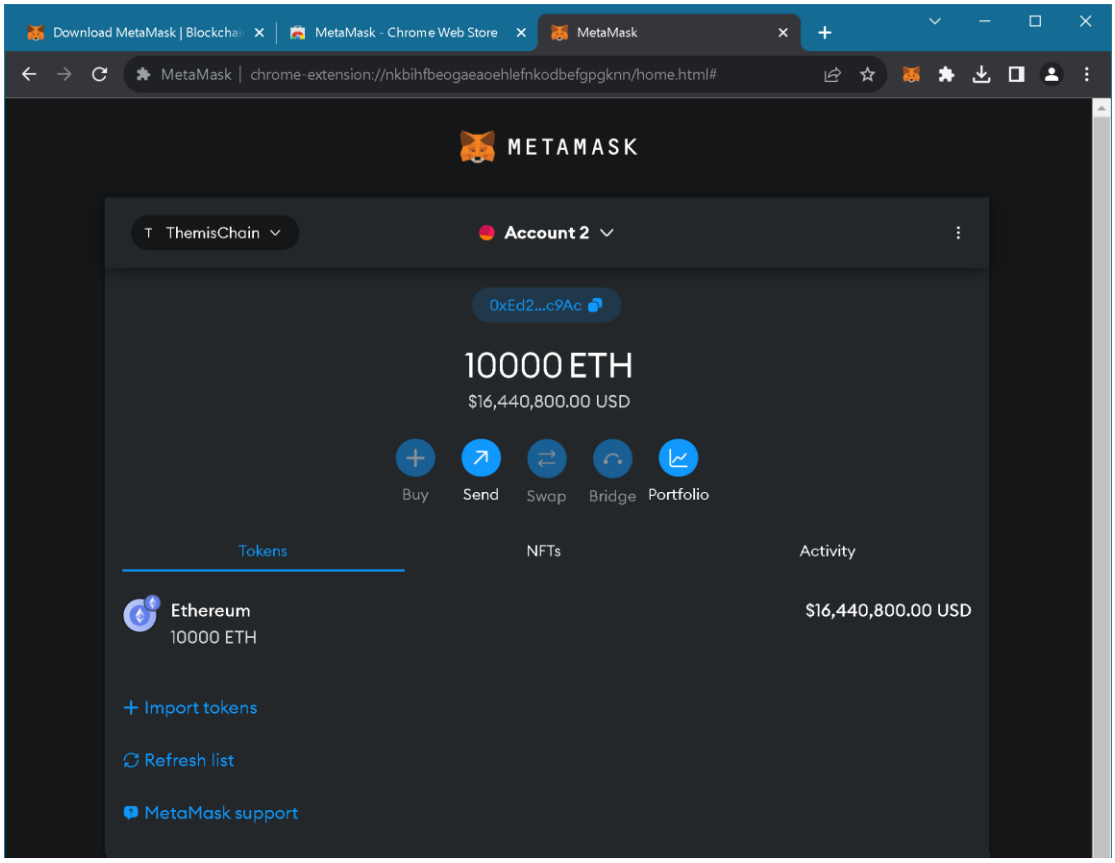

Figure 72 - The first node account and its balance in "ETH"

3. Node.js Modules

Node.js modules are installed using the "*npm install*" command, in which the modules are reinstalled based on the contents of the "package.json" file, which is located in the application folder.

Similarly to the above-mentioned installation method, the files accompanying the project include the folder "Application_files (install)", for which the installation method will be explained below.

3.1. The "npm install" command

Within the "Application_files (Install)" folder, use the "Shift" + right-click combination to display the "Open PowerShell window here" option (Figure 73). After selecting it, a "Windows PowerShell" window appears, in which we proceed to enter the command "npm install" (Figure 74). During the process, it is expected that various messages will be displayed, which we ignore. Furthermore, after its completion (Figure 75), a few vulnerabilities are observed, which we also ignore.

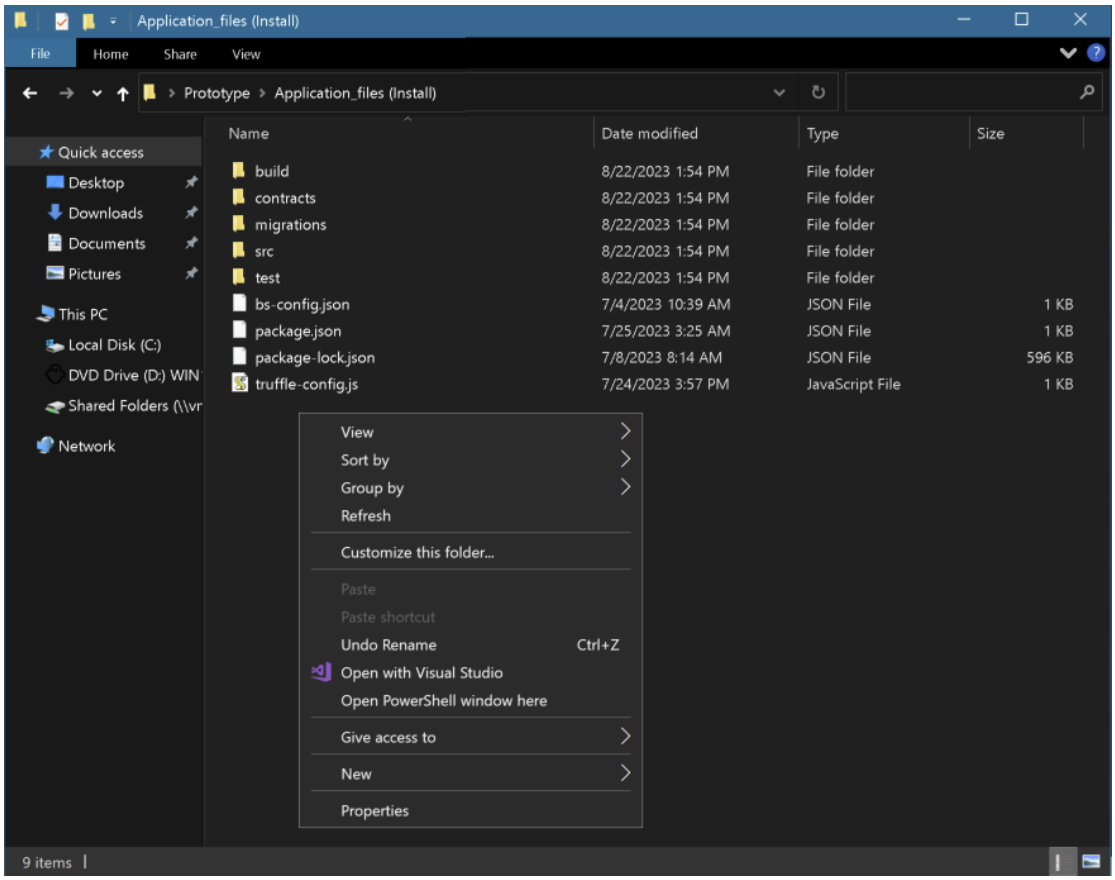

Figure 73 - The "Application_files (Install)" folder

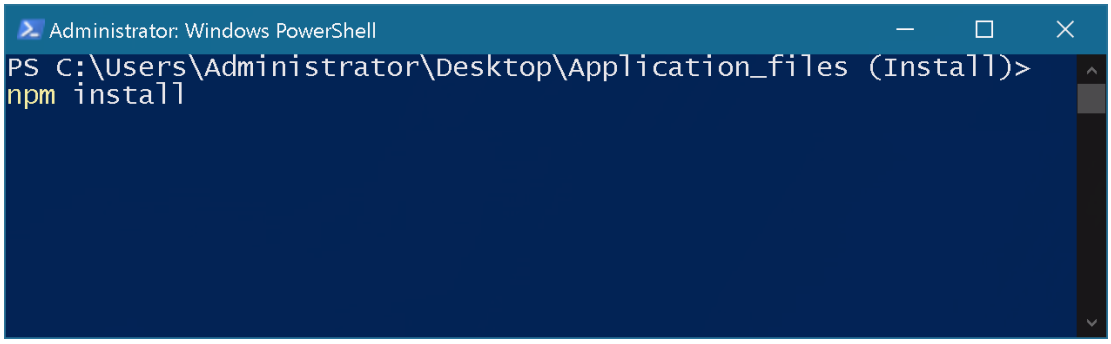

Figure 74 - Running of the "npm install" command

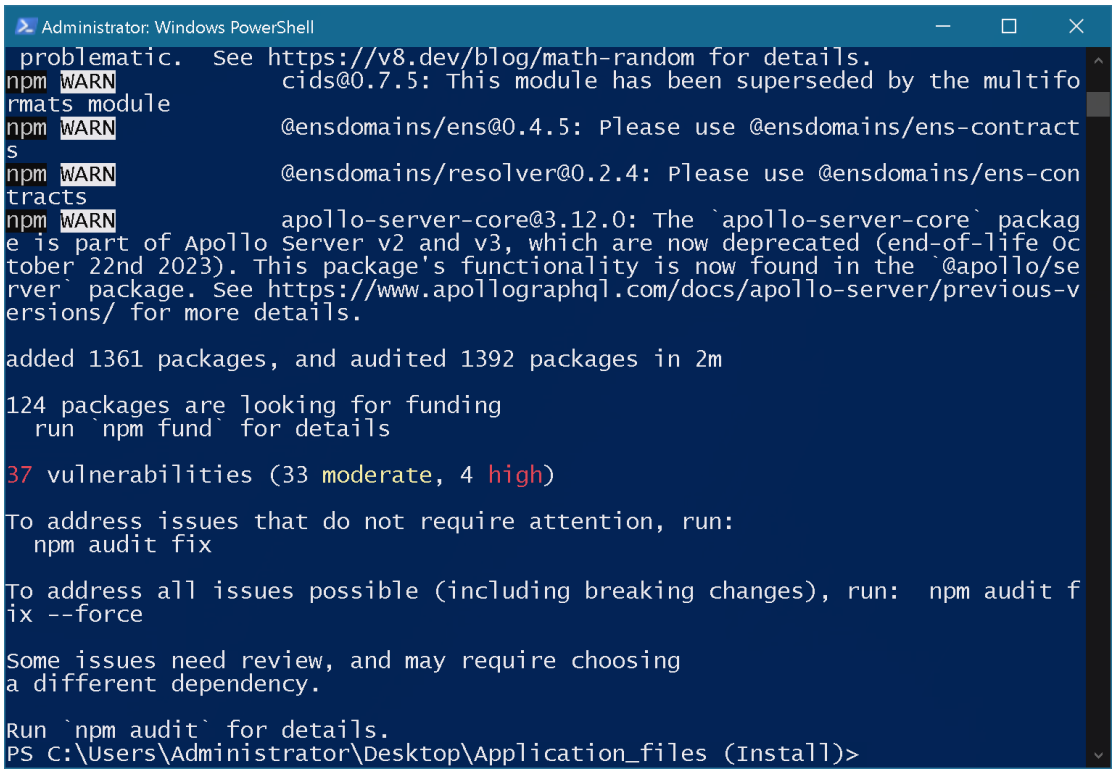

Figure 75 - The results of the "npm install" command

7.3. Use case description

The use case that will be presented below concerns the role of the administrator and his interaction with all the functions of the application. These functions include:

- The display of the investigator's profile
- The addition of a new investigator
- The management of all investigators registered on the blockchain
- The opening of a new case
- The view of active cases assigned to an investigator
- The management of all cases registered on the blockchain
- The addition of an evidence in a case
- The search for evidence in a case
- The transfer of ownership of a piece of evidence
- The view of the chain of custody of an evidence

Having added to Metamask the account of the first node - corresponding to one of the three administrators (see pages 72, 79) - the next step is to activate a local server, using the "lite-server" module of Node.js, on which the code of the application will be executed to be accessible from the web browser, through the address "http://localhost:3000/". Running a "Windows PowerShell" window within the folder containing the application files (see page 100), type the command "npm run dev" (Figure 76). The server that is activated (Figure 77) should remain active throughout the execution of the application.

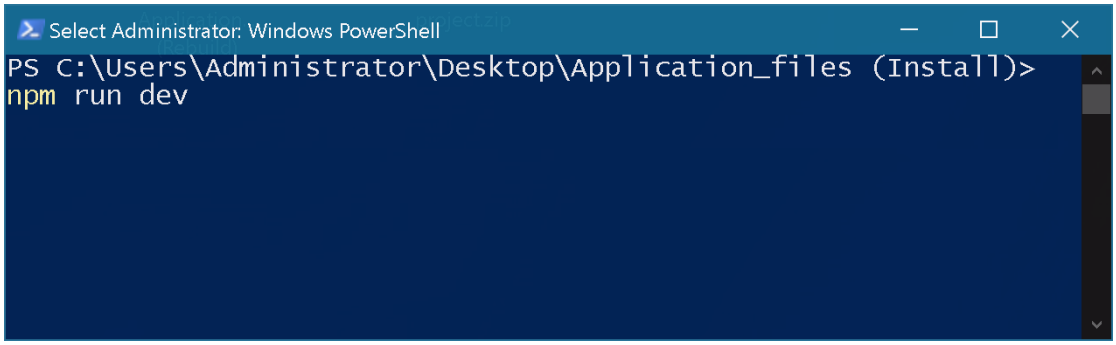

Figure 76 - Running the command "npm run dev"

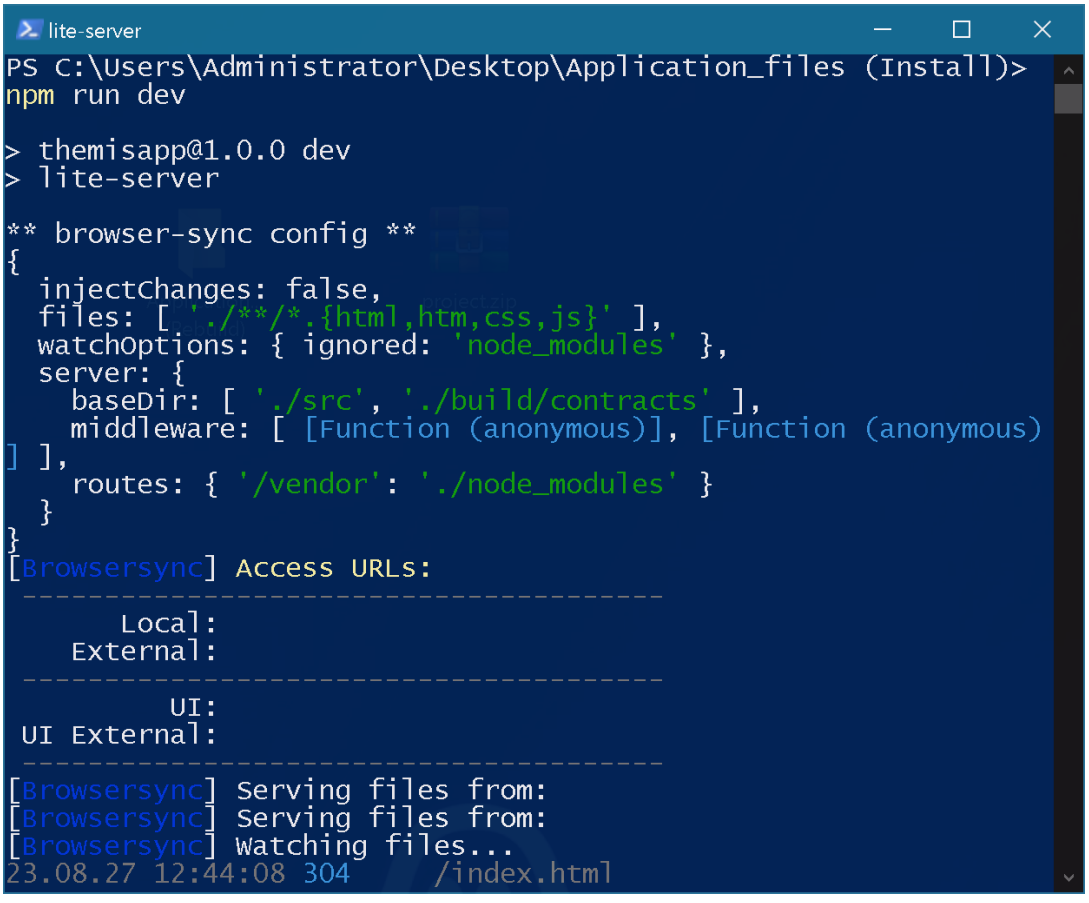

Figure 77 - Running the local server using the "lite-server" module

- **View the investigator's profile**

Once the user goes to the home page of the application (see page 83) and successfully logs in with his wallet account, through the Metamask pop-up window, he is redirected to the page containing the investigator's profile - whether he is a simple investigator or an investigator with administrator rights - which contains his identity details such as: his full name, email, his blockchain wallet address and his active cases (Figure 78).

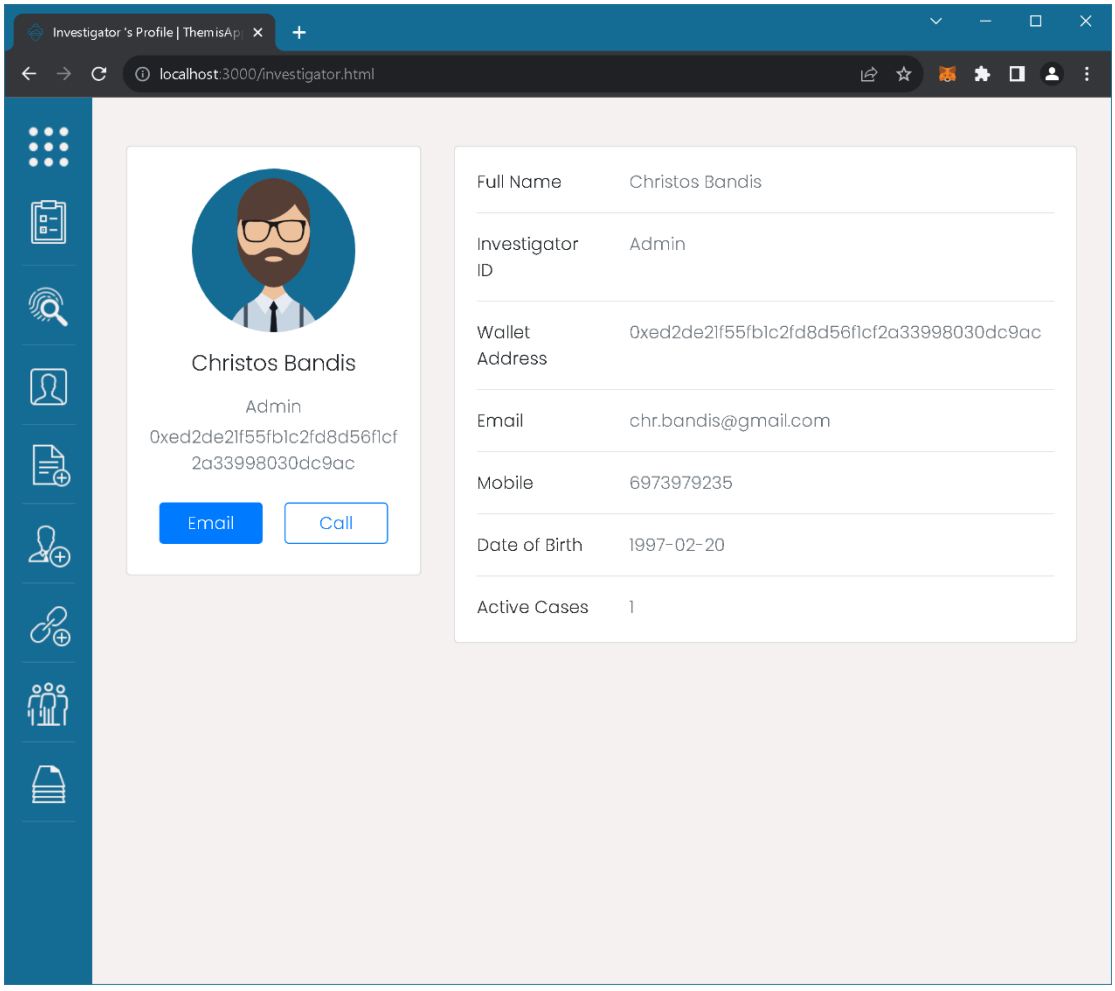

Figure 78 - The page containing the investigator/administrator profile

- **Adding a new investigator**

The next function of the application to be presented is the addition of a new investigator to the application and thus to the blockchain. Through the menu on the left of the application, the administrator goes to the page "Add a new Investigator", where he fills in the form with the appropriate identity details of the new investigator (Figure 79).

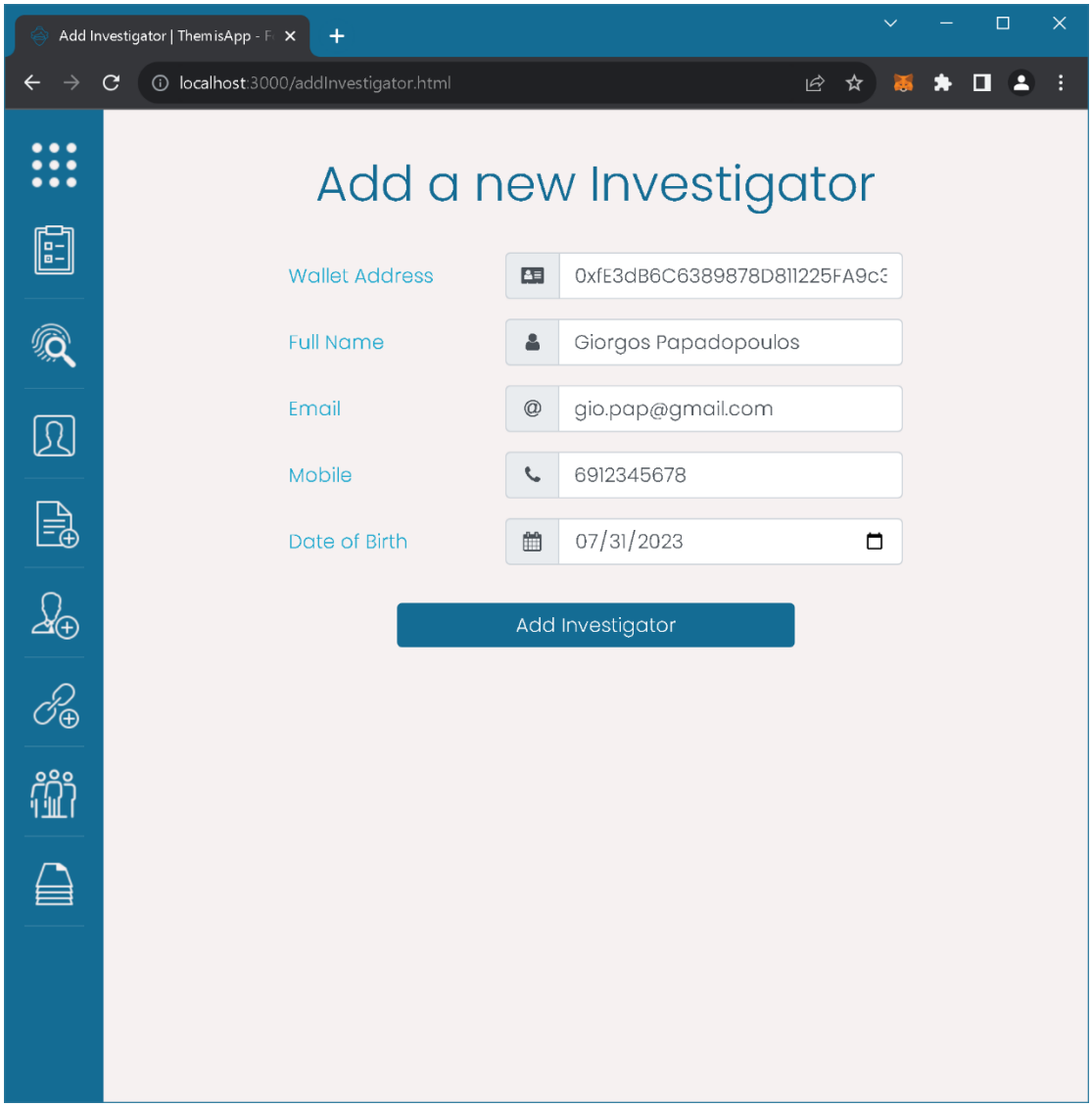

Figure 79 - The "Add a new Investigator" page

Clicking on the "Add Investigator" button displays the Metamask transaction approval window, which confirms what was mentioned in Chapter 6 (see page 59), that the transaction processing has zero cost, despite the large size of the data it contains (Figure 80). The same is true for the entire range of transactions within the application.

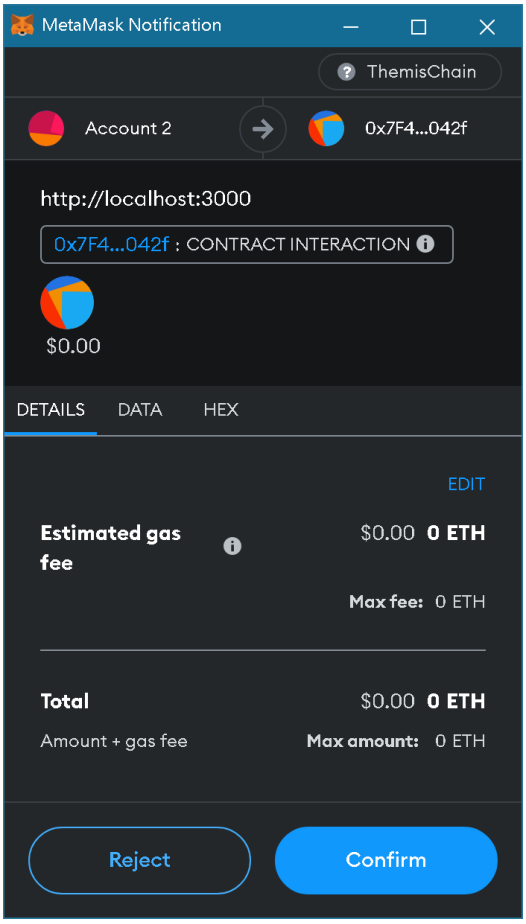

Figure 80 - Confirmation of zero transaction costs within the application

- **Management of all investigators registered on the blockchain**

An administrator has the right to check all the investigators registered on the blockchain, from the "Manage Investigators" page (Figures 81 and 82). The table presented within the page contains details of the investigators' identity, such as: the name, the ID (for administrators it is "Admin", while for investigators it is a random four-digit number) and the wallet address of the administrator who registered the identity.

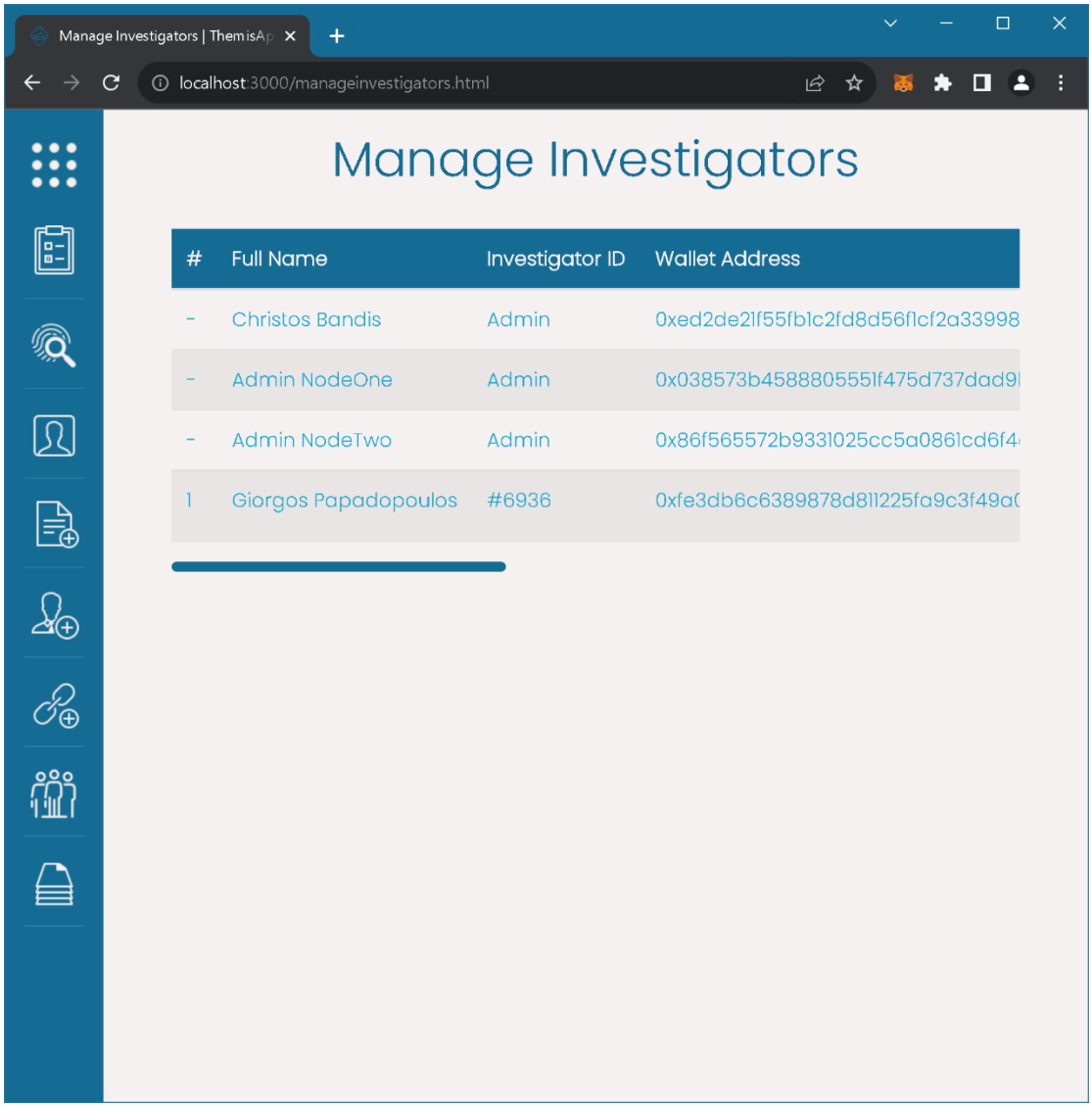

Figure 81 - The "Manage Investigators" page (1/2)

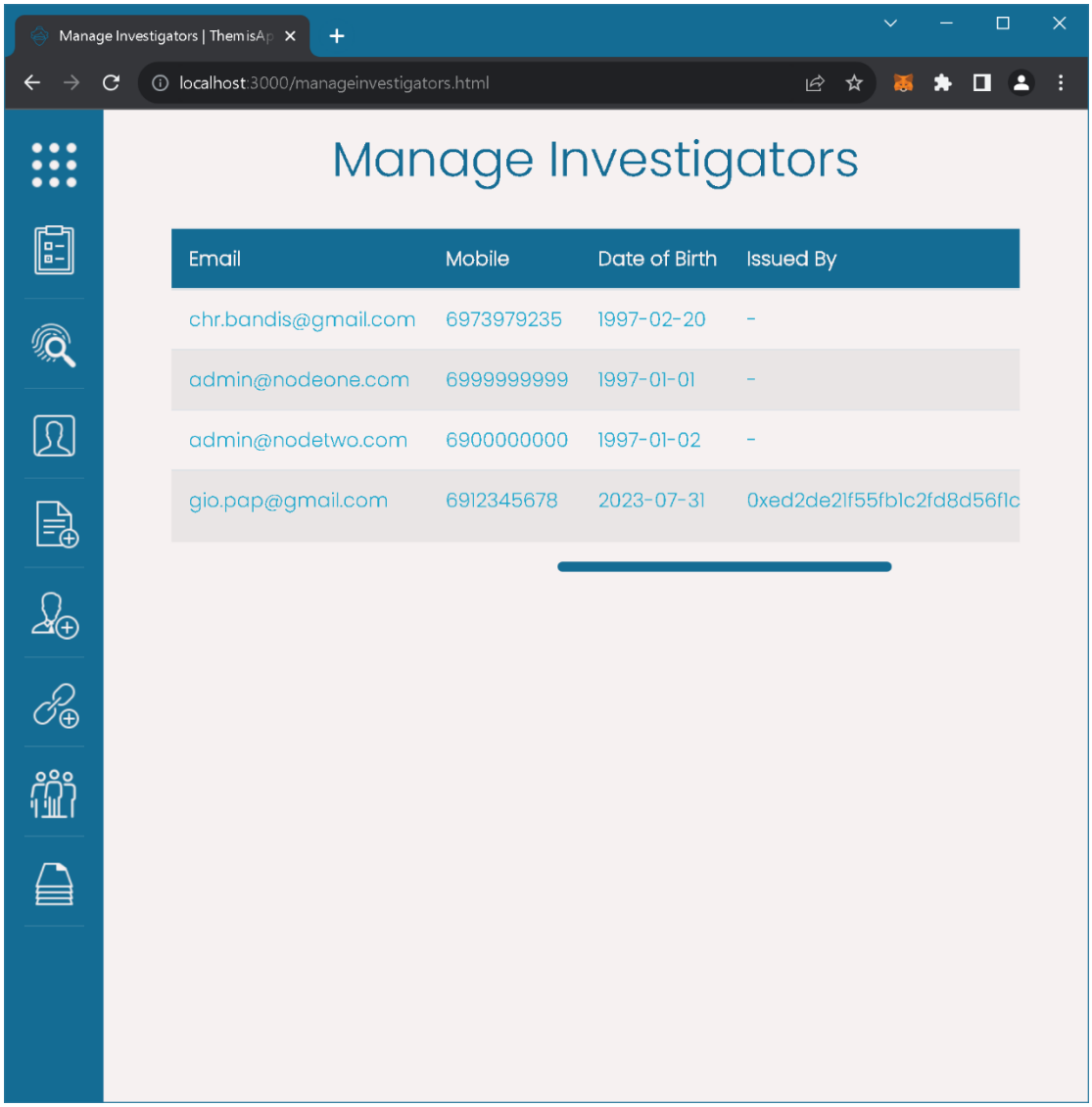

Figure 82 - The "Manage Investigators" page (2/2)

- **Opening a new case**

The opening of a new case is done through the form located at "Open a new case", which asks for the name, description and wallet address of the investigator to whom the case will be assigned (Figure 83). Once the process is complete, two additional buttons appear just below the "Open Case" button: the "Add case evidence" and "Add case investigator" buttons (Figure 84). At this point in time, neither of these two actions will be performed, as they will be performed individually later on.

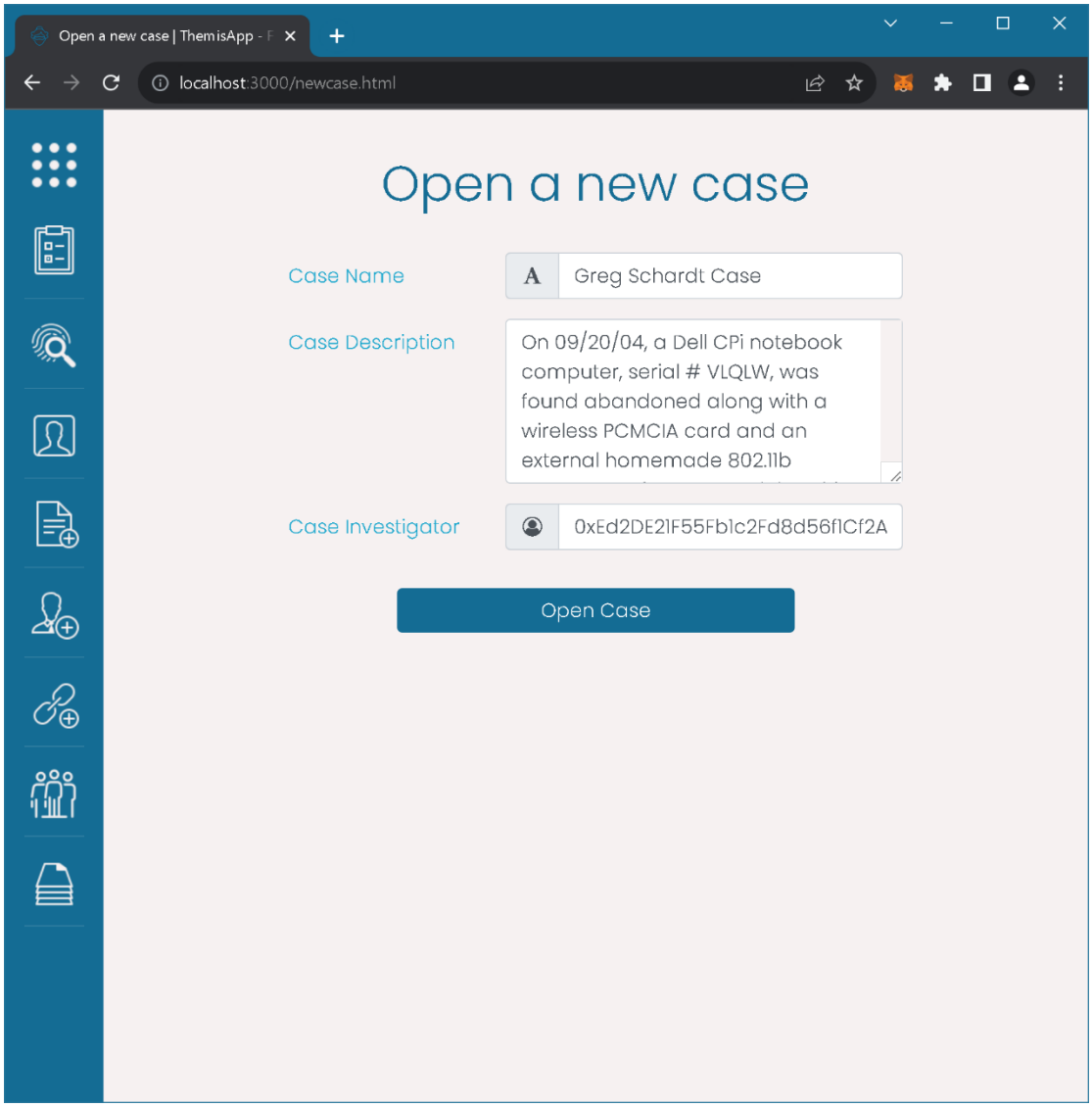

Figure 83 - The "Open a new case" page (1/2)

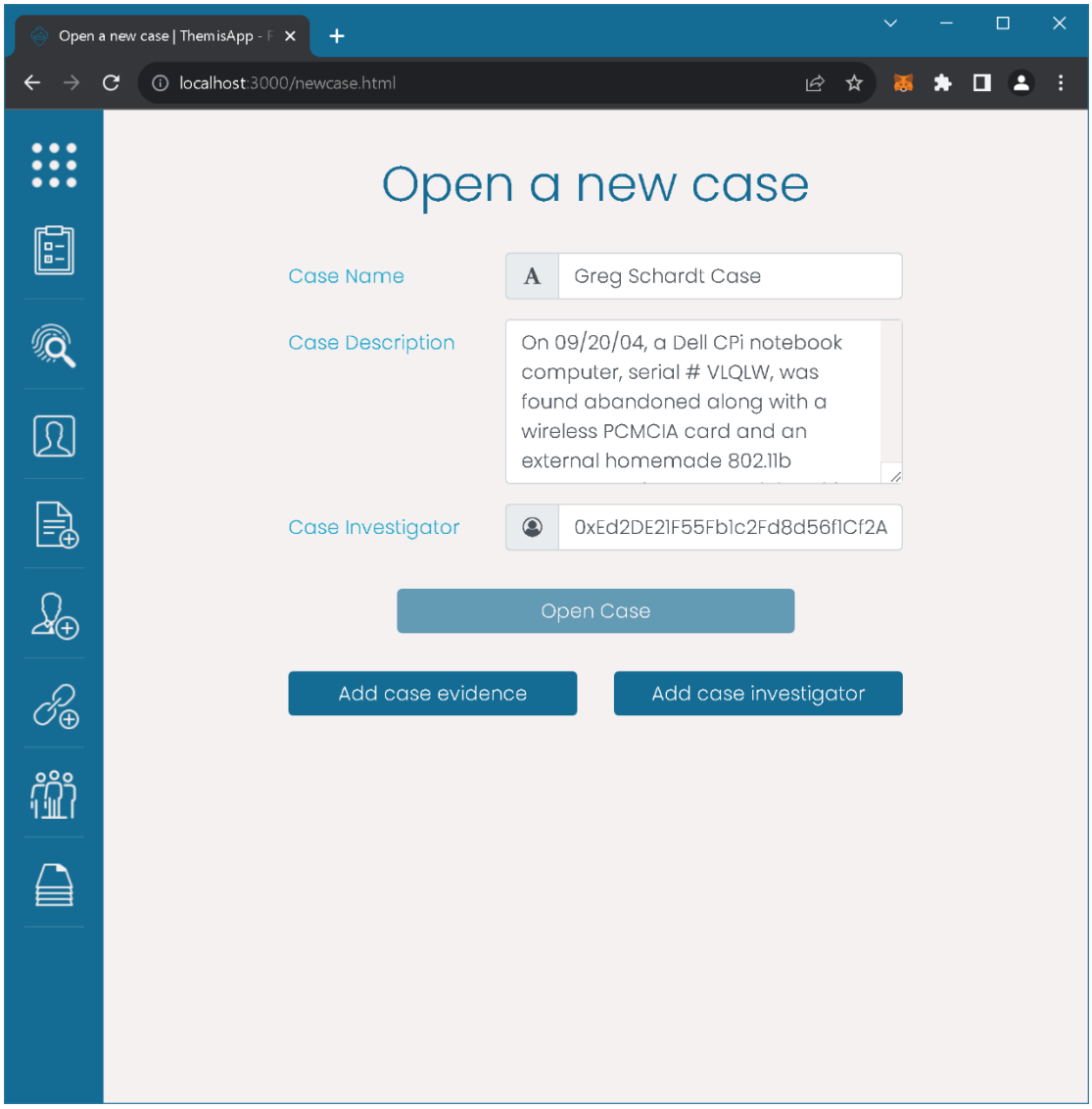

Figure 84 - The "Open a new case" page (2/2)

- **View the active cases assigned to an investigator**

An investigator-administrator has the possibility to monitor his active cases on the "Active Cases" page (Figure 85 and 86).

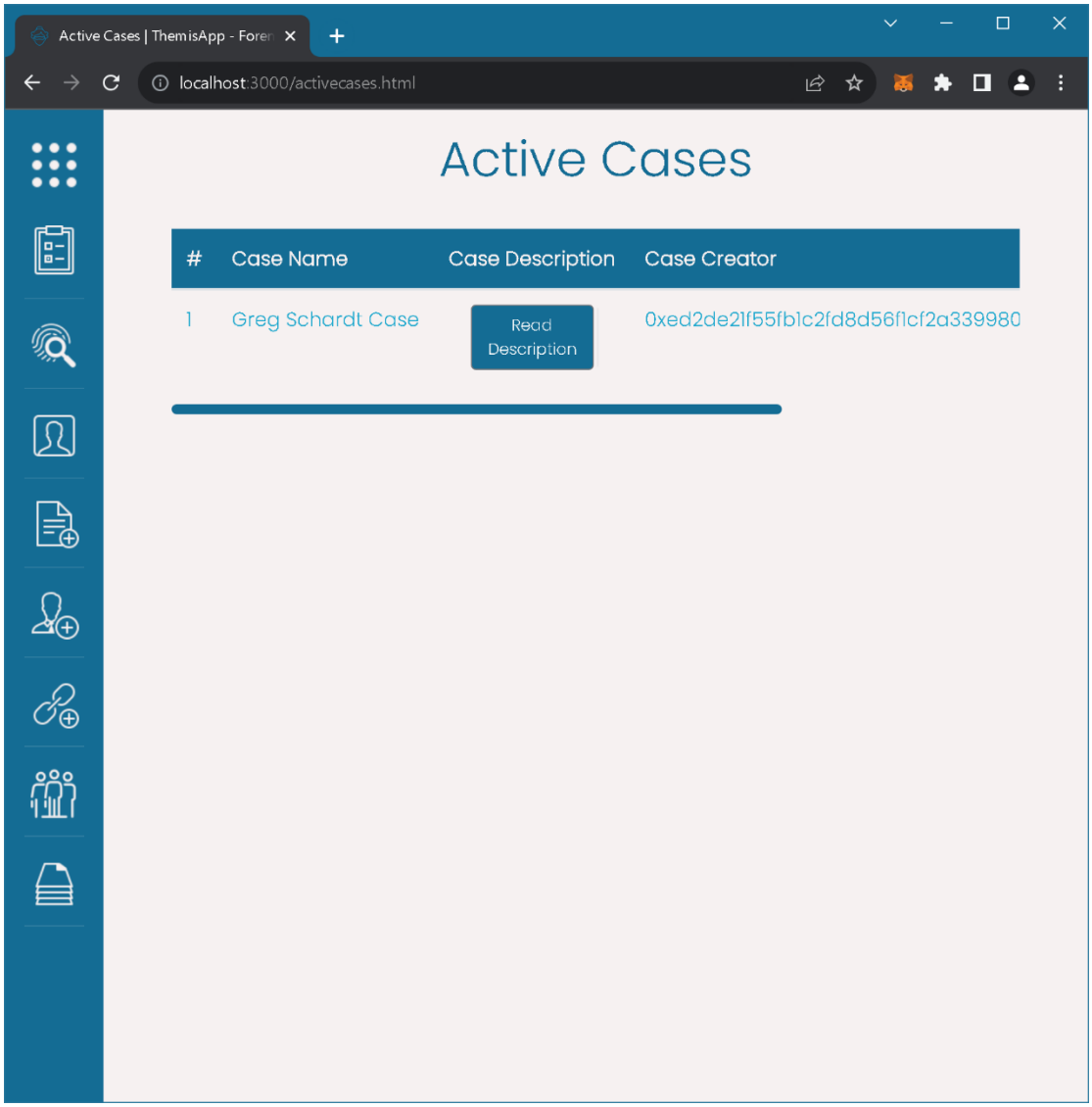

Figure 85 - The "Active Cases" page

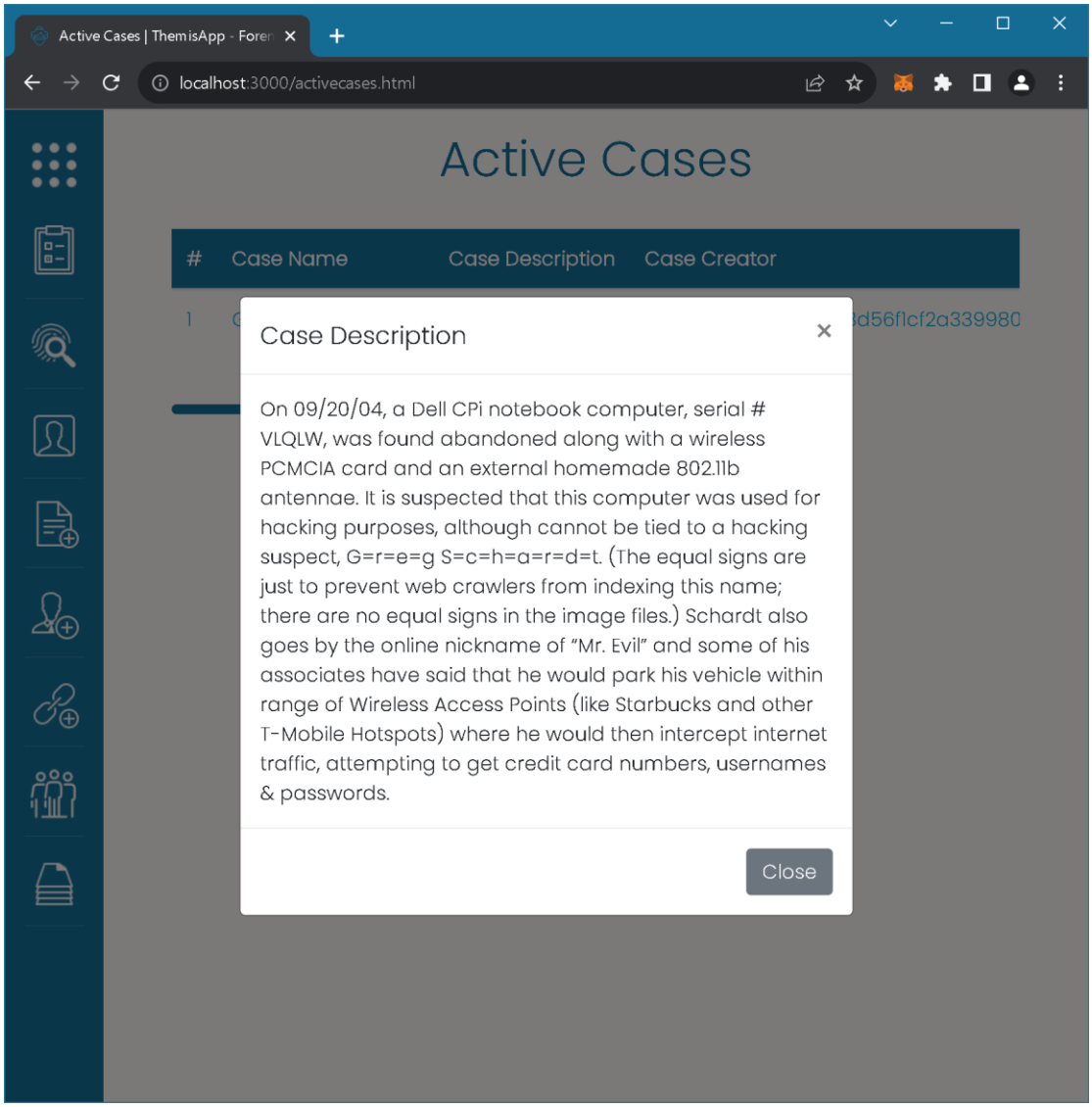

Figure 86 - View case description

- **Management of all cases registered on the blockchain**

Similarly to the management of all investigators, an administrator has the right to check all cases registered on the blockchain. This is accomplished through the "Manage Cases" page (Figures 87 and 88). The table contained on the page presents information about each case, but also provides a set of actions that can be applied to each of them. In Figure 89, the process of adding an additional investigator to the case using his wallet address is depicted, while Figure 90 shows the result of the process, with the case now assigned to two investigators.

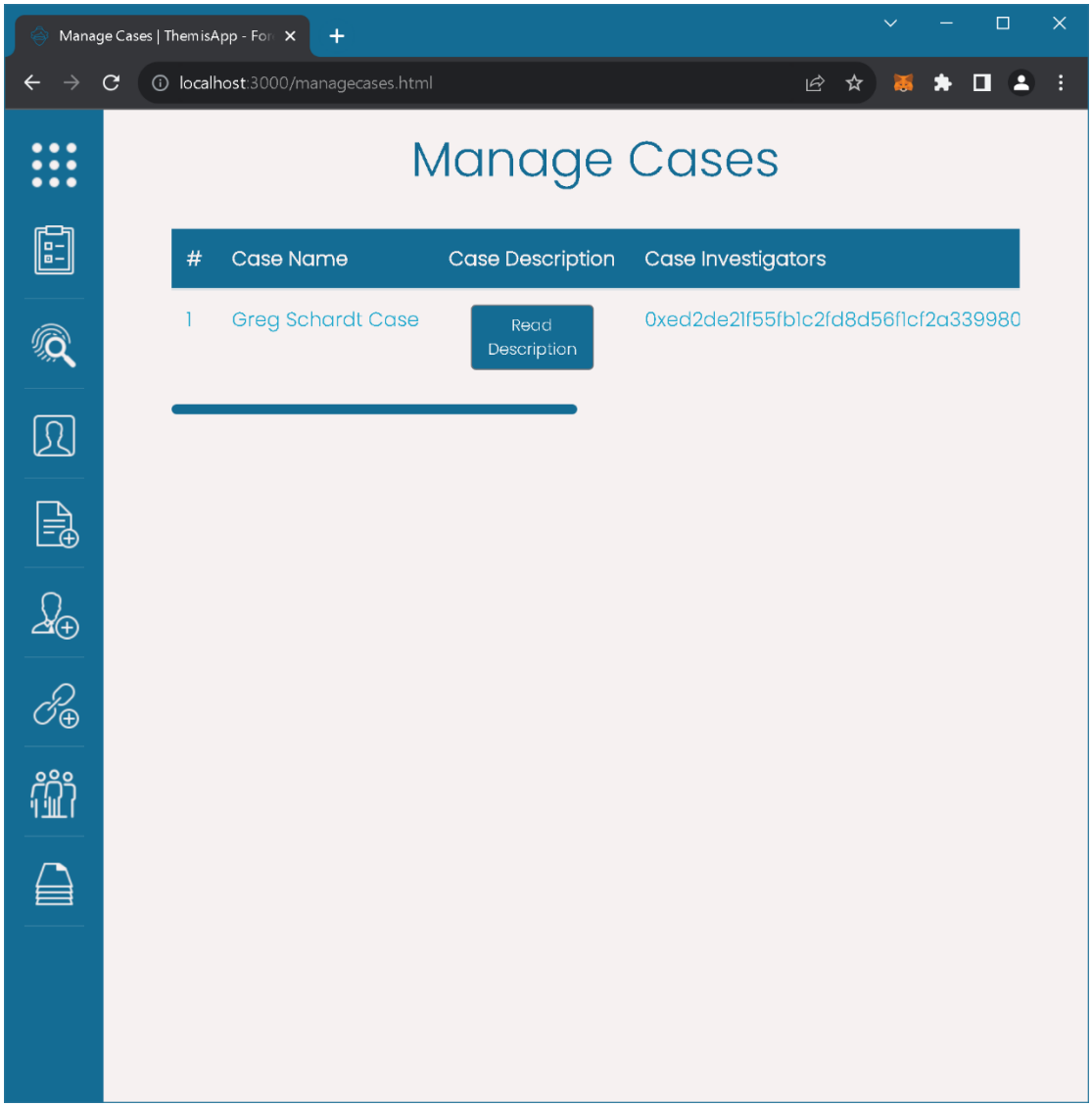

Figure 87 - The "Manage Cases" page (1/2)

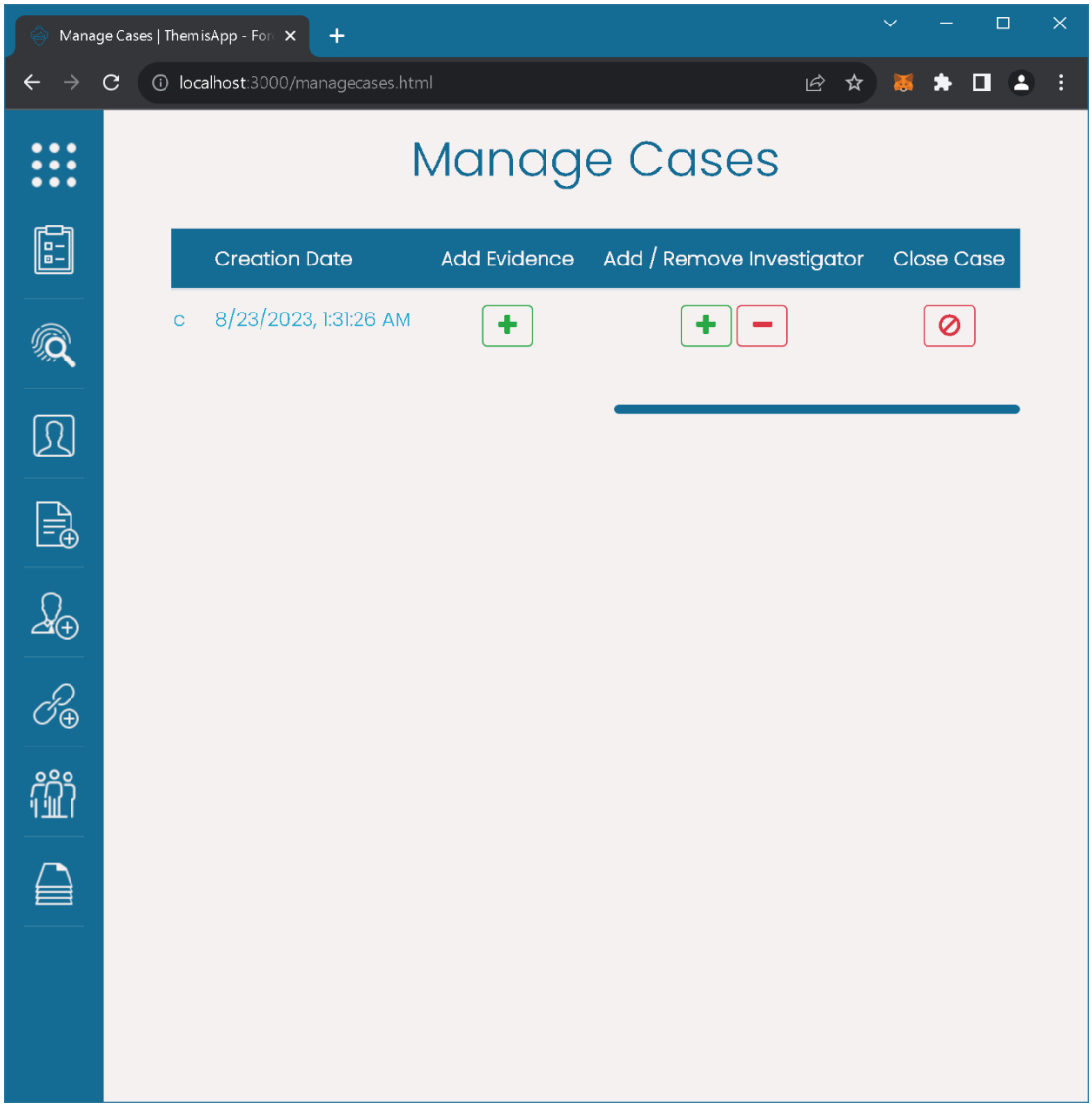

Figure 88 - The "Manage Cases" page (2/2)

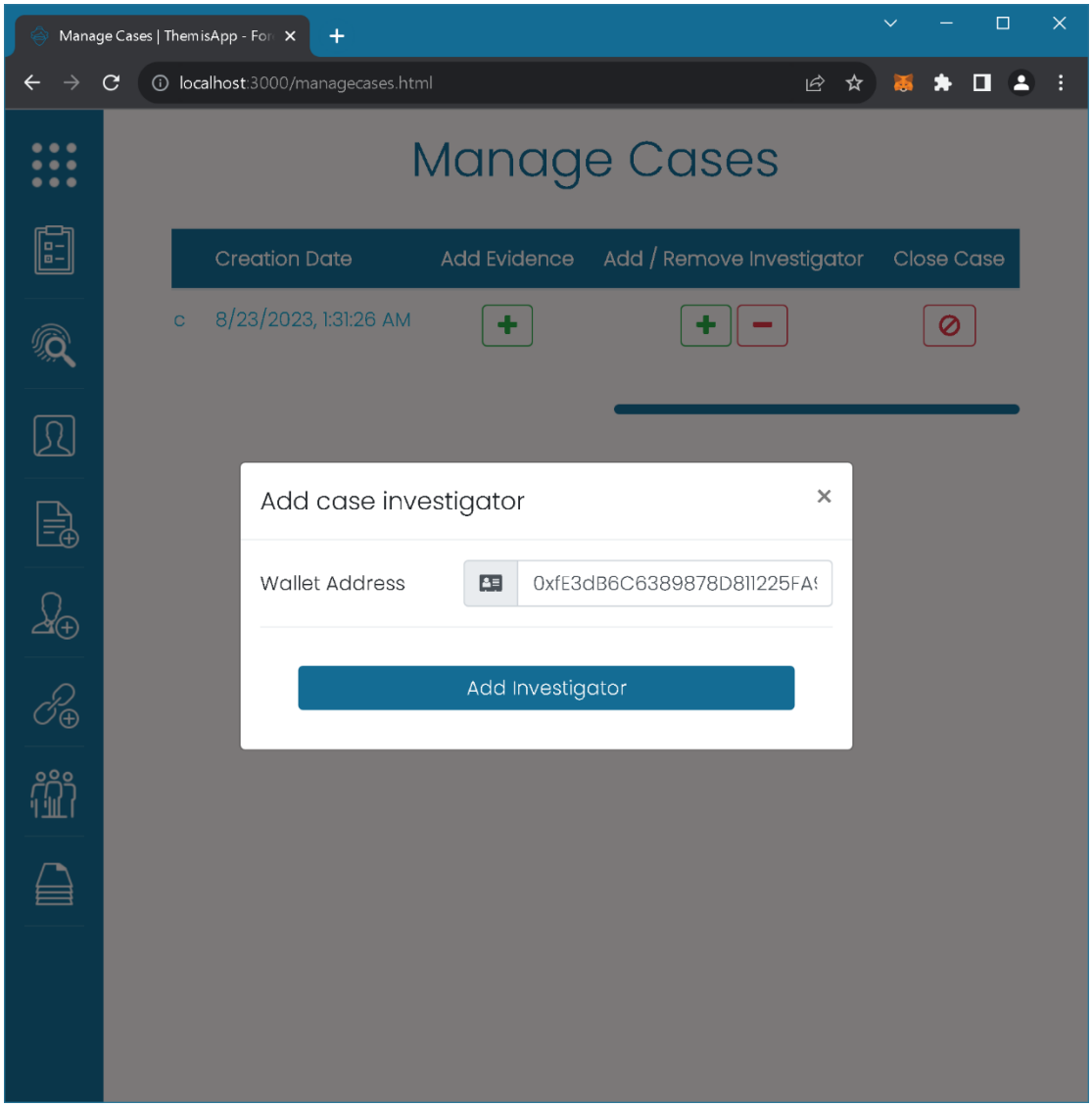

Figure 89 - Adding an additional investigator to the case

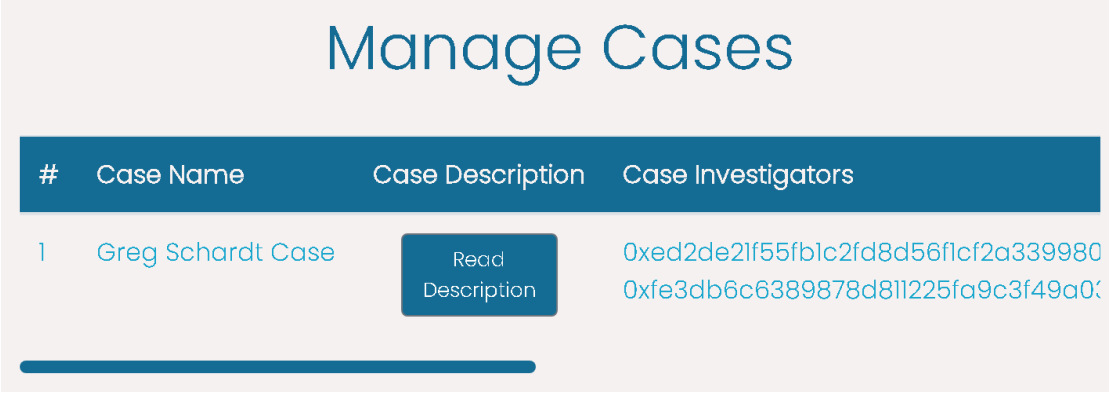

Figure 90 - The result of adding a new investigator to the case

- **Adding evidence to a case**

Apart from the case of opening a new case, the addition of evidence can also be carried out in secondary time by filling in the form of the "Add Case Evidence" page (Figure 91). To add evidence, the administrator is required to know the ID of the desired case, which can be searched on the "Manage Cases" page, and the hash of the evidence in "SHA-256" format, as well as to know what it is about so that he can assign a name and a description to it.

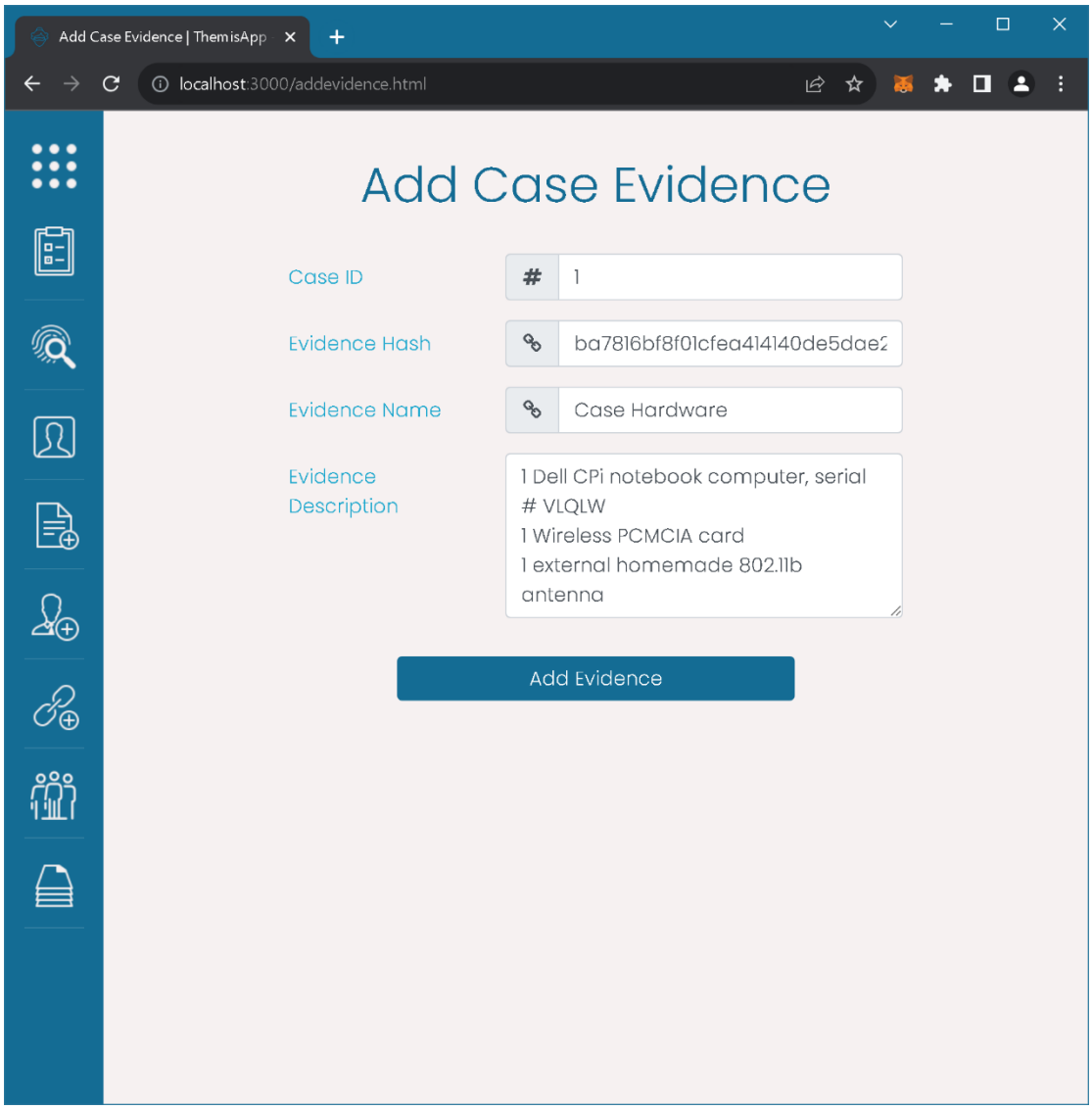

Figure 91 - The "Add Case Evidence" page

- **Searching for evidence in a case**

A common function of the application for investigators and administrators is to search for evidence. In order to perform any action, any of the two roles of the system should lie with the investigators of the case to which the evidence they are seeking belongs. Thus, by typing the ID of the case in the search, the evidence of the case is displayed as results (Figure 92), from which the investigator can then select the one he/she wishes to view the full information (Figure 93).

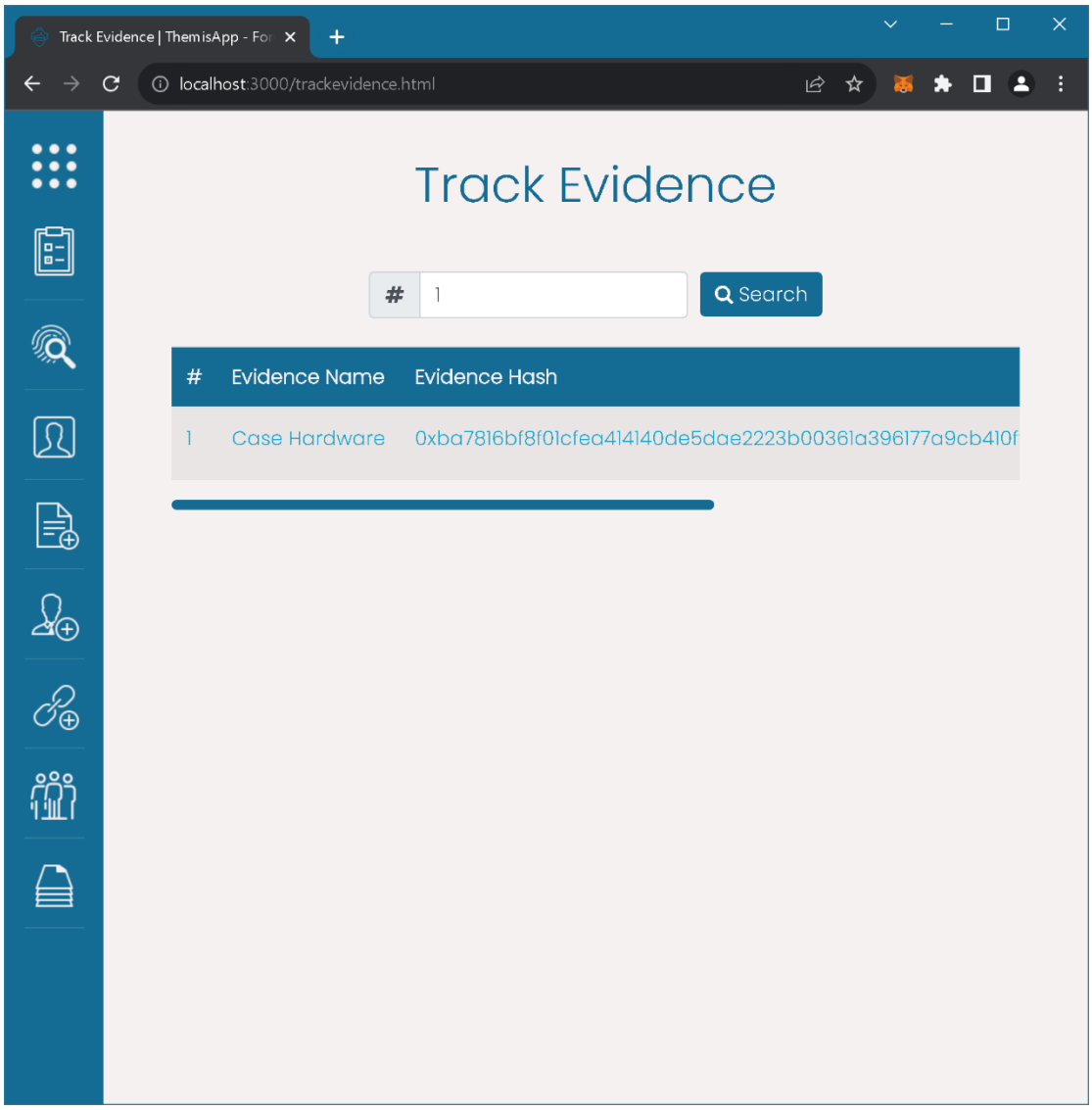

Figure 92 - The "Track Evidence" page

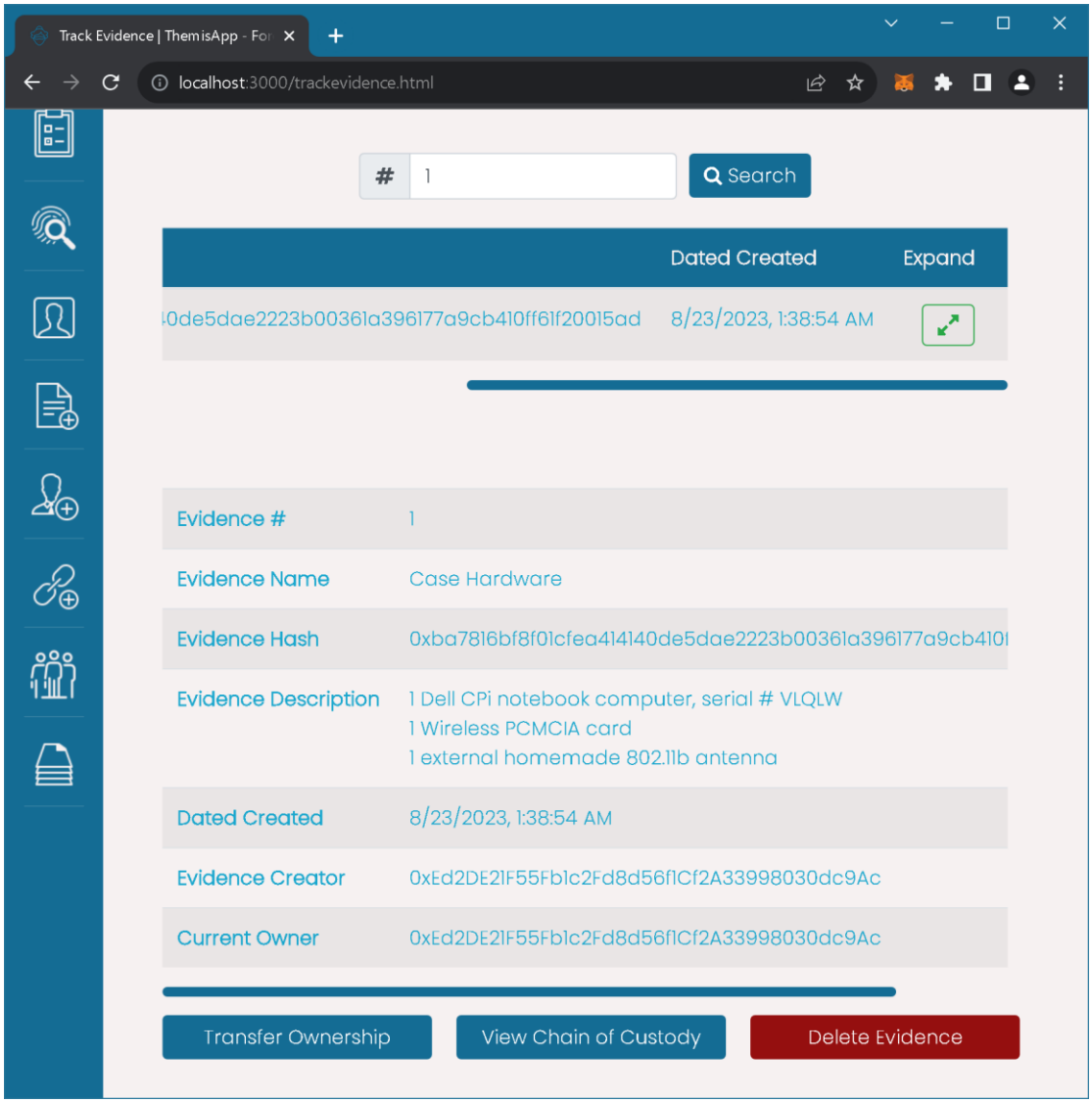

Figure 93 - The full information of the evidence

- **Transfer of ownership of evidence**

When viewing the full information of a piece of evidence, three buttons appear at the bottom of the page. As shown in the image above, these are: “Transfer Ownership”, “View Chain of Custody” and “Delete Evidence”. By clicking on the “Transfer Ownership” button, the administrator has the possibility, by entering the address of the recipient and the reason - description of the action, to transfer the ownership of the evidence to another investigator (Figure 94). The result of this action is shown in Figure 95, where the address of the “Evidence Creator” is now, different from that of the “Current Owner”.

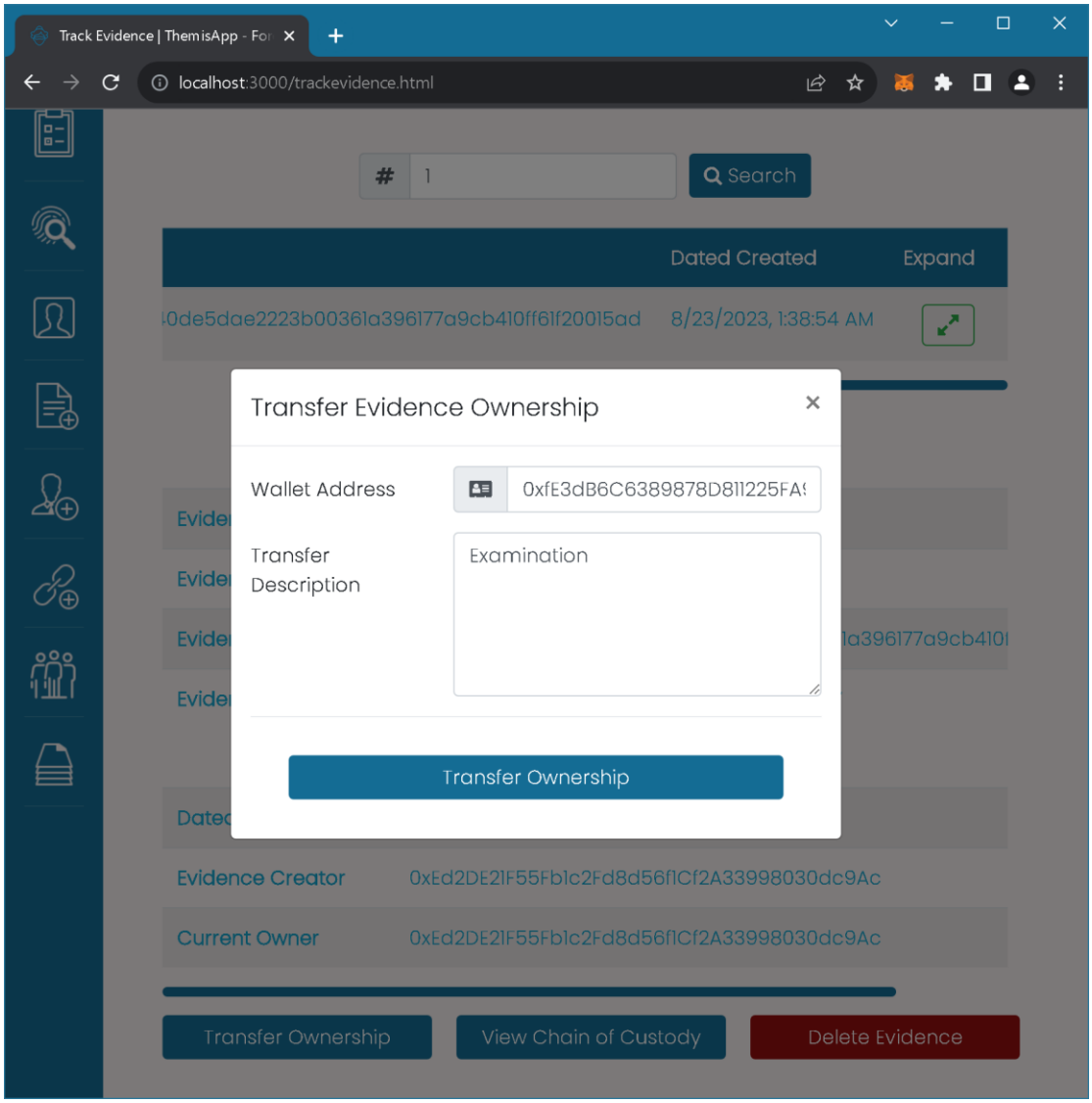

Figure 94 - Transfer evidence ownership form

Evidence Creator	0xEd2DE21F55Fb1c2Fd8d56f1Cf2A33998030dc9Ac
Current Owner	0xfE3dB6C6389878D811225FA9c3f49a03d38E6118

Figure 95 - The effect of the evidence ownership transfer

- **View the chain of custody of evidence**

After the transfer of ownership of the evidence is completed, its chain of custody is updated, including all the necessary data involved, such as the hash of the evidence, the day and time of the transaction and the recipient's wallet address (Figure 96).

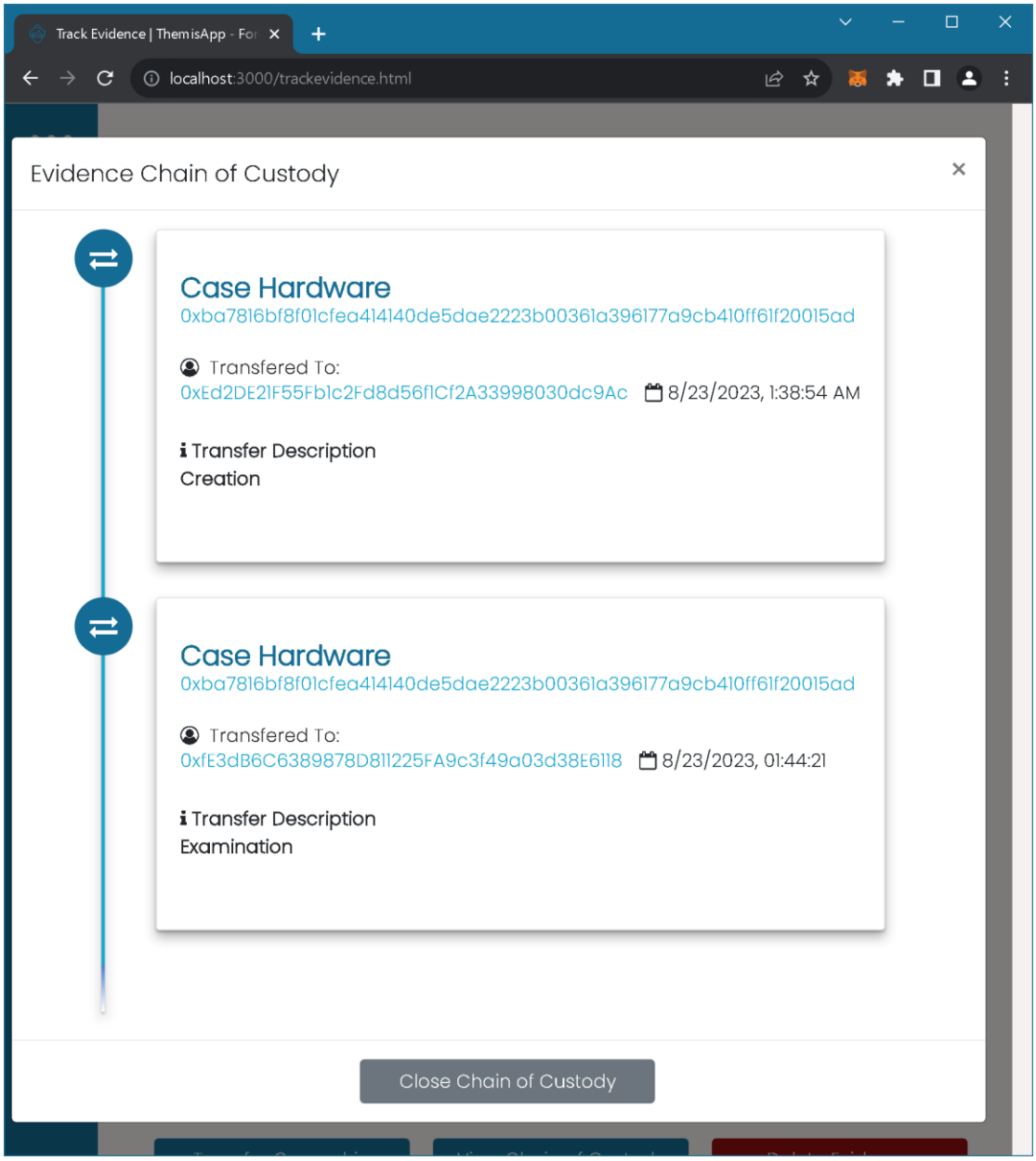

Figure 96 - The chain of custody of the evidence

7.4. Testing procedure

Conducting tests is part of the development process of an application, and it benefits in verifying the correctness and validating the functional characteristics of the

application (Pedamkar, 2019). The testing procedures that follow are nominally as follows: 1. Unit Testing, 2. Integration Testing, 3. Functional Testing, 4. Security Testing, 5. Compatibility Testing and 6. Regression Testing. Each of these presents a scenario, which consists of the scenario execution steps, expected and final results.

1. Unit Testing

Unit Testing is a testing methodology that targets discrete software units. Its purpose is to verify that each unit of software code performs as expected. It is performed in the development phase of the code to verify its accuracy. The units under the test may include functions, methods, procedures, modules or objects. (Hamilton, 2019a). For this testing process, the programming language "Javascript" and the "Chai" module of "Node.js" were used. The files containing the test code are located within the "test" folder in the application files.

Script: Testing the methods of smart contracts

Objective: Execute unit tests on individual methods of smart contracts within the private Ethereum blockchain "ThemisChain" and the decentralized chain of custody application "Themis".

Steps to execute the script:

1. **Preparation:**
   - Create a local testing environment for the private blockchain.
   - Deployment of smart contracts on the local blockchain.
2. **Case Smart Contract:**
   - Testing of methods relating to cases and evidence.
   - Provide sample case and evidence data and simulate adding them to the blockchain.
   - Confirmation that the methods return the expected response.
3. **Investigator Smart Contract:**
   - Testing methods involving investigators.
   - Provision of sample data on the identity of investigators.
   - Confirmation that the methods return the expected response.

4. **Issued Smart Contract:**
   - Testing access control features that ensure that only authorized participants can interact with smart contracts.
   - Verify that the smart contract denies access to unauthorized users.
5. **Embedded in the chain of custody application:**
   - Confirmation that the application interfaces correctly with smart contracts.
   - Confirmation that the user interface of the application reflects the results of the smart contracts functions.

Expected results:

- The functions of smart contracts are performed without errors.
- The evidence data is uploaded to the blockchain with the necessary details.
- Transfers of ownership of evidence automatically update chain of custody.
- The access control features properly restrict unauthorized access.
- The integration of smart contracts functionality into the chain of custody application is seamless and accurate.

Final results:

As can be seen in Figure 97, all the above expected results are fully covered.

```
Contract: Case
  ✔ Should successfully add a Case to caseList mapping (313ms)
  ✔ Should successfully show if a Case exists
  ✔ Should successfully get the name of a Case
  ✔ Should successfully show info about the active cases of an investigator
  ✔ Should successfully close a case (59ms)
  ✔ Should successfully assign a case to an investigator
  ✔ Should successfully remove an investigator from a case (49ms)
  ✔ Should successfully show the investigators of a case
  ✔ Should successfully show if an investigators exists in a case
  ✔ Should successfully add evidence to a case (313ms)
  ✔ Should successfully get collapsed info of an evidence
  ✔ Should successfully get expanded info of an evidence
  ✔ Should successfully get the hash of an evidence
  ✔ Should successfully count the evidences of a case
  ✔ Should successfully get the current owner of an evidence
  ✔ Should successfully transfer the evidence (50ms)
  ✔ Should successfully show the chain of custody of an evidence
  ✔ Should successfully delete an evidence (92ms)

Contract: Investigator
  ✔ Should successfully add a new investigator (83ms)
  ✔ Should successfully show investigator 's name
  ✔ Should successfully remove an investigator (74ms)
  ✔ Should successfully check if an investigator with the same wallet address exists
  ✔ Should successfully check if an investigator with the same id exists

Contract: Issued
  ✔ Should successfully check if an admin exists

Contract: Issued
  ✔ Contract "Issued" successfully deployed

Contract: Investigator
  ✔ Contract "Investigator" successfully deployed

Contract: Case
  ✔ Contract "Case" successfully deployed

Contract: Evidence
  ✔ Contract "Evidence" successfully deployed


28 passing (2s)
```

Figure 97 - The list of successful "Unit Tests" of the application

2. Integration Testing

Integration Testing groups and tests the various separate modules of software (Hamilton, 2019b).

Script: Interaction between the private blockchain "ThemisChain" and the decentralized application chain of custody "Themis".

Objective: Ensure the seamless integration and proper functioning of the chain of custody application with the private blockchain. This test validates that the application components interact properly with smart contracts and data storage within the blockchain.

Steps to execute the script:

1. **Preparation:**
   - Confirmation that the private blockchain network and the chain of custody application are operational.
2. **Interaction of smart contracts:**
   - Access the application interface and create a new evidence entry.
   - Control of the interaction between the frontend of the application and the corresponding smart contract.
   - Verify that the data entered in the application is accurately recorded and stored in the smart contract's storage.
3. **Transfer evidence ownership:**
   - Selecting previously created evidence and initiating a process of transferring ownership to another investigator.
   - Monitoring the communication between the application and the smart contract during the transfer process.
   - Confirmation that the smart contract updates the ownership status of the evidence and records the transfer event.
4. **Consistency of data:**
   - Create a new evidence registration directly through the smart contract on the blockchain.
   - Access to the application interface and automatic synchronization of its data with the blockchain.
   - Confirmation that the application is receiving recently added data and displaying it accurately.

Expected results:

- **Smart contract interaction:** the application should successfully communicate with the smart contract and update the evidence data on the blockchain.

- **Transfer evidence ownership:** The application and the corresponding smart contract should interact to facilitate the transfer of ownership of evidence and update the ownership status.
- **Data consistency:** the direct addition of evidence to the blockchain should be reflected in the application data after synchronization.

Final results:

The final results of the above test scenario validate the seamless interaction between the "Themis" application and the private blockchain "ThemisChain" and ensure that the elements of the application, such as smart contracts and data retention, work in harmony, providing a functional and reliable solution for evidence management. Furthermore, the successful completion of this trial contributes to the research objective to demonstrate the effective synergy between private blockchain and chain of custody applications.

3. Functional Testing

Functional Testing is performed to check whether the application meets the functional requirements stated during the analysis of its structure (see page 85).

Script: Functional test of the decentralized chain of custody application "Themis".

Objective: Validate the basic functions of the chain of custody application. This test ensures that the features and functions of the application work as intended, satisfying the requirements described in its structural analysis.

Steps to execute the script:

1. **Preparation:**
   - Confirmation that the private blockchain network and the chain of custody application are operational.
2. **User authentication and authorization :**
   - Test user access control by performing actions restricted for execution by specific system roles.

- Confirmation that only authorized users can perform actions such as transfer of ownership of evidence.
- Confirm that unauthorized users receive appropriate error messages when attempting restricted actions.

3. **Generate and record evidence :**
   - Log in as an administrator and go to the "Add Case Evidence" page.
   - Create a new evidence entry, providing accurate information.
   - Verification that the application records the evidence.
4. **Transfer evidence ownership - Monitoring the chain of custody:**
   - Initiate the process of transferring ownership of a piece of evidence from one investigator to another.
   - Confirm that the application is updating the property details and creating a transfer tab.
   - Confirmation that the transfer process includes the appropriate sender, recipient and evidence hash information.
5. **Immutability:**
   - Confirmation that the data registered on the blockchain cannot be tampered with.
   - Control the access and processing rights of users who have a "investigator" role.
   - Confirmation that the characteristics of the evidence remain accurate and identical to the original physical copy.

Expected results:

- **User authentication and authorization:** Successful execution of key actions by authorized users and corresponding rejection for unauthorized users.
- **Generating and recording evidence:** Accurately recording the details of the evidence.
- **Transfer evidence ownership - Monitoring the chain of custody:** Correctly updating ownership details and maintaining a history of transfer.
- **Immutability:** Safeguarding the immutability of data recorded on the blockchain.

Final results:

This functional test scenario verifies that the key functions of the chain of custody application are functioning as intended, aligned with the described requirements. Successful completion of this test demonstrates that the application effectively facilitates the management of evidence and ensures the integrity of the chain of custody of evidence.

4. Security Testing

Security Testing is the testing of an application for its protection against external and internal threats (Acharya, 2022).

Script: Security test of the decentralized chain of custody application "Themis".

Objective: Evaluate the security aspects of the chain of custody application, ensuring that sensitive data remains protected, and the application is resilient against common security vulnerabilities.

As mentioned in subchapter 6.1 (see page 65), the "ThemisChain" blockchain and by extension the "Themis" application were created and intended for academic purposes and not for real-world use cases. Therefore, no emphasis was placed on trying to maintain data security rules and protection against cyber-attacks. However, a theoretical approach to a Security Testing scenario would include the following steps:

Steps to execute the script:

1. **Preparation:**
   - Confirmation that the private blockchain network and the chain of custody application are operational.
2. **Attempt to bypass authentication:**
   - Attempt to bypass the authentication mechanism by invading restricted areas without valid credentials.
3. **XSS (Cross-Site Scripting) attacks:**
   - Attempt to insert a malicious script through the fields of a form.
4. **Security of the blockchain:**

- Evaluating the security of the private blockchain by attempting unauthorized transactions or hacking into the blockchain records.
- Confirmation that the blockchain network maintains the integrity of its data and prevents unauthorized changes.

Expected results:

- **Attempt to bypass authentication:** Unauthorized access should be blocked, and appropriate authentication should be enforced.
- **XSS (Cross-Site Scripting) attacks:** The application should filter and escape user input to prevent XSS attacks.
- **Blockchain security:** Unauthorized transactions or hacking attempts should be prevented by the blockchain.

5. Compatibility Testing

The type of testing that evaluates how an application runs and behaves across different platforms, servers, network environments and hardware configurations is called "Compatibility Testing". Its purpose is to ensure that an application runs smoothly with optimal performance across different browsers, configurations, databases and software versions (Acharya, 2022).

Script: Compatibility check of the decentralized chain of custody application "Themis".

Objective: Ensure that the chain of custody app is compatible with different devices and browsers, providing a consistent user experience across different platforms.

Steps to execute the script:

1. **Preparation:**
   - Confirmation that the chain of custody application is operational.
   - Set up a test environment with various devices and browsers.
2. **Device compatibility:**
   - Testing the application on different devices, such as desktops, laptops, tablets and smartphones.

3. **Web browser compatibility :**
    - Access to the application through different web browsers, including “Google Chrome”, “Mozilla Firefox”, “Microsoft Edge” and “Safari”.

Expected results:

- **Device compatibility:** The app should adapt responsively to different devices and screen sizes.
- **Web browser compatibility:** The application should work consistently and display correctly across different web browsers.

Final results:

The success of the compatibility testing scenario ensures that the chain of custody application delivers a consistent and reliable user experience across a range of devices (Figures 98 and 99) and web browsers (Figures 100 and 101). This compatibility ensures wider accessibility and usability for those involved in digital forensic processes.

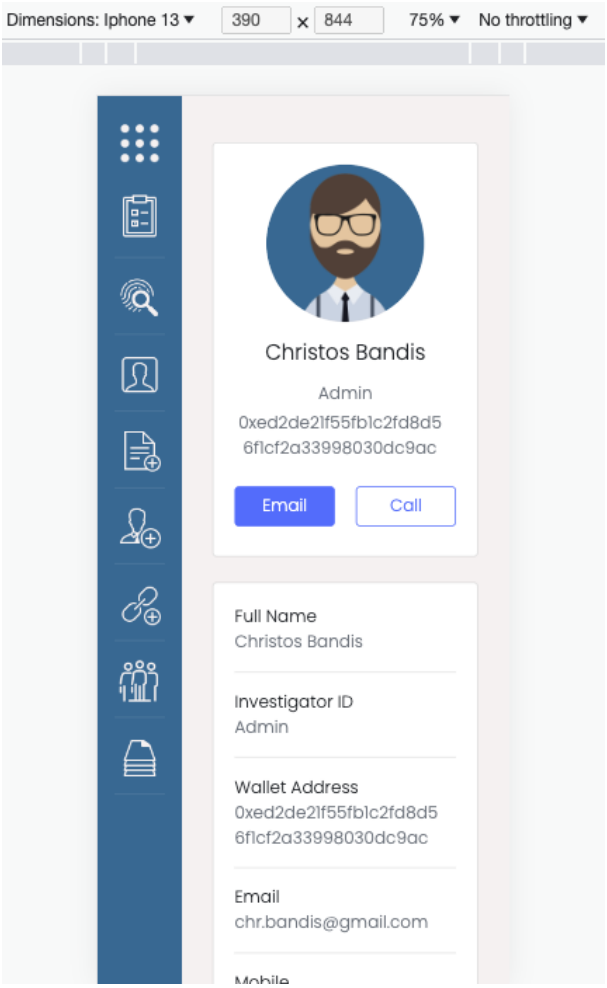

Figure 98 - The appearance of the investigator's profile page on an "iPhone 13" (smartphone) (Developer Tools - Google Chrome)

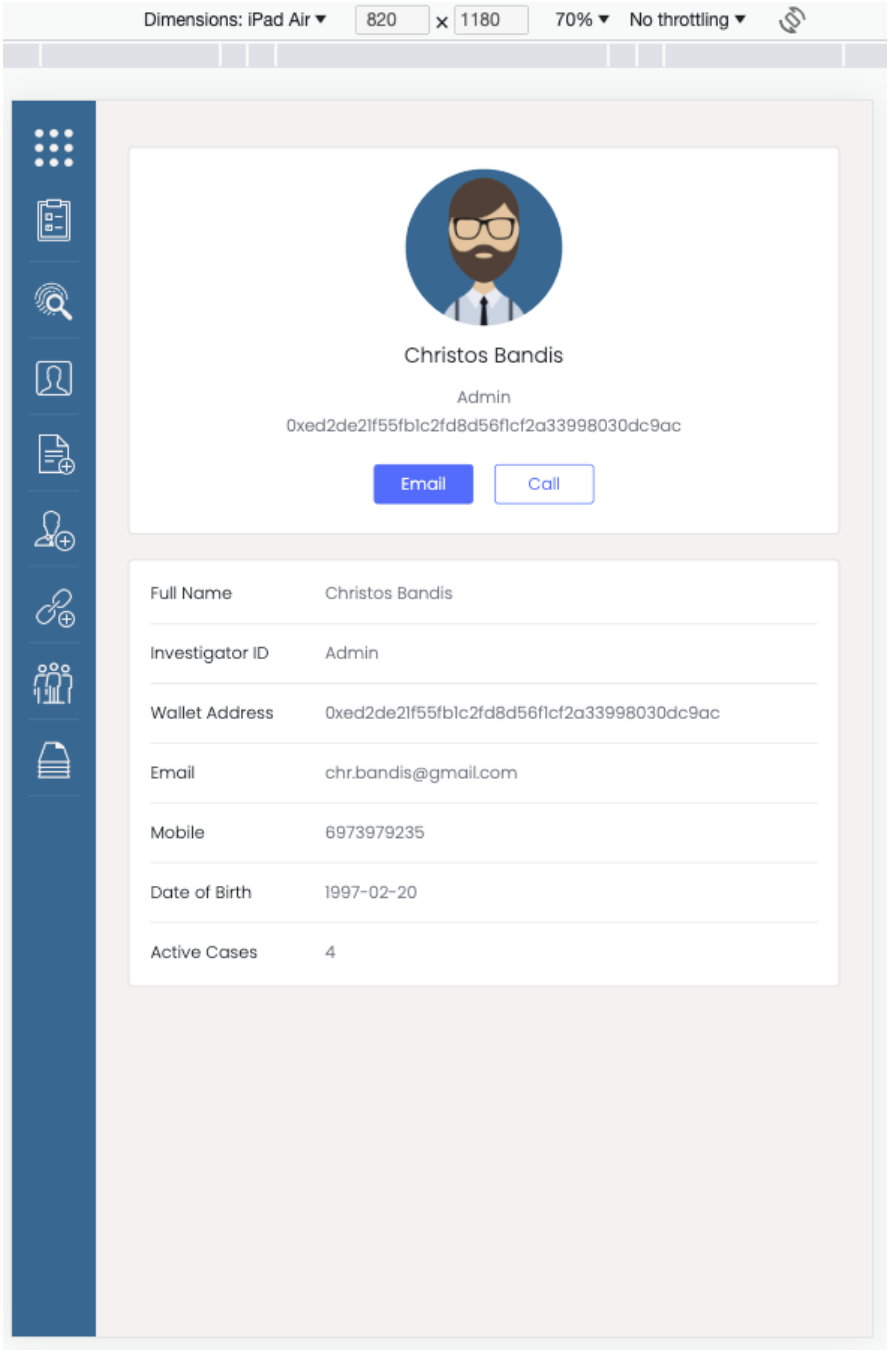

Figure 99 - The appearance of the investigator's profile page on an "iPad Air" (tablet) (Developer Tools - Google Chrome)

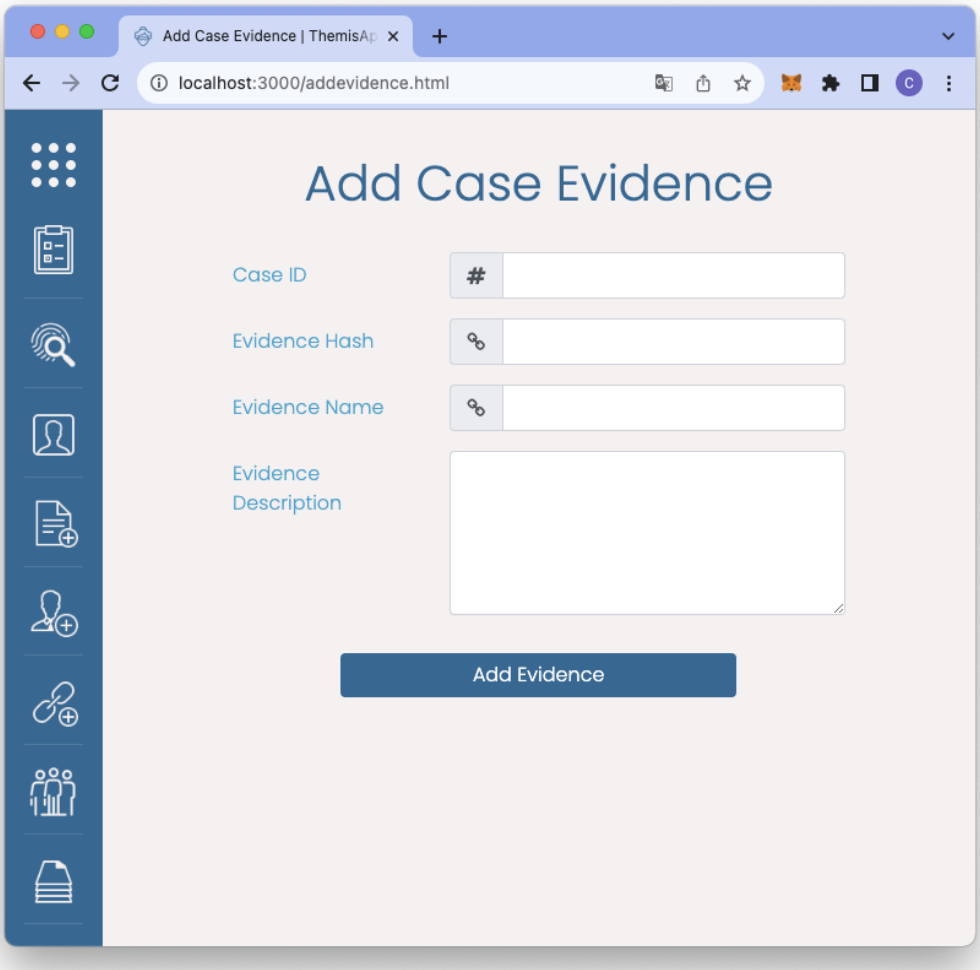

Figure 100 - The appearance of the "Add Case Evidence" page in the "Google Chrome" web browser

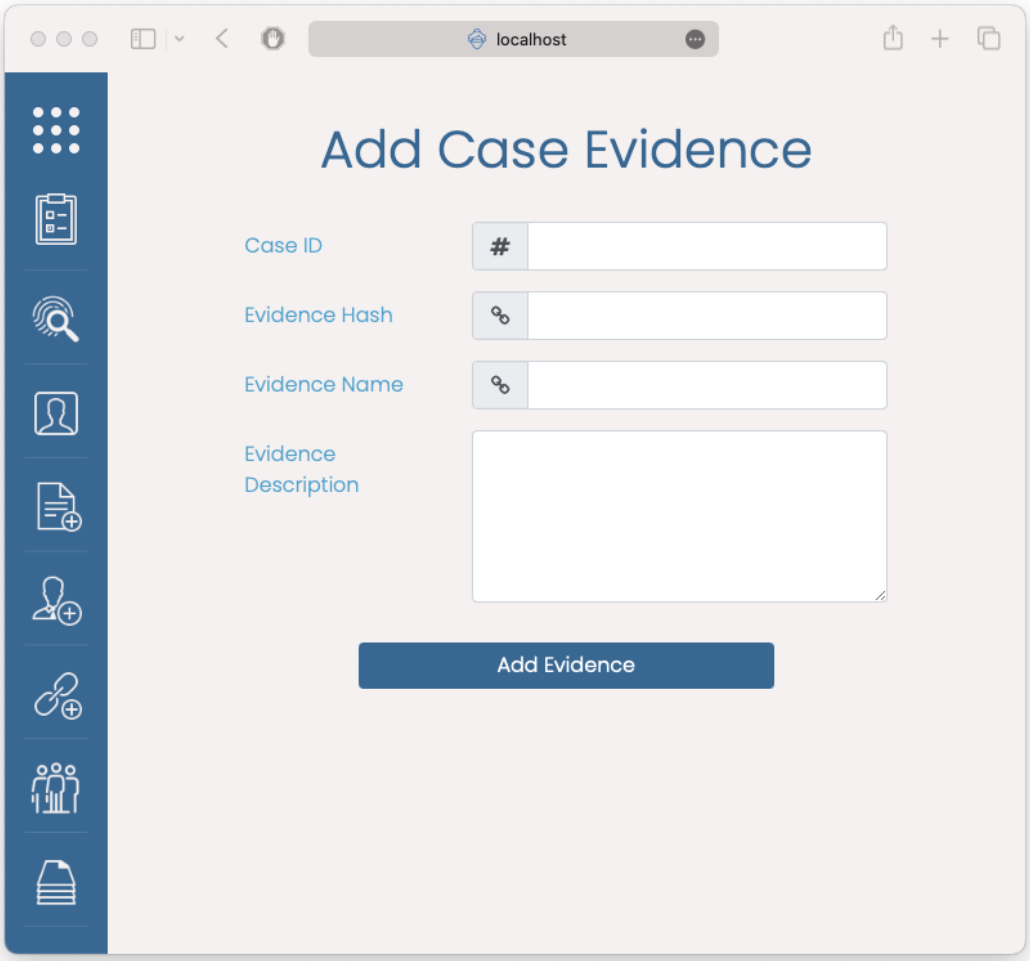

Figure 101 - The appearance of the "Add Case Evidence" page in the "Safari" web browser

6. Regression Testing

Regression Testing is defined as the testing of how an integrated application works after modifying any functionality, component or module. Its goal is to ensure that the existing functions of the application remain unaffected after new modifications.

Script: Regression test of the decentralized application chain of custody "Themis".

Objective: Ensure that the latest modifications and enhancements made to the chain of custody application do not negatively affect existing functionality or add new defects.

Steps to execute the script:

1. **Preparation:**

- Confirmation that the chain of custody application is operational.
- Keep track of recent changes, improvements or bug fixes that have been implemented.

2. **Test reference point:**
   - Establish a benchmark by conducting a full test of the existing application functionality.
3. **Implementation of amendments:**
   - Implementation of recent changes, improvements or bug fixes.
4. **Regression test:**
   - Test all the features, workflows and scenarios previously tested and covered in previous test phases.
   - Confirmation that the recent changes have not caused any unwanted side effects or regressions.
5. **Data integrity test:**
   - Verification that recent changes have not affected the integrity of the data, in the case of data storage, retrieval and handling.
6. **User interface test:**
   - Confirm that the user interface and design elements remain consistent and functional.
7. **Compatibility testing:**
   - Verify that recent changes have not affected compatibility with different devices and browsers.
8. **Performance test:**
   - Measuring the performance of the application before and after the changes to find any performance degradation.
9. **Safety test:**
   - Confirmation that the changes have not introduced any new vulnerabilities or security weaknesses.

Expected results:

- All previously tested features and workflows should continue to work as expected.
- The integrity of the data should be maintained without discrepancies.
- User interface elements must remain intact.
- Compatibility with devices and browsers should not be compromised.

- Performance metrics should remain stable or improve.
- No new security vulnerabilities should be introduced.

Final results:

Regression Testing ensures that recent modifications and improvements made to the chain of custody application have not adversely affected its existing functionality. By confirming that recent changes do not introduce regressions or defects, the overall quality and reliability of the application is maintained, providing a robust solution for digital forensics processes.

7. Other observations

During the writing of the application code, two bugs were encountered, which do not affect the correct operation of the application and thus, their fixes will be approached theoretically. These bugs are:

1. In the process of transferring ownership of evidence to an investigator who is not registered in the case to which the evidence belongs, although the process is completed, the number of active cases of the investigator remains the same (does not increase). Of course, in this way the security and integrity of the evidence is compromised, as this transfer may be the result of a malicious act. Therefore, the most appropriate solution to the problem would be to control the investigators to whom the case has been assigned so that if they do not own the property, the procedure is immediately cancelled.
2. After deleting an investigator, who has one or more active cases, from the blockchain, his/her address remains in the list of investigators assigned to the case. This error can be handled in two ways: either not allowing deletion in the first place and displaying the appropriate message to the user, or automatically deleting the investigator from the case as well.

Chapter 8: Conclusions

The research conducted in the context of this thesis highlighted the potential synergies of private blockchain and chain of custody applications in the field of digital forensics. Modern developments in the technological field have highlighted the criticality of efficient collection, secure storage, meticulous preservation and methodical analysis of digital forensic evidence, a key tool for fighting cybercrime and presenting well documented evidence in legal proceedings. In the field of cyber investigations, the management of digital evidence and its chain of custody is becoming increasingly important, serving as a fundamental concern that exerts a profound impact on the effectiveness of cybercrime investigations, as they link discrete events and activities that facilitate the conduct of integrated criminal investigations.

The inherent characteristics of blockchain technology ensure properties such as integrity, transparency, authenticity, security and data auditability, potentially making it the best choice for the creation and meticulous monitoring of the chain of custody in the field of digital forensic processes. The maintenance of the chain of custody is highlighted as crucial, as it has the potential to document the accuracy of evidence by indicating the possibility of tampering during the stages of collection and subsequent analysis.

Through the development of the private Ethereum Blockchain “ThemisChain” and the decentralized chain of custody application “Themis”, a comprehensive exploration of their interconnected dynamics was achieved. The convergence of these technologies has charted a path towards increased integrity and enhanced data transparency and security in the digital forensics landscape.

The exploration of the structure, design and implementation of the private Ethereum Blockchain and chain of custody application revealed a robust framework that not only optimizes evidence management but also addresses the challenges associated with ensuring traceability and non-repudiation of evidence. The use of blockchain technology features has forged a reliable and tamper-resistant evidence repository, promoting a secure chain of custody, throughout the research lifecycle.

The findings of this study mark an important step in the effort to transform the realm of digital forensic processes. The collaboration between private blockchain and chain of custody applications offers a transformative change in the way evidence is organized, preserved and presented.

In conclusion, this paper highlights the importance of collaborative innovation between blockchain technology and research methodologies, setting the stage for an upcoming era of increased security, transparency and accountability in digital forensics. The results presented highlight the imperative for continued research and refinement of similar solutions in the evolving landscape of digital evidence management. As technology evolves, the potential for enhancing the integrity and reliability of digital forensics through new applications of blockchain technology remains limitless.

8.1. Future actions

In the wake of this research, several potential avenues for subsequent exploration and development emerge, presenting prospects for meticulously enhancing and expanding the harmonious fusion of private blockchain and chain of custody applications in the field of digital forensics. Initially, future development goals include the integration of the developed model (ThemisChain & ThemisApp) into a potential court system, which will be accessed by the executives of the respective court (e.g. lawyers and judges) to check and verify the integrity of evidence themselves. Next, at the code level, future development plans for the application include two objectives. The first of these is the process of enhancing the hash of the evidence by adding random characters (salting) to each of them, increasing their complexity. The second, as mentioned in subchapter 4.1 (see page 31), is to periodically record the state of the private blockchain on a public one, ensuring the integrity of the ledger itself. Another future goal is to transform the private blockchain, which as noted in subchapter 6.1 (see page 65) based on its connection rules is intended for academic use, into a network capable of handling real forensic data. Furthermore, validation of the application to a range of real forensic scenarios could provide a solid basis for extensive case studies and empirical evaluations. Furthermore, deepening the integration of advanced cryptographic techniques, such as zero-knowledge proofs or homomorphic encryption, has the potential to increase the privacy and confidentiality dimensions of the application, ensuring the privacy of sensitive information during the research process. These upcoming trajectories collectively possess the power to enhance research contributions and steer the field of digital forensics towards increased efficiency, reliability and technological advancement.

Chapter 9: Appendix

9.1. Project scheduling

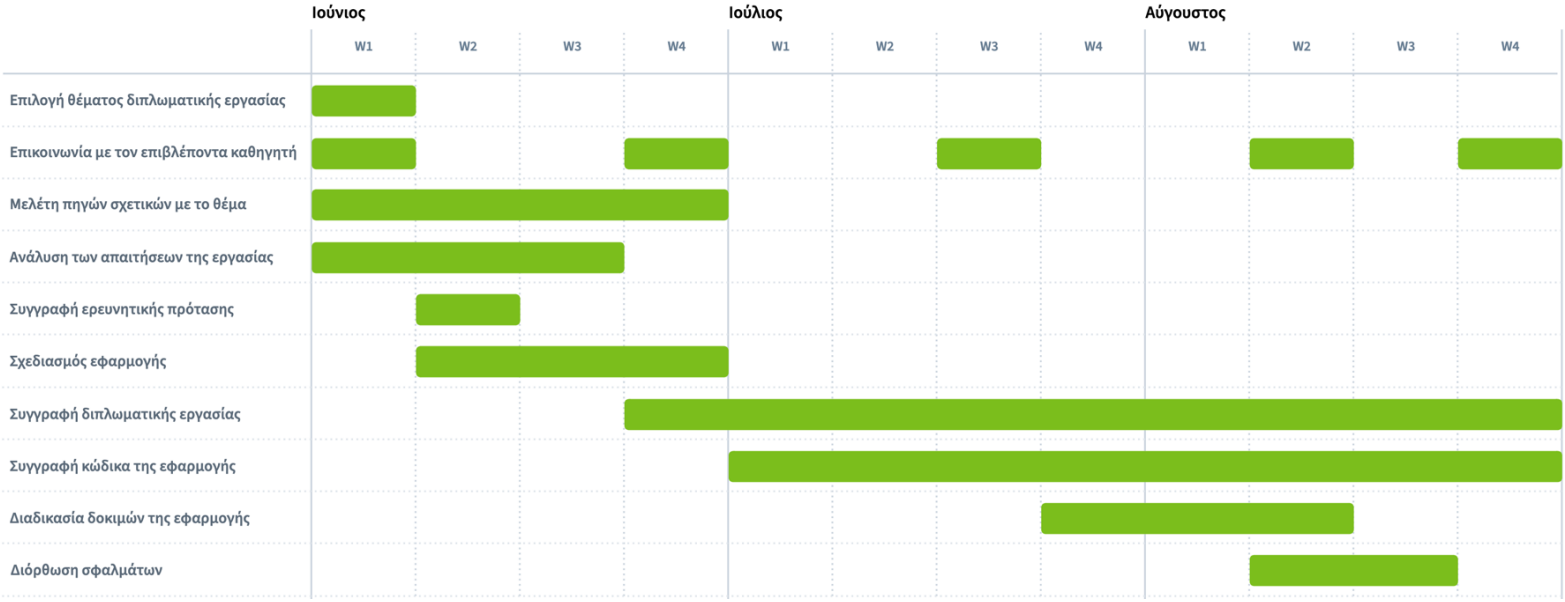

Figure 102 - Gantt chart

The Gantt chart above shows the expected time span of the project, from topic selection to delivery. The time is measured in weeks, starting from the first week of June, when the selection of the project topic takes place. The timing of the distribution of activities is intended to make the project more efficient. As an indication, communication with the supervisor was set to take place every three weeks to ensure sufficient data for presentation and feedback. It is also worth mentioning that the study of the sources took the equivalent of four (4) weeks, as it was preceded by research and content evaluation. The analysis of the task requirements, which started at the same time as the study of the sources, took three (3) weeks due to the complexity of the topic. The time spent on the design of the application did not exceed three (3) weeks, as the simplicity and at the same time the usability of the application were the main criteria for its implementation. Finally, the remaining time was allocated to writing the thesis and the application code. In the latter, the tests that the application underwent and the fixing of the bugs that were presented were included, in which considerable amount of time was spent to address possible issues and make the application complete.

9.2. References

Acharya, D.P. (2022). Understanding Different Types of Application Testing. [online] Geekflare. Available at: https://geekflare.com/understanding-application-testing/ [Accessed 2 Aug. 2023].

Ahmad, L., Khanji, S., Iqbal, F. and Kamoun, F. (2020). Blockchain-based chain of custody. Proceedings of the 15th International Conference on Availability, Reliability and Security. doi:https://doi.org/10.1145/3407023.3409199

Anand, A. (2020). Breaking Down : SHA-256 Algorithm. [online] Medium. Available at: https://infosecwriteups.com/breaking-down-sha-256-algorithm-2ce61d86f7a3 [Accessed 23 Jul. 2023].

Antolin, M. (2022). What Is Proof-of-Authority? [online] CoinDesk. Available at: https://www.coindesk.com/learn/what-is-proof-of-authority/ [Accessed 24 Jul. 2023].

Antonopoulos, A. and Wood, G. (2018). Mastering Ethereum: Building Smart Contracts and DApps. O'Reilly Media.

Buterin, V. (2013). Ethereum White Paper: A Next Generation Smart Contract & Decentralized Application Platform. [online] Available at: https://github.com/ethereum/wiki/wiki/White-Paper [Accessed 25 Jul. 2023].

CardBoard. (2020). How to Prioritize Agile Stories. [online] Available at: https://cardboardit.com/2020/08/how-to-prioritize-agile-stories/ [Accessed 26 Jul. 2023].

Castor, A. (2017). A (Short) Guide to Blockchain Consensus Protocols. [online] CoinDesk. Available at: https://www.coindesk.com/markets/2017/03/04/a-short-guide-to-blockchain-consensus-protocols/ [Accessed 24 Jul. 2023].

Chen, H. and Liang, D. (2022). Adaptive Spatio-Temporal Query Strategies in Blockchain. ISPRS International Journal of Geo-Information, 11(7), p.409. doi:https://doi.org/10.3390/ijgi11070409.

Christidis, K. and Devetsikiotis, M. (2016). Blockchains and Smart Contracts for the Internet of Things. IEEE Access, [online] 4(4), pp.2292–2303. doi:https://doi.org/10.1109/access.2016.2566339.

coin98.net. (2022). What Is EVM (Ethereum Virtual Machine)? How Does EVM Work? [online] Available at: https://coin98.net/what-is-evm [Accessed 25 Jul. 2023].

Ćosić, J. and Bača, M. (2010). (Im) Proving Chain of Custody and Digital Evidence Integrity with Time Stamp. The 33rd International Convention MIPRO, pp.1226–1230.

Ethereum.org (2023). Introduction to dapps. [online] Ethereum.org. Available at: https://ethereum.org/en/developers/docs/dapps/ [Accessed 25 Jul. 2023].

Fernandez-Carames, T.M. and Fraga-Lamas, P. (2020). Towards Post-Quantum Blockchain: A Review on Blockchain Cryptography Resistant to Quantum Computing Attacks. IEEE Access, 8, pp.21091–21116. doi:https://doi.org/10.1109/access.2020.2968985.

Giova, G. (2011). Improving Chain of Custody in Forensic Investigation of Electronic Digital Systems. International Journal of Computer Science and Network Security, Vol. 11, pp.1–9.

Goldmann, M. (2019). App Vs. dApp. [online] Cryptopolitan. Available at: https://www.cryptopolitan.com/app-vs-dapp/ [Accessed 25 Jul. 2023].

Gondek, C. (n.d.). What is a Consortium Blockchain? [online] originstamp.com. Available at: https://originstamp.com/blog/what-is-a-consortium-blockchain/#what-is-consortium-blockchain [Accessed 25 Jul. 2023].

Guo, J., Wei, X., Zhang, Y., Ma, J., Gao, H., Wang, L. and Liu, Z. (2022). Antitampering Scheme of Evidence Transfer Information in Judicial System Based on Blockchain. 2022, pp.1–19. doi:https://doi.org/10.1155/2022/5804109.

Gupta, R. (2018). Hands-on cybersecurity with blockchain : implement DDoS protection, PKI-based identity, 2FA, and DNS security using blockchain. Birmingham ; Mumbai: Packt.

Hamilton, T. (2019a). Unit Testing Tutorial: What is, Types, Tools, EXAMPLE. [online] Guru99.com. Available at: https://www.guru99.com/unit-testing-guide.html [Accessed 2 Aug. 2023].

Hamilton, T. (2019b). Integration Testing: What is, Types, Top Down & Bottom Up Example. [online] Guru99.com. Available at: https://www.guru99.com/integration-testing.html [Accessed 2 Aug. 2023].

Jeong, J., Kim, D., Lee, B. and Son, Y. (2020). Design and Implementation of a Digital Evidence Management Model Based on Hyperledger Fabric. J. Inf. Process. Syst., 16, pp.760–773. doi:https://doi.org/10.3745/JIPS.04.0178.

Kahate, A. (2017). Cryptography and Network Security. 3rd ed. McGraw Hill Education India Pvt Ltd.

Kaley, A. (2021). Mapping User Stories in Agile. [online] Nielsen Norman Group. Available at: https://www.nngroup.com/articles/user-story-mapping/ [Accessed 26 Jul. 2023].

Khan, A.A., Uddin, M., Shaikh, A., Laghari, A.A. and Rajput, A. (2021). MF-Ledger: Blockchain Hyperledger Sawtooth-enabled Novel and Secure Multimedia Chain of Custody Forensic Investigation Architecture. IEEE Access, pp.1–1. doi:https://doi.org/10.1109/access.2021.3099037.

Liu, B., Si, X. and Kang, H. (2022). A Literature Review of Blockchain-Based Applications in Supply Chain. Sustainability, 14(22), p.15210. doi:https://doi.org/10.3390/su142215210.

Lone, A. (2017). Forensic-chain: Ethereum blockchain based digital forensics chain of custody. Scientific and practical cyber security journal, Vol. 1.

Lone, A.H. and Mir, R.N. (2019). Forensic-chain: Blockchain based digital forensics chain of custody with PoC in Hyperledger Composer. Digital Investigation, 28, pp.44–55. doi:https://doi.org/10.1016/j.diin.2019.01.002.

Lozupone, V. (2018). Analyze encryption and public key infrastructure (PKI). International Journal of Information Management, 38(1), pp.42–44. doi:https://doi.org/10.1016/j.ijinfomgt.2017.08.004.

Lu, Y. (2019). The blockchain: State-of-the-art and research challenges. Journal of Industrial Information Integration, [online] 15, pp.80–90. doi:https://doi.org/10.1016/j.jii.2019.04.002.

Lucidchart (2019). Introducing Types of UML Diagrams | Lucidchart Blog. [online] Lucidchart.com. Available at: https://www.lucidchart.com/blog/types-of-UML-diagrams [Accessed 27 Jul. 2023].

Majumder, P. (2022). Understanding Distributed Ledger Technology. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2022/07/understanding-distributed-ledger-technology/ [Accessed 24 Jul. 2023].

Mavridis, I. (2015). Information Security on the Internet. Association of Greek Academic Libraries.

Mishra, D. (2021). API Use-Case Prioritization Approach and Methodology. [online] Available at: https://www.linkedin.com/pulse/api-use-case-prioritization-approach-methodology-debasisa-mishra/ [Accessed 26 Jul. 2023].

Nakamoto, S. (2008). Bitcoin: a Peer-to-Peer Electronic Cash System. [online] Available at: https://bitcoin.org/bitcoin.pdf [Accessed 23 Jul 2023].

Oliveira, J., Rosado, M. and Faria P.M. (2020). Zeroconf Network Retail Kiosk for Fish Products Traceability. doi:https://doi.org/10.23919/cisti49556.2020.9141069.

Oliveira, M.T., Carrara, G.R., Fernandes, N.C., Albuquerque, C.V.N., Carrano, R.C., Medeiros, D.S.V. and Mattos, D.M.F. (2019). Towards a Performance Evaluation of Private Blockchain Frameworks using a Realistic Workload. 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN). doi:https://doi.org/10.1109/icin.2019.8685888.

Pedamkar, P. (2019). Application Testing | Complete Guide to Application Testing. [online] Available at: https://www.educba.com/application-testing/ [Accessed 2 Aug. 2023].

Prasad, A. and Pandey, J. (2016). Digital Forensics. Uttrakhand Open University.

Rasjid, Z.E., Soewito, B., Witjaksono, G. and Abdurachman, E. (2017). A review of collisions in cryptographic hash function used in digital forensic tools. Procedia Computer Science, 116, pp.381–392. doi:https://doi.org/10.1016/j.procs.2017.10.072.

Rochmadi, T. and Heksaputra, D. (2019). Forensic Analysis in Cloud Storage with Live Forensics in Windows (Adrive Case Study). International Journal of Cyber-Security and Digital Forensics, Vol. 8(4).

Sathyaprakasan, R., Govindan, P., Alvi, S., Sadath, L., Philip, S. and Singh, N. (2021). An Implementation of Blockchain Technology in Forensic Evidence Management. 2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE). doi:https://doi.org/10.1109/iccike51210.2021.9410791.

Scrum.org (n.d.). What is Scrum? [online] Scrum.org. Available at: https://www.scrum.org/resources/what-scrum-module [Accessed 26 Jul. 2023].

Sinha, D. (2023). What is an Epic in Agile? examples and Differences. [online] knowledgehut.com. Available at: https://www.knowledgehut.com/blog/agile/what-is-an-epic-agile [Accessed 26 Jul. 2023].

Tardi, C. (2021). Genesis Block Definition. [online] Investopedia. Available at: https://www.investopedia.com/terms/g/genesis-block.asp [Accessed 23 Jul 2023].

Tikhomirov, S. (2018). Ethereum: State of Knowledge and Research Perspectives. Foundations and Practice of Security, pp.206–221. doi:https://doi.org/10.1007/978-3-319-75650-9_14.

Umoren, O., Singh, R., Awan, S., Pervez, Z. and Dahal, K. (2022). Blockchain-Based Secure Authentication with Improved Performance for Fog Computing. Sensors, 22(22), p.8969. doi:https://doi.org/10.3390/s22228969.

Wang, W., Hoang, D.T., Hu, P., Xiong, Z., Niyato, D., Wang, P., Wen, Y. and Kim, D.I. (2019). A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks. IEEE Access, [online] 7, pp.22328–22370. doi:https://doi.org/10.1109/access.2019.2896108.

Wheelbarger, S. (2009). History of Computer Forensics. [online] Criminal Justice Collaboratory. Available at: http://colbycriminaljustice.wikidot.com/cyberforensics [Accessed 26 Jul. 2023].

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper. [online] Available at: https://ethereum.github.io/yellowpaper/paper.pdf [Accessed 25 Jul. 2023].

World Bank (2017). International Bank for Reconstruction and Development / the World Bank. In: FinTech Note | No. 1. [online] Washington, DC: World Bank Group. Available at:

https://openknowledge.worldbank.org/server/api/core/bitstreams/5166f335-35db-57d7-9c7e-110f7d018f79/content [Accessed 24 Jul. 2023].

Wrike (2019). What is a Project Charter in Project Management? [online] Wrike.com. Available at: https://www.wrike.com/project-management-guide/faq/what-is-a-project-charter-in-project-management/ [Accessed 26 Jul. 2023].

Xiong, Y. and Du, J. (2019). Electronic evidence preservation model based on blockchain. Proceedings of the 3rd International Conference on Cryptography, Security and Privacy - ICCSP '19. doi:https://doi.org/10.1145/3309074.3309075.

Xu, M., Chen, X. and Kou, G. (2019). A systematic review of blockchain. Financial Innovation, [online] 5(1). doi:https://doi.org/10.1186/s40854-019-0147-z.

9.3. Glossary

Agile: Development methodology

Amazon Web Services (AWS): Amazon's cloud services platform

API: A method of communication between systems

ETH: The Ethereum Blockchain currency

Ethereum: Blockchain Protocol

Gas limit: The limit of "fuel" on the Ethereum Blockchain

Gas price: The price of "fuel" on the Ethereum Blockchain

Gas: The "fuel" of the Ethereum Blockchain

Genesis block: the first block on a blockchain

Geth Javascript Console: Development and interaction environment with a "Geth" node

Geth: Implementation of Ethereum in "Go" programming language

Hyperledger Fabric: Blockchain Platform

MetaMask: Wallet software for interacting with the Ethereum blockchain

Opcode: part of a machine language instruction that specifies the function to be performed

Scrum: Framework of the "Agile" methodology

Scrypt: Hashing algorithm

Single point of failure: A part of a system that, if it fails, will stop the whole system from working.

Turing-Complete: Computationally complete system

Validator: Verifier in the "PoA" consensus algorithm

9.4. File "genesis.json"

```
{
  "config": {
    "chainId": 1997,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "clique": {
      "period": 5,
      "epoch": 30000
    }
  },
  "nonce": "0x0",
  "timestamp": "0x64b9319f",
  "extraData":
"0x0000000000000000000000000000000000000000000000000000000000000000038573b458880555
1f475d737dad9b91b5a1702586f565572b9331025cc5a0861cd6f4a3d7b134dfed2de21f55fb1c2fd8d56f1
cf2a33998030dc9ac0000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000",
  "gasLimit": "0x47b760",
  "difficulty": "0x1",
  "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "alloc": {
    "038573b4588805551f475d737dad9b91b5a17025": {
      "balance": "1000000000000000000000"
    },
    "86f565572b9331025cc5a0861cd6f4a3d7b134df": {
      "balance": "1000000000000000000000"
    },
    "ed2de21f55fb1c2fd8d56f1cf2a33998030dc9ac": {
      "balance": "1000000000000000000000"
    }
  },
  "number": "0x0",
  "gasUsed": "0x0",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000"
}
```

9.5. File "static-nodes.json"

```
[
```

```
"enode://5deb2157dec732289584cb8abfe78a0cdb1f3fcfad956e552a4ba4fa69261fc74fb596c5b566a1673a787850ac7c0a176dd4760dbdd9b07e9353e5643123693b@18.232.0.78:30303",

"enode://5b8af4884487c73faeb298558d73efc2f31c3f50486199206d9108b012001e89f0cd6deb10694d2294222fbd2a6dc2e2e0a02a72faeb9d1b5c1e472ebe1c6f64@52.73.112.80:30303",

"enode://45bfacf8f9a7b61f3a69cb8465b5de4655ae42062d760a21342a575691feca81363ddff20fe0f5c67daf40be31983612a4ec738da34e9da5248ba678f6949a22@52.20.6.247:30303"
]
```

9.6. File “geth.service”

```
//Geth Service Node0
[Unit]
Description=Go Ethereum Client

[Service]
User=root
Type=simple
Restart=always
ExecStart=/usr/local/bin/geth --networkid 1997 --datadir /opt/ThemisChain/node0/data --port 30303 --ipcdisable --syncmode full --rpc --allow-insecure-unlock --rpccorsdomain "*" --rpcport 8545 --rpcaddr "172.31.83.3" --unlock 0xEd2DE21F55Fb1c2Fd8d56f1Cf2A33998030dc9Ac --password /opt/ThemisChain/node0/password.txt --mine --rpcapi personal,admin,db,eth,net,web3,miner,ssh,txpool,debug,clique -ws --wsaddr 172.31.83.3 --wsport 8546 --wsorigins "*" --wsapi personal,admin,db,eth,net,web3,miner,ssh,txpool,debug,clique --maxpeers 25 --etherbase 0 --gasprice 0 --targetgaslimit 99999999

[Install]
WantedBy=default.target

//Geth Service Node1
[Unit]
Description=Go Ethereum Client

[Service]
User=root
Type=simple
Restart=always
ExecStart=/usr/local/bin/geth --networkid 1997 --datadir /opt/Themischain/node1/data --port 30303 --ipcdisable --syncmode full --rpc --allow-insecure-unlock --rpccorsdomain "*" --rpcport 8545 --rpcaddr "172.31.89.21" --unlock 0x038573b4588805551f475d737Dad9b91b5a17025 --password /opt/Themischain/node1/password.txt --mine --rpcapi personal,admin,db,eth,net,web3,miner,ssh,txpool,debug,clique -ws --wsaddr 172.31.89.21 --wsport 8546 --wsorigins "*" --wsapi personal,admin,db,eth,net,web3,miner,ssh,txpool,debug,clique --maxpeers 25 --etherbase 0 --gasprice 0 --targetgaslimit 99999999

[Install]
WantedBy=default.target
```

```
//Geth Service Node2
[Unit]
Description=Go Ethereum Client

[Service]
User=root
Type=simple
Restart=always
ExecStart=/usr/local/bin/geth --networkid 1997 --datadir /opt/ThemisChain/node2/data --port 30303 --ipcdisable --syncmode full --rpc --allow-insecure-unlock --rpccorsdomain "*" --rpcport 8545 --rpcaddr "172.31.80.130" --unlock 0x86f565572b9331025CC5a0861cd6F4A3d7b134dF --password /opt/ThemisChain/node2/password.txt --mine --rpcapi personal,admin,db,eth,net,web3,miner,ssh,txpool,debug,clique -ws --wsaddr 172.31.80.130 --wsport 8546 --wsorigins "*" --wsapi personal,admin,db,eth,net,web3,miner,ssh,txpool,debug,clique --maxpeers 25 --etherbase 0 --gasprice 0 --targetgaslimit 99999999

[Install]
WantedBy=default.target
```

9.7. File “node0account.json”


```
{"address":"ed2de21f55fb1c2fd8d56f1cf2a33998030dc9ac","crypto":{"cipher":"aes-128-ctr","ciphertext":"771ed23a278df5cf495095332f465c146113d607e06c64f6e037396b80961191","cipherparams":{"iv":"a26f7150dd843146361d4c2e267baf27"},"kdf":"scrypt","kdfparams":{"dklen":32,"n":262144,"p":1,"r":8,"salt":"773be3df3310212f13902908ee597517a2285d8e996d386c9303a803897f460a"},"mac":"c15e509cecd4fdfc02fc8e84c07c46d921f82c36341a26431fd684212739d7b0"},"id":"632a7b3c-f2e3-4e6f-99d4-b03160024bb3","version":3}
```


9.8. Smart contracts code

9.8.1. Smart contract “Issued.sol”

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract Issued {

    //Administrator counter
    uint public adminCounter;

    //Administrator info struct
    struct AdminInfo {
        address walletAddress;
        string fullName;
        string email;
        uint mobile;
```

```
        string dob;
    }

    //Administrator lists
    mapping(address => AdminInfo) public adminList;
    mapping(uint => AdminInfo) public adminCountList; //This list is used by the admins to render every
administrator info

    //Constructor used to create three administrators with the setAdminInfo() function. These addresses
correspond to the authorized nodes of the private blockchain (ThemisChain)
    constructor() {
        setAdminInfo(0xEd2DE21F55Fb1c2Fd8d56f1Cf2A33998030dc9Ac, "Christos Bandis",
"chr.bandis@gmail.com", 6973979235, "1997-02-20"); //Node 0
        setAdminInfo(0x038573b4588805551f475d737Dad9b91b5a17025, "Admin NodeOne",
"admin@nodeone.com", 6999999999, "1997-01-01"); //Node 1
        setAdminInfo(0x86f565572b9331025CC5a0861cd6F4A3d7b134dF, "Admin NodeTwo",
"admin@nodetwo.com", 6900000000, "1997-01-02"); //Node 2
    }

    //onlyAdmin modifier used in "Investigator" and "Case" smart contracts
    modifier onlyAdmin() {
        require(adminList[msg.sender].walletAddress == msg.sender, "You are not allowed");
        _;
    }

    //Private function to add an administrator (can be called only within this smart contract)
    function setAdminInfo (address _walletAddress, string memory _fullname, string memory _email, uint
_mobile, string memory _dob) private {
        adminList[_walletAddress].walletAddress = _walletAddress;
        adminList[_walletAddress].fullName = _fullname;
        adminList[_walletAddress].email = _email;
        adminList[_walletAddress].mobile = _mobile;
        adminList[_walletAddress].dob = _dob;

        adminCountList[adminCounter].walletAddress = _walletAddress;
        adminCountList[adminCounter].fullName = _fullname;
        adminCountList[adminCounter].email = _email;
        adminCountList[adminCounter].mobile = _mobile;
        adminCountList[adminCounter].dob = _dob;

        adminCounter++;
    }

    //Function to check if an administrator exists
    function adminExists (address _walletAddress) public view returns (bool) {
        if (adminList[_walletAddress].walletAddress == _walletAddress) {
            return true;
        } else {
            return false;
        }
    }
```

```
}
```

9.8.2. Smart contract “Investigator.sol”

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

import "./Issued.sol";

contract Investigator is Issued {

    //Investigator counter
    uint public investigatorCounter;

    //Array that keeps a history of the IDs assigned to investigators
    uint[] public investigatorIdArray;

    //Investigator info struct
    struct InvestigatorInfo {
        address walletAddress;
        uint invId;
        string fullName;
        string email;
        uint mobile;
        string dob;
        address issuedBy;
    }

    //Investigator lists
    mapping (address => InvestigatorInfo) public investigatorList;
    mapping (uint => InvestigatorInfo) public adminInvestigatorList; //This list is used by the admins to render every investigator info

    //Events
    event IsSuccessful(bool result);
    event AddressExists(bool exists);

    //Function to add an investigator, with the "onlyAdmin" modifier from "Issued" smart contract, which means that only an admin can interact with it
    function addInvestigator (address _walletAddress, uint _invId, string memory _fullName, string memory _email, uint _mobile, string memory _dob, address _issuer) public onlyAdmin {
        investigatorCounter++;    //Investigator counter starts with 1

        if (investigatorList[_walletAddress].walletAddress != _walletAddress) {    //Investigator existence check by wallet address
            emit AddressExists(false);
            investigatorList[_walletAddress].walletAddress = _walletAddress;
            investigatorList[_walletAddress].invId = _invId;
            investigatorList[_walletAddress].fullName = _fullName;
            investigatorList[_walletAddress].email = _email;
            investigatorList[_walletAddress].mobile = _mobile;
```

```
        investigatorList[_walletAddress].dob = _dob;
        investigatorList[_walletAddress].issuedBy = _issuer;

        adminInvestigatorList[investigatorCounter].walletAddress = _walletAddress;
        adminInvestigatorList[investigatorCounter].invId = _invId;
        adminInvestigatorList[investigatorCounter].fullName = _fullName;
        adminInvestigatorList[investigatorCounter].email = _email;
        adminInvestigatorList[investigatorCounter].mobile = _mobile;
        adminInvestigatorList[investigatorCounter].dob = _dob;
        adminInvestigatorList[investigatorCounter].issuedBy = _issuer;

        investigatorIdArray.push(_invId);   //Push investigator ID to the idArray
        emit IsSuccessful(true);
    } else {
        emit AddressExists(true);
    }
}

//Function which returns the name of a certain investigator
function getInvestigatorName (address _walletAddress) public view returns (string memory) {
    return investigatorList[_walletAddress].fullName;
}

//Function to delete an investigator, with the "onlyAdmin" modifier from "Issued" smart contract, which
means that only an admin can interact with it
function removeInvestigator(address _walletAddress, uint _invId) public onlyAdmin {
    investigatorList[_walletAddress].walletAddress =
0x0000000000000000000000000000000000000000;
    investigatorList[_walletAddress].invId = 0;
    investigatorList[_walletAddress].fullName = "";                //In blockchain is impossible to delete
something permanently, like in a traditional
    investigatorList[_walletAddress].email = "";                   //databases. Instead, delete means to
assign the default value in a variable. For example,
    investigatorList[_walletAddress].mobile = 0;                   //the default value of a string variable is
"" (empty) and of a uint is 0
    investigatorList[_walletAddress].dob = "";
    investigatorList[_walletAddress].issuedBy = 0x0000000000000000000000000000000000000000;

    adminInvestigatorList[_invId].walletAddress = 0x0000000000000000000000000000000000000000;
    adminInvestigatorList[_invId].invId = 0;
    adminInvestigatorList[_invId].fullName = "";
    adminInvestigatorList[_invId].email = "";
    adminInvestigatorList[_invId].mobile = 0;
    adminInvestigatorList[_invId].dob = "";
    adminInvestigatorList[_invId].issuedBy = 0x0000000000000000000000000000000000000000;

    emit IsSuccessful(true);
}

//Function to check if an investigator exists
function investigatorExists (address _walletAddress) public view returns (bool) {
```

```
        if (investigatorList[_walletAddress].walletAddress == _walletAddress) {
            return true;
        } else {
            return false;
        }
    }

    //Function to check if an investigator 's ID in the IdArray
    function invIdExists (uint _invId) public view returns (bool) {
        for (uint i = 0; i < investigatorIdArray.length; i++) {
            if (investigatorIdArray[i] == _invId) {
                return true;
            }
        }
        return false;
    }
}
```

9.8.3. Smart Contract “Case.sol”

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

import "./Evidence.sol";
import "./Issued.sol";

contract Case is Evidence, Issued {

    //Case ID counter
    uint public caseId;

    //Case info struct
    struct CaseInfo {
        uint caseNum;
        string caseName;
        string caseDesc;
        address[] caseInvestigators;
        address creator;
        string creationTime;
        mapping (uint => Evidence.EvidenceInfo[]) evidenceList;
        bool closed; //default false
    }

    //(Not used) Modifier to check if an evidence was deleted
/*  modifier evidenceNotDeleted(uint _caseId, uint _evidenceId) {
        bool isDeleted = caseList[_caseId].evidenceList[_caseId][_evidenceId].deleted;
        require (isDeleted == false, "There is no such evidence");
        _;
    } */

    //Case List
```

```
    mapping (uint => CaseInfo) public caseList;

    //Events
    event CaseExists(bool result);
    event IsSuccessful(bool result, uint id);
    event IsSuccessful_Evidence(bool result);
    event IsSuccessful_Investigator(bool result);

    //Function to open a new case, with the "onlyAdmin" modifier from "Issued" smart contract, which
means that only an admin can interact with it
    function openNewCase (string memory _caseName, string memory _caseDesc, address
_caseInvestigator, address _caseCreator, string memory _creationTime) public onlyAdmin {
        caseId++;   //Case Id starts with 1

        if (caseList[caseId].caseNum != caseId) {   //Case Id existence check
            emit CaseExists(false);
            caseList[caseId].caseNum = caseId;
            caseList[caseId].caseName = _caseName;
            caseList[caseId].caseDesc = _caseDesc;
            caseList[caseId].caseInvestigators.push(_caseInvestigator);
            caseList[caseId].creator = _caseCreator;
            caseList[caseId].creationTime = _creationTime;
            caseList[caseId].closed = false;

            emit IsSuccessful(true, caseId);
        } else {
            emit CaseExists(true);
        }
    }

    //Function to check if a case exists based on its ID
    function caseExists (uint _caseId) public view returns (bool) {
        if (caseList[_caseId].caseNum == _caseId) {
            return true;
        }
    }

    //Function which returns the name of a case based on its ID
    function getCaseName (uint _caseId) public view returns (string memory) {
        return caseList[_caseId].caseName;
    }

    //Function which returns the active cases of a certain investigator
    function activeCases (uint _caseId, address _walletAddress) public view returns (uint, string memory,
string memory, address, string memory, bool) {
        CaseInfo storage thisCase = caseList[_caseId];

        address[] memory caseInvestigatorsArray = caseList[_caseId].caseInvestigators;
        uint len = caseInvestigatorsArray.length;

        for (uint i = 0 ; i < len; i++){
```

```
            if (caseList[_caseId].caseInvestigators[i] == _walletAddress) {
                return (thisCase.caseNum, thisCase.caseName, thisCase.caseDesc, thisCase.creator,
thisCase.creationTime, thisCase.closed);
            }
        }
    }

    //Function to close a case based on its ID and delete its evidence, with the "onlyAdmin" modifier from
"Issued" smart contract, which means that only an admin can interact with it
    function closeCase (uint _caseId) public onlyAdmin {
        caseList[_caseId].caseNum = 0;                                  //In blockchain is impossible to
delete something permanently, like in a traditional
        caseList[_caseId].caseName = "";                                //database. Instead, delete
means to assign the default value in a variable. For example,
        caseList[_caseId].caseDesc = "";                                //the default value of a string
variable is "" (empty) and of a uint is 0
        caseList[_caseId].creator = 0x0000000000000000000000000000000000000000;
        caseList[_caseId].creationTime = "";
        caseList[_caseId].closed = true;

        uint caseLen = caseList[_caseId].caseInvestigators.length;
        for (uint i = 0 ; i < caseLen; i++){
            delete caseList[_caseId].caseInvestigators[i];
        }

        Evidence.EvidenceInfo[] storage thisEvidence = caseList[_caseId].evidenceList[_caseId];

        uint evidLen = thisEvidence.length;
        for (uint j = 0 ; j < evidLen; j++){
            delete thisEvidence[j];          //Delete doesn 't actually delete the element but it assigns the
default value to it (see comment above)
        }
    }

    //Function to assign an investigator to a case, with the "onlyAdmin" modifier from "Issued" smart
contract, which means that only an admin can interact with it
    function addCaseInvestigator (uint _caseId, address _investigator) public onlyAdmin {
        caseList[_caseId].caseInvestigators.push(_investigator);
        emit IsSuccessful_Investigator(true);
    }

    //Function to remove an investigator from a case, with the "onlyAdmin" modifier from "Issued" smart
contract, which means that only an admin can interact with it
    function removeCaseInvestigator (uint _caseId, address _investigator) public onlyAdmin {
        caseList[_caseId].caseInvestigators.push(_investigator);

        uint caseLen = caseList[_caseId].caseInvestigators.length;
        for (uint i = 0 ; i < caseLen; i++){
            if (caseList[_caseId].caseInvestigators[i] == _investigator) {
                delete caseList[_caseId].caseInvestigators[i];
            }
```

```
        }
        emit IsSuccessful_Investigator(true);
    }

    //Function which returns the investigators of a particular case and the total number of them
    function getCaseInvestigators (uint _caseId) public view returns (address[] memory, uint) {
        address[] memory CaseInvestigators = caseList[_caseId].caseInvestigators;
        uint len = CaseInvestigators.length;
        address[] memory investigatorResult = new address[](len);

        for (uint i = 0 ; i < len; i++){
            investigatorResult[i] = CaseInvestigators[i];
        }
        return (investigatorResult, len);
    }

    //Function to check if a case has already been assigned to an investigator
    function invExistsInCase (uint _caseId, address _walletAddress) public view returns (bool) {
        address[] memory CaseInvestigators = caseList[_caseId].caseInvestigators;
        uint len = CaseInvestigators.length;

        for (uint i = 0 ; i < len; i++){
            if (CaseInvestigators[i] == _walletAddress) {
                return true;
            }
        }
    }

    //Function to add evidence to a case, with the "onlyAdmin" modifier from "Issued" smart contract,
which means that only an admin can interact with it
    function addCaseEvidence (uint _caseId, bytes memory _evidenceHash, string memory
_evidenceName, string memory _evidenceDesc, address _creator, string memory _dateCreated) public
onlyAdmin {
        Evidence.EvidenceInfo[] storage thisEvidence = caseList[_caseId].evidenceList[_caseId];
        Evidence.EvidenceInfo storage singleEvidence = thisEvidence.push();

        singleEvidence.evidenceHash = _evidenceHash;
        singleEvidence.evidenceName = _evidenceName;
        singleEvidence.evidenceDesc = _evidenceDesc;
        singleEvidence.creator = _creator;
        singleEvidence.dateCreated = _dateCreated;
        singleEvidence.transferChain.push(_creator);
        singleEvidence.transferDesc.push("Creation");
        singleEvidence.transferTime.push(_dateCreated);
        singleEvidence.deleted = false;

        emit IsSuccessful_Evidence(true);
    }

    //Function which returns evidence 's info
```

```
    function getCaseEvidenceCollapsed(uint _caseId, uint _evidenceId) public view returns (bytes
memory, string memory, string memory, bool) {
        Evidence.EvidenceInfo memory thisEvidence =
caseList[_caseId].evidenceList[_caseId][_evidenceId];
        return (thisEvidence.evidenceHash, thisEvidence.evidenceName, thisEvidence.dateCreated,
thisEvidence.deleted);
    }

    //Function which returns evidence 's info
    function getCaseEvidenceExpanded(uint _caseId, uint _evidenceId) public view returns (bytes
memory, string memory, string memory, address, string memory) {
        Evidence.EvidenceInfo memory thisEvidence =
caseList[_caseId].evidenceList[_caseId][_evidenceId];
        return (thisEvidence.evidenceHash, thisEvidence.evidenceName, thisEvidence.evidenceDesc,
thisEvidence.creator, thisEvidence.dateCreated);
    }

    //Function which returns evidence 's hash
    function getEvidenceHash(uint _caseId, uint _evidenceId) public view returns (bytes memory) {
        Evidence.EvidenceInfo memory thisEvidence =
caseList[_caseId].evidenceList[_caseId][_evidenceId];
        return (thisEvidence.evidenceHash);
    }

    //Function which returns the total number of the evidences in a case
    function evidenceCount (uint _caseId) public view returns (uint){
        Evidence.EvidenceInfo[] storage thisEvidence = caseList[_caseId].evidenceList[_caseId];
        return thisEvidence.length;
    }

    //Function which returns the current owner of an evidence
    function getCurrentOwner (uint _caseId, uint _evidenceId) public view returns (address) {
        Evidence.EvidenceInfo memory thisEvidence =
caseList[_caseId].evidenceList[_caseId][_evidenceId];
        address[] memory chain = thisEvidence.transferChain;
        return chain[chain.length - 1];
    }

    //Function to transfer an evidence from one investigator to another, with the "onlyAdmin" modifier from
"Issued" smart contract, which means that only an admin can interact with it
    function transferEvidence (uint _caseId, uint _evidenceId, address _transferTo, string memory
_transferDesc, string memory _transferTime) public onlyAdmin {
        Evidence.EvidenceInfo storage thisEvidence =
caseList[_caseId].evidenceList[_caseId][_evidenceId];

        thisEvidence.transferChain.push(_transferTo);
        thisEvidence.transferDesc.push(_transferDesc);
        thisEvidence.transferTime.push(_transferTime);

        emit IsSuccessful_Evidence(true);
    }
```

```
    //Function which returns the chain of custody of an evidence
    function viewChainOfCustody (uint _caseId, uint _evidenceId) public view returns (uint, string
memory, bytes memory, address[] memory, string[] memory, string[] memory) {
        Evidence.EvidenceInfo memory thisEvidence =
caseList[_caseId].evidenceList[_caseId][_evidenceId];

        address[] memory transfChain = thisEvidence.transferChain;
        uint transfArrayLen = transfChain.length;       //transfArrayLen is the same as the length of
timeChain and descChain
        address[] memory transfResult = new address[](transfArrayLen);

        string[] memory timeChain = thisEvidence.transferTime;
        string[] memory timeResult = new string[](transfArrayLen);

        string[] memory descChain = thisEvidence.transferDesc;
        string[] memory descResult = new string[](transfArrayLen);

        for (uint i = 0 ; i < transfArrayLen; i++){
            transfResult[i] = transfChain[i];
            timeResult[i] = timeChain[i];
            descResult[i] = descChain[i];
        }
        return (transfArrayLen, thisEvidence.evidenceName, thisEvidence.evidenceHash, transfChain,
timeChain, descChain);
    }

    //Function to delete a certain evidence, with the "onlyAdmin" modifier from "Issued" smart contract,
which means that only an admin can interact with it
    function deleteEvidence (uint _caseId, uint _evidenceId) public onlyAdmin {
        Evidence.EvidenceInfo storage thisEvidence =
caseList[_caseId].evidenceList[_caseId][_evidenceId];

        thisEvidence.evidenceHash =
"0x0000000000000000000000000000000000000000000000000000000000000000";
        thisEvidence.evidenceName = "";
        thisEvidence.evidenceDesc = "";
        thisEvidence.creator = 0x0000000000000000000000000000000000000000;
        thisEvidence.dateCreated = "";

        uint transfArrayLen = thisEvidence.transferChain.length;
        for (uint i = 0 ; i < transfArrayLen; i++){
            thisEvidence.transferChain[i] = 0x0000000000000000000000000000000000000000;
            thisEvidence.transferDesc[i] = "";
            thisEvidence.transferTime[i] = "";
        }
        thisEvidence.deleted = true;

        emit IsSuccessful_Evidence(true);
    }
}
```

9.8.4. Smart contract “Evidence.sol”

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract Evidence {

    //Evidence info struct (used in "Case" smart contract)
    struct EvidenceInfo {
        bytes evidenceHash;
        string evidenceName;
        string evidenceDesc;
        address creator;
        string dateCreated;
        address[] transferChain;
        string[] transferDesc;
        string[] transferTime;
        bool deleted; //default false
    }
}
```