ΜΗΤΡΟΠΟΛΙΤΙΚΟ ΚΟΛΛΕΓΙΟ

University of East London

# SCHOOL OF ARCHITECTURE, COMPUTING & ENGINEERING

**BSc in Computer Science**

## Thesis entitled:

Authentication using Blockchain technology


## THESIS EDITING

BANDIS CHRISTOS
U2121211

## SUPERVISOR:

BOURAKIS ANDREAS

## INSTITUTION

METROPOLITAN COLLEGE CAMPUS
THESSALONIKIS


*May, 2022*

# Table of Contents

# Abstract

A multitude of public and non-public services are now available in digital form, making them accessible to all citizens through their smart devices. The digitization of services has led to an increase in the rate of development of relevant applications, which are often characterized by their low cost that comes with the downside of a system vulnerable to errors and threats. The services offered, until recently, by the providers, make use of the AAA (Authentication, Authorization and Accounting) Framework, which is based on the Client - Server Model (Vishnia and Peters, 2020). The main feature and vulnerability of this model is the fact that service providers maintain full control over the data of the citizens they process and are therefore a frequent target of malicious users. Blockchain technology, thanks to the principles it professes, may bring about significant changes in the ways personal data is managed, but also in the way the entire internet operates, giving it a more decentralized character. Although it has existed in the field for decades, Blockchain has gained its reputation in recent years through cryptocurrencies and more specifically Bitcoin, the first of them. However, it is still in the early stages of development. Through the elaboration of this thesis, an extensive research will be carried out in the wider field of Blockchain and digital identity and an application will be developed that will enable the decentralized identification of citizens in a Blockchain authentication service.

**Keywords**

Smart Devices, Blockchain, Personal Data, Internet, Cryptocurrencies, Decentralized Identification

# Project scheduling

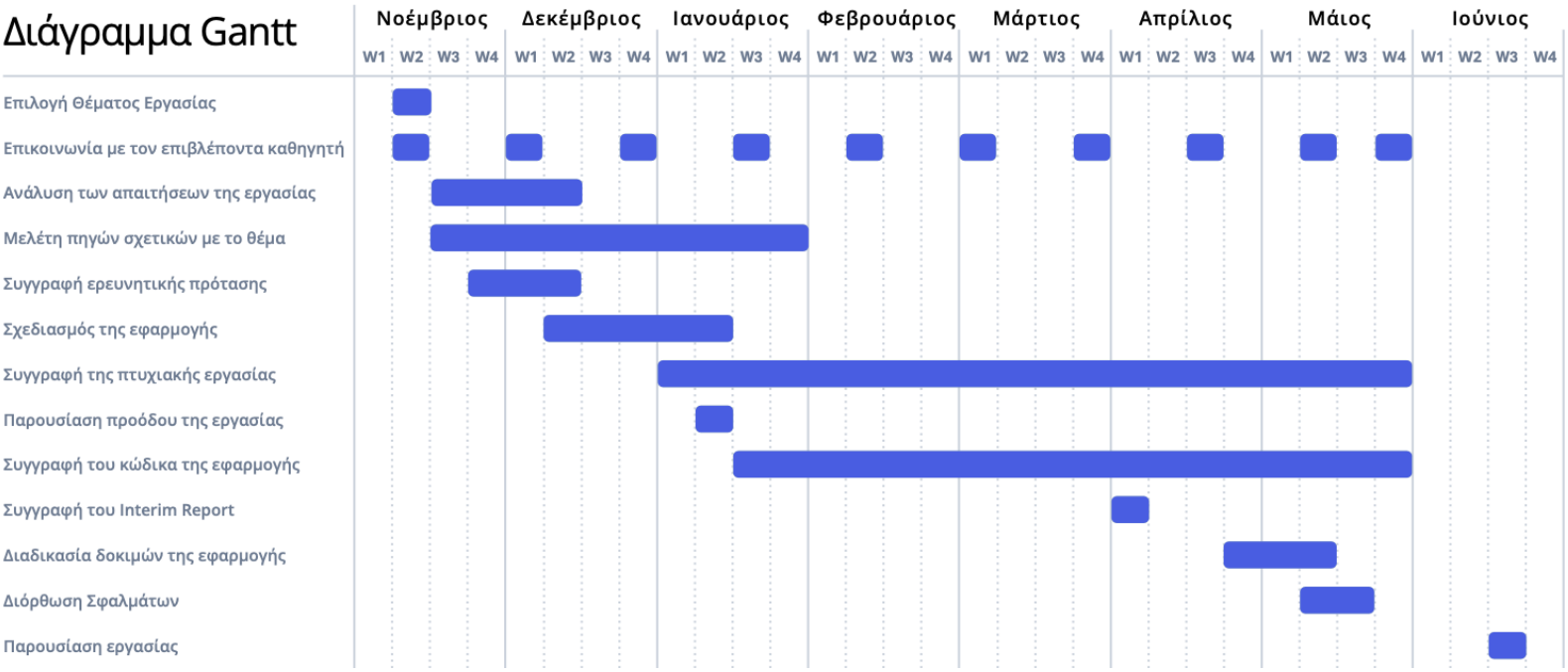


*Figure 1 - Gantt chart*

The above Gantt chart shows the expected time span of the project, from the assignment of the subject to its delivery. The time is measured in weeks and starts in the second week of November, when the selection of the project topic takes place. The method of time-sharing of activities aims at a more efficient completion of the project. For example, communication with the supervisor was set to take place every three weeks in order to have sufficient data for presentation and feedback. It is also worth mentioning that the analysis of the project requirements took the equivalent of one month, due to the complexity of the topic. The study of the sources, which starts at the same time as the analysis of the requirements, took eight (8) weeks, as it was preceded by research and evaluation of their content. The time spent on the design of the application did not exceed five (5) weeks, as the simplicity and yet usability of the application was a key criterion for its implementation. Finally, the remaining time was dedicated to writing the thesis and the code of the application along with the testing procedure and the errors that occurred.

# 1. Introduction

The history of Blockchain goes back to 1991, when scientists Stuart Haber and W. Scott Stornetta presented a technology for timestamping digitized files in order to validate their integrity and prevent corruption of the data they contained. This technology consisted of a system that made use of an encrypted and therefore secure "chain" of blocks in which the files were stored (Iredale, 2018).

This technology was used for the first time seven (7) years later, in 1998, when computer scientist and Blockchain pioneer Nick Szabo presented "Bit Gold", the first attempt to create a decentralized digital currency, which until today was considered the ancestor of Bitcoin (Hayes, 2022). The BitGold concept, however, did not succeed, as it succumbed to a major vulnerability known as the "Double-Spending Problem", in which Bit Gold holders were allowed to spend twice the number of coins they held. Thus, the evolution of Blockchain technology stagnated until, in 2008, a developer or group of developers under the name "Satoshi Nakamoto" (whose identity remains unknown to this day) presented a paper concerning a new digital currency, Bitcoin, addressing the weakness of its "ancestor" and making integrated use of Blockchain technology (Iredale, 2018).

Blockchain is defined as the technology in which the ownership of information and data concerning the sectors of the economy is verified, but also having political, social and legal implications (Logaras, 2018). The main characteristic of this technology is that a continuous chain of data is created, the contents of which are stored in blocks that are inextricably linked to each other in chronological order. Through Bitcoin, the Blockchain technology has developed significantly and has become widely known, so that it is used in more and more ways (e.g. Money Transfer, Decentralized Applications, Smart Contracts).

In the modern era, when the use of the internet is a basic part of everyday life, citizens need to ensure the protection of their personal data. Identity, perhaps the most important of these, is the cornerstone of human development and is involved in all types of transactions between the state, citizens and business organizations, which, combined with the improper use of the Internet, can turn it into a tool for surveillance, violating the principles of human rights. This is the reason why a new, human-centric model is needed for identity management systems. In this way, every citizen will have greater

control over their identity and, to a greater extent, over their personal data, and will be able to choose which entities they trust to give access to their data. The implementation of such a system requires the existence of strong cryptographic technology, transparency and direct accessibility, characteristics that are found, among others, in Blockchain technology. As Imran Bashir (2017) states, “The computer code becomes the law”.

This paper is divided into chapters in such a way as to achieve a complete understanding of the subject, analyzing terms and areas such as: the Blockchain technology and Digital Identity. More specifically, within the next chapter, Blockchain, its structure, Ethereum Blockchain, Decentralized Applications and Smart Contracts will be discussed. Then, the elements of Digital Identity, the known methods of identification, the present identity management systems in general, the concepts of security and privacy and the ways in which electronic identification will benefit from the principles of Blockchain technology will be examined. Finally, the details of the development plan of a fully functional decentralized Blockchain-based identification application developed for the needs of this thesis will be discussed, as well as the use case addressed by this particular implementation of the application.

# 2. The Blockchain

Originally, in computer science, the term Blockchain corresponded to a form of data structure and sharing. In today's terms, Blockchain is a new approach to distributed databases, which are controlled by groups of individuals and used to store and share information. More specifically, Blockchain is defined as a data structure in which it is possible to create a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a business network (IBM, n.d.). An asset can be tangible (house, car, cash, land) or intangible (intellectual property, patents, copyrights, brand name). Almost anything of value can be identified and traded on a Blockchain, reducing risk and cost for all parties involved (IBM, n.d.).

Blockchain is divided into four types: public, permissioned, private and consortium:

- Public Blockchains, such as Bitcoin, are large-scale distributed networks, free for all to participate in, requiring the use of a native token for their operation and consisting of open-source code, maintained by the respective community. This type of Blockchain is not suitable for enterprise use cases, as it is characterized by the significant computing power required and the lack of transaction privacy.
- Permissioned Blockchains, such as Ripple, differ in part from public Blockchains because the roles of their constituent units are controlled, since participation is only allowed by invitation. However, they remain large, distributed systems whose operation requires the use of a native token. As far as their code is concerned, it can be either “open” or “closed”.
- Private Blockchains tend to be smaller than the aforementioned and do not make use of a token. They are usually run by an organization that controls who joins and maintains the ledger, thus enhancing trust and confidentiality between participants. A private Blockchain can operate behind a corporate firewall and even be hosted on the premises of the company in question.
- Consortium Blockchains are made up of a number of organizations, which share the responsibility of maintaining them. These pre-selected organizations determine transaction submission and data access permissions. Such

Blockchains are considered ideal for businesses that have trusted members and process confidential information.

Each of the above types, taking advantage of their strong encryption layers, as well as asymmetric public-private key encryption, allow the secure management of the ledger by the participants, without the presence of a central authority to enforce rules. The removal of the central authority from the prevailing database structure is one of the most significant changes brought about by Blockchain technology (Laurence, 2017).

Blockchain is now being called the "fifth evolution" of computing or "the lost trust level of the internet" (Laurence, 2017). It is the ideal solution for organizations that cannot afford to suffer a single point of failure, as it makes it virtually impossible for sensitive information to be compromised by malicious users and cybercriminals.

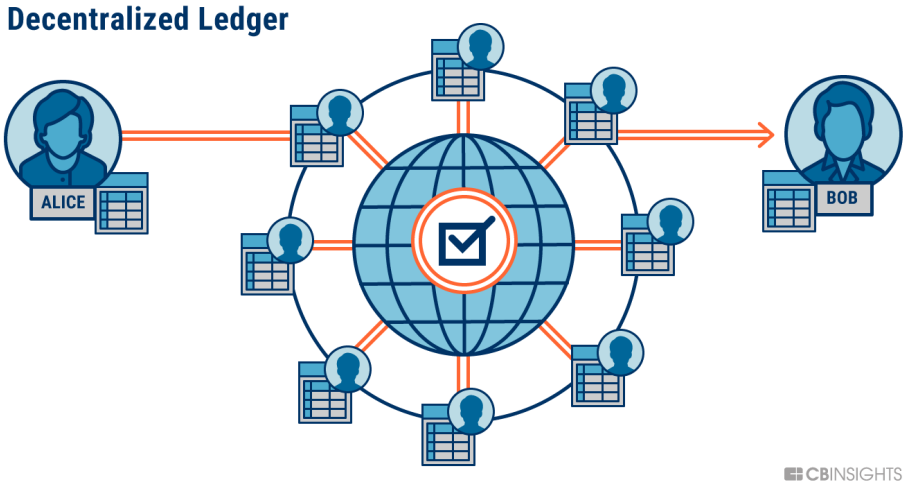


*Figure 2 - Representation of a Blockchain network (CB Insights Research, 2018)*

## 2.1. The structure of the Blockchain

Blockchain technology is not new to computer science, as it is based on a group of existing technologies that are widely used across the industry. In particular, Blockchain is a peer-to-peer network, structured by blocks, which are distributed ledgers containing the recorded transactions that took place at a certain point in time and are updated by each participant in the Blockchain (Gupta, 2018). Then, the blocks are linked together (chaining) through a cryptographic mathematical representation (hash). This mathematical representation is designed to protect the integrity of the data, since each new block contains the hash of the previous one and in this way makes it immediately perceivable to modify already registered data. Finally, it is made up of a network of independent computers called “nodes” which manage all the transactions that take place within the Blockchain.

More specifically, a block is a single unit on the Blockchain that has the form of a data structure and is composed of meta-data. A miner collects the valid transactions recorded in a certain period of time and, in combination with the hash of the previous block, calculates the new hash that is extracted. However, each hash has a unique form, for the calculation of which the miner, after continuous efforts, has to discover an arbitrary number corresponding to it. This number is called “number used once” or “number once” (nonce) and separates the blocks into “signed” (those with nonce) and “unsigned” (those without nonce). The nonce discovery process is called “mining” (Shackelford and Myers, 2016).

If any tampering takes place in the data of a pre-existing block, the hashes of the subsequent blocks, which are successively linked to it, will change, thus creating an immutable log that will inform the Blockchain participants of the event. Given this, one by one the hashes of the blocks lead to the first of them under the name “Genesis Block”.

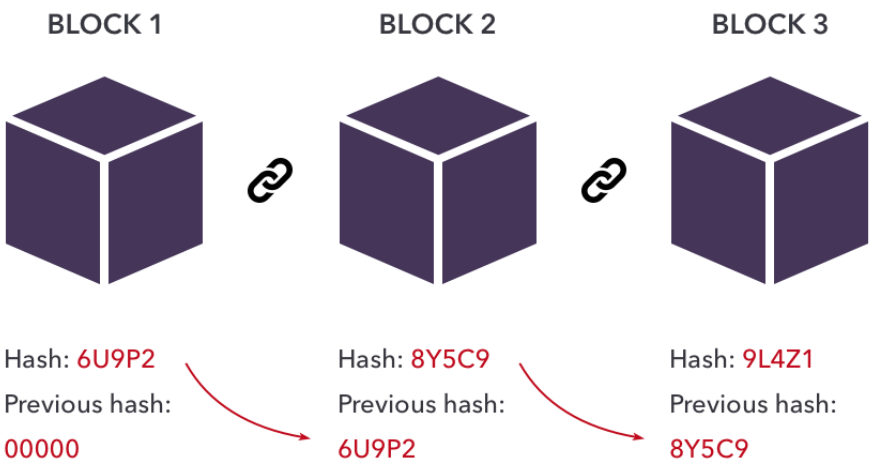


*Figure 3 - Example of a connection between the "block"*

As for the cryptographic function that creates the hash, it is the way to extract a fixed-length alphanumeric sequence from a string of any given length. This sequence, which, in addition to "hash", is called a "Message Digest", cannot be reversed to obtain the input data and therefore can be used to check the integrity of the data (Gauravaram et al., 2006). The hash produced from a data set is always the same, no matter how many times it is recomputed, and a small change results in a completely different result. Hence, the cryptographic function is also known by the designation "One-way hash function".

A hash has three main properties: "collision free", "hiding" and "puzzle friendly". Collision-free means that it is extremely unlikely that two different input data will be found to have the same hash. For example, the hash of a string "x" and the hash of a string "y" are always different, despite the number of recalculations. With the property of "hiding", it becomes infeasible to convert the hash to its original form, i.e., the input data, and the phrase "puzzle-friendly" represents the ease of computing a hash of given data (Treiblmaier, 2019).

There are several ways of calculating a hash. In the cryptocurrency world, with Bitcoin being a popular example, the SHA-265 algorithm is used to generate a 256-bit fixed-length hash in each block.

Also worth mentioning is the "hash tree" or "Merkle Tree". A Merkle Tree is a binary tree, consisting of hash pointers, which ensures that each node must have the same, uncorrupted, unchanged and valid data. If there is a modification of the data in one node, then the changes must be propagated to the others. The Merkle Tree consists of blocks that contain transactions and are placed, in groups of two, in the leaves of the

tree. Each group of blocks has hash pointers, the combination of which corresponds to the next (nearest to the root) level of the tree. This process is repeated until a single block is created which is called the root hash or root of the tree (Jal, 2018).

In peer-to-peer networks, data verification is considered time-consuming and computationally expensive. Using a Merkle Tree, in place of the data, only the hash is sent to the receiver, who then verifies it based on the root of the tree. In this way, it is possible to securely and efficiently verify larger data structures, as well as ensure their integrity.

In conclusion, in the client-server model, which is found in a multitude of web applications, user data is stored on centralized servers and is under the exclusive control of their administrators, resulting in the modification or deletion of databases if the security of the administrators is breached.

In peer-to-peer networks, and thus in Blockchain, nodes provide a portion of their resources, such as a percentage of their processing power or storage space, which are readily available to all participants without the need for a central server. Unlike the aforementioned databases, all nodes of a Blockchain maintain a copy of the data, resulting in the information remaining available even if some nodes go offline.

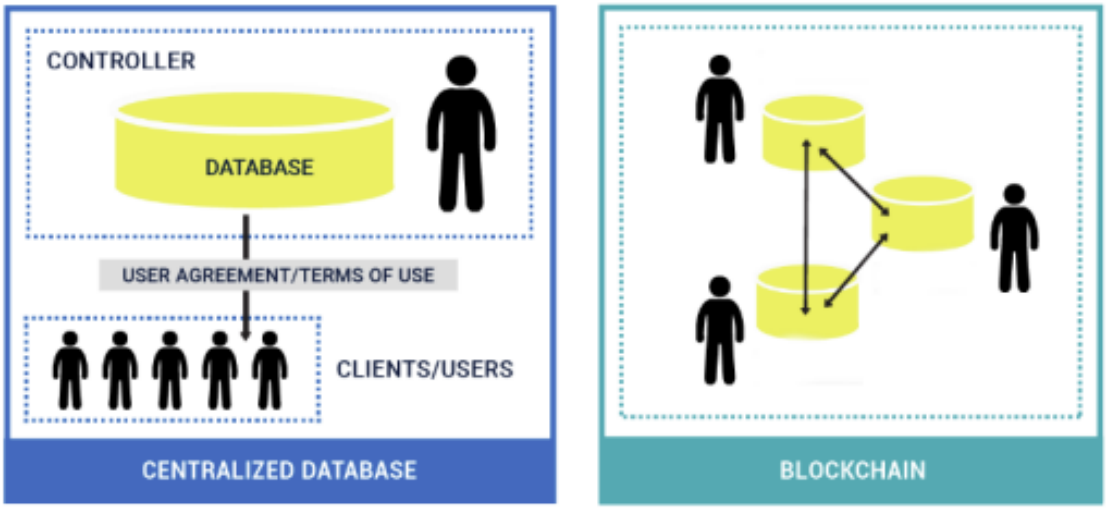


*Figure 4 - Comparison between centralized database and Blockchain*

## 2.2. Consensus Mechanism

The popularity of the Blockchain technology comes from the creation of Bitcoin. An outgrowth of this rapid development is the demonstration that a group of individuals, unknown to each other, could be active on the internet in a system that desensitized participants to issues of deception among themselves. In the world of

Blockchain, consensus is the process of developing an agreement between a group of individuals, usually riddled with distrust, to ensure that everything has been considered before actually communicating. As everyone can participate and submit information, evaluating everyone's goals and unanimously deciding on a desired policy are beneficial tactics to avoid any attempts at fraud.

Each Blockchain has its own algorithms for creating agreement within its network on the entries added to it. There are different models for consensus creation because each Blockchain is used for different purposes. Some are engaged in transacting commercial value, others are available for data storage, and others delve into system and contract security. The expected threat and the degree of trust the network has in the nodes involved in the Blockchain operation will determine the type of consensus algorithm they will resort to in order to settle their ledger. For example, Bitcoin and Ethereum are subject to high-risk threats, so they make use of a strong consensus algorithm called Proof of Work (PoW). Some other algorithms are Proof of Stake (PoS), Delegated Proof of Stake (DPoS) and Proof of Burn (PoB).

Consensus algorithms solve the “Byzantine Generals' Problem” (Byzantine fault): “How do we know that the information we are looking at has not changed internally or externally?”. Due to human intervention in data manipulation being almost always possible, data reliability is a major problem for computer science.

## 2.3 Ethereum Blockchain

Ethereum is one of the most developed and accessible Blockchains in the ecosystem. It is considered the industry leader in innovation and use cases of Blockchain technology. Understanding this technology is important because it is leading the way in the areas of Smart Contracts and Decentralized Autonomous Organizations (DAOs).

Ethereum may be one of the most complex Blockchains ever built. It retains the “traditional” structure of a Blockchain however, it adds at its core a “Turing-Complete” programming language (a complete language that allows programmers to solve any computational problem) and the “Ethereum Virtual Machine” (EVM), which specializes in executing the code of smart contracts. The Ethereum protocol can cope with almost any issue raised in the average of the known programming languages, with

the differentiation that it comes with the benefits and security of Blockchain due to its integration with it (Laurence, 2017).

The Ethereum protocol has established a whole new kind of applications, whereby any government, business or organization has the ability to gain existence within Ethereum. Currently, the Ethereum platform is being explored for digital asset management (a new class of online assets that can represent an entire digital asset, such as a Bitcoin currency or a digital representation of a real-world asset, such as a work of art), financial instruments (such as mortgage-backed securities), asset ownership registration, such as land and decentralized autonomous organizations (DAOs), a new way to organize and manage digital assets. Finally, it is also used to secure Blockchain applications and smaller Blockchain networks (Laurence, 2017).

## 2.4. Decentralized Applications (Decentralized Applications)

Decentralized Applications (dApps) are digital applications or programs that exist and run on a Blockchain, as opposed to traditional applications that run on a server. The dApps are outside the jurisdiction and control of a single authority and use smart contracts for their logic (Frankenfield, 2021).

A centralized application is owned by a single company, while its software resides on one or more servers controlled by the same company. The user interacts with the application by sending and receiving data to and from the company's server. Once the data is stored on the server, the user is deprived of the right to control it and has no access to information about how it is stored, what security precautions have been taken, who can read it and so on. In contrast, a decentralized application running on a Blockchain allows users to transact directly with each other, avoiding dependence on one entity to store and manage their personal and business data, gaining full control over them (Figure 5). Therefore, user privacy and lack of censorship are ensured.

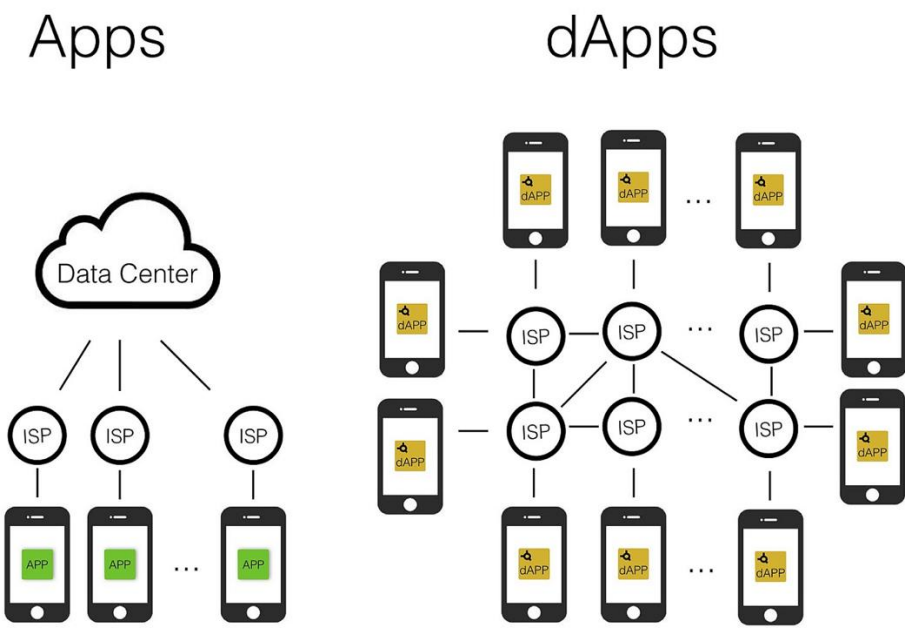


*Figure 5 - Graphical representation and comparison of centralized and decentralized applications (Ray, 2021)*

Ethereum is a flexible platform for creating decentralized applications, providing the infrastructure needed for developers to focus their efforts on finding innovative uses for applications. This facilitates the rapid growth of dApps in a variety of industries, including banking and finance, gaming, social media and online shopping (Frankenfield, 2021).

The dApps primarily, use the programming language “Javascript” as an interface to a node on the Ethereum Blockchain, which they then communicate with via a smart contract. They also provide a user interface to facilitate the management of connections to them (Rajneesh Gupta, 2018).

The maintenance of a decentralized application is worthwhile. Once deployed, a dApp will likely need ongoing changes for optimization purposes or to correct bugs or security risks. According to Ethereum, updating dApps is an intractable problem for developers because the data and code deployed to the Blockchain are difficult to modify.

## 2.5. Smart Contracts

Smart contracts are programs stored in a blockchain that are executed when predefined conditions are met. They are typically used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without the involvement or loss of time of any intermediary. They can also automate a workflow, moving to the next action when conditions are met (IBM, 2022). As early as 1990, Nick Szabo described these contracts as “a computational transaction protocol that executes the terms of a contract-agreement”. It can therefore be understood that contracts of this type are characterized by three specificities: automatic enforceability, self-validation and the ability to be understandable, secure and unambiguous (Bashir, 2017).

Although they can be programmed for any supported Blockchain type, Ethereum is the widely preferred choice, as it provides scalability of processing. In Ethereum, each contract is assigned a unique address so that it can be identified. This address is calculated by hashing its creator's address on the Blockchain and the number of transactions executed.

The storage of data on a Blockchain is characterized by its immutability, however the same is not true for smart contracts. Each smart contract executed on Ethereum maintains its own variable storage space. This storage space can be thought of as a sizable array, initially consisting of zeros. Each value in the array is 32 bytes wide and there is a total of $2^{256}$ such values. A smart contract can read or write to a value located at any location in its storage (Marx, 2018).

So far, smart contracts have been extended to a multitude of sectors. The ability to issue and verify digital identity, international transfer of goods through credit mechanisms and simplify processes without further costs, clarity in financial systems and transactions, and efficiency of connectivity between constituent parties are some of the use cases now being considered globally (Chamber of Digital Commerce, 2016).

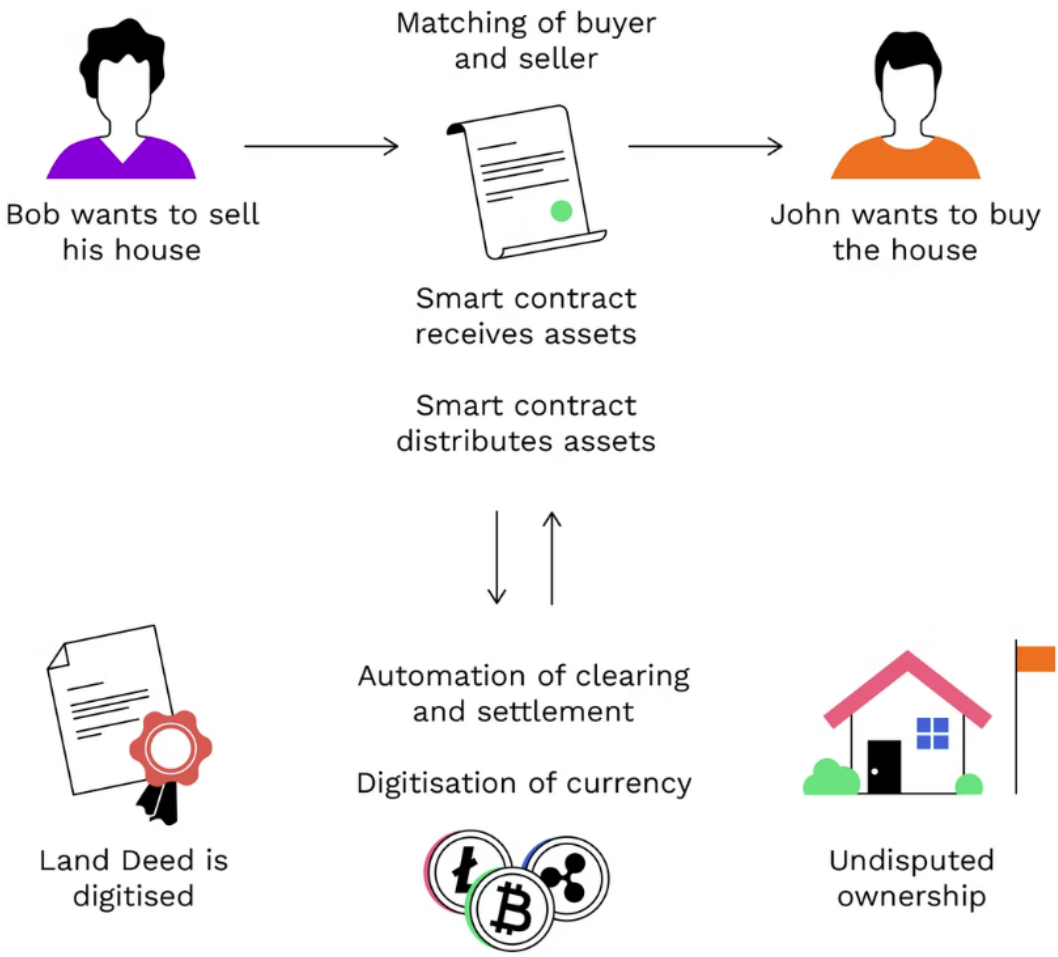


*Figure 6 - Graphical representation of a use case of a smart contract (Bitpanda, n.d.)*

# 3. The digital identity

As a natural extension of the definition of identity, digital identity is characterized by a finite set of attributes that allows a person, animal, thing or process to be uniquely identifiable and their authenticity to be electronically authenticated to third parties (Allende López, 2020). Each digital identity is represented by one or more identifiers and a set of attributes that are unique within a defined context.

Proving a person's digital identity faces several challenges. For example, identity verification is no longer applied in the form that applies to physical forms of identity. However, it also presents many advantages, as it offers access to global digital services without the need for physical presence or a physical form of identity. This benefits in interacting with a multitude of possibilities, allowing remote delivery of services, otherwise considered inaccessible, in real time to communities and populations with limited access (Allende López, 2020).

Digital identities are created and used as part of a lifecycle that includes four fundamental stages: a) registration, including validation; b) issuing documents or credentials; c) authentication; and d) identification for the provision of services or transactions (World Bank, 2018a).


*Figure 7 - Examples of digital identity in three different cases (Allende López, 2020)*

Digital identity enables individuals to set aside the constraints of the real world, facilitates the immediacy and reliability of connections and transactions, as well as the provision of digital services. In a reality where everything is going digital day by day, the existence of robust, useful and scalable digital identity management systems is considered vital for electronic identification and awareness of the identity of the end

recipient of the communicated data. It is clear that because of this, each individual is in control of their data and therefore decides with whom to share it and for what purpose.

According to McKinsey (2019), "a digital identity standard is verified and certified with a high degree of assurance, unique, with individual consent, privacy of users and assurance of control over their personal data. This can promote inclusion, formalization and digitization. For example:

- 45% of women aged 15 and over in low-income countries lack identity, which is only 30% of men.
- In addition, 1.7 billion people will be able to access financial services.
- 90% of customer onboarding costs could potentially be reduced.
- The economic value of a country's GDP (Gross Domestic Product) could be between 3%-13% in 2030 due to digital identity."

It is commonly accepted that in the socio-political field, economic exclusion is reduced. It is a fact that in developing countries, less than 50% of the population has a bank account. Thanks to electronic identification, both women and children gain access to socio-political and economic processes, but more importantly, they are provided with the possibility to exercise their political rights (Dahan and Hanmer, 2015). Moreover, it should not be forgotten that "unlike the male population, the female population, in most cases, does not have access to personal identification" (Dahan and Hanmer, 2015). The contribution of e-identification to the management of the migration issue is considered particularly important, due to its fulfillment of the exercise of asylum and protection rights while being hosted in a foreign country or while in transit (Manby, 2016).

## 3.1. Identification methods

Identification is defined by John Hartley (2011) as "a process involving the assertion of the characteristics of one's identity in order to make sense of the self". The main means of identification for all nations is the Identity Card. This is a necessary identification document, based on an individual's characteristics, issued by a state or police authority, which has a unique identifier and the information it contains is kept in a state database.

In general, Credential Technologies are used to identify a person, which can be divided into: biometric information, cards and mobile phones (World Bank, 2018b).

The term biometric information includes the identification of an individual's unique physical characteristics to identify and validate their identity (Das, 2016). Biometric information is divided into primary (face, fingerprint, etc.) and secondary (e.g. signature) (World Bank, 2018b). Undeniably, biometric information is a universal yet unique and indivisible element (Das, 2016).

As far as cards are concerned, they fall into three categories: simple - or non-electronic, digital and smart. Firstly, non-electronic cards, apart from their unique identifier, contain the holder's basic demographic information, such as full name and date of birth. They are also likely to carry a photograph. This category includes the above-mentioned “Identity Card”. Next, digital cards (RFID) make use of radio frequencies to recognize and identify the information stored on the RFID tag on the surface of each card (World Bank, 2018b). Its use, in particular, is limited to processes such as: opening a door with a lock with a corresponding technology, product tracking, and contactless credit and debit card payments. It then goes on to talk about smart cards, which take their name from the technologies they are equipped with. They incorporate a microchip and a processing unit, designed in such a way that they are activated when they come into contact with a reader. They are already widely used in many countries, particularly for voting, bank account opening and driver's license application processes (World Bank, 2018b).

The next and most recently developed identification technology is mobile phone identification (Figure 8). This technology includes a number of methods, some of which are the One Time Password (OTP) method, where a unique code with a specific activation period is sent to the user's device, and the Authenticator Mobile App method, which combines the logic of the OTP method with the use of a secret key to generate a six or eight digit code, with an extremely short activation period, necessary to identify the user.


*Figure 8 - Mobile phone identification methods (World Bank, 2018b)*

## 3.2. Identity Management Systems

An identity management system is composed of a set of services aimed at managing an individual's identity information; it is structured in three levels, each of which is responsible for adapting the data processing rules and giving data holders access to the system. Nominally, these levels are as follows: Base, Lifecycle and Access and Use.

The Base of an identity management system is the layer that defines the rules for accessing the data in the system and is composed of the following parts (Pato and Rouault, 2003):

- Authentication Provider: Is the entity responsible for the initial authentication of a person wishing to connect to an identity. Initial authentication techniques include mechanisms such as password verification, smart card verification, biometric data scans, etc.
- Policy Control: The processing of information associated with an identity is regulated by a set of rules that control the management of the data held in the system and under what conditions it can be shared.
- Auditing: Auditing methods are a mechanism for recording the creation, modification and use of data.

The Lifecycle is the series of actions responsible for setting up the elements involved in issuing electronic identities and managing the data they contain. Its characteristics are as follows:

- Provisioning: This is the automation of processes that allow the coordination of the lifecycle of an identity (e.g. its creation or termination).
- Longevity: The maintenance of the record of modifications to an identity throughout its lifecycle.

Finally, the Access and Usage layer defines the ways of accessing data in the system and incorporates the:

- Single Sign-in: The identity card holder gains access to all the services interconnected with the system, as his/her identity is authenticated all at once during his/her first login to the system and no further actions are required.
- Personalization: The provision of information about the applications used by the identity holder.
- Access Management: Separation of access levels based on already defined privileges and rules.

For the most part, identity management systems are centralized, with the result that identity data is controlled entirely by one or more organizations rather than by the identity holders.

The first attempt to develop a fully decentralized identification system was presented with the creation of “Namecoin”. Namecoin went against the prevailing thinking at the time, which was influenced by an article published by Bryce Wilcox-O'Hearn in 2001 on namespace in computer systems. According to that paper, it was practically impossible to design a system for secure, distributed and human-readable credential selection. This statement is still known as “Zooko's Triangle” (Figure 9).

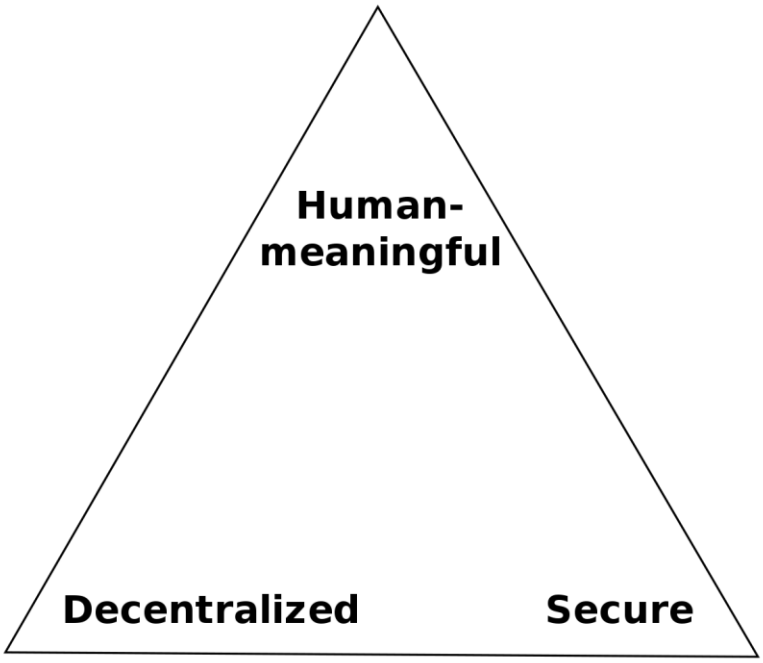


*Figure 9 - The Zooko's triangle (Wikipedia, 2020)*

Blockchain technology enables the selection, in a distributed manner, of a human-readable credential, as well as the matching and verification of name-value pairs without the intervention of intermediaries.

## 3.3. Security and Privacy

Due to the drive to ensure data protection, portability and interoperability, digital identities and the corresponding management systems are evolving and increasingly tending towards more decentralized approaches, rather than the fully centralized ones that currently characterize them. As new technologies are developed, regulators gain a better understanding of the digital world, governments and private organizations discover better ways to interact digitally, and users gain more confidence in the existing digital identity management systems that continue to be proposed and adopted.

In the area of identity security, this is achieved through the stages of the identification of the individual with his or her real, accurate data. Firstly, the original message is encrypted to convert it into a form that is incomprehensible to humans, which requires decryption in order to be read. According to IBM (2019), encryption is the process of converting a simple, readable text (plain text) into an unreadable form called ciphertext. Cryptography is used to achieve identification, authentication and

authorization. Identification is reduced to the assertion that a person is who he or she claims to be. Authentication refers to proving that the person is indeed who he or she claims to be. Authorization refers to the access that the individual gains to specific resources as a result of the authentication that has taken place. Then, using the "Message Digest" method, the ciphertext is converted into a numerical representation which is immediately hashed and re-encrypted, thus creating a digital signature. Finally, there is the "Public Key Infrastructure" (PKI) system that relates to facilities, policies and services that support the use of public key encryption to authenticate the participants of a transaction (IBM, 2019a) and the digital certificates that guarantee the correct association of a specific public key with an entity.

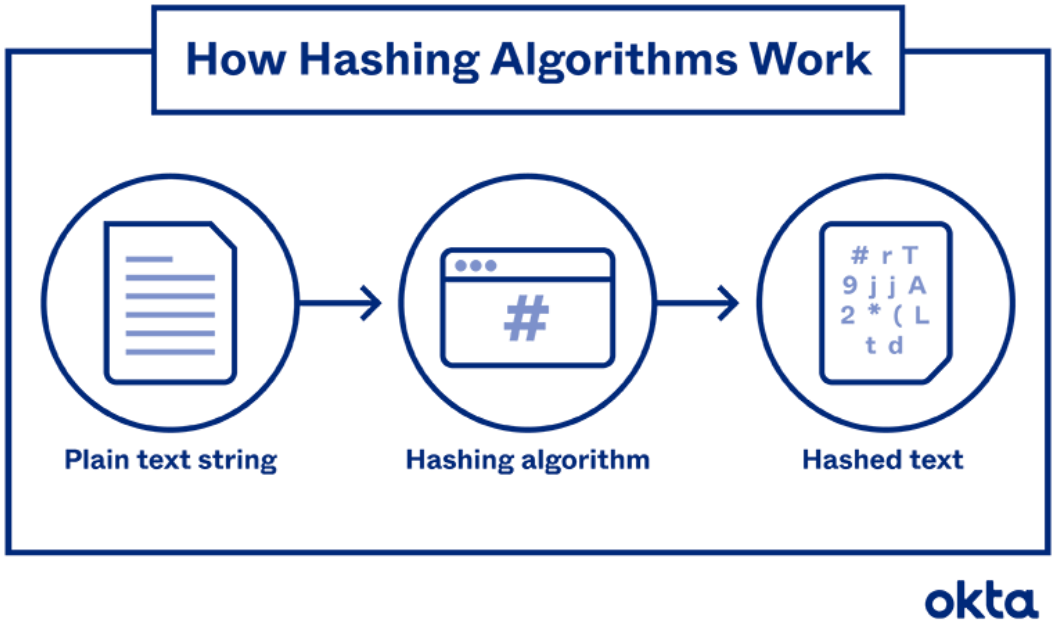


*Figure 10 - Mode of operation of the hash algorithm (Okta, n.d.)*

The concept of privacy is different from the specification of the existence of security in the system. It is defined as the ability of an individual or group to isolate information about themselves and thereby control the boundaries within which a third party can interact with it. There are the following areas of privacy: informational privacy, which refers to public documents and records; physical privacy, which refers to the integrity of the human body; privacy of telecommunications; and privacy of domestic and family sanctuary (The Public Voice, n.d.).

In electronic identification systems, the organization, whether public or private, must develop a decent policy framework regarding the instilling of trust in the individuals involved, the storage of their data and their protection from threats, as well

as the ways of managing them (World Bank, 2018b). Without these privacy policies, a sense of trust in the system will not be established.

Due to the aforementioned, on 27 April 2016, the European Commission announced the "General Data Protection Regulation" (GDPR), which refers to the new rights that citizens of the European Union have regarding their personal data, one of the most important of which is the right to withdraw consent. This was prompted by the continued use of individuals' personal data by companies for the ultimate purpose of providing services, particularly online, such as targeted advertising services. The main concern of the European Union with this regulation is to provide individuals with control over their personal data and to simplify the regulatory environment for international businesses by consolidating the regulation within the European Union (Pandit, O'Sullivan and Lewis, 2018).

For historical and non-historical reasons, the experience of using digital identity today is ambiguous, with poor standards or interoperability. It is also surrounded by insecurity, as indicated by the almost daily reports of attacks and data breaches (Lyons et al., 2016). Despite the efforts of institutions and governments over the years, threats to user data have not ceased to exist. A typical example is the case involving the consulting firm "Cambridge Analytica" and the social network "Facebook", where the latter gave unlimited access to personal data of more than 87 million of its users without their consent. Some other examples of threats - breaches are: the "Equifax" data breach case in 2017 (143 million credentials compromised), the "Adult Friend Finder" case in 2016 (413 million accounts stolen) and the "Anthem" case in 2015 (78 million accounts compromised). No proactive approach appears to be 100% secure, but early detection of the problem could prevent these accounts from being abused.

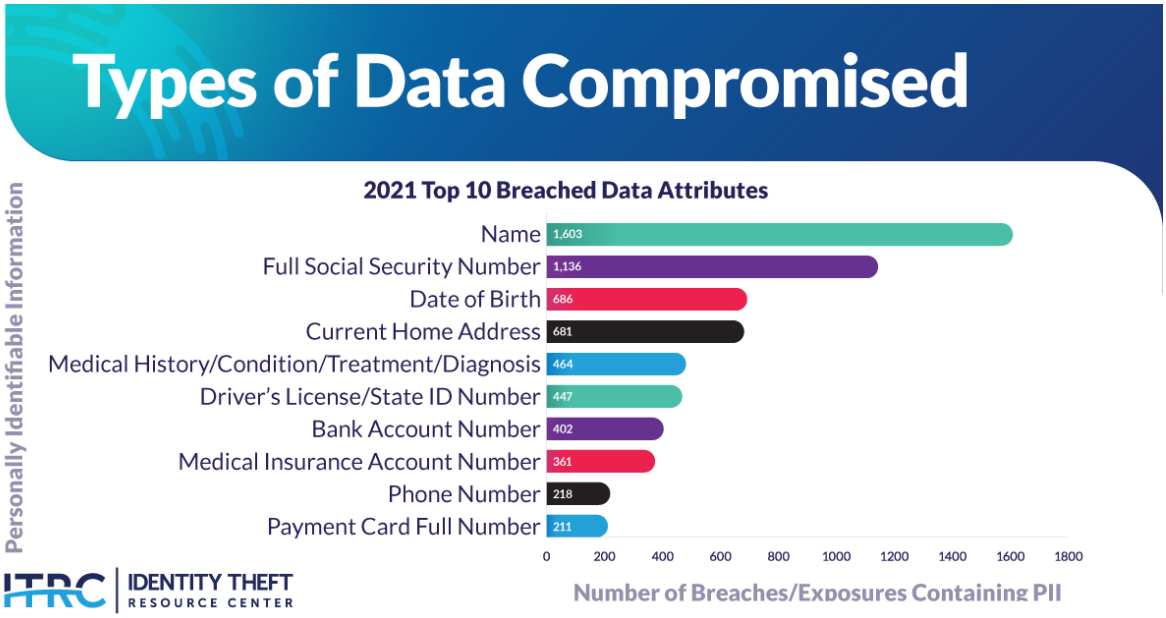


*Figure 11 - Graph from the Identity Theft Resource Center's report on data breaches in 2021*

## 3.4. The use of Blockchain technology in identification

Blockchain technology can be seen as a mechanism for achieving integrity in distributed software systems and can replace the role currently played by the referred trusted entities (e.g. government agencies, banks). This is due to its specificity, i.e. not requiring the involvement of a third, centralized mechanism during communication between two entities. In a Blockchain authentication system, the data is stored within the chain thus offering absolute security. Each user (hereinafter "holder") creates a digital wallet, represented by an address on the Blockchain. An organization called an "Issuer" performs the necessary actions to register a new digital identity, the information of which is stored in the holder's wallet. A third entity, the "Verifier", takes part in the process and interacts with the holder, requesting information about his/her identity through transactions. Subsequently, the holder chooses which identity data he or she wishes to share, so that the Verifier can then confirm its authenticity. Throughout the process, trust is maintained between the parties to each transaction, due to the public and private key encryption and the transparency of how identities are created and verified.

However, it is commonplace that all technologies, including Blockchain, have unfavorable characteristics. One of the main ones is the lack of possibility to recover the login codes or the private key of a wallet in case of their loss, due to the absence of a central server. The level of complexity of the technology is also considered a major

drawback, since the structure of the system does not ensure that a holder will own only one private key, which results in the possibility of creating multiple identities for the same person.

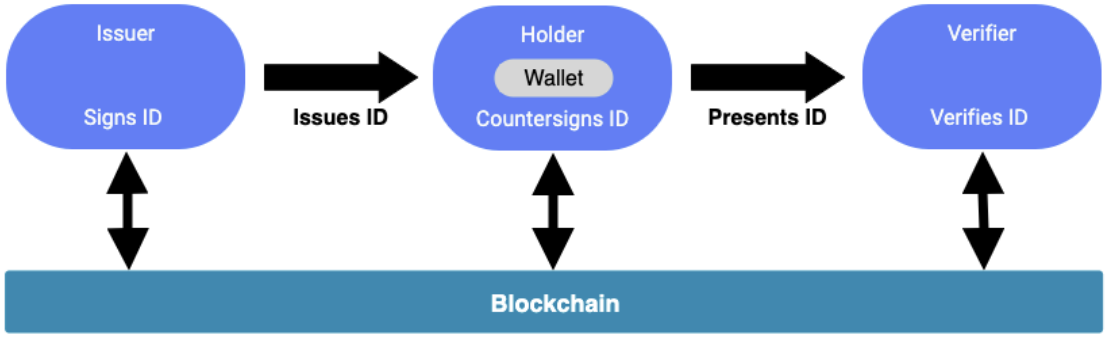


*Figure 12 - How a Blockchain-based authentication system works*

# 4. Presentation of the identification application “IDEN”

The application “IDEN” (Identification Eden) is an identification system that makes use of Blockchain technology. It was developed on the Ethereum Blockchain and takes advantage of any features implied with it. It involves three roles, “Issuer”, “Citizen” and “Verifier” and takes advantage of smart contracts to perform its functions. The Issuer is responsible for registering the identity details of the Citizen and the Verifier on the Blockchain. The data is entered via a form in the issuer's application interface and then stored in the wallet corresponding to the address provided by the citizen or verifier. Then the citizen can, from the corresponding interface, check his/her data and reply to incoming requests to share his/her data. Finally, the verifier has the ability, through its own interface, to request a citizen's data, confirm its ownership and perform the desired actions. At the security level, it inherits all the traits of smart contracts, so all data is encrypted, with continuous checks to avoid possible errors.

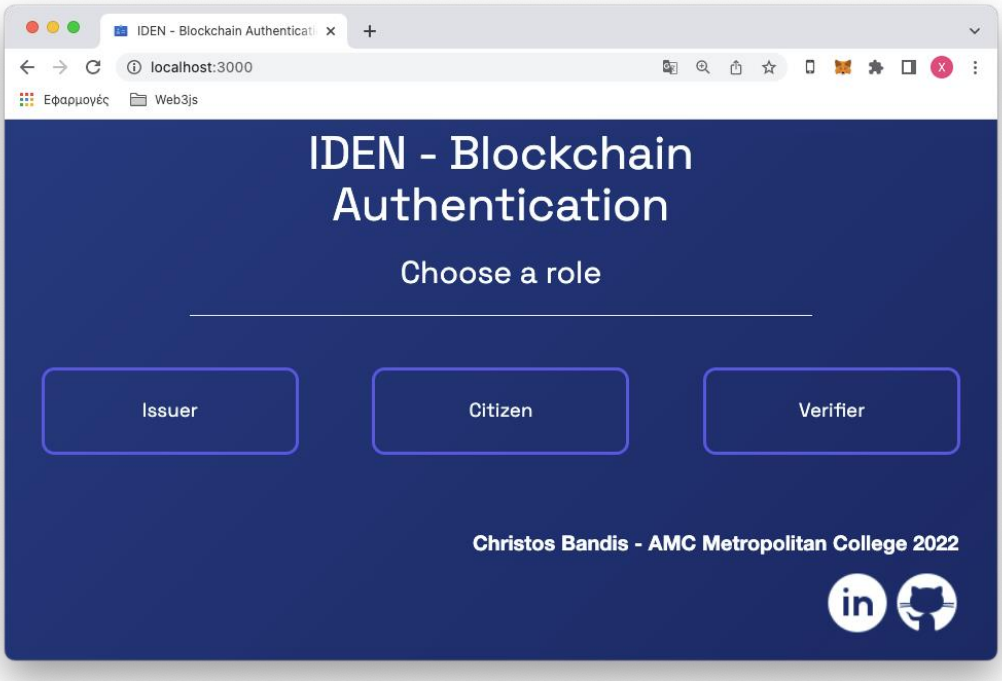


*Figure 13 - The home page of the “IDEN” application*

## 4.1. Existing problem

To date, the use of digital identity has facilitated banking and citizen-to-state transactions, as more than 1.5 billion people around the world do not possess civilian identities (The World Bank Group, GSMA and Secure Identity Alliance, 2016). However, the major drawback of this development, without it being noticeable to the

majority of people, is that the identity and its data are not managed by the holder; the result of which is the exposure of parts of this identity to for-profit organizations. Legislation such as the "General Data Protection Regulation" was confronted with such problems of inappropriate management of personal data. Nevertheless, the "explosion" of the internet, the increase in the use of mobile devices (e.g., smartphones, tablets) and the obsession of organizations to collect data have exacerbated the situation of data leakage issues.

As mentioned in the previous chapters, in recent years, Blockchain technology, despite the fact that it is not a new venture in the field of technology, is constantly developing and laying the foundations for a new era in both the financial and socio-political sector. The changes that this technology is going to bring about are so radical that a multitude of scientists in the industry - and outside of it - are calling it "the new internet".

The use of Blockchain technology for the verification of digital identity data is an important development as it protects sensitive personal data from unauthorized leakage, allowing users to be the sole owners of their data. It also provides a solution to the problem found mainly in simple, physical identities. This problem relates to cases where one element of a person's identity (e.g. surname) needs to be verified, but the other elements (e.g. date of birth) remain visible to the verifier without the owner's permission. For example, a prospective employee can share with the company's recruiter his grades at the university he attended, without disclosing his nationality.

Based on the aforementioned, the main problem that the application is required to address is the improper management of personal data by organizations that have gained access to it and the ways in which Blockchain technology can benefit in maintaining its integrity and security. To solve it more properly, it is important to break it down into smaller sub-problems:

- Finding the best method of self-management of data.
- Research in the field of Blockchain to create the optimal infrastructure for implementation.
- Exclusive control of the details of an identity by its owner.
- Immediate and valid confirmation of the ownership of an identity directly from the verifier, without the intervention of a third party.

As a result of the above analysis of the problem, the following specification requirements that the application must meet are divided into functional and non-functional requirements.

The operational requirements include the following:

- Control the access rights of the user attempting to interact with the application.
- Control of the wallet address provided by the holder when registering on the Blockchain.
- Registration of the holder's identity data (citizen or verifier) in the storage of the corresponding smart contract.
- Creation of a request for the disclosure of identity data by the verifier and consent by the addressed citizen.
- Informing the citizen when a request awaits consent.
- Verifying the details of an identity via Blockchain by comparing the hash of the provided data with the hash corresponding to the issuer's smart contract and checking the wallet address of the identity issuer when requested.
- Add/Create a medical history record for the citizen.
- Keeping identity and medical history in the citizen's wallet to enhance privacy.
- Ability to create and read the QR code extracted from the holder's wallet address.

The following are classified as non-operational requirements:

- Full immediacy and availability of the application.
- High data encryption, provided by Blockchain and smart contracts.
- Support for a wide range of devices (e.g. computers, smartphones).
- Simultaneous service for multiple users.
- Friendly graphical interface for users, regardless of experience.
- Fast transactions (based on the rate of Ethereum Blockchain usage at a given time).
- Low transaction costs, sending only the crucial data via transactions (also depending on the usage rate of the Ethereum Blockchain at a given time).
- Proper management of data within smart contracts, with the aim of speeding up access and avoiding errors.

## 4.2. Design and modelling

The following diagrams show the full structure of the system and the most efficient ways of exploiting Blockchain technology for the development of the identification application. Next, the actions that each role in the application can perform are shown, as well as key parts of the application's appearance. The "Unified Modeling Language" (UML) was used to create the diagrams.

At the code level, the classes created correspond to each role individually. There are also two more classes, one for notification requests and one for medical history. Each class, as distinguished in the "Class Diagram" in Figure 14, contains the variables and methods necessary for the application to function. Also shown is the connection and the relationship between the classes, as well as their cardinality ratio.

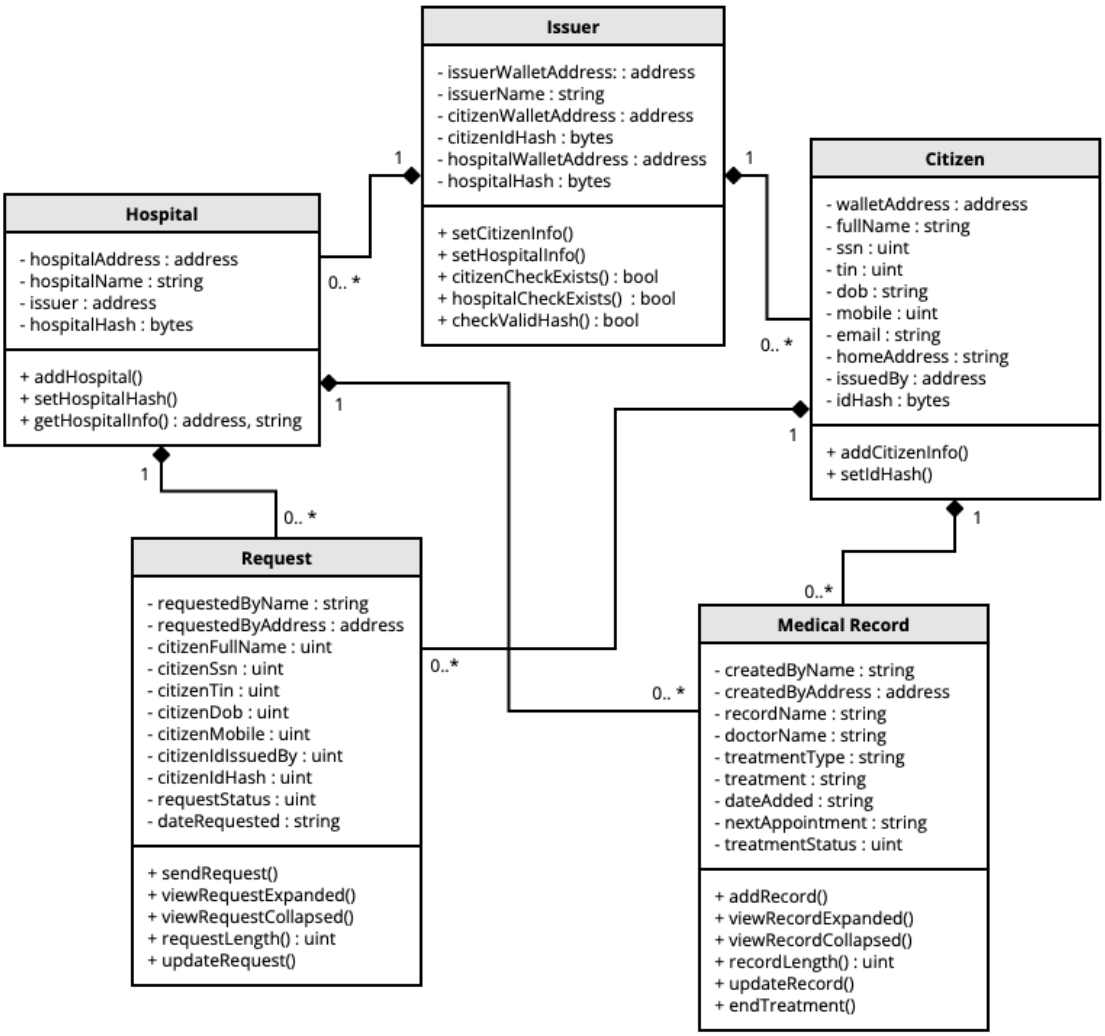


*Figure 14 - Class Diagram*

A feature of the Solidity programming language, used to write the smart contracts, is that no “getter” (a method that returns the value of a variable) is required to be created during programming, as it is created automatically. Similarly, it was deemed unnecessary to create a “setter” (a method that sets or updates the value of a variable), with a few exceptions, because most variables are updated via appropriate methods. The above observations can be seen in the diagram in Figure 14 and in the Object / Instance Diagram, in Figure 15, which shows diagrammatically, the execution of a complete use case of the application with real data (from the registration of a user, to the addition/creation of a medical history for him/her).

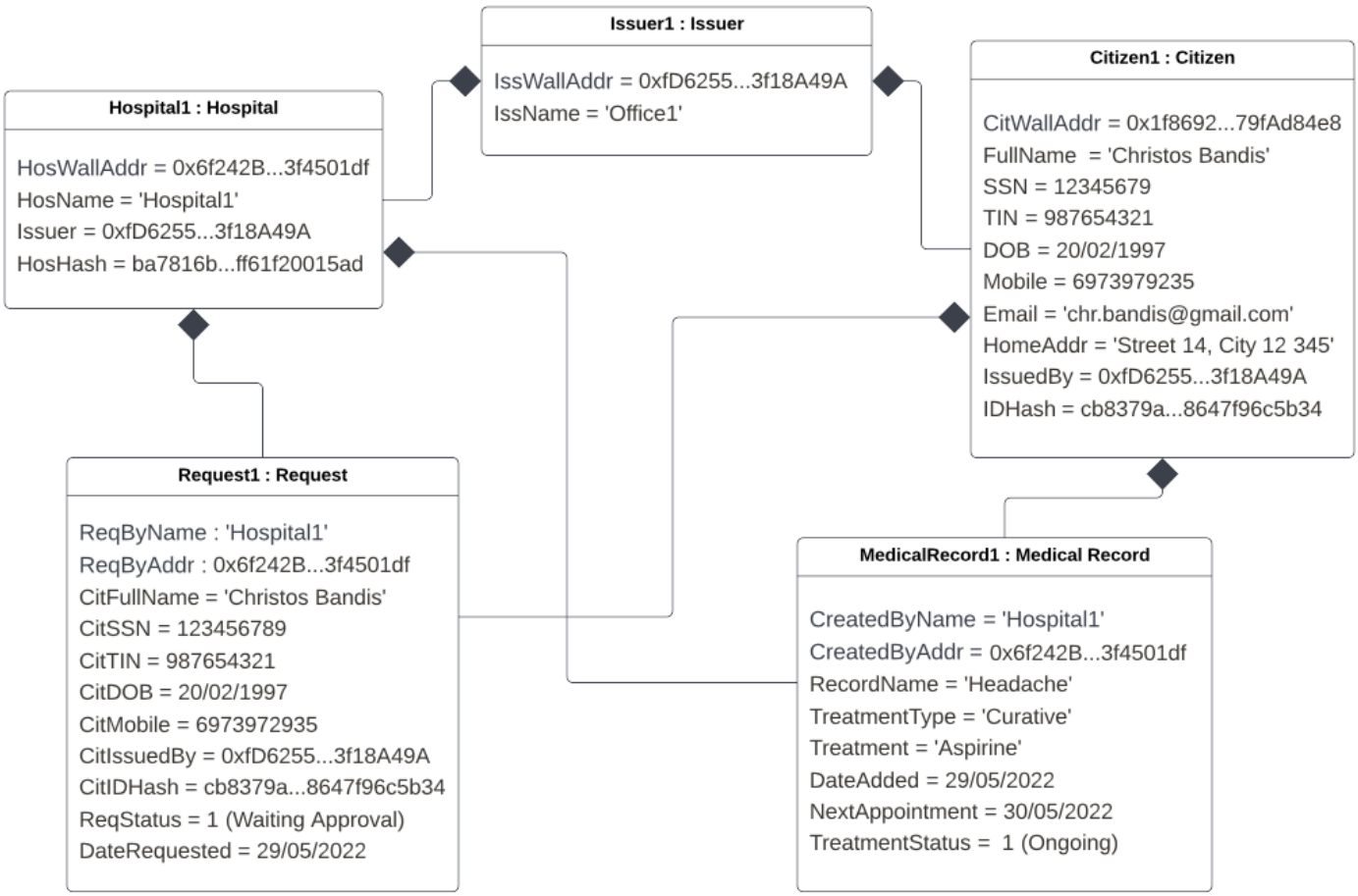


*Figure 15 - Object / Instance Diagram*

The operation of the application is characterized by the simplicity of the structure of the sub-functions that contribute to it. As can be seen in the following Flowcharts (Figures 16, 17, 18), the steps of the procedures followed in the execution of the application, depending on the use case, have been drawn up in such a way as to benefit the correct flow of data throughout its operation. Another feature of Flow Charts

is that they visualize the methods of execution of the code, showing the ways in which it has been organized. The aforementioned can also be observed through an Activity Diagram. As an example, Figure 19 illustrates a diagram that represents the set of processes and steps performed in the application by the other roles and includes the process of requesting data disclosure, confirming ownership and adding/generating medical history.

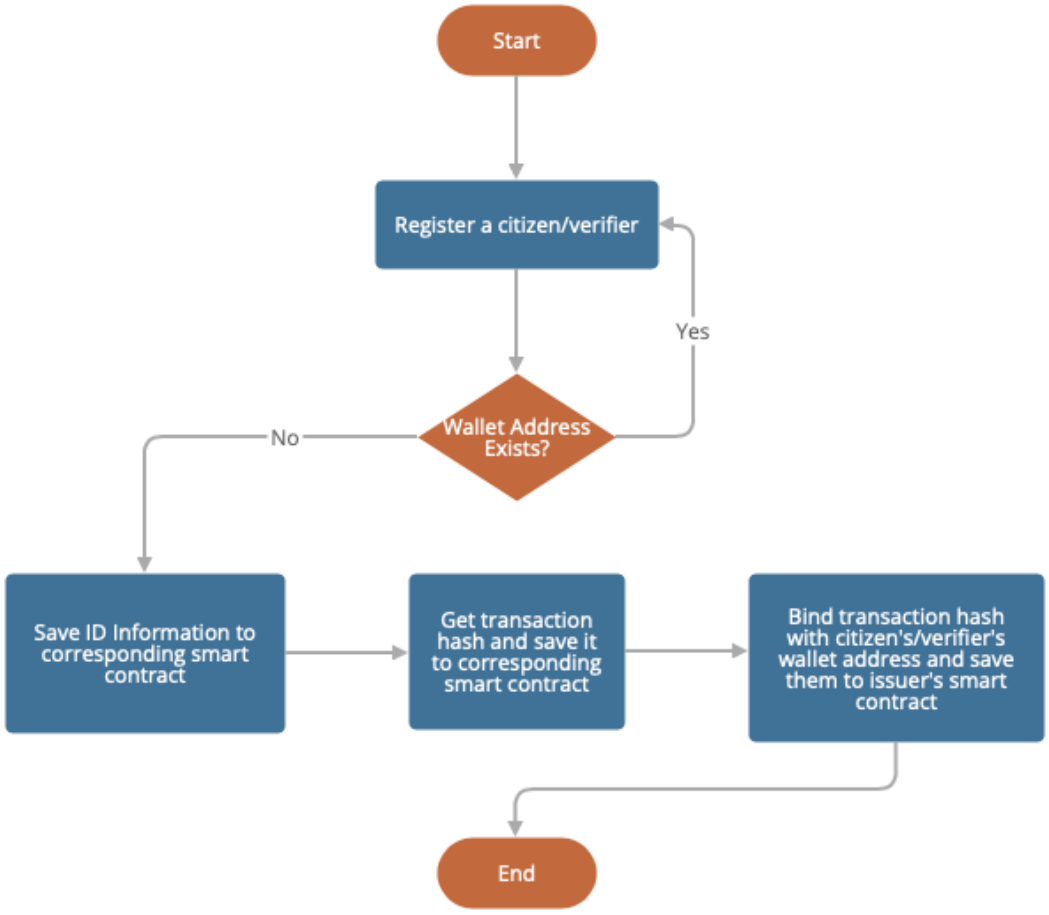


*Figure 16 - Issuer Flowchart*

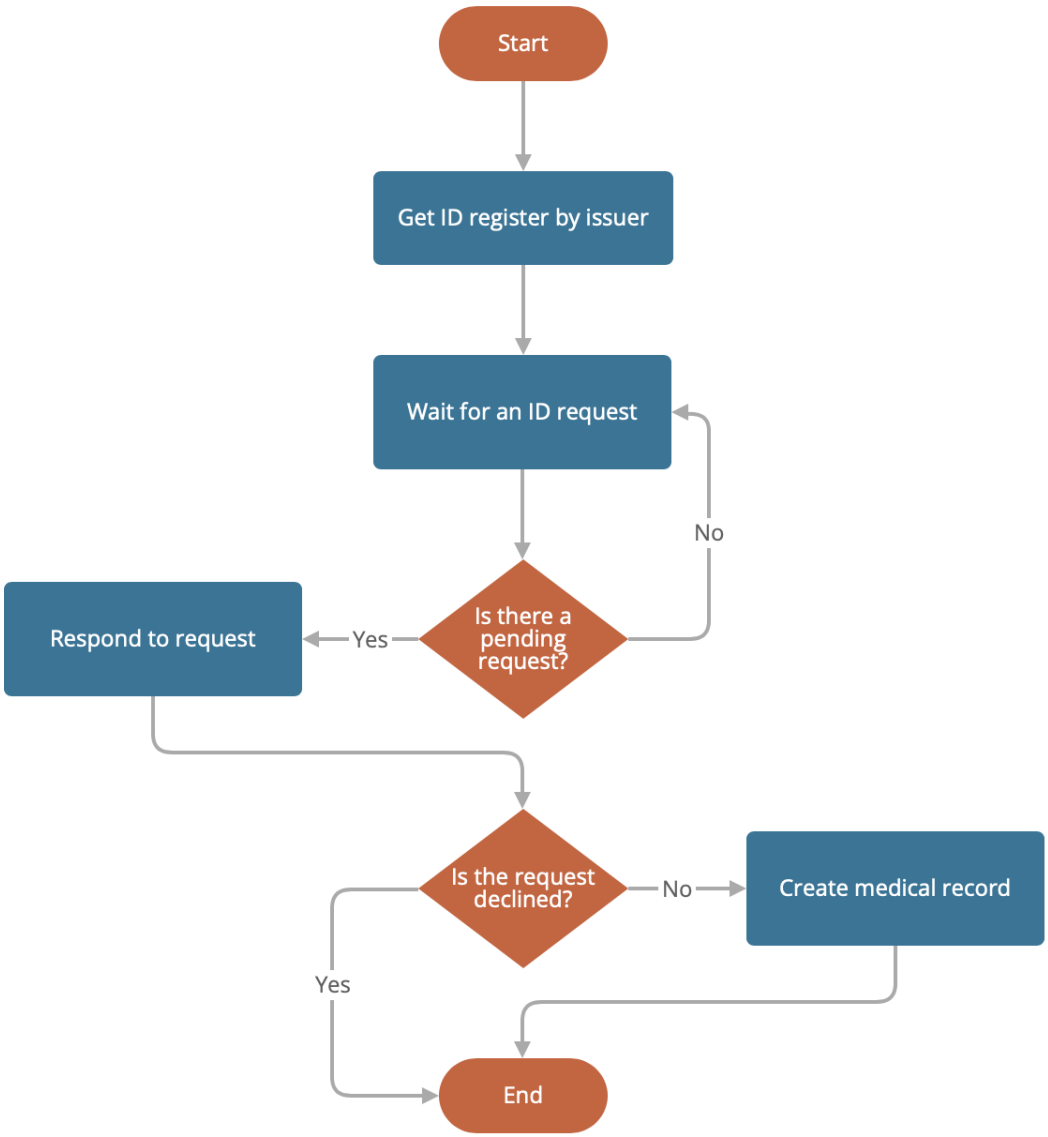


*Figure 17 - Citizen Flowchart*

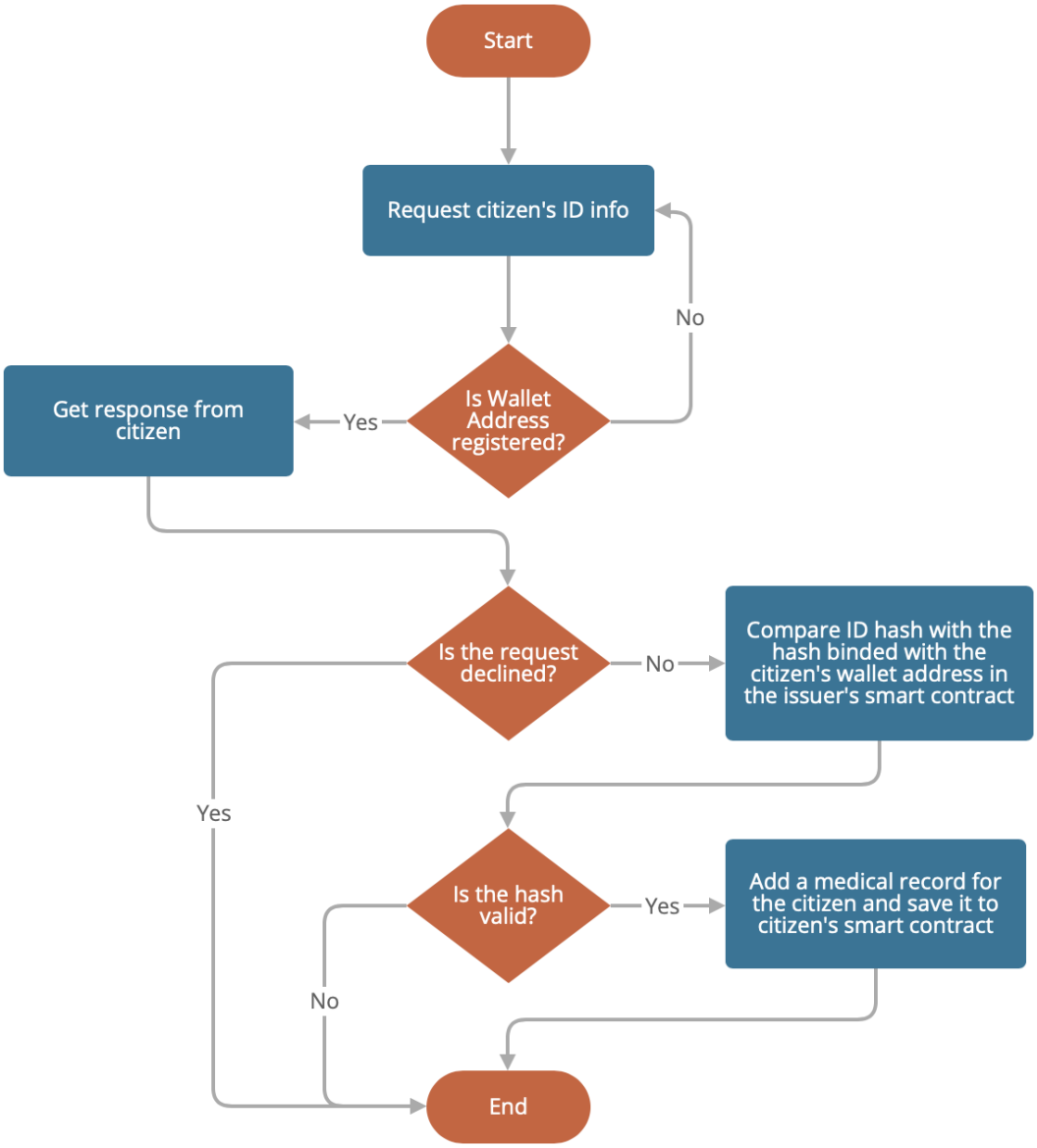


*Figure 18 - Verifier Flowchart*

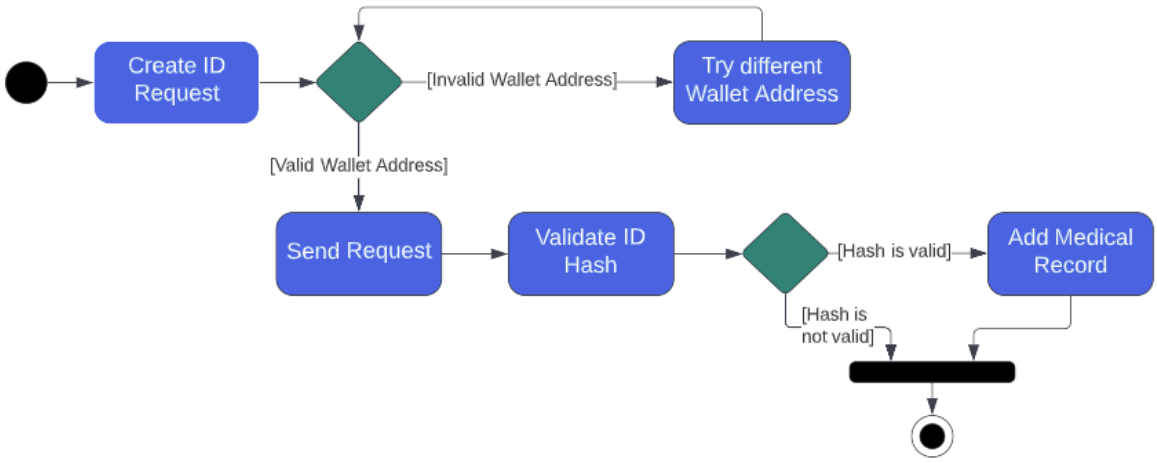


*Figure 19 -Verifier Activity Diagram*

The presentation of the application model includes the Sequence Diagram (Figure 20) which shows the way and the order in which a role, in this case the Issuer, interacts with the basic functions of the application and its background (wallet, Blockchain network) in the long term. Similarly to the Issuer, the other two roles also interact with the application's core functions, hence they are represented by the diagram below.

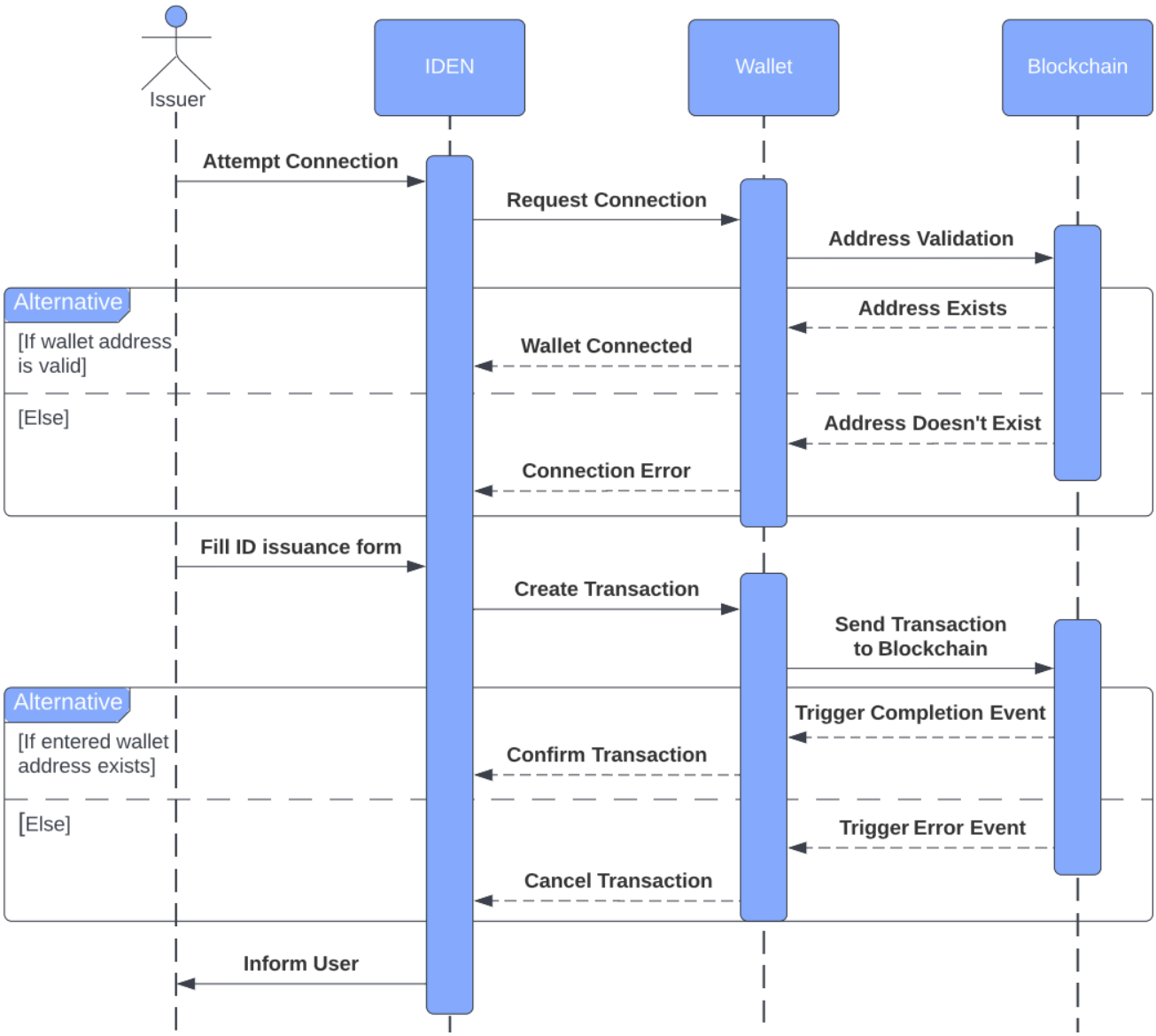


*Figure 20 - Issuer Sequence Diagram*

Regarding the design and development of the user interface, particular emphasis was placed on the usability of the application rather than its appearance, as it is aimed at users of all experience levels. As can be seen in the following representative figures (Figure 21, 22, 23, 24), the options available in each case are clearly visible, with clear

steps to complete the desired action and, in case of an error or omission of one or more steps, the user is informed by a message that prevents the continuation of the operation. It is worth noting that, the “Issuer” “Citizen” “Verifier” menus would have been three different interfaces of the application, but for ease of development they were included in one.

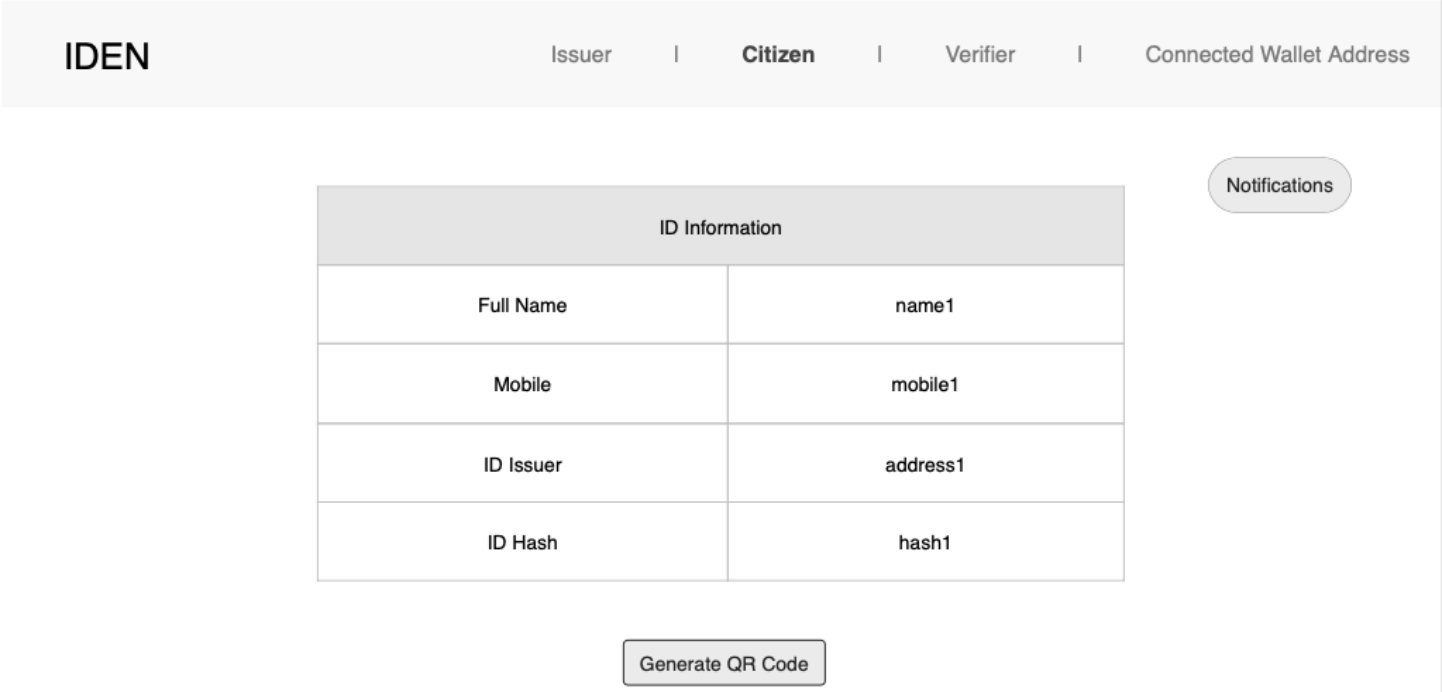


*Figure 21 - Citizen ID Page Mockcup*

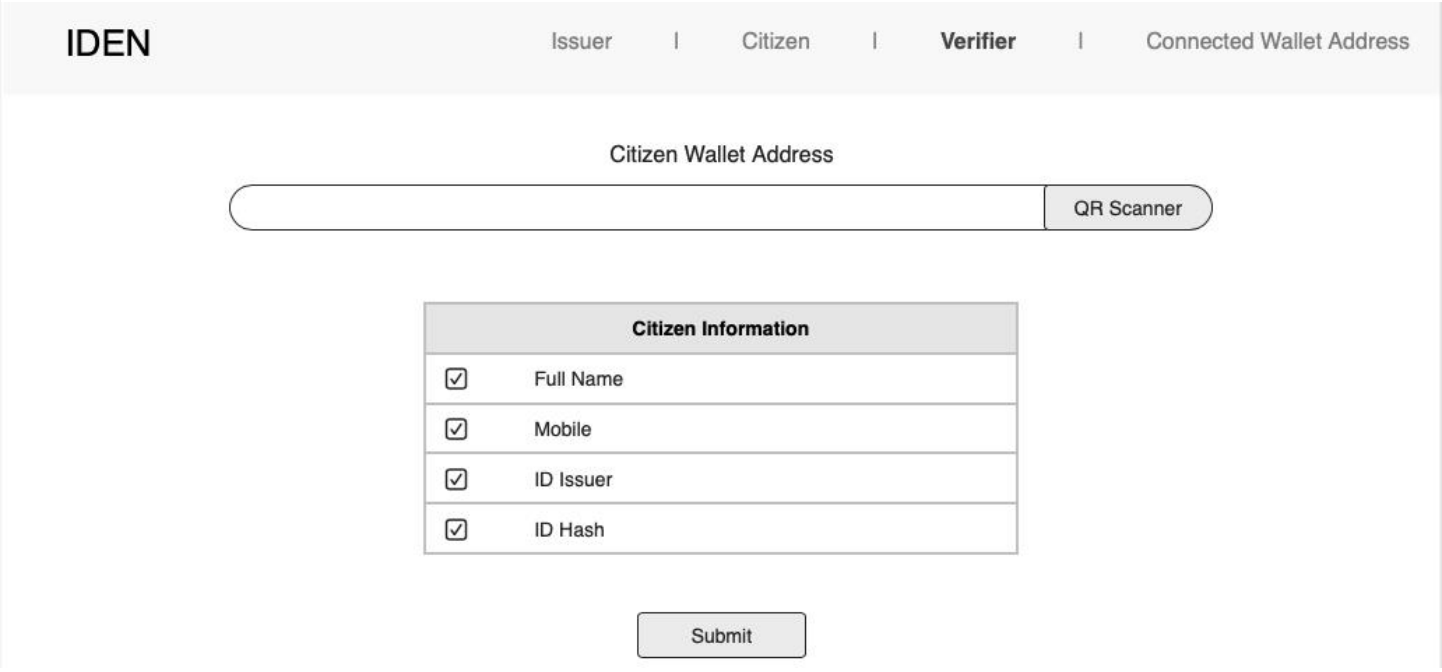


*Figure 22 - ID Request Page Mockup*

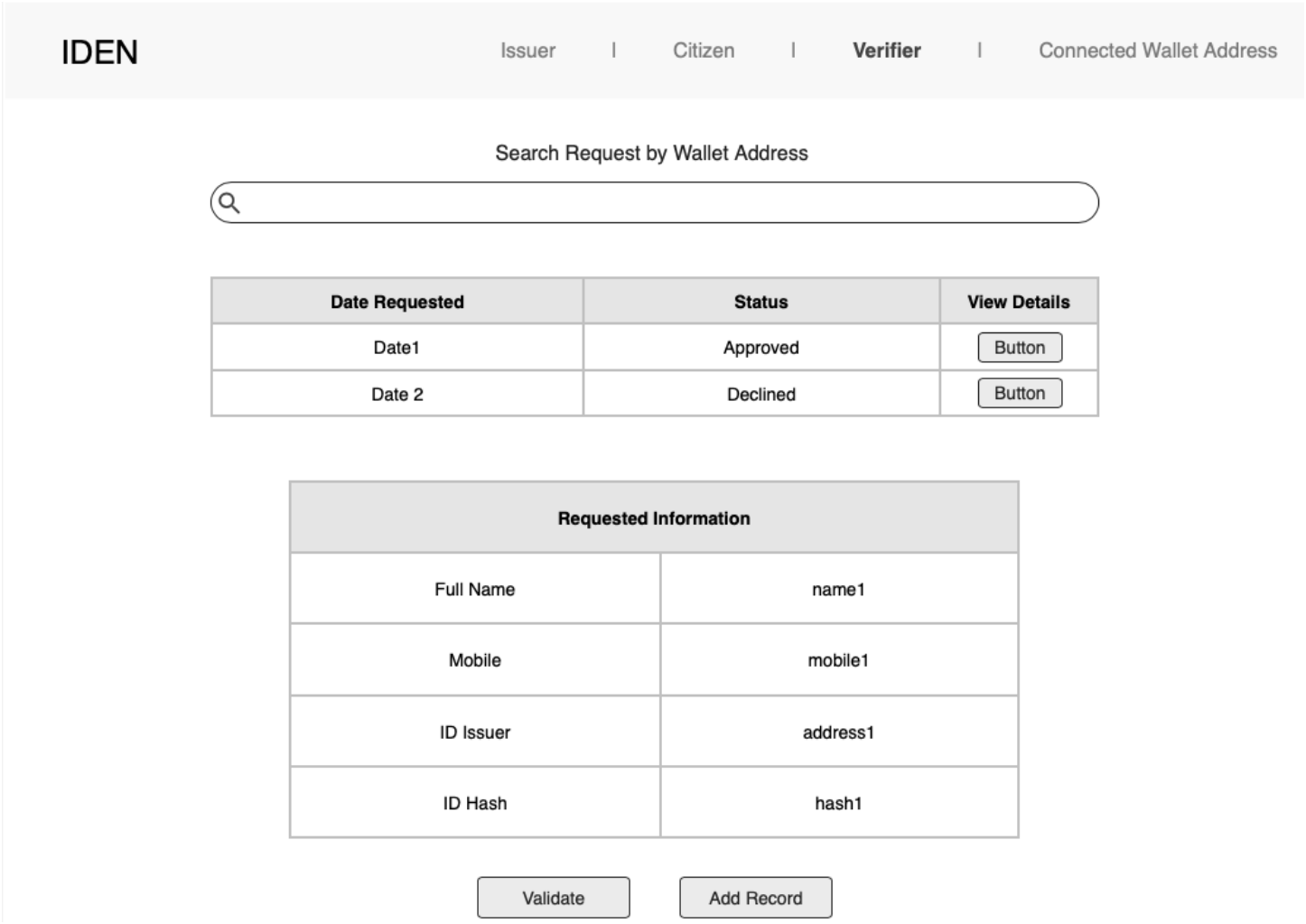


*Figure 23 - Verifier ID Request Management Page Mockup*

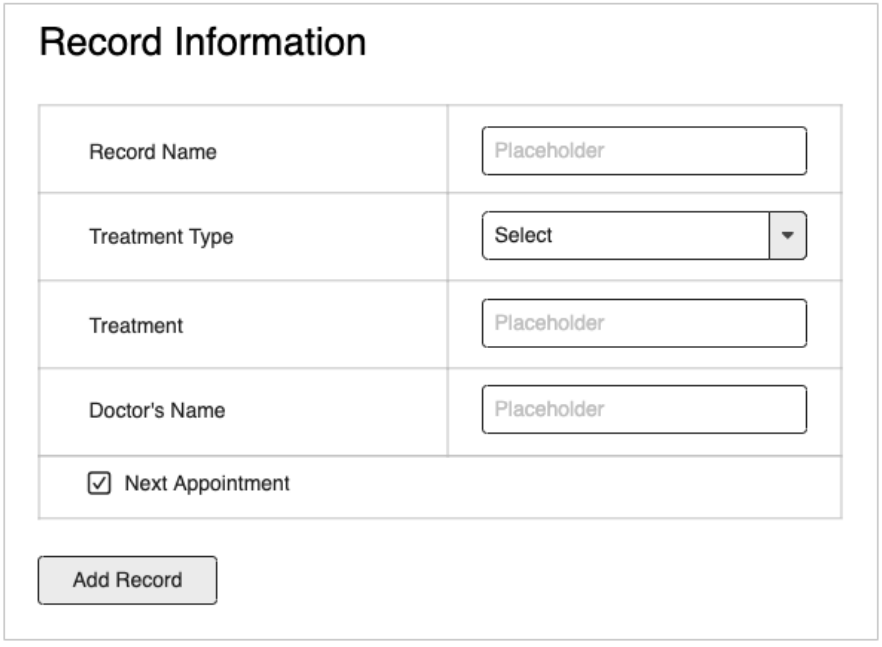


*Figure 24 - Create medical record Window Mockup*

All the above also apply to the other use cases of the application, such as the registration of a user and the citizen's response to a data sharing request. It is worth mentioning the adaptation of the application's appearance to all types of devices, with a typical example being Figure 25, where the wireframe of the use case of the citizen's login to the application and the display of his data sharing requests and the history of his medical records is presented.

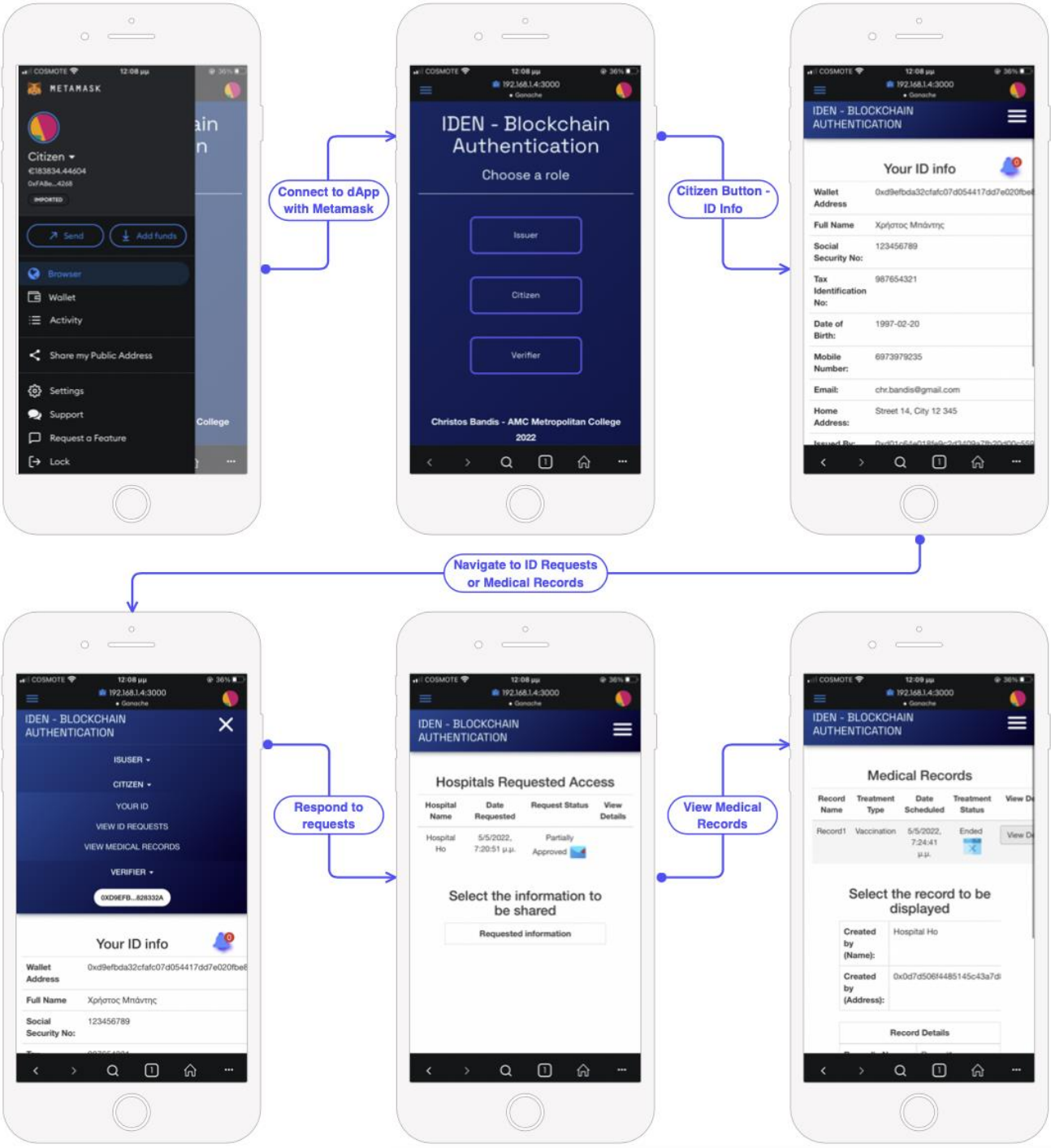


*Figure 25 - Citizen's use cases Wireframe (Mobile)*

## 4.3. Methodology

In order to enable the preparation of this thesis, a qualitative research design was applied, which involves the study and comparison of published papers, books and scientific articles related to the field of Blockchain and more specifically Blockchain Authentication. Then, taking advantage of the values and principles of the Agile methodology, the process of gradual development of the application was put in place, which was characterized by a series of perpetual checks and tests. This methodology was chosen for the flexibility and continuous evolution that it brings to decision-making regarding the future development of the application and its functionalities.

To address the complex problems encountered during the application development process, the Scrum management framework was used, which describes a set of tools and roles whose goal is to facilitate the structuring and management of projects that benefit from the Agile methodology (AppDividend, 2022). The diagrams, reports and tables that follow were created using the "Visual Paradigm" application.


*Figure 26 - Scrum management tool (Scrum Process Canvas)*

### 4.3.1. Reports

# Product Owner Report

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

*Table 1 - Project owner report*

# Scrum Master Report

| Member | Christos Bandis |
|---|---|
| Responsibilities | • Facilitates the creation of epics.<br>• Coordinates the creation of the traffic plan.<br>• Helps maintain the deterrent log.<br>• It ensures that issues affecting development are discovered and resolved. |

*Table 2 - Scrum master report*

# Scrum Team Report

| Member | Christos Bandis |
|---|---|
| Responsibilities | • Commitment that user stories will be done within a sprint (recurring fixed time frame).<br>• Identify risks and implement mitigation actions to reduce potential errors.<br>• Full development of the product or service. |

| | |
|---|---|
| | • Writing the code of smart contracts.<br>• Design and development of the user interface of the application.<br>• Continuous testing of the functions of the application. |

*Table 3 - Scrum team report*

# Project Charter

A project charter is a formal, usually short document that describes the project in its entirety (Wrike, 2019).

## 1. Project Vision

The vision of the project is to create an identification application through Blockchain Technology. Unlike existing identity management systems, the application will provide secure data encryption and full management of identity details solely by the identity owner. It is addressed to all users who wish to exercise their right to privacy and do not want their data to be under the control of public or private organizations.

## 2. Project Mission

Through Blockchain technology, users of the app will be able to keep their personal data within their digital wallet on the Blockchain and will have full control over the data they share in cases where verification is requested by another entity. Nominally, some of the key objectives of the application are:

- Decentralization of identification methods.
- Encryption of data and transactions between citizens and organizations.
- Avoiding the use of personal data for advertising purposes.
- Ensuring user privacy.

## 3. Project Success Criteria

The success of the application will depend on the criteria of security and speed of transactions, initially in test periods and then when published on a public test

Blockchain. Also, a key criterion is the usability of the application by users of all levels of experience and the adaptability of its user interface to a multitude of devices.

# Use Case Report

## 1. Use Case Diagram

The following Use Case Diagram (Figure 27) presents a generic use case that includes all the functions of the application, from logging each user into the system and creating the identity of the citizen and the verifier, to sending a request to share identity data, confirming their ownership and adding/creation of a medical history by the verifier.

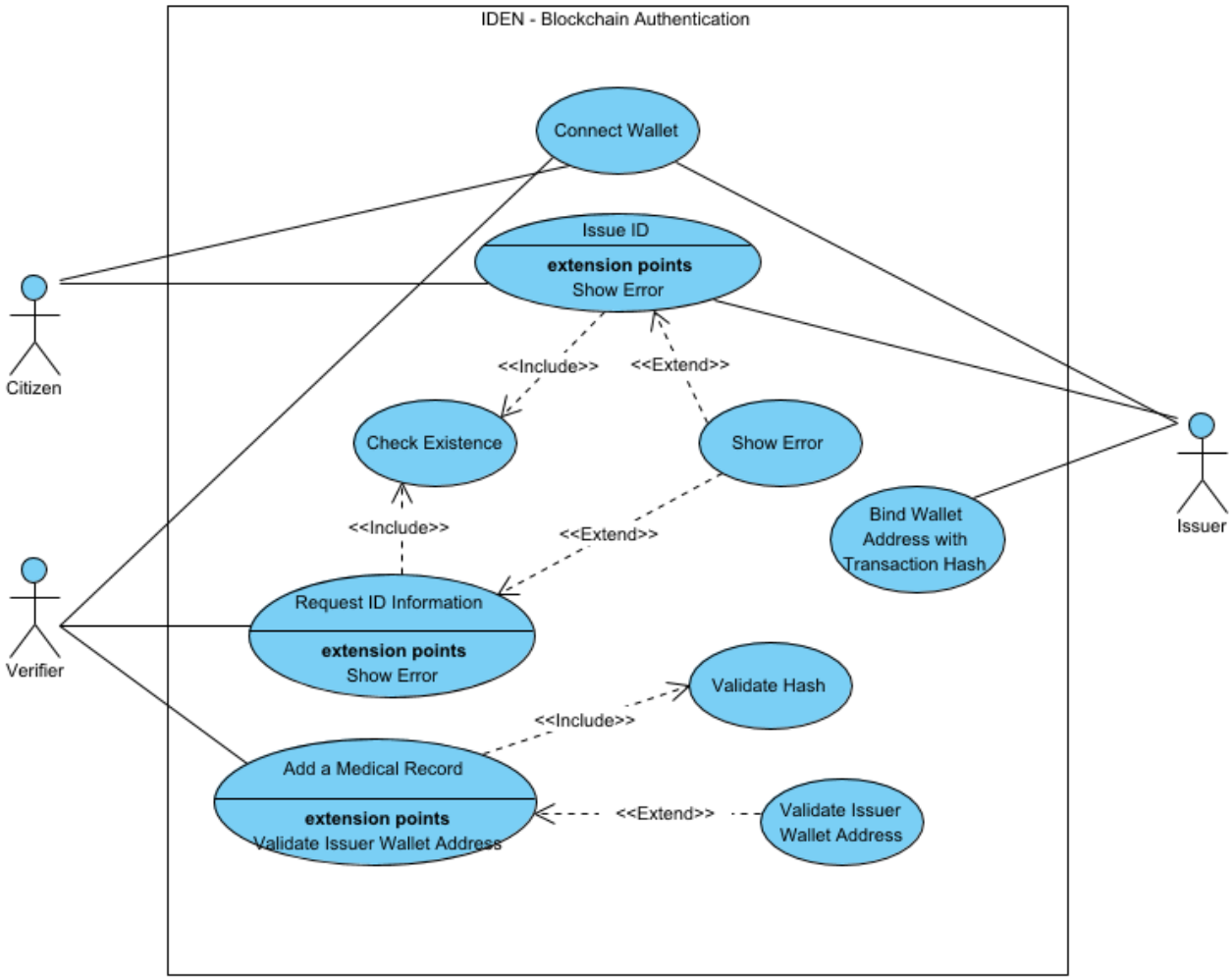


*Figure 27 - Use Case Diagram*

## 2. Prioritized Use Cases

Use Cases Prioritization is a method by which organizations can identify potential use cases, analyze the business value derived from them, and rank them based on their impact on their strategic objectives (Mishra, 2021). In the table below, the "Priority" column indicates the importance of the use cases in terms of delivering greater and more immediate business benefits, the "Size" column contains a subjective assessment of the effort required to support each case, and the "Complexity" column includes a subjective assessment of the relative difficulty in supporting each case.

| Name | Description | Priority | Size | Complexity |
|---|---|---|---|---|
| Validate Hash | Comparison and validation of the hash of the provided citizen's identity data with the hash matched to his/her address in the issuer's smart contract. | Must | Very Large | High |
| Validate Issuer Wallet Address | Verification of the issuer's address on the Blockchain in order to ensure the validity of the communicated data. | Should | Large | High |
| Check Existence | 1) The issuer checks whether the address provided by the citizen during registration exists on the Blockchain. 2) The verifier checks whether the address provided by the citizen at the request for identity disclosure exists on the Blockchain. | Must | Medium | Medium |
| Bind Wallet Address with Transaction Hash | The issuer matches the citizen's provided wallet address with the hash of the authentication transaction and stores the result in the storage of the smart contract. | Must | Medium | Medium |
| Add a Medical Record | Add/Create medical history for the citizen. | Should | Medium | Medium |
| Request ID Information | The verifier sends a request to the citizen to share the selected identity data. | Must | Small | Low |

| | | | | |
|---|---|---|---|---|
| Issue ID | Issuing an identity for the citizen or verifier. | Must | Small | Low |
| Connect Wallet | The user connects to the app via his/her digital wallet. | Must | Very Small | Low |
| Show Error | Display an error if the address provided by the citizen does not exist in the Blockchain. | Should | Very Small | Low |

*Table 4 - Use Case Prioritization*

## Epics Report

Epics are high-level functionality or generally defined requirements. They can be broken down into smaller pieces, called "User Stories". In the table below, the "Priority" column includes the level of importance of epics in terms of delivering greater and more immediate business benefits, while the "Risk" column represents the level of uncertainty around the successful completion of an epic.

| Name | Description | Parent Use Case | Priority | Risk |
|---|---|---|---|---|
| Add/Create medical history | Display of a transaction window to add/create medical history and store the data in the citizen's wallet | Add a Medical Record | Should | Low |
| Error display | Display an error if the address provided by the citizen does not exist in the Blockchain. | Show Error | Could | Low |
| Registration of identity data | Display a transaction window to verify the citizen or authenticator identity. | Issue ID | Must | High |
| Verification of the provided hash | Comparison of the hash of the citizen's provided identification data with the hash assigned to his/her address in the issuer's smart contract. | Validate Hash | Must | Medium |

| | | | | |
|---|---|---|---|---|
| Login to the application | Display a dialog box to connect the use of the wallet application. | Connect Wallet | Must | Low |
| Selection of desired items | After confirming the existence of the provided citizen's address, the verifier selects the details of the citizen's identity he/she wishes to check. | Request ID Information | Must | Medium |
| Send request | Display a transaction window to send the request to the citizen. | Request ID Information | Must | Medium |
| Verification of issuer's address | Compare the address of the identity issuer with a list of known addresses when requested by the request. | Validate Issuer Wallet Address | Should | Medium |
| Checking the existence of an address | Checking whether the citizen address provided during registration or notification request exists in the Blockchain. | Check Existence | Must | Low |
| Matching an address with the hash | Matching and storing the citizen's address with the hash of his/her identity in a "key-value" format. | Bind Wallet Address with Transaction Hash | Must | Low |

*Table 5 - Reference of "Epics"*

# Product Backlog

The "Product Backlog" is an evolving, ordered list containing everything needed to improve the product (Scrum.org, 2016).

## 1. User Story Map

User Story Mapping is a lean, design-based mapping method that describes the actions expected by the Scrum team that users perform in order to accomplish their goals in a digital product (Kaley, 2021).

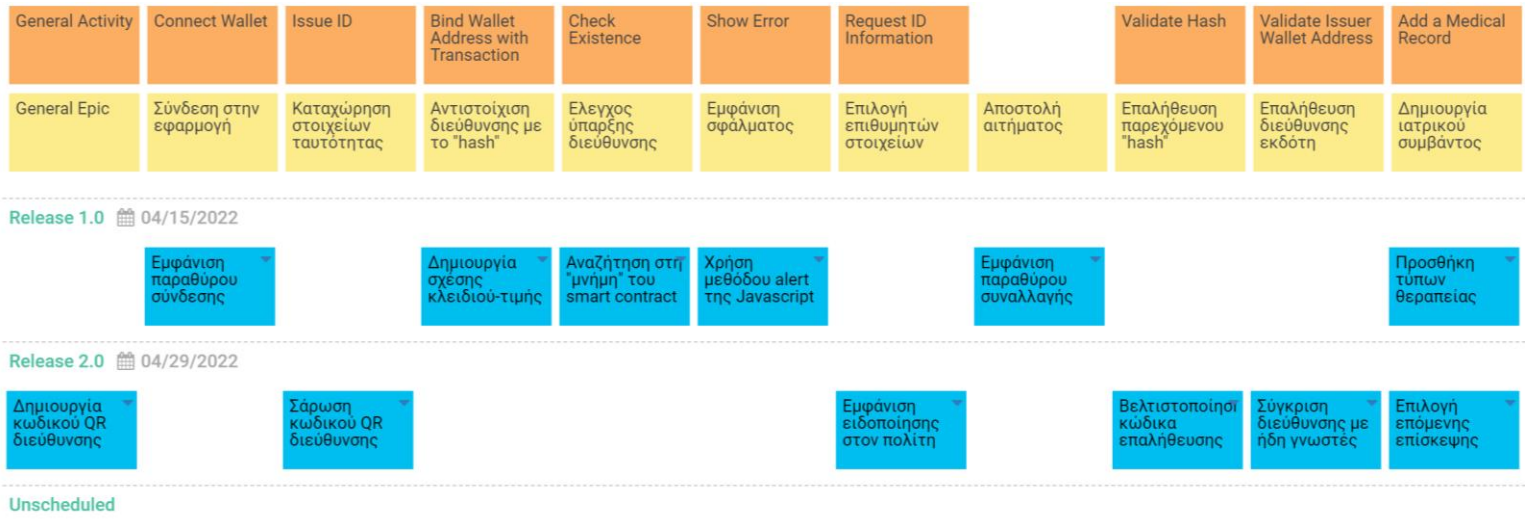


*Figure 28 - User Story Map*

## 2. Prioritized User Stories

The prioritization of "User Stories" refers to the unanimous decision of the Scrum team members as to the organization of the product features. The organization of the stories starts from the most important features so that the product can go to market as soon as possible (CardBoard, 2020). Each "User Story" has an associated Acceptance Criteria that determines its completion. Acceptance criteria provide clarity to the team about what is expected from a user story, remove ambiguity from the requirements and help shape expectations. Story points is a number that represents an estimate of the overall size of a user story. The overall size of a user story is assessed by taking into account the risk, the amount of effort required and the level of complexity of the story.

| Name | Description | Epic | Status | Acceptance criteria | Story Points | Priority | Risk |
|---|---|---|---|---|---|---|---|
| QR address code scanning | As an issuer/verifier, I want to have the option to scan the QR code of a citizen's wallet address in the context of typing it. | Registration of identity data | Approved | The issuer or verifier should have the option to scan a QR code in the citizen address input box. | 2 | Should | Low |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Create QR address code | As an ID owner, I wish to export a QR code consisting of the address of my wallet. | General Epic | Approved | The holder should have the option to export a QR code under the box containing his/her ID details. | 2 | Should | Low |
| Show notification to the citizen | As an identity owner, I wish to be informed when there is an unanswered data sharing request. | Selection of desired items | Approved | New notifications should be shown on the citizen's home page. | 2 | Should | Low |
| Optimisation of verification code | As a verifier, I want to be sure of the identity data that a citizen communicates to me. | Verification of the provided hash | Approved | Use an appropriate method to compare and verify the hash of the citizen's identity. | 5 | Must | High |
| Choice of next visit | As a citizen, I want to have the option of a follow-up visit when necessary. | Add/Create medical history | Approved | When adding/ creating a medical history there should be an option for the next visit. | 3 | Should | Low |
| Adding treatment types | As a citizen, I want the hospital that has taken care of me to cover a wide range of cases. | Add/Create medical history | Approved | When adding/ creating a medical history there should be a list of available | 3 | Should | Low |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | treatment options. | | | |
| 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 application, the login window to the application should automatically appear from the wallet application. | 1 | Must | Low |
| Creating a key-value relationship | As a issuer, I wish to keep, for security reasons, a list of the addresses and hashes of the id registrations I have made. | Matching an address with the hash | Approved | Configuring the code of the issuer's smart contract to support such a feature. | 2 | Must | Medium |
| Searching the storage of the smart contract | As an issuer/verifier I only wish to perform actions for existing citizen addresses. | Checking the existence of an address | Approved | Check the address provided by the citizen before any action is taken. | 2 | Must | Medium |
| Using the Javascript alert method | As a user of the application, I wish to be informed of any event that takes place during my “stay” on the application. | Error display | Approved | Display a message about anything that happens while using the application. | 1 | Should | Low |
| Show transaction window | As a user of the application, I want the transaction processing | Send request | Approved | When requesting a transaction, the | 1 | Must | Low |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | window to appear automatically. | | | corresponding dialog box should automatically appear. | | | |
| Comparison of address with already known addresses | As a verifier, I want to confirm the ownership of an identity, not only by verifying its hash, but also by verifying the address of its issuer. | Verification of issuer's address | Approved | Check the issuer's wallet address, when requested, by comparing it with those contained in a list of known addresses. | 4 | Must | High |

*Table 6 - User Stories Prioritization*

# Release Plan

A Release Plan allows the Scrum team to have an overview of the releases and delivery schedule for the product they are developing, so that they can comply with the product owner's expectations.

## 1. Project Deliverables

A deliverable is a tangible or intangible good or service to be produced during a project.

| Deliverable | Description | planned release date | Priority | Status | Owner |
|---|---|---|---|---|---|
| Initial model of the decentralized identification application | An application model that exploits the benefits of Blockchain technology in order to better manage | 2022-04-01 | High | Done | Christos Bandis |

| Deliverable | Description | planned release date | Priority | Status | Owner |
|---|---|---|---|---|---|
| | personal data and avoid their misuse. | | | | |

*Table 7 - Project Deliverables*

## 2. Release Configuration

| Release | Description | planned release date |
|---|---|---|
| Release 1.0 | Fully functional application, based on the application model presented in the deliverable. | 2022-04-15 |
| Release 2.0 | Enrichment of the application with new functions and renaming it to “IDEN - Blockchain Authentication” | 2022-04-29 |

*Table 8 - Project Releases*

## 4.4. Structure analysis

After continuous study and testing, the most efficient method of project completion and the tools that contributed to the implementation of the application were defined. Its development was based on a local Blockchain network, the nodes of which communicated with the application through a “library” of the programming language “Javascript”, while for the writing of the smart contracts the programming language “Solidity” was used. For the execution of the code, it was deemed necessary to have the “Node.js” execution environment, which included sub-applications that helped to facilitate the required processes.

More specifically, the connection to the Blockchain is achieved through a “REST API” (system-to-system communication method), which returns a “JSON” (JavaScript Object Notation) file containing all the elements that make up a smart contract. In the development of modern decentralized applications, it is customary to use a “library” of the respective programming language, which contains ready-made methods and procedures that facilitate communication with the Blockchain, serving mainly in the encoding and decoding of JSON files. For the development of this work, the library “Web3.js” of the programming language Javascript was used.

Regarding smart contracts, there are other languages besides “Solidity” for their development, which are all compiled into “bytecode” (a type of binary code) that is addressed to “EVMs” (Deol and Wiesner, 2021). The “EVM” (Ethereum Virtual Machine), as mentioned in Section 2.3, is a virtual machine “placed” in the core of the Ethereum Blockchain, whose purpose is to interpret and execute the code of smart contracts. Nominally, some smart contract programming languages are Solidity, Vyper, LLL and Mutan, with Solidity being the most popular of them.

The underlying architecture of the compiler of Solidity comes from ECMAScript (European Computer Manufacturers Association Script) and because of this many compare its appearance to that of Javascript. The code of a smart contract starts with a statement of the copyright license type that characterizes it. Then, the phrase “pragma solidity” and the version of the compiler are found. This line of code is already precompiled and does not need to be recompiled at each execution. Its purpose is to guarantee the compiler version, a feature that greatly enhances security by ensuring that the code of the smart contract is not unstable, despite future changes that may be implemented in the language structure (Deol and Wiesner, 2021). During the development of the application, the most recent version of Solidity was “0.8.13” and, the “^” symbol distinguished in Figure 29, means that the smart contract code can be processed by compilers of that version and later versions. To conclude, it continues with the name of the smart contract, which starts with a capital letter followed by brackets, containing the main code of the contract.

```
1   // SPDX-License-Identifier: MIT
2   pragma solidity ^0.8.13;
3
4   contract Issuer {
5
```

*Figure 29 - Code initiation sample in Solidity*

The process of writing the smart contracts was carried out in such a way that only the basic actions and methods necessary for the operation of the application are executed on the Blockchain. The reason why this method was chosen is that storing a large amount of data on a Blockchain, such as Ethereum, comes along with a large monetary cost. As mentioned in Chapter 2, the operation of public Blockchains depends

on the use of a native token. Ethereum's "Yellow Paper" states that storing a word consisting of 256 bits consumes 20.000 gas (the "fuel" used for transactions within the Blockchain) (JAXenter, 2019). By converting these bits into bytes (8 bits = 1 byte), it can be concluded that the size of this word is 32 bytes. Therefore, to store one (1) kilobyte (1024 bytes - 1024 / 32 = 32 words), 32 * 20,000 = 640,000 gas will be needed. As the use of the Blockchain and the value of the Ethereum cryptocurrency are constantly changing, there is no "cheaper" method of storage for this network. For example, at the time of writing (May 2022), the price of fuel is 52 "gwei" (Ethereum's smallest unit of measurement), or 0.0000000679 "ETH" (1 gwei corresponds to 0.00000000130 Ethereum - ETH). Based on this, storing one kilobyte of data would cost an average of 640,000 * 0.0000000679 = 0.043456 ETH, or €79.23 at the current ETH/EUR exchange rate.

In conclusion, apart from the checks that are performed using the Javascript language at the browser level, the methods of smart contracts have been "designed" in such a way that their function attributes are examined before the execution of any code and then an "event" is triggered that sends the appropriate message, where required. More specifically, at the smart contract level, a check is made to verify the matching of a wallet address to an existing citizen or verifier identity, a crucial process, as the wallet addresses are a key element of the core functionality of the application. Furthermore, the project structure includes a smart contract called 'Ownable', which is about the wallet addresses of issuers. Its main purpose is to apply a rule (modifier) to the functions of the other smart contracts, allowing them to be executed only by the owner of one of these addresses. In this way, the security of the system is enhanced, because if a malicious user becomes involved in the operation of the application and attempts to perform any action, pending transactions are automatically rolled back and never completed. In the project files accompanying the work, this rule has been removed, but not the smart contract itself, since when the "Ganache" application is started to activate a local Blockchain, the wallet addresses that will be generated will be different from those used to develop the application, which implies with the aliasing of the issuer address and finally, the impossibility of performing all the actions.

## 4.5. Technical parts

The technologies and applications used during the development of the application are the following:

- **Ganache: A**pplication that provides a local, private 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 implementing smart contracts on various Blockchain platforms, mainly Ethereum.
- **Javascript:** Object-oriented programming language for enriching and increasing the dynamics of the graphical environment of the application.
- **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.
- **Web3.js:** Collection of Javascript libraries that allow the development of "Web3" applications and interaction with a local or remote Ethereum node.
- **Chai:** Behavioral-Driven Development / Test-Driven Development (BDD/TDD) library for Node.js that 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.
- **QR Code Javascript Libraries:** Javascript libraries for interacting with Quick Response Codes (QR Codes).
- **Visual Studio Code:** Code editing application.
- **Metamask:** Software wallet in the form of a web browser extension, used to interact with Blockchains, such as Ethereum.
- **Infura:** Infrastructure to develop and publish "Web3" applications to main or public test Blockchains.
- **Visual Paradigm:** Modeling and project management application.
- **macOS:** Operating system on which the app was developed.

### 4.5.1. Installation guide for necessary applications (Windows)

This chapter is about installing the technical parts needed to run the application code. More specifically, it will be mentioned how to install the local Blockchain "Ganache", the wallet "Metamask" in the form of an extension for the browser "Google Chrome", "Node.js", as well as the necessary modules for the operation of the application. For a better understanding of the content of the current and the following chapter, the first person plural will be used along with the third person singular.

#### 1. Ganache

Ganache, as mentioned above, is a simulation application of a fully functional local Blockchain for developing applications on the Ethereum Blockchain. The versions of the application are available in the "ganache-ui" repository on "Github" (Figure 30), and the link https://github.com/trufflesuite/ganache-ui/releases/download/v2.5.4/Ganache-2.5.4-win-setup.exe automatically downloads the installation file.

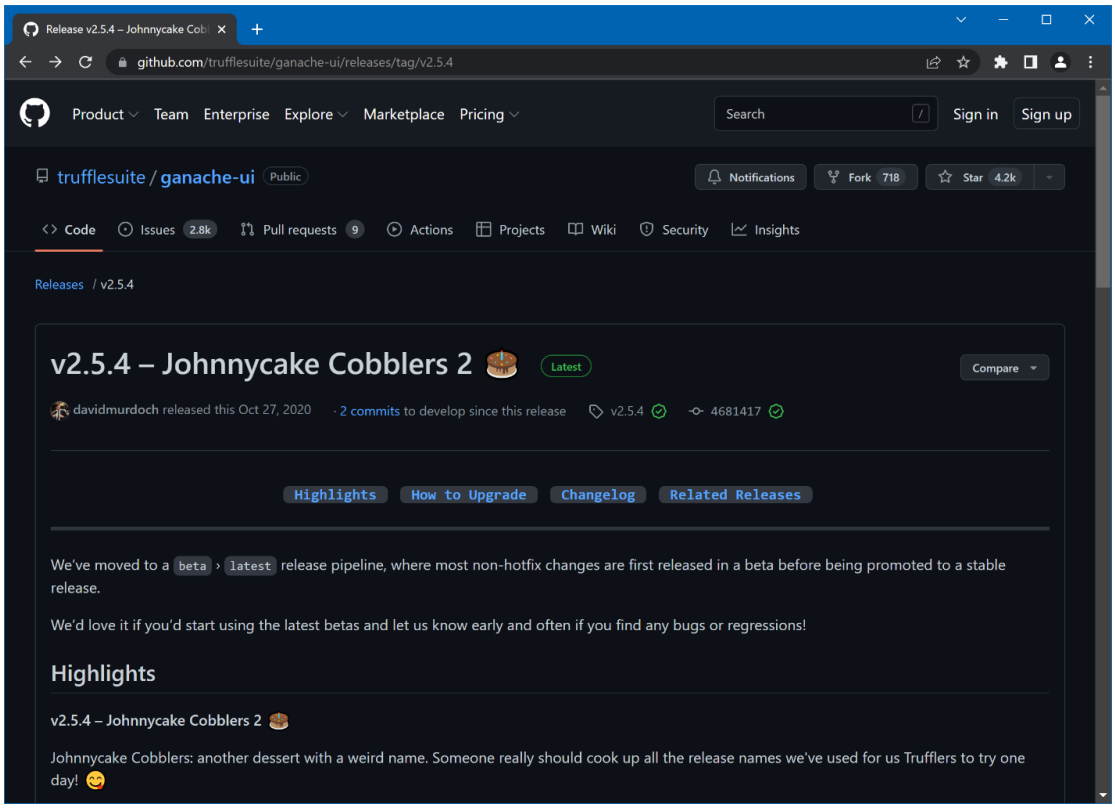


*Figure 30 - The "Ganache" repository on "Github"*

All options during the installation remain as they are. Upon completion, we select the "Run Ganache" box and press the "Finish" button.

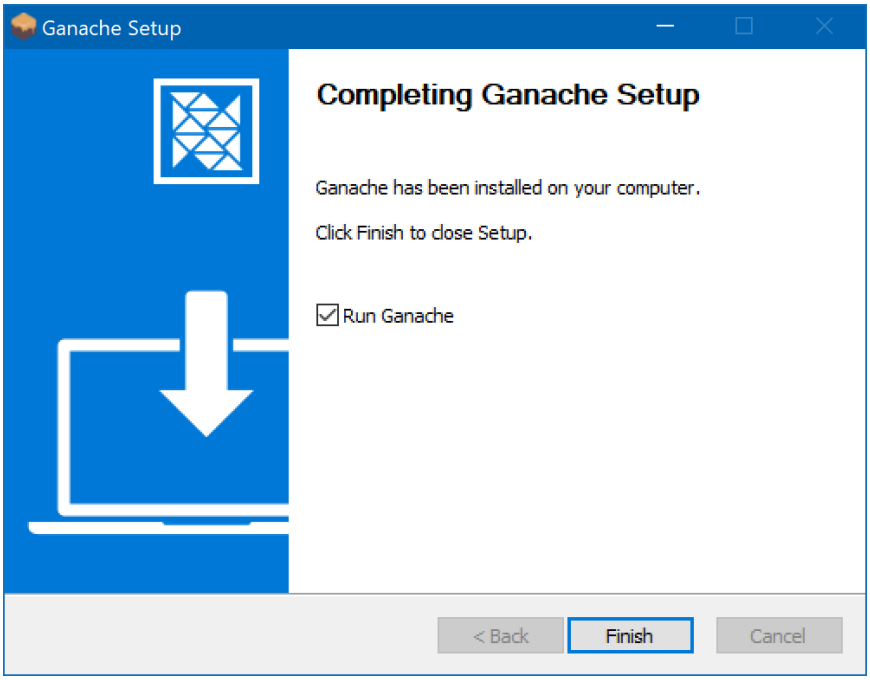


*Figure 31 - Completing the installation of "Ganache"*

Once the window appears, we click on the "QUICKSTART" button, checking first if the word "ETHEREUM" is written under it (Figure 32).

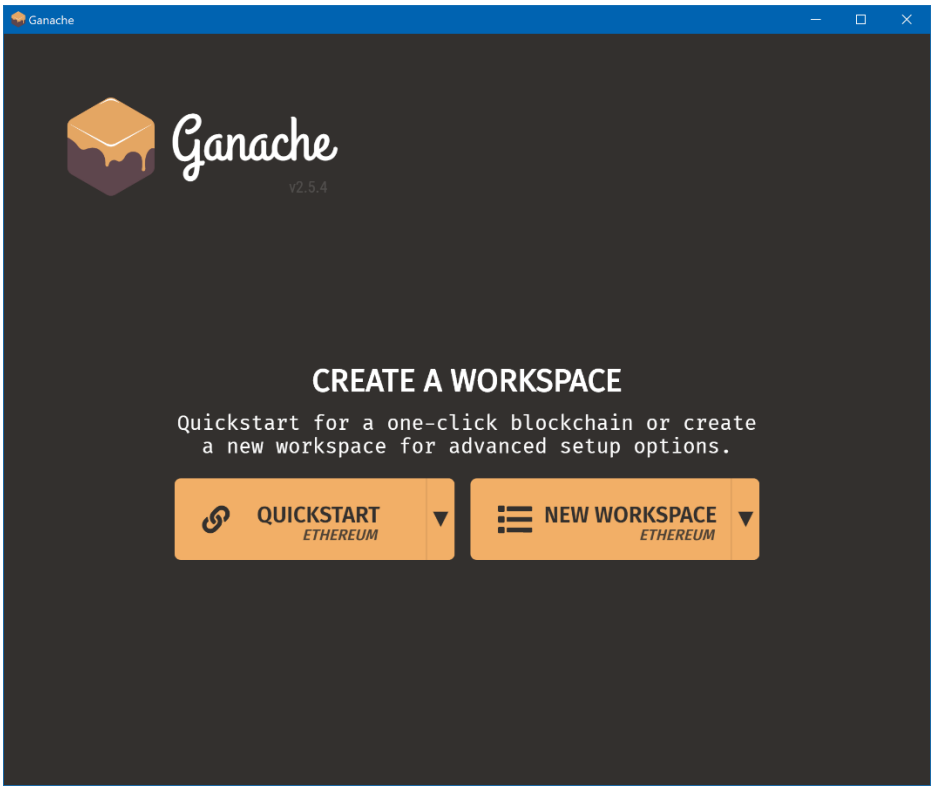


*Figure 32 - Home screen of "Ganache"*

This action will create a new local Blockchain, containing ten (10) accounts with a balance of 100 ETH each (Figure 33). Other features distinguished in the active window are the “Mnemonic”, a phrase with twelve (12) random words that will be needed in the installation of Metamask, information about the current Blockchain network, such as the current block, the Network ID, etc.

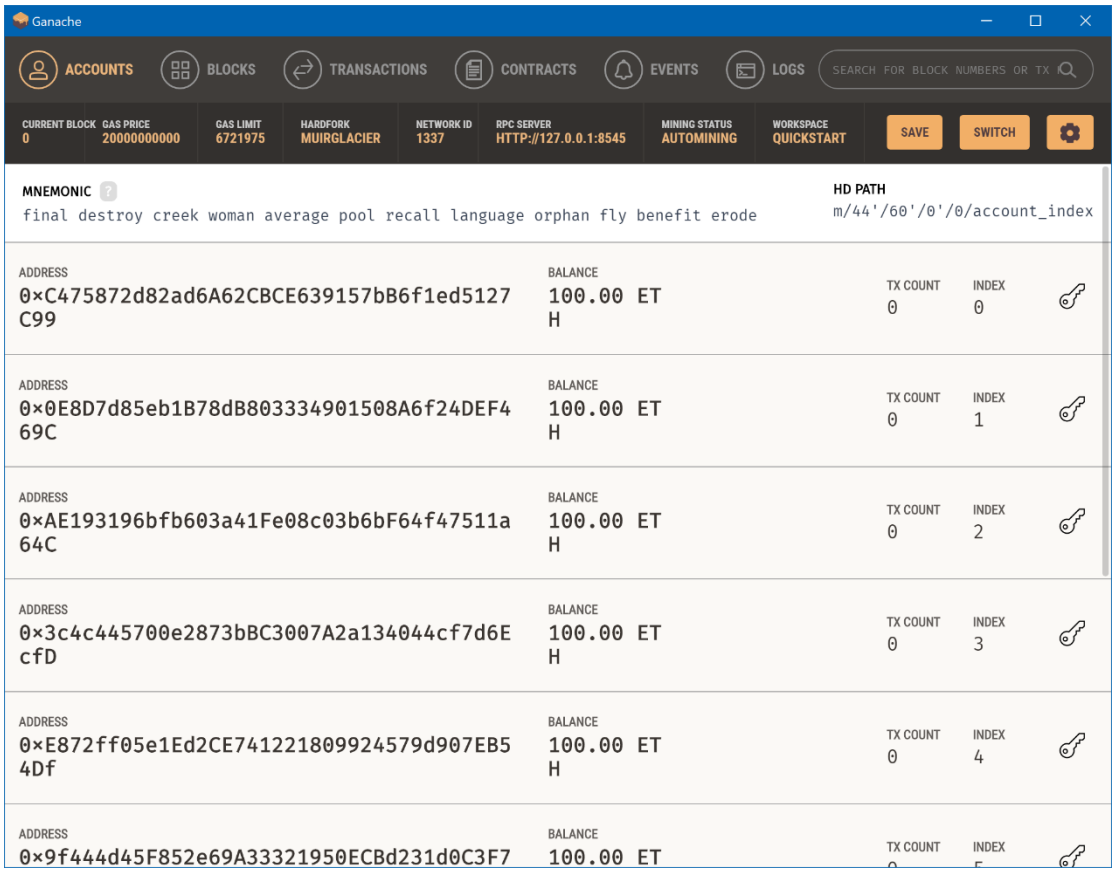


*Figure 33 - Blockchain management window*

By clicking on the gear in the upper right part of the window, you can go to the network settings, where you can modify items such as the name of the workspace (Figure 34), the RCP Server address and the Port Number (Figure 35).

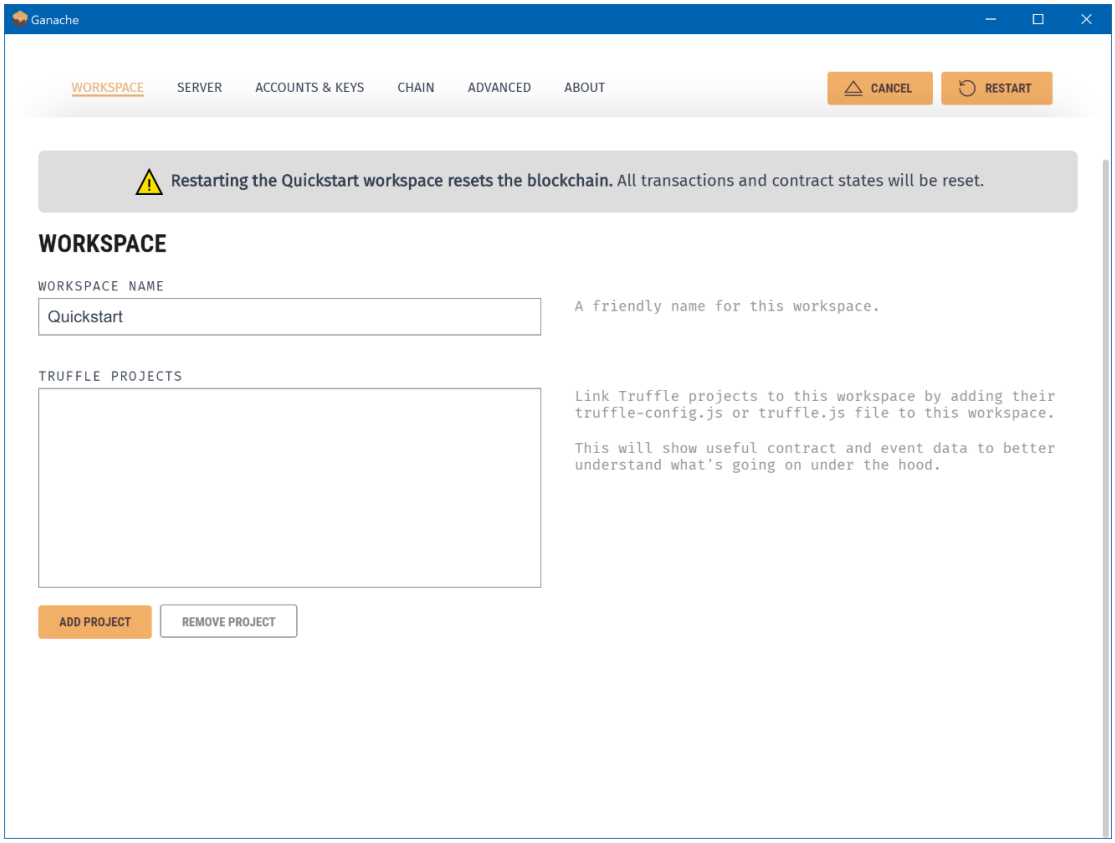


*Figure 34 - Workspace name window*

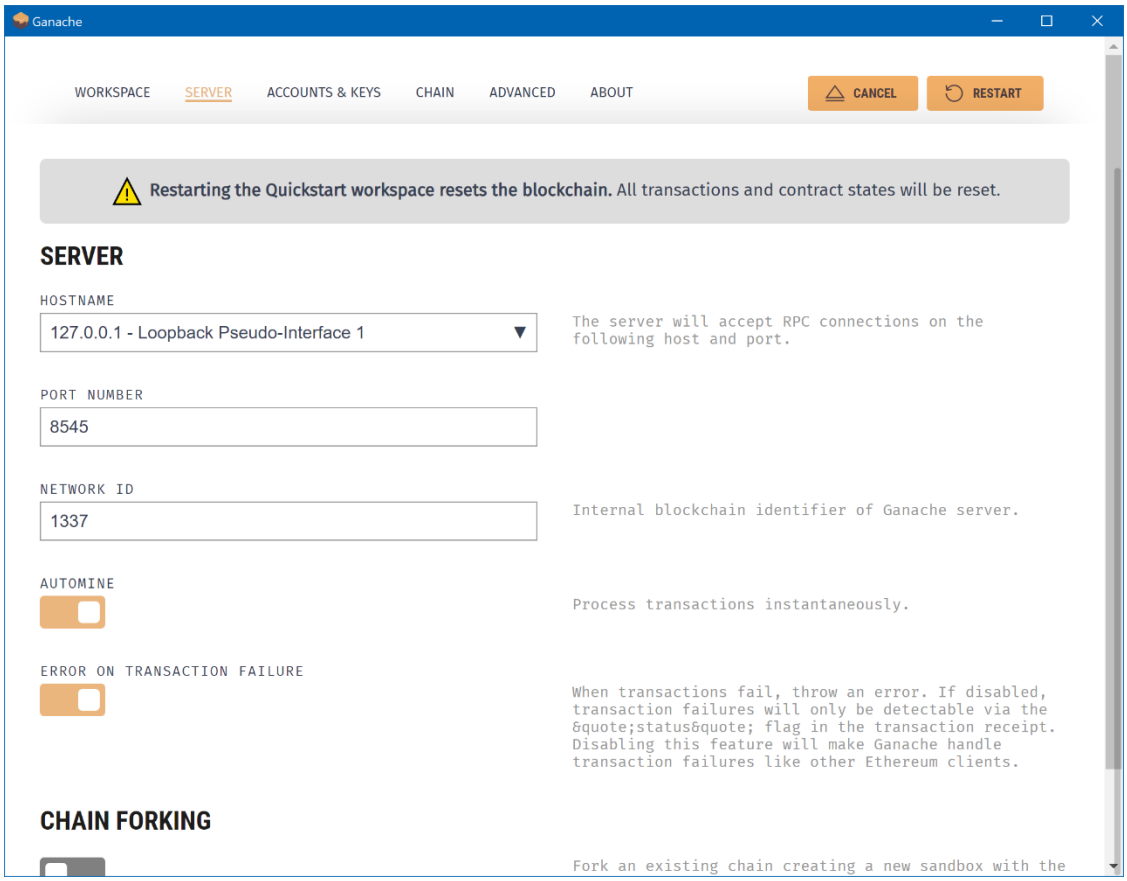


*Figure 35 - Server modification window*

For the communication of the application and Metamask with the local Blockchain network, a specific “Port Number” and “Network ID” are required. More specifically, the “Port Number” must correspond to “8545” and the “Network ID” to “1337”. If there are different 4-digit numbers in the “Ganache” network settings, then we need to modify them and then click “Save and Restart” in the top right of the window to apply the changes.

## 2. Metamask

The “Metamask” wallet application is available through its official website: https://metamask.io/download/ (Figure 36), but also through the online store of the preferred browser, in this case “Google Chrome”, at: https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn (Figure 37).

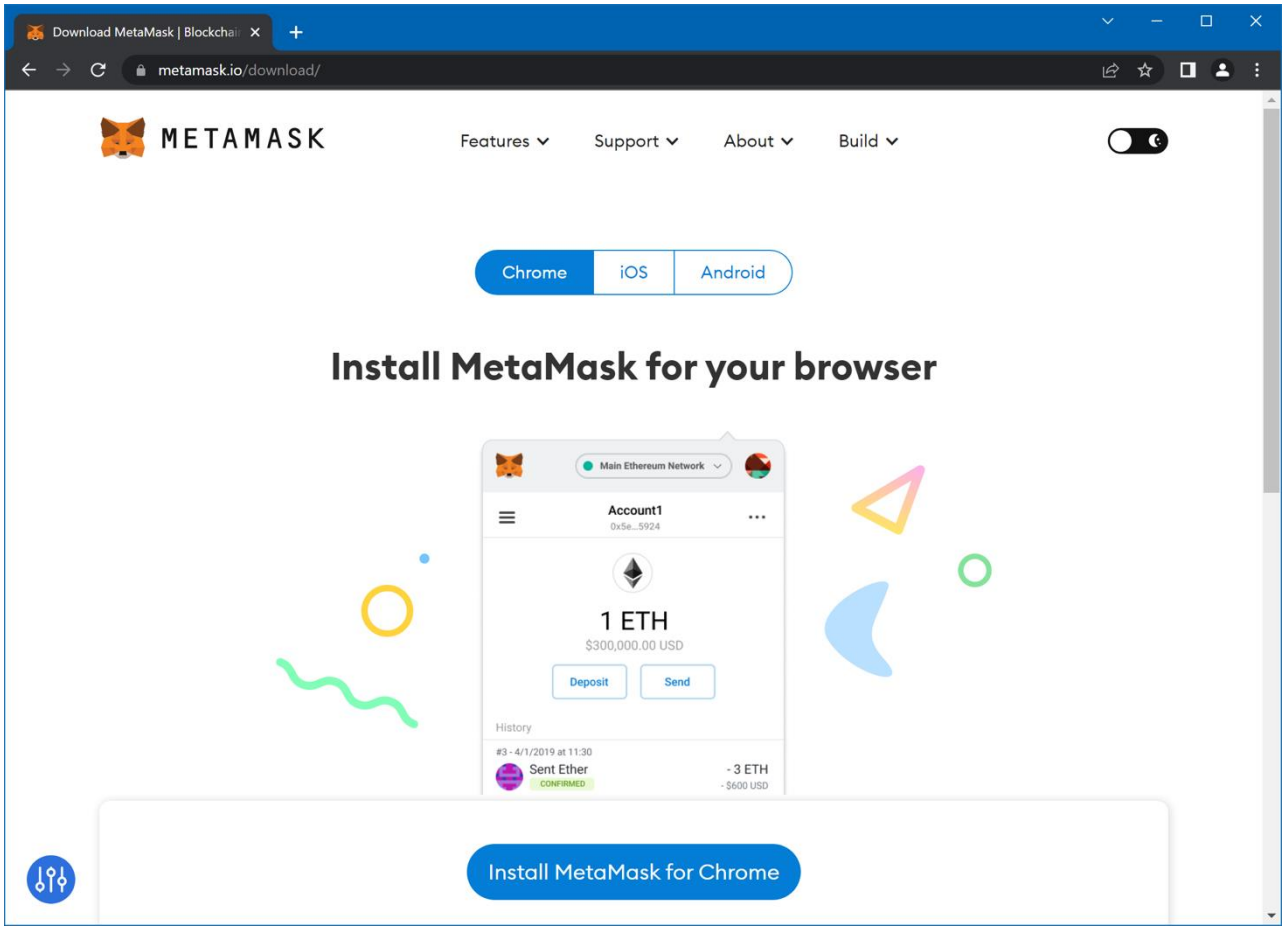


*Figure 36 - Official website of “Metamask”*

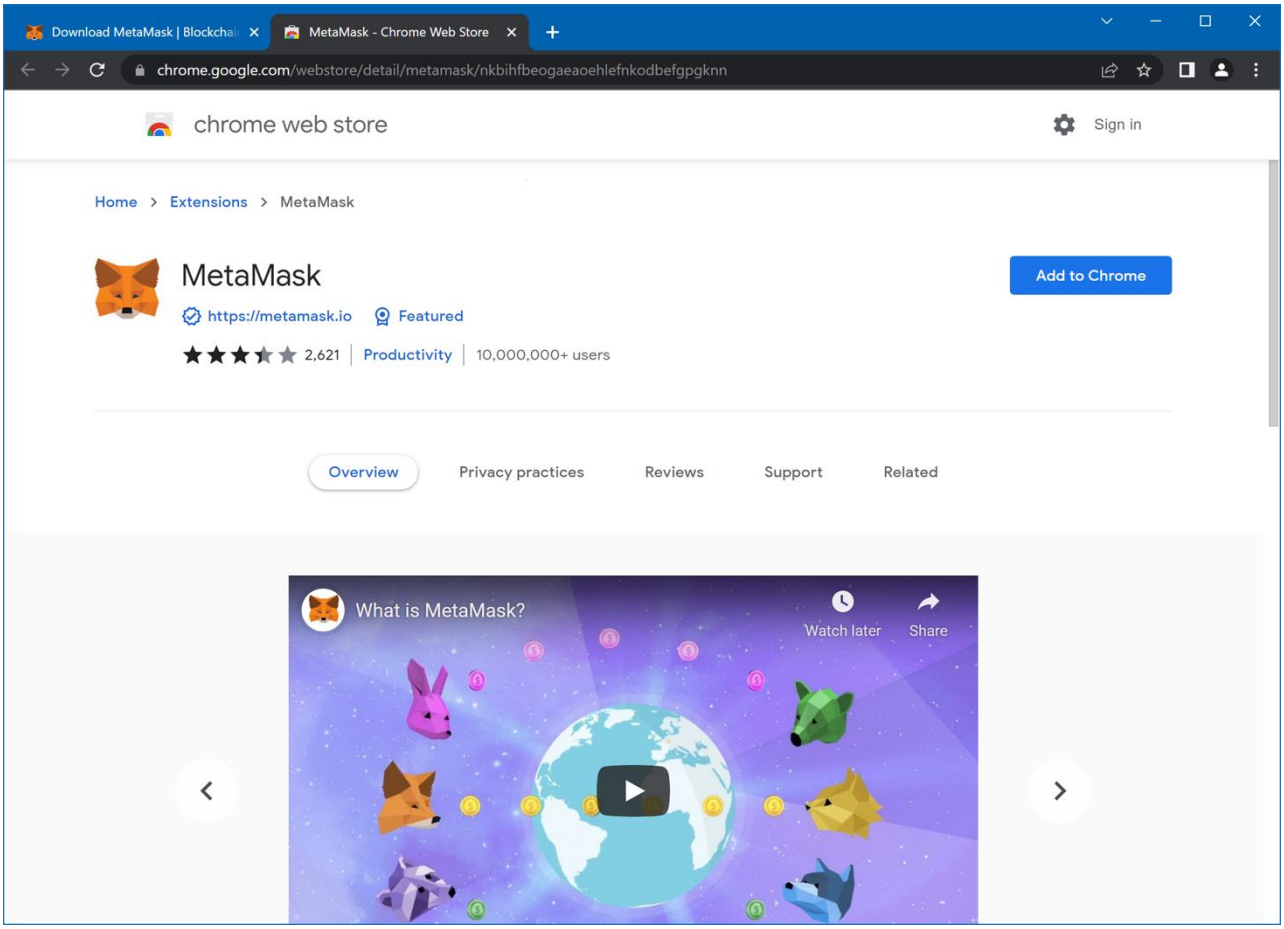


*Figure 37 - "Google Chrome" online store*

Before the download process starts, the message in Figure 38 appears, asking the user for certain permissions. Clicking on the "Add extension" button adds the extension to the browser.

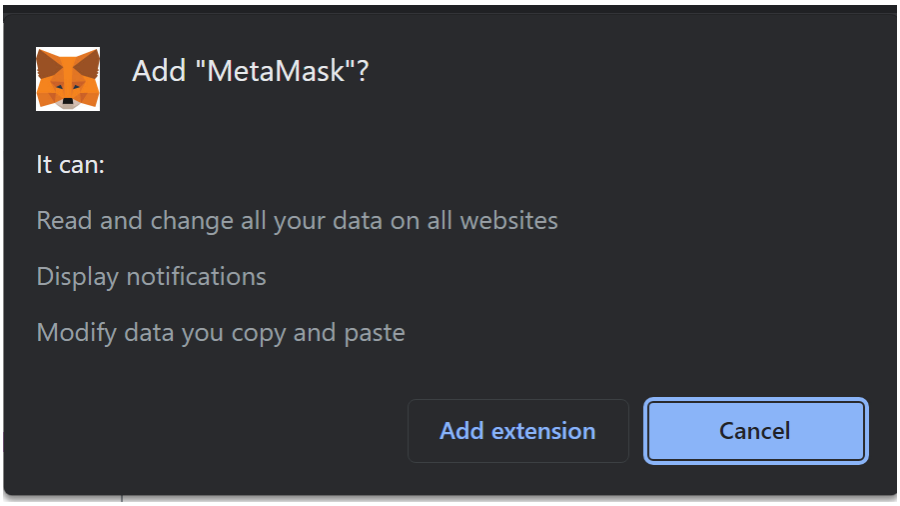


*Figure 38 - Approval message to add the extension to "Google Chrome"*

Once the installation is complete, Metamask's home page is displayed in a new tab (Figure 39), followed by the options of importing an existing wallet via "Mnemonic" or creating a new one (Figure 40).

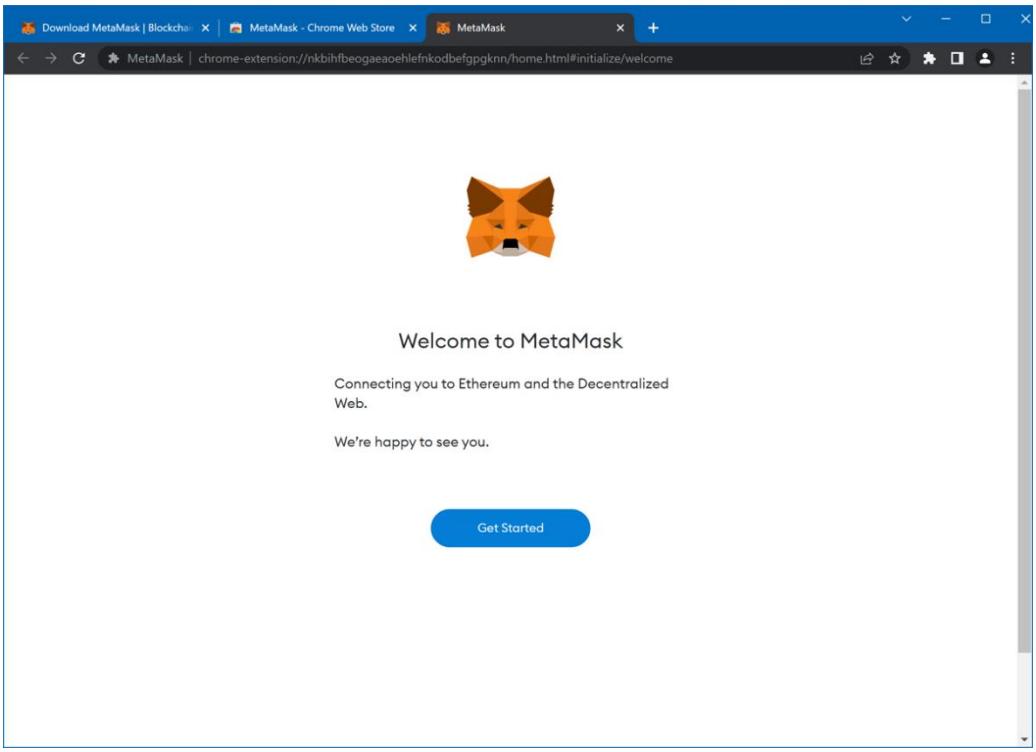


*Figure 39 - Home page of "Metamask"*

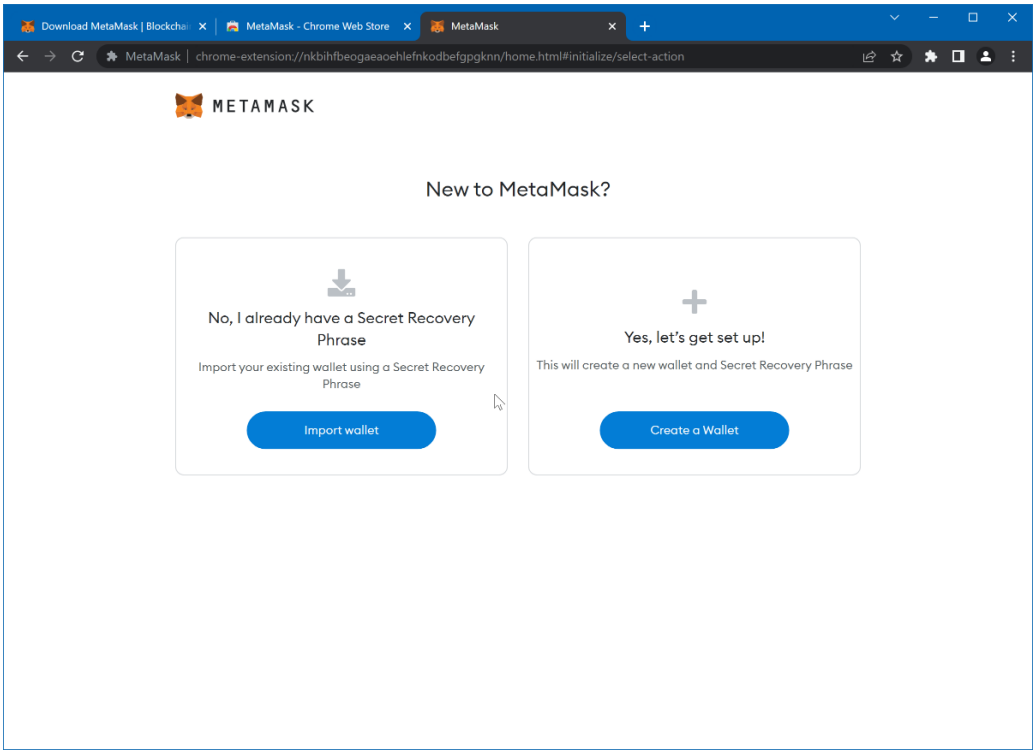


*Figure 40 - Options to import or create a "wallet"*

By selecting the "Import wallet" option, twelve boxes are displayed, one for each word of the "Mnemonic" and below that a box for entering a new password (Figure 41). At this point, we open the Ganache window and copy the "Mnemonic" as mentioned in the "Ganache" installation (see page 55) and paste it in any field, according to the message in the middle of the window. After typing the desired password as well, we proceed to import the wallet account.

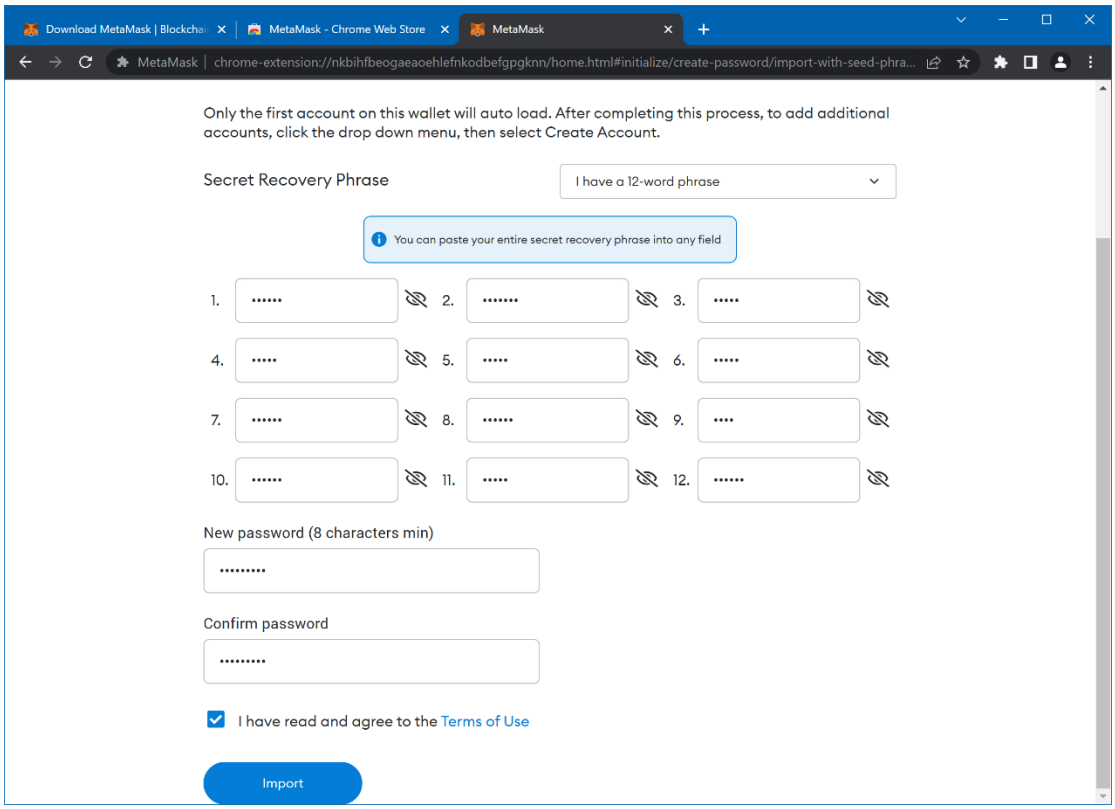


*Figure 41 - Importing an existing wallet account*

We are then "taken" to the wallet management interface, which contains information about the account balance in "ETH" and other cryptocurrencies based on "Ethereum", the account activity, etc.

For the application to work, it is necessary to add the local network of "Ganache" to the list of available networks of "Metamask". This is achieved by selecting "Ethereum Mainnet" in the top right corner of the window and then clicking on "Show/hide test networks" (Figure 42).

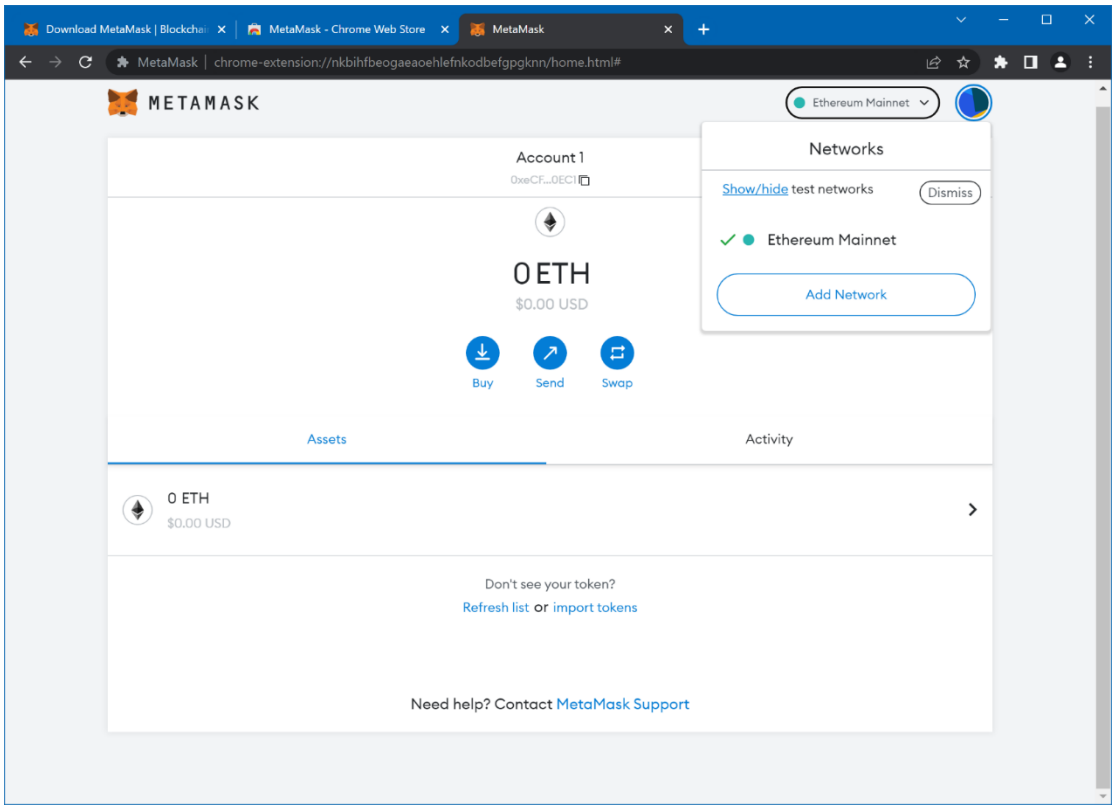


*Figure 42 - "Metamask" wallet management environment*

From there, we turn on the "Show test networks" switch at the top of the page we will be taken to (Figure 43). We can confirm that the test networks are displayed by returning to the wallet management interface and clicking on "Ethereum Mainnet" (Figure 44).

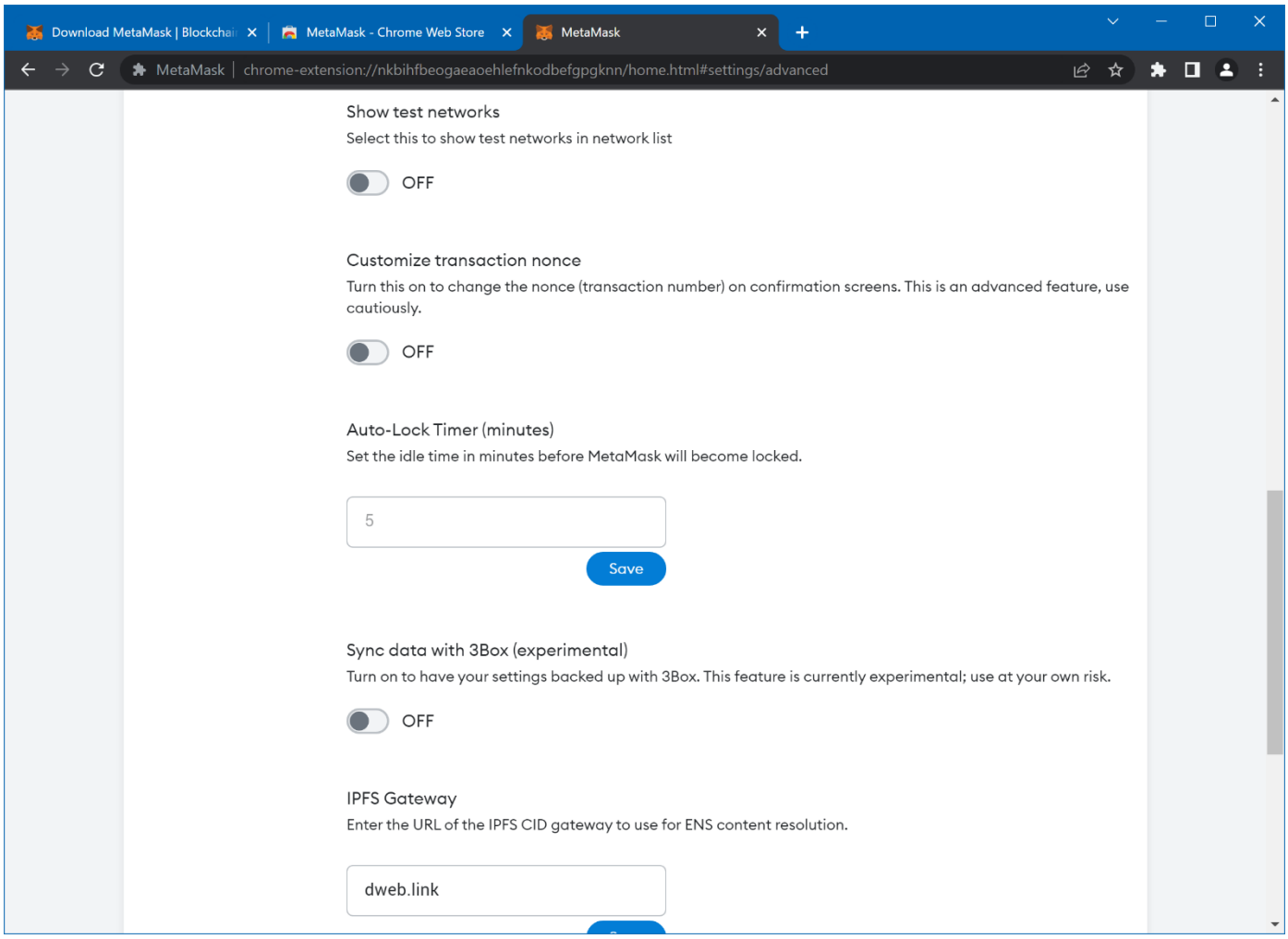


*Figure 44 - Test network activation switch*

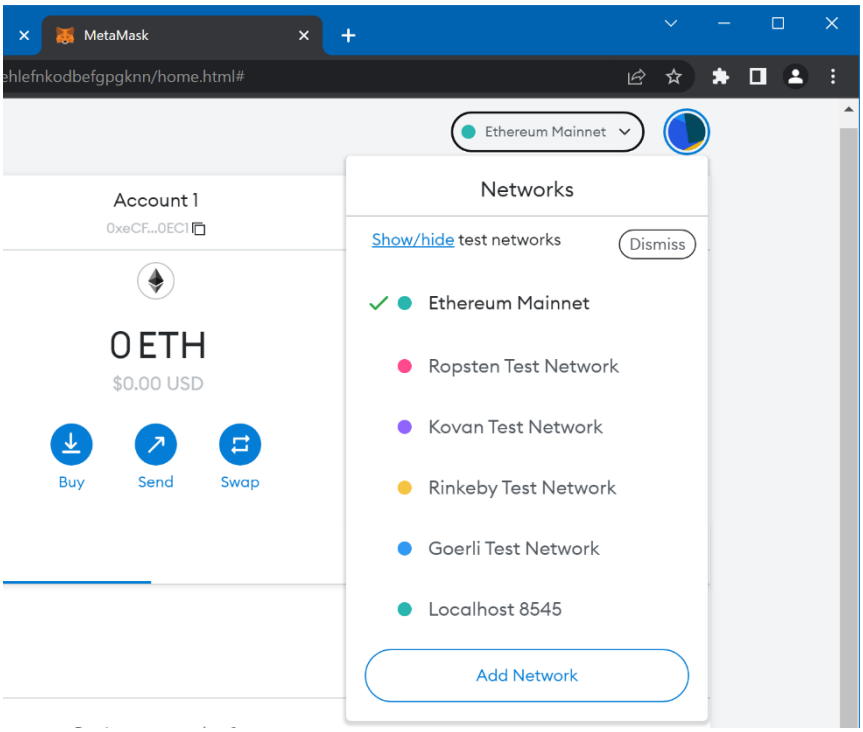


*Figure 43 - The predefined test blockchain networks included in "Metamask"*

From the menu in Figure 44, select the network “Localhost 8545”. We will notice that the account balance will change from 0 ETH to 100 ETH, which is implied by the successful connection to the local network of “Ganache”.

## 3. Node.js

The installation of “Node.js” is equally simple. It is downloaded from its official website via the link: https://nodejs.org/en/download/ (Figure 45).

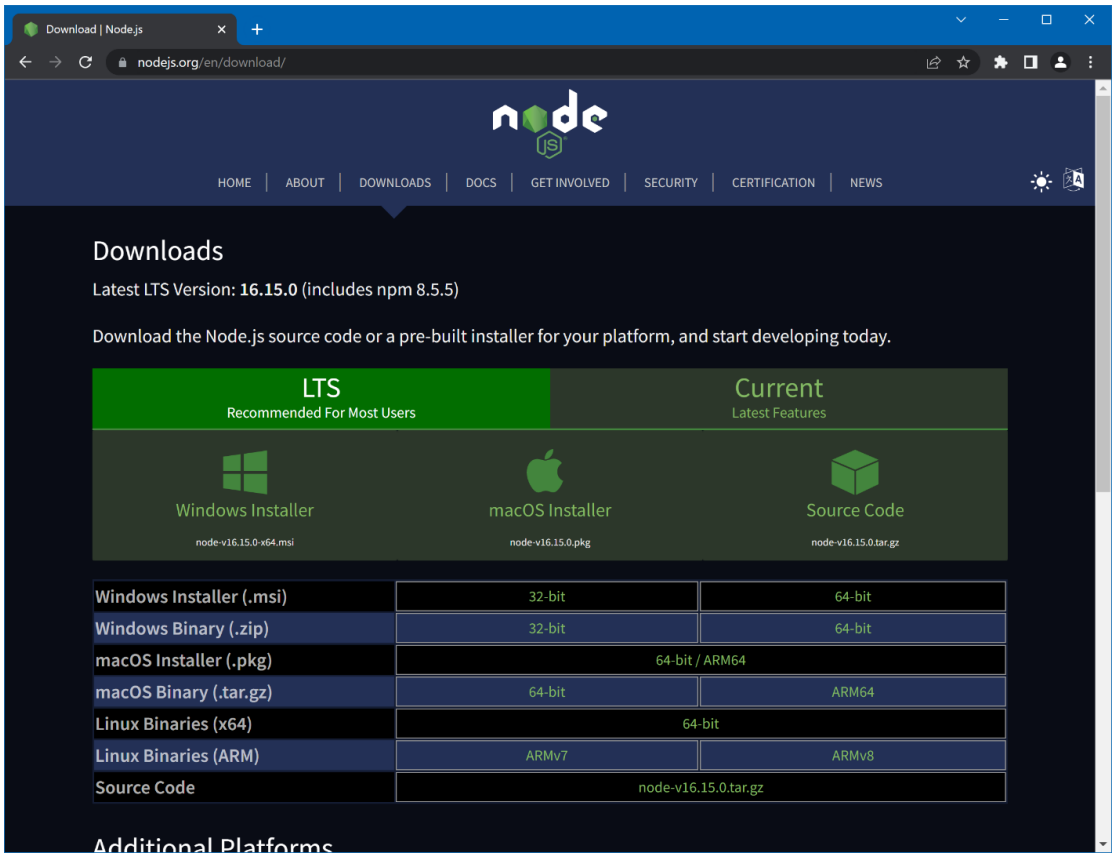


*Figure 45 - Download website of “Node.js”*

After the download is complete, we run the installation file and start the process. All options are left as they are, and it is important to note that the checkbox in Figure 46 must be checked, as it is necessary to install additional tools to run the application.

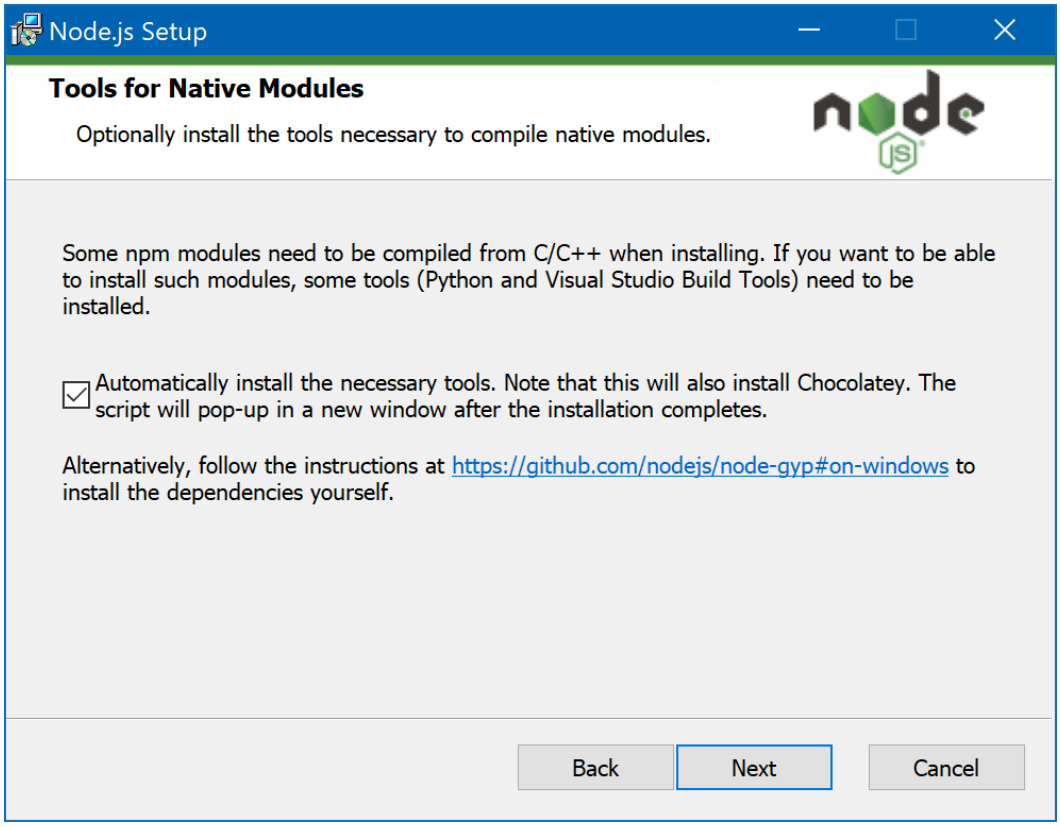


*Figure 46 - Required option when installing "Node.js"*

After the process is completed and the "Finish" button is pressed, two "CMD" (command line) windows appear. First, the window shown in Figure 47 appears. To execute the file, we are prompted to press any key. Immediately, this window will close and the window in Figure 48 will appear, in which the same action is required. Then, a "Windows PowerShell" window will make its appearance, in which the installation of the tools will be completed (Figure 49).

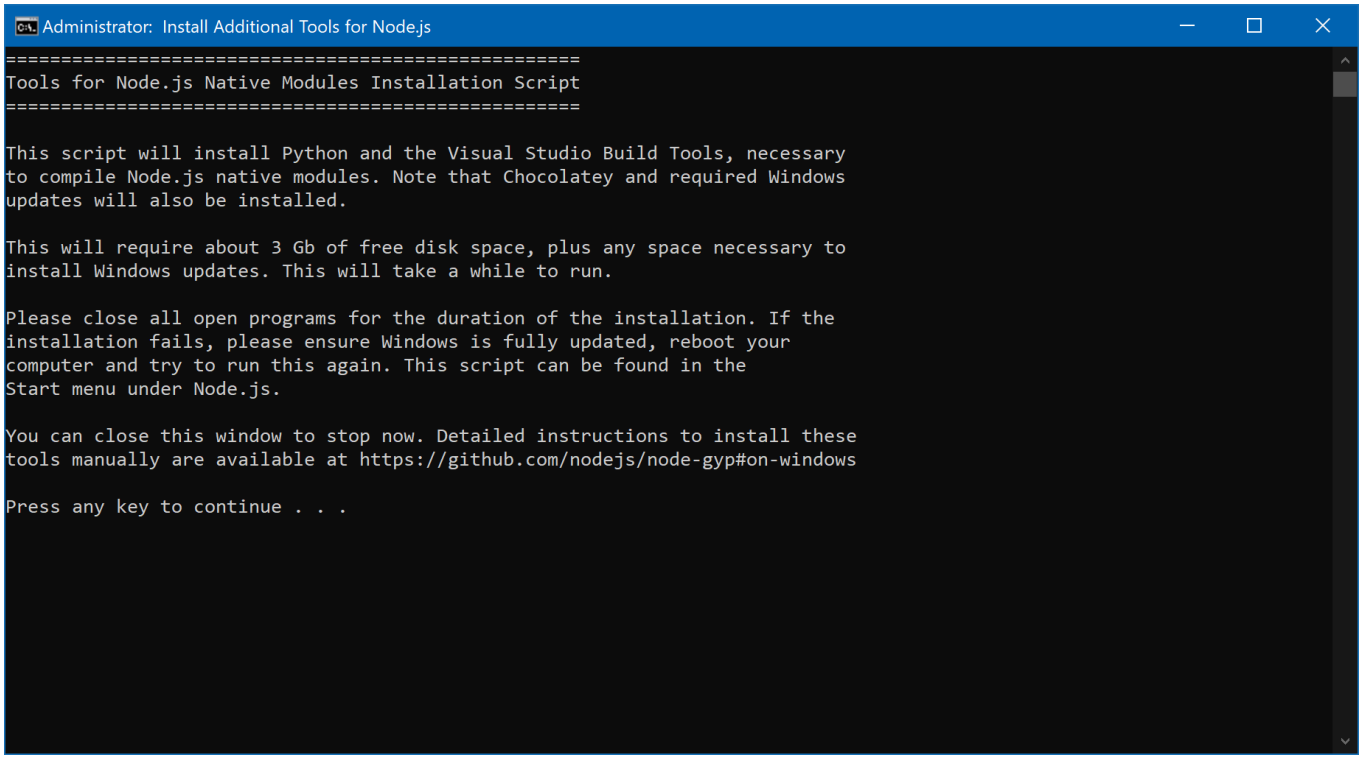


*Figure 47 - First "CMD" window for installing additional "Node.js" tools*

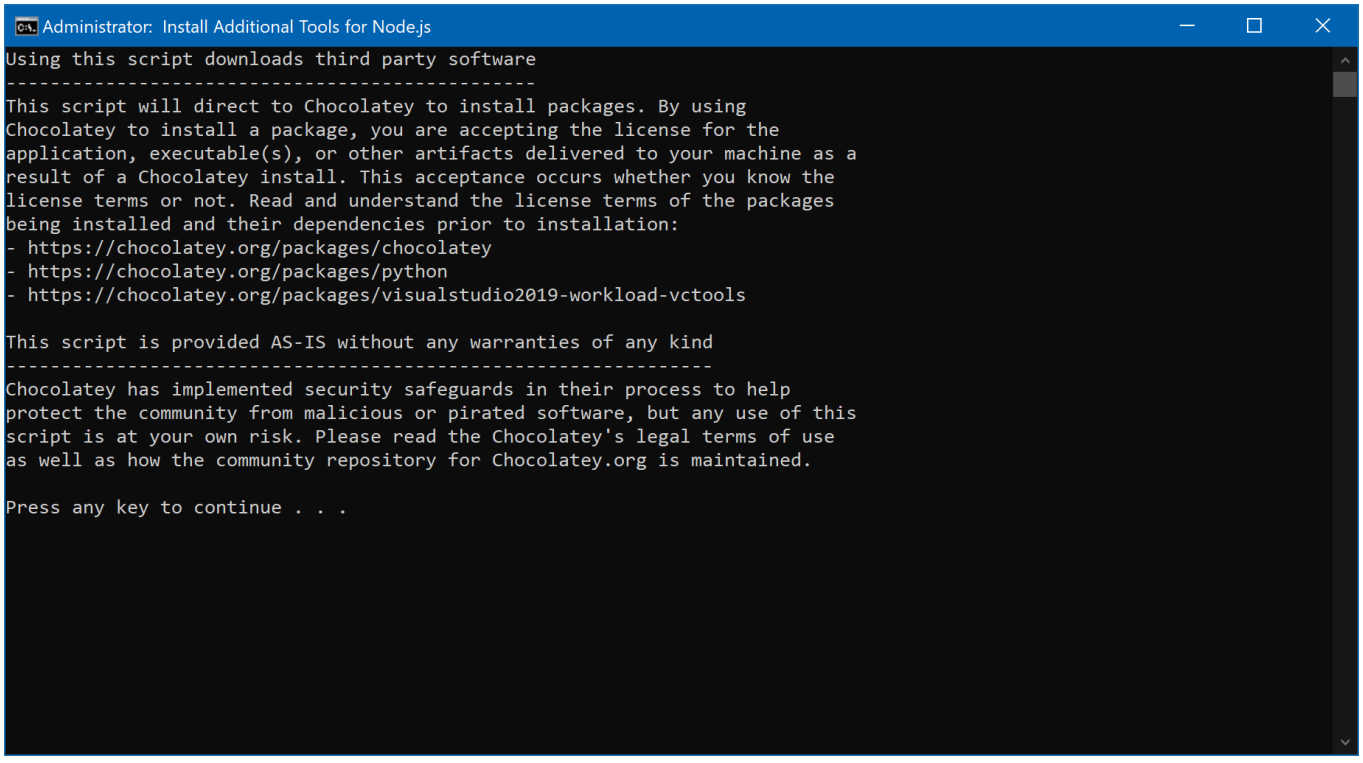


*Figure 48 - Second "CMD" window for installing additional Node.js tools*

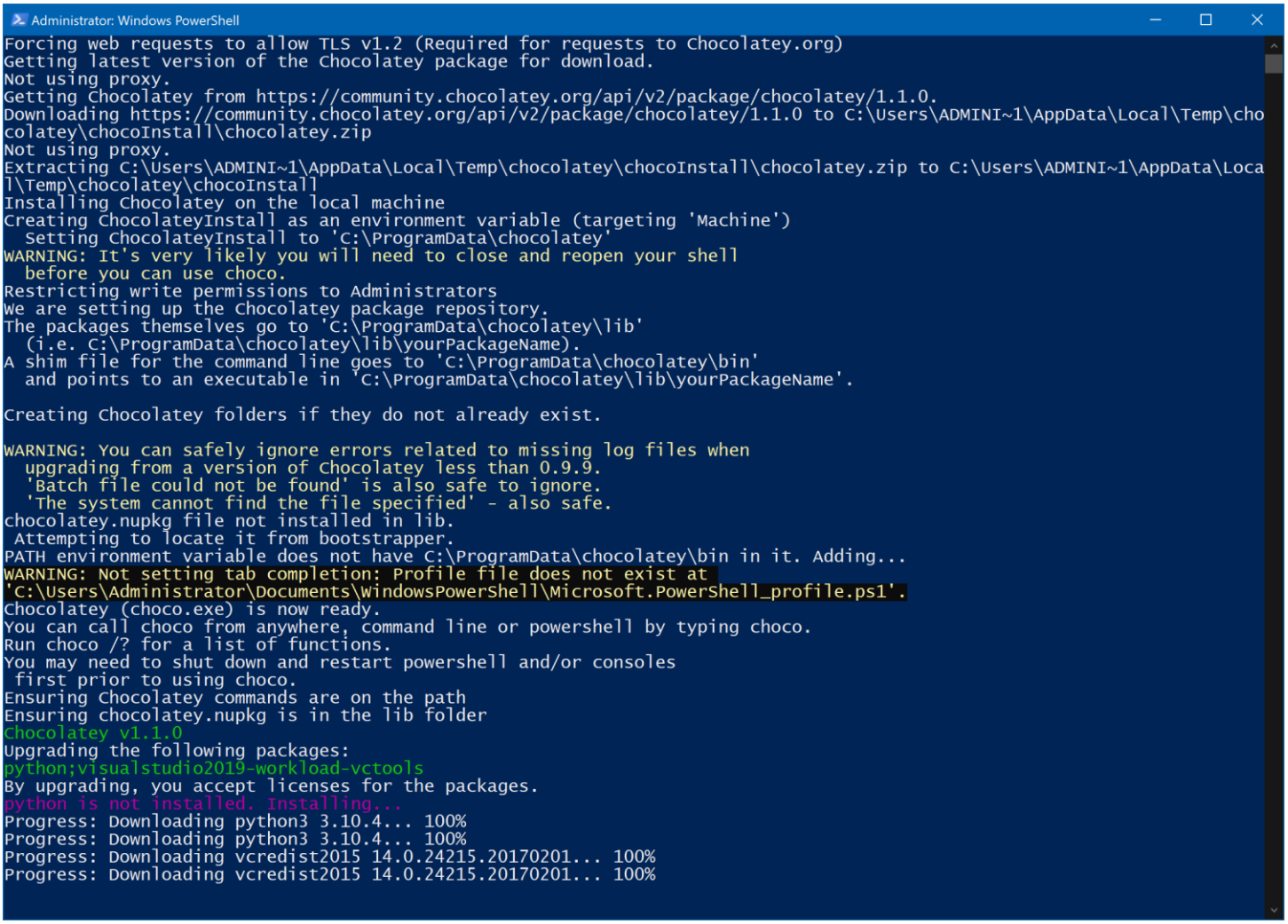


*Figure 49 - Start installation of additional tools in "Windows PowerShell"*

The installation will take from fifteen (15) to twenty (20) minutes, because, as indicated in Figure 47, the size of the files is about 3GB. Once the process is complete, the "Microsoft PowerShell" window will look like the one in Figure 50.

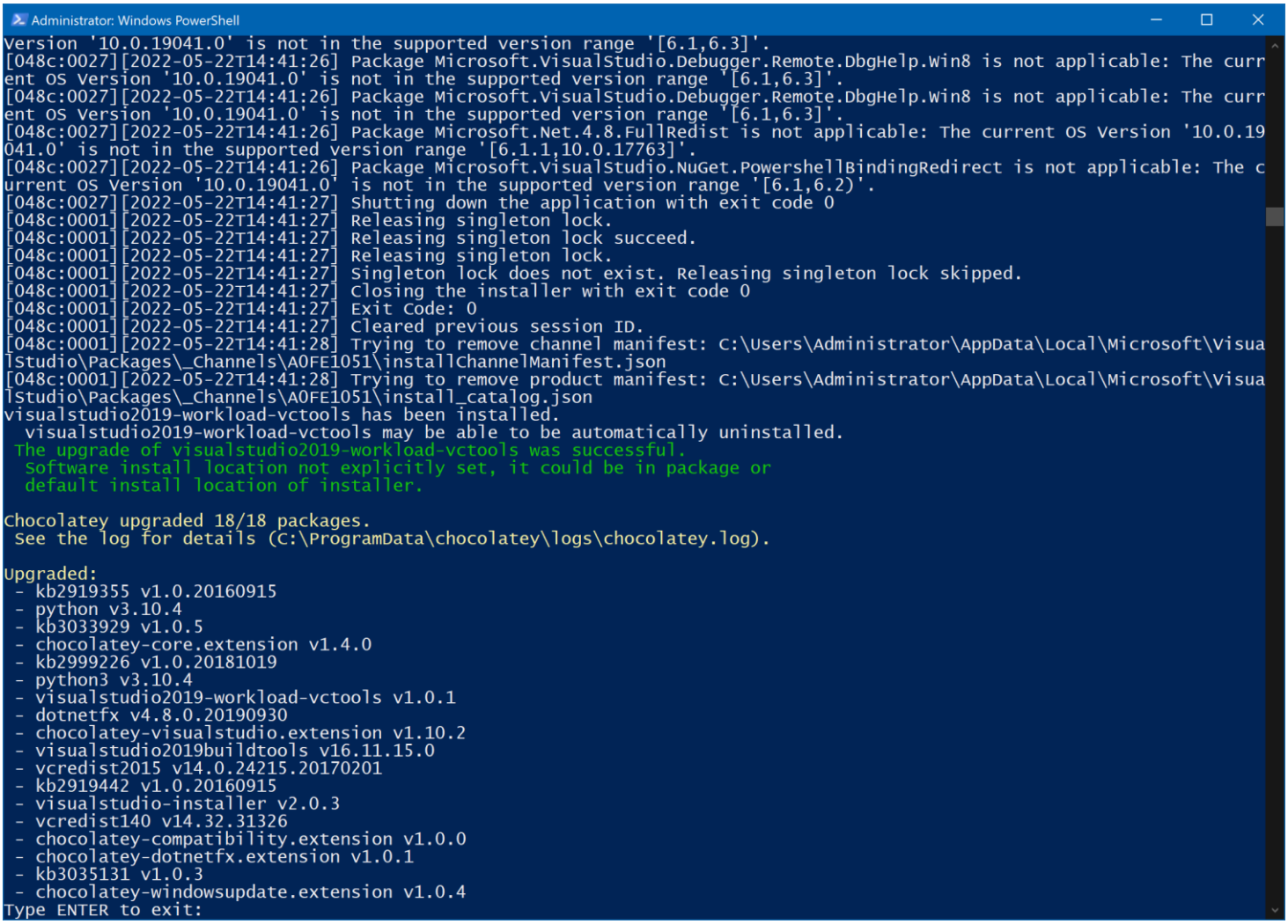


*Figure 50 - Completing the installation of the additional tools of "Node.js"*

## 4. Node.js Modules

Installing modules can be done in two ways: using the "npm rebuild" command or the "npm install" command. With the "npm rebuild" command, modules are "rebuilt" in order to be compatible with the current version of "Node.js" that is installed on the system. The duration of this action is estimated to be between two (2) and five (5) minutes. On the other hand, using "npm install", the modules are re-installed based on the contents of the "package.json" file located inside the application folder. This process is also estimated to take between two (2) and five (5) minutes. Between these two commands, "npm install" is recommended, because a "clean" installation of the modules is performed and therefore the possibility of errors is reduced.

Similarly to the above-mentioned installation modes, the project files include two folders: the "Prototype (rebuild)" and the "Prototype (install)". Below, both installation modes will be explained in the corresponding folders.

## 4.1. The "npm rebuild" command

Within the "Prototype (rebuild)" folder, use the "Shift" + right-click combination to bring up the "Open PowerShell window here" option (Figure 51). After selecting it, a "Windows PowerShell" window pops up in which we type the command "npm rebuild" (Figure 52). As mentioned above, the process takes between two (2) and five (5) minutes and from its results, visible in Figure 53, we ignore the lines starting with "npm notice".

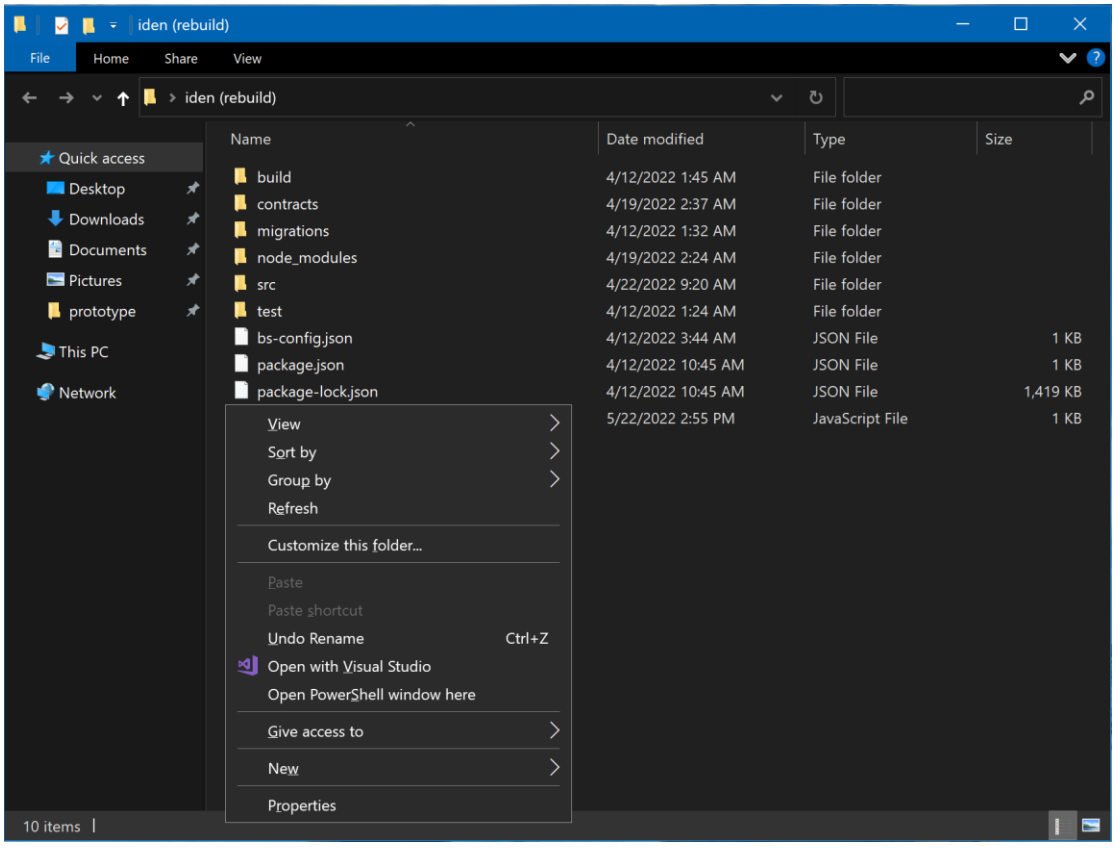


*Figure 51 - The "Prototype (rebuild)" folder*

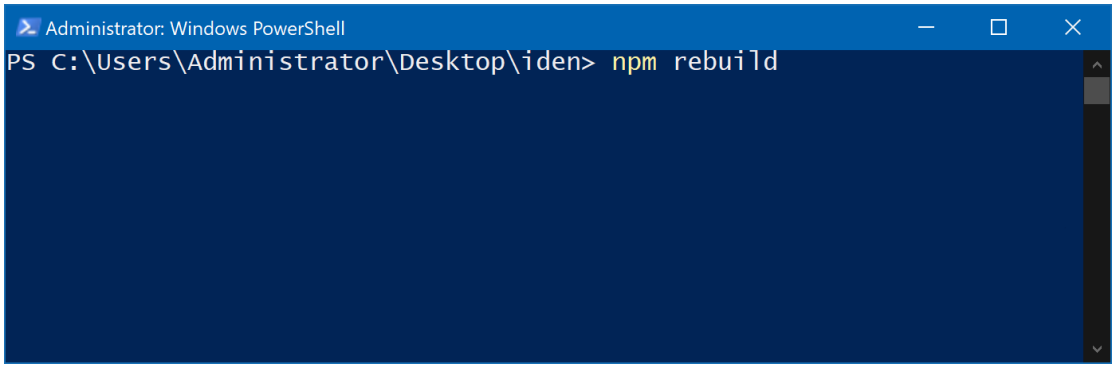


*Figure 52 - Running the "npm rebuild" command*

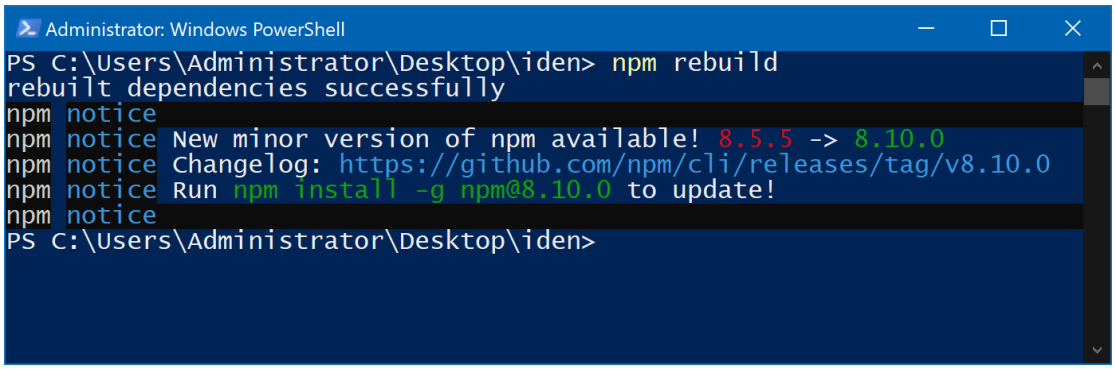


*Figure 53 - Results of the "npm rebuild" command*

## 4.2. The "npm install" command

The "npm install" command is executed in the same way. Using the combination of "Shift" + right click inside the "Prototype (install)" folder, the menu appears, in which we select "Open PowerShell window here". In the new "Windows PowerShell" window we type the command "npm install" (Figure 54). During the execution several messages will appear which we ignore. Also, once the process is complete (Figure 55) a few vulnerabilities are listed, which we similarly ignore.

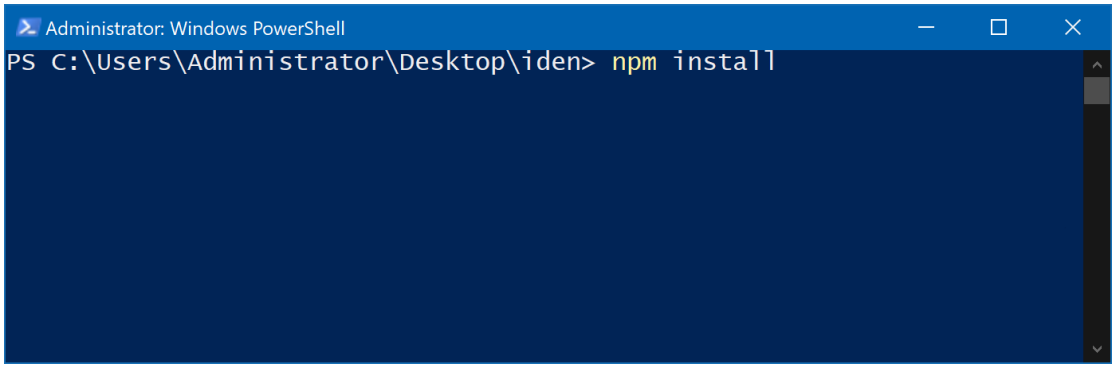


*Figure 54 - Running the command "npm install"*

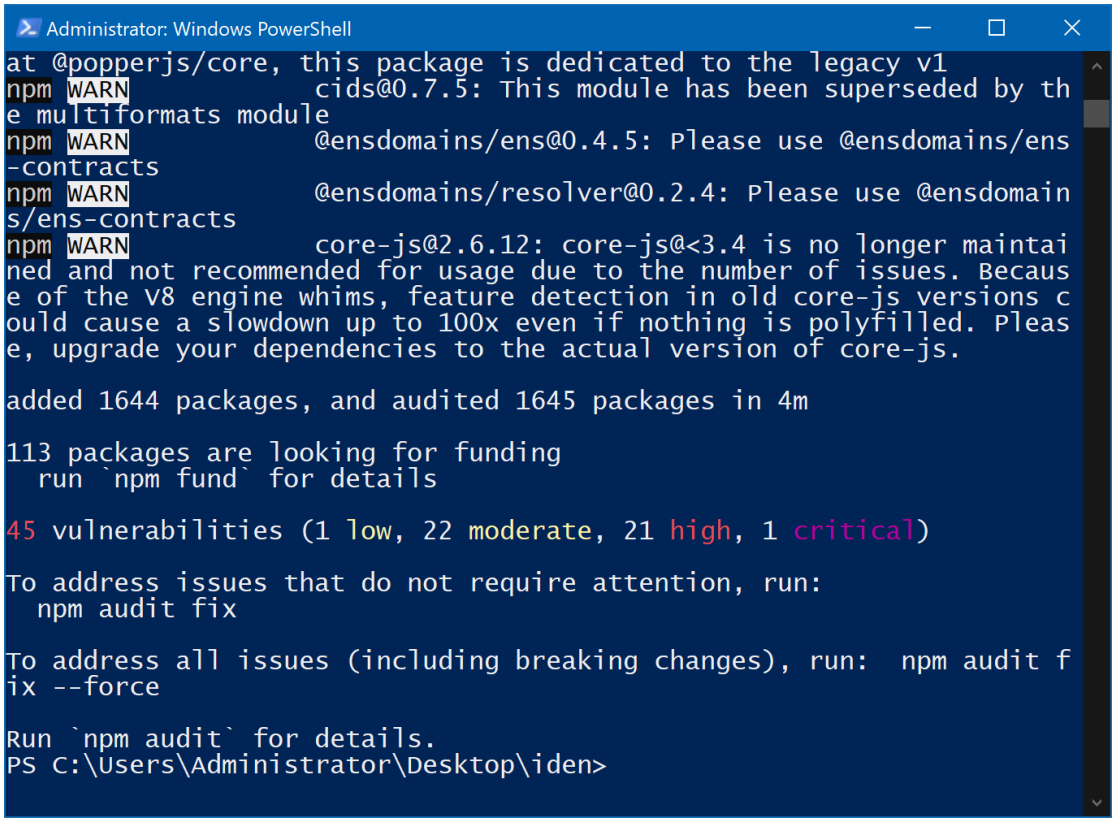


*Figure 55 - Results of the "npm install" command*

### 4.5.2 Application Execution Guide

As the application consists of three roles (issuer, citizen and verifier), it is necessary to add the corresponding "wallet" accounts in "Metamask". Due to the insertion of an existing "wallet" using the "Mnemonic" provided by "Ganache" (see page 67), the first of the ten accounts created during the installation has been automatically added. Therefore, two more wallet accounts should be added to Metamask. After connecting to the "Ganache" local network via Metamask (see pages 68, 69, 70), click on the circle with the blue outline at the top right of the wallet management interface and then on "Import Account" (Figure 56). Next, the "Private Key" account import page appears (Figure 57).

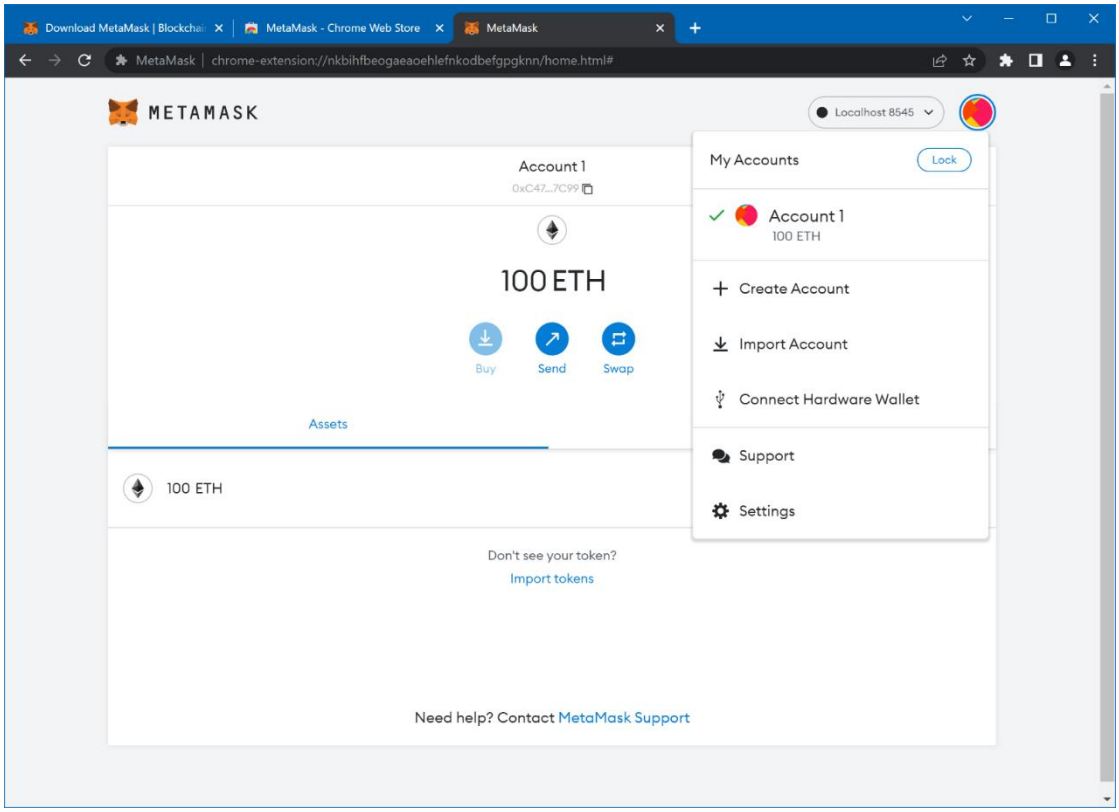


*Figure 56 - Wallet account management menu in Metamask*

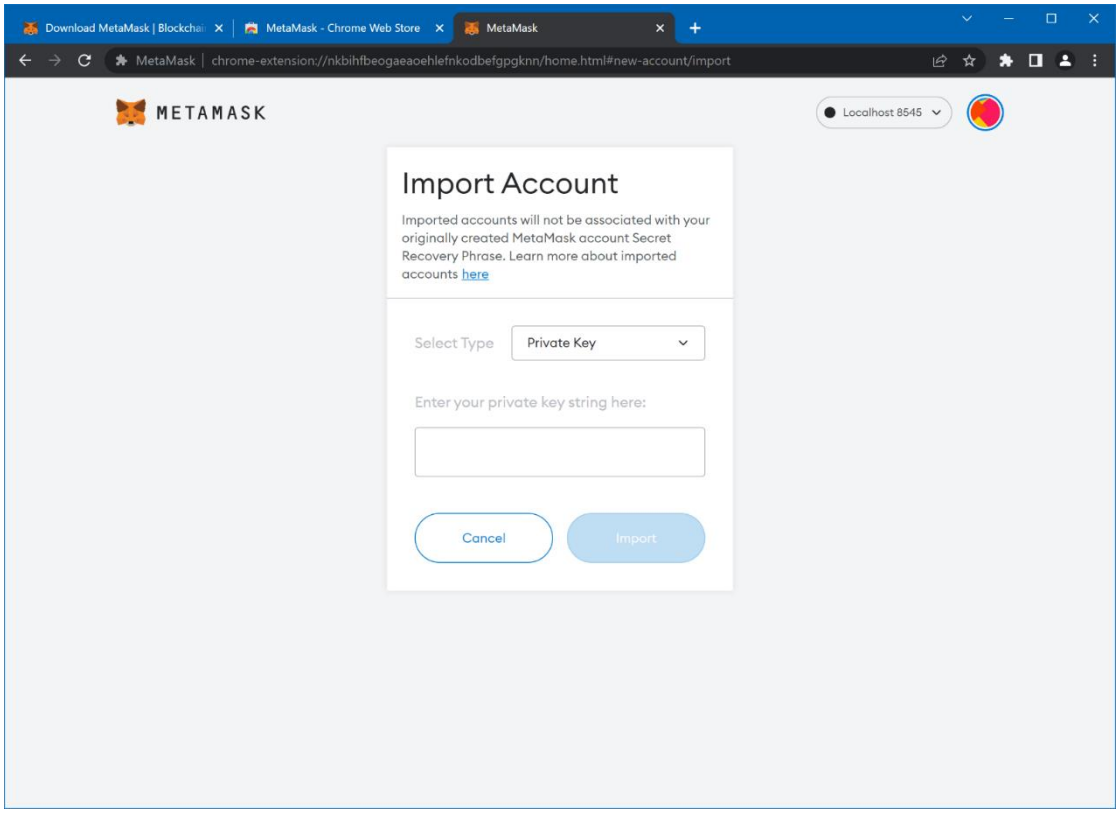


*Figure 57 - Account import via private key page*

Finally, we return to the "Ganache" Blockchain management window (see page 55) and click on the key icon located on the right of the account with "Index" 1 (the first automatically added account has Index 0). Then, the account information window (Figure 58) appears, from which we copy the alphanumeric corresponding to the "Private Key", paste it into the empty field in Figure 57 and click on the "Import" button.

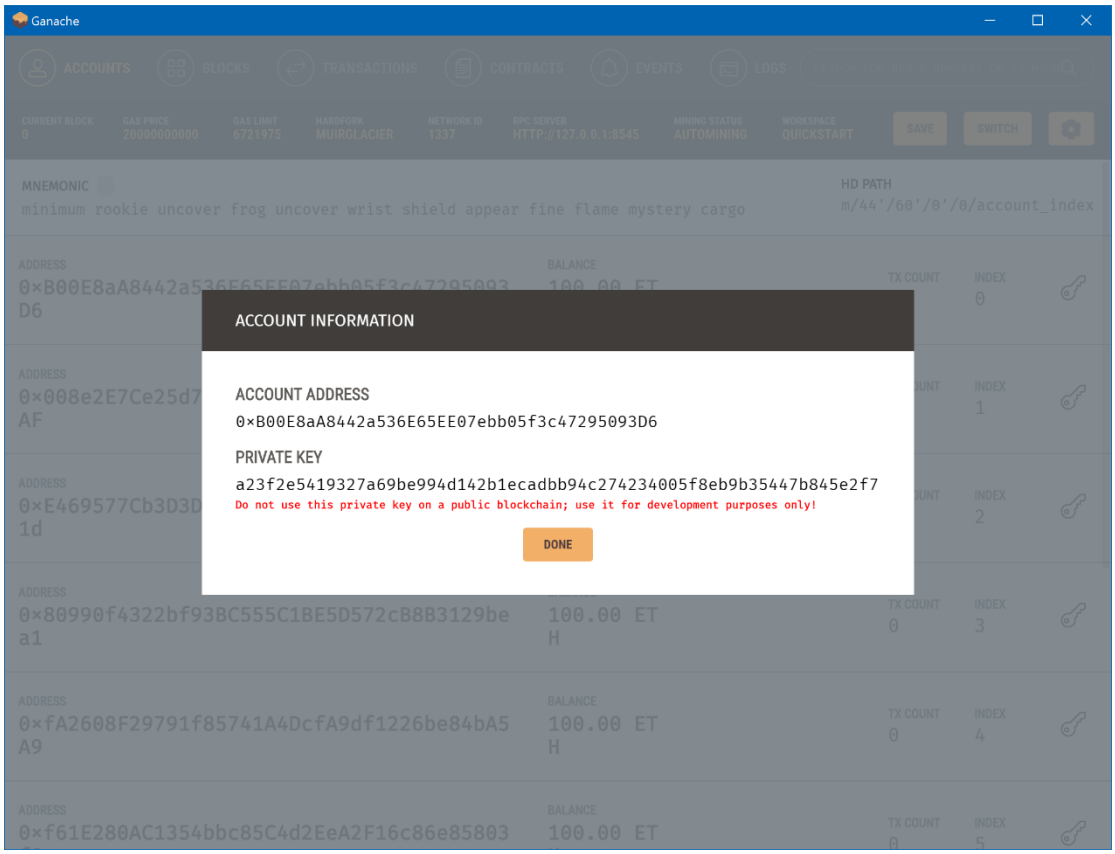


*Figure 58 - Account information window in "Ganache"*

We follow the same procedure for the next "Ganache" account (with Index 2), so that the Metamask account management menu looks like the one in Figure 59.

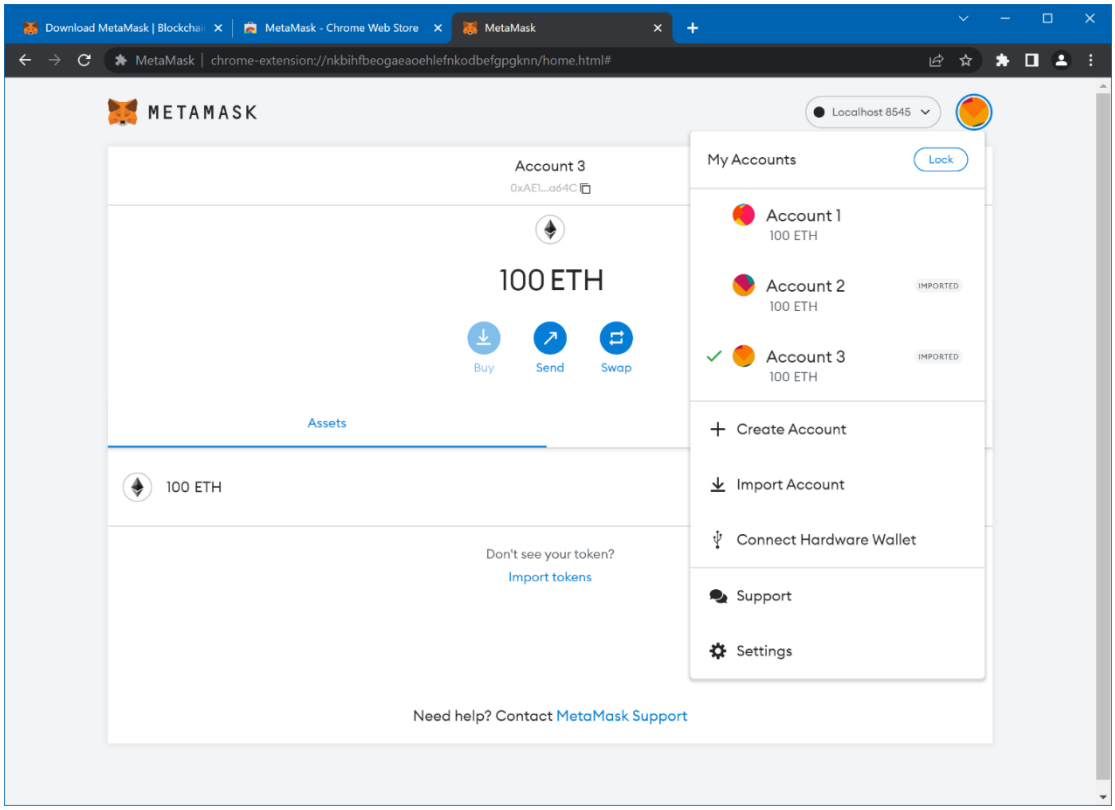


*Figure 59 - Wallet account management menu in Metamask with the three accounts connected*

Next, we open a Windows PowerShell window in the application folder (see page 68), after one of the two commands has been completed: “npm rebuild” and “npm install” (see pages 68, 69, 70). As seen in Figure 60, we type the command “node_modules/.bin/truffle migrate --reset” to deploy the smart contract code to the local Blockchain and set it ready to run.

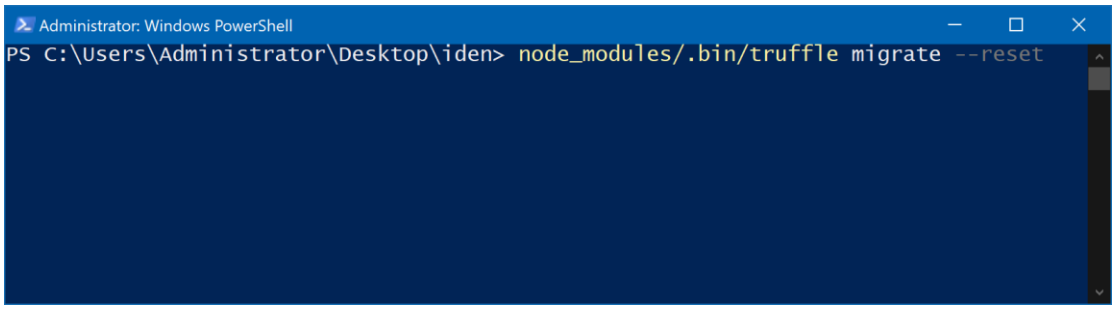


*Figure 60 - Running the command “node_modules/.bin/truffle migrate --reset”*

In case the error in Figure 61 occurs, we go to the Windows search and open a new Windows PowerShell window with administrator privileges. There, we enter and

execute the command “Set-ExecutionPolicy RemoteSigned” (Figure 62) and then type the character “A” as prompted (Figure 63).

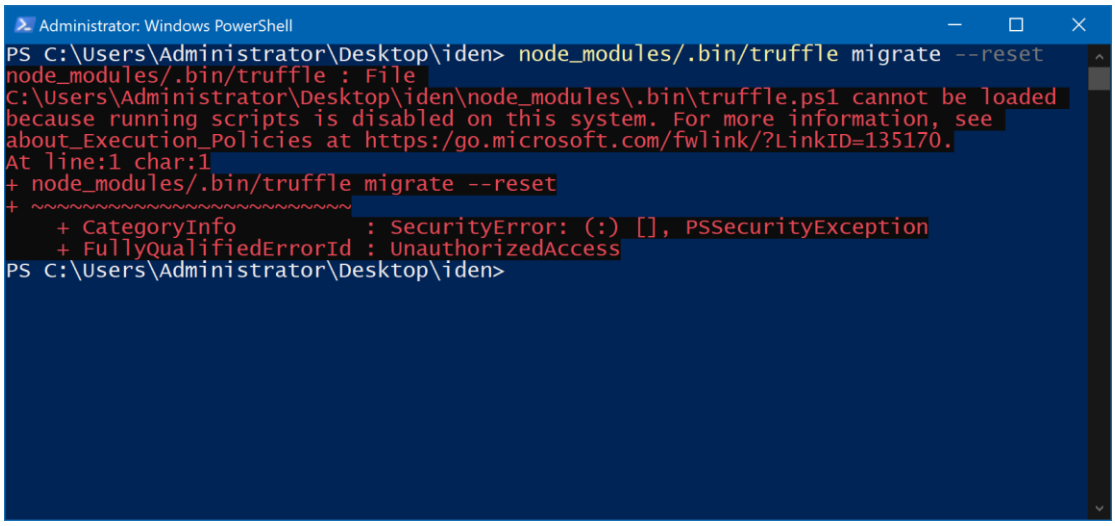


*Figure 61 - Error when running the command “node_modules/.bin/truffle migrate --reset”*

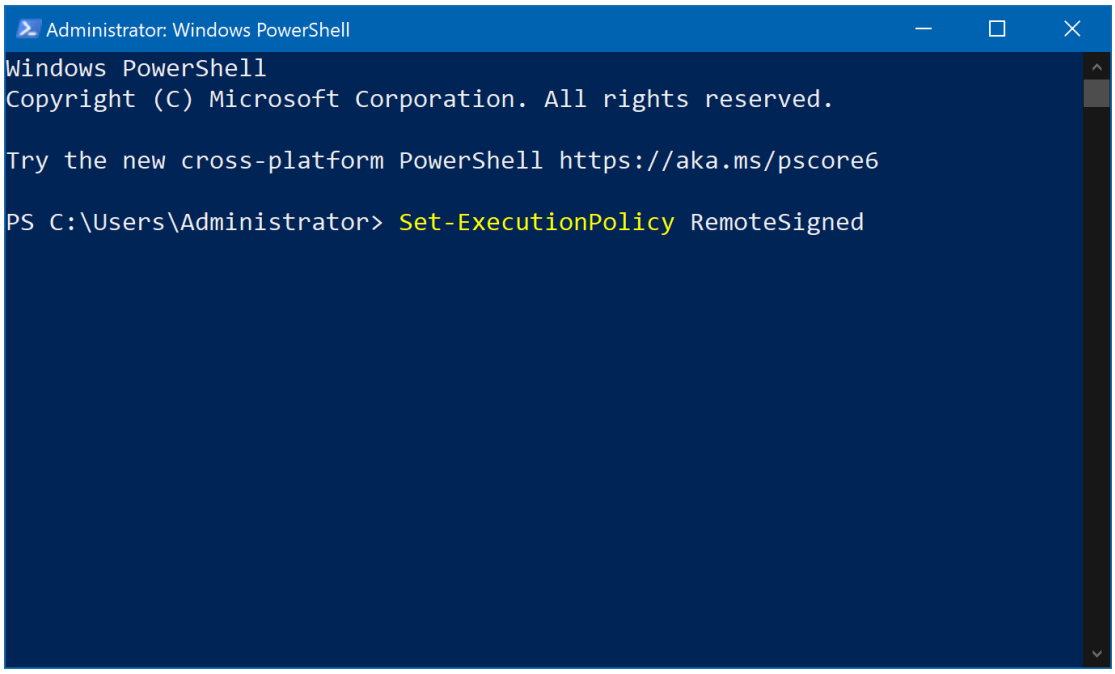


*Figure 62 - Running the “Set-ExecutionPolicy RemoteSigned” command (1/2)*

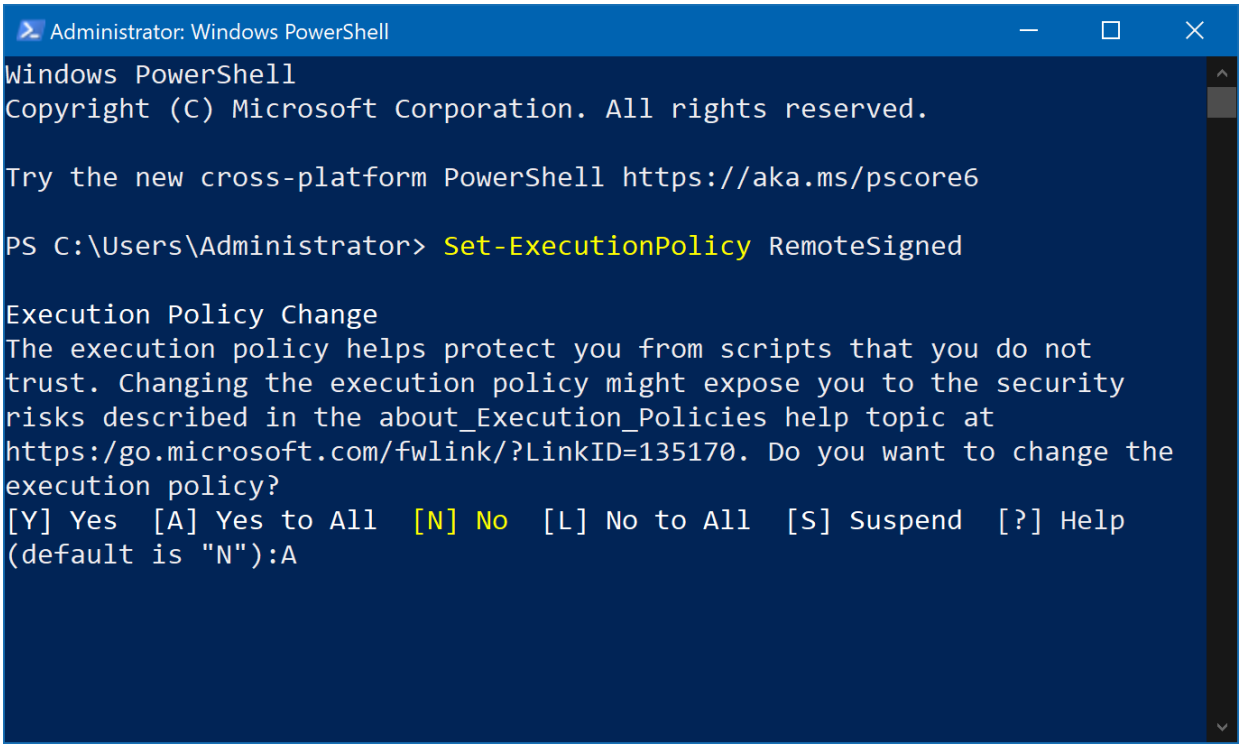


*Figure 63 - Running the "Set-ExecutionPolicy RemoteSigned" command (2/2)*

After the process is complete, return to the previous Windows PowerShell window and run the "node_modules/.bin/truffle migrate --reset" command again (Figure 64).

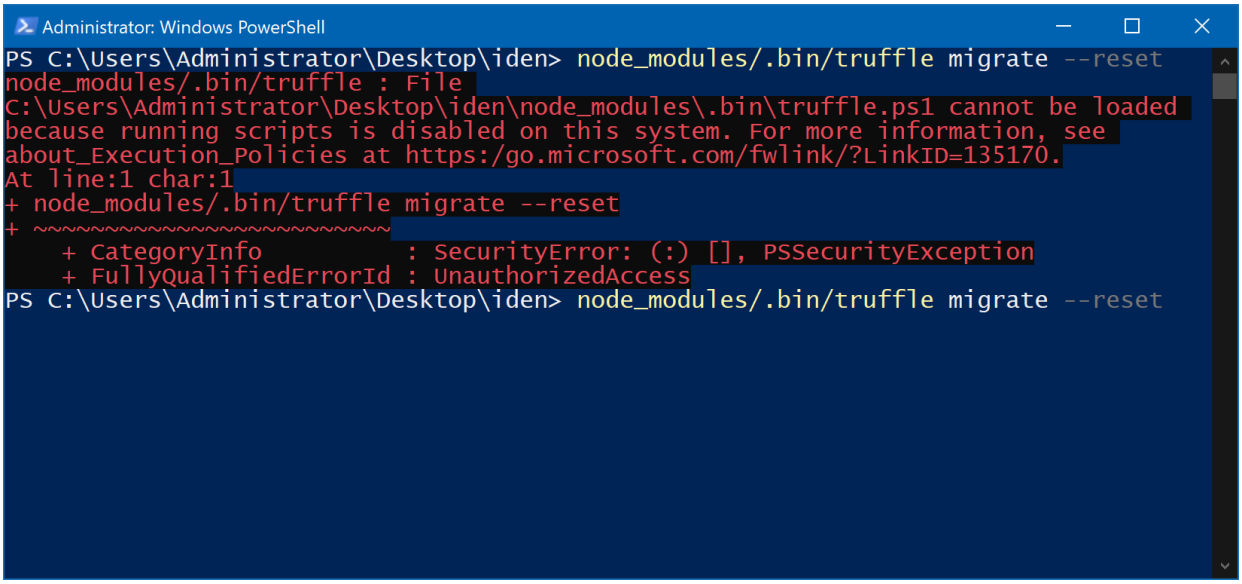


*Figure 64 - Rerunning the command "node_modules/.bin/truffle migrate --reset"*

Once the above command is executed, the compilation of the smart contracts into "bytecode" starts (see page 50) (Figure 65), and once the process is completed, for each smart contract stored on the Blockchain, information such as its address, the account that was charged for the transaction (in "Ganache" it is the account with "Index" 0), the account balance, and the total cost to perform the transaction (see pages 50, 51) are listed (Figure 66).

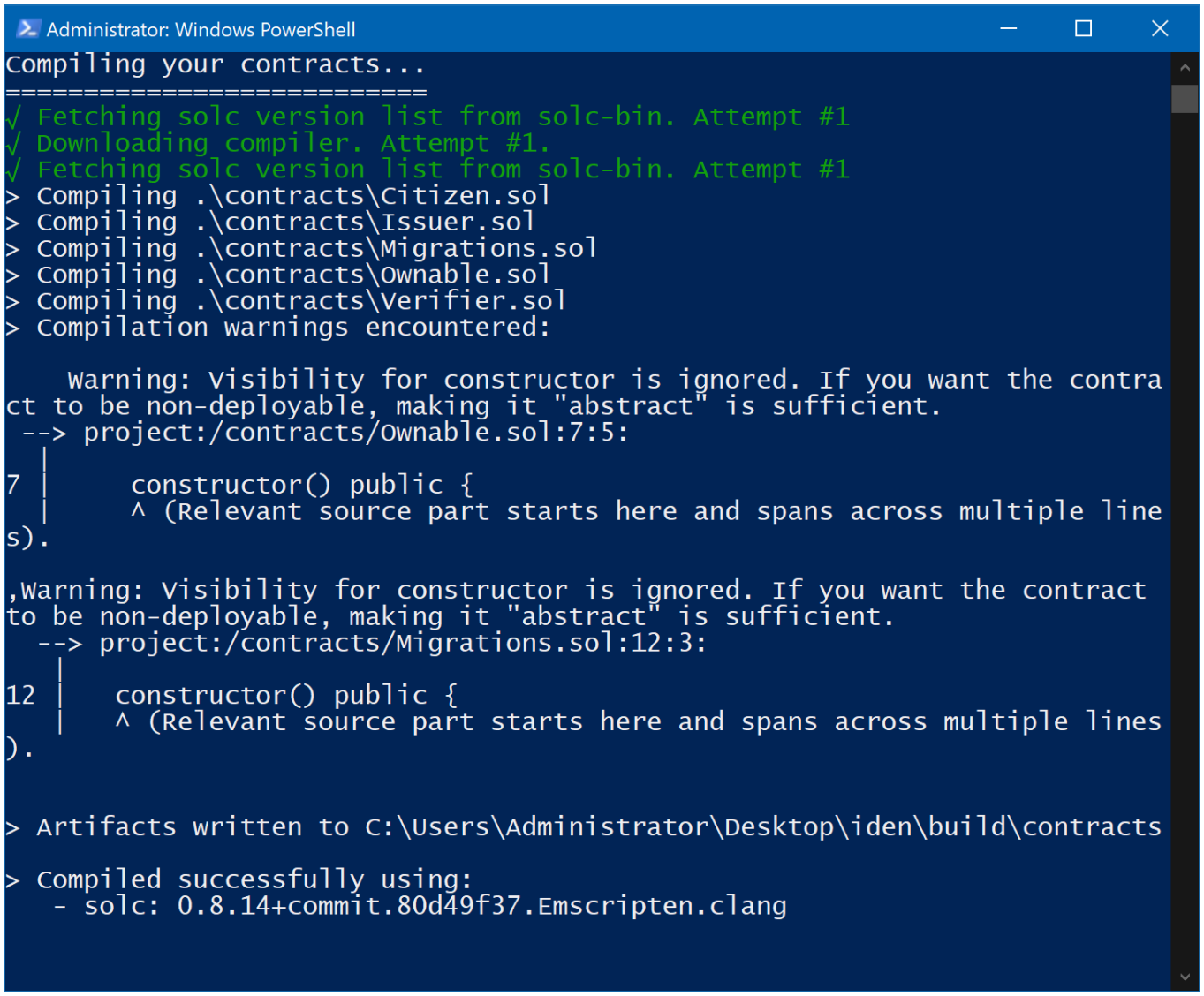


*Figure 65 - The process of compiling the code of smart contracts into "bytecode"*

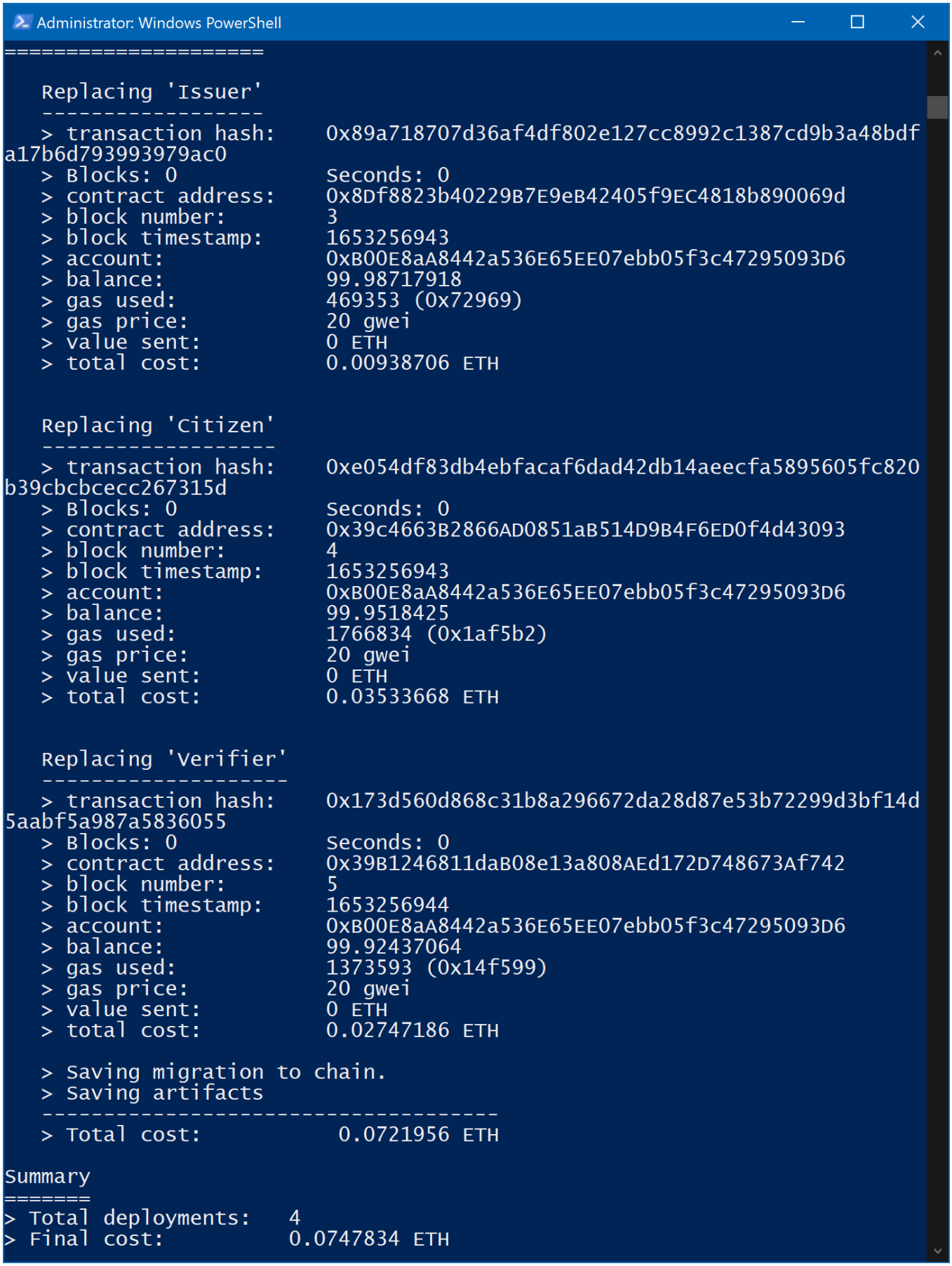


*Figure 66 - Completing the compilation process and presenting information for each smart contract*

Finally, in order for the application to run and be accessible from the web browser, one more command must be executed within the application folder using Windows PowerShell (see page 68). This command is “npm run dev” (Figure 67) and its purpose is to start a local server on which the application code will be executed alongside Node.js and its modules. Figure 68 shows the server being run, using the Node.js module “lite-server”, which should remain active throughout the execution of the application.

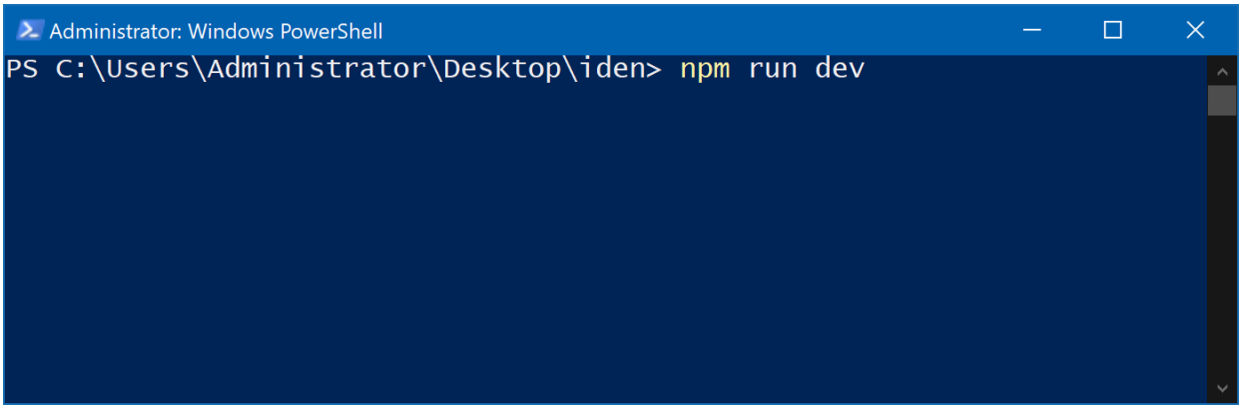


*Figure 67 - Running the command “npm run dev”*

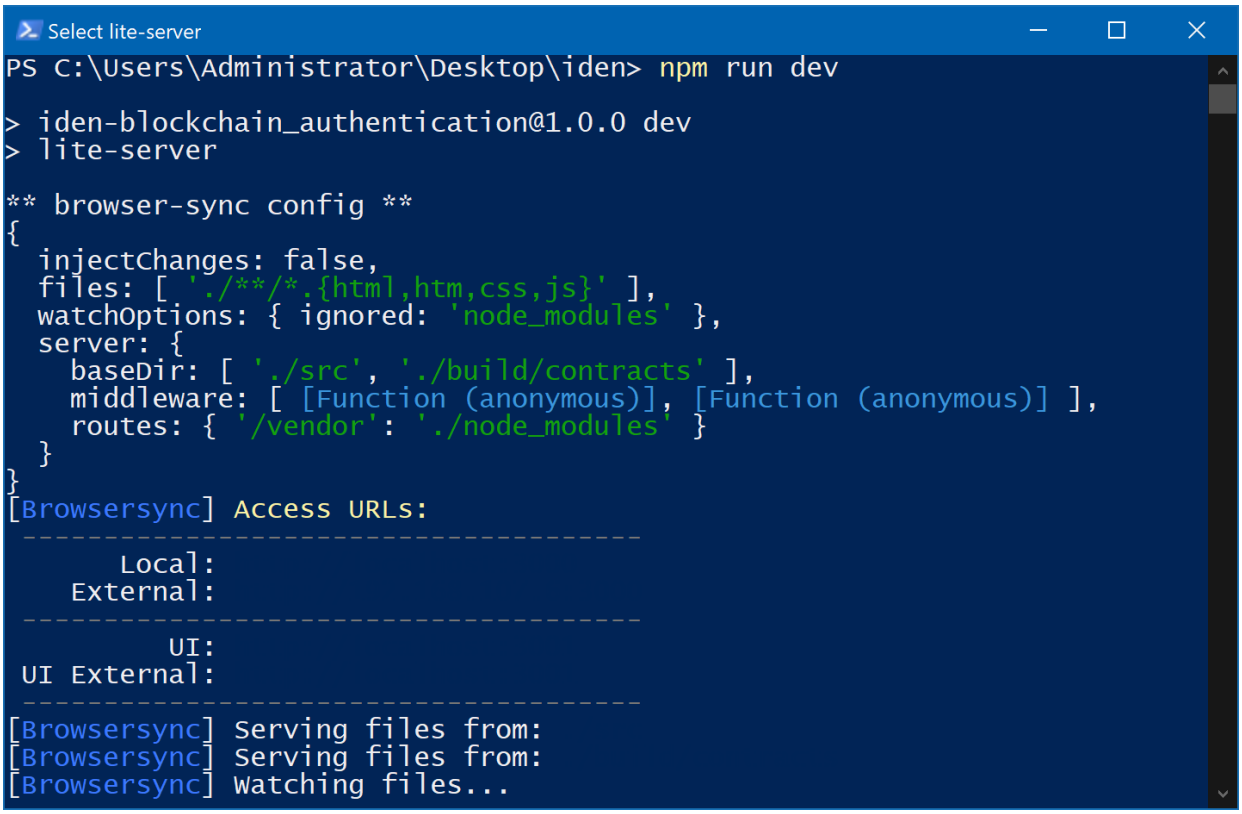


*Figure 68 - Example of the module “lite-server “*

## 4.6. Use case description

As can be seen from the diagrams in Chapter 4.2 “Design and Modelling”, the application is based on the use case in which a citizen, after having his/her identity registered on the Blockchain by an issuer, visits a hospital for an examination. Below will be presented the set of stages of the use case, which briefly are: the registration of the identity of the citizen and the hospital on the Blockchain, the sending of an identity disclosure request by the hospital to the citizen, the viewing of the request and the response by the citizen, the verification of the citizen's details by the hospital through the issuer's smart contract, and finally, the addition/creation of a medical history for the citizen by the hospital. For a better understanding of the content of the current chapter, along with the third person singular, the first person plural will be used.

Having added the remaining wallet accounts to Metamask (see pages 70, 71, 72, 73) and activated the local server (see page 78), the application is accessed via the “http://localhost:3000/”. It is essential to confirm that the selected account is the one named “Account 1” (hereinafter “issuer”) (Figure 69), as the accounts “Account 2” (hereinafter “citizen”) and “Account 3” (hereinafter “hospital”) will be used for the other two roles.

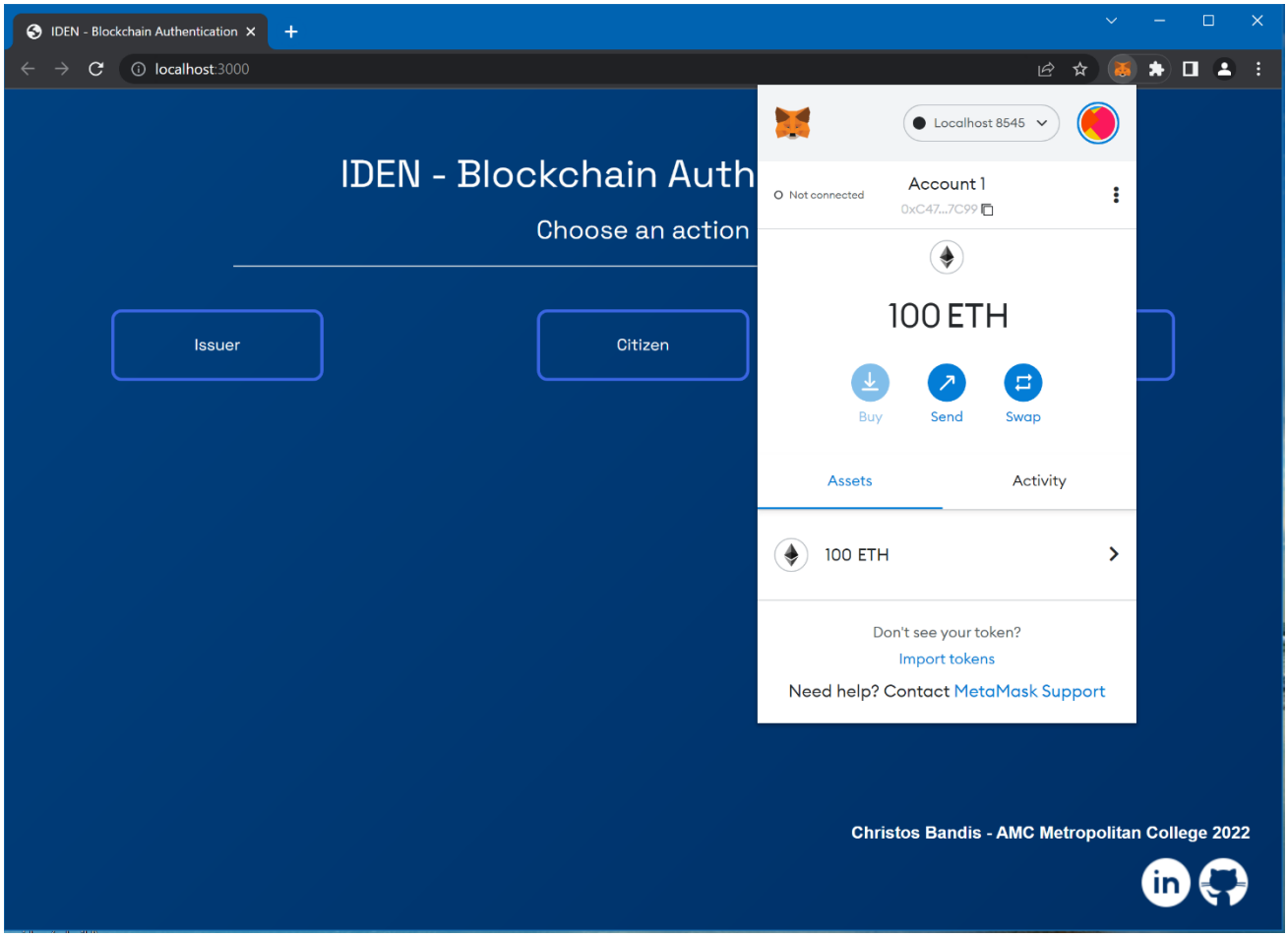


*Figure 69 - Confirmation that the selected account is "Account 1"*

First, clicking on the "Issuer" button will display the page for selecting the type of identity to be registered. Once the page has finished loading, the Metamask window pops up, which asks for the account to be connected to the application (Figure 70). Once logged in, we click on the "Issue a citizen's ID" link.

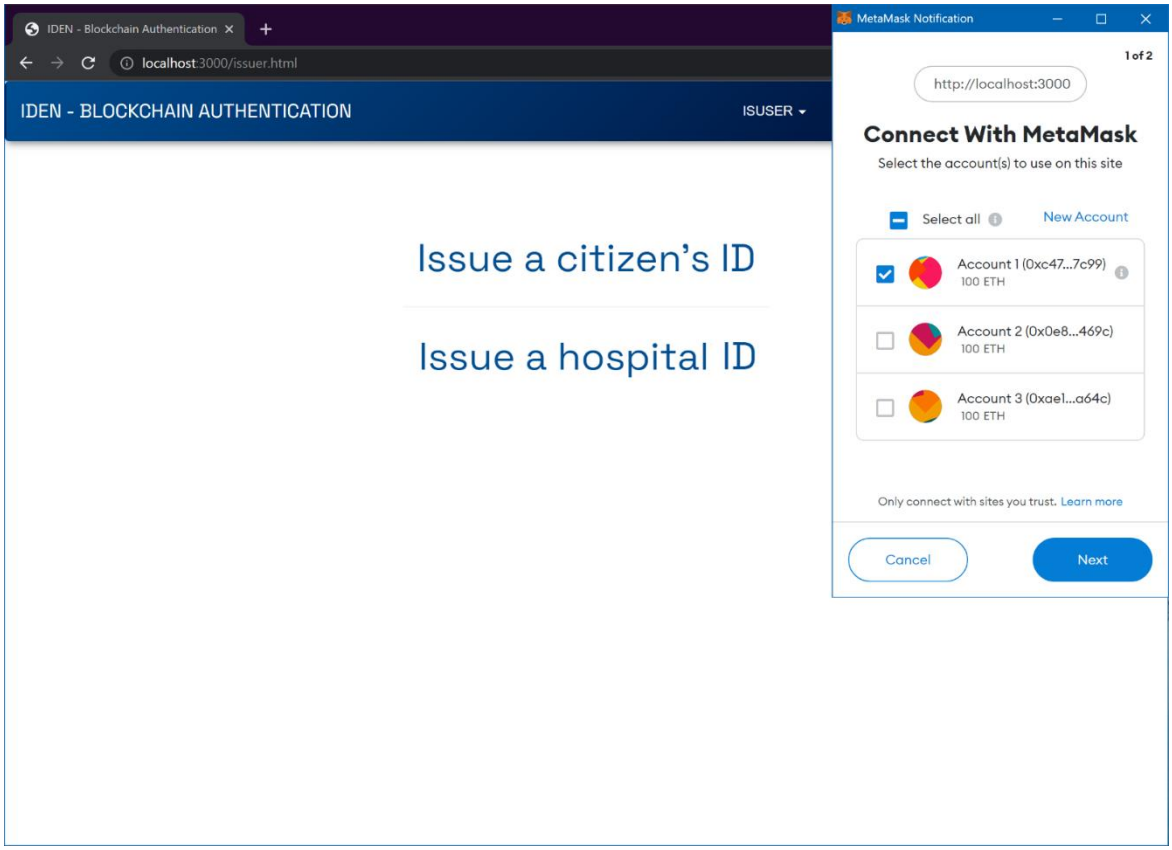


*Figure 70 - Metamask's window for connecting an account to the application*

First, from the Ganache administration window (see page 55), copy the alphanumeric corresponding to the address of the account with Index 1, i.e. citizen, and paste it into the first field of the form in Figure 71. Then, we fill in the remaining fields with the citizen's identification data and press the "Submit" button.

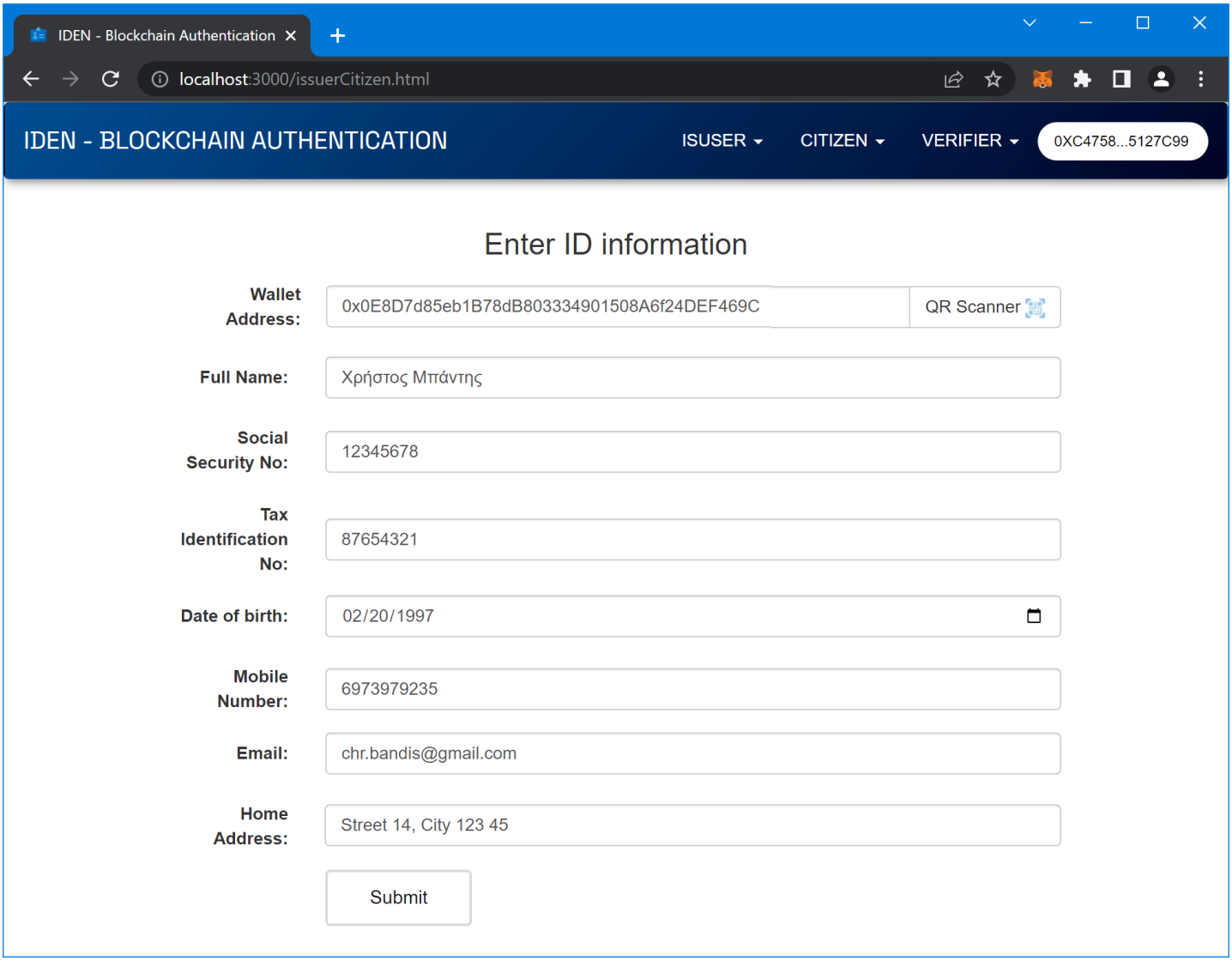


*Figure 71 - Citizen's ID registration form*

After we click on the "Submit" button, three transaction approval windows (Figure 72) appear in order, each for a different purpose. The first transaction involves storing the data in the citizen's smart contract and matching it with the citizen's address; the second transaction stores, in the citizen's identity data, the hash (see page 9) of the first transaction (this is done in a second time, as the first transaction has to be executed first and then its hash has to be calculated) and finally, the reason for the third transaction is to match the citizen's address with the generated hash of the identity registration transaction and store them in the issuer's smart contract, for the verification of the validity of the citizen's identity by the verifier.

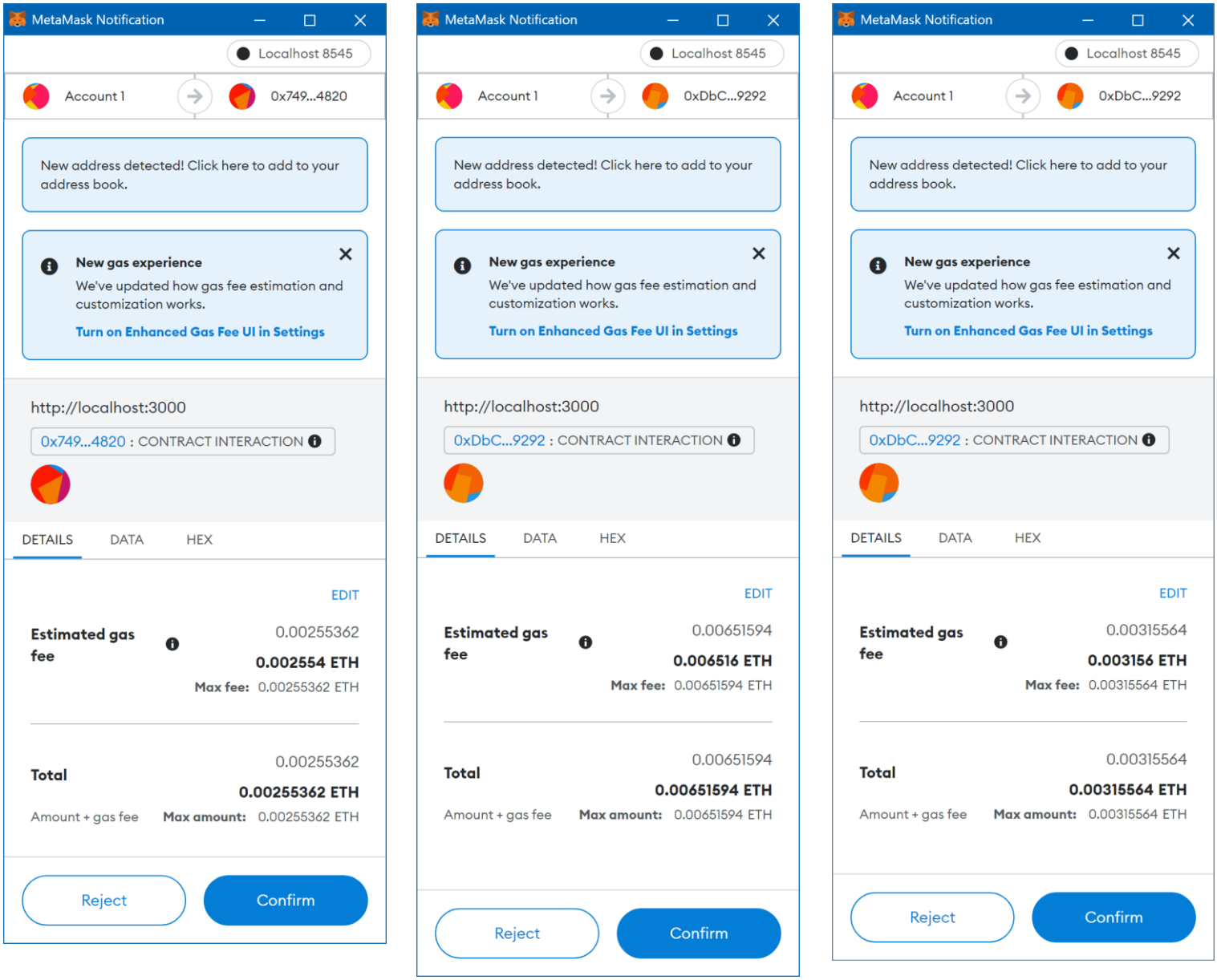


*Figure 72 - Transaction approval windows when registering an identity on the Blockchain*

The same procedure is found when registering the hospital's identity. Returning to the page for selecting the type of identity to be registered (Figure 70), click on the "Issue a hospital ID" link. From the Ganache administration window, copy the alphanumeric corresponding to the address of the account with Index 2, i.e. the hospital, and paste it into the first field of the form in Figure 73. Then, we enter the name of the hospital and press the "Submit" button. As in the citizen's identity registration, the three transaction approval windows are displayed, which perform the corresponding actions.

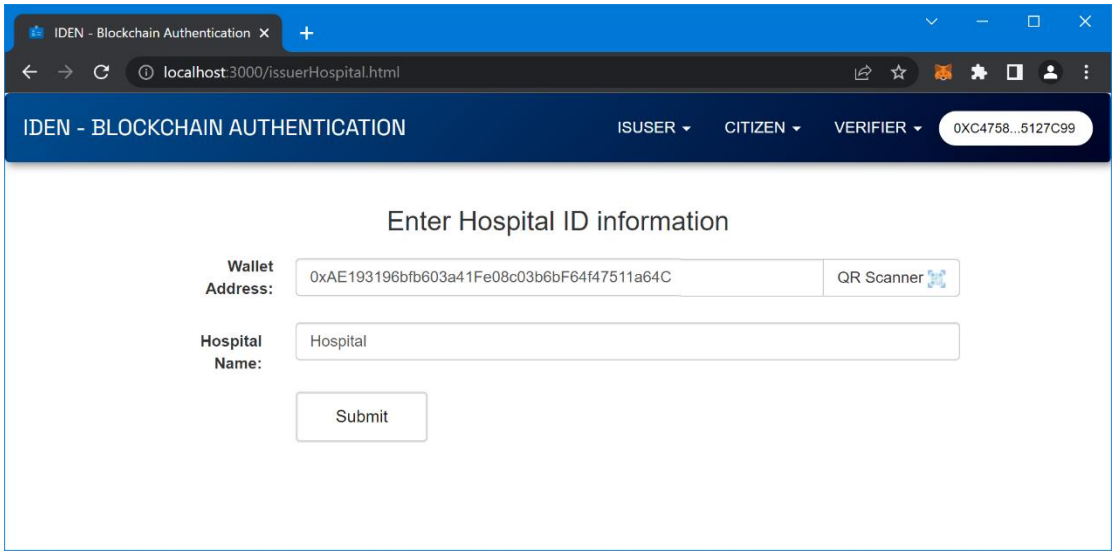


*Figure 73 - Hospital ID registration form*

Moving to the citizen role, we return to the home page of the application by clicking on the logo in the upper left corner, select the second account (citizen) in Metamask and click "Connect" (Figure 74). Once connected, we click on the "Citizen" button on the home page to be taken to the ID page corresponding to this account (Figure 75).

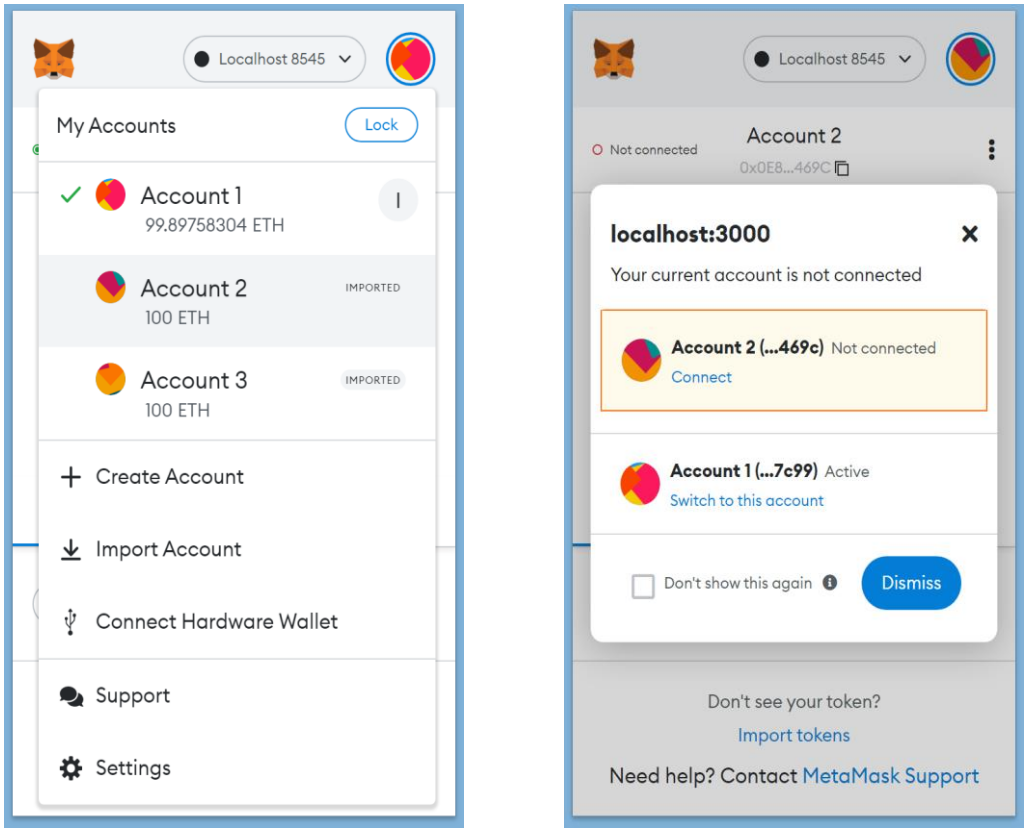


*Figure 74 - Connecting the second account (citizen) to the application*

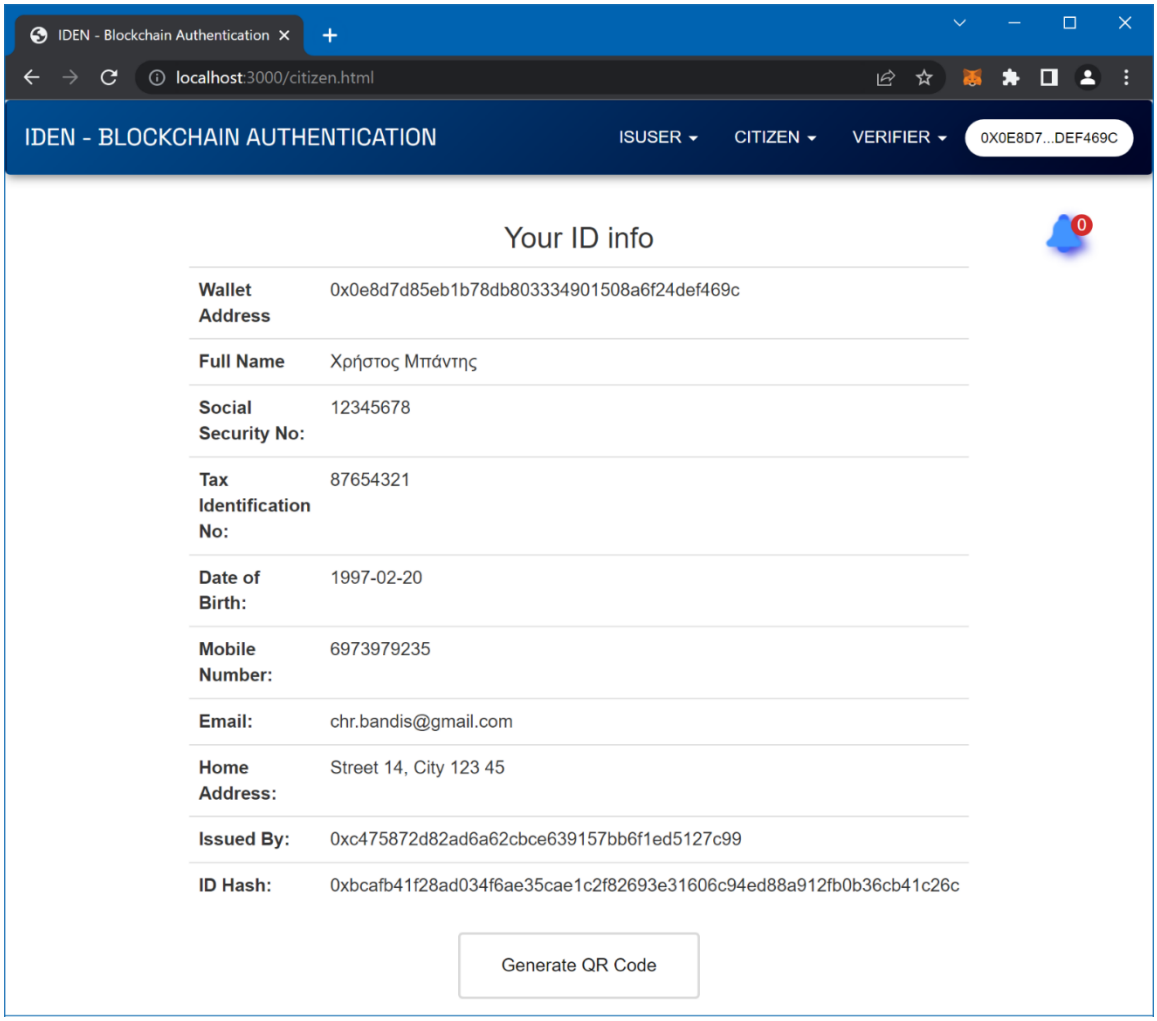


*Figure 75 - Page with the citizen's identity information*

On the page depicted above, in addition to the citizen's details, we can see the address of the issuer's account that registered the ID (Issued By), as well as the hash of the transaction (ID Hash). At the bottom of the page is the “Generate QR Code” button, which serves to generate a QR code corresponding to the citizen's address, for use when adding/generating a medical history, when the address is required to be entered. Also, at the top right of the page, there is a bell-shaped icon, the purpose of which is to display the number of pending identity notification requests.

Continuing in the role of the verifier-hospital, the process of linking the account (Account 3) is the same as for the citizen (see page 84). After returning to the home page of the application, we click on the “Verifier” button and are taken to the citizen’s identity information notification request page (Figure 76). Here, the hospital account user enters the address in the first field, either by typing it in or by scanning the QR code provided by the citizen. He then selects the details he wishes to verify and sends

the request to the citizen via transaction. It is worth noting that the "ID Hash" box remains selected, with no possibility of deselection, as it is the element on which the authentication of the data is based.

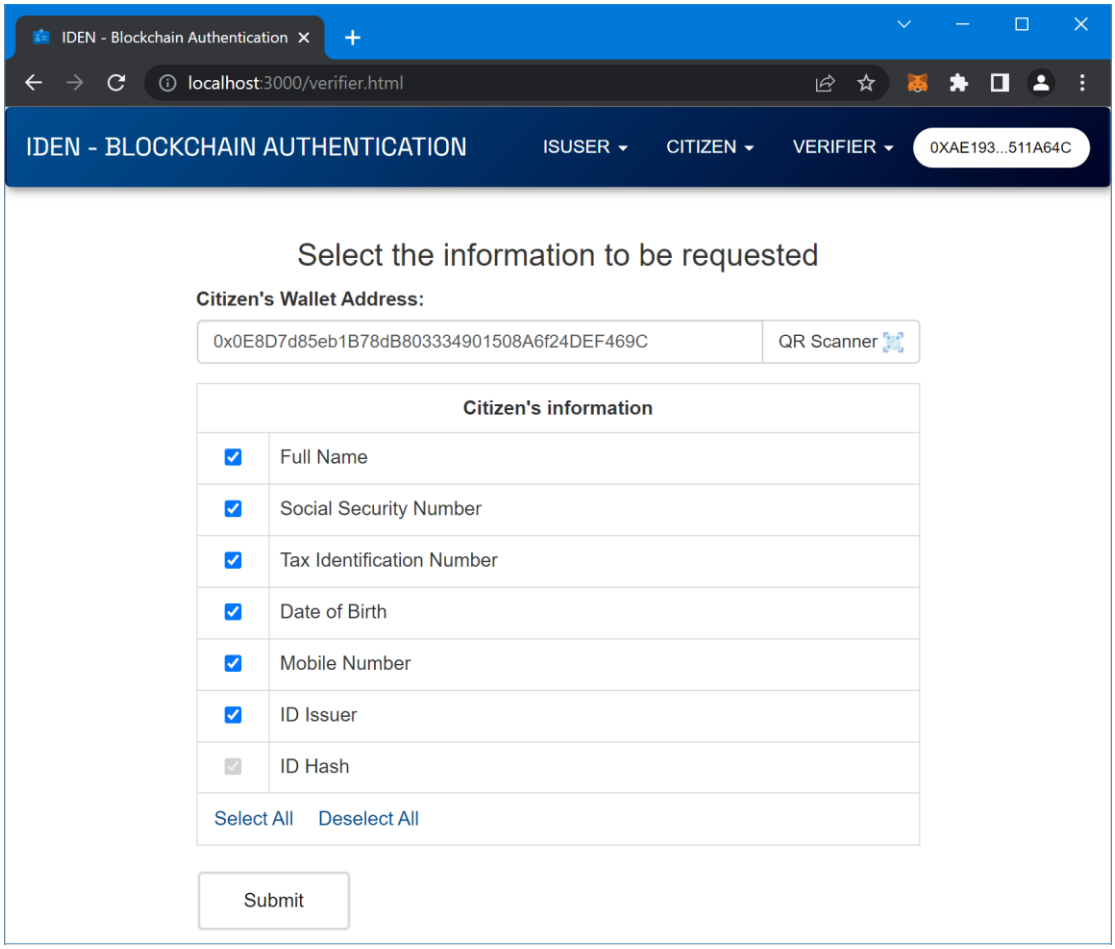


*Figure 76 - Citizen's identity information notification request page*

As soon as the transaction is completed, the notifications icon on the citizen's page informs him of the unanswered request (Figure 77). By clicking on it or using the application menu, the citizen goes to the requests page (Figure 78), which contains all the requests he has received.

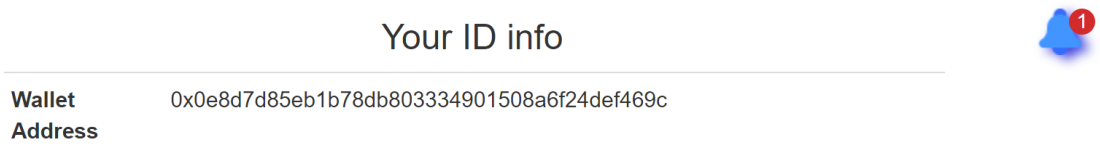


*Figure 77 - Notifying the citizen of the unanswered request via the notification icon*

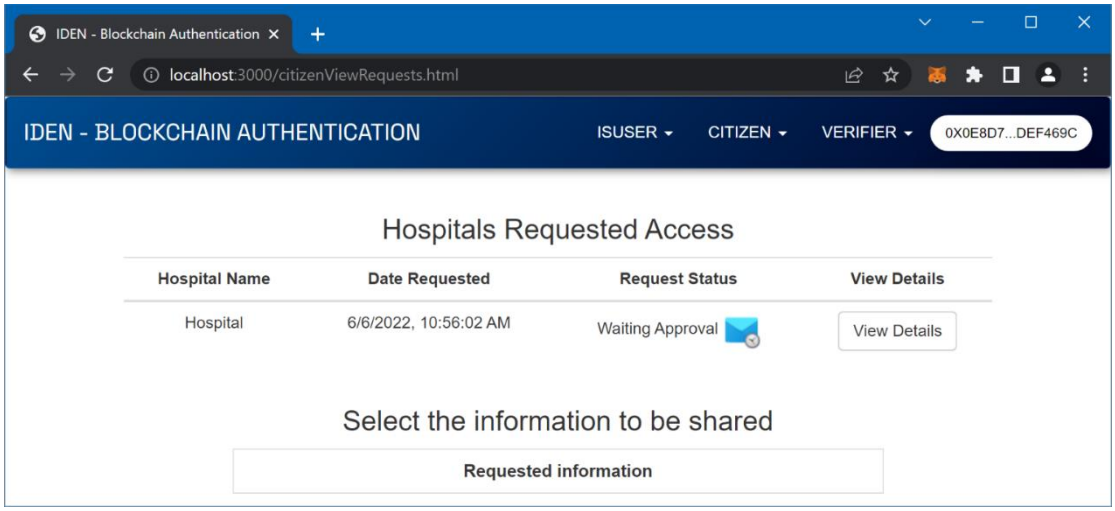


*Figure 78 - Citizen's requests page*

The contents of the request are displayed by clicking on the "View Details" button. The name and address of the applicant are listed at the beginning of the request details. The checkboxes in front of each requested item are the citizen's response, as selecting any box implies sharing the corresponding information with the hospital (Figure 79). After the citizen selects the items he wants to share, he sends them via a new transaction.

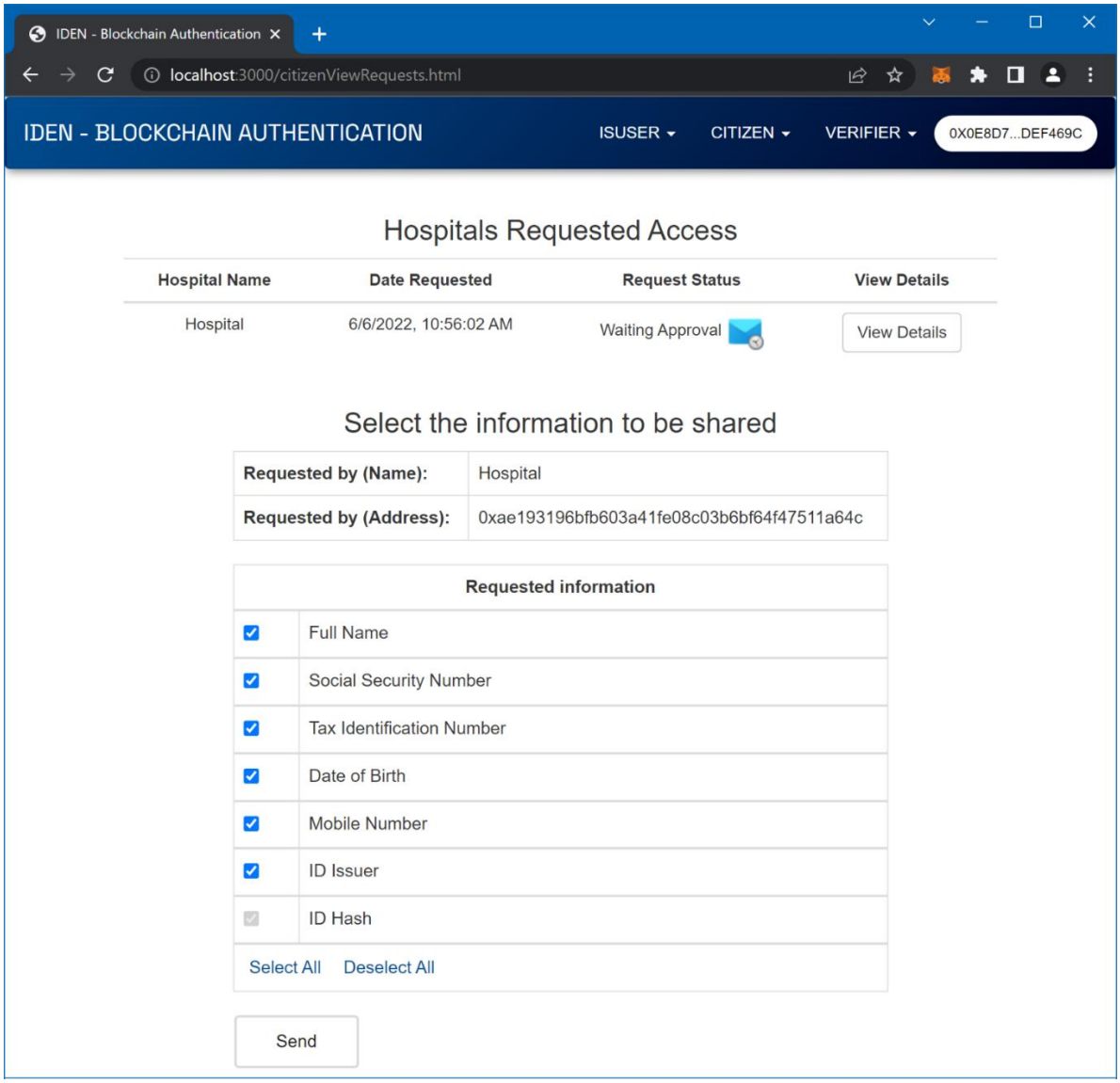


*Figure 79 - Selection of identity data to be shared by the citizen*

The user of the hospital's account, in turn, navigates through the menu to the "View Requests" page, where he enters in the search box the address for which he wishes to locate the requests and by pressing the "View Details" button, he sees the details shared by the citizen (Figure 80).

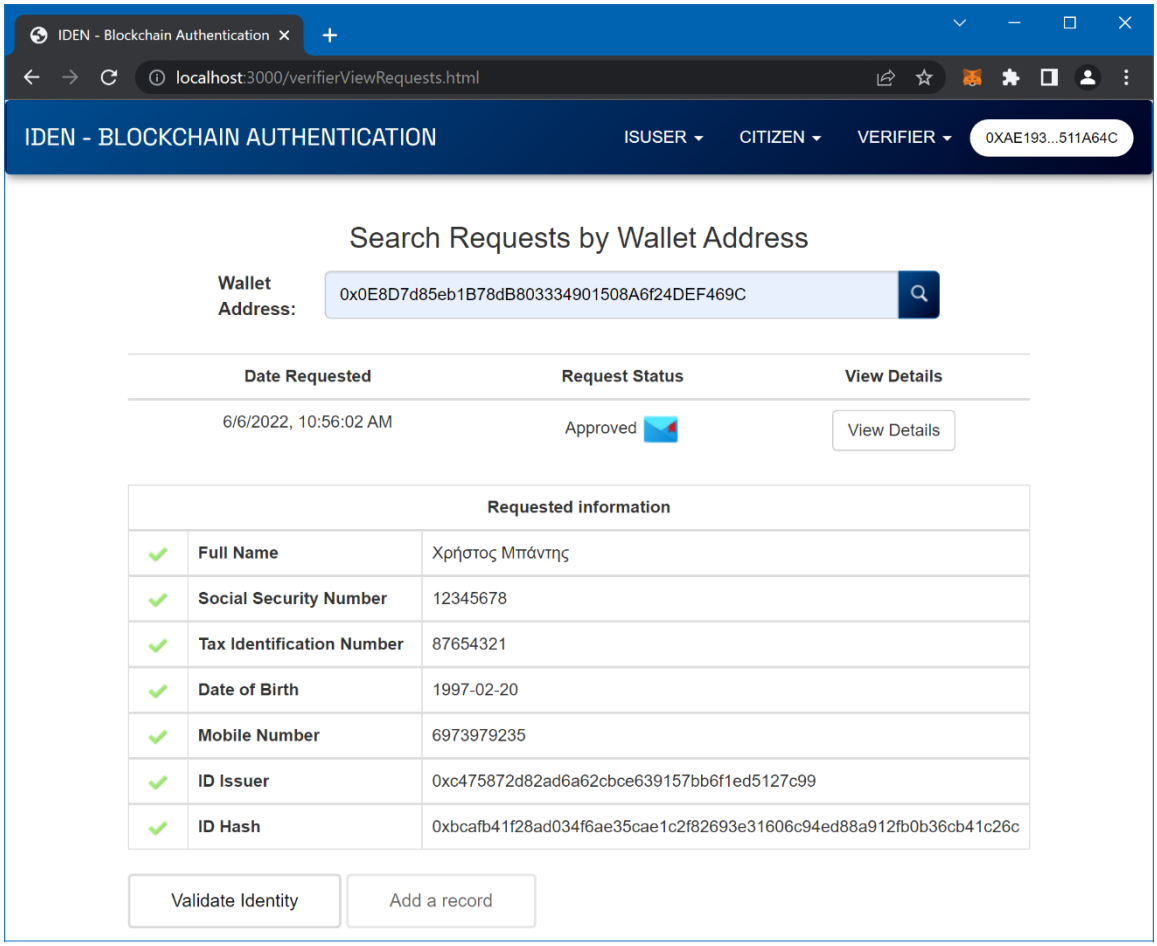


*Figure 80 - Details of an acceptable request for an identity notification*

In the image above, it is easy to see that the "Add a record" button is disabled. This is because it is not possible to perform any action unless the hospital verifies the validity of the information. The verification of the information is done with the "Validate Identity" button and the result is displayed as a message at the top of the page (Figure 81).

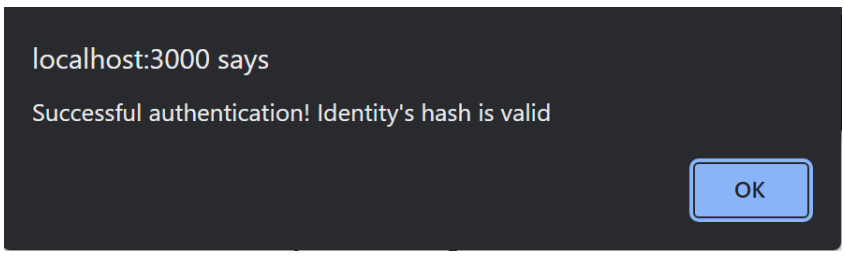


*Figure 81 - Successful validation message for citizen's identity information*

The user of the hospital account can then add/create a new medical record via the “Add a record” button. By clicking on this button, a window pops up containing a form to enter all the necessary information (Figure 82). Through this form, the user can name the record, select the type of treatment from a list of treatments, enter the prescription and the name of the supervising doctor, as well as enter a new appointment for review.

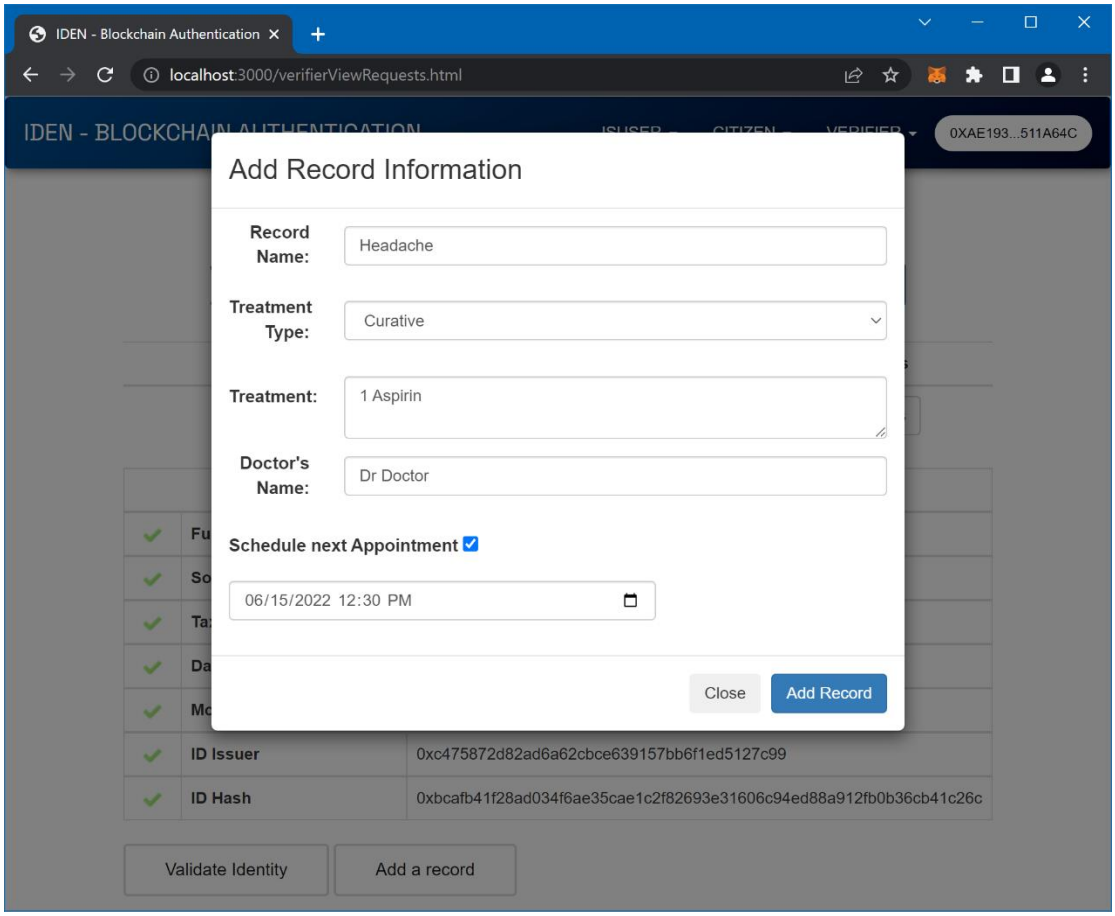


*Figure 82 - Add/Create medical history*

Upon completion, the new medical history is visible through the “View Medical Records” page of the citizen's menu (Figure 83), and also from the corresponding page in the verifier - hospital menu, after searching for the desired citizen address (Figure 84). The medical history page visible in the hospital, has two additional options, updating the record and completing the treatment.

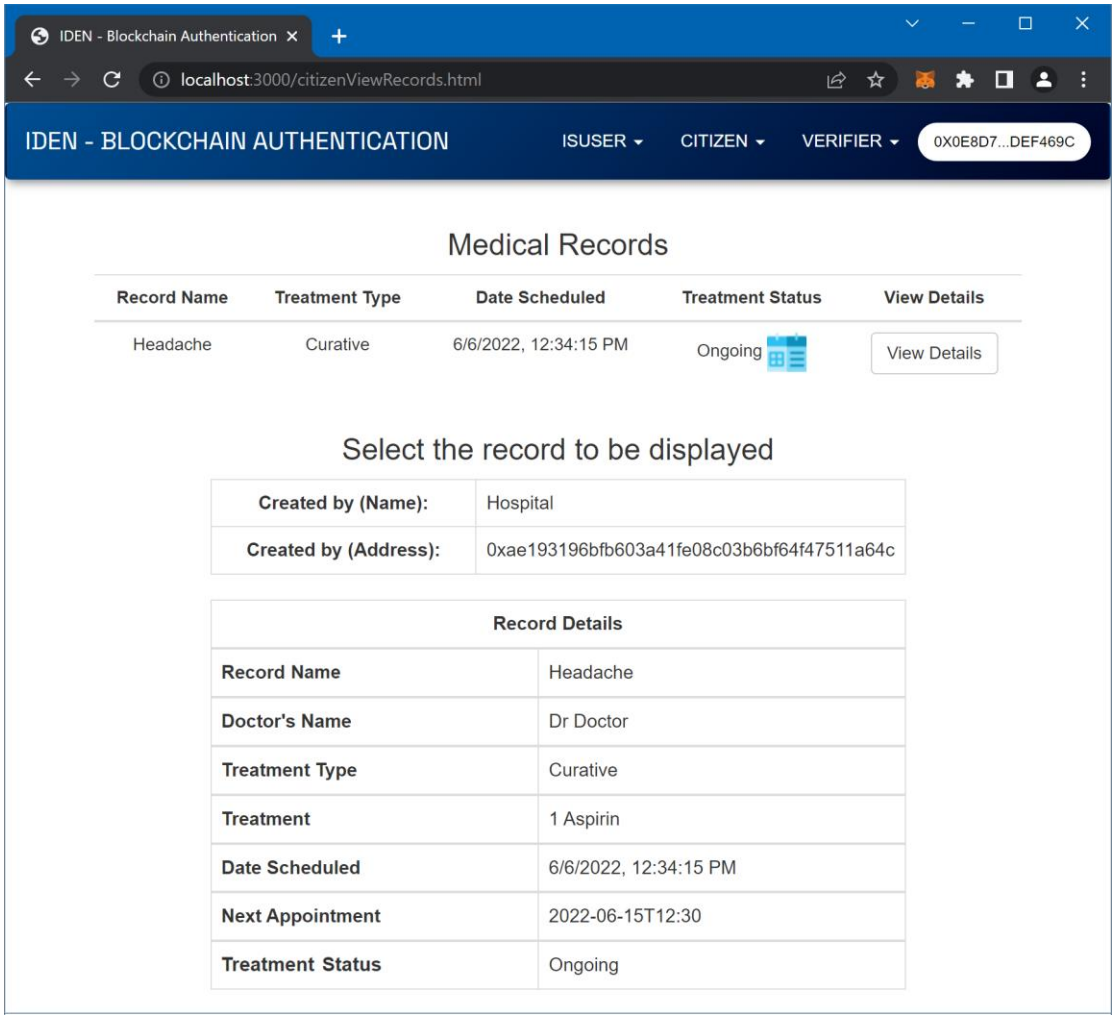


*Figure 83 - Medical records page from the citizen's account*

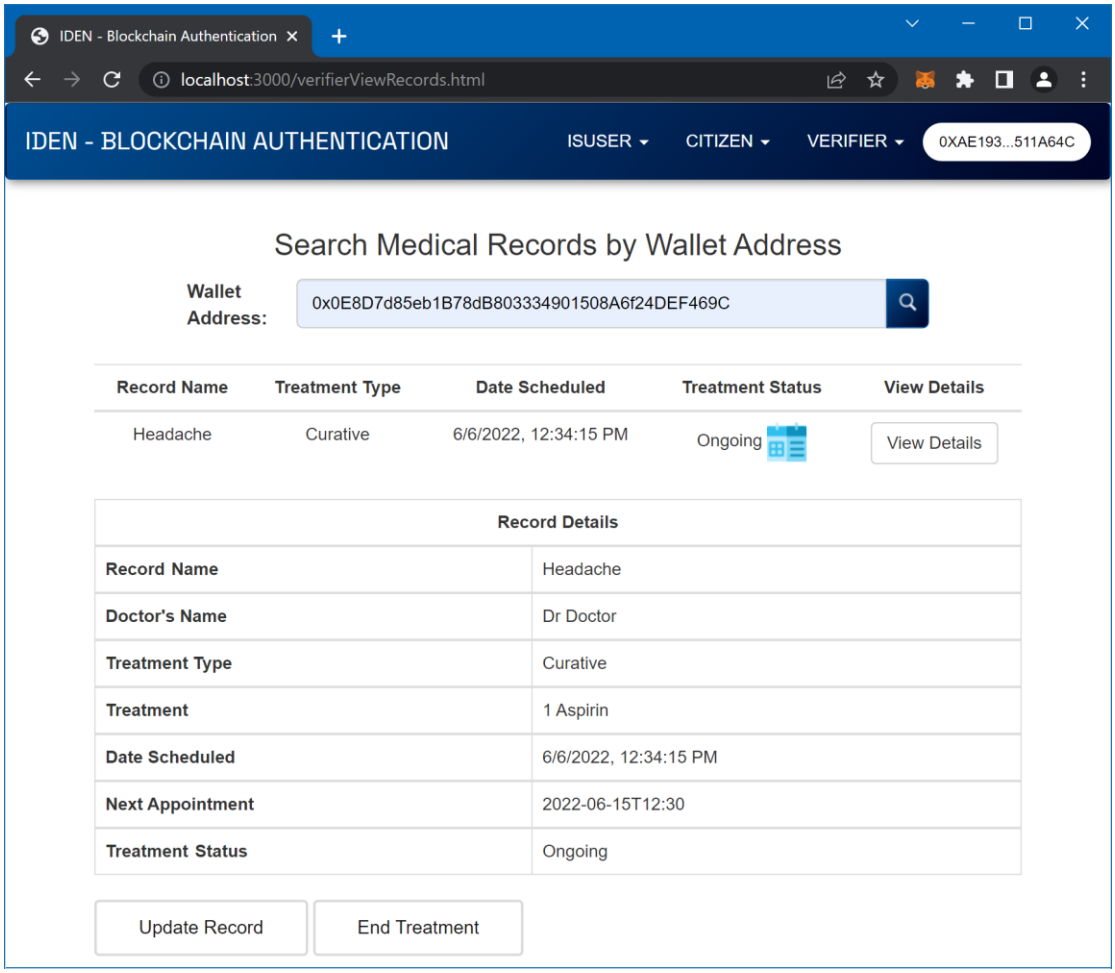


*Figure 84 - Medical records page from the verifier - hospital account*

It is worth noting that the smart contracts of the application, in addition to the local Blockchain of Ganache, are also deployed on the public Blockchain test networks "Ropsten" and "Rinkeby". Which means that the transactions and the above use case in general, can be carried out by any system accessing the app, without the existence, locally, of the smart contracts, provided that there is an available ETH balance on one of the two networks.

## 4.7. Testing procedure

The testing process is defined as the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs, and improving performance (IBM, 2019b).

The tests to which the application was subjected are presented below. Nominally, these are: Unit Testing, Performance Testing, GUI Testing, Cross-Browser Testing, Regression Testing and Sanity Testing.

### 1. Unit Testing

Unit Testing is a type of software testing where individual units or components of software are tested. Its purpose is to validate that each unit of software code performs as expected (Hamilton, 2019a). In this project, the test code is written in Javascript programming language, and the "chai" module of Node.js is used. The set of test files is located within the project's "test" folder and each test is of the format "(name).test.js". In the following example of Unit Testing code (Figure 85), a copy of the deployed on the Blockchain, smart contract is created and then a check is performed for the existence of a citizen with the specified wallet address.

Issuer.test.js > ...

```
1   const { assert } = require("chai");
2
3   const Issuer = artifacts.require('./Issuer.sol');
4
5   contract('Issuer', (accounts) => {
6       before(async () => {
7           this.Issuer = await Issuer.deployed()
8       })
9
10      it('Should check if a Citizen exists', async () => {
11          const citizenExists = await this.Issuer.checkExists('0xFABeceBf00E1EA7A835d01f86412A3fA472E4268');
12          assert.equal(citizenExists, true);
13      });
14
```

*Figure 85 - Example of Unit Testing code of the file "Issuer.test.js"*

### 2. Performance Testing

Performance Testing is a test measure that evaluates the speed, responsiveness and stability of a computer, network, software or device under a sizable workload (Gillis, n.d.). Due to the application's direct dependence on the Ethereum Blockchain rather than a central database, it is not possible to test its speed and stability under

adverse conditions. Approaching the topic theoretically, as long as the Ethereum Blockchain is operational and supported by its nodes, the more responsive and accessible the application will be, regardless of the number of concurrent users.

## 3. Graphical User Interface (GUI) testing

Graphical User Interface Testing is a type of software testing that tests the graphical user interface. The purpose of these tests is to ensure that the functions of the application are performed according to specifications by testing screens and elements such as menus, buttons, icons, etc. (Hamilton, 2019a).

To test the graphical environment of the application, the "Developer Tools" of the "Google Chrome" browser were used and more specifically the "Device Toolbar" option (Figure 86). This made it possible to test the appearance and correct functioning of all the graphical elements of the application in a variety of screen resolutions and devices.

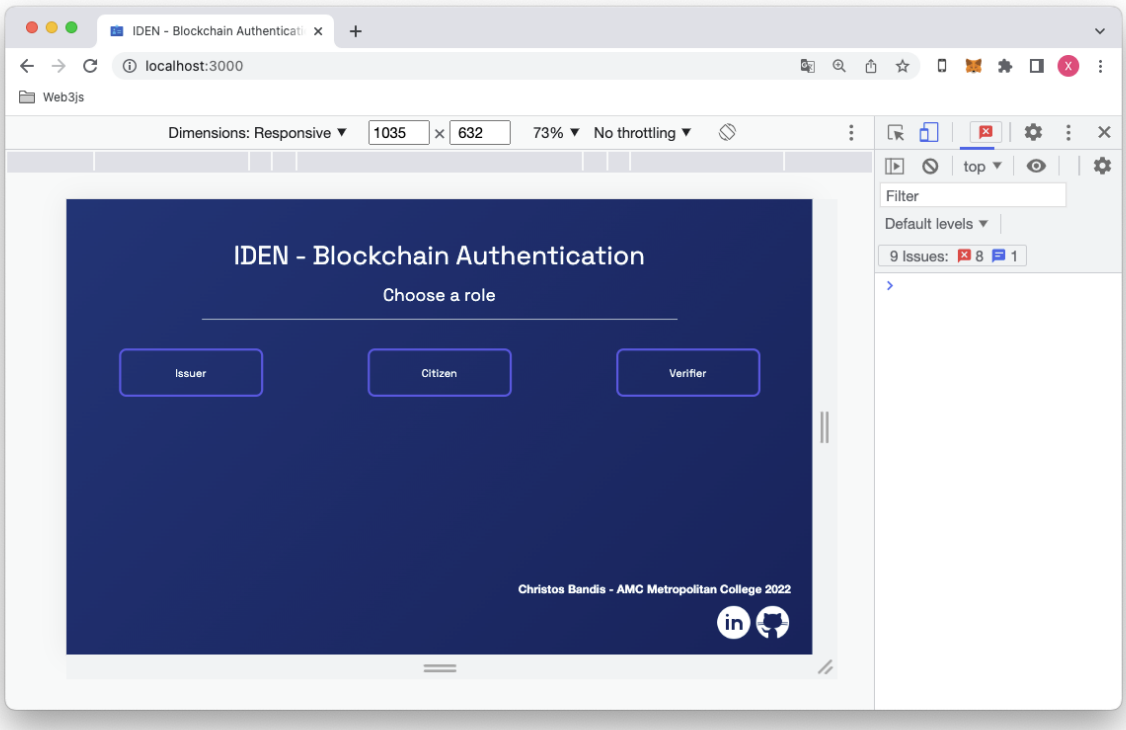


*Figure 86 - Device toolbar (Developer Tools - Google Chrome)*

## 4. Cross-Browser Testing

Cross Browser Testing is a type of testing to monitor the functioning of the application in different browsers (Rungta, 2020). The modern browsers were used for

the testing are: “Google Chrome”, “Mozilla Firefox”, “Brave”, “Microsoft Edge” and “Safari”. The result of the test was that in all browsers the application has the same behavior and appearance (form fields, buttons, fonts), with the only difference of the calendar appearance in the date input fields, and this due to its peculiarity of being created entirely from “HTML” code, without any additional configuration. More specifically, its appearance is identical in browsers based on the open source “Chromium” engine (Chrome, Brave, Edge) and different in “Firefox” and “Safari”. Also, in browsers for which no wallet extension has been developed (e.g. Safari), the functions of the application are not executed, turning it into a simple website.

### 5. Regression Testing

Regression Testing is defined as a type of software testing designed to confirm that a recent change to the application code has not negatively impacted existing functionality (IBM, 2019b). Throughout the development of the application, regression testing was performed, as any critical changes to the code required testing for any bugs. In addition, these tests were beneficial in fixing defects and performance issues.

### 6. Sanity Testing

Sanity Testing is a subset of regression testing. Once a version of the software is developed, sanity tests are performed to ensure that the changes introduced in the code set work as expected. This testing is a checkpoint for determining the continuity of testing for the current release. During the development of the project, logic testing was applied before each regression test was performed, aiming to save time from unnecessary testing.

### 7. Other observations

With the use case presented in Chapter 4.6, it is clear that a user (citizen) address can be used in the case of hospital ID registration and vice versa. This is because of the peculiarity of “Ganache” to create a different dozen accounts at each startup, therefore it was not possible to list specific addresses for the purpose of checking them (see page 58).

Also worth mentioning is the approval process for the three transactions generated when an identity is registered. Because of the “linking” of these transactions

(see pages 82, 83), it is observed that if one of the last two is cancelled while the corresponding error message is displayed, the preceding transaction(s) cannot be undone, due to the specificity of the Blockchain being irreversible.

# 5. Conclusions

In this paper, extensive research was conducted on Blockchain technology and digital identification, as well as the Ethereum Blockchain-based identification application "IDEN" was presented, which uses smart contracts to perform its operations.

Digital identity technologies, despite the benefits they provide to society, have until now been the number one target for malicious users and for-profit services, due to the unorthodox management of data by third parties. While privacy laws have been established, this is reinforced by the unprecedented use of the internet and mobile devices.

In general, Blockchain technology is founded on transparency, strong encryption and instant accessibility - features that, among others, are said to be vital for a multitude of everyday activities, such as banking, voting systems, supply chain management and document validation. Based on this, anything that is in physical/analogue form, such as a value of a good or the identity details of a person or object, can now be registered and maintained online (Laurence, 2017). Thanks to this technology, it is possible to design a decentralized system for secure data management.

The digitization of the identification process seems to be possible now through Blockchain. Decentralized identity management systems, combined with smart contracts, bring integrity and trust to transactions between entities. The identity owner is the only one with access to their personal data, restoring the security not provided by traditional identity management systems, while the verifier who requests to verify some of this data ceases to worry about potentially falsified identity data.

It is common knowledge that Blockchain technology is still in the early stages of development, surrounded by incomplete legislation. However, as it "matures", more and more efficient and secure options are being presented that resolve the social, political and economic issues that exist around the world. According to Schwab (2016), "We are on the verge of a technological revolution that will fundamentally change the way we live, work and relate to each other. In its scale, scope and complexity, this transformation will be unlike anything humanity has experienced in the past. We do not yet know how it will unfold, but one thing is clear: the response to it must be thorough,

involving all stakeholders in global politics, from the public and private sectors to academia and society."

## 5.1. Future actions

The decentralized identification application "IDEN", developed for the needs of this project, is a fully functional approach to the requirements of the assigned topic, but this does not make it complete. Future development goals include the inclusion of more use cases, such as the use of the application for data verification in a governmental mechanism, private organizations, etc. The objectives also include research into other types of Blockchain, such as Solana, Cardano and Holochain, in order to support the application more widely.

# 6. Appendix

## 6.1. References

Allende López, M. (2020). *Self-Sovereign Identity: The Future of Identity: Self-Sovereignity, Digital Wallets, and Blockchain*. Inter-American Development Bank. doi:10.18235/0002635.

AppDividend. (2022). *What is Agile Scrum Master and How Scrum Process Works*. [online] Available at: https://appdividend.com/2022/01/11/what-is-agile-scrum-master/ [Accessed 30 Apr. 2022].

Bashir, I. (2017). Mastering blockchain: distributed ledger technology, decentralization, and smart contracts explained. Birmingham - Mumbai Packt March.

Bitpanda (n.d.). *What are Smart Contracts and how do they work?* [online] www.bitpanda.com. Available at: https://www.bitpanda.com/academy/en/lessons/what-are-smart-contracts-and-how-do-they-work/ [Accessed 20 Apr. 2022].

CardBoard. (2020). *How to Prioritize Agile Stories*. [online] Available at: https://cardboardit.com/2020/08/how-to-prioritize-agile-stories/ [Accessed 30 Apr. 2022].

CB Insights Research. (2018). *What is Blockchain Technology?* [online] Available at: https://www.cbinsights.com/research/what-is-blockchain-technology/ [Accessed 18 Apr. 2022].

Chamber of Digital Commerce (2016). *Smart Contracts: 12 Use Cases for Business & Beyond A Technology, Legal & Regulatory Introduction — Foreword by Nick Szabo*. Washington, D.C.

Dahan, M. and Hanmer, L. (2015). *The Identification for Development Agenda: Its Potential for Empowering Women and Girls*. [online] Washington, DC: World Bank. Available at: https://openknowledge.worldbank.org/handle/10986/22795 [Accessed 24 Apr. 2022].

Das, R. (2016). *Adopting Biometric Technology Challenges and Solutions*. CRC Press.

Deol, R. and Wiesner, T. (2021) 'Ethereum Blockchain Developer Bootcamp With Solidity (2022)' [Recorded lecture], *Udemy*. Available at: https://www.udemy.com/course/blockchain-developer/learn/lecture/17026898#overview [Accessed: 17 March 2021].

Frankenfield, J. (2021). *Decentralized Applications – dApps*. [online] Investopedia. Available at: https://www.investopedia.com/terms/d/decentralized-applications-dapps.asp [Accessed 20 Apr. 2022].

Gauravaram, P., McCullagh, A. and Dawson, E. (2006). The legal and practical implications of recent attacks on 128-bit cryptographic hash function. *First Monday*, 11(1). doi:10.5210/fm.v11i1.1306.

Gillis, A.S. (n.d.). *What is Performance Testing?* [online] Available at: https://www.techtarget.com/searchsoftwarequality/definition/performance-testing [Accessed 5 May 2022].

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 5 May 2022].

Hamilton, T. (2019b). *GUI Testing Tutorial: User Interface (UI) TestCases with Examples*. [online] Available at: https://www.guru99.com/gui-testing.html [Accessed 5 May 2022].

Hartley, J. (2011). *Communication, Cultural and Media Studies The Key Concepts*. London: Routledge.

Hayes, A. (2022). Blockchain Explained. [online] Investopedia. Available at: https://www.investopedia.com/terms/b/blockchain.asp [Accessed 17 Apr. 2022].

IBM (2019b). *What is software testing?* [online] Ibm.com. Available at: https://www.ibm.com/topics/software-testing [Accessed 5 May 2022].

IBM (2022). *What are smart contracts on blockchain?* [online] www.ibm.com. Available at: https://www.ibm.com/topics/smart-contracts [Accessed 20 Apr. 2022].

IBM (n.d.). *What is blockchain technology? - IBM Blockchain*. [online] www.ibm.com. Available at: https://www.ibm.com/topics/what-is-blockchain [Accessed 18 Apr. 2022].

IBM. (2019a). *Cryptographic concepts*. [online] Available at: https://www.ibm.com/docs/en/ibm-mq/9.0?topic=mechanisms-cryptographic-concepts [Accessed 26 Apr. 2022].

Iredale, G. (2018). The History of Blockchain Technology: Must Know Timeline. [online] 101 Blockchains. Available at: https://101blockchains.com/history-of-blockchain-timeline/ [Accessed 17 Apr. 2022].

Jal, A. (2018). Secure The Data In Multi Cloud Using Erasure Code And Merkle Hash Tree Algorithm. *International Journal of Recent Trends in Engineering and Research*, 4(4). doi:10.23883/ijrter.2018.4198.w12zv.

JAXenter. (2019). *No, you don't store data on the blockchain - here's why*. [online] Available at: https://jaxenter.com/blockchain-data-164727.html [Accessed 1 May 2022].

Kaley, A. (2021). *Mapping User Stories in Agile*. [online] Nielsen Norman Group. Available at: https://www.nngroup.com/articles/user-story-mapping/ [Accessed 30 Apr. 2022].

Laurence, T. (2017). *Blockchain for dummies*. Hoboken, Nj: John Wiley & Sons.

Logaras, K. (2018). Blockchain technology, its applications and legal aspects. [online] Available at: https://www.naftemporiki.gr/story/1363055/i-texnologia-blockchain-oi-efarmoges-tis-kai-oi-nomikes-ptuxes-tis [Accessed 17 Apr. 2022].

Lyons, T., Courcelas, L. and Timsit, K. (2016). *The European Union Blockchain Observatory and Forum*. [online] European Union. Available at: https://www.eublockchainforum.eu/sites/default/files/report_identity_v0.9.4.pdf [Accessed 26 Apr. 2022].

Manby, B. (2016). *Identification in the Context of Forced Displacement: Identification for Development*. [online] Washington, DC: World Bank. Available at: https://openknowledge.worldbank.org/handle/10986/24941 [Accessed 24 Apr. 2022].

Marx, S. (2018). *Understanding Ethereum Smart Contract Storage*. [online] Available at: https://programtheblockchain.com/posts/2018/03/09/understanding-ethereum-smart-contract-storage/ [Accessed 20 Apr. 2022].

*McKinsey* (2019). *Infographic: What is good digital ID? | McKinsey*. [online] Available at: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/infographic-what-is-good-digital-id [Accessed 24 Apr. 2022].

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 30 Apr. 2022].

Okta. (n.d.). *Hashing Algorithm Overview: Types, Methodologies & Usage | Okta*. [online] Available at: https://www.okta.com/identity-101/hashing-algorithms/ [Accessed 26 Apr. 2022].

Pandit, H.J., O' Sullivan, D. and Lewis, D. (2018). Queryable Provenance Metadata For GDPR Compliance. In: *Procedia Computer Science*. SEMANTiCS 2018 – 14th International Conference on Semantic Systems.

Pato, J. and Rouault, J. (2003). *Identity management: The drive to federation*. Technical White Papers.

Ray, S. (2021). *What is a DAPP?* [online] Medium. Available at: https://towardsdatascience.com/what-is-a-dapp-a455ac5f7def [Accessed 20 Apr. 2022].

Rungta, K. (2020). *Cross Browser Testing using Selenium WebDriver*. [online] www.guru99.com. Available at: https://www.guru99.com/cross-browser-testing-using-selenium.html [Accessed 5 May 2022].

Schwab, K. (2016). *The Fourth Industrial Revolution: what it means, how to respond*. [online] World Economic Forum. Available at: https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/ [Accessed 6 May 2022].

Scrum.org (2016). *What is Scrum?* [online] Scrum.org. Available at: https://www.scrum.org/resources/what-is-scrum [Accessed 30 Apr. 2022].

Shackelford, S. and Myers, S. (2016). Block-by-Block: Leveraging the Power of Blockchain Technology to Build Trust and Promote Cyber Peace. *SSRN Electronic Journal*. doi:10.2139/ssrn.2874090.

The Public Voice, (n.d.). *Privacy: Background – thepublicvoice.org*. [online] Available at: https://thepublicvoice.org/issues_and_resources/privacy-background/ [Accessed 26 Apr. 2022].

The World Bank Group, GSMA and Secure Identity Alliance (2016). *Digital Identity: Towards Shared Principles for Public and Private Sector Cooperation*. [online] Washington, DC: World Bank. Available at: https://www.gsma.com/mobilefordevelopment/wp-content/uploads/2016/07/Towards-Shared-Principles-for-Public-and-Private-Sector-Cooperation.pdf [Accessed 28 Apr. 2022].

Treiblmaier, H. (2019). Toward More Rigorous Blockchain Research: Recommendations for Writing Blockchain Case Studies. *Frontiers in Blockchain*, 2. doi:10.3389/fbloc.2019.00003.

Vishnia, G.R. and Peters, G.W. (2020). AuditChain: A Trading Audit Platform Over Blockchain. *Frontiers in Blockchain*, 3. doi:10.3389/fbloc.2020.00009.

Wikipedia (2020). *Zooko's triangle*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Zooko%27s_triangle [Accessed 26 Apr. 2022].

World Bank (2018a). *Catalog of Technical Standards for Digital Identification Systems (English).* Washington, D.C.: World Bank.

World Bank (2018b). *Technology Landscape for Digital Identification*. [online] Washington, DC: World Bank. Available at: https://openknowledge.worldbank.org/handle/10986/31825 [Accessed 25 Apr. 2022].

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 30 Apr. 2022].

## 6.2. List of Figures

## 6.3. List of Tables



## 6.4. Glossary

**Agile:** Development methodology

**Byte:** Unit of measurement of the amount of information

**Bytecode:** Type of binary code

**ECMAScript:** A Javascript standard for ensuring the interoperability of web pages

**ETH:** The Ethereum Blockchain currency

**Ethereum:** Blockchain Protocol

**Gas:** The “fuel” of the Ethereum Blockchain

**Gwei:** Ethereum's smallest unit of measurement (ETH)

**Hash:** Cryptographic Mathematical Representation

**HyperText Markup Language - HTML:** Web Markup Language

**JavaScript Object Notation - JSON:** Text file for data transmission

**Javascript:** Programming language

**Merkle Tree:** Hash tree

**Node.js:** Javascript code execution environment

**REST API:** Method of communication between systems

**Solidity:** Programming language for smart contracts

**Turing-Complete:** Computationally complete system

**Yellow Paper:** research paper that has not yet been published academically