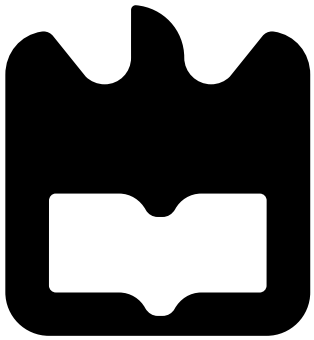
Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática
2016

Daniel Filipe Catita Martins

Utilização de Blockchain na aplicação de Master Data Management
Utilization of Blockchain in the application of Master Data Management



Daniel Filipe Catita Martins

Utilização de Blockchain na aplicação de Master Data Management
Utilization of Blockchain in the application of Master Data Management

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor José Manuel Matos Moreira, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professor Doutor Joaquim Arnaldo Carvalho Martins
Professor Catedrático, Universidade de Aveiro

vogais / examiners committee

Professor Doutor José António Rodrigues Pereira de Faria
Professor Auxiliar, Universidade do Porto

Professor Doutor José Manuel Matos Moreira
Professor Auxiliar, Universidade de Aveiro

agradecimentos

Ao meu orientador, Professor José Moreira, o meu forte agradecimento pelo apoio incondicional, disponibilidade e acessibilidade. Por ter confiado em mim ao longo de todo este processo.
Ao Bruno Oliveira, o gestor de todo este processo. Obrigado pela dedicação, pelo conhecimento partilhado e por nunca ter estado longe, mesmo estando.
À Deloitte agradeço a oportunidade, o tema da dissertação e o suporte científico, tão útil na elaboração deste projeto.
Não esqueço o agradecimento à minha família, pelo esforço que foi para eles eu estudar numa cidade diferente, pelos conselhos, apoio e estabilidade.
Por fim, aos meus amigos, por se terem mantido próximos e apoiado em todos os momentos.

keywords Blockchain, Master Data Management, Smart Contracts, Golden Record, Dados Mestre

abstract

Como o nome indica, Master Data Management (MDM) gere dados mestre: o conjunto de informação nuclear necessária e partilhada pelos sistemas de uma empresa. Dependendo do âmbito da organização, dados mestre podem ser dados de clientes, caso todos os sistemas considerem a informação dos clientes crítica para a suas operações e decisões.
Um conceito fundamental de MDM é o Golden Record, um registo com a melhor e mais valiosa informação sobre uma determinada entidade, formada através da aplicação de regras e métodos nos dados existentes que estão espalhados pelos vários sistemas. É a versão única da verdade.
As soluções de Master Data Management existentes são dependentes de um local centralizado onde fica guardada a informação mais valiosa. A solução proposta é disruptiva para a lista de ofertas, combinando o conceito de Master Data Management com a tecnologia Blockchain, resultando numa solução MDM distribuída e descentralizada.
A Blockchain consiste numa cadeia de blocos que requerem esforço computacional para que se adicionem novos blocos ao fim da cadeia e onde os blocos não podem ser alterados sem repetir o esforço computacional para todos os blocos seguintes, o que resulta num ambiente confiável.
Os participantes de uma rede Blockchain possuem uma cópia total da Blockchain, tornando-a assim uma rede distribuída de informação.
Os protocolos e requisitos de segurança eliminam a necessidade de existência de um intermediário entre as os participantes de uma transação e tornam a Blockchain ideal para guardar objetos de valor.
A solução envolve Ethereum: uma plataforma Blockchain onde as transações têm funcionalidade programável, conhecida como Smart Contracts, pedaços de código que contêm um conjunto de dados e funções executáveis que estão disponíveis a partir de um endereço público.
Como todos os dados inseridos a partir de um Smart Contract estão sujeitos às mesmas regras específicas de standardização, correspondência e fusão, todos os participantes da rede vão possuir os mesmos Golden Records, resultando na visão única de entidade.
Para facilitar a utilização da solução, foram desenvolvidos um wrapper e uma interface de utilizador, para permitir que o utilizador não necessite de interagir diretamente com a Blockchain.

keywords Blockchain, Master Data Management, Smart Contracts, Golden Record, Master Data

abstract As the name implies, Master Data Management (MDM) manages Master Data: the set of core information needed and shared in the systems of an enterprise. Depending on the scope of the organization, master data could be data about clients if all the systems consider client information critical for their operations and decision making.
A fundamental concept of MDM is the Golden Record, an entry with the best and more valuable information about an entity, formed through the application of rules and methods on the data that exists scattered over the systems. It is the single version of the truth.
Master Data Management solutions are dependent of a centralized data hub that holds the most valuable information. The solution proposed disrupts the list of offers, combining the Master Data Management concept with the Blockchain technology, resulting in an MDM solution with a distributed data hub that is truly decentralized.
A Blockchain consists on a chain of blocks that requires computational work to attach new blocks to the end of the chain, and where blocks cannot be changed without redoing the computational effort for all the following blocks, resulting on a trusted environment.
Participants of the Blockchain network hold a full copy of the Blockchain, making it a distributed network of information.
Its security protocols and requirements eliminate the need for an intermediary between transactions and make Blockchain ideal to save things of value.
The solution involves Ethereum: a Blockchain platform where transactions have programmable functionality, known as Smart Contracts, pieces of code containing a set of data and executable functions that are available through the public address.
As all the data inserted through the programmed Smart Contracts goes through the same specific cleansing, matching and merging rules, all the network participants will be in possession of the same Golden Records, resulting in a single view of entity.
To facilitate the utilization of the solution, a wrapper and user interface were developed, granting that the user does not need to interact directly with the Blockchain.

Contents

List of Figures

List of Tables

1 Introduction

Different systems of an organization co-exist and might obtain information from different sources, sometimes ignoring existing data on other data systems. This characteristic leads to variations in the various data sets and imposes a challenge for an organization to use the data that matters the most in an effective way.

The implementation of a Master Data Management solution aims to solve this issue, by creating a Master Data Management Hub from where the most relevant data for the organization is accessible.

In the context of big organizations where frequently the same entities of the real world, such as Clients, are represented on multiple systems, it becomes necessary to implement a Master Data Management (MDM) strategy to achieve a unique vision of entity.

The challenge of achieving a unique vision of entity in enterprises of considerable size demands innovative and disruptive approaches, leading to the choice of Blockchain as the core technology to implement a decentralized MDM solution, where it is possible to enforce standardization, matching and merging rules and consult Golden Records.

The choice of Blockchain, the backbone of Bitcoin, as the core technology for the solution is also justified by it being an emergent technology and companies all over the world are competing to find usages other than cryptocurrency to revolutionize the way data, knowledge and value is stored and shared.

The work presented in this dissertation consisted on the development of a functional platform and respective testing, as a proof-of-concept for the utilization of Blockchain in the application of MDM.

The implemented Blockchain platform is capable of sustaining an MDM solution that is able to provide a unique vision of client throughout an enterprise, by enforcing the MDM rules of standardization, matching and merging, being distributed and available to all the demanding parties. Although it is a very specific proof of concept, its functionalities could be adapted and extended to solve other problems and deal with data records of different kinds.

This dissertation presents all the research, work, results and conclusions behind the development of the proof-of-concept and is divided in six different chapters:

- The first and current chapter presents the motivation behind this work, the objectives it aims to fulfil and the structure of the document.
- The second chapter, State-of-the-Art, reveals the results of the research on Master Data Management and Blockchain.
- The third chapter, Requirements Elicitation and System Design, presents the proposed solution for combining MDM and Blockchain, and explains the Functional and Technical Design of the solution, establishing the requirements and the definition of the architecture.
- The fourth chapter refers to the Implementation of the solution, exposing the decisions and challenges behind the exercise of implementing this application.
- The fifth chapter presents the tests conducted to analyse the success of the implementation keeping in mind its objectives and requirements, and respective results.
- Finally, this dissertation ends with a sixth chapter that presents the conclusions of the overall production and suggests related future work.

2
State-of-the-Art

This chapter is the written result of the research on Master Data Management and Blockchain, presenting the theoretical fundamentals behind the technology and solution proposed with this dissertation.

2.1 Master Data Management

An enterprise is often composed of many applications referring to multiple, and in some cases disparate, sets of data that are supposed to represent the same entity in the real world, or similar data sets that are supposed to represent different entities [1]. This lack of a single version of truth is a direct result of the different systems co-existing within the company that obtain information from different sources and ignore existing data on another systems, leading to inconsistencies in the various data sets, making it impossible for an organization to effectively use the data that matters the most [2].

To solve this issue, Master Data Management concentrates on the business processes, data quality, standardization and integration of information systems, defining the most trusted and unique version of critical data [3]. It is an approach that describes, creates, manages and maintains a reliable, sustainable, secure and consistent environment, providing a single source of critical data [2].

This chapter presents an overview of Master Data and Reference Data, explains why it is important to manage this kind of data with Master Data Management, describes the most common architecture patterns and summarizes some of the most popular software solutions on the market.

2.1.1 Master Data

Master Data encompasses the business entities that are agreed on and shared across an enterprise and are fundamental for the organization's operations and decision-making. Being the set of core information that a business owns and uses across organizational units among

businesses [4], the exact definition of what is the enterprise's Master Data depends on the its perspective and scope of action [5].

Master Data is needed across the diverse systems of an enterprise and it is important for all parts of an organization to agree on that information, in order to have a common understanding of what defines a customer, a product, or a purchase, for example [4]. This brings us to the need of managing the Master Data, ensuring there is a single version of truth within an organization [6], the Golden Record.

Reference Data is data used to categorize other data within the company, defining the set of permissible values to be used by other data fields and is generally unform and company-wide. Examples include information such as currency codes, country codes or, for example, the possible values that can be used to describe the job position of a given worker [7].

According to some authors, Reference Data can also be considered Master Data [8] and, although it is also shared across the enterprise and somewhat critical to the good functioning of the enterprise, Reference Data is not necessarily associated with real-world entities [5].

2.1.2 Principles and Motivations

More than a technologic approach for data systems, the foundation of Master Data Management is the organization, its people and processes associated with the business [9], which leads to the necessity of educating the people and rethink process flows around the existence of a Hub that is the single source of truth regarding Master Data or a way to find it [4].

In order to establish an MDM function, the needs and objectives of the organization must be defined, giving a broad view of the impact the MDM will make in the business and how to implement it. Usually, the results encompass more effective work, as the processes are streamlined, and reporting is improved, a direct result of the data quality improvement [3].

The organization must clearly define what is its Master Data and core processes, have a clear definition of what is to be shared across the enterprise [4], and identify roles and responsibilities related to the MDM process. Allen and Cervo [9] discriminate the key factors that should drive the priorities on which an MDM system must focus:

- Business Value

  Organizations dealing with various data domains should evaluate which need MDM to determine the benefits and the increase of value subjacent to that implementation versus its cost.

- Volume

  Organizations that deal with a large volume of data and a vast number of parties that share that data will benefit from an MDM approach to facilitate the access to fundamental information.

- Volatility

  Companies where data is more volatile, changing frequently over time, benefit from an MDM implementation so there exists a place where there is guarantee that the most updated information is available.

- Reusability

Organizations that possess sets of data that is highly shared and needed across the business benefit from an MDM hub to allow a reliable set of information to flow through the enterprise's systems.

- Complexity

  Companies that deal with very complex data, with a large number of attributes and properties will find an MDM implementation very useful in order to create a process of simplification, through rules and methods, centered on an unique place where data is then available.

Having inconsistent Master Data leaves the business in a more challenging position [10], as systems within the organization might be operating over outdated or incomplete information. One of the objectives of an MDM approach is having the data standardized across the enterprise, ensuring consistence [3]. Being so, standards must be defined for content and structure, which will influence reporting in a positive way [11], as it will be consistent and uniform across all the organization.

A blatant case would be the ability to present the company as one to a customer, having several departments: Following an MDM approach and achieving an unique view of entity that is shared across the enterprise with standardized processes, an unique customer identification would also be achievable, and from the customer's perspective, the enterprise complex structure would be irrelevant, as all the organization shares the same information and language. A natural consequence would be an improvement in customer service, leading to an increase in customer satisfaction, and productivity, as there is no need for two systems to create a different entry about the same customer [2].

2.1.3 Creating the Golden Record

In the heart of an MDM implementation lies the Golden Record: a data entry consisting of the best and more valuable possible information that is available regarding an unique entity, allowing system participants who wish to access data regarding that entity to retrieve undoubtedly the most relevant and unambiguous information.

The Golden Record refers to the single truth and has to be created from all the records from various systems. This Golden Record is achieved by calculating an unique entry from several records regarding the same real world entity in disparate systems, selecting the most correct, complete and actual attributes.

Data that is being used to create a Golden Record should be validated so only correct data is part of the single source of truth. Incorrect data, in terms of formatting, might be corrected through methods of cleansing and transformation when creating the Golden Record, or discarded.

It is also possible to assign priorities regarding the source of the information (data from system A should prevail over data from system B, for example) or rules to compare and decide in one from various data attributes (client address with more characters should prevail over client addresses with less characters, for example).

As depicted in the Figure 2.1, an example organization had data regarding John and Jane Doe scattered across various systems. To create the Golden Record, data was selected from different systems to create two complete entries with standardized structures. A decision making method has to be in place in order to determine which data to select, validate it, and how to correct it, if needed [12].

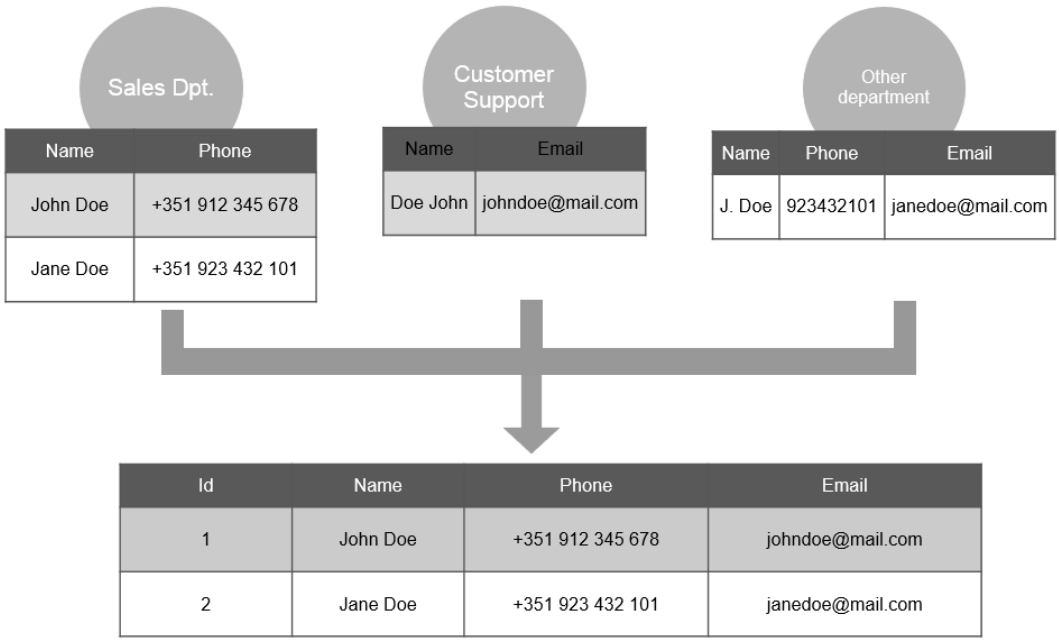


Figure 2.1: Example of the process of creating the Golden Record

In order to achieve and maintain the Golden Record quality, two concepts should be studied and applied: Data Stewardship and Data Governance.

- **Data Stewardship**

Data Stewardship is the set of activities that ensure data-related work is performed according to policies and practices, in order to improve data reusability, accessibility and quality. Being as important as it is, it led to the creation of a new role: the Data Steward, a person responsible for managing data in a corporation [13].

- **Data Governance**

Data Governance is the overall control of the data and its availability, usability and integrity, through processes that ensure that important data assets are formally managed through the enterprise [14].

One can say that Data Governance is strategic while Data Stewardship is tactical: Data Governance is long-term and global and includes tasks such as defining the policies, the goals, roles and responsabilities. Data stewardship is more specific and short-term, regarding specific data elements, including tasks like data definition, business rules definition and setting data quality targets [15].

2.1.4 MDM Implementation Styles

Dreibelbis et al [4] state that the Implementation Style — how the solution is implemented — is one of the three primary dimensions of an MDM system, the others being Domains of Master Data — what Master Data needs to be managed — and the Methods of Use — how the Master Data is going to be used.

Some implementations styles are known as analytical working as an expansion of a data warehouse, and used more often to supply the data for a business inteligence solution not affecting upstream applications. On the other hand, operational implementations are a natural evolution of typical analytical implementations: operational implementations encompass an MDM hub that can be directly referenced by source systems, assuring a single view of Master Data in the core systems [9].

Implementations also differ by being either a system of record or a system of reference. In an MDM that is a system of record, the MDM system is the best source of truth and the authoritative data source; in an MDM solution that is system of reference, the replica of Master Data is known to be synchronized with the system of record — the data sources — in a managed way to mantain the quality of the data within the replica (system of reference) and the sources (system of record) [4].

Four different implementation styles for MDM systems are normally described (Consolidation, Registry, Coexistence, Transaction Hub) [16].

2.1.4.1 Consolidation

In this approach, depicted in the Figure 2.2, Master Data from various existing systems and databases is agglomerated into a single managed MDM Hub where it is transformed, cleansed and matched, producing a Golden Record (trusted source) that is available to downstream systems that use, but do not update, the Master Data [4], working as an intermediary between the sources and the consumers of the data. This is an analytical-type system of reference, with an MDM Hub where data is managed to ensure quality.

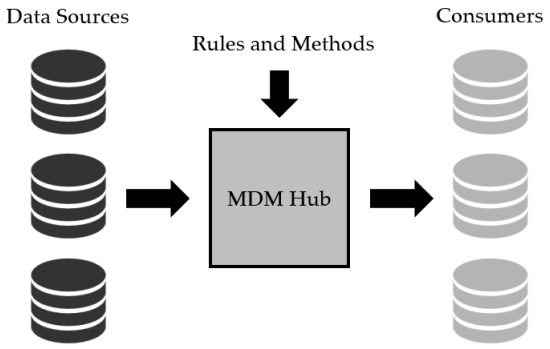


Figure 2.2: Consolidation-type implementation.

2.1.4.2 Registry

In a registry style implementation, depicted in the Figure 2.3, the MDM Hub maintains only the minimum set of necessary information to identify a Master Data record and locate it

at its source [9], unlike the Consolidation style implementation whose MDM Hub maintains a copy of the Golden Record. It serves as read-only system of reference to other applications and it is only able to clean and match the identifying information, assuming the sources manage the quality of their data [4], not updating the sources in any way. This is an operational-type system of reference.

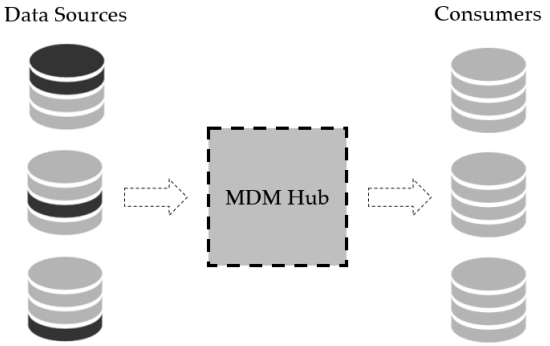


Figure 2.3: Registry-type implementation.

2.1.4.3 Coexistence

In this type of implementation, data stored in the MDM system is synchronized with source systems. As in the consolidation-type implementation, a Golden Record is generated and can be queried and updated within the MDM system, but in this implementation data is fed back to source systems and published to downstream systems, as illustrated in the Figure 2.4. An MDM system implemented this way can serve as an authoritative source of Master Data, managing the quality of data as it is imported to the system [4]. This type is a system of reference, as it is not a single place where Master Data is stored and updated, and it is an operational system.

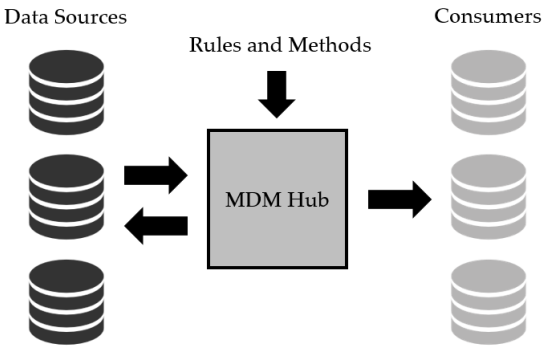


Figure 2.4: Coexistence-type implementation.

2.1.4.4 Transactional Hub

In a transactional hub implementation, the hub is a centralized single source of truth, physically storing all attributes of an entity and used to publish master attributes to other systems [9]. It can be seen as an evolution from the consolidation and coexistence implementations but it is, however, a system of record, where updates to Master Data happen directly to this system, as depicted in the Figure 2.5. After updates are accepted, they are distributed to other applications [4]. A Transactional Hub type implementation is an operational system of record.

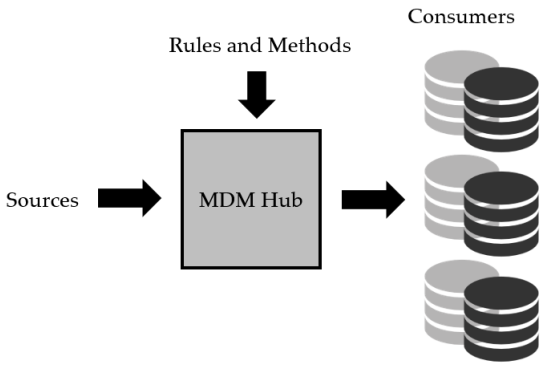


Figure 2.5: Transactional Hub-type implementation.

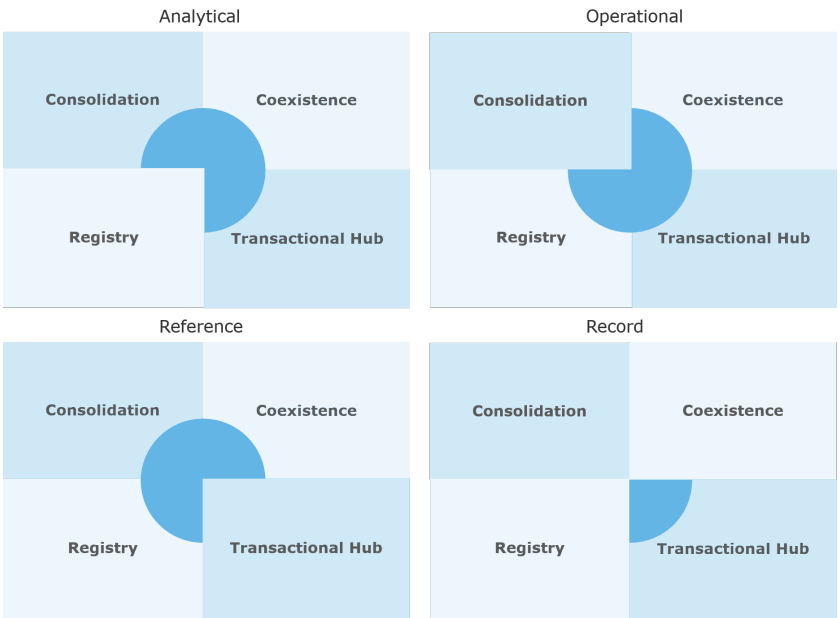


Figure 2.6: Summary of the implementation styles methods and types of system.

Figure 2.6 summarizes the implementation styles methods and types of system.
It is also common to use Hybrid Implementations, which combine a number of implemen-

tation styles.

2.1.5 Existing tools

There are various MDM solutions available in the market. These are software products that allow to: support global identification, linking and synchronization of information across several data sources; create and manage a central system of record or reference for Master Data; enable single view of entity to all business systems; support Master Data stewardship and governance requirements [17].

Some softwares are specialized on specific domains, such as customer data, focusing on management of the data related to customers, or product data, focusing on management of the data related to products [18], which means that the decision of what software to use might depend on the kind of data the organization will be dealing with.

According to Gartner and its Magic Quadrant (Figure 2.7) focusing on MDM solutions to customer data MDM solution softwares, the leaders in the business are Informatica, IBM (InfoSphere MDM), Oracle (Siebel UCM) and TIBCO Software [17]. These are briefly described below.

Figure 2.7: Magic Quadrant for Master Data Management of Customer Data Solutions [17].

- **InfoSphere MDM**

InfoSphere MDM, by IBM, is an MDM solution with collaborative and operational capabilities, capable of managing Master Data for single or multiple domains. It includes multiple editions, but the most known are the Standard Edition and the Advanced Edition [19]. Standard edition users may, at any time, upgrade to the Advanced Edition without any aditional

installations. Although Standard Edition is restricted to Registry-style implementations, it might be an attractive option for organizations seeking a less invasive MDM.

InfoSphere delivers functionality through pre-packaged web services used to seamlessly integrate MDM into existing businesses, ensures high quality via a probabilistic approach to monitor and enforce policies, and employs statistical techniques to resolve and manage data quality issues automatically. It has cloud support to integrate cloud data with organization data.

It should be noted that, at the time of writing this dissertation, there has been over one year since their last version release [17].

- **Informatica**

Informatica claims that their MDM Solution, Informatica MDM, is easy to deploy and flexible [20]. Although it is party-data oriented, it can readily model other data domains. Comparing to other solutions, Informatica MDM is featured in the majority of situations for MDM of Customer Data. Informatica products are not integrated and require several acquisitions, even though its core MDM solution continues to be invested on [17]. This software solution allows users to build their own rules to identify Golden Records. This software solution allows multiple implementation architectures and is integrated with Informatica's Data Quality, Data Integration and Business Process Management products to suport the MDM needs. Also, it comes with a feature called Entity 360 that provides way to visualize and search Master Data records and their relationships [21].

- **Siebel UCM**

Siebel UCM — Universal Customer Master —, by Oracle, is a platform configured to store a clean and unified profile for customer data, and being so, includes features to cleanse, store and manage this data [22]. It is based on a robust and extensible party model, which results on not being the first choice when a multi domain platform is required [17].

One of its strenghts is the Integration Architecture, composed of web-services and Oracle Fusion Middleware, a library of several software products from Oracle Corporation.

Siebel UCM has a integrated Data Quality Module, that allows, for example, data cleansing configuration [23].

- **TIBCO Software**

TIBCO solution for MDM is a multi domain solution based on one single platform, serving Master Data of all types [24]. It is composed by a collaborative information manager, analytics and studio, that allows the creation of data models, and business rules and processes within an intuitive interface [17].

TIBCO Software allows to import data from multiple sources simultaneously, has an intelligent MDM-integrated search and easy integration with in-memory databases. Its architecture fits well with real time and distributed services.

2.2 Blockchain

Blockchain is an innovative technology: a shared ledger, distributed to all the participants in a network supported by consensus protocols and cryptography, allowing transactions to occur and be validated without the need of an intermediate. This section presents how Blockchain works, its origin supporting Bitcoin — a cryptocurrency system —, Blockchain adaptations and some examples of innovative applications that are using this technology.

2.2.1 Bitcoin as Blockchain's origin

Bitcoin is an electronic payment system based on cryptographic proof instead of trust, allowing willing parties to transact directly without the need for a trusted intermediary [25]. It was invented by Satoshi Nakamoto and published in 2008, being one of its most innovative features the fact it is decentralized: there is no central server on which the Bitcoin runs but a peer-to-peer network of connected computers, known as nodes [26].

Each Bitcoin is owned by a wallet, designated by a public address, and its owner, that holds a private key that corresponds to the address public key. A Bitcoin owner can create a transaction [27] in order to send funds to other wallets.

At the center of the Bitcoin network is Blockchain: a database, usually called ledger, as it holds the transactions that have ocurred in the past as well as the current holders of the funds. Nakamoto describes Blockchain as being a chain of blocks that requires computational effort to build, attaching blocks to the end of the chain, and where blocks cannot be changed without redoing the computational effort for all the subsequent blocks [25], keeping the Bitcoin network resilient against users trying to douple-spend their funds, an important requirement of a financial database [26].

Sharing a common origin, it might be difficult to differentiate the Blockchain from the Bitcoin. Usually it is traced a parallel between Bitcoin and Napster, the pioneering peer-to-peer music sharing service: Napster was shut down but peer-to-peer found justifiable uses such as Skype, Spotify and even Bitcoin. In the same way, Blockchain may find uses other than supporting the Bitcoin, should those uses be other cryptocurrencies or more "out-of-the-box" applications [28].

2.2.2 How Blockchain works

One of Blockchain's simplest descriptions comes from the open source code of Bitcoin. Without much detail, Nakamoto described that nodes collect transactions into blocks and scan through nonce values to satisfy proof-of-work requirements. A solved block will be broadcasted, so all the network adds the new block to the end of a chain of blocks. This description also states that the first transaction of each block is special and its purpose is to reward the block creator with a new coin [29].

On the Blockchain Network, participants must mantain concensus on the rules to determine validity of transactions, which transactions have occurred in the system, and that the currency has value. Transactions must be publicly announced for the system participants to agree on a single history of the order in which they were received, to avoid double-spending of coins [25].

When a new transaction is broadcasted to all nodes of the network, each node collects the new transaction into a block and works on solving a difficult computational problem on

a competitive process known as Mining [30]. When a node finds a solution for the block it is working on, it broadcasts the block to all the other nodes and is rewarded with bitcoins [25].

On the network, there are two types of nodes: mining nodes, that participate in the mining process trying to solve blocks and be rewarded, and passive nodes, like wallets, that do not solve blocks but still possess a full copy of the Blockchain [26].

It is possible to break the process into distinct phases and explain them with a litte more detail.

2.2.2.1 Collect new transactions into a block

Transactions are created by network participants and are the content to be stored in the Blockchain. On Blockchains supporting cryptocurrencies, like Bitcoin, the transaction is created when an owner transfers coins to other participant, and cryptography methods — such as digital signatures — are applied so the beneficiary can verify the chain of ownership, as illustrated in the Figure 2.8.

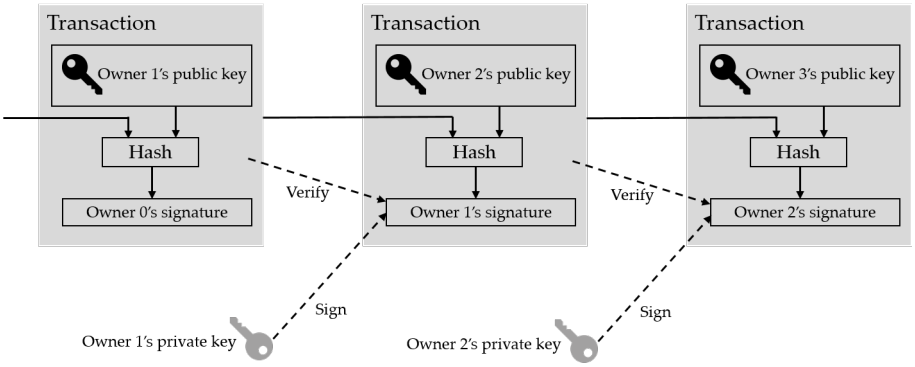


Figure 2.8: Bitcoin Blockchain's transactions. A participant owner of coins, transfers coins to another participant by signing a hash[1]of the previous transaction, and the public key of the new owner [25].

When a node produces a transaction, it is broadcasted to the Blockchain network to be validated by other nodes. This unconfirmed transaction is stored in a local database until rules of validation — such as double-spending verification — are applied and the transaction is added to a block and considered confirmed [26].

A block is composed of two main parts:

- Header

  The header of a block contains a reference to the previous block, attributes such as a timestamp and a nonce[2], and a Merkle tree root structure[3] to summarize all the transactions within the block. The nonce is used on the partial hash inversion problem [26].

[1]Result of a calculation (hash algorithm) that serves as a "fingerprint" proving the source of the coins that are going to be transfered.

[2]Arbitrary number that can only be used once.

[3]Tree structure in which every non-leaf node is labelled with the hash of the labels or values of its child nodes.

- Content

  A block also contains the confirmed transactions, a validated list of digital goods and instructions, their amounts and the addresses of the involved parties. A single block contains several transactions that were produced in the same time frame [31].

Each block is protected with a partial hash inversion proof-of-work.

2.2.2.2 Proof-of-work

A proof-of-work is a computation problem that is thought to be difficult to perform but whose result should be easy to verify [27]. It is a requirement for a service user to prove that they have performed a costly action in order do protect the network from attacks such as spam or DDoS (Denial of Service) [32].

To secure the Blockchain against tamper attempts, the network requires proof-of-work to be performed on blocks [26]. Bitcoin's Blockchain proof-of-work requires finding a value that begins with a pre-defined number of zero bits when hashed, and the computational cost required grows exponentially with the number of zero bits required [25]. This kind of solution-verification proof-of-work algorithm is known as partial hash inversion proof-of-work.

Miners provide computational power in order to solve the proof-of-work problem trying nonces until one of them creates the expected result. This process is known as mining and puts miners in the network competing for rewards for solving blocks [26]. When a miner node solves the proof-of-work problem, it broadcasts the block to all nodes, who re-verify the transactions in the block [25].

2.2.2.3 Add a block to the Blockchain

The first block in a Blockchain is known as the Genesis block. As blocks are broadcasted to the network, nodes accept it by using its hash as the previous block when creating a new block [25]. The accepted block is now going to be a Parent block to a subsequent one [26]. The last block added to the chain is called the Blockchain head.

However, if two miner nodes solve a new block at approximately the same time, a fork in the Blockchain occurs, creating a new branch that could persist for several blocks as different miners could believe that either branch is the legitimate one [26]. A fork occurence is depicted in the Figure 2.9.

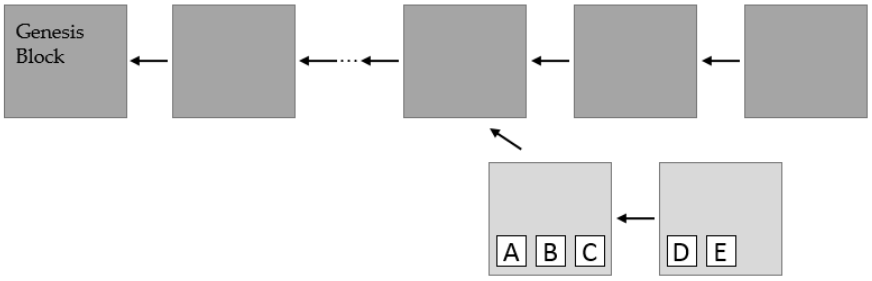


Figure 2.9: Blockchain with the occurrence of a fork. A, B, C, D and E represent transactions.

Eventually, one branch becomes longer and that one is considered legitimate and nodes procede working on it [25] discarding the blocks of the ilegitimate branch. Transactions of the discarded blocks are reintroduced to the network as pictured in the Figure 2.10.

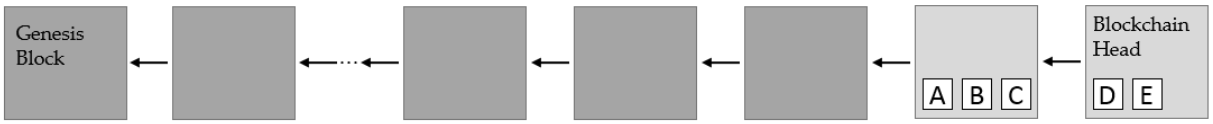


Figure 2.10: Blockchain with the fork solved. A, B, C, D and E were reintroduced to the legitimate chain.

The longer branch is not necessarily the one with the biggest ammount of blocks: length is measured by combining the amount of work required to add all the blocks in the chain [26]. Being so, the longer branch is always the one with the most computational work involved on its creation.

2.2.3 Permissioned Blockchains

Blockchains can be public (permitionless), like the Bitcoin Blockchain, or private (permissioned). In a public network, anyone can be a part of the network and read or write data to the ledger. On permissioned Blockchains participants are known and permissions could be assigned to update or read the information in the chain. This kind of private Blockchains might prove useful when participants come from the same organization and trust between nodes is not an issue.

Companies that build their own private Blockchain can specify the speed of the transactions, create different assets and minimize security methods such as cryptographic signatures and the mining process [33].

2.2.4 Blockchain variations

Bitcoin is the first Blockchain application, allowing the transfer of a digital coin in a decentralized way, but many more are surfacing by tweaking Nakamoto's idea and adapting the Blockchain to comply specific needs and support innovative applications.

One of the biggest supporters of Blockchain is IBM, investing on its own chain in order to simplify customer life, and offering Blockchain-as-a-Service, providing business solutions to clients based on the Blockchain. In addition, IBM is contributing with code to a free, open-source project called Hyperledger [34]. Hyperledger is a collaborative project from the Linux Foundation created to improve Blockchain technology [35].

The Ripple Transaction Protocol is a variation of the Bitcoin Blockchain providing instant low cost international payments for banks and financial services companies. On this protocol, transactions are validated by concensus rather than using a proof-of-work approach as trust is assumed between the participants [36]. The banking group Santander is working on a proof of concept using Ripple for international payments [37], being one of its biggest investors.

Also a variation from the Bitcoin Blockchain, Ethereum is an open-source decentralized platform that allows a network of peers to run Smart Contracts, computer programs that can

automatically execute the terms of a contract [38] (transactions with programming functionalities). The contract is in the public ledger and a triggering event, such as an expiration date, makes the contract execute itself according to the coded terms.

Microsoft is also providing Blockchain-as-a-Service on their cloud platforms enabling developers from customer organizations to deploy their own Blockchain using Bitcoin, Ripple, Ethereum or other protocols [39].

MultiChain is an open platform for building Blockchains, allowing organizations to design their own distributed ledgers, manage permissions, controlling who can send or receive transactions, create blocks or connect to the Blockchain [40].

Some companies opt by using a public Blockchain (essentialy Bitcoin's) in order to create their applications:

- Voatz

  Voatz aims to make election cheaper, more transparent and secure. This system is cheaper than the current american voting system and fraud is virtually impossible [28].

- Tierion

  Tieron creates a verifiable record of any data or business process on the Blockchain, collecting data from several possible sources and generating a Blockchain receipt for each record, providing irrefutable proof that data was recorded in the Bitcoin Blockchain [41].

- Filament

  Filament uses a Blockchain to enable smart devices to securely communicate with each other, uploading data to the cloud via Blockchain Smart Contracts [42].

- Tradle

  Tradle is a startup that hopes to extend the Bitcoin Blockchain to non-financial applications, allowing not only financial trades but any kind of transactions [43].

2.2.5 Common elements to all Blockchains

Even with all the adaptations and different applications using Blockchain, some traits are retained: the Blockchain is decentralized, being distributed across a network of computers and a copy of the entire chain is available to all participants; Each new block needs to be verified and validated by a concensus of the network participants; Transactions are digitally signed with cryptography methods so they can be traced back; Blockchains make use of security mechanisms, such as proof-of-work problems, to make it hard to change records; and all transactions are time-stamped, a useful trait when tracking information. This common workflow to all Blockchains is illustrated in the Figure 2.11.

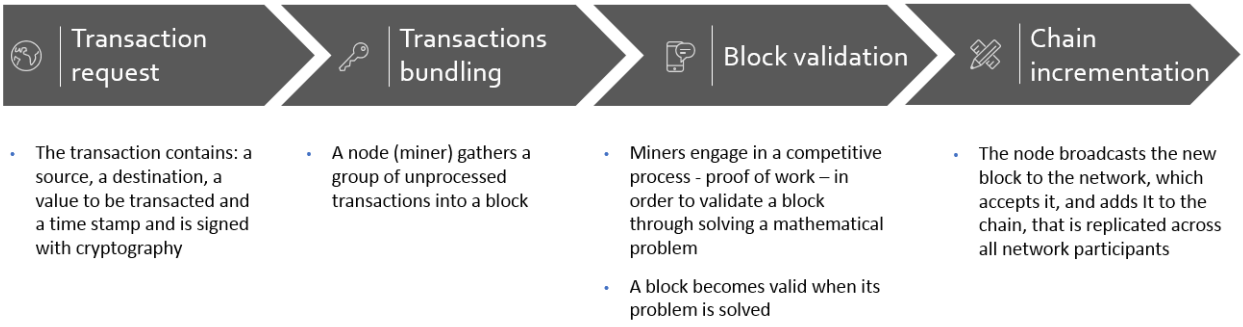


Figure 2.11: Common workflow to all Blockchains.

2.3 Ethereum

Ethereum is a secure decentralised transaction ledger, much as Bitcoin, running a public Blockchain platform and using a cryptocurrency called ether [26]. Ethereum was proposed by Vitalik Buterin in 2013 and funded through a public crowdsale in 2014, intending to extend Bitcoin by providing a Blockchain with a built-in Turing-complete programming language to create contracts that can be used to build decentralized applications [44].

2.3.1 Ethereum ecosystem

Ethereum can be considered a transaction-based state machine, that begins in the genesis state and the execution of transactions transform it into the current final state. This state is made up of accounts and state transitions are direct transfers of value and information between these accounts. The main cryptocurrency used within Ethereum is called ether, and it is used to pay transaction fees [44].

2.3.1.1 Accounts

An Ethereum account is an object represented by a 20-byte address and is composed of four elements:

- Nonce

  A nonce is a counter that is used to guarantee that a transaction can only be processed one time.

- Ether balance

  The ether balance is the value that represents the account's current balance of ether.

- Contract code

  Code that some accounts may have to access and manipulate their internal storage, send messages or create contracts.

- Storage

  Account's internal storage that is empty by default.

There are two types of accounts within the Ethereum network: externally owned accounts and contract accounts. An externally owned account is controlled by a private key, has no code and can send transactions. Contract accounts have a code associated that is executed by transactions received from other accounts, and an internal storage that is acessible and can be manipulated through the code execution [45].

2.3.1.2 Transactions

Following the state-machine description, valid Transactions represent arcs between two states. A transaction is a data package that stores information — a message — to be sent from an externally owned account. All action on the Ethereum Blockchain is triggered by transactions fired from externally owned accounts.

Any cryptocurrency has standard fields that are expected to be contained within a transaction:

- The address of the recipient
- The signature of the sender
- The amount to transfer from the sender to the recipient

Additionally, Ethereum transactions include a few more fields:

Field	Description
Data	Has no function by default; Ethereum virtual machine has an opcode with which a contract can access the data.
STARTGAS Value	Represents the maximum number of computational steps the transaction execution is allowed to take, in order to prevent infinite loops.
GASPRICE Value	Represents the fee the sender is paying per computational step.

Table 2.1: Ethereum transaction fields

There are two types of transactions: message calls and contract creation calls. Contract creation calls result in the creation of new accounts with a code associated with it, and message calls are created by contract accounts to produce and execute a message that leads to the recipient account running its code.

2.3.1.3 Smart Contracts

Smart contracts are structures built on top of a Blockchain that contain value and are unlockable if certan conditions are met. Ethereum Blockchain's Smart Contracts are even more powerful because of the added programmable capabilities [45].

Ethereum Smart Contracts always execute a specific piece of code when a message or transaction trigger its functions, and have direct control over their own data storage and ether balance.

It is possible for Smart Contracts to send message calls to other contracts, and therefore interact with methods and variables that may exist there.

Ethereum contracts are written in EVM code — Ethereum virtual machine code. EVM code is a low-level stack-based bytecode language that consists of a series of bytes, each byte

representing an operation. The operations have access to three data storage spaces: a stack, a memory that is an infinitely expandable byte array that resets after the computation ends, and a long-term storage [45].

2.3.1.4 Blocks and Mining

As in a traditional Blockchain, transactions are grouped into blocks that are chained together, referencing the previous block of the chain, and work as a ledger, recording all the transactions that have occured in the network [45]. However, unlike Bitcoin Blockchain, Ethereum blocks also contain an identifier for the final state in order to avoid the requirements of running all the contract codes again to reconstruct the final state [26]. The advantage of this approach is the fact that network nodes do not need to store the whole Blockchain.

Like all Blockchain technologies, Ethereum uses an incentive-driven protocol of consensus to ensure the security of the network: Miners produce blocks, and a block is only valid if it contains proof of work of a given difficulty.

The block validation algorithm of Ethereum involves the following phases:

1. Verifying that each previous referenced block exists, is valid and satisfies the relation to the present block.
2. Checking the block number, difficulty, transaction list and gas limit.
3. Verifying the proof of work on the block.
4. Validating the transactions within the block, as the accumulated gas used should not exceed the gas limit of the block.
5. Reward the miner with ether.

Ethereum proof of work algorithm is likely to be replaced by a proof of stake algorithm in a future release. The reasoning behind this change lies within the costs of attacking the network: using proof of work algorithms, the costs are the same to what is spent running the system normally and high security can only be achieved at high operating costs. Proof of stake aims to solve this problem as each validator owns some stake in the network and trust is directly related to the amount of value the validator owns. It is called "at stake" because fraudulent actions will result on the loss of that amount [46].

2.3.1.5 Gas and Fees

Ethereum is fuelled by its cryptocurrency, ether, which is used to pay transaction fees.

Gas is the name given to the execution fee that transaction senders have to pay for any operation made in the Ethereum Blockchain. The price of gas is decided by the miners, who can chose to refuse to process transactions if they have a lower gas value than the miner minimum limit.

Block headers contain an attribute named "Gas Limit" that represents the current limit of gas that can be spent per block, and "Gas Used" that declares the cumulative amount of gas used in the transactions of the block. As expected, the "Gas Used" amount cannot surpass the "Gas Limit" value.

"Gas Limit" is also present in the transactions themselves, and represent the maximum amount of gas that should be used in executing that transaction.

2.3.2 Ethereum and combining MDM with Blockchain

Ethereum has programming capabilities through Smart Contracts, which is a requirement for the solution presented in this dissertation to combine MDM with Blockchain, as will be explained in the next chapter. Being so, it makes sense to, instead of creating an entirely new Blockchain, make use of the Ethereum Blockchain. Ethereum allows the creation of private test-nets that can be used to test the proof of concept of this dissertation.

Ideally, a team should focus on developing a Blockchain dedicated to the storage of data instead of using a Ethereum solution that is still heavily connected with cryptocurrency — ether — and imagined to function in public environments, thus requiring fees to process transactions. On the solution presented within this dissertation, all ether and gas features of the Ethereum shall be hidden from the final user.

3

Requirements Elicitation and System Design

The current chapter presents proposed solution to create an MDM Solution using Blockchain.

3.1 Combining MDM and Blockchain

Organizations struggle to achieve a single view of entity across its diverse systems and, even though there might be investments on database systems to model Master Data Management solutions, these systems require big financial efforts — regarding technology and support — and work to set policies of data agreements and enforcements of these policies.

Keeping in mind the objectives of MDM it is possible to find a few points of convergence between MDM implementation styles and Blockchain on which we can support a new solution for MDM using Blockchain.

- MDM requires information to be shared across many different systems

  A Blockchain solution is decentralized and every node of the network has a copy of the Blockchain. Being so, all the systems would be in possession of the same information.

- MDM defines the most trusted and unique version of commonly shared data

  In order to add a new block to the Blockchain, network's nodes have to agree on the validity of the transactions, creating a trusted environment that dismisses the need for an intermediary. As all elements of the Blockchain are referenced by an unique address, from where transaction information can be retrieved, all the systems of the organization should be able to access the same information that, if processed and transformed, could represent the Golden Records of critical data for that organization.

- MDM is an approach that maintains a sustainable, secure and consistent source of data

  Making alterations on blocks that are already part of the chain is virtually impossible, as it requires that all the subsequent blocks to be reconstructed and a majority of network nodes to agree on all the changes. This characteristic results on a safe and stable source of data, ideal to maintain an MDM solution.

The objective of this investigation is to assess if it is possible to achieve a unique vision of client throughout an organization with multiple systems, using an MDM approach implemented with the Blockchain technology, taking advantage of its distributed architecture, safety and consensus methods to create an innovative and disruptive solution.

This chapter presents the theorized solution to prove this concept, describing the Blockchain features that are to be extended and leveraged in order to apply MDM rules to data. The case of study selected to this investigation aims to achieve a unique view of client Golden Records, even though the solution might be extended to other situations in the future if proven viable.

On the solution proposed, each and every department of a given organization that intends to share or access client information is connected to a Blockchain network. Specifically in this case of study, the solution is supposed to function within an organization, the participants of the network are known and approved, so it makes sense to use a private Blockchain solution with defined permissions where trust is not an issue, allowing the simplification of transaction validation and mining processes.

Instead of being a ledger for cryptocurrency transactions, this Blockchain system is going to store client data, and the MDM rules to create the Golden Record for client information are applied the moment the data is inserted into the Blockchain.

Each client record will be accessible through a unique address from which information can be retrieved. This address should be known for all the systems that constitute the network so all systems have access to a unique complete view of a client.

3.1.1 MDM Hub

On an MDM implementation, the MDM Hub is the place where Master Data is managed, stored and kept synchronized with the systems that use that Master Data. In this case, the Blockchain itself might be considered the MDM Hub, as it is accessible to all network participants and the data will be stored within blocks of the chain. In this section it is detailed how the MDM Hub will work in this context.

Within the Blockchain, client information will be managed and stored within Smart Contracts, and those will be the Blockchain elements that mimic the usual MDM Hub functionalities of cleansing, matching and merging information.

Existing in the context of Blockchains, and as previously explained, Smart Contracts are stored and replicated on a distributed platform, containing pre-written logic that can be executed by a network of nodes, meaning that every node of the system will be in possession of all Smart Contracts as soon as they are deployed to the Blockchain network.

The hypothesis presented in this dissertation aims to extend the concept of Smart Contracts, expanding their functionality through the implementation of programming capabilities, in order to enable Smart Contracts to preserve client data and to be the entities responsible of all the programmatic logic inherent to an MDM system, therefore capable of cleansing, matching and merging information, all of this in structures that are replicated and shared on the network, being accessible to all the participants.

The following sections will detail the functional architecture design of the solution, bearing in mind that the organization's systems will constitute the nodes of the Blockchain network and Smart Contracts will be used to create and store client Golden Records.

3.1.2 System actors

The actors have different responsibilities ranging from administrating the network to being end users of the information stored in the system.

Being a Blockchain network the backbone of the system, the system actors will be Nodes of the Blockchain. Nodes are entities that hold a full copy of the Blockchain and are responsible for validating information and passing it on to other systems, as it is a peer-to-peer network. This means that all participants of this solution should have access to all the information stored within the Blockchain, making this MDM implementation truly distributed.

In the solution theorized, two kind of system actors will exist: the administrator and the user. As the name implies, the administrator will administrate the network and define the MDM rules that are to exist on the Data Smart Contract Factory, and the user represents the Organization's systems that wish to access and insert Master Data.

3.1.2.1 Administrator

The system administrator is responsible of managing the network and define the rules used to achieve the Golden Records of the information the MDM system is supposed to manage.

Managing the network consists on defining who can access, introduce or update the information. On the solution proposed, this management includes writing the Smart Contract responsible for validating the data and creating client records, therefore defining the MDM rules that should be applied to the data being inserted.

Being the system administrator is not an one time job, as updates to the system rules and list of users should be part of its responsibilities.

Along with its already mentioned duties, the system administrator also has access to the network and the information within as the other user systems.

3.1.2.2 Users

Organization's systems that wish or must have access to Master Data, either to access it or be participative in inserting new information or updating existent entries, should also be part of the Blockchain network.

This means they should have a public address and a private key, and be a full node of the Blockchain, having a full copy of all the blocks and transactions and therefore having access to the existing client records and being able to add new client records.

In order to make any alterations within the Blockchain, either by adding or updating client records, the users must provide their private key to prove their identity. Data inserted by these actors should comply rules defined by the system administrator.

3.1.2.3 Client records

To have the client information available to all the organization systems, client records will also have to be stored within the Blockchain. This is possible through Smart Contracts: having programming capabilities allows the Smart Contracts to possess data structures and methods to access the data stored within, making these Blockchain entities ideal to store information (in this case, client information).

In this solution, the Smart Contracts that hold client information are called Data Smart Contracts. When created, Data Smart Contracts are broadcast to all the Blockchain network

and a unique address is assigned to each contract. The Data Smart Contract methods are available through the Smart Contract unique address that all network participants must possess, which means all the organization systems will have access to the same information regarding the same entity: in this case resulting on a unique view of client.

Data Smart Contracts are created through the Data Smart Contract Factory, another kind of Smart Contract explained in the next section, and, in this particular solution, possess a structure with the client attributes. The Smart Contract has constructors expecting arguments from the factory to populate this structure, and the information stored is then retrieved through get methods that the Data Smart Contracts have defined. It should be noted that the information stored within Data Smart Contracts has already been processed and worked on by the Data Smart Contract Factory that created the Data Smart Contract.

Data Smart Contracts represent, in this situation, a unique client, meaning that the data within the Smart Contract is supposed to be the Golden Record of that client.

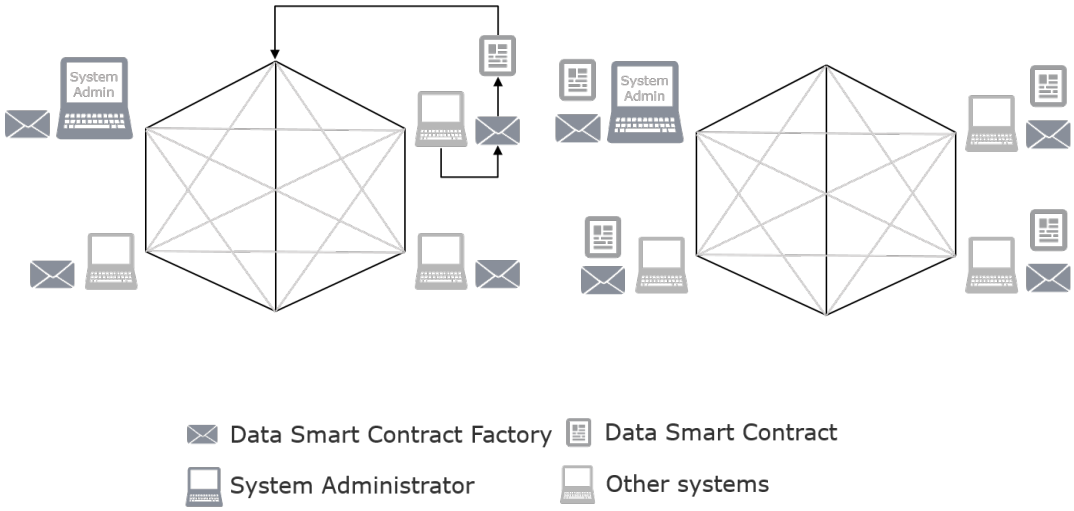


Figure 3.1: Data Smart Contract being created and broadcast to the network

An update do a Data Smart Contract implies the creation of a new Data Smart Contract.

3.1.2.4 Rules of Validation / Creating the Golden Record

Creating this innovative solution, it is important to keep in mind that the Golden Records should be available to all the systems for which the information is critical, and methods should be in place in order to standardize the data structures and validate the information.

Assuming the existence of programming functionality within Smart Contracts, the core element of the solution presented to implement an MDM system with Blockchain is going to be the Data Smart Contract Factory. The Data Smart Contract Factory is a structure with pre-written logic that describes the MDM rules defined in order to achieve the Golden Record. It will have methods that accept, validate and process client information, and then create the final client record. This means that, in this Blockchain solution, Smart Contracts should be able to interact with, and even create, other Smart Contracts.

The Data Smart Contract Factory is created by an administrator, who defines the rules and methods the Smart Contract functions should apply to information that is being inserted.

This special Smart Contract is then broadcast to the Blockchain network and accessible through an address, so any node of the system can feed it information that will be part of the client record. The Figure 3.2 depicts the replication of a Smart Contract on all the participant nodes.

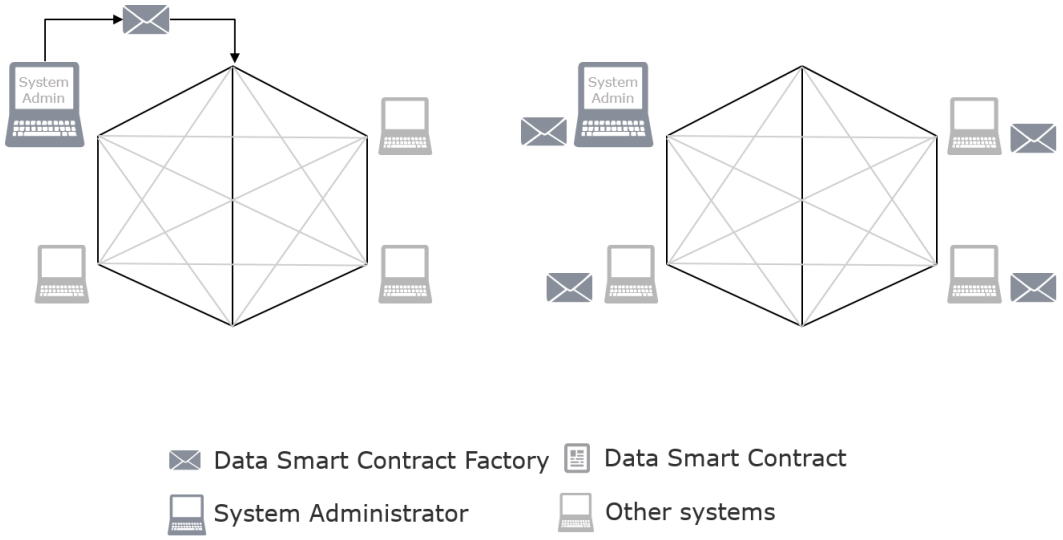


Figure 3.2: Data Smart Contract Factory being created and broadcast to the network

An update to the Data Smart Contract Factory implies the creation of a new Data Smart Contract Factory.

3.1.2.5 Index

As stated in the previous sections, Smart Contracts are accessed through unique addresses. The direct implication of this characteristic is the need for all the system participants to know at any given moment the addresses of the entities they wish to access.

This creates the critical need for the participants to maintain an index system on which the contract addresses are stored and updated as Smart Contracts are created, functioning as an index of addresses. Then, to access a specific client entry, the index has to be queried in order to retrieve the address that corresponds to that client, allowing to retrieve Master Data related with that client.

On the solution presented, the Index is, in itself, a Smart Contract whose address should be known to all the network participants. This Smart Contracts holds mapping structures, storing the addresses of all the other Smart Contracts that exist in the system, should they be Data Smart Contracts or the Data Smart Contract Factory. When a Data Smart Contract or Data Smart Contract Factory is created, its address is added (or updated) to the Index. To access information regarding a client, the Index is called in order to obtain that client address, and only then it is possible to access the client information.

3.1.2.6 Gatekeeper

Another fundamental piece of the solution is the Gatekeeper. The Gatekeeper is another Smart Contract and holds the list of accounts and their respective types, being useful to verify

the author of a new Smart Contracts.

As an example, only administrators can create Data Smart Contract Factories. When a new Data Smart Contract Factory is created, the Gatekeeper checks if the account address of the author is, in fact, an administrator, and the system will only begin to use that Data Smart Contract Factory if it is.

3.1.3 Wrapper

In the context of this dissertation, there will be constructed a wrapper to facilitate the communication between the several components of the the core solution presented before, comprised of Blockchain Smart Contracts, blocks and addresses, and the user.

It should be noted that all the solution regarding the use of Blockchain to create a MDM system is still capable of functioning with direct interactions between the user and the Blockchain.

3.2 Functional Design

This section describes the functional design of the solution proposed to combine MDM and Blockchain. Along this section, the features of the system are going to be explained as the use cases are depicted and the flows of information through the system are detailed. At the end of this chapter, it should be clear what the system does, before dwelling into how the system does it.

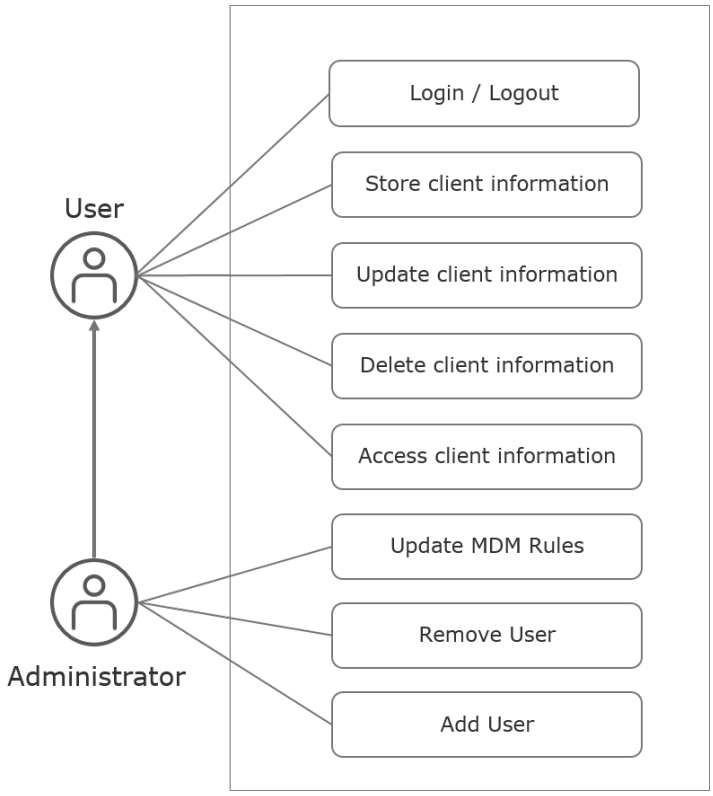


Figure 3.3: Use Cases Diagram of the solution

The Use Cases Diagram depicted in the Figure 3.3 is the simplest representation of the interactions between the users of the system and the system itself. Users (system users) are able to access, store, update and delete client information and the admin (system administrator) is a user that can also manage MDM rules and system users. The tables 3.1 to 3.8 detail the use cases of the system.

Use Case:	Log in/Log out
Actor:	User
Description:	The user stores client information
Pre-conditions:	- A Gatekeeper and an Index already exist in the Blockchain; - User account exists and is registered in the Gatekeeper.
Main course:	To log into the system, the user makes use of the system interface, and introduces its address and password.
Post-conditions:	- The user logs into the system; - The correct interface is shown to the user.

Table 3.1: Use case - Log in/Log out.

Use Case:	Store client information
Actor:	User
Description:	The user stores client information
Pre-conditions:	- User is logged in; - A Data Smart Contract Factory already exists in the Blockchain.
Main course:	To store information about a client, the user makes use of the system interface to input client data.
Post-conditions:	- A new Data Smart Contract is deployed to the Blockchain; - The address of the new Data Smart Contract is added to the Index; - The user receives a message expressing the successfulness of the action.

Table 3.2: Use case - Store client information.

Use Case:	Update client information
Actor:	User
Description:	The user updates information on a client.
Pre-conditions:	- User is logged in; - A Data Smart Contract Factory already exists in the Blockchain; - User knows a client Data Smart Contract address.
Main course:	To update client information, the user makes use of the system interface to search a client, and update its entry.
Post-conditions:	- A new Data Smart Contract is deployed to the Blockchain; - The address of the new Data Smart Contract is added to the Index; - The user receives a message expressing the successfulness of the action.

Table 3.3: Use case - Update client information.

Use Case:	Delete a client entry
Actor:	User
Description:	The user deletes client information.
Pre-conditions:	- User is logged in; - User knows a client Data Smart Contract address.
Main course:	To delete a client entry, the user makes use of the system interface to search a client, and delete its entry.
Post-conditions:	- The address of the new Data Smart Contract is removed from the Index; - The user receives a message expressing the successfulness of the action.

Table 3.4: Use case - Delete a client entry.

Use Case:	Access client information
Actor:	User
Description:	The user accesses a client record.
Pre-conditions:	- User is logged in; - User knows a client Data Smart Contract address.
Main course:	To access client information, the user makes use of the system interface to search for a client.
Post-conditions:	- The client record is shown to the user.

Table 3.5: Use case - Access client information.

Use Case:	Update MDM rules
Actor:	Admin
Description:	The admin updates the active MDM rules.
Pre-conditions:	- Admin is logged in; - There is a library of possible rules to apply to the Data Smart Contract Factory.
Main course:	To update the active MDM rules, the admin makes use of the system interface to select the new Data Smart Contract Factory rules.
Post-conditions:	- The Data Smart Contract Factory address is updated in the Index; - The admin receives a message expressing the successfulness of the action.

Table 3.6: Use case - Update MDM rules.

Use Case:	Add new user
Actor:	Admin
Description:	The admin creates a new user account.
Pre-conditions:	- Admin is logged in;
Main course:	To create a new user account, the admin makes use of the system interface to input the user alias and password.
Post-conditions:	- The Gatekeeper is updated with the new account; - The admin receives a message expressing the successfulness of the action.

Table 3.7: Use case - Add new user.

Use Case:	Remove user
Actor:	Admin
Description:	The admin removes a user account.
Pre-conditions:	- Admin is logged in;
Main course:	To create a new user account, the admin makes use of the system interface to input the user address to remove.
Post-conditions:	- The Gatekeeper is updated, removing the account; - The admin receives a message expressing the successfulness of the action.

Table 3.8: Use case - Remove user.

3.2.1 Functional Requirements

Functional requirements specify the criteria that should be used to evaluate specific behaviours or functions of the system. These are the functional requirements of the system proposed on this dissertation:

- **The system interface should allow the user to log into the system as an admin or a user.** All Blockchain interactions require an account that is defined by an address. In order to use that account, the user must know its password. When a user logs into the system, all subsequent transactions are from that account.
- **The system interface should allow the user to log out from the system.** As the system should allow a user to log in, when a user is done with the system, the possibility to log out should also exist.
- **The system should allow the storage of client information.** Because the main objective of the solution is to achieve a single vision of client across an organization, the system must be able to accept client information and store it in the Blockchain.
- **The system should allow the update of client information.** As the system holds client information, it should permit to update and modify some of the data.
- **The system should allow to delete client information.** As a system that holds client records, there should be a way to remove an entry from the system. Even though data cannot be removed from the Blockchain, a deleted entry should not appear to an user.
- **The system should allow the user to search for client information.** It is crucial for a system that holds client information and wants to attest the application of Master Data rules, that the user is able to search for client records.
- **The system should allow an administrator to manipulate MDM rules.** To implement an MDM system, MDM rules should be customizable by the system administrator. These rules should be enforced when a new client entry is added to the Blockchain.
- **The system should allow an administrator to add new users to the system.** To guarantee that the system is only used by authorized entities, the administrator should be able to create new accounts.
- **The system should allow an administrator to remove users from the system.** To guarantee that the system is only used by authorized entities, the administrator should be able to remove existing accounts.

3.2.2 Non-Functional Requirements

Non-Functional requirements specify the criteria that should be used to evaluate the operation of a system. These are the non-functional requirements of the system proposed on this dissertation:

Concurrent Access	Description
Read / Read	More than one system participant access the same record; Both participants have access to the same information.
Write / Read	More than one system participant access the same record; At least one of them is making changes on the record; If the read access is made after the changes, the contract shown should have the updated information.
Write / Write	More than one system participant access the same record; More than one participant is making changes on the record; The first write will be discarded, even though its information is still safe in the Blockchain.
Delete / Write	More than one system participant access the same record; More than one participant is making changes on the record; At least on participant is trying to delete de client entry; If a write access is made after a delete, the contract will not appear to the users.
Delete / Read	More than one system participants access the same record; At least one of them is deleting the record; If the read access is made after the delete, the contract will not appear to the user.

Table 3.9: Non-Functional requirements related with concurrent accesses

Quality of the Solution	Description
Response time	Accesses to client information should be almost immediate.
Number of participants	The network should be able to handle a big and scalable number of users.
Availability	If a peer of the network stops working, the information should still be available to the other participants.
Consistency	All participants should be able to access the same information about the same client.

Table 3.10: Non-Functional requirements related with the quality of the solution

3.3 Technical Design

This section describes the technical design of the solution proposed to combine MDM and Blockchain. Along this chapter, the components of the system are explained as well as their interactions, providing a clear view of how the system operates in order to comply its features.

3.3.1 Architecture Definition

The physical network is composed by computers interconnected that together form a peer-to-peer environment, and when a new node is added to the network, it has to be added as a peer by at least another node that is already part of the network. The presence of a new node is announced to the rest of the network so all the nodes include the newcomer to the pool of existent peers. A new node also has to download from its peers the full extent of the Blockchain or it will not be able to access Smart Contracts, and therefore the Master Data.

On this physical network runs the software that allows the users to interact with the system and use it as an MDM solution.

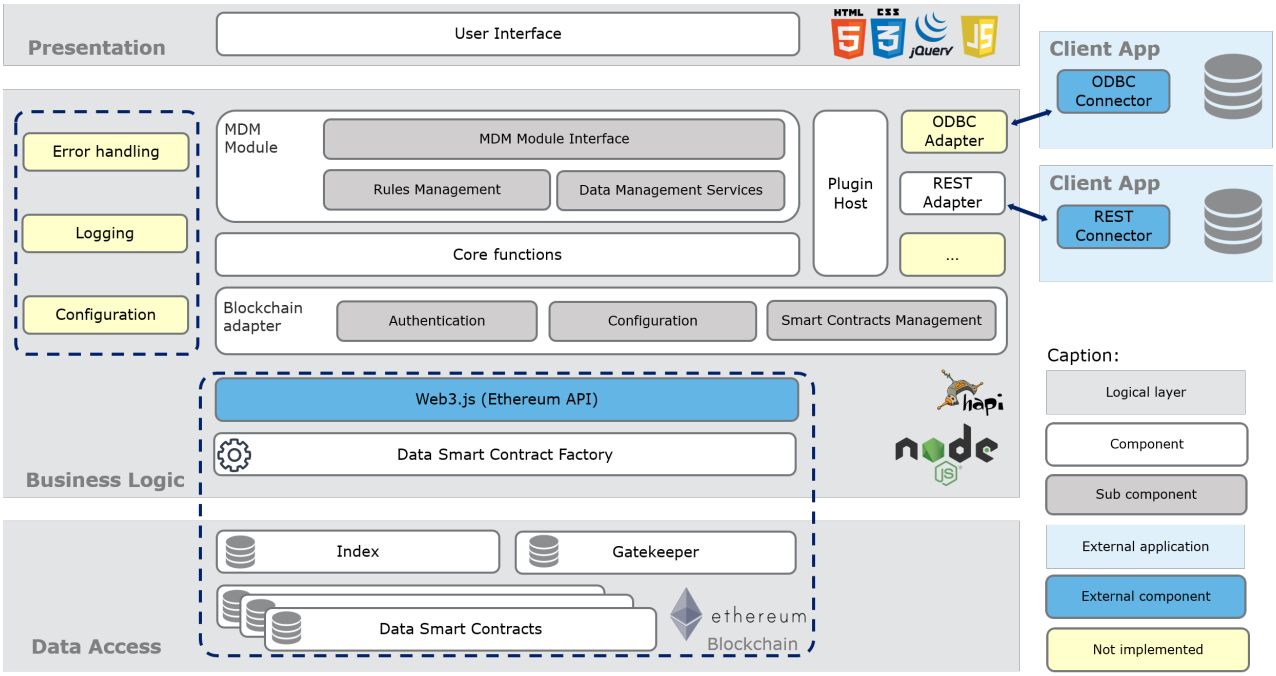


Figure 3.4: Architecture diagram of the solution

The Figure 3.4 displays the Architecture diagram of the system and divides the solution into three logical layers: the Presentation layer, the Business Logic layer and the Data Access layer. The layer details are explained in the next sub sections.

3.3.1.1 Presentation layer

The presentation layer is responsible for the communication between the users and the application, displaying information to the users by encompassing an interface that allows requests to the system.

The components of the Presentation layer are detailed in the Table 3.11.

Component	Description	Development Tools
User Interface	The User Interface is the component the user interacts with. It is the place that allows the user to insert data he wishes to insert in the Blockchain or retrieve existing records. The system administrator manages the network and the functions of the Data Smart Contract Factory through the User Interface.	HTML5, CSS, Javascript, JQuery

Table 3.11: Components of the Presentation layer

The user interface was developed in HTML5 and CSS, making use of jQuery and Javascript.

- HTML5

  HTML is the standard language used to structure and present web content. HTML5 is the fifth version of HTML.

- CSS

  CSS is a language used to describe the presentation of content written in a markup language such as HTML.

- Javascript

  Javascript is the programming language for web pages.

- JQuery

  JQuery is a Javascript library that simplifies the Javascript programming.

3.3.1.2 Business Logic layer

The business logic layer is where the parts of the system that process commands, make logical decisions and coordinate the system in general are. In this solution, the business logic layer is composed of an MDM module, the Core functions module, and the Blockchain adapter that allows communication with the Ethereum API and therefore with the Data Smart Contract Factory. On this layer, one can also find the components for Error handling, Logging and general Configuration, and finally the Plugin Host.

The components of the Business Logic layer are described in the Table 3.12 and going to be developed with Node.JS and Ethereum.

Node.js is an open-source, cross-platform runtime environment for developing server-side Web applications, with many of its modules being written in javascript.

Ethereum is a Blockchain platform with programming capabilities.

Component	Description	Development Tools
MDM Module	Module that is responsible for the interaction between the system and the user interface, deals with the Master Data Management rules that are to be enforced in the implementation plus the data that gets in or out of the system.	Node.JS, Hapi JS
MDM Module Interface	Server-side component consisting of an interface with exposed endpoints in order to communicate with the presentation layer via the transfer of information.	Node.JS, Hapi JS
Rules Management	Unit of the system where the possible rules the MDM system can possibly enforce are defined.	Node.JS
Data Management Services	Group of functions used to manage the data within the system and specify the active attributes of the Data Smart Contracts.	Node.JS
Core Functions	Component on which the core functions of the system are defined, merging the information of the components above to deploy contracts to the Blockchain.	Node.JS
Blockchain Adapter	Module of the system responsible for the communication between with the Blockchain.	Node.JS, Ethereum
Auth.	Component that calls methods of authentication, making use of the public addresses of the participants and their private keys.	Node.JS, Ethereum
Blockchain Configuration	Unit responsible for the configuration of the server in order to connect with the proper Blockchain, crucial to further interaction.	Node.JS, Ethereum
Smart Contract Management	Group of methods used to create and manipulate Smart Contracts.	Node.JS, Ethereum
Plugin Host	Hosts plugins, such as a REST adapter, in order to make the solution apt to outside communication, scalable and open to improvements, extending the solution's functionalities.	Node.JS, Hapi JS
Web3.js Ethereum API	Javascript API compatible with Ethereum that implements a stateless, light-weight remote procedure call protocol to talk to an Ethereum node from inside a Javascript application.	Ethereum, Javascript
Data Smart Contract Factory	Smart contract with pre-written login that describes the rules defined in order to insert data into the Blockchain after the application of those rules. Possesses methods that accept, validate and process data. Creates Data Smart Contracts.	Ethereum

Table 3.12: Components of the Business Logic layer

3.3.1.3 Data Access layer

The Data Access layer provides the access to the data stored in the Blockchain. Its components are described in the Table 3.13.

Component	Description	Development Tools
Data Smart Contracts	Smart contract with constructors expecting arguments from the Data Smart Contract Factory to populate this structure, and the information stored is then retrieved through methods that Data Smart Contracts have defined.	Ethereum
Index	Smart contract where other Smart Contract addresses are recorded and updated.	Ethereum
Gatekeeper	Smart contract where other system user accounts are stored and with methods to check if a given address is a user or an admin.	Ethereum

Table 3.13: Components of the Data Access layer

3.3.2 Components Diagram

The components diagram is a useful way to describe the components of the solution and how they interact with each other.

The Figure 3.5 represents the components diagram of the solution proposed and helps to identify the major components that interact with each other within the system.

The Blockchain, where the Master Data is going to be created and stored, is composed of: the Data Smart Contract Factory that creates Data Smart Contracts, the Index and Gatekeeper Smart Contracts and the Ethereum API Web3.js. As stated in previous chapters, the purpose of the Data Smart Contract Factory is to create Data Smart Contracts and transform data to achieve the Golden Records. Data Smart Contract Factory and Data Smart Contracts have interfaces through which their methods and information are available to outside the Blockchain via the Web3.js API.

The Blockchain adapter interacts with the Blockchain via the Web3.js API and provides callable methods related with Authentication, Configuration of the Blockchain, and management of transactions and Smart Contracts. These methods are available to the Core functions component.

The Core functions component interacts with the Blockchain adapter, providing its functionalities to the Plugin Host and the MDM Module. This component acesses the local Index to retrieve addresses used to insert and retrieve data to and from the Blockchain.

The MDM Module offers functionality regarding the Master Data Management objectives to the system, dealing with the rules management and data management. It accesses the Core functions methods and provides information to the User Interface and to the Plugin Host, via a Web API.

Finally, the User Interface is the component the user interacts with. Following a user request, the User Interface component calls methods from the MDM Module and awaits responses that are to be displayed in the interface.

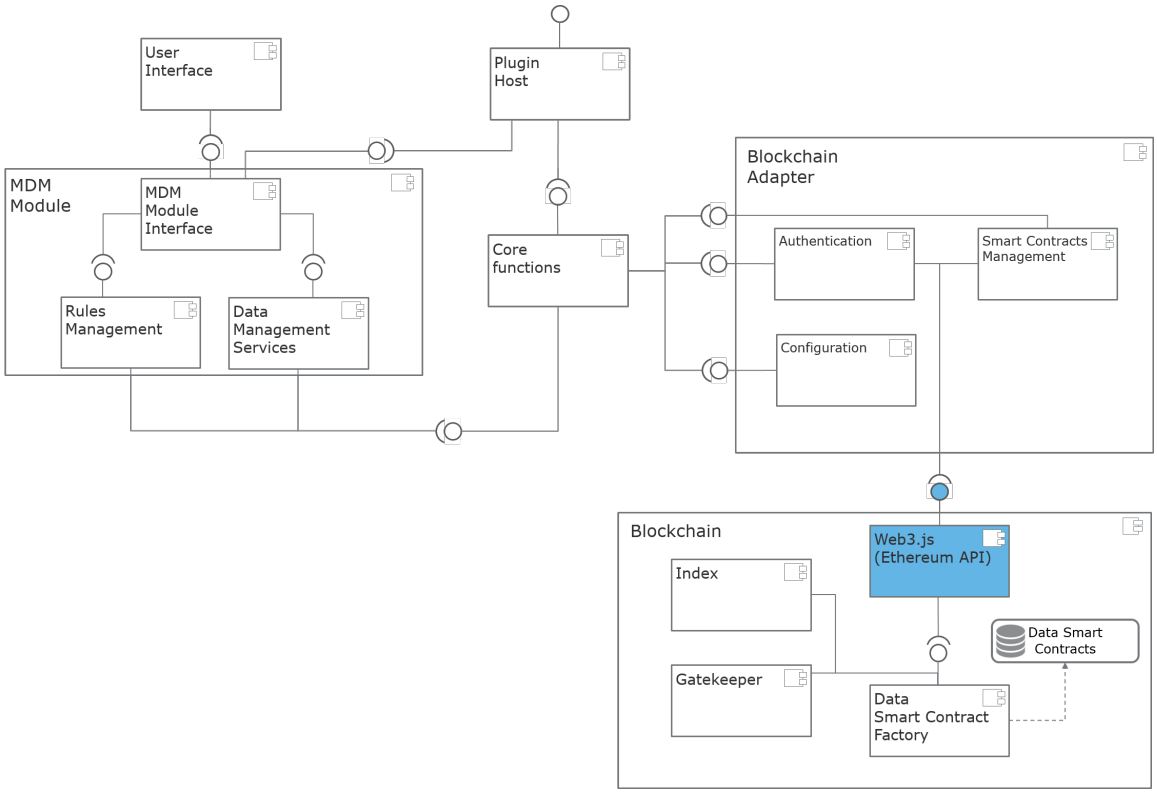


Figure 3.5: Components Diagram of the solution

3.3.3 System Interactions

In order to better understand how the various components of the solution are supposed to interact with each other, this section will address each of the previous detailed use cases and present an individual diagram that displays how the system is supposed to respond to comply the user demands.

- Log in

  As depicted in the Figure 3.6, to log into the system, the user interacts directly with the system interface. After inserting its address and password, the system queries the Index in the Blockchain to discover the address of the Gatekeeper that is then accessed to verify the validity of the account and to check which type of account (user or admin) is trying to login. Finally, the Interface displays a new page according to the type of account that logged in.

- Store client information

  The system interactions that will occur when a user is trying to store client information are shown in the Figure 3.7. First, the user must be logged in the system and access the interface through which client information will be written. The system queries the Index Smart Contract in the Blockchain to discover the address of the current Data Smart Contract Factory and the client information is sent to that Data Smart Contract

Figure 3.6: Flow of actions regarding the log in to the system.

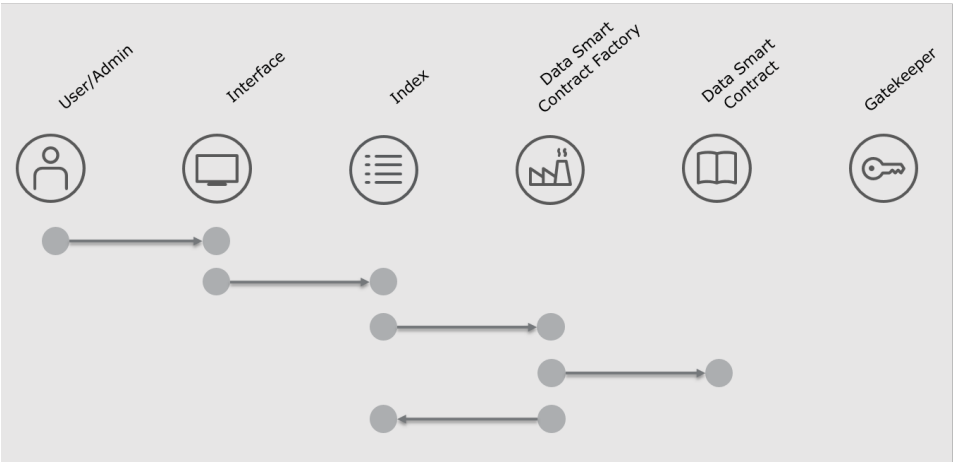


Figure 3.7: Flow of actions regarding the addition of a new client record to the system.

Factory. When a new Data Smart Contract is deployed by the Factory, its address is added to the Index Smart Contract to further utilizations.

- Update client information

  The Figure 3.8 shows the interactions within the system needed to update a Client record. As expected, the user must be logged in. First, the user uses the interface to search for a client record. The system queries the Index Smart Contract to find the Data Smart Contract related with the client the user is trying to update. That client record is then displayed in the interface and the user can input new client information. To deploy it, the system must query the Index Smart Contract to discover the Data Smart Contract Factory interface, through which the new Data Smart Contract regarding that client will be deployed, finally updating its address in the Index Smart Contract.

- Delete a client entry

  To delete client information the user must be logged in. First, the user uses the interface

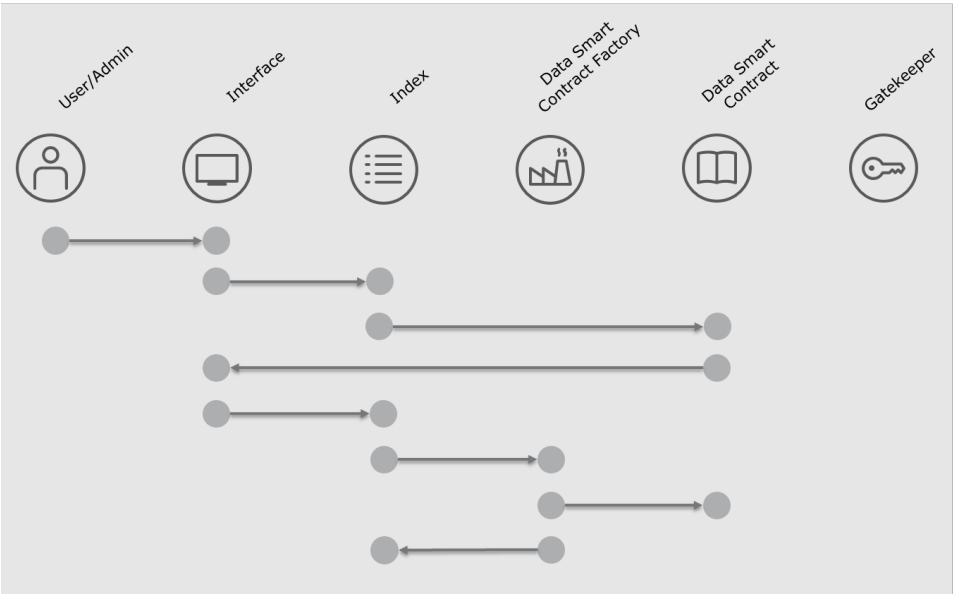


Figure 3.8: Flow of actions regarding the update of a client record

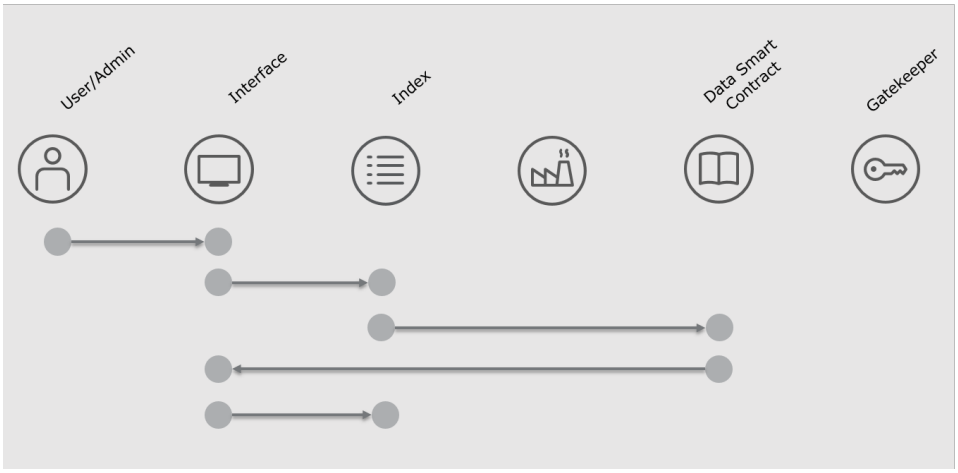


Figure 3.9: Flow of actions regarding the deletion of a client record

to search for a client record. The system queries the Index Smart Contract to find the Data Smart Contract related with the client the user is trying to update. That client record is then displayed in the interface and the user can choose to delete this record. To delete it, the system must query the Index Smart Contract to remove this specific client address from its list. This sequence of actions is shown in the Figure 3.9. – alterar imagem.

- Access client information

  As depicted in the Figure 3.10, to search and access client information, the user must be logged in, and use the interface to search for a client. The system queries the Index Smart Contract to find the addresses of the clients that correspond to the search input

Figure 3.10: Flow of actions regarding the access to client information.

and displays the Data Smart Contracts regarding those addresses.

- Update MDM rules

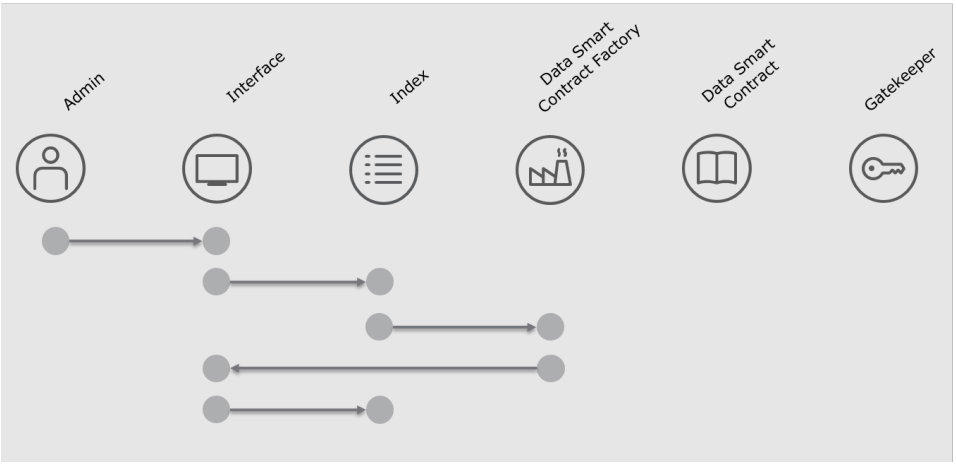


Figure 3.11: Flow of actions regarding the update of the active MDM rules.

Updating the MDM rules of the system means creating a new Data Smart Contract Factory where the MDM rules to enforce are defined. So, as shown in the Figure 3.11, the admin must be logged in and use the interface to create the new Data Smart Contract Factory. First, the system will query the Index to discover the address of the current Factory to present the current rules to the admin. The admin will then alter the rules and a new Data Smart Contract Factory will be deployed. As expected, the address of the newly created Smart Contract is updated in the Index Smart Contract for further utilizations.

- Add new user / Remove user

  Finally, to update the list of users and administrators that can use the system, the administrator must be logged in and use the interface. The system will query the

Figure 3.12: Flow of actions regarding the control of the system participants.

Index interface to retrieve the Gatekeeper Smart Contract address. The Gatekeeper will return the existing users and administrators of the system and the administrator, through the interface, will be able to add or remove new users/administrators to the list on the Gatekeeper Smart Contract. This sequence of actions is depicted in the Figure 3.12.

4
Implementation

Throughout this chapter, the implementation of the solution described in the previous sections of this dissertation is explained, along with the decisions backing off this implementation. It starts with an overview of the implementation, and then details each main component of the solution, to show how it is possible to achieve what was proposed.

4.1 Overview

One of the main requirements was to keep all the MDM logic inside the Blockchain, in order to create an MDM solution using Blockchain, not only as a support system but as the core of the solution. The solution implemented achieves this, granting that the MDM Solution is totally implemented in the Blockchain, only with extra user friendly additions that were created out of the chain and are complements to the solution in order to facilitate its utilization and extend its functionalities.

The Figure 4.1 summarizes that characteristic of this solution, exposing which elements were implemented within the Blockchain: the Index, the Data Smart Contract Factory, the Data Smart Contract and the Gatekeeper. As explained in the Chapter 3, these four types of Smart Contracts comprise the MDM logic, dealing with merging and matching rules, cleansing rules, record storing and data access, providing an unique view of client throughout an organization.

In the end, it is possible for the user to make use of this MDM Solution, just by having in his possession the address of the Index Smart Contract and knowledge on how to interact directly with the Blockchain.

Instead of developing an entirely new Blockchain to suit the needs of the hypothesis, Ethereum was chosen to create an MDM solution. The basis for choosing Ethereum to develop this solution is explained in the Section 2.4.2 of this dissertation and can be outlined by the need to have Smart Contracts with programming capabilities, in order to develop the Data Smart Contracts, Data Smart Contract Factories, Index and Gateway.

For the sake of being a proof-of-concept, one can use Ethereum to create a private

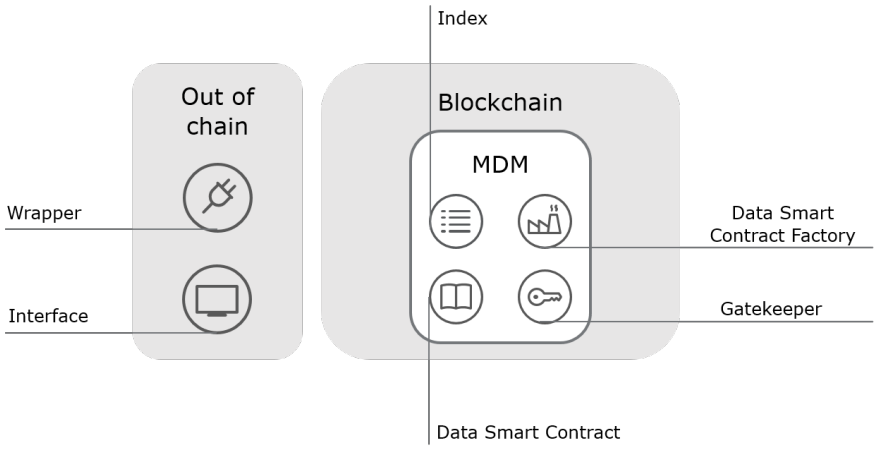


Figure 4.1: Division of the main components. MDM logic exists only inside the Blockchain while the Interface and a Wrapper are created out of the chain to aid the utilization and extend the functionalities of the solution.

Blockchain network and choose to ignore its monetized features and needs, by creating accounts with huge amounts of ether and very low gas costs for transactions.

Being so, when creating the genesis block for the Blockchain, it was defined what Ethereum knows as the "coinbase account" (the prime administrator of the network) and supplied this account with a very high value of ether. When other accounts are created, it is transferred some of the ether of the coinbase account to the newly created. It is also within the genesis block that the value of gas needed per transaction is defined, which in this implementation was a very residual amount.

To create and interact with a private Ethereum network, it is necessary to install Geth. Geth is a command-line interface that allows users to run Ethereum nodes and use Web3.js to interact with the Blockchain. When running a node, it is necessary to discriminate the genesis block and the directory on which the Blockchain is to be stored. There are also several flag options that might be used in order to allow external interactions or to define a maximum number of nodes, for example.

In order to set up an Ethereum Blockchain network composed of two or more nodes, each of the nodes need to possess the same genesis block file, so this file (or its contents) has to be shared among the participants. The first node of the network has to allow the existence of peers.

For a node A connect to another node B, it has to share its enode (Ethereum node) address, which will be used by the node B in a specific Web3.js command to add peers. The new node will then begin downloading the Blockchain to a given directory and is able to interact with the Blockchain.

It is possible to import a file within the creation of the node where the peers are declared, to simplify the network of peers creation on larger networks.

4.2 Smart Contract Programming

To deploy the MDM logic into the Blockchain and create a full MDM solution, it was necessary to develop and program 4 different Smart Contracts. Ethereum programming capabilities allows this, providing a few different languages such as Solidity or Serpent. While Serpent is more similar to Python, Solidity is closer to Javascript and is more popular, granting more user support and overall programming examples. These are the reasons why Solidity was chosen to program the Smart Contracts to implement this solution.

4.2.1 Solidity

Solidy is one of the high-level languages that is used to create code for the Ethereum Virtual Machine, allowing the creation of Smart Contracts with pre-written logic and decentralized applications. It is publicized as being similar to JavaScript but presents several limitations, which will be detailed in the next section, related to challenges.

After writing a contract in Solidity, it is necessary to compile its code, and this was achieved using an installed compiler — solc — but could be done with online compilers or other existent solutions.

Using Solidity, four different Smart Contracts where developed: Data Smart Contracts, Data Smart Contract Factory, Index and Gateway.

4.2.1.1 Data Smart Contracts

Data Smart Contracts, named DSC, contain a fixed data public variable, of the type address, that is supposed to contain the address of the next DSC related to that Client record. This means that, for the first client record, this value will be null, but as soon as the client record is updated and a new DSC is deployed, this value will be updated in the first DSC to contain the address of the updated DSC.

In terms of variables, the DSC's also contain an integer to declare the number of active attributes (e.g. name, phonenumber, email). This integer has no impact but is helpful for the interface to present information based on the number of active attributes.

Finally, regarding the data contained in the DSC's, there is also a variable with the number of fields and depends on the active attributes set by the Data Smart Contract Factory.

Regarding the methods of the Smart Contract, there is the constructor to populate the data fields as the DSC is created, a function getClient to retrieve the information recorded on the Smart Contract and a function setAddress to update the address when the client record is updated, as explained before.

4.2.1.2 Data Smart Contract Factory

The Data Smart Contract Factory is the most complex Smart Contract of the solution, even though a big part of its variables are simply used for auxiliary purposes by the wrapper and have no impact in the Blockchain solution as a whole, or to help with the manipulation of strings, a big challenge over which is dedicated a section further in this document.

It is also complex because a big portion of the code is not set in stone, as it depends on the active MDM rules and attributes.

However, some of the code of this contract is not dependent of those factors: this contract contains the address of the index Smart Contract, the list of active attributes and rules, and the address of the most recent Data Smart Contract Factory when there is one.

The constructor of this contract updates the address of the Data Smart Contract Factory on the Index Smart Contract, a function addClient, with variable arguments representing the active client attributes, and creates a client after the application of the active rules, and a method updateClient that works in a similar way but where matching and merging rules, if they exist, are not applied. Both these client related functions also update or create entries in the Index Smart Contract.

Finally there is a function to set the new factory address if there is an update in the active rules or attributes and a new factory is deployed.

4.2.1.3 Index

The Index Smart Contract is a core piece of the solution as it holds the addresses to all Smart Contracts that make part of the solution. It contains the address to the Data Smart Contract Factory and to the Gatekeeper, plus a mapping of Clients (with the client addresses). For auxiliary purposes, it also contains an integer that represents the number of existing client records.

Regarding these contract methods, there is a function addToIndex, that is called when a new DSC is deployed regarding a new client (not an update), and adds the DSC address to the mapping structure and increments the counter; a function RemoveFromIndex, that does the opposite; updateAddress, when it is supposed to replace an address of a client and not create a new entry; a function SetFactory, that verifies, using the Gatekeeper, if the user trying to call the function is an administrator, and if it is, updates the factory address, and finally, a retrieveClientAddress used to retrieve the address of a specific client from the mapping structure.

4.2.1.4 Gatekeeper

Finally, the Gatekeeper Smart Contract deals with recording the users and the administrators, and has methods to verify if a given address corresponds to a user or an administrator.

This contract has two mappings, one for users and other for administrator, and two counters to record the number of entries in both mappings, for auxiliary purposes. When the Gatekeeper is created, the address of the user that deployed that Smart Contract is automatically added to both lists as an administrator.

The Gatekeeper Smart Contract is then composed with methods: the method addUser, that adds a new user to the mapping if it is an administrator trying to add the user, the addAdmin, to add new administrators to the list of administratorss, the removeUser, to remove users from the list of users, the method removeAdmin, to delete an administrator from the list of administrators, and finally the methods isAdmin and isUser, to verify is an address is an administrator or if it is a user.

4.2.2 Challenges

Solidity is not prepared for everything, either for safety reasons or due to the lack of development, and presents some setbacks when developing Smart Contracts. For instance, it does not do well with variables without a fixed length such as strings, resulting on the

lack of pre-built methods of string manipulation. It is a true challenge to count the number of characters of a string, compare two strings or make modifications on strings. Also not very clear is how two different contracts interact and which types of variables can be passed between contracts. Both these aspects are crucial to a solution that is based on the interaction of contracts and manipulation of strings to apply cleansing or matching rules, and were solved during the implementation of this proof-of-concept work.

4.2.2.1 String manipulation

To surpass the challenge mentioned, it was necessary to find another way to store strings and manipulate its content. The solution found was to transform the string into an array of bytes. This way, the variable (the array) would have a fixed size and therefore capable of being passed between contracts. The figure 4.2 displays the "life-cycle" of the string as it is stored into a Smart Contract and then retrieved.

Figure 4.2: Visual representation of what happens when storing a string to apply cleansing or matching rules.

It was necessary to develop a library of structures and methods related to string manipulation, by transforming the generated array of bytes, according to the library of MDM rules predefined.

- **String to upper, lower or proper case**

  In order to implement this functionality, it was necessary to create two Solidity mappings, one containing the lower-case alphabet letters and respective counterparts, and another containing the upper-case letters and its respective counterparts. This was essential, as even having a byte of arrays where it was possible to retrieve the character value, it was not possible to use that value in additions or subtractions to make alterations and a full substitution is required.

  This way, if the method in place is to make sure that the attribute is always lower case, the function will check, byte by byte, if the character is upper case and substitute it with its lower case counterpart, as assigned in the mapping.

- **Add a space every 3 characters**

  This function makes use of the byte array and pushes a space character every 3 bytes.

- **Hamming distance calculator**

  To implement the matching rules, where, as defined by an administrator in the Data Smart Contract Factory it is necessary to match attributes if they are similar to a certain degree, it was necessary to create an Hamming Distance calculator method that makes use of the array of bytes to check how different two strings are.

Methods to transform strings into an array of bytes and vice-versa were also constructed to support all the other functions.

4.2.2.2 Interaction between Smart Contracts

Interaction between contracts is necessary if we want a Smart Contract to create other Smart Contracts, for example when the Data Smart Contract Factory creates multiple Data Smart Contracts, or if some methods of a Smart Contract need to access methods of another Smart Contract, as when a new Data Smart Contract is deployed, the Data Smart Contract Factory calls the Index method that adds the new address to the mapping.

To interact with a contract from outside of the Blockchain, it is necessary to know the address of the Smart Contract. Between contracts this is more complex and requires that in the Solidity code to be compiled, all the code from all the contracts that interact with each other should be compiled together. As an example, when compiling the Data Smart Contract Factory, the code of this Smart Contract should include the code for the Data Smart Contract, the Index and the Gatekeeper, but to compile the Index, there only needs to be included the Gatekeeper code.

After this point it is possible to call the contract methods and public variables as objects (e.g. DSC Client = new DSC();).

4.3 Wrapper

To facilitate the utilization of this MDM in Blockchain solution, a multi tiered wrapper was developed using Node.js, making it easier to integrate with other applications and to extend the functionality. This wrapper encapsulates and abstracts the Blockchain specifics to the end user or programmer.

The wrapper was developed with Node.js due to the fact that communication between the wrapper and Ethereum's Web3.js would become trivial, plus Node.js allows the installation of modules that would prove useful in the project development:

The module "Underscore" allows the use of templates and their further filling. It was used to design the Smart Contracts and the web pages structure, to allow the filling with code dependent on the current needs of the user, such as MDM rule choices or different pages for the administrator or the user.

The module "fs" allows the access to the local file system, a requirement in this project as the address of the Index Smart Contract needs to be stored locally by the nodes.

One specific module of the Wrapper is the Plugin Host. The Plugin Host exists to extend the functionality of the overall solution. The desired plugins should be placed within a specific directory that, when its content is changed (e.g. a new plugin file was added to the directory), the Plugin Host tries to run that file, making it necessary that the external plugin follows specific rules, such as having a method "start()".

For the sake of testing the Plugin Host, some solution's methods were made available:

- Get Client
- Add Client
- Update Client
- Delete Client

Allowing plugins to search, add, update or delete client records from the Blockchain.

Finally, it was created a REST Adapter to test this extended functionality.

To create the REST Adapter and the server on which the user interface runs, it was used Hapi Js: a framework for building applications and services, installed as an Node.Js module, that allows the creation of a simple server and the definition of routes to respond to user queries.

4.4 Interface

A Web-based User Interface was created, which fully demonstrates the MDM capabilities and system management. The interface displays at all moments the alias and address of the current logged user.

Through the UI it is possible to:

- Log into the system

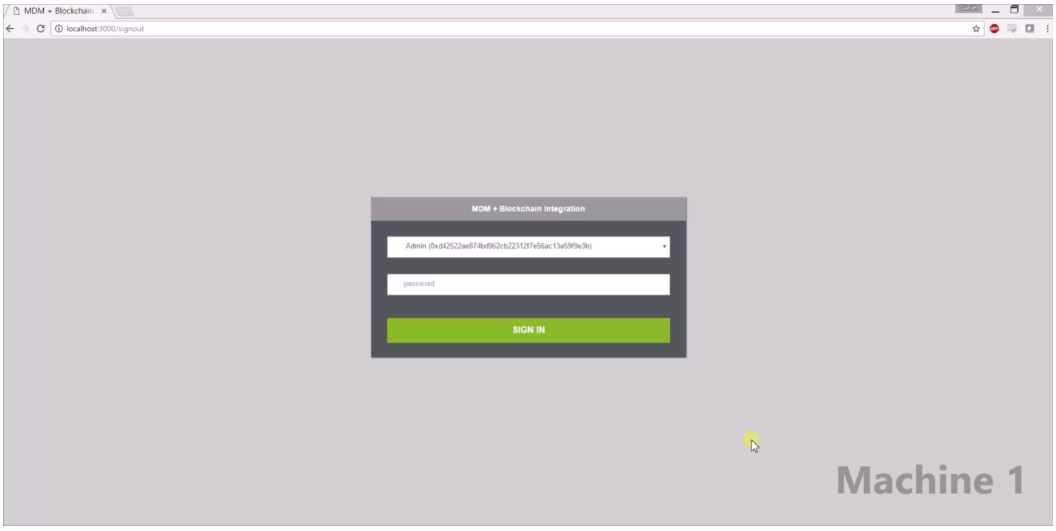


Figure 4.3: Interface to log into the system.

- Insert client records

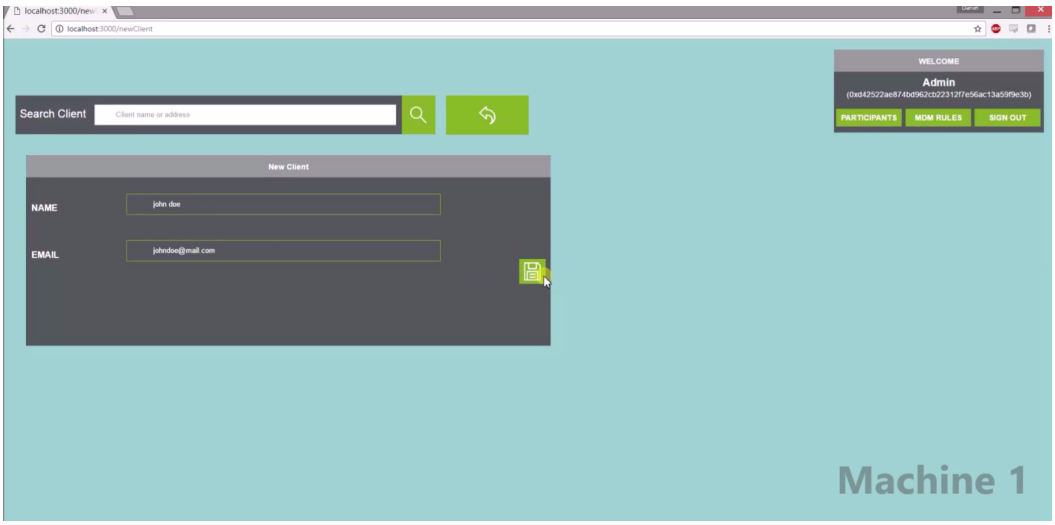


Figure 4.4: Interface to add new client records in the system.

- Search, update and delete Golden Records

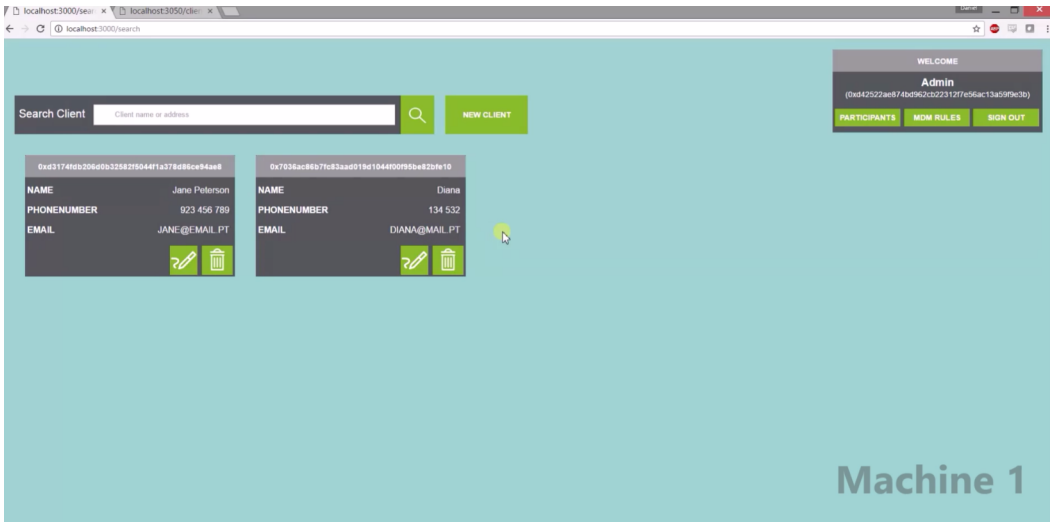


Figure 4.5: Interface to search, update and delete Golden Records in the system.

- Maintain the cleansing, matching and merging rules

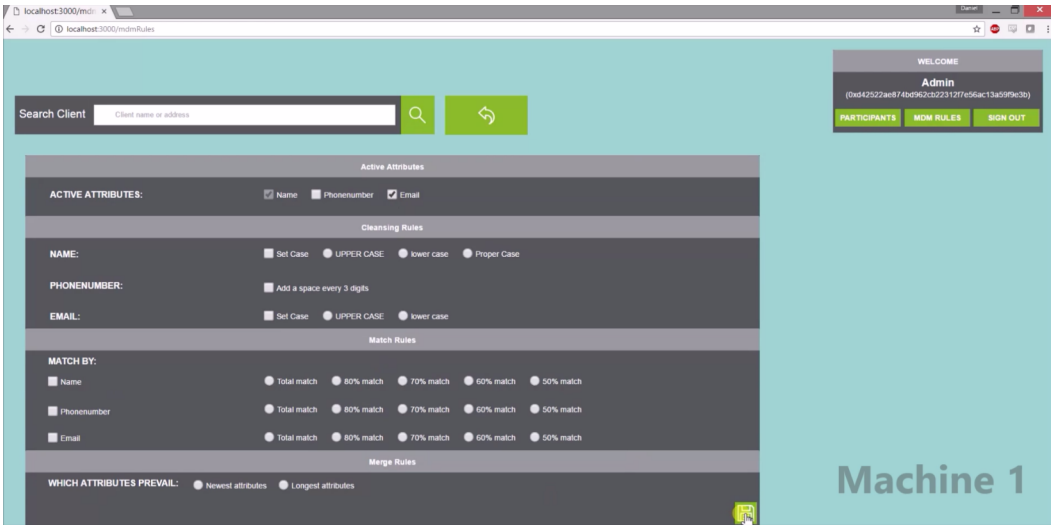


Figure 4.6: Interface for selection of the MDM rules to enforce.

- Manage the users of the system

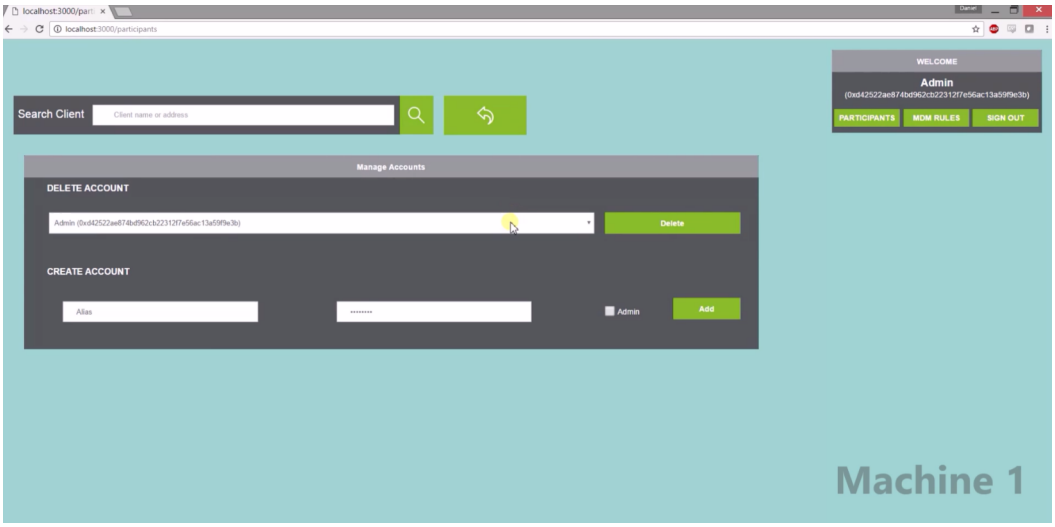


Figure 4.7: Interface to add and remove participants from the system.

It should be noted that the interface differentiates the users from the administrators, displaying less options and functionalities to the users.

In the appendix section of this dissertation is possible to find the mock-up pictures that led to the final interface presented.

5
Tests and Results

This chapter presents the tests and respective results to access if the solution implemented is working as expected. Full details of the tests conduction may be found in the appendix section of this dissertation.

- Access the portal front-page as a user

No. Steps	**User Profile**	**Description**	**Importance**	**Test Successful**
3	User	Test if a user is able to use the interface to log into the system.	Medium	YES

Table 5.1: Access the portal front-page as a user

Table 5.1 refers to the test performed as a user to log in to the system through the user interface.

- Add a new client record to the system

No. Steps	User Profile	Description	Importance	Test Successful
5	User	Test if it is possible to add a new client record having only a "name" attribute.	High	YES
5	User	Test if it is possible to add a new client record having both "name" and "phone number" attributes.	High	YES
5	User	Test if it is possible to add a new client record having the "name", "phone number" and "email" attributes.	High	YES

Table 5.2: Add a new client record to the system

Table 5.2 presents the set of tests performed to attempt to add new client records into the system, with different Data Smart Contract Factories in place, requiring different attributes.

- Delete a client record from the system

No. Steps	User Profile	Description	Importance	Test Successful
5	User	Test if it is possible to remove a client record from the system.	High	YES

Table 5.3: Delete a client record from the system

Table 5.3 summarizes the test performed to delete a client record from the system.

- Log out from the system

No. Steps	User Profile	Description	Importance	Test Successful
1	User	Test if it is possible to log out from the system using the interface.	Medium	YES

Table 5.4: Log out from the system

Table 5.4 refers to the test performed as a user to log out from the system through the user interface.

- MDM Cleansing rules

No. Steps	User Profile	Description	Importance	Test Successful
5	User	Test if the rule of transforming the attribute value into Upper Case works.	High	YES
5	User	Test if the rule of transforming the attribute value into Proper Case works.	High	YES
5	User	Test if the rule of transforming the attribute value by adding a space every 3 characters works.	High	YES
5	User	Test if the rule of transforming the attribute value into Lower Case works.	High	YES
5	User	Test if several rules of attribute value transformation work at the same time.	High	YES

Table 5.5: MDM Cleansing rules

Table 5.5 sums the set of tests conducted to test the different MDM Cleansing rules implemented on the system.

- MDM Match and Merge rules

No. Steps	User Profile	Description	Importance	Test Successful
5	User	Test if the matching of two records by "name" is working by using two records with the same "name", and so the records are merged with the prevalence of the newest information.	High	YES
5	User	Test if the matching of two records by "name" is not working by using two records with a different "name", and so the records are not merged.	High	YES
5	User	Test if the matching of two records by "name"is working by using two records with the same "name", and so the records are merged with the prevalence of the longest information.	High	YES
5	User	Test if the matching of two records by "name", if the "name" is 50% similar, is working by using two records with a similar "name", and so the records are merged with the prevalence of the newest information.	High	YES

No. Steps	User Profile	Description	Importance	Test Successful
5	User	Test if the matching of two records by "phone number", is working by using two records with the same "phone number", and so the records are merged with the prevalence of the newest information.	High	YES
5	User	Test if the matching of two records by "phone number", if the "phone number" is 80% similar, is working by using two records with a similar "phone number", and so the records are merged with the prevalence of the longest information.	High	YES
5	User	Test if the matching of two records by "phone number", if the "phone number" is 70% similar, is not working by using two records with a very different "phone number", and so the records are not merged.	High	YES
5	User	Test if the match two records by "email", is working by using two records with the same "email", and so the records are merged with the prevalence of the newest information.	High	YES
5	User	Test if the match two records by "email", if the "email" is 60% similar, is working by using two records with a similar "email", and so the records are merged with the prevalence of the longest information.	High	YES
5	User	Test if the match two records by several attributes is working by using two records, where all the attributes are supposed to be equal , using two different records so the merging of the records is not supposed to work.	High	YES
5	User	Test if the match two records by several attributes is working by using two records, where all the attributes are supposed to be matched using different rules , using two similar records so the merging of the records is supposed to work.	High	YES

Table 5.6: MDM Match and Merge rules

Table 5.6 represents the set of tests used to test different sets of active MDM match and

merge rules.

- Search for a client record

No. Steps	User Profile	Description	Importance	Test Successful
2	User	Test if it is possible to search for a client record.	High	YES

Table 5.7: Search for a client record

Table 5.7 refers to the conducted test performed in order to test the search function for clients.

- Update a client record

No. Steps	User Profile	Description	Importance	Test Successful
5	User	Test if it is possible to update a client record.	High	YES

Table 5.8: Update a client record

Table 5.8 represents the test performed related with the update of user records.

- Access the portal front page as an admin

No. Steps	User Profile	Description	Importance	Test Successful
3	User	Test if it is possible to log into the system as an admin using the system interface.	Medium	YES

Table 5.9: Access the portal front page as an admin

Table 5.9 refers to the test performed as an admin to log in to the system through the user interface.

- Add a new participant to the system

No. Steps	User Profile	Description	Importance	Test Successful
4	User	Test if it is possible, as an admin, to add a new participant to the system.	Medium	YES

Table 5.10: Add a new participant to the system

Table 5.10 represents the test used to test the creation of new participant accounts in the system.

- Remove a participant from the system

No. Steps	User Profile	Description	Importance	Test Successful
5	User	Test if it is possible, as an admin, to remove a participant from the system.	Medium	YES

Table 5.11: Remove a participant from the system

Table 5.11 represents the test used to test the deletion of participant accounts from the system.

- Create a new Data Smart Contract Factory

No. Steps	User Profile	Description	Importance	Test Successful
5	User	Test if it is possible, as an admin, to create a new Data Smart Contract Factory.	High	YES

Table 5.12: Create a new Data Smart Contract Factory

Table 5.12 refers to the test conducted in order to check if the deployment of Data Smart Contract Factories was functioning.

6
Conclusions and Future Work

The elaboration of this proof-of-concept project required, in a first instance, the in-depth research of Master Data Management and the cutting-edge technology that is Blockchain. While the first is well documented, implemented and tested, the "trust-ledger" is fairly recent and it is difficult to find accurate information as journalistic pieces and scientific articles may even contradict each other and there are still groups of users that are adverse to Blockchain uses other than to support cryptographic transactions. Only recently big enterprises and projects began to use Blockchain as the back-bone of disruptive solutions, such as the one presented in this dissertation, that aim to prove Blockchain usefulness and validity outside the financial world. However, what these enterprises are doing with Blockchain and how it is still unclear.

In a second instance, the development phase of the project, it was necessary the knowledge of programming technologies like Javascript, Solidity, HTML and CSS. Each of these technologies presented a new challenge, due to the lack of previous experience, but specifically Solidity, the Ethereum smart-contract programming language, due to the lack of documentation and the language itself, as it is not in an advanced state of development and is lacking many functionalities a programmer might be used to.

Thankfully, previous knowledge in Java ended up being a facilitator when trying to learn the new technologies, and Javascript, by using Node.JS, proved being an accelerator of the implementation phase by exploring the existing synergies between Ethereum and its javascript API.

Ethereum in itself was a key factor of this project by having implemented contracts with programming capabilities and allowing the creation of test networks resulting on not having to create a Blockchain platform from scratch but exploiting Ethereum functionalities.

It is important to note that, according to the tests performed, the solution implemented works as intended: it is possible to achieve a unique view of client using Blockchain to implement MDM, proving that one could use Blockchain to store information in a decentralized way, without the need for an intermediary, and make information available to a set of peer-to-peer nodes at the same time and even to apply a set if standardization, matching or merging rules to that information, opening up Blockchain to a world of possible new uses and applications,

not necessarily monetized.

To make sure this or other similar solutions are ready to implement in an enterprise, some tweaks and new tests could be suggested. For example, a solution where all MDM logic is not inside Blockchain Smart Contracts. One would assume that having the Index of Data Smart Contracts outside the chain could result in a more efficient index and therefore a faster response time in all use cases. Additionally, there should be a higher focus on the security of the solution, exploring cryptographic methods to store data in Smart Contracts and rethinking the login system and therefore creating a safer environment. Finally, the solution is always expansible by adding possible attributes that the records could have and create new MDM rules the system can enforce.

Bibliography

[1] David Loshin. *Master Data Management*. Morgan Kaufmann Publishers, 2009.

[2] A. Cleven and F. Wortmann. *Uncovering Four Strategies to Approach Master Data Management*. 43rd Hawaii International Conference on System Sciences (HICSS), Jan 2010.

[3] R. Vilminko-Heikkinen and S. Pekkola. Establishing an organization's master data management function: A stepwise approach. pages 4719–4728, Jan 2013.

[4] A. Dreibelbis, I. Milman, P. van Run, E. Hechler, M. Oberhofer, and D. Wolfson. *Enterprise Master Data Management: An SOA Approach to Managing Core Information*. IBM Press, 2008.

[5] J. R. Talburt and Y. Zhou. *Entity Information Life Cycle for Big Data: Master Data Management and Information Integration*. Morgan Kaufmann Publishers, 2015.

[6] An Oracle White Paper. *Supporting Your Data Management Strategy with a Phased Approach to Master Data Management*. 2016.

[7] W. Chen et al. *A Practical Guide to Managing Reference Data with IBM InfoSphere Master Data Management Reference Data Management Hub*. IBM Redbooks, 2013.

[8] Tyler Graham. *Microsoft SQL Server 2012: Master Data Services, Second Edition*. McGraw-Hill/Osborne Media, 2012.

[9] M. Allen and D. Cervo. *Multi-Domain Master Data Management: Advanced MDM and Data Governance in Practice*. Morgan Kaufmann Publishers, 2015.

[10] J. Vosburg and A. Kumar. *Managing Dirty Data in Organizations using ERP: lessons from a case study*. Industrial Management and Data Systems, Vol. 101 Iss: 1, pp.21 - 31, 2001.

[11] C. Loser, C. Legner, and D. Gizanis. *Master Data Management For Collaborative Service Processes*. International Conference on Service Systems and Service Management 2004, 2004.

[12] E. Gomede and R. Barros. *Master Data Management and Data Warehouse : An Architectural Approach for Improved Decision-Making*. IEEE, 2013.

[13] Customer data integration master data management. `http://www.cdi-mdm.com/datastewardship.html`. Accessed: 07-06-2016.

[14] What is data governance (dg)? `http://searchdatamanagement.techtarget.com/definition/data-governance`. Accessed: 07-06-2016.

[15] Rachel Haines. Data governance is strategic; data stewardship is tactical. https://infocus.emc.com/rachel_haines/data-governance-is-strategic-data-stewardship-is-tactical/. Accessed: 07-06-2016.

[16] Alexander Maedche. *An ERP-centric Master Data Management Approach.* AMCIS 2010, 2010.

[17] A. White, O'Kane B., Palanca T., and Moran M. *Magic Quadrant for Master Data Management of Customer Data Solutions.* Gartner, 2015.

[18] A. White, O'Kane B., Palanca T., and Moran M. *Magic Quadrant for Master Data Management of Product Data Solutions.* Gartner, 2015.

[19] TIBCO Software Inc. *IBM - InfoSphere Master Data Management.* http://www-03.ibm.com/software/products/en/infosphere-master-data-management. Accessed: 31-03-2016.

[20] Informatica. *MDM - Master Data Management Software — Informatica US.* https://www.informatica.com/products/master-data-management.html#fbid=EgqJn5KbgRV. Accessed: 31-03-2016.

[21] Informatica master data management. https://www.informatica.com/content/dam/informatica-com/global/amer/us/collateral/brochure/enable-business-agility-mdm-tech_brochure_7159.pdf. Accessed: 07-06-2016.

[22] Oracle. *Bookshelf v8.0: About Siebel Universal Customer Master Concepts.* https://docs.oracle.com/cd/B40099_02/books/UCMSIA/UCM_Overview4.html. Accessed: 31-03-2016.

[23] Configuring siebel data quality cleansing for siebel ucm. https://docs.oracle.com/cd/B40099_02/books/UCMSIA/UCM_Administration25.html. Accessed: 07-06-2016.

[24] TIBCO Software Inc. *TIBCO MDM 9.0 — TIBCO.* http://www.tibco.com/products/automation/master-data-management/mdm. Accessed: 31-03-2016.

[25] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System.* The Cryptography Mailing list, 2008.

[26] P. Franco. *Understanding Bitcoin: Cryptography, Engineering and Economics.* Wiley Finance Series, 2015. Chapters 2 and 7.

[27] J. Kroll, I. Davey, and W. Felten. *The Economics of Bitcoin Mining, or Bitcoin in the Presence of Adversaries.* Workshop on the Economics of Information Security, 2013.

[28] Forbes: Why innovative companies are using the blockchain. http://www.forbes.com/sites/jonathanchester/2016/01/11/why-innovative-companies-are-using-the-blockchain. Accessed: 13-04-2016.

[29] Satoshi Nakamoto. *Original bitcoin main.h.* 2009.

[30] C. Barski and C. Wilmer. *Bitcoin for the Befuddled.* No Starch Press, 2015. Chapters 1 and 8.

[31] John ratcliff's code suppository: How to parse the bitcoin blockchain. http://codesuppository.blogspot.pt/2014/01/how-to-parse-bitcoin-blockchain.html. Accessed: 14-04-2016.

[32] Cryptorials: What is 'proof of work'? http://cryptorials.io/glossary/proof-of-work/. Accessed: 14-04-2016.

[33] Forbes: Why companies like orange silicon valley are working with private blockchain startups. http://www.forbes.com/sites/jonathanchester/2016/02/17/beyond-bitcoin-why-companies-like\-orange-silicon-valley-are-working-with-blockchain-startups/. Accessed: 14-04-2016.

[34] Wall street journal: Ibm bets on bitcoin ledger - wsj. http://www.wsj.com/articles/ibm-bets-on-bitcoin-ledger-1455598864. Accessed: 15-04-2016.

[35] A linux foundation collaborative project: Hyper ledger foundation. https://www.hyperledger.org/. Accessed: 15-04-2016.

[36] Ripple: Welcome to ripple — ripple. https://ripple.com/. Accessed: 15-04-2016.

[37] Business insider: Santander innoventures invests 4 million in ripple labs. http://uk.businessinsider.com/santander-innoventures-invests-4-million-in-ripple-labs-2015-10. Accessed: 15-04-2016.

[38] Ethereum foundation: Ethereum project. https://www.ethereum.org/. Accessed: 15-04-2016.

[39] Coindesk - microsoft explores adding ripple tech to blockchain toolkit. http://www.coindesk.com/microsoft-hints-future-ripple-blockchain-toolkit/. Accessed: 15-04-2016.

[40] Multichain: Multichain — open source private blockchain platform — ditributed ledger. http://www.multichain.com/. Accessed: 15-04-2016.

[41] Tierion - cloud datastore — backed by the blockchain. https://tierion.com/features. Accessed: 15-04-2016.

[42] Secure wireless networks - filament. http://filament.com/. Accessed: 15-04-2016.

[43] Tradle. https://tradle.io/. Accessed: 15-04-2016.

[44] G. Wood. *Ethereum: A secure decentralised generalised transaction ledger*. Ethereum. Accessed: 23-07-2016.

[45] Ethereum - ethereum whitepaper. https://github.com/ethereum/wiki/wiki/White-Paper. Accessed: 23-07-2016.

[46] Ethereum - ethereum wiki. https://github.com/ethereum/wiki/wiki. Accessed: 14-07-2016.

A

Interface mock-ups

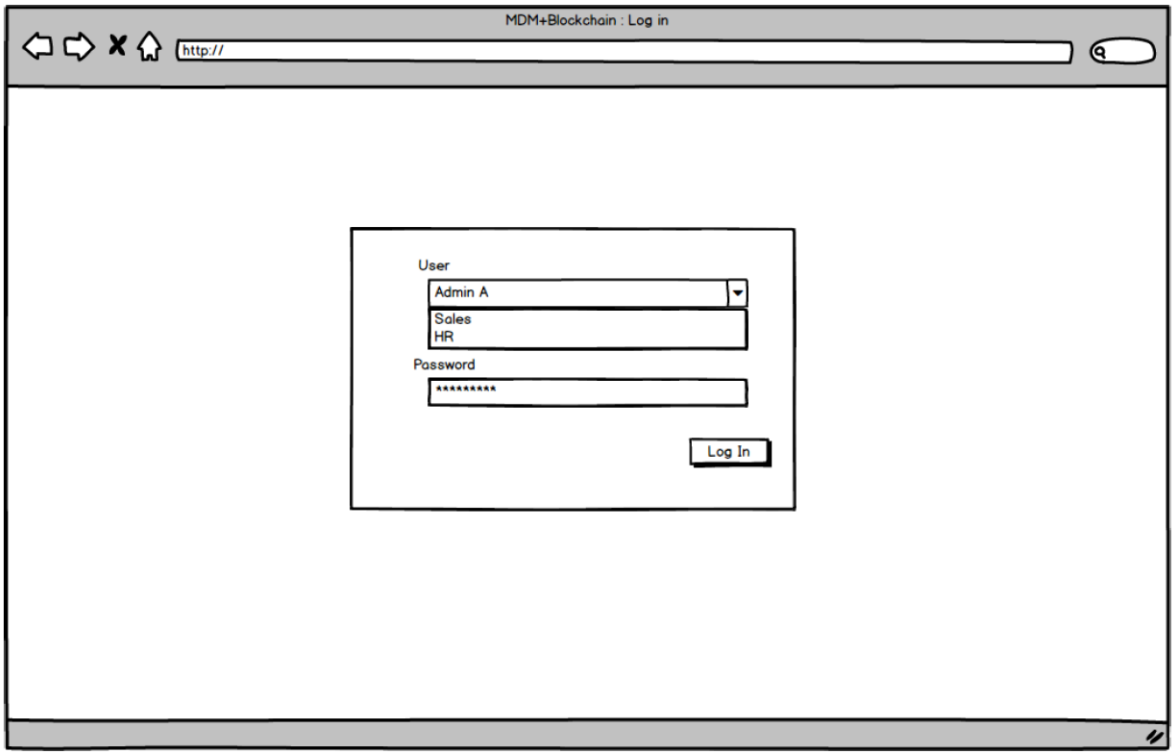


Figure A.1: Mock up of the login page.

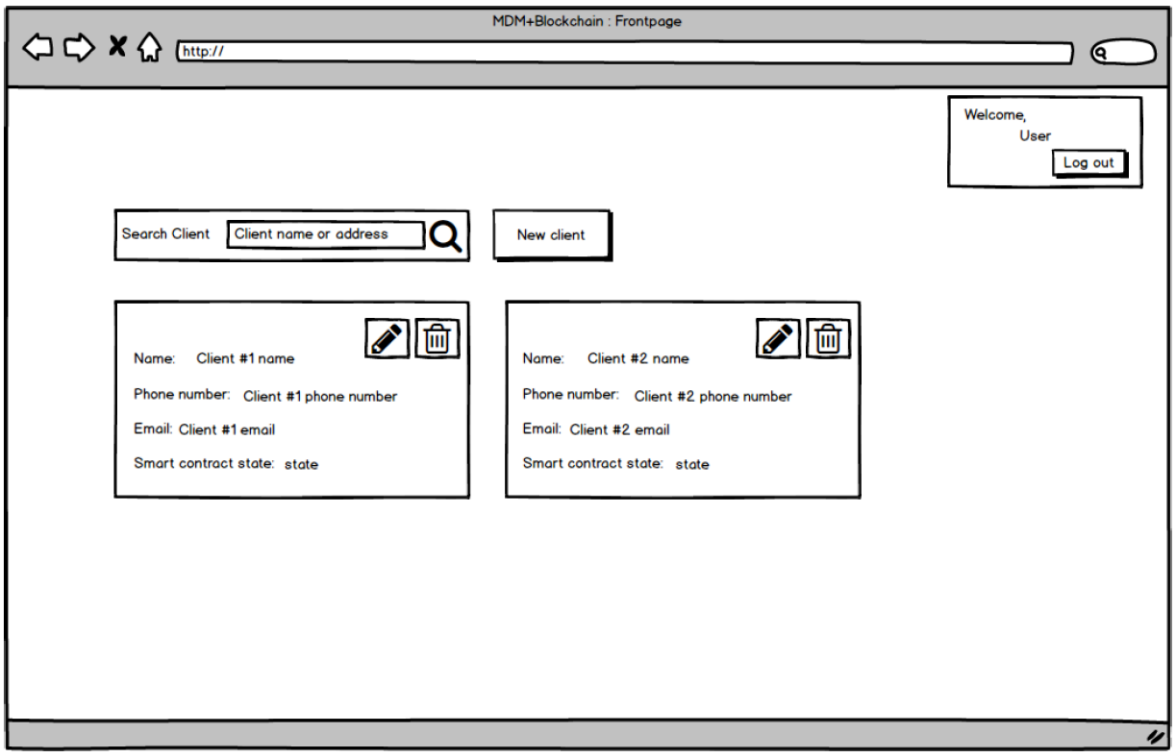


Figure A.2: Mock up of the front page, displaying search results.

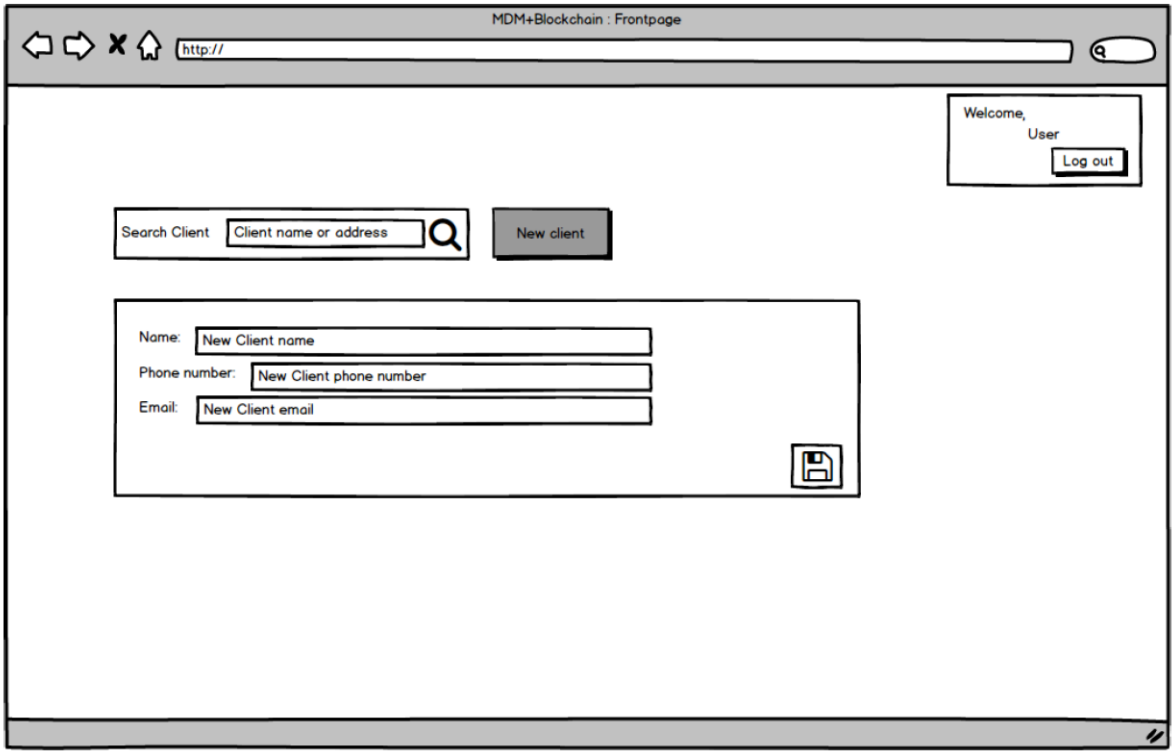


Figure A.3: Mock up of the new client page.

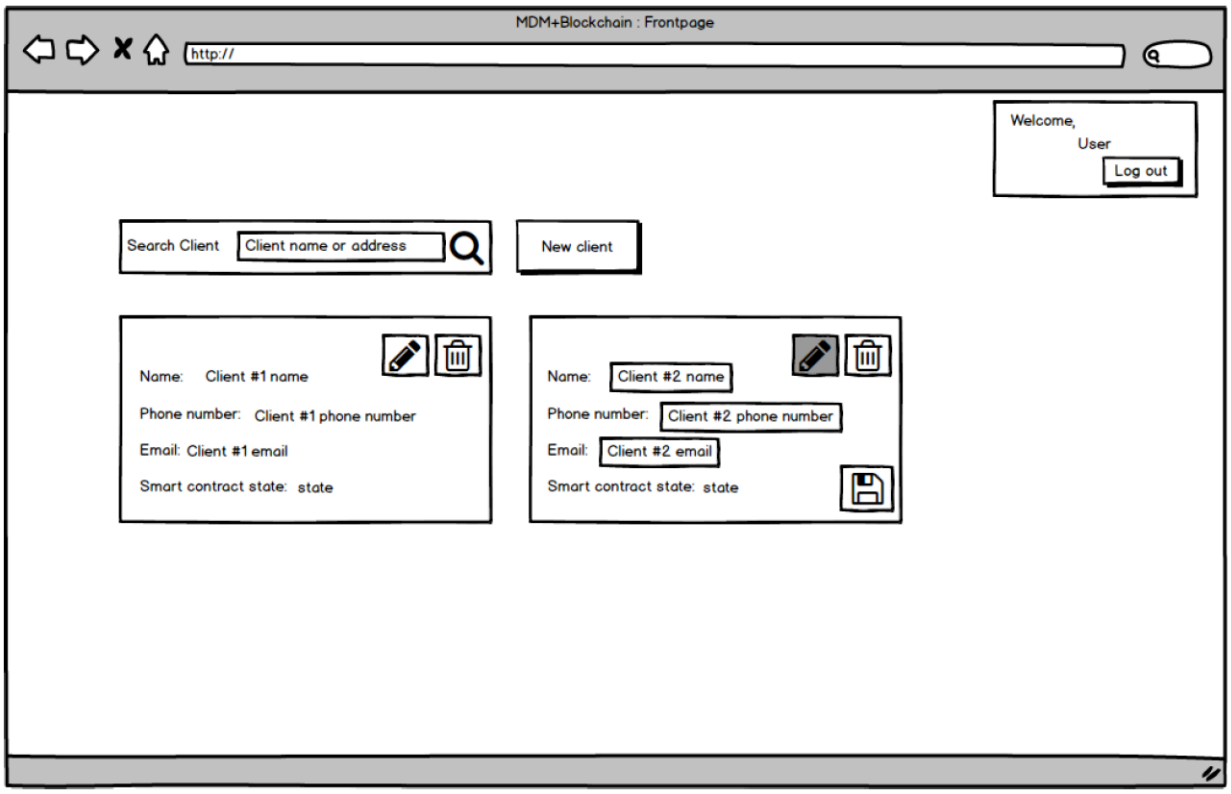


Figure A.4: Mock up of the update page.

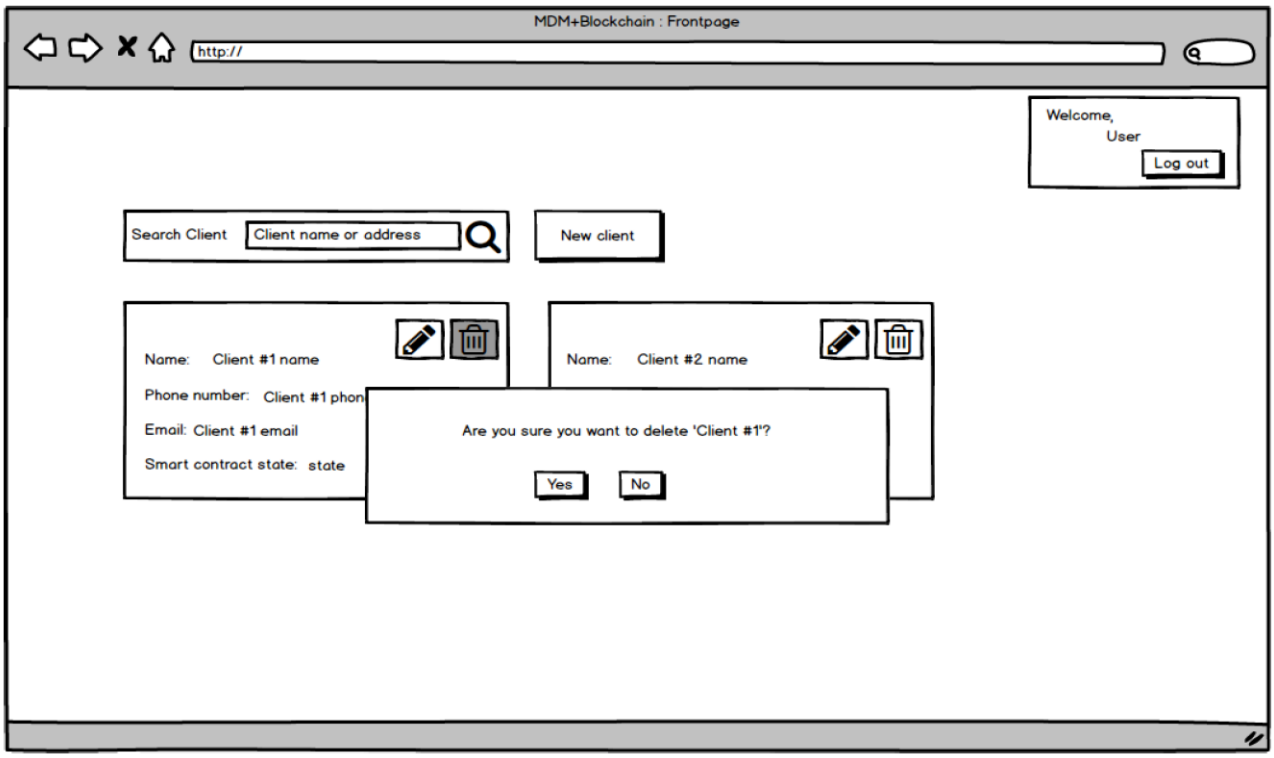


Figure A.5: Mock up of the deletion event.

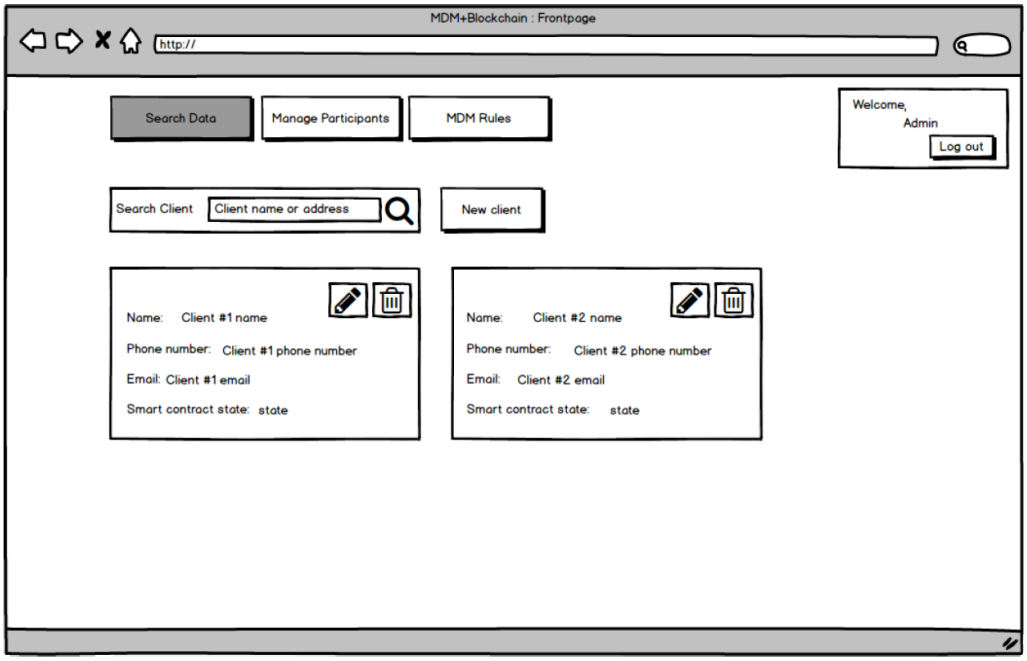


Figure A.6: Mock up of the front page as an admin, displaying more buttons

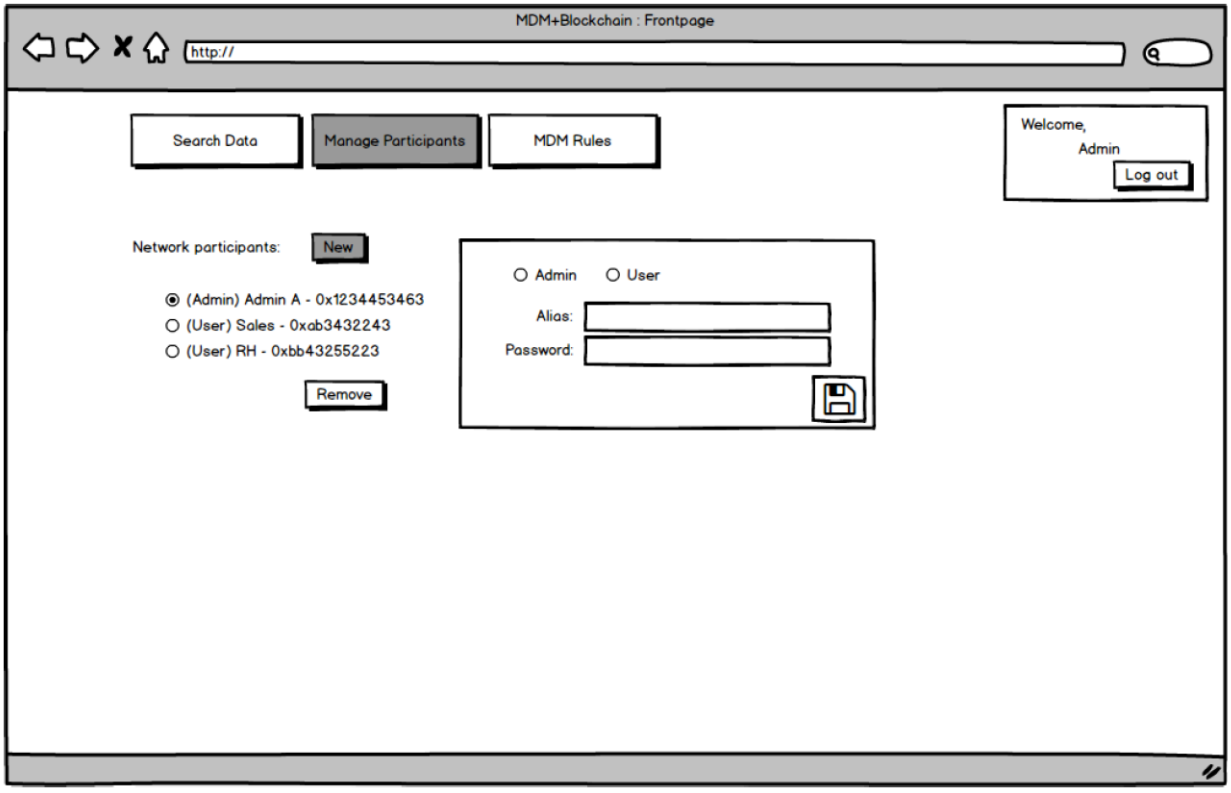


Figure A.7: Mock up of the manage participants page.

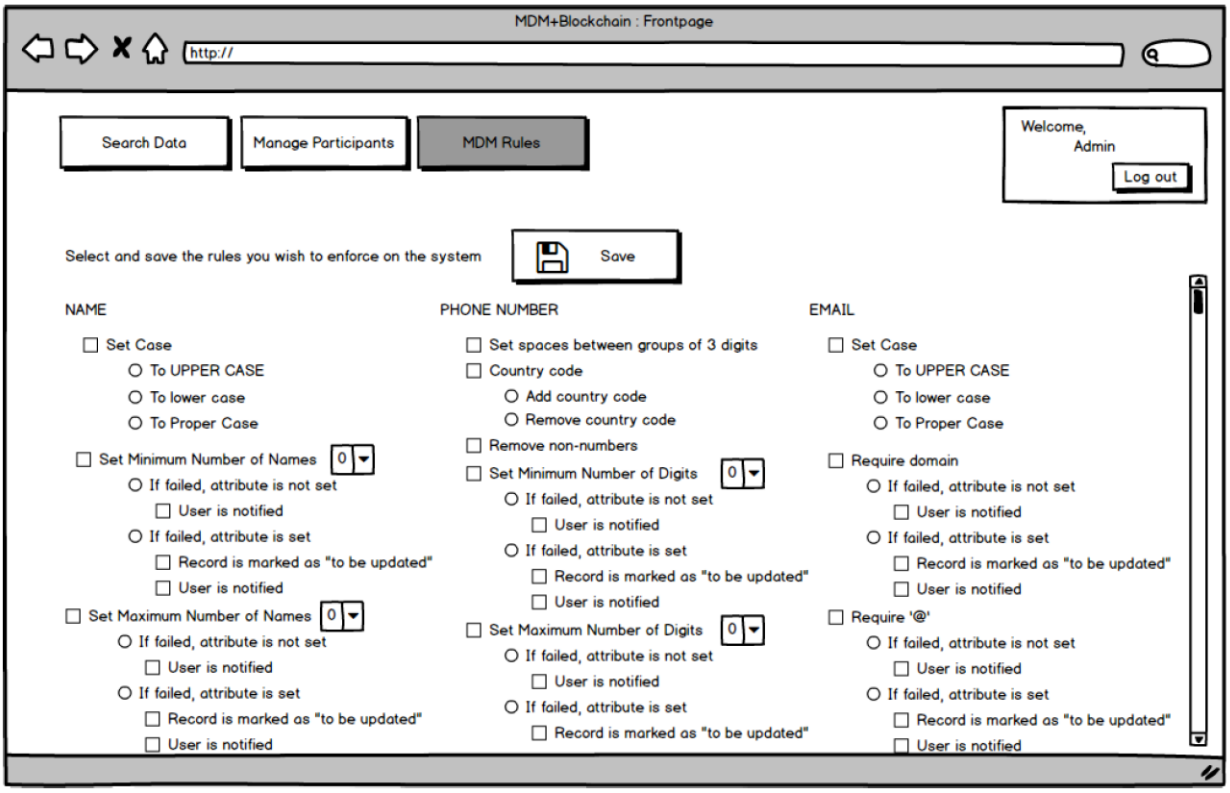


Figure A.8: Mock up of the manage MDM Rules page.

B

Tests Description

Importance	Medium	**Preconditions**		
User Profile	User	1. There is at least one user account in the system; 2. The admin knows the password to one of the users account.		
Number of Steps	3			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	The interface is accessed	A webpage with a login box appears with a select box that have an admin account. There is also a box to write a password	As expected	YES
2	In the select box, select one user account. In the password field, write the password for that account	The selected account and address is displayed, and the field password is filled with the password	As expected	YES

3	Press "Sign in"	The admin is redirected to the portal front page, that have a search box, a new user button, and a box with the logged account information along with a sign out button	As expected	YES

Table B.1: Test if a user is able to use the interface to log into the system.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0;		
Number of Steps	5	2. A Data Smart Contract Factory already exists in the system:		
Execution Date	30-08-2016	a) Requiring that all client entries have "name".		
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select "New client"	It is presented the new client page, with a form to introduce the client name on the field "Name".	As expected	YES
2	In the field "Name", write "John Doe"	The field "Name" is filled with "John Doe".	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field "Search Client", write "John Doe"	The field "Search Client" is filled with "John Doe"	As expected	YES
5	Press "enter" or click in the magnifying glass button to search	A box with the address regarding "John Doe" appears with "John Doe" in the name field	As expected	YES

Table B.2: Test if it is possible to add a new client record having only a "name" attribute.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0;		
Number of Steps	5	2. A Data Smart Contract Factory already exists in the system:		
Execution Date	30-08-2016	a) Requiring that all client entries have "name" and "phone number".		
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select "New client"	It is presented the new client page, with a form to introduce the client name on the field "Name" and the client phone number on the field "Phone number"	As expected	YES
2	In the field "Name", write "John Doe" and in the field "Phone number" write "123456789"	The field "Name" is filled with "John Doe" and the field "Phonenumber" is filled with "123456789".	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field "Search Client", write "John Doe"	The field "Search Client" is filled with "John Doe"	As expected	YES
5	Press "enter" or click in the magnifying glass button to search	A box with the address regarding "John Doe" appears with "John Doe" in the name field and "123456789" in the phone number field.	As expected	YES

Table B.3: Test if it is possible to add a new client record having both "name" and "phone number" attributes.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have "name", "phone number" and "email".		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select "New client"	It is presented the new client page, with a form to introduce the client name on the field "Name" the client phone number on the field "phone number" and the client email on the field "Email".	As expected	YES
2	In the field "Name", write "John Doe", in the field "Phone number" write "123456789" and in the field "Email" write "john@doe.com"	The field "Name" is filled with "John Doe", the field "Phone number" is filled with "123456789" and the field "Email" is filled with "john@doe.com"	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field "Search Client", write "John Doe"	The field "Search Client" is filled with "John Doe"	As expected	YES

5	Press "enter" or click in the magnifying glass button to search	A box with the address regarding "John Doe" appears with "John Doe" in the name field, "123456789" in the phone number field and "john@doe.com" in the email field	As expected	YES

Table B.4: Test if it is possible to add a new client record having the "name", "phone number" and "email" attributes.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. Execute test case #1		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, fill the field "Search Client" with "John Doe"	The field "Search Client" is filled with "John Doe"	As expected	YES
2	In the field "Name", write "John Doe"	The field "Name" is filled with "John Doe"	As expected	YES
3	Press "enter" or click in the magnifying glass button to search	A box with the address regarding "John Doe" appears with "John Doe" in the name field and buttons to delete or update the information	As expected	YES
4	Click the delete button	An alert box appears asking for confirmation	As expected	YES
5	At the alert box, press "Yes, delete it"	The page reloads and the "John Doe" information box disappeared	As expected	YES

Table B.5: Test if it is possible to remove a client record from the system.

Importance	Medium	**Preconditions**		
User Profile	User	1. Execute test case #0		
Number of Steps	1			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, click “Sign out”.	The interface reloads into the login page	As expected	YES

Table B.6: Test if it is possible to log out from the system using the interface.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have ”name” b) Transforming names to Upper Case		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select “New client”	It is presented the new client page, with a form to introduce the client name on the field “Name”.	As expected	YES
2	In the field “Name”, write “John Doe”.	The field “Name” is filled with “John Doe”	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field “Search Client”, write “John Doe”.	The field “Search Client” is filled with “John Doe”.	As expected	YES

5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “John Doe” appears with “JOHN DOE” in the name field	As expected	YES

Table B.7: Test if the rule of transforming the attribute value to upper case works.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0;		
Number of Steps	5	2. A Data Smart Contract Factory already exists in the system:		
Execution Date	30-08-2016	a) Requiring that all client entries have ”name”		
		b) Transforming names to Proper Case		
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select “New client”	It is presented the new client page, with a form to introduce the client name on the field “Name”.	As expected	YES
2	In the field “Name”, write “John Doe”.	The field “Name” is filled with “John Doe”	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field “Search Client”, write “John Doe”.	The field “Search Client” is filled with “John Doe”.	As expected	YES
5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “John Doe” appears with “John Doe” in the name field	As expected	YES

Table B.8: Test if the rule of transforming the attribute value to proper case works.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have "name" and "phone number" b) Transforming phone numbers to add a space every 3 digits		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select "New client"	It is presented the new client page, with a form to introduce the client name on the field "Name" and the client phone number on the field "Phone number".	As expected	YES
2	In the field "Name", write "John Doe" and in the field "Phone number" write "123456789".	The field "Name" is filled with "John Doe" and the field "Phone number" is filled with "123456789".	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field "Search Client", write "John Doe".	The field "Search Client" is filled with "John Doe".	As expected	YES
5	Press "enter" or click in the magnifying glass button to search	A box with the address regarding "John Doe" appears with "John Doe" in the name field and "123 456 789" in the phone number field.	As expected	YES

Table B.9: Test if the rule of transforming the attribute value by adding a space every 3 characters works.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have "name" and "email" b) Transforming email to lower case		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select "New client"	It is presented the new client page, with a form to introduce the client name on the field "Name" and the client email on the field "Email".	As expected	YES
2	In the field "Name", write "John Doe" and in the field "Email" write "jOhN@dOe.COM".	The field "Name" is filled with "John Doe" and the field "Email" is filled with "jOhN@dOe.COM".	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field "Search Client", write "John Doe".	The field "Search Client" is filled with "John Doe".	As expected	YES
5	Press "enter" or click in the magnifying glass button to search	A box with the address regarding "John Doe" appears with "John Doe" in the name field and "john@doe.com" in the email field.	As expected	YES

Table B.10: Test if the rule of transforming the attribute value into lower case works.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have "name", "phone number" and "email" b) Transforming name to proper case c) Transforming phone numbers to add a space every 3 digits d) Transforming email to lower case		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select "New client"	It is presented the new client page, with a form to introduce the client name on the field "Name", the client phone number on the field "Phone number" and the client email on the field "Email".	As expected	YES
2	In the field "Name", write "john doe", in the "Phone number" field write "123456789" and in the field "Email" write "jOhN@dOe.COM".	The field "Name" is filled with "john doe", the field "Phone number" is filled with "123456789" and the field "Email" is filled with "jOhN@dOe.COM".	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field "Search Client", write "John Doe".	The field "Search Client" is filled with "John Doe".	As expected	YES

5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “John Doe” appears with “John Doe” in the name field, “123 456 789” in the phone number field and “john@doe.com” in the email field.	As expected	YES

Table B.11: Test if several rules of attribute value transformation work at the same time.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have ”name” and ”phone number”; b) Clients are matched if the new client name is equal to an existing client; c) Newest information prevails. 3. The following client already exists in the system: Name: John Doe; Phone number: 123456789		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select “New client”	It is presented the new client page, with a form to introduce the client name on the field “Name” and the client phone number on the field “Phone number”.	As expected	YES
2	In the field “Name”, write “John Doe” and in the field “Phone number” write “987654321”.	The field “Name” is filled with “John Doe” and the field “Phone number” is filled with “987654321”.	As expected	YES

3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field “Search Client”, write “John Doe”.	The field “Search Client” is filled with “John Doe”.	As expected	YES
5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “John Doe” appears with “John Doe” in the name field and “987654321” in the phone number field.	As expected	YES

Table B.12: Test if the match two records by ”name”, if the ”name” is equal, is working by using two records with the same ”name”, and the records are being merged with the prevalence of the newest information.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have ”name” and ”phone number”; b) Clients are matched if the new client name is equal to an existing client; c) Newest information prevails. 3. The following client already exists in the system: Name: John Doe; Phone number: 123456789		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select “New client”	It is presented the new client page, with a form to introduce the client name on the field “Name” and the client phone number on the field “Phone number”.	As expected	YES

2	In the field “Name”, write “Johnny Doe” and in the field “Phone number” write “987654321”.	The field “Name” is filled with “Johnny Doe” and the field “Phone number” is filled with “987654321”.	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field “Search Client”, write “John Doe”.	The field “Search Client” is filled with “John Doe”.	As expected	YES
5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “John Doe” appears with “John Doe” in the name field and “123456789” in the phone number field and another with the address regarding “Johnny Doe” appears with “Johnny Doe” in the name field and “987654321” in the phone number field.	As expected	YES

Table B.13: Test if the match two records by ”name”, if the ”name” is equal, is not working by using two records with a different ”name”, and the records are not merged.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have "name" and "phone number"; b) Clients are matched if the new client name is equal to an existing client; c) Longest information prevails. 3. The following client already exists in the system: Name: John Doe; Phone number: 123456789		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select "New client"	It is presented the new client page, with a form to introduce the client name on the field "Name" and the client phone number on the field "Phone number".	As expected	YES
2	In the field "Name", write "John Doe" and in the field "Phone number" write "98765".	The field "Name" is filled with "John Doe" and the field "Phone number" is filled with "98765".	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field "Search Client", write "John Doe".	The field "Search Client" is filled with "John Doe".	As expected	YES

5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “John Doe” appears with “John Doe” in the name field and “123456789” in the phone number field.	As expected	YES

Table B.14: Test if the match two records by ”name”, if the ”name” is equal, is working by using two records with the same ”name”, and the records are being merged with the prevalence of the longest information.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have ”name” and ”phone number”; b) Clients are matched if the new client name is at least 50% similar to an existing client name; c) Newest information prevails. 3. The following client already exists in the system: Name: John Doe; Phone number: 123456789		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select “New client”	It is presented the new client page, with a form to introduce the client name on the field “Name” and the client phone number on the field “Phone number”.	As expected	YES
2	In the field “Name”, write “Joao Dor” and in the field “Phone number” write “98765”.	The field “Name” is filled with “Joao Dor” and the field “Phone number” is filled with “98765”.	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES

4	In the field “Search Client”, write “Jo”.	The field “Search Client” is filled with “Jo”.	As expected	YES
5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “Joao Dor” appears with “Joao Dor” in the name field and “98765” in the phone number field.	As expected	YES

Table B.15: Test if the match two records by ”name”, if the ”name” is 50% similar, is working by using two records with a similar ”name”, and the records are being merged with the prevalence of the newest information.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have ”name” and ”phone number”; b) Clients are matched if the new client phone number is equal to an existing client phone number; c) Newest information prevails. 3. The following client already exists in the system: Name: John Doe; Phone number: 123456789		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select “New client”	It is presented the new client page, with a form to introduce the client name on the field “Name” and the client phone number on the field “Phone number”.	As expected	YES
2	In the field “Name”, write “Mary Marshal” and in the field “Phone number” write “123456789”.	The field “Name” is filled with “Mary Marshal” and the field “Phone number” is filled with “123456789”.	As expected	YES

3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field “Search Client”, write “*”.	The field “Search Client” is filled with “*”.	As expected	YES
5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “Mary Marshal” appears with “Mary Marshal” in the name field and “123456789” in the phone number field.	As expected	YES

Table B.16: Test if the match two records by ”phone number”, if the ”phone number” is equal, is working by using two records with the same ”phone number”, and the records are being merged with the prevalence of the newest information.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have ”name” and ”phone number”; b) Clients are matched if the new client phone number is at least 80% similar to an existing client phone number; c) Longest information prevails. 3. The following client already exists in the system: Name: John Doe; Phone number: 123456789		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select “New client”	It is presented the new client page, with a form to introduce the client name on the field “Name” and the client phone number on the field “Phone number”.	As expected	YES

2	In the field “Name”, write “John” and in the field “Phone number” write “12345677911”.	The field “Name” is filled with “john” and the field “Phone number” is filled with “12345677911”.	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field “Search Client”, write “*”.	The field “Search Client” is filled with “*”.	As expected	YES
5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “John Doe” appears with “John Doe” in the name field and “12345677911” in the phone number field.	As expected	YES

Table B.17: Test if the match two records by ”phone number” if the ”phone number” is 80% similar, is working by using two records with a similar ”phone number”, and the records are being merged with the prevalence of the longest information.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have "name" and "phone number"; b) Clients are matched if the new client phone number is at least 70% similar to an existing client phone number; c) Newest information prevails. 3. The following client already exists in the system: Name: John Doe; Phone number: 123456789		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select "New client"	It is presented the new client page, with a form to introduce the client name on the field "Name" and the client phone number on the field "Phone number".	As expected	YES
2	In the field "Name", write "Mary Marshal" and in the field "Phone number" write "0000".	The field "Name" is filled with "Mary Marshal" and the field "Phone number" is filled with "0000".	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field "Search Client", write "*".	The field "Search Client" is filled with "*".	As expected	YES

5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “Mary Marshal” appears with “Mary Marshal” in the name field and “0000” in the phone number field and a box with the address regarding “John Doe” also appears with the pre-existing information.	As expected	YES

Table B.18: Test if the match two records by ”phone number”, if the ”phone number” is 70% similar, is not working by using two records with a very different ”phone number”, and the records are not merged.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have “name” and “email”; b) Clients are matched if the new client email is equal to an existing client email; c) Newest information prevails. 3. The following client already exists in the system: *Name: John Doe; Email: john@doe.com*		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select “New client”	It is presented the new client page, with a form to introduce the client name on the field “Name” and the client email on the field “Email”.	As expected	YES
2	In the field “Name”, write “Mary Marshal” and in the field “Email” write “john@doe.com”.	The field “Name” is filled with “Mary Marshal” and the field “Email” is filled with “john@doe.com”.	As expected	YES

3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field “Search Client”, write “*”.	The field “Search Client” is filled with “*”.	As expected	YES
5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “Mary Marshal” appears with “Mary Marshal” in the name field and “john@doe.com” in the email field.	As expected	YES

Table B.19: Test if the match two records by ”email”, if the ”email” is equal, is working by using two records with the same ”email”, and the records are being merged with the prevalence of the newest information.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have “name” and “email”; b) Clients are matched if the new client email is at least 60% similar to an existing client email; c) Newest information prevails. 3. The following client already exists in the system: *Name: John Doe; Email: john@doe.com*		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select “New client”	It is presented the new client page, with a form to introduce the client name on the field “Name” and the client email on the field “Email”.	As expected	YES

2	In the field “Name”, write “Joao” and in the field “Email” write “joxx@dxx.come”.	The field “Name” is filled with “Joao” and the field “Email” is filled with “joxx@dxx.come”.	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field “Search Client”, write “*”.	The field “Search Client” is filled with “*”.	As expected	YES
5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “John Doe” appears with “John Doe” in the name field and “joxx@dxx.come” in the email field.	As expected	YES

Table B.20: Test if the match two records by ”email”, if the ”email” is 60% similar, is working by using two records with a similar ”email”, and the records are being merged with the prevalence of longest information.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have “name” and “email”; b) Clients are matched if: the new client name is equal, the new phone number is equal and the email is equal to an existing client. c) Longest information prevails. 3. The following client already exists in the system: *Name: John Doe; Phonenumber: 123456789; Email: john@doe.com*		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select “New client”	IIt is presented the new client page, with a form to introduce the client name on the field “Name”, the client phone number on the field “Phone number” and the client email on the field “Email”.	As expected	YES
2	In the field “Name”, write “John Doe”, in the field “Phonenumber” write “123456789” and in the field “Email” write “john@doe.pt”.	The field “Name” is filled with “John Doe”, the field “Phone number” is filled with “123456789” and the field “Email” is filled with “john@doe.pt”.	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field “Search Client”, write “*”.	The field “Search Client” is filled with “*”.	As expected	YES

5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “John Doe” appears with “John Doe” in the name field, “123456789” in the phone number field and “john@doe.pt” in the email field. Another box with the address regarding “John Doe” appears with “John Doe” in the name field, “123456789” in the phone number field and “john@doe.com” in the email field.	As expected	YES

Table B.21: Test if the match two records by several attributes is working by using two records, where all the attributes are supposed to be equal, using two different records so the merging of the record is not supposed to work.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. A Data Smart Contract Factory already exists in the system: a) Requiring that all client entries have “name” and “email”; b) Clients are matched if: the new phone number is at least 70% similar and the email is at least 80% similar to an existing client. c) Newest information prevails 3. The following client already exists in the system: *Name: John Doe; Phonenumber: 123456789; Email: john@doe.com*		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the portal front-page, select “New client”	It is presented the new client page, with a form to introduce the client name on the field “Name”, the client phone number on the field “Phone number” and the client email on the field “Email”.	As expected	YES
2	In the field “Name”, write “Johnny Doe”, in the field “Phonenumber” write “123456700” and in the field “Email” write “john@doe.cop”.	The field “Name” is filled with “Johnny Doe”, the field “Phone number” is filled with “123456700” and the field “Email” is filled with “john@doe.cop”.	As expected	YES
3	Click the floppy disk button to save	A successful message is shown indicating that the client is being deployed to the blockchain and then the portal front page is presented	As expected	YES
4	In the field “Search Client”, write “*”.	The field “Search Client” is filled with “*”.	As expected	YES

5	Press “enter” or click in the magnifying glass button to search	A box with the address regarding “Johnny Doe” appears with “Johnny Doe” in the name field, “123456700” in the phone number field and “john@doe.cop” in the email field.	As expected	YES

Table B.22: Test if the match two records by several attributes is working by using two records, where all the attributes are supposed to be matched using different rules, using two similar records so the merging of the records is supposed to work.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. The following client already exists in the system: *Name: John Doe; Phonenumber: 123456789; Email: john@doe.com*		
Number of Steps	2			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the field "Search Client", write "John Doe".	The field "Search Client" is filled with "John Doe".	As expected	YES
2	Press "enter" or click in the magnifying glass button to search	A box with the address regarding "John Doe" appears with "John Doe" in the name field, "123456789" in the phone number field amd "John@Doe.Com" in the email field.	As expected	YES

Table B.23: Test if it is possible to search for a client record.

Importance	High	**Preconditions**		
User Profile	User	1. Execute test case #0; 2. Execute test case #7; 3. The following client already exists in the system: *Name: John Doe; Phone number: 123456789; Email: John@Doe.Com*		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	In the box with the address regarding "John Doe", press the pencil/update button.	The user is redirected to the update client page. The field name is filled with "John Doe", the field phone number is filled with "123456789" and the field email is filled with "John@Doe.Com".	As expected	YES

2	In the field "Phone number", erase it's content and write "999999999"	The field "phone number" is filled with "999999999".	As expected	YES
3	Click the save button.	A success message is shown and the user is redirected to the front page.	As expected	YES
4	In the field "Search Client", write "John Doe".	The field "Search Client" is filled with "John Doe".	As expected	YES
5	Press "enter" or click in the magnifying glass button to search	A box with the address regarding "John Doe" appears with "John Doe" in the name field, "999999999" in the phone number field amd "John@Doe.Com" in the email field.	As expected	YES

Table B.24: Test if it is possible to update a client record.

Importance	Medium	**Preconditions**		
User Profile	Admin	1. There is at least one admin account in the system; 2. The Admin knows the password to one of the admins account.		
Number of Steps	3			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	The interface is accessed.	A webpage with a login box appears with a select box that have an admin account. There is also a box to write a password.	As expected	YES

2	In the select box, select one admin account. In the password field, write the password for that account.	The selected account and address is displayed, and the field password is filled with the password.	As expected	YES
3	Press "Sign in".	The admin is redirected to the portal front page, that have a search box, a new user button, and a box with the logged account information along with a sign out button. Additionally, there is a "Participants" button and a "MDM Rules" button.	As expected	YES

Table B.25: Test if it is possible to log into the system as an admin using the system interface.

Importance	Medium	**Preconditions**		
User Profile	Admin	1. Execute test case #8		
Number of Steps	4			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	Click the "Participants" button.	A page with a form to delete and add participants appear. In the form to add participants, there is a field for the alias and other field for the password. Additionally, there is a check box to check if the new account is supposed to be an admin.	As expected	YES

2	In the alias field, write “User”, and in the password field write “user-pass”.	The alias field is filled with “User”, and the password field is filled with the password.	As expected	YES
3	Press “Add”.	A success message appears and the admin is redirected to the front-page.	As expected	YES
4	Press “Sign out”.	The admin is redirected to the login page and “User” is one of the select box choices.	As expected	YES

Table B.26: Test if it is possible, as an admin, to add a new participant to the system.

Importance	Medium	**Preconditions**		
User Profile	Admin	1. Execute test case #8; 2. Execute test case #9.		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	Click the “Participants” button.	A page with a form to delete and add participants appear. In the form to add participants, there is a field for the alias and other field for the password. Additionally, there is a check box to check if the new account is supposed to be an admin.	As expected	YES
2	Select the user “User” in the select box.	The selected participant is “User”.	As expected	YES
3	Press “Delete”.	An alert box appears asking if the admin is sure he wants to delete the participant.	As expected	YES

4	Press “Yes, Delete it”.	The admin is redirected to the login front page.	As expected	YES
5	Press “Sign out”.	The admin is redirected to the login page and “User” is not one of the select box choices.	As expected	YES

Table B.27: Test if it is possible, as an admin, to remove a participant from the system.

Importance	Medium	**Preconditions**		
User Profile	Admin	1. Execute test case #8.		
Number of Steps	5			
Execution Date	30-08-2016			
Steps	**Description**	**Expected Result**	**Obtained Result**	**Test Successful**
1	Click the “MDM Rules” button.	A page with a form to select the active attributes, cleansing rules, matching and merging rules appear.	As expected	YES
2	Select the attributes “name”, “phone number”, “email”, the cleansing rules set case (for name) Proper Case, Add a Space every 3 digits (for phone number) and Set case (for email) to lower case; Select the match rules to match by phone number (80% match) and merging rules so the newest attributes prevail.	The form displays the selected options.	As expected	YES

3	Press the floppy disk/save button.	A success message appears and the user is redirected to the front page.	As expected	YES
5	Select the “MDM Rules” button.	The previously selected options are marked on the form.	As expected	YES

Table B.28: Test if it is possible, as an admin, to create a new Data Smart Contract Factory.