FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Peer to Peer Insurance over Blockchain

Ricardo Jorge Matos Figueiredo

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Prof. António Miguel Pimenta Monteiro

Second Supervisor: Eng. Ricardo Fernando Martins

July 24, 2017

Peer to Peer Insurance over Blockchain

Ricardo Jorge Matos Figueiredo

Mestrado Integrado em Engenharia Informática e Computação

July 24, 2017

Abstract

General Insurance is responsible for risk protection on all activity sectors, form health to work safety insurance, thus achieving a highlight position today. Despite its importance, its traditional eel has some major flaws which can disrupt its future. It has a centralized way of operation, which makes it inefficient, less transparent, with client-trust issues, creating conflicts of interest between insurance companies and policyholders. Furthermore, it's an industry with relative difficulty in innovation, which contrasts with today's services powered by new technologies.

New disruptive technologies like Blockchain and Artificial Intelligence, have the capacity to compromise the value chain of traditional insurance, creating a challenge in its stability and future. Associated to this new kind of technologies, comes a new way of thinking perpetuated by the new generations based on the new trend of sharing economy. They give priority to new technology based services, which are decentralized, more convenient, more price competitive, transparent, and efficient and they prefer services which put the user more in control instead of the current less personalized options. Social Insurance, based on social economy, enables people in need of insurance to connect and pool their money and risks. It offers coverage that is cheaper, more transparent, and more relevant to the customer. Members of this type of insurance are both policyholders and underwriters.

Blockchain is a decentralized transaction ledger shared amongst all nodes participating in the system. Every node has an updated copy of the database and cannot update it without the consensus of the network, removing the need for having a central authority or trusted third party to monitor the system. Each Blockchain implementation may have its own consensus mechanism (e.g. Proof-of-Work and Proof-of-Stake) to ensure that one node on its own cannot change the database without being validated by the network.

Bitcoin is the world's most well-known Blockchain implementation, a public ledger for all transactions made in with this digital currency. However, Blockchain technology can be applied to multiple use cases and industries using Smart Contracts (a collection of code that runs on the network) to define the rules of the business. It fits the purpose for this business model, because it has peer to peer communication by default and has no central authority, making the service more efficient and transparent.

The main goal is to study the use of blockchain technology in a social insurance service integrated in a traditional insurance company, thus creating a proof of concept backed up by a business model.

Resumo

A Indústria Seguradora ocupa nos dias de hoje uma posição dominante relativamente ao modo como opera no mercado. É centralizadora por natureza, o que faz com que o serviço seja pouco eficiente, pouco transparente e de fraca confiança, criando até situações de conflitos de interesses com os detentores das apólices. De um modo geral, a indústria seguradora tradicional é vista com desconfiança por parte da sociedade, mas dada a sua necessidade, tem-se mantido numa posição dominante e estável no mercado atual.

Contudo, é uma indústria com relativa dificuldade em inovar o que contrasta com o paradigma atual de serviços assentes em tecnologias de ponta. Estas novas tecnologias disruptivas têm a capacidade de comprometer a cadeia de valor das seguradoras tradicionais, constituindo um sério problema para o seu futuro, e, assentes nestas novas tecnologias está também associada uma mudança no mindset social, sendo agora altamente valorizada a utilização de tecnologias de ponta, a transparência, eficiência, e a preferência por serviços de caracter mais pessoal que possibilitam ao utilizador um maior controlo. Posto isto, existem já em desenvolvimento, modelos de negócio de seguradoras direcionados para o futuro, como por exemplo o chamado “Social Insurance”. O “Social Insurance" permite às pessoas com necessidade de um seguro, conectarem-se entre si e distribuir o risco, fazendo essencialmente “self insuring”. Neste modelo os membros são ambos detentores da apólice e subscritores e assim, desta forma é criado um serviço mais eficiente, transparente e que deixa ao utilizador a possibilidade de maior controlo sobre a gestão do seu capital investido no seguro.

Todos estes fatores comprometem de certo modo o futuro sustentável do modelo de negócio das seguradoras tradicionais e é sobre este problema que assenta este tema de dissertação. O objetivo principal passa pela exploração da tecnologia "Blockchain" em seguros sociais, criando um protótipo funcional e um modelo de negócio que dê suporte.

Acknowledgements

I would first like to thank my supervisor Professor António Pimenta Monteiro from the Faculty of Engineering of the University of Porto, for all the guidance and support that he gave me throughout this project.

I would like express my sincere appreciation to my second supervisor, Engineer Ricardo Fernando Martins, for all guidance and support that gave me throughout this project. I really appreciate the persistence and help that he gave me in finding solutions for the hardest problems that I have encountered during this work.

I would like to thank to may co-workers in Deloitte Digital for all support, guidance and good spirit during this dissertation.

I would like to thank to Faculty of Engineering of the University of Porto for being a second home to me during these period of my life, and for giving me plenty of opportunities to improve my life as a person and to help others improving theirs.

I would to like to thank to Tuna de Engenharia da Universidade do Porto for being a second family to me. "This group give us so much and asks for so little". Here I lived great moments of music and friendship that I will never forget.

I would like to thank to all my friends, with a special appreciation to João Pedro Morais, João Lemos, Sérgio Pinto, João Brandão, Pedro Mesquita, Miguel Almeida, Mariana Pissarra, Tiago Catalão, Inês Faria, Sofia Malheiro, Raquel Ribeiro, Rafael Lopes, Pedro Casanova, Pedro Afonso, João Reis, Sara Lemos, Ana Melo and Mariana Santos for all support during this dissertation.

I would like to thank my parents, Maria Figueiredo and José Figueiredo for being very supportive on all moments of my life, specially during this project, for all good values that taught me and for all advices that gave me in most hard times.

I would like to thank to my grandfather Basilio Matos for being one of the most important persons in my life.

Ricardo Jorge Matos Figueiredo

"Words were not given to man
in order to conceal his thoughts."

José Saramago

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

MVP	Most Valuable Product
P2P	Peer to Peer
PoW	Proof of Work
PoC	Proof of Concept
PoS	Proof of Stake

Chapter 1

Introduction

This document focus on the study and development of a project and dissertation about "Peer to Peer Insurance over Blockchain".

This chapter describes the context and background, as well as the problem, objectives and expected benefits that will be addressed during the development of this project.

1.1 Context

Insurance has been an important service to society since back to ancient times. It started supposedly with the Hammurabi Code, the first written insurance policy which appeared on a Babylonian obelisk, offering, basic insurance for individuals if a personal catastrophe made it impossible to pay back a recovery debt, and continued throughout history always trying to stabilize individuals and the economy against risk. It is important in almost every activity in society, from all kinds of industry activities to health care and life insurance. At the same time, it had to pay attention to all kinds of change and always be adaptive to new market needs and new technologies [Maj16]. Today's Insurance has bigger challenges to take care of. It has to adapt to a set of pressures from the digital revolution, new costumer needs, new market players and regulations.

The digital revolution has been having a great impact on every industry, disrupting almost every service and every product. This transformation based on the development of the Internet, backed up the development of technologies like the smart phone, increasing processing power and connectivity between consumers, allowing large amounts of data to be stored and analyzed. At the same time that the amount of data is growing, its quality is also more complex and diverse. For example, with the help of sensors installed on smart devices we can now measure heart rate, traveled distance, among other features, adding the possibility to change decision making by providing useful quality information helping risk evaluation and management. Business processes that are repetitive and require little decision-making skills across multiple value chains are being automated and therefore, shifting industries focus toward exploring automation of more complex high

risk processes. This automation and decentralization of services make products more transparent, efficient and user centric then their traditional counter parts.

Costumer needs are also changing and their views about service excellence are getting more demanding for better user centric services based in digital technologies. New competitors are appearing in the market, having new business models centered in customer experience instead of product development and already backed up by new digital technologies and new social trends like sharing economy. Sharing economy is an economic system where users source assets or services as per need constituting a common set of providers that essentially share the underlying assets with other users [Cap16]. Peer-to-Peer is the new wave of insurance, based on social economy, that promotes collective buying power to offer insurance cheaper [Sah16]. It enables people in need of insurance to connect and pool their risks, essentially self-insuring. In this model, members are both policyholders and underwriters.

Blockchain, a new form of data storage and management created by Satoshi Nakamoto in 2009, has emerged from Bitcoin and has the power to disrupt insurance by its own nature. It's a decentralized transaction ledger shared amongst all nodes participating in the system. Every node has an updated copy of the database and cannot update it without the consensus of the network, removing the need for having a central authority or trusted third party to monitor the system. This characteristics match both new customer demands and new technological scenario, therefore is a potential powerful technology that threats the status quo of the insurance world [Nak09].

Besides new trends and technology improvement, insurance industry has been also transformed over the past several years by new regulations. With the financial crisis of 2008 the financial sector suffered new regulations which inspired new ones on the insurance sector. This regulations requires new flexible data models and more transparency of their business processes [Waj16]. All this factors contribute to the future destabilization of traditional insurance. They may have a strong presence in today's market, but it may be compromised due to a slow adaptation to this new trends and regulations.

1.2 Problem

Insurance is in a process of change. New technologies, new customer needs, new market players and regulations are currently shaping how insurance occupies its position in today's market. Disruption compromises traditional values chains, new customer expectations, and demand for more transparency, efficiency, decentralized and user centric services are constantly increasing as new regulations pressure insurance for cost reductions and more efficiency.

Instead of being centered on legacy operations, Insurance must look forward to reinvent its products in order to be able to adapt to the new social, technological and market conditions. Therefore, Blockchain technology is, in a first look, a great opportunity for insurance and it can be crucial for this transformation, having characteristics that are in accordance with this new conditions, like peer to peer communication and decentralized authority (does not need to have a central authority to validate transactions). On the other hand, it's still a world to look up for, mainly to know

how it could be used in insurance in a concrete case, exploring the advantages and challenges that it creates.

1.3 Objectives

The main objective of this thesis is to analyze the benefits and challenges of blockchain and explore how it could be integrated in a peer to peer insurance system. This will be backed up by a prototype integrating a blockchain database into a peer to peer application that manages underwriting and claim processes. It's also going to be made a simplified business model in order to give consistency to the solution.

1.4 Expected Benefits

This project is expected to provide relevant and carefully analyzed information about how traditional insurance could use blockchain technology in its value chain forwarding the adaptation to new changes in the insurance ecosystem.

Chapter 2

State of the Art

2.1 General Insurance

2.1.1 Insurance Overview

Insurance has been following society since ancient times, helping it to deal with unexpected situations which tend to happen in high risk environments, giving security by protecting people or entities from risk of uncertain outcomes, not eliminating it but transferring it totally or partially to a third party.

Insurance started with a very distinct form from the one we know today. Early risk distributing methods appeared in China in the 3rd millennia BC, were merchant traders traveling treacherous river rapids would distribute their cargos across multiple vessels in order to distribute the loss in case any catastrophe happened to the ships. Later in 2100 BC in Babylonia, traders were encouraged to pay for their own risks with loans that were paid back with interest, but only after the goods arrived safely. The Greeks and The Romans introduced the first primordial forms of life and health insurance by creating organized guilds/societies which afforded members certain benefits like burial rite costs and traveling expenses of members in the army in exchange of regular contributions. In the 14th century in Genoa, appeared the first insurance contracts which allowed insurance to be separated form investment, composed by insurance pools backed up by pledges of landed states. Loyd's Coffee House in London was a place where merchants, ship-owners, and underwriters met to transact business. By the end of the 18th century, this was one of the first modern insurance companies. In the 19th century the state has becoming part of the insurance scenario, protecting workers against sickness and disability, either temporary or permanent, destitute old age, and unemployment. Although forms of insurance have existed since ancient times, it was only in the last 100 years insurance assumed the form we recognize today. In these beginnings the main problem with insurance was getting established outside of very small insures, raising capital needed to be able to effectively cover their clients [Mas]. In recent years, insurance has becoming more diverse, expensive and embraces the majority of society activities.

2.1.2 Insurance Operation

Insurance is needed for various reasons to protect against unforeseen events, some of those reasons include legal obligations like car insurance, freeing up capital, bringing business confidence, providing protection against catastrophes, and insure a large list of things like motor, household, buildings, health, travel, mobile phones, aviation, among other things.

There are many stakeholders in the insurance system. From the point of view of who wants to purchase an insurance service, we have a Policyholder which is the person or entity that pays the premium, the Insured whose life, health or property is insured and the Beneficiary who receives the benefit. From the point of view of the insurance company we have the Insurer which is the insurance company, the Underwriter who evaluates, accepts or rejects a proposal and the Actuary which is who determine the cost of the insurance. Between the client and the company we have the Broker who places insurance business with any insurance company, the Agent who brings in business for the insurance company and the Claimant which is who makes a formal request for claim/benefit.

When someone purchases insurance, they are purchasing an insurance policy and an insurance contract is established between an individual or a business and an insurance company. Risk is the exposure to the chance of injury or loss, risk of financial loss is transferred an insurer company in exchange of a premium payment.

There are three types of conditions established in a normal insurance contract, general conditions, specific conditions and special conditions. General conditions are those which are embedded with an insurance policy and are printed in the contract. Specific conditions are those which make the contract unique, they have the information about the policy holder, the insured, the beneficiaries, the insured object, the covered risks, the premium amount, contract duration, etc. Special conditions are those which alter the standard policy coverage.

The Insured is not always secure from financial responsibility. There is no unlimited liability, existing limits that define the maximum compensation of the insurer in case of accident and a deductible or excess payment which is the responsibility of the insured in case of accident. This is important to share the risk between the insurance company and to control risky behaviors from the insured.

There are various types of insurance companies, general insurance, life insurance, health care insurance, composite companies, reinsurance companies, captive insurance companies, insurance exchanges and self-insures.

General insurance includes the most common insurance services like motor, property, liability and professional risks insurance. Life Insurance offer insurance to protect against the risk of death and disability. Healthcare Insurance cover the cost of medical expenses on treatment of illnesses and normally integrate the general insurance group. Composite Companies are insures that offer a mix of General and Life insurance products. Reinsurance companies offer a risk carrier product available for other insurance and reinsurance companies, accepting to manage their risk. Captive Insurance Companies are owned and operated by the organization it insures. Insurance Exchanges

are comprised groups of individuals or companies carrying out underwriting activities in a 'real market'. Self-insurers are companies that cover their own risk, where normally there is money put aside to provide a risk fund and, generally, is not considered an insurance.

Underwriting is the process by which an insurer assesses the acceptability and the transfer of values risk for a premium. It's composed by various phases: data gathering, risk assessment, premium calculation, quote proposal, issue policy and premium collection. Data gathering is the first step in the underwriting selection process and evolves gathering information about the insured and what is being insured for evaluating the risk so therefore the premium could be calculated. Once the information is gathered it is important to make a risk assessment, which is verifying if the risk that was previously evaluated falls into the insurance company guidelines in order to be accepted by the company. Premium calculation is the calculation of the right premium to charge for the risk, and it depends on the data available and the class of business being priced. Quote proposal is the proposal offered to the possible client, and it contains information about the annual premium, proposed period of cover, the policy limits, specific coverage, any exclusions provided by the policy and rating information used to price risk, so that the policy holder can verify that the information used is correct. Once the costumer accepts the quote the policy is issued, and if a broker or other intermediary is involved, they will normally issue policy documentation or certificates of insurance on behalf of the insurers. The final phase is premium collection which is the process to collect the premium. It can be by direct billing in which the bill is sent directly from the insurance company to the insured and the payment is sent directly to the insurance company, or by agency billing, where the agent or broker bills the insured, collects form the insured and then pays the insurance company.

2.1.3 Health Insurance

Healthcare Insurance has the goal of supporting totally or partially the medical expenses and is composed by three types of healthcare insurance: refund, pure network, or mixed.

In Refund Healthcare Insurance, the client pays first for all the expenses involving treatment and assistance, and next is refunded by the insurance company according with the terms previously established in the contract. In this system, the insured has free choice of doctors, clinics, hospitals and services. In Pure Network healthcare insurance, the client has a limited choice of doctors, services and facilities and the insurance will only cover a previously accorded value with the healthcare institution leaving the client to pay the rest. In a mix healthcare insurance the user can choose between the preselected medical services available in their insurance policy, paying a percentage of the total medical bill or they could simply choose a medical service outside the insurance network, pay all the expenses and get a refund by the insurance company according with the insurance policy. The healthcare insurance market is currently evolving from a refund system to a network system with the objectives of reducing and minimizing the increasing medical costs in medical care. That can also improve the quality of the service because the services available to clients are pre picked by the insurance company assuring better quality, a greater service consumption with the same financial limits, reduces administrative work and allows companies to

have more control of the premiums. The acceptance of a healthcare insurance involves the verification of several aspects about the possible insured like social behaviors, geographic location and individual characteristics like sex, age, height and weight.

2.2 Insurance Challenges and Trends

Emerging trends in the insurance industry are a combination of business and technology themes making them the future drivers in this industry [Jos16]. Insurance is in a process of integration into a new world composed by a drastic change in social behavior, consumer demands, market regulations and technological advancements. This should be viewed as an opportunity for improvement and change, because it will have great importance on this industry's global future. This chapter reports current trends, technology evolution and adaptation, new challenges, and new players in the market.

2.2.1 Background and the 2008 Financial Crisis

In the final quarter of the 20th century, insurance experienced the rise of big insurance companies dominating the market, namely through merges and acquisitions and the beginning of a new business model - direct sales. During this period of time, solutions with desktop and server side computing appeared on the market which made big insurance companies take a leap forward and stabilize as leaders of the market [EMC16]. This momentum started to decrease due to reduced consumer spending on insurance products. Also the financial crisis of 2008 caused lower return on fixed investments which were the main income for most insurers. Although it does not appear as a main threat, the crisis marked insurance in many adverse ways because they depend largely on the financial sector [Sch09].

2.2.2 Costumer Behavior and Technology

The digitalization megatrend is transforming the expectations of consumers across the globe [MaDH+16]. New digital technologies are revolutionizing and disrupting almost every product and service, making them more transparent, efficient and user centric. They also affect insurance by creating new opportunities and challenges driven by, data growth, automation, artificial intelligence, new devices and new distribution channels.

In the next ten years customer behavior will change as the majority of the adult population will be replaced by the "Millennials" generation, making them the predominant market force [EMC16]. These new type of users have different values related to products and services. They prefer a digital base self-service costumer centric experience, where the user has the power to use and change the product as they wish, basically putting them more in control of the product and removing power from the company selling the product. Examples of these new self-service types of services are Walmart's scan and go, Starbucks's-payment and Citibank's snapshot application, which makes processes more efficient, simple and convenient for the client. Customers today

have far greater access to information, Internet and digital devices are changing the way companies / brands interact with their customers [Jos16]. This new concept has been pushing Insurance to adapt from the old distributed systems and, from a survey made by Forrester in 2016, there's been a rise in the use of digital services, where 25 percent of US life insurance buyers use digital applications like, web/mobile site, social communities, online ads, and videos [Jos16].

With a generation mainly using digital services as their preferences, new doors of opportunities are opened to insurance. The ability to extract value from data is increasing, because the quantity and quality of data that is being stored is rising. "Big Data" is becoming a great opportunity to manage large amounts of information from multiple sources to find and demonstrate behavior patterns using machine learning techniques, and therefore can be used to get better customer insights create new products. It is apparent that with more quality information, better automatized decisions can be made, making products more efficient, reliable and more personalized for each customer needs. Automatizing more complex and risky processes, for example, can help in fraud identification. Personalized customer interactions like chatbots, can shorten claims cycle time and improve forecasting of claims [MaDH+16]. Another advantage of digital devices is "realtime" information where customers, who can no longer wait for information updates, can check their insurance state in realtime using digital devices powered by Internet. In short, with the help of Big Data, Machine Learning and Artificial Intelligence, insurance has the opportunity to automatize more risky processes and therefore create more efficient products in terms of automatic decision making and customer needs.

Besides trying to adapt to costumer and technological demands, general insurance has another threat to its stability, which is the entrance of new players in the market. There are new companies entering the insurance market which already offer products based on digital technology, appealing to future user trends. These companies are called Insurtechs and most of them are based on a new insurance trend based on social economy.

2.2.3 Social Economy and Peer to Peer Insurance

Social Economy is a rising trend in consumption that makes consumers share their underutilized assets and making money in the process. It basically allows collaborative consumption of services, enabling groups of people to share expenses and to have a peer to peer product experience, offering them less costly products and a better customer experience by introducing a social interaction, generally enjoyed by nowadays consumers. We can see this services in car sharing, office sharing and on-demand workforce, for example with Uber and Zipcar which enable ride sharing [Cap16]. Insurance is also been affected by this trend, which caused the emergence of a new type of insurance based on social economy, Peer to Peer or Social Insurance.

Social Insurance allows individuals who are in need of insurance to pool their premiums and share risk, making them in charge of managing the money involved in the insurance process. The main objectives of social insurance is to reduce the cost of insurance, by reducing administrative costs and fraud through peer pressure between the platform users, and to give more transparency and control to users because all the details are available to the group members limiting the group

size of claims and re-establishing community trust in the service. On the contrary, Traditional Insurance is not seen as a transparent and trusted service, mainly because sometimes enters in a state of conflict with costumers making millennials to have a special interest in this new type of insurance [Ber16]. There are Insurtechs based in peer to peer insurance already operating in the market like Guevara, Lemonade and Friendsurance.

2.3 Blockchain

2.3.1 Concept - Bitcoin Overview

E-Commerce relies almost on third party institutions to process and validate electronic transactions of money. Although it's a system that works well, it has at the same weaknesses related with the fact that it is a trust based model. The cost of transactions mediation is high, limiting the minimum practical transaction size, making small transactions almost impossible to be made, and the time required for a transaction to be approved is considerably high for a web based process [Nak09]. In order to solve these and other issues, in 2009 a person or group of persons under the pseudonym of Satoshi Nakamoto created an electronic based currency named Bitcoin.

This new electronic base coin and payment system is totally decentralized, not requiring third parties to validate transactions, but instead it relies on cryptographic proof for it. This cryptographic validation system is what's called a Blockchain, and it can be used to serve other use cases besides payments, because it's a different approach to data management, storage and coding.

Blockchain technology provides an electronic public ledger without central authority, which is a record for all transactions that have taken place within a specific protocol [MvG14]. It has a decentralized data structure, a public key system to authenticate users and transactions and a consensus system to approve a block made of transactions, therefore preventing "double spending", or in other words, preventing the transaction of the same object two consecutive times, which in case of a cryptocurrency it prevents the use of the same money more than one time by the same owner.

2.3.2 Architecture - Bitcoin Overview

In terms of architecture, bitcoin was the first system to implement a blockchain database, therefore it makes sense to initially explore this technology by the bitcoin point of view.

The Bitcoin ledger is a state transition system [But11], where the electronic coin is a chain of digital signatures with a public key system for owner/user authentication and ownership approval. Coins are transferred from user to user by digitally signing a hash of the previous transaction and the public key of the next user and adding these to the end of the chain [Nak09]. This public key transaction system solves the problem of the authenticity of payees, preventing transaction sender form spending coins that do not exist, and from spending other peoples coins, but there's still a remaining flaw of double spending that cannot be solved only with a transaction system like this. Double spending consists on making the transaction of the same coin twice. For example in the

real world we can't use the same 10 euros bill to pay for lunch, and in the next hour use the same bill to pay for another item, because when we pay with the bill the first time we give it to the cashier and are no more the person who possesses it. With digital cash, this prevention is more difficult because coins are simply digital signatures and we could simply copy and paste the same code and buy different things, never losing money, creating a double spending problem. The normal approach to this issue is having a trusted third party controlling all the coins in a transaction, but in a decentralized model like this, this problem is approached with another method. Bitcoin created a consensus system where there is no need of a trusted central authority controlling all coins in the system, where there is no need for trust at all. The protocol ensures that the same coins are not transmitted twice by the same owner at the same time, solving the double spending problem. In this consensus system, all transactions are publicly announced to all nodes participating in the system and all the participants agree on a single order of transactions that were received. This "agreement" consists in all nodes trying to produce packages of transactions called "blocks" every 10 minutes containing a timestamp, a nonce, and a hash referencing the previous block and a list of all transactions that have taken place since the previous block, therefore creating a chain of blocks all linked together [But11]. From a network point of view the system functions in this order:

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a difficult proof-of-work for its block.
4. When a node finds a proof-of-work, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and not already spent.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash as illustrated in the figure 2.1.

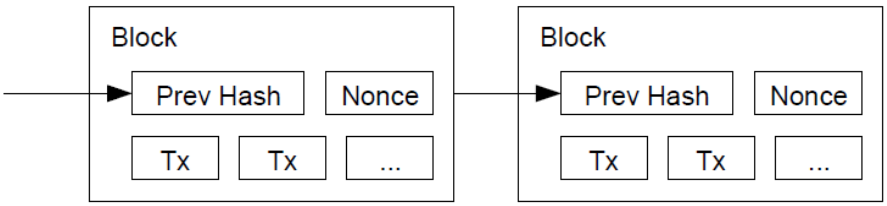

Figure 2.1: Bitcoin Blockchain - Block agregation [Nak09]

2.3.3 Consensus, Mining and Incentive - Bitcoin Overview

A blockchain consensus is the algorithm that insures block validation up on its creation. In other words, its necessary up on the formation of a new block that the transactions it contains occurred

in a specific timestamp in order to be possible to add it to the chain. Bitcoin uses a Proof-Of-Work algorithm for this case, which is a function that is hard to compute, but easy to verify, with the purpose of making the block creation computationally "hard" to do, preventing attackers from remaking the entire blockchain in their favor [But11]. This system is similar to Adam Back's Hashcash [Bac02], and involves scanning for a value that is hashed, or "Mined", the word usually used to describe this process. The hash begins with a number of zero bits, creating exponential work in the number of zero bits required and allowing verification to be verified by executing a single calculation. In the Bitcoin case a nonce is incremented in the block until a value is found that gives the block's hash the required zero bits.

In a system with this format, is essential for nodes to have an incentive to mine the blocks in order to maintain blockchain's integrity, security and efficiency. In Bitcoin the incentive is made by in two ways, the first by "issuing" new coins in circulation, because the first transition in a block is a special transition that starts a new coin owned by the creator of the block, the other incentive comes from a transition fee involved in every transaction. When making a transaction, besides the money being transitioned, there's also an additional fee for the node that successfully mines the block containing the transaction, giving an additional financial incentive for block mining. This incentives make nodes to be honest and if any attacker node assemble more CPU power than all the honest nodes it is more profitable to stay honest and use the power to issue new coins [Nak09].

2.3.4 Other consensus systems

There are other blockchains with different consensus protocols currently in development other than Proof of Work. The following table shows a set of some already functional blockchains with different consensus systems 2.1:

Table 2.1: Blockchains list [MvG14].

Name	Description	Based on currency?	Validation process	Year launched
Bitcoin	Cryptocyurrecy and payment protocol	Bitcoins	PoW	2009
BlackCoin	Cryptocyurrecy and payment protocol	Blackcoins	PoS	2014
Ripple	Payment protocol and currency exchange	Ripples	Consensus Ledger	2011
NameCoin	Distributed domain name management, developed from the first fork of the Bitcoin protocol	Namecoins	PoW	2011
Bitmessage	Distributed communication protocol used to sent encrypted messages	No	PoW	2012
Ethereum	Blockchain operating system to support and host distributed applications	Ether	PoW	2015
Hyperledger	Open source platform for creating private currencies private currecnies or recording assets, and allowing their transfer	No	Consensus Pool	2014
Zerocash	Privacy-preserving version of Bitcoin, where payment transactions do not countain any public information about the payment's origin, destination, or amount.	Zerocoins and Blackcoins	Zero-Knowledge Proof	2014
Ethermint	Ethereum framework operating over tendermint consensus	No	BFT	-

2.3.4.1 Proof of stake

Proof of stake is a consensus system that is not based on a proof to a challenge, which results on the consumption of large quantities of energy used in processing power, instead is based on the ownership of a certain quantity of wealth/power in the system, usually is digital money. This method was introduced on the Bitcointalk forum in 2011, but the first implementation of the first digital currency based on this method was Peercoin in 2012, being a hybrid between PoS and PoW [Gee].

PeerCoin hybrid design is based on the notion of coin age, which is a measure of time for who holds the coin in a particular moment, and is simply defined as currency amount times holding period. In this system there are two types of blocks, proof-of-work and proof-of-stake blocks [KN12]. These last blocks are associated with a special transaction called "coinstake", on which

the owner pays himself resulting on the consumption of his coin age, proving that it as access to a certain amount of wealth (coins) and gaining the right to generate a block for the network. [Vas]. The structure of the *coinstake* is the following:

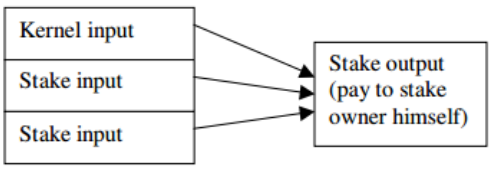

Figure 2.2: Structure of Proof-of-Stake transaction [KN12]

In this model, PoS does not let money be made through block generation and stakeholders will only be rewarded through transaction fees. PoW is used here only for the initial distribution of the currency.

2.3.4.2 Consensus ledger

Consensus Ledger is a consensus algorithm used in the Ripple Ledger, which provides a shared ledger, giving information about the state of account balances. This algorithm operates through a set of nodes which constitute the network of the platform and can be validating nodes or tracking node. Tracking nodes are responsible for querying the ledger and distributing client transactions. Validating nodes have the same functionalities as Tracking nodes with the addition of validating transactions and therefore contributing to the ledger sequence. As the transactions submitted by client applications are propagated through the network, the system applies its consensus mechanism [CSB].

The consensus mechanism consists on a series of rounds, each round having a unique task described in the following list [CSB]:

1. Each node takes all valid transactions it has seen prior to the beginning of the consensus round that have not already been applied, and makes them public in the form of a list known as the “candidate set”.

2. Each node then amalgamates the candidate sets of all servers on its UNL, and votes on the veracity of all transactions.

3. Transactions that receive more than a minimum percentage of “yes” votes are passed on to the next round, if there is one, while transactions that do not receive enough votes will either be discarded, or included in the candidate set for the beginning of the consensus process on the next ledger.

4. The final round of consensus requires a minimum percentage of 80 percent of a server’s UNL agreeing on a transaction. All transactions that meet this requirement are applied to the ledger, and that ledger is closed, becoming the new last-closed ledger.

2.3.4.3 Zero-knowledge pool

Zero-knowledge pool is a consensus algorithm used in Zerocash, which is a full-fledged ledger-based digital currency with strong privacy guarantees [BSCG+]. In this algorithm two parties are involved, the prover A and the verifier B, and it allows prover A to show that he has a credential, without having to give B the exact number [JB10].

2.3.4.4 Tendermint blockchain consensus - Byzantine Fault Tolerance

Tendermint is a consensus algorithm that does not require the concept of mining like Bitcoin and instead is based on a modification of the Byzantine Generals Problem. This mechanism operates on a peer-to-peer connected nodes network which transmit information between each other through *gossip* and keeps a complete sequence of events in a form of blocks as the blockchain in Bitcoin. Valid transactions are grouped in blocks and each block is divided in three parts, a Header containing merkle hashes for various chain states, a Transactions Hash containing all transactions which are to be processes and a Validation hash containing all the signatures assigned to the block. The following figure illustrates a Tendermint block:

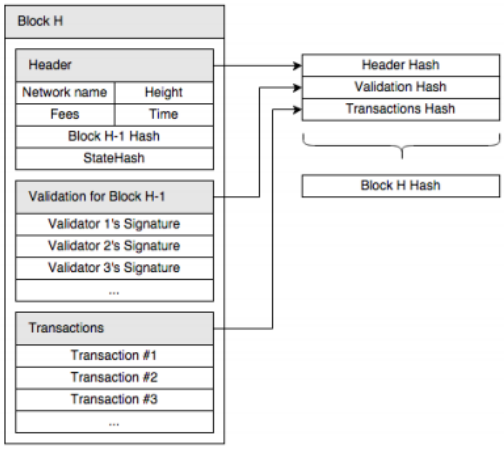

Figure 2.3: Tendermint block structure [Kwo14].

A block is *valid* if all the transactions in the block are valid and there are sufficient signatures included in the validation. These validation is made by validating nodes with a specific voting power, which have the opportunity to participate in the consensus system, by voting to agree on the next block to be added to the chain. There are three types of votes, *prevote*, *precommit* and *commit*, being the block valid if there are at least 2/3 of total of total voting power committing votes to the block and a round-based protocol is run to determine the next one. Each round is composed of three steps, *Propose*, *Prevote*, and *Precommit*, and two special steps *Commit* and *NewHeight*. This state machine is illustrated in the following figure:

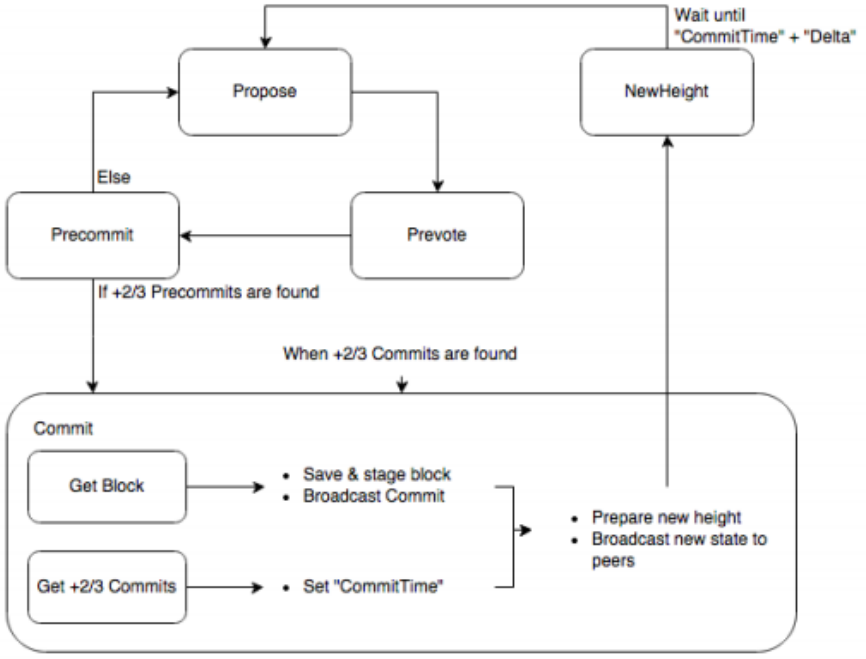

Figure 2.4: Overview of state machine [KN12].

The first step is the *Propose* step, on which the proposed for that round broadcasts a proposal to its peers via gossip.

In *Prevote* phase, all nodes decide about the proposed block and transmit all prevotes for the round to their peers. If the node is locked or received a proposed block, it broadcasts that prevote, otherwise it signs a special nil prevote.

On *Precommit*, if a validator node receives more than 2/3 of prevotes for a particular block the it signs and broadcasts a precommit for that block. Only if the node receives more than 2/3 validation, it can pass to the Commit step.

Commit step is the end of the round and it must have two conditions satisfied in order to be succeeded. The first is that the node must receive a the block by the network, sign it and then broadcasts a commit for that block. In the second condition the node has to wait until it receive at least 2/3 of commits for the block precommitted by the network.

NewHeight step purpose is to gather additional commits for the previously committed block at height *H-1*, in order to block proposals to include more than the minimum 2/3 of commits [Kwo14]

2.4 Ethereum

2.4.1 Background

Ethereum is a blockchain implementation [MvG14] which merges together the concepts of scripting, altcoins, and on-chain meta-protocols. It has the capacity to deploy to the blockchain pre-programed pieces of information and to change them according to needs, basically transforming the system in a decentralized blockchain database allowing the deploy of smart contracts which are scripted pieces of software enforced when some pre-determined conditions are met [But11]. The language in which this smart contracts are written is called Solidity and has a JavaScript plug named web3 for communication with a node.js frameworks.

2.4.2 RPC, Web3 and DApp

Ethereum brought a new concept for application, which is Decentralized Applications, and to considered as one they must meet the following criteria:

1. The application must be completely open-source, it must operate autonomously, and with no entity controlling the majority of its tokens. The application may adapt its protocol in response to proposed improvements and market feedback but all changes must be decided by consensus of its users.

2. The application's data and records of operation must be cryptographically stored in a public, decentralized blockchain in order to avoid any central points of failure.

3. The application must use a cryptographic token (Bitcoin or a token native to its system) which is necessary for access to the application and any contribution of value from (miners / farmers) should be rewarded in the application's tokens.

4. The application must generate tokens according to a standard crytptographic algorithm acting as a proof of the value nodes are contributing to the application (Bitcoin uses the Proof of Work Algorithm).

On this basis, Ethereum as a set of tools like a JASON RPC API and Web3.js plugin, in order to make possible to develop this kind of applications.

JSON RPC API is a stateless, remote procedure call protocol. Primarily this specification defines several data structures and the rules around their processing. It is transport agnostic in that the concepts can be used within the same process, over sockets, over HTTP, or in many various message passing environments. It uses JSON (RFC 4627) as data format [Eth].

Web3.js library is javascript plugin which can connect to a local Ethereum node through giving an interface for the RPC methods, making possible to connect an Ethereum node to a front-end application.

There are also two types of general Ethereum networks, the public one, which is a global scale public network of Ethereum nodes connected, and the private ones, are personalized networks made for small, private use cases.

2.4.3 Transitions and Mining

Instead of transitions of coins like the Bitcoin, Ethereum has objects called accounts. These accounts can be external or contract. External owned accounts are controlled by private keys and contract accounts controlled by their code, with the possibility of external accounts interact with each other by making transactions of contract accounts. These objects are identified each by a 20-byte address and are composed by a nonce, an ether balance, and the contract code.

Transactions in Ethereum are signed data packages sent by external account storing messages created by external or contract accounts and an internal coin called Ether. The Ethereum state transition function is defined as the following algorithm [But11]:

1. Check if the transaction is well formed, if not return an error.

2. Calculate the transaction fee, and determine the sending address from the signature. Subtract the fee from the sender's account balance and increment the sender's nonce. If there is not enough balance to spend, return an error.

3. Initialize GAS = STARTGAS, and take off a certain quantity of gas per byte to pay for bytes in the transaction.

4. Transfer the transaction value from the sender's account to the receiving account. If the receiving account does not exit, create it. If the receiving account is a contract, run the contract's code either to completion or until the execution runs out of gas.

5. If the value transfer failed because the sender did not have enough money, or the code execution ran out of gas, revert all state changes except the payment of the fees, and add the fees to the miner's account.

6. Otherwise, refund the fees for all remaining gas to the sender, and send the fees paid for gas consumed to the miner.

The following picture illustrates the Ethereum State Transition Function:

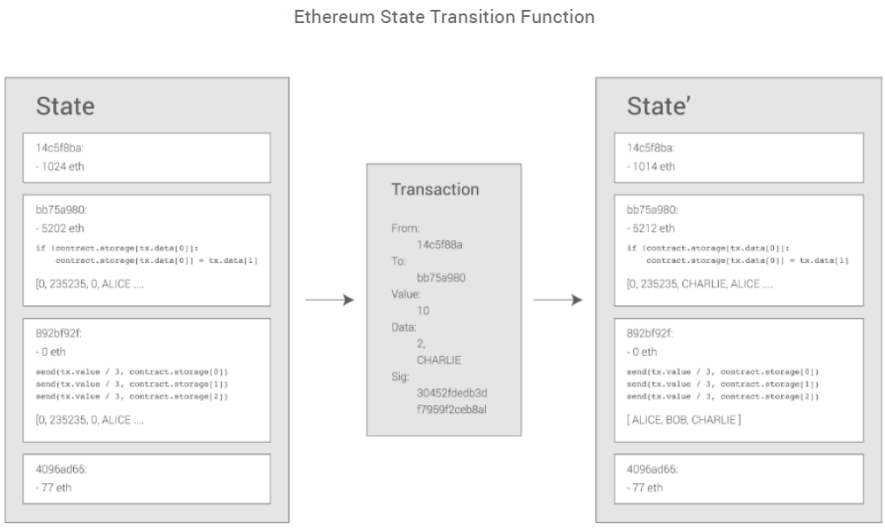

Figure 2.5: Ethereum State Transition [But11]

In terms of mining it's similar to Bitcoin, but it differs on the fact that in Ethereum each block has both the transaction list, the most recent block and the block number, and difficulty are also stored on the block. The mining algorithm is the following [Ref. Ethereum white paper]:

1. Check if the previous block referenced exists and is valid.

2. Check that the timestamp of the block is greater than that of the referenced previous block and less than 15 minutes into the future.

3. Check that the block number, difficulty, transaction root, uncle root and gas limit (various low-level Ethereum-specific concepts) are valid.

4. Check that the proof of work on the block is valid.

5. Let S[0] be the STATEROOT of the previous block.

6. Let TX be the block's transaction list, with n transactions. For all in in 0...n-1, setS[i+1] = APPLY(S[i],TX[i]). If any applications returns an error, or if the total gas consumed in the block up until this point exceeds the GASLIMIT, return an error.

7. Let SFINAL be S[n], but adding the block reward paid to the miner.

8. Check if SFINAL is the same as the STATEROOT. If it is, the block is valid; otherwise, it is not valid.

2.4.4 Ethermint

Ethermint is the Ethereum framework with a Tendermint consensus mechanism instead of the traditional PoW. It enables all the benefits of ethereum without having to run PoW miners. All ethereum components related to miners and network peers hare disable in order to only function the components related with Tendermint consensus. Once all Ethermint nodes are up and running, the access through RPC to a local node and contract management are done exactly the same as in Ethereum. Ethermint only supports private networks.

2.5 Blockchain around the World

Blockchain technology is a subject that has been attracting the business world in a way that many companies are already investing on the development of blockchain applications. For example Deloitte has two Blockchain laboratories, one in Ireland and one in UK, which more than one hundred PoC's applications using blockchain technology. Another example is the first blockchain PoC made in Portugal, APFIPP a Fund Management blockchain application.

Chapter 3

Methodologies

This chapter describes the methodologies used to produce this work and that are in accordance and part of Deloitte Digital Transformation Framework. It makes a theoretical introduction about all the methodologies, how they appeared, advantages and disadvantages and the benefits they carry through out every stages of this work. Later on is going to be explained how these methodologies are used on Deloitte's working platform and how they are applied to this project. Finally, it will present the application of this methodologies as well as a mapping to the results and conclusions of this application.

3.1 Service Design

Services are everywhere, having a special role in developed countries, contributing up to 3/4 of the gross domestic product in most countries, but still receiving less financing than other products [Seg13], making important to study how their creation is done, in order to maximize the profit associated with them.

Due to the increasing demand for user-centric services, companies realize that it is becoming essential to integrate potential future users and stakeholders in the service creation process, giving room for a new discipline, Service Design, which tries to explore and shape the desires of the users in more customer centric services. This new paradigm for services started to gain space and importance in early 1990's, through the efforts in two institutions, Politecnico di Milano in Italy and Koln International School of Design in Germany [Seg13].

Service Design can be defined as the use of a design base way of working when improving or developing people-intensive systems through the engagement of users [Seg13]. The main objective of this discipline is to put the user as the central key in service innovation and requirements development, and is normally achieved by a set of practices which come from other areas of design and research, like workshops, interviews, artifact review and idea summarization.

This concept is a key part of Deloitte Digital practices, being applied in a methodology concept part of the Deloitte Digital Framework.

3.2 Agile Methodologies

The world is in constant change and what is granted now in terms of services and customer desires may not be granted in the future. Companies are increasingly realizing that the old, sequential approach to developing new products simply won't get the job done (Harvard). This new paradigm allied with the need for service design for more user centric services, implies drastic changes on how software is developed in order to make it able to adapt to constant changes of requirements and to keep up with a more competitive market, characterized by the countless startups appearing on the market, making imperative to software to improve quality and change requirements quickly. Therefore a new approach to software development was created, called "Agile Manifesto". Agile Manifesto was written in February of 2001 by seventeen independent-minded software practitioners developing a set of general rules uncovering better ways of developing software:

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

That is, while there is value in the items on the right, we value the items on the left more.
The manifesto has also a set of 12 principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity—the art of maximizing the amount of work not done—is essential.

11. The best architectures, requirements, and designs emerge form self-organizing teams.

12. At regular intervals, the team reflects

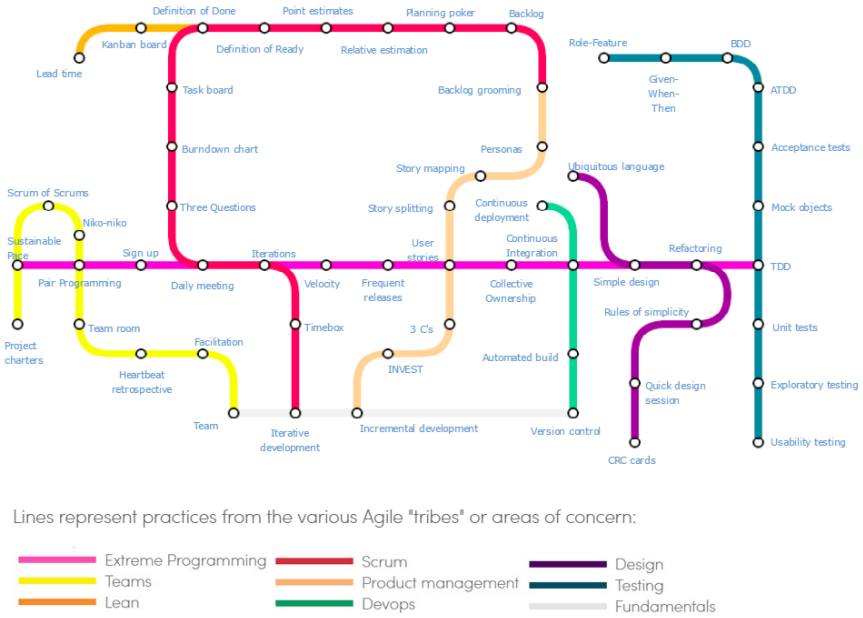

Figure 3.1: Subway Map to Agile Practices [All]

There are several agile practices as illustrated in the figure 3.1, being SCRUM the practice used in Deloitte. SCRUM consists in a process for incrementally building software in complex environments[RJ00]. Next will be explained the details about this particular practice and how it is used by Deloitte Digital.

3.3 Deloitte Digital Transformation Framework

"Digital is the new era of business" [Dela], in other words, having an enterprise digital ecosystem is essential to compete and win in the market today, generating value by leveraging connectivity engaging internal and external stakeholders. Also digitization of processes making them more efficient, experience innovation exploring new services, and data insights to create more transparency on interactions between organizations and stakeholders help establishing the digital ecosystem. All of this points are only possible with an "attitude" change, becoming more flexible and open minded. Following this idea, Deloitte created a framework in order to help its clients to make the transition from the legacy work model to the new digital one.

Deloitte Digital Transformation Framework is not a methodology, but instead is a set of tools and is based in four fundamental principles:

1. Customer Lenses – Costumers know best their own needs.
2. Co-Creation – Work with clients.
3. Prototyping/MVP – MVP definition and rapid prototyping is our secret to align expectations and incorporate feedback faster.
4. Trial And Error – Failing is an opportunity to improve.

Furthermore, this transformation has cycle periods divided in three main categories, which are IMAGE, DELIVER and RUN, as shown in the figure 3.2.

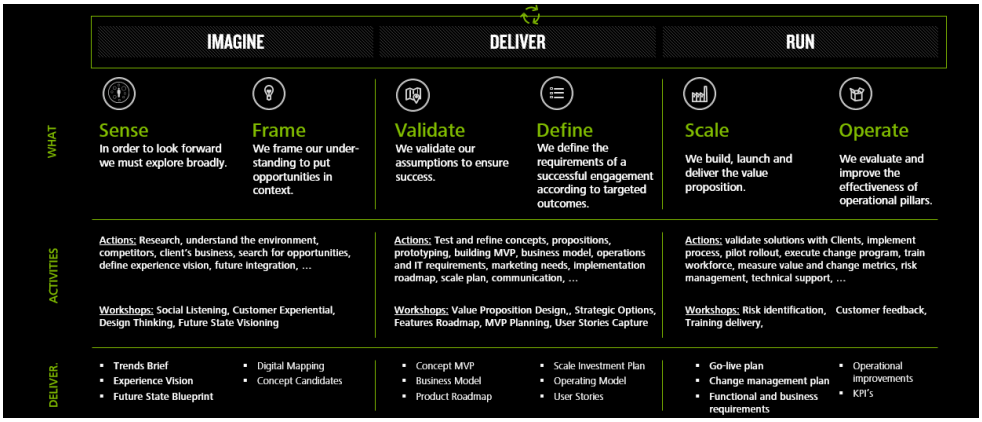

Figure 3.2: Transformation Cycle Phases [RJ00]

IMAGINE is about sensing the outside world, mainly the possible external stakeholders and framing the results in possible opportunities to future exploration. It involves understanding the environment through workshops and social listening. As a result the outcomes are trends belief, experience vision, future state vision, digital mapping and concept candidates.

DELIVER involve validating previous assumptions through prototyping, roadmap implementation and requirement definition, making user stories. RUN involves scaling and operating the value proposition, outputting a go-live plan, operational improvements, functional and business requirements and a change management plan.

Service Design and SCRUM are part of this framework, being the main guidelines in order to fulfill most of the outputs.

3.3.0.1 Service Design

Service Design applied to Deloitte Digital Transformation Framework is composed by a 6 modules journey, as depicted in the figure 3.3:

1. Observe Behavior.

2. Create Personals.
3. Create As-Is Journey.
4. Brainstorm and create to-be journey.
5. Story board the desired journey.
6. Prototype the journey.

Figure 3.3: Service Design Journey [Delb]

Observe Behavior is where information about customers opinions and desires are collected and is done by making specialized interviews, conversations, workshops and quizzes. It is important to adopt a beginners mindset, in order to collect the most personal data possible. Assumptions can induce misconceptions and stereotypes, and can restrict the amount of real empathy built during the interaction. The main objective is to conduct the customer to deliver the most personal non guided information.

During observation mode is important to divide the guideline into three parts: What?, How?, and Why?, in order to drive to deeper levels of observation. What? is about collecting the pain points about what the person thinks about the main subject of the interview. How? is collecting the emotional and physical outputs. For example, do the person appear rushed when answering questions? And Why? is understanding why the person is reacting in a particular way. In general, its important to make a soft introduction, create a big empathy during the interview and explore emotions on the climax of the conversation for greater outputs.

Every service is developed taking into account a certain social group or targeted audience, to give a sense of reality and solidity to the outputs collected in the Observe Behavior phase it is also necessary to create a Persona, which is an archetypes, or a fictional character to better represent these social groups. Each persona has to have personal information, like name, age, country, city, hobbies, strong and weak personality points and a brief resume of their daily life. The following figure illustrates a persona:

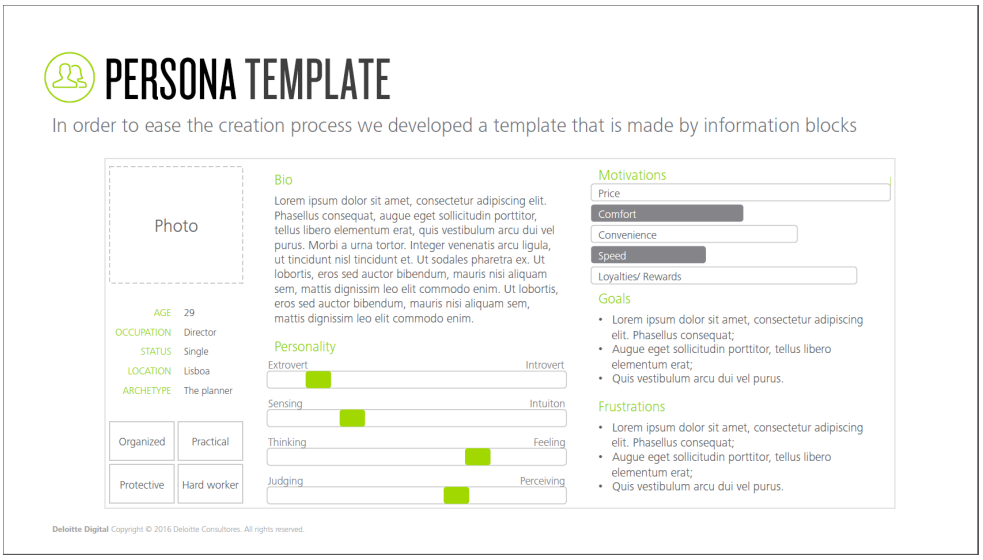

Figure 3.4: Example of a Persona [Delb]

The next phases of Service Design, As-Is Journey, Brainstorm, To-Be Journey and Prototyping, are done using what was observed and using the persona(s) previously created. As-Is Journey consists in creating a customer journey mapping all problems that exists at the moment of the observing behavior, in other words, it is mapping all the data research in the beginning to a use case story using the persona(s) as the main user character. The following picture illustrates a customer journey:

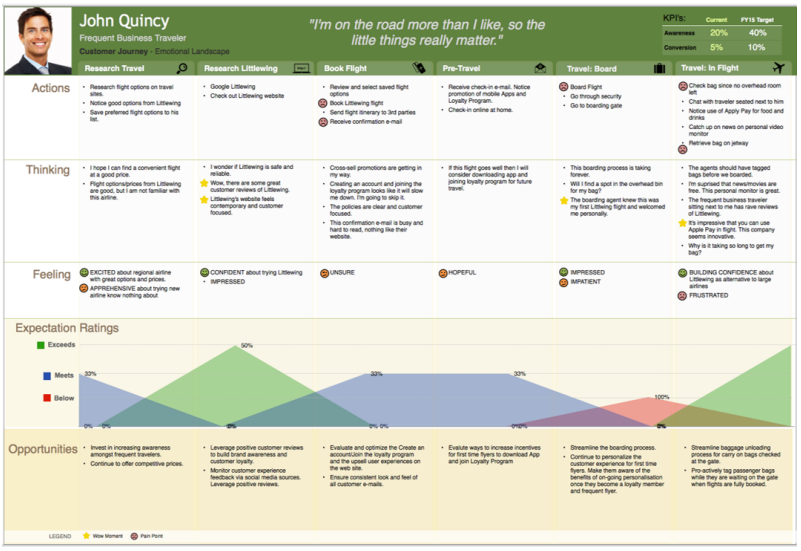

Figure 3.5: Example of a Customer Journey [Delb]

After that there must be some sessions of brainstorming in order to solve the problems mapped in the as-is journey. This part is normally done with interactive workshops and sessions of ideation,

and is an important part of the process because is when customers are put to imagine how they would like the service to be. After brainstorming a to-be journey and a full story of the evolution of the service and is made, is done just like the previous journey just with the objective of mapping the solutions collected in the brainstorm phase. The last last part is to make a simple service prototype (mockup) in order to test with real users.

3.3.0.2 SCRUM

As said before Deloitte practices SCRUM as the main tool for software development. This method divides the project into different iterations or sprints and each sprint is divided in tasks named user stories meant to be in the sprint time. A user story is a phrase describing a desire for a persona / customer and has the following structure, “As a: Who? I want: What? So that: Why”. For example “As a user I want to be able to input my credentials in order to enter the website”. For complex projects, user stories are grouped in themes, epics and activities to help organize and distribute work in context. After definition, is important to give each user story an estimation, but instead of using time duration, an abstract measure called story points is used. Finally is important to organize the user stories in a backlog, create sprints, add user stories and begin working. User stories that are not finalized in a sprint pass directly to the next one. A sprint is the main unit of control of a project and has the duration of between 2 and 4 weeks, for delivering work frequently.

Chapter 4

Requirements and Functionalities

Throughout this chapter, all the requirements and functionalities of this project will be described and explained. It will showcase the output of the methodologies described in chapter 3, like workshops, persona creation, customer journeys and in the end the mapping of these outputs to specific requirements and functionalities are showcased. A client vision of the platform is adopted, in order to give focus to main objectives in workshops.

4.1 Sensing, Brainstorm and Ideation

A research was made using the tools described in Chapter 3, in order to evaluate opinions, feelings, and desires of today customers around the topics of traditional health insurance, social insurance, and decentralized services. This allowed to examine the problems of traditional health insurance and to explore how blockchain technology could integrate a peer to peer health insurance solution.

This study is composed by three components. The first one is about knowing how customers view traditional insurance, in terms of processes, pricing, and customer experience. It is basically a way to probe how insurance is viewed in current society. The second one is to map this knowledge to customer journeys and the third one is to brainstorm on how blockchain could be a solution in scenarios of social insurance. These components are distributed in one quiz and three workshops and the targeted audience are the "Millennials" generation, because they are the future users in the insurance market.

4.1.1 On-line quiz and the first workshop

For the first part of the study an on-line quiz was made dealing about insurance in general, traditional insurance, social insurance, and blockchain technology and an exploratory workshop also took place in Deloitte Digital Porto Studio.

4.1.1.1 On-line quiz

The quiz was sent to all FEUP's student community and consisted in a series of thirteen questions divided in two major sections. The first section was to get basic information like name, birth date and to know if the quizzed person already had an experience with an insurance product and only if the answer was affirmative to this last one question, he could pass to the next section. Basically it served as a filter to make the quiz only for insurance experienced people making it more efficient. The second quiz had questions about general insurance, social insurance and blockchain in order to explore basic public knowledge about these topics. The questions were the following:

1. Which types of insurance have you subscribed?
2. Which channels do you use when interacting with an insurance service (claim and premium processing)?
3. Are you satisfied with existing insurance offers and service quality?
4. What would you improve in these services?
5. Are you familiarized with this type of insurance?
6. What's your opinion about it (advantages/disadvantages)?
7. In case of Peer to Peer Health Insurance, which personal information would you be willing to share with your insurance group?
8. What would make you change from a standard health insurance to a peer to peer one?
9. Do you use digital based services like Airbnb or Uber?
10. Which advantages do you consider to be the most important regarding the format assumed by these services?
11. From these advantages, which ones do you consider opportunities to improve insurance experience?
12. Are you familiarized with this technology? (Blockchain)
13. Applying this technology to Peer to Peer Insurance how do you see the opportunity for every user to have a vote regarding how insurance contracts are made and executed?

Twenty answers were obtained from the quiz, as illustrated in the figure 4.1, being that forty percent of the quizzed are unsatisfied with the quality of today insurance, appointing transparency as their main issues.

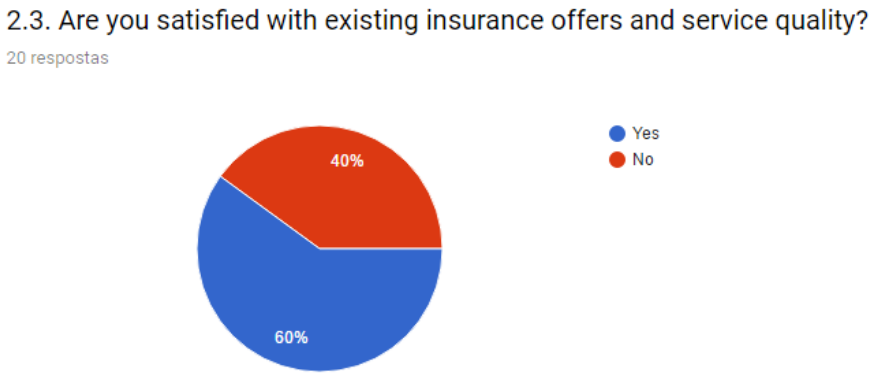

Figure 4.1: Insurance satisfactions result.

4.1.1.2 First workshop

This workshop took place in Deloitte Digital Porto Studio and had 5 participants with an age rage from 22 to 25 years. It had the purpose to discuss the advantages and disadvantages of general insurance and Peer to Peer Insurance and was divided in three parts, **Sensing**, **Learning** and **Brainstorm**.

Sensing involved a guided conversation about general insurance, where the participants had the opportunity to write down on post-its their opinions and experiences with insurance and then put them on a white board with a "good and bad" table. The pain points found were the following:

- Lack of proximity between the service and their clients.
- Processes take too long, like policy creation and claim management.
- Lack of information about claim evaluation.
- Lack of alerts / information about service state.
- Lack of clarity of what's being covered by the insurance service.

From these points, policy creation and claim management were the insurance processes that raised up more complaints.

Learning involved introducing and exploring the subjects of social economy, social insurance and blockchain. It was explained the concept of peer to peer insurance, their advantages and disadvantages and was showcased some already existing companies on the market. Then it was introduced the concept of blockchain, how it started with Bitcoin, how it worked, advantages, disadvantages, what was Ethereum, a smart contract and possible future applications.

Brainstorm involved trying to imagine a peer to peer insurance service with blockchain technology. It was a group conversation with all the elements in the workshop and the main objective was to identify main features for a possible peer to peer health insurance with blockchain.

One output of this brainstorm was the total automation of decentralized insurance policies using blockchain technology. Using the decentralized and trustless nature of this technology, the group came to the conclusion, that the process of policy creation in a peer to peer insurance group could be more transparent, and efficient, by making all the data available with all peers and by clean democratic voting.

The following figures illustrate the environment of the workshop:

Figure 4.2: Sensing conversation

Figure 4.3: Brainstorm conversation with an Ethereum node running in the background

This was a more open workshop not intended primarily to solve any of insurance problems but to explore them in terms of existence.

4.1.2 Persona and As-Is Customer Journey

The first part of the study gave sufficient information in order to create a Persona and an As-Is Customer Journey, which were made in order to personify and portrait all the gathered information.

4.1.2.1 Persona - Geek John

The Persona's name is John and he is going to be the client personification throughout all the project. John is a proactive software developer, has a Master Degree in Informatics Engineering and is a fan of sports, playing regularly with his friends. He is a typical "Millennials" person, always preferring digital services, listening to his friends opinions before taking decisions, looking for shared services to save money and a very traveled person. His characterization can be seen in the figure:

Figure 4.4: Persona - Geek John

4.1.2.2 As-Is Customer Journey

John's customer journey takes place when he is going to open his first bank account and is offered an insurance product. The underwriting process, both on the initial part and in the end when the process finalizes, the time until a claim is issued, claim management, and refund were explored. Also his actions, thoughts and feelings in order to give a more detailed view about all the story were described. The figure 4.5 maps the events to these three types of data (actions, thoughts and feelings).

	[1] Underwriting Process – Pt. 1.	[2] Underwriting Process – Pt. 2.	[3] Time until a claim is issued.	[4] Claim management.	[5] Refund.
ACTIONS	- Talking to the broker (bank manager). - Legal Paper filling.	- Waiting for underwriting conclusion.	None.	- Issuing the claim to the insurance company. - Claim management using an online service. - Delivery of the original expenses in bank office.	- Response received 72h after submission. - Refund of 50% of expenses.
THINKING	- Paper filling is outdated.	- Long Process.	- How my insurance policy is going? - Are there new products available? - Where does my money go?	- Why submitting expenses two times, one online and the other putting? - Too much time to obtain response!	- Too much time for response. - Why only 50% of expenses? - How it's the evaluation made?
FEELING	- Tired. - Bored. - Confused.	- Stressed. - Impatient	- Stressed. - Worried.	- Stressed. - Frustrated.	- Disappointed. - Frustrated. - Confused.

Figure 4.5: Costumer Journey pain points

4.1.3 Claim management - Second and third workshop

With the first workshop and the As-Is Customer Journey, it became clear that both policy creation and claim management were subjects that needed improvement. It was noticed that social insurance wasn't a strong solution because it lacked the "decentralization" also from technology. Claim management still needed to be explored a lot more, so two more workshops directed to claim management were made. The first one was on Deloitte Digital Porto Studio and the second one on Faculdade de Engenharia da Universidade do Porto.

4.1.3.1 Second Workshop

This workshop took place in Deloitte Digital Porto Studio and had 10 participants. It had the purpose to explore how blockchain technology could integrate claim management in social insurance. It was divided as the first workshop, but with a few changes, like group brainstorm changed to a group/team ideation session, which became the second point to be treated in the workshop.

Sensing was almost inexistent, given the fact that none of the participants have had an experience with insurance, so there were no opinions about whether the service is good or bad in terms of transparency, pricing and quality. On the other hand this released time to the second point, team ideation session. This activity consisted in the division of the participants in two separate groups and a "challenge" for them to complete. Each group had to forget about insurance, social insurance and blockchain, and try to idealize how they would manage money in a clean scenario like this:

- There was a common pool of money.
- Each group member deposits money monthly to the pool.

- In case of an health problem, money would be released from the pool to pay for treatments.

- Think how this money should be managed.

In the end both groups presented their model and debated the pros and cons about each model. This activity is basically the idealization of a claim management system in a social insurance service and has the objective to make both groups to come up with a management model. The results weren't the expected, as both groups diverged from the main objective of the activity not focusing particularly on the "money" management (claim management). This had to do to the lack of direction introduced when presenting the rules for the activity. In the last part of the workshop it was given an introduction to the same subjects as the first one, with the addition of a demonstration of an Ethermint node working. It was shown to participants the astonishing difference between the speed of Ethermint and Ethereum and how Ethermint was more appropriate to the solution being developed for this project.

The following figures illustrate the environment of the workshop:

Figure 4.6: Group ideation

Figure 4.7: Group ideation

4.1.3.2 Third Workshop

This workshop was made due to the lacking of information collected in the second one. This one took place in FEUP and had 9 participants. In terms of schedule, this workshop was intended to only focus on claim management and how blockchain could integrate a peer to peer insurance, and only being made a group ideation activity.

There was the same division of the participants as the last workshop, but the scenario was much mode detailed and directed to the purpose of the workshop:

- There was a common pool of money.
- Each group member deposits money monthly to the pool.
- In case of an health problem money would me released form the pool to pay for treatments.
- Think how this money should be managed.
- Have a specific scenario to manage.
- Assume that the group already exists and is functioning and the money is being deposited for a year
- One of the group members falls during a friendly football match and receives treatment in the hospital.
- Besides paying for the initial treatment, the group has to pay for physiotherapist for 2 months.

Given this more directed and specific case, the results were the following:
Group One:

- Reporting with expense and medical report.
- Money withdrawal only with democratic voting between members.
- Decision supported with at least 75 percent of votes.
- Between 75 and 90 percent of votes only 70 percent of money can be released.

Group Two:

- Multiple types of coverage for each member, so different payments
- Reporting with expense and medical report.
- The percentage of money being released depends on the percentage of the up votes.

Analyzing this two situations both groups agreed in a democratic and open voting for claim management, with differences in the percentage of money being released from the pool. In the first group only with 75 percent of the votes can the money be released, and in the second group the percentage of money release varies with te percentage of votes in favor, in this case, 1 "yes" vote is enough for money to be withdrawn from the pool.

After this activity an introduction to social insurance and blockchain was given to the participants and a final discussion about the integration of this technology in social insurance took place. All members agreed that this technology fits well for the purpose of the previews scenarios presented in the last activity, being that blockchain brings the peer to peer concept and transparency to the service.

The following figures illustrate the environment of the workshop:

Figure 4.8: Presentation in FEUP's Auditorium.

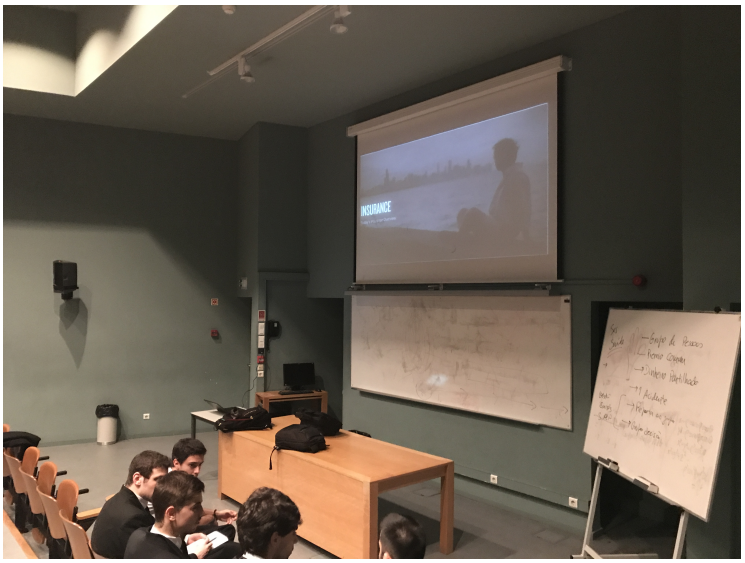

Figure 4.9: Group Ideation

4.1.4 To-Be Customer Journey

After the research and workshops, was gathered enough information to idealize a To-Be customer journey. This journey involves John creating a peer to peer insurance group in a mobile application, adding all his friends, and in the future when one of them has a claim, they all decide democratically about money use to pay for the claim.

4.2 Solution - Platform overview

Taking into account the study previously presented and the use cases for policy creation and claim management, the presented solution involves creating an insurance policy and claim management services for a peer to peer health insurance supported by blockchain technology. In this section will be explained and exposed the actors involved in the system, problems and opportunities, main requirements and functionalities and the overall concept of the system.

4.2.1 Actors

Considering the cases explored in the workshops, the main actor to be considered is the insurance client. The main reason for this, is that, as said earlier, the study was meant to be addressed only to final consumers and their desires. Because this is a service meant to be integrated and provided by an already functioning insurance company, the insurance company has the role of a secondary actor in the system.

4.2.2 Problems and Opportunities

As concluded by the research, both insurance subscription and claim management are problematic areas that need solution. Their problems have the most profound impact on user experience and can compromise insurance in a long term, but from another perspective, these problems can be opportunities to change the service and adapt to the future.

With this mindset, its possible to map the major problems to possible solutions. The objective is to divide the insurance service into two main areas: insurance subscription and claim management and analyze which problems are solved with only a web version of the service, a standard peer to peer insurance solution, and a peer to peer solution with blockchain as main technological feature. The following table maps the problems to the opportunities based on these three technological solutions:

- For Insurance Subscription

Table 4.1: Insurance Subscription

Pain Points	**Opportunities**		
	Online Service	*Standard Social Insurance*	*Blockchain*
Non personalized services	-	Social Insurance permits the client to personalize its own service	-
Expensive services	-	Social Insurance is cheaper than traditional insurance	With blockchain, technological expenses are cheaper, turning overall services cheaper too
Paper filling subscription	Online subscription using digital devices	Social Insurance is based on digital services	-

- For Claim Management

Table 4.2: Claim management Opportunities

Pain Points	**Opportunities**		
	Online Service	*Standard Social Insurance*	*Blockchain*
Personal documentation livery	Online documentation submission	Social Insurance uses online management platforms	-
Time-consuming process	-	Social Insurance is quicker to evaluate claim situations	With blockchain, the service becomes technologically decentralized becoming more efficient and faster
Inconclusive results	-	Social Insurance permits more documentation transparency	-

4.2.3 Overall Solution Concept

The main objective for this solution is to create a blockchain based service for peer to peer health insurance claim management and policy creations. The overall characteristics of the platform are:

- Policy Creation Module
  - Add Friends to an insurance policy
  - Select a coverage for the policy
  - Send the request to all users involved
  - Accept or Reject an insurance policy request
- Claim Management Module
  - View all claim requests
  - Accept of Reject each one
  - Create one request for approval
  - Send to the platform all the legal documents involving the claim

4.2.4 Requirements and Functionalities

In terms of specific requirements and Functionalities, they are the following:

- Sign in Screen
  - The system should allow the user to register in the platform by filling a form with email, username, password, phone number and home address.
  - The system should allow the user to register in the platform by using social networks like Facebook or Twitter.
- Login Screen
  - The system should allow the user to login using a username and a password.
  - The system should allow the user to login using social networks accounts like Facebook and Twitter.
  - The system should redirect the user to community management page after a successful login.
- P2P Insurance contract management screen
  - The system should allow the user to see a list of P2P Insurance contract on which he belongs to.
  - The system should allow the user to create a new P2P Insurance contract.

  - The system should allow the user to see a list of requests to join P2P insurance contract.
  - The system should allow the user to accept/reject each request.
  - The system should allow the user to select a P2P Insurance contract.

- P2P Insurance contract creation screen
  - The system should allow the user to select group size number.
  - The system should allow the user to add users as policyholders to the contract.
  - The system should allow the user to select coverage offers to add them to the contract.
  - The system should allow the user to input the desired monthly premium to be paid by each members of the group.
  - The system should allow the user to send the contract, as requested to the others policy holders.

- P2P Insurance group contract screen
  - The system should allow the user to see the policyholders.
  - The system should allow the user to remove policyholders that canceled the contract [Contract Creator]
  - The system should allow the user to add new policy holders to replace the removed ones [Contract Creator].
  - The system should allow the user to see the coverage supported by the contract.
  - The system should allow the user to see the fee that is being taxed by the Insurance Company.
  - The system should allow the user to see the premium.
  - The system should allow the user to see the money pool balance.

- P2P Insurance group claim management screen
  - The system should allow the user to see all group claim requests.
  - The system should allow the user to create a new claim.
  - The system should allow the user to select a claim.

- P2P Insurance group claim screen
  - The system should allow the user to accept/reject a claim.
  - The system should allow the user to view the state of the claim.
  - The system should allow the user to view the details of the claim.

- P2P Insurance group claim creation screen
  - The system should allow the user to input specific claim data.
  - The system should allow the user to send the claim request.

4.2.5 Business Model Overview

4.2.5.1 Context

A business model was made in order to give consistency to the solution. This is a conceptual business model because it lacks real scenario validation. Blockchain is not considered for this model, being focused only to P2P Insurance. Blockchain does not appear to have a great impact in this proposed business model. However it is expected a great impact in its operational utilization.

4.2.5.2 Model Diagram

The business model is illustrated in the following figure:

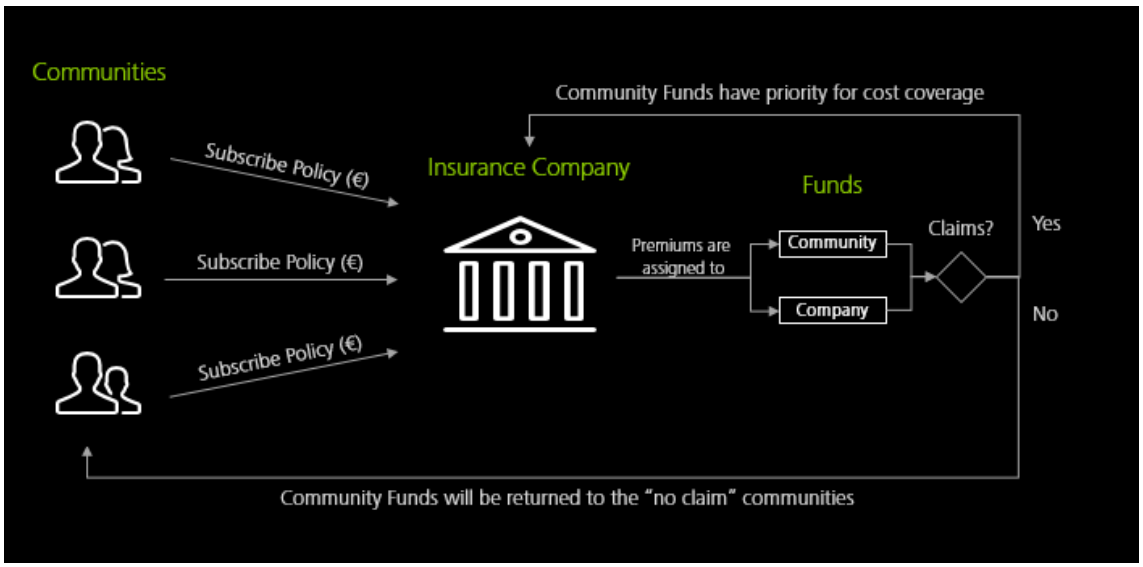

Figure 4.10: Business model.

This model is composed by two types of entities, Communities and the Insurance Company. Communities are the groups of users that are aggregated in insurance policies or insurance groups. The Insurance Company is the company responsible for providing P2P Insurance service.

The flux of this model is the following:

1. Each user can be in one or more insurance community.
2. Each insurance community has a premium associated to its members
3. Each community pays the sum of all users premiums to the Insurance Company.
4. The money is split into the community pool and the insurance company.
5. When a claim situation appears, the community pool has priority to pay for the costs. Only if the cost is superior to the money existing in the pool, is that is the Insurance Company acting as a re-insurer covering the costs.

This model can be exemplified in the following customer journey using John as main Persona. The customer journey is described in the following steps:

1. John and his friends join an health insurance community.
2. Each member of the group pays a monthly premium.
3. A percentage of the sum of the users premiums is for the community pool and the other percentage for the Insurance Company.
4. One day one of Johns friends have an accident while playing football.
5. He reports a claim for a three week physiotherapy recovery to the P2P Insurance platform.
6. The costs can be supported by the community pool and the money is withdrawn from the pool to pay for the costs.

4.2.5.3 Benefits

This simplified business model has some expected benefits like having clients motivation to attract more clients a higher number of policies subscription, fraud prevention because the sense of community helps people behave more properly, smaller number of claims and reduced costs for the insurance company due to the fact that for lower claims the community pools pays the costs.

Chapter 5

Architecture

This chapter will be focused on the architecture used in the prototype. It's going to explain the architectural basis of an Ethereum/Ethermint decentralized application and how is modified to serve the purpose for this project. Furthermore it explains the ideal architecture for a decentralized application with this characteristics. Also is important to refer that in this case we have two approaches for the architecture, one for the disposition of Ethermint nodes between clients, and the other about the application itself.

5.1 First architecture planning - Deloitte Digital Architecture

5.1.1 Digital Architecture

Deloitte Digital is a new competency group at Deloitte Portugal. Being a new structure in the firm comes with a strong necessity to build up foundations and a structure quality in order to develop solutions to attack client needs. So, in response to this need, a prototype architecture was developed, capable of filling up this necessity.

Digital Architecture, as is called, is the base solution for any project either web or mobile and it is based on recent trends of web development. It has a modular architecture to insure a more consistent adaptation do changes and was made in a very generic form in order to be adaptable to any client needs and for faster deployment. This architecture was composed initially by three main layers, Front-End Layer, Back-End Layer and Data Layer. Later a Middleware Layer was added and other components mentioned more bellow. The Layers and their components can be viewed in the Figure:

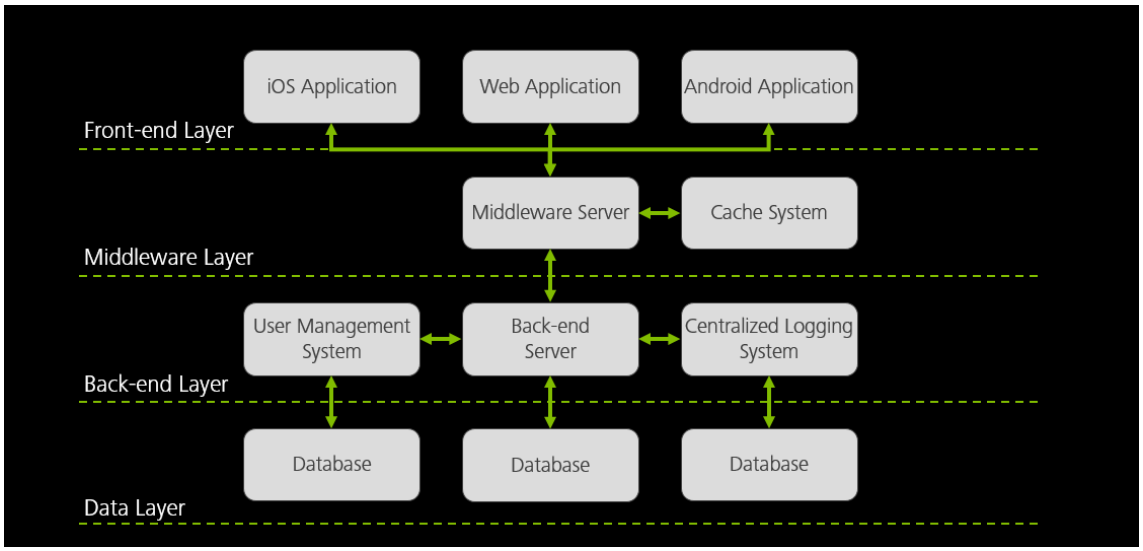

Figure 5.1: Digital Architecture Layers.

The Web Application is divided in two components, a web server built with node.js and express.js and a presentation layer run on top of Angular.js.

The Middleware involves a web server built over node.js using Falcor.js. This feature adds up for a more secure system because adds an access layer to the back-end. The system is simpler because the back-end, made generally of multiple servers only has to know how to connect to the Middleware making general the system more reliable and modular.

The back-end is built over node.js with hapi.js and has the responsibility to keep and manage access to data and business logic. It has a REST API which allows other applications to interact with it, access, manage and store data through a resource based interface. It has a Centralized logging system, which is important to keep current details of everything that happens in the application. It has also a User Management System for an empowered architecture with authentication and authorization capabilities.

The Database component is responsible to store all data being managed by the application and is made using MongoDB.

5.1.2 Digital Architecture - Decentralized Application

The first architecture planned for this project was transforming the Digital Architecture in order to meet the requirements for a decentralized application still using Ethereum as blockchain framework. The Middleware Layer, the components, cache system, user management system and centralized logging system where removed in order to give more simplicity to the prototype, keeping the centralized database in MongoDB. Furthermore a new component was integrated, constituted by an endpoint connection to an Ethermint node built over node.js with web3.js plug in for communication. This architecture is displayed in the figure:

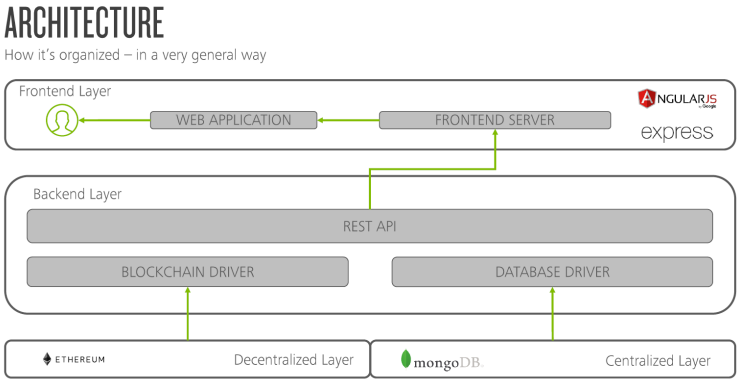

Figure 5.2: Digital Architecture in a DAPP format.

Apart from the application architecture, it was important to understand how this would work in a real case scenario with insurance companies and multiples clients organized in multiple insurance groups, so, it was designed an architecture for node distribution across the participants in the blockchain network. This design considers the architecture of the application and is applied for both insurance companies and clients. The following figure shows the overall node distribution:

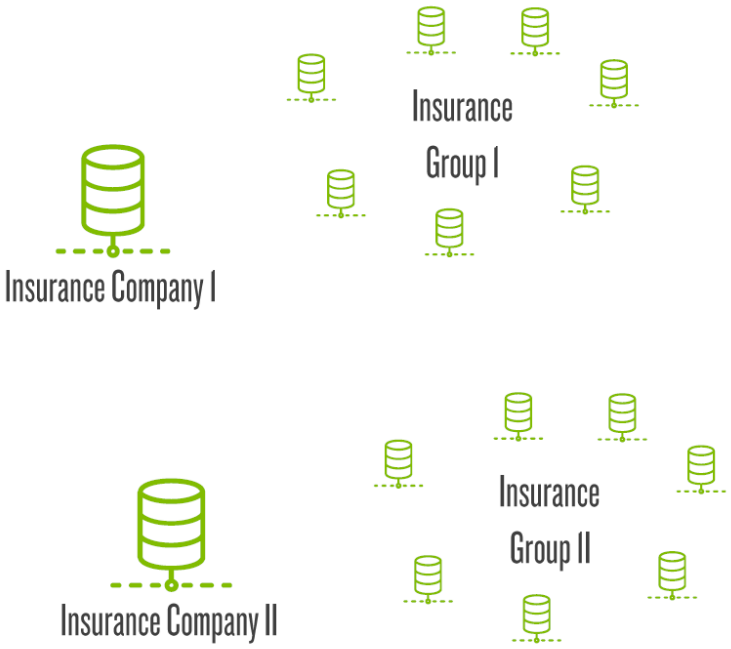

Figure 5.3: Blockchain nodes architecture.

The main characteristics of this design are the following:

- Every user in the system uses a light node.

- Light nodes can keep track of information, but can't contribute to mining.
- The insurance company is a node in the system.
- The network must have more than one insurance company.
- Only insurance companies contribute for transaction validation.
- Use of Ethereum light nodes in smart-phones.

This architecture has some flaws in terms of decentralization and of transferring decision power to the user.

In this concept, the user has no "decision power", because in order for the application to execute on mobile devices it must have light nodes. Light nodes are nodes that only have access to information but doesn't contribute to data validation. So, the hole concept loses "points in favor" regarding the necessity for a more decentralized service. As seen on chapter 2, validating data on a blockchain based on the Ethereum framework requires computational power. Aside from being too costly in terms of disk space, proof of work based blockchains require a big amount of computational power. Because this prototype has a centralized database, there's a need for a back-end which, makes not fully decentralized, make possible the existence of data that is only in the possession of the company offering the service.

Summarizing, this architecture is able to work with the Ethereum framework, but due to the lack of optimization to work with smartphones and more regular client based devices, it makes the concept loosing a full decentralized architecture.

5.2 Ethermint - Ideal Architecture

As seen previously, Ethereum does not constitute the ideal blockchain framework to develop this solution. This has to due with the fact that regular digital platforms like desktops or smartphones, do not meet the minimum processing requirements in order to validate transactions based on Proof of Work. This consensus system need high processing power which is incompatible with such devices and besides that, is very inefficient and has an high energy cost. So, in order to make the prototype, it was important to look for and to choose a blockchain platform capable of running validation nodes on low processing devices. Therefore, Ethermint was chosen to replace Ethereum in the development. This choice had to do with two main factors. The first one, and most important, it has a low processing consensus system capable of running in smartphones. The second one is to do with the fact that it maintains all the development API existent Ethereum which is the framework used in Deloitte Blockchain PoC's like APFIPP facilitating the development.

Another solution that was in consideration, was to use light nodes for clients and to insert the application into Ethereum public network. In this solution the whole public network provided sufficient processing power to validate and to make safe all transactions in the blockchain. The reason why this was not adopted as solution has to do with the fact that in terms of infrastructural costs, it

was always dependent on the public value of *Ether*, Ethereum's internal coin, used to manage all information inside the network like managing smart contracts and that is not very constant along time, causing potential problems in infrastructural costs on smart contract management.

Figure 5.4: Ether cost variation since the beginning of 2017 [Ind].

5.2.1 Application and node architecture

For the integration of Ethermint into this new architecture a few changes were made. In the application level, the centralized database in MongoDB was discarded, because, if the idea is to have a total decentralized service, everything has to be the blockchain. Not having a centralized database, there is no need for a back-end server, resulting in a direct connection between the front-end and the blockchain endpoint.

The new node architecture, as illustrated in the figure 5.5 to support this prototype has the following characteristics:

- Every user in the system is a validator node in the network.

- Only one insurance company exists and is a validator node.

Figure 5.5: Fully decentralized node architecture.

Chapter 6

Project Implementation

This chapter describes the implementation of the solution presented in previous chapters. It starts with a brief presentation of the technologies used and the environment setup used in development and will end with the implemented functionalities followed by images of the application.

6.1 Technologies and Development Environment

This prototype is composed of two main layers, Front-End layer and Blockchain Layer. For the Front-End Layer were used the technologies already presented in Deloitte Digital Architecture, namely, Node.js and Express.js and angular.js. For the Blockchain Layer was used Ethermint Blockchain Framework, which uses Solidity for writing smart contracts, and Web3.js library to enable communication between the Front-End and de Blockchain Layer.

The working environment used to develop this application consisted in two centOs servers to allocate the Ethermint nodes and is divided into four groups, DevOps, contract creation and deployment, web3.js integration and front-end development.

Relative to DevOps, were used two platforms, JIRA and Bitbucket. JIRA permits to manage all the areas involving SCRUM, like user stories creation and management, task creation and management and sprint management. Bitbucket was the GIT repository for all the code.

For contract creation and deployment was used a framework called truffle which permits [Fra]:

- Built-in smart contract compilation, linking, deployment and binary management.
- Automated contract testing for rapid development.
- Scriptable, extensible deployment and migrations framework.
- Network management for deploying to any number of public and private networks.
- Interactive console for direct contract communication.
- Configurable build pipeline with support for tight integration.
- External script runner that executes scripts within a Truffle environment.

Truffle has its own file structure, presented in the figure 6.1. The "contracts" folder is where all solidity contracts are written, the "build" folder is where compiled contracts stored and "migrations" folder is where migrations to the blockchain are defined. A migration of a contract is the deployment to de node. All interactions are done through the terminal.

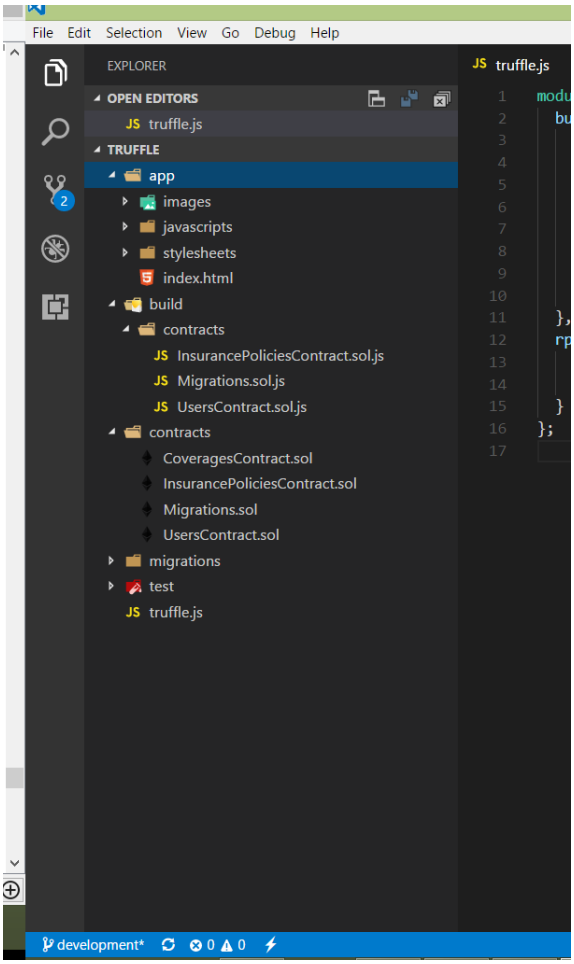

Figure 6.1: Truffle file system.

The code of the smart contracts and of the web3.js plugin integration is available in the appendix A.

6.2 Implemented Functionalities

In the previous chapter was presented a list of functionalities to implement on a Peer to Peer Health Insurance with Blockchain.

However due to the small time that normally is available to develop a dissertation not all functionalities were developed and there was a necessity to prioritize the best functionalities in order to make the prototype.

For this prototype it was chosen the insurance subscription use case, where a user creates an insurance group, invites people and waits for confirmation. The following list presents the main functionalities developed for this proof of concept:

- Login Screen - figure 6.2
  - The system should allow the user to login using a username and a password.
  - The system should redirect the user to community management page after a successful login.
- P2P Insurance contract management screen - figure 6.3
  - The system should allow the user to see a list of P2P Insurance contract on which belongs to.
  - The system should allow the user to create a new P2P Insurance contract.
  - The system should allow the user to see a list of requests to join P2P insurance contract.
  - The system should allow the user to accept/reject each request.
  - The system should allow the user to select a P2P Insurance contract.
- P2P Insurance contract creation screen - figure 6.4
  - The system should allow the user to select group size number.
  - The system should allow the user to add users as policyholders to the contract.
  - The system should allow the user to select coverage offers to add them to the contract.
  - The system should allow the user to input de desired monthly premium to be paid by each members of the group.
  - The system should allow the user to send the contract as request to the others policy holders.
- P2P Insurance group contract screen - figure 6.5
  - The system should allow the user to see the policyholders.
  - The system should allow the user to see the coverage supported by the contract.
  - The system should allow the user to see the premium.

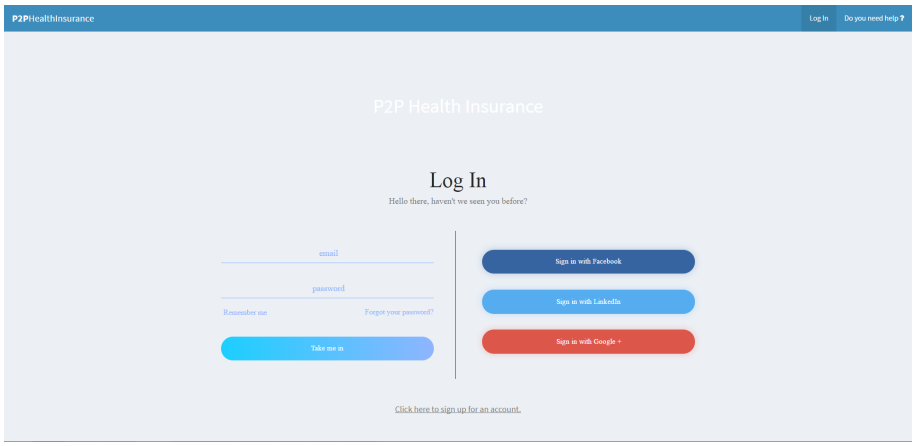

Figure 6.2: Login screen.

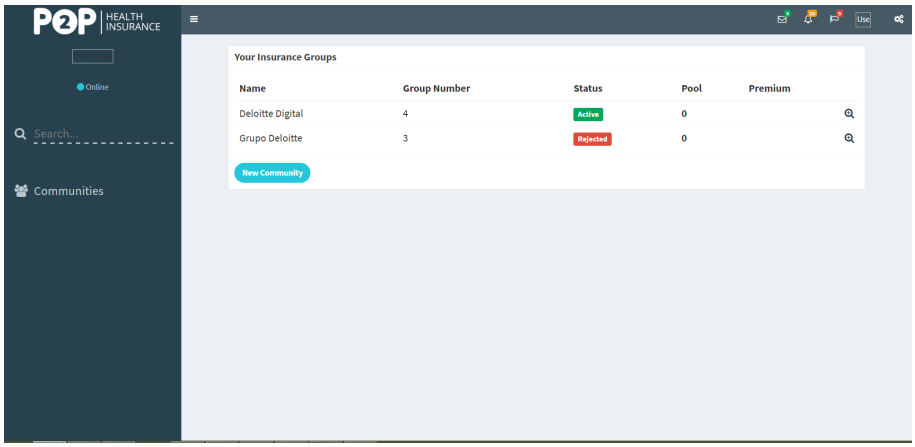

Figure 6.3: Insurance Policies / Groups management screen.

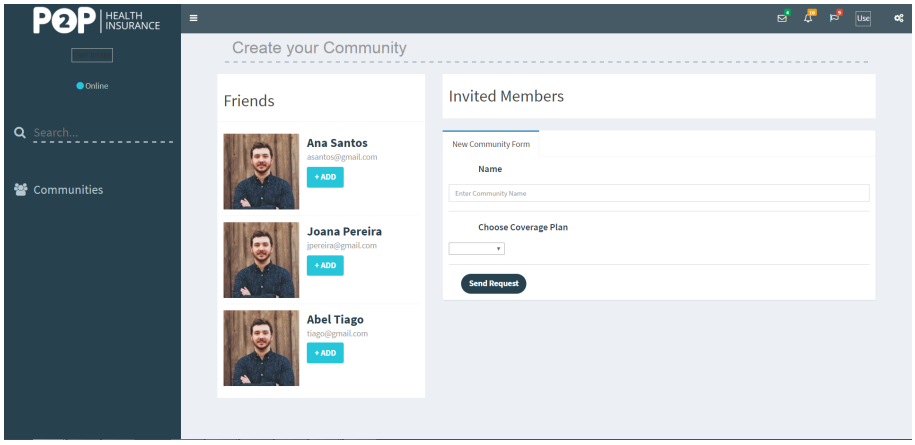

Figure 6.4: New insurance policy / group screen.

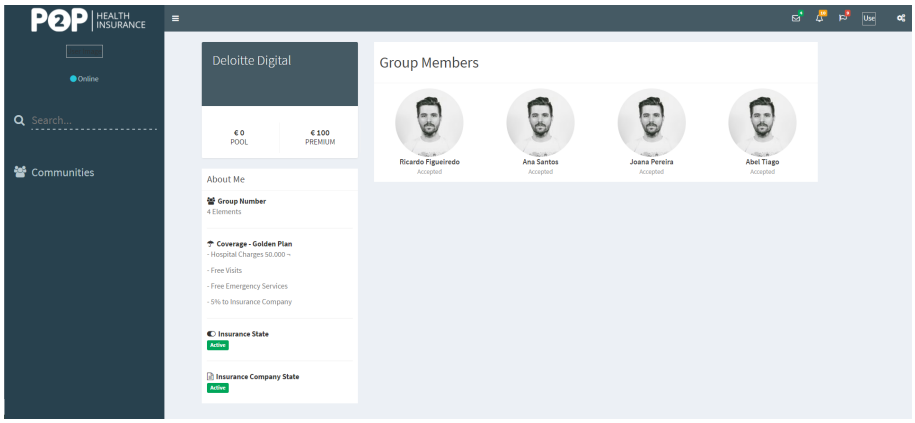

Figure 6.5: Insurance Policy / Group screen.

All of these functionalities where divided in user stories and each one grouped in 1 week sprints.

The following picture shows an example of the popup display when a transaction is validated:

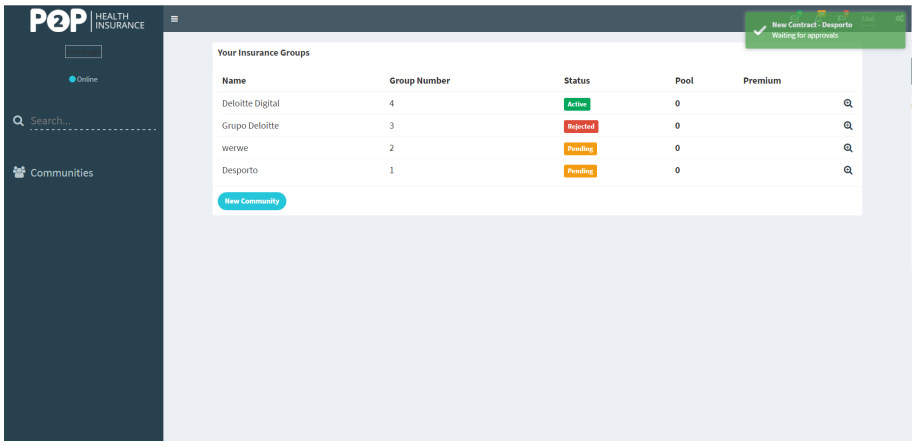

Figure 6.6: Creation insurance policy - Blockchain notification transaction approval.

This front-end design is based on a prototype from a previous Deloitte thesis about "Peer to Peer Health Insurance" [dC16].

Chapter 7

Conclusion and Future Development

This work permitted to understand the in-depth of Traditional Insurance, Social Insurance and Blockchain, their future and how these subjects can relate in favor of innovation.

The main objective of this project was the study of the integration of a Blockchain platform in a Peer to Peer Health Insurance service. Initially was made a research of the state of the art of Traditional Insurance, Social Insurance and Blockchain technology. Then was made a sensing study to find out what are the desires of the future users of insurance, their vision about traditional insurance, how they view the "perfect" service, and their opinion about social insurance. With the help of the information gathered in this last activity, became possible to idealize the introduction of blockchain technology on a Peer to Peer Health Insurance service, namely in the part of insurance policy creation and claim management, being that only insurance policy creation was implemented in a prototype.

There were some problems during the development of this project. One of them stands with the fact that is still difficult to explain "ordinary" people the concept of blockchain, decentralized computing and it benefits for future services. Another problem appeared during this work was the need to change the blockchain technology from Ethereum to Ethermint, where a new technology needed to be learned.

The next steps for this project are to implement claim management module and to explore other possibilities for blockchain usage in insurance.

References

[All] Agile Alliance. https://www.agilealliance.org/.

[Bac02] Adam Back. Hashcash - a denial of service counter-measure. http://www.hashcash.org/papers/hashcash.pdf, aug 2002.

[Ber16] Oliwia Berdak. Disrupting finance: Social insurance. Technical report, FORRESTER, 2016.

[BSCG+] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza†. Zerocash: Decentralized anonymous payments from bitcoin. Technical report, MIT.

[But11] Vitalik Buterin. Ethereum white paper: A next generation smart contract & decentralized application platform. Technical report, Ethereum, 2011.

[Cap16] Capgemini. Top 10 trends in insurance in 2016. Technical report, Capgemini, 2016.

[CSB] Dave Cohen, David Schwartz, and Arthur Britto. Ripple. Technical report.

[dC16] Armindo Barbosa de Carvalho. *Peer to Peer Health Insurance*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2016.

[Dela] Deloitte. Digital transformation framework [powerpoint presentation].

[Delb] Deloitte. Service design - training kit [powerpoint presentation].

[EMC16] EMC2. Making sense of today's insurance market. Technical report, EMC2, 2016.

[Eth] Ethereum. Json rpc api. https://github.com/ethereum/wiki/wiki/JSON-RPC.

[Fra] Truffle Framework. Truffle framework. http://truffleframework.com/docs/.

[Gee] Block Geeks. Proof of work vs proof of stake: Basic mining guide. https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/.

[Ind] World Coin Index. Ethereum eth. https://www.worldcoinindex.com/coin/ethereum.

[JB10] Lum Jia Jun and Brandon. Implementing zero-knowledge authentication with zero knowledge. Technical report, 2010.

[Jos16] Anish Raj & Prasad Joshi. Changing face of the insurance industry. Technical report, InfoSys, 2016.

[KN12] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. Technical report, 2012.

REFERENCES

[Kwo14] Jae Kwon. Tendermint: Consensus without mining. Technical report, Tendermint, 2014.

[MaDH+16] William Marcoux, PK Paran andAndrew Dyson, Adam Hartley, Melanie James, Scott Thiel, Patrick van Eecke, and Anthony Day. Digital transformation in the insurance sector. Technical report, DLA PIPER, 2016.

[Maj16] Majesco. Future trends: A seismic shift underway. Technical report, Majesco, 2016.

[Mas] Papper Masters. https://www.papermasters.com/history-insurance-industry.html.

[MvG14] Michael Mainelli and Chiara von Gunten. Chain of a lifetime: How blockchain technology might transform personal insurance. Technical report, Long Finance, 2014.

[Nak09] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Bitcoin, 2009.

[RJ00] Linda Rising and Norman S. Janoff. The scrum software development process for small teams. *IEEE SOFTWARE*, 2000.

[Sah16] Arindam Saha. Peer-to-peer insurance back to basics. *Research Gate*, mar 2016.

[Sch09] Sebastian Schich. Insurance companies and the financial crisis. *FINANCIAL MARKET TRENDS*, 2009.

[Seg13] Fabian Segelstrom. *Stakeholder Engagement for Service Design*. PhD thesis, Department of Computer and information Science Linkoping University, SE-581 83 Linkoping, Sweden, 2013.

[Vas] Pavel Vasin. Blackcoin's proof-of-stake protocol v2. Technical report.

[Waj16] Henri Wajsblat. 4 business trends to watch in the insurance industry for 2017. http://iireporter.com/4-business-trends-to-watch-in-the-insurance-industry-for-2017/, mar 2016.

Appendix A

Contracts Code

```
pragma solidity ^0.4.2;

contract InsurancePoliciesContract {

  address owner;

  enum State{
    NONE,
    WAITING,
    ACCEPTED,
    REJECTED
  }

  struct Stakeholder{
    bytes32 userId;
    State state;
  }

  struct Coverage{
    bytes32 coverageName;
    bytes32[] coverageOffers;
    uint coveragePremium;
  }

  struct InsurancePolicy{
    bytes32 insurancePolicyId;
    bytes32 insurancePolicyName;
    bytes32 creatorUserId;
    uint groupNumber;
    uint sharedPool;
    Coverage coverage;
    State insurancePolicyState;
    uint  acceptedStakeHoldersNumber;
```

```
34      Stakeholder insuranceCompany;
35      bytes32[] policyHoldersIds;
36      mapping(bytes32 => Stakeholder) policyHolders;
37    }
38
39    uint lastBlockNumberUsed;
40    bytes32 lastBlockHashUsed;
41
42    bytes32[] insurancePoliciesIds;
43    mapping(bytes32 => InsurancePolicy) insurancePolicies;
44    mapping(bytes32 => bytes32[]) policiesOfPolicyHolders;
45
46    event CreateInsurancePolicy(address indexed from, bytes32 insurancePolicyName,
          bytes32 creatorId, uint groupNumber, bytes32 insuranceCompanyId, bytes32[]
          policyHoldersIds);
47    event ChangePolicyHolderState(address indexed from, bytes32 insurancePolicyId,
          bytes32 policyHolderId, State newState);
48    event ChangeInsuranceCompanyState(address indexed from, bytes32 insurancePolicyId
          , State newState);
49    event UpdateInsurancePolicyState(address indexed from, bytes32 insurancePolicyId,
           State newState);
50
51    function InsurancePoliciesContract(){
52      owner = msg.sender;
53    }
54
55    function createInsurancePolicy(bytes32 _insurancePolicyName, bytes32
          _creatorUserId, bytes32 _insuranceCompanyId, bytes32[] _policyHoldersIds,
          bytes32 _coverageName, bytes32[] _coverageOffers, uint _coveragePremium)
          public payable returns (bool){
56
57      uint128  wager = uint128(msg.value);
58      lastBlockNumberUsed = block.number - 1 ;
59      lastBlockHashUsed = block.blockhash(lastBlockNumberUsed);
60      uint128  lastBlockHashUsed_uint = uint128(lastBlockHashUsed) + wager;
61      bytes32 _insurancePolicyId = sha3(lastBlockHashUsed_uint);
62
63      insurancePoliciesIds.push(_insurancePolicyId);
64      insurancePolicies[_insurancePolicyId].insurancePolicyId = _insurancePolicyId;
65      insurancePolicies[_insurancePolicyId].insurancePolicyName =
            _insurancePolicyName;
66      insurancePolicies[_insurancePolicyId].creatorUserId = _creatorUserId;
67      insurancePolicies[_insurancePolicyId].groupNumber = _policyHoldersIds.length;
68      insurancePolicies[_insurancePolicyId].sharedPool = 0;
69      insurancePolicies[_insurancePolicyId].insurancePolicyState = State.WAITING;
70      insurancePolicies[_insurancePolicyId].acceptedStakeHoldersNumber = 0;
71      insurancePolicies[_insurancePolicyId].insuranceCompany.userId =
            _insuranceCompanyId;
72      insurancePolicies[_insurancePolicyId].insuranceCompany.state = State.ACCEPTED;
```

```
73      insurancePolicies[_insurancePolicyId].policyHoldersIds = _policyHoldersIds;
74      for(uint i = 0; i < _policyHoldersIds.length; i++){
75
76        Stakeholder memory _phTemp;
77        _phTemp.userId = _policyHoldersIds[i];
78        if(_phTemp.userId == _creatorUserId){
79            _phTemp.state = State.ACCEPTED;
80            insurancePolicies[_insurancePolicyId].acceptedStakeHoldersNumber++;
81        }
82        else{
83          _phTemp.state = State.WAITING;
84        }
85
86        insurancePolicies[_insurancePolicyId].policyHolders[_policyHoldersIds[i]] =
              _phTemp;
87        policiesOfPolicyHolders[_policyHoldersIds[i]].push(_insurancePolicyId);
88      }
89
90      insurancePolicies[_insurancePolicyId].coverage.coverageName = _coverageName;
91      insurancePolicies[_insurancePolicyId].coverage.coverageOffers = _coverageOffers
            ;
92      insurancePolicies[_insurancePolicyId].coverage.coveragePremium =
            _coveragePremium;
93
94      CreateInsurancePolicy(msg.sender, _insurancePolicyName, _creatorUserId,
            insurancePolicies[_insurancePolicyId].groupNumber, _insuranceCompanyId,
            _policyHoldersIds);
95
96      return true;
97    }
98
99    function changePolicyHolderState(bytes32 insurancePolicyId, bytes32
          policyHolderId, State newState) returns (bool){
100
101     insurancePolicies[insurancePolicyId].policyHolders[policyHolderId].state =
            newState;
102
103     if(newState == State.REJECTED){
104         insurancePolicies[insurancePolicyId].insurancePolicyState = State.REJECTED;
105         UpdateInsurancePolicyState(msg.sender, insurancePolicyId, newState);
106     } else if (newState == State.ACCEPTED){
107         insurancePolicies[insurancePolicyId].acceptedStakeHoldersNumber++;
108         if(insurancePolicies[insurancePolicyId].acceptedStakeHoldersNumber == (
                insurancePolicies[insurancePolicyId].groupNumber)){
109             insurancePolicies[insurancePolicyId].insurancePolicyState = State.
                    ACCEPTED;
110             UpdateInsurancePolicyState(msg.sender, insurancePolicyId, newState);
111         }
112     }
```

```
113
114     ChangePolicyHolderState(msg.sender, insurancePolicyId, policyHolderId, newState
            );
115
116     return true;
117   }
118
119   function changeInsuranceCompanyState(bytes32 insurancePolicyId, State newState)
          returns (bool){
120
121     insurancePolicies[insurancePolicyId].insuranceCompany.state = newState;
122
123     if(newState == State.REJECTED){
124         insurancePolicies[insurancePolicyId].insurancePolicyState = State.REJECTED;
125         UpdateInsurancePolicyState(msg.sender, insurancePolicyId, newState);
126     } else if(newState == State.ACCEPTED){
127         insurancePolicies[insurancePolicyId].acceptedStakeHoldersNumber++;
128         if(insurancePolicies[insurancePolicyId].acceptedStakeHoldersNumber == (
                insurancePolicies[insurancePolicyId].groupNumber)){
129             insurancePolicies[insurancePolicyId].insurancePolicyState = State.
                    ACCEPTED;
130             UpdateInsurancePolicyState(msg.sender, insurancePolicyId, newState);
131         }
132     }
133
134     ChangeInsuranceCompanyState(msg.sender, insurancePolicyId, newState);
135
136     return true;
137   }
138
139   function getCoverage(bytes32 insurancePolicyId) public constant returns (bytes32,
          bytes32[], uint){
140     return(
141       insurancePolicies[insurancePolicyId].coverage.coverageName,
142       insurancePolicies[insurancePolicyId].coverage.coverageOffers,
143       insurancePolicies[insurancePolicyId].coverage.coveragePremium
144     );
145   }
146
147   function getInsurancePolicyBasicInfo(bytes32 insurancePolicyId) public constant
          returns(uint, uint, State, bytes32, uint){
148     return(
149       insurancePolicies[insurancePolicyId].groupNumber,
150       insurancePolicies[insurancePolicyId].sharedPool,
151       insurancePolicies[insurancePolicyId].insurancePolicyState,
152       insurancePolicies[insurancePolicyId].insurancePolicyName,
153       insurancePolicies[insurancePolicyId].acceptedStakeHoldersNumber
154     );
155   }
```

```
156
157   function getInsurancePolicyStakeHoldersInfo(bytes32 insurancePolicyId) public
          constant returns(bytes32, State, bytes32[], uint256[]){
158
159     bytes32[] memory phIdsTemp = new bytes32[](insurancePolicies[insurancePolicyId
            ].groupNumber);
160     uint256[] memory phStateTemp = new uint256[](insurancePolicies[
            insurancePolicyId].groupNumber);
161
162     for(uint i = 0; i < insurancePolicies[insurancePolicyId].groupNumber; i++){
163       phIdsTemp[i] = insurancePolicies[insurancePolicyId].policyHolders[
              insurancePolicies[insurancePolicyId].policyHoldersIds[i]].userId;
164       phStateTemp[i] = uint(insurancePolicies[insurancePolicyId].policyHolders[
              insurancePolicies[insurancePolicyId].policyHoldersIds[i]].state);
165     }
166
167     return(
168       insurancePolicies[insurancePolicyId].creatorUserId,
169       insurancePolicies[insurancePolicyId].insuranceCompany.state,
170       phIdsTemp,
171       phStateTemp
172     );
173   }
174
175   function getInsurancePolicyCreator(bytes32 insurancePolicyId) public constant
          returns(bytes32){
176     return insurancePolicies[insurancePolicyId].creatorUserId;
177   }
178
179   function getInsuranceCompanyState(bytes32 insurancePolicyId) public constant
          returns(State){
180     return (insurancePolicies[insurancePolicyId].insuranceCompany.state);
181   }
182
183   function getPolicyHolderState(bytes32 insurancePolicyId, bytes32 policyHolderId)
          public constant returns(State){
184     return (insurancePolicies[insurancePolicyId].policyHolders[policyHolderId].
            state);
185   }
186
187   function getNumberOfPoliciesByHolder(bytes32 policyHolderId) public constant
          returns (uint){
188     return policiesOfPolicyHolders[policyHolderId].length;
189   }
190
191   function getPoliciesFromPolicyHolder(bytes32 policyHolderId) public constant
          returns(bytes32[]){
192
```

```
193     bytes32[] memory policiesTemp = new bytes32[](policiesOfPolicyHolders[
            policyHolderId].length);
194
195     for(uint i = 0; i < policiesOfPolicyHolders[policyHolderId].length; i++){
196       policiesTemp[i] = policiesOfPolicyHolders[policyHolderId][i];
197     }
198
199     return policiesTemp;
200   }
201 }
```

```
1  pragma solidity ^0.4.2;
2
3  contract UsersContract{
4
5      address owner;
6
7      struct User{
8          address account;
9          bytes32 userId;
10         bytes32 firstName;
11         bytes32 lastName;
12         bytes32 email;
13         bytes32 homeAddress;
14         bytes32 cellphoneNumber;
15         bytes32 phoneNumber;
16         bytes32[] friendsIds;
17     }
18
19     uint lastBlockNumberUsed;
20     bytes32 lastBlockHashUsed;
21
22     bytes32[] usersIds;
23     mapping (bytes32 => User) users;
24
25     function UsersContract(){
26         owner = msg.sender;
27     }
28
29     function addUser(address account, bytes32 _firstName, bytes32 _lastName,
           bytes32 _email, bytes32 _homeAddress, bytes32 _cellphoneNumber, bytes32
           _phoneNumber) public payable returns (bytes32){
30
31         uint128  wager = uint128(msg.value);
32         lastBlockNumberUsed = block.number - 1 ;
33       lastBlockHashUsed = block.blockhash(lastBlockNumberUsed);
34       uint128  lastBlockHashUsed_uint = uint128(lastBlockHashUsed) + wager;
```

```
35        bytes32 _userId = sha3(lastBlockHashUsed_uint);
36
37        usersIds.push(_userId);
38        users[_userId].account = account;
39        users[_userId].userId = _userId;
40          users[_userId].firstName = _firstName;
41          users[_userId].lastName = _lastName;
42          users[_userId].email = _email;
43          users[_userId].homeAddress = _homeAddress;
44          users[_userId].cellphoneNumber = _cellphoneNumber;
45          users[_userId].phoneNumber = _phoneNumber;
46
47          return _userId;
48      }
49
50      function addFriendship(bytes32 userIdOne, bytes32 userIdTwo) public returns (
            bool){
51
52          users[userIdOne].friendsIds.push(userIdTwo);
53          users[userIdTwo].friendsIds.push(userIdOne);
54
55          return true;
56      }
57
58      function getUserId(address account) public constant returns (bytes32){
59          for(uint i = 0; i < usersIds.length; i++){
60              if(users[usersIds[i]].account == account)
61                  return users[usersIds[i]].userId;
62          }
63      }
64
65      function getUserPersonalInfo(bytes32 userId) public constant returns (bytes32,
            bytes32, bytes32, bytes32, bytes32, bytes32){
66          return (users[userId].firstName, users[userId].lastName, users[userId].
                email, users[userId].homeAddress, users[userId].cellphoneNumber, users[
                userId].phoneNumber);
67      }
68
69      function getUserAccount(bytes32 userId) public constant returns (address){
70          return users[userId].account;
71      }
72
73      function getFriendsNumber(bytes32 userId) public constant returns (uint){
74          return users[userId].friendsIds.length;
75      }
76
77      function getFriendsIds(bytes32 userId) public constant returns (bytes32[]){
78
```

```
79          bytes32[] memory friendsIdsTemp = new bytes32[](users[userId].friendsIds.
                length);
80
81          for(uint i = 0; i < users[userId].friendsIds.length; i++){
82              friendsIdsTemp[i] = users[userId].friendsIds[i];
83          }
84
85          return friendsIdsTemp;
86      }
87  }
```

```
1   app.service('Web3Service', ['$http', '$rootScope', function ($http, $rootScope) {
2
3       var web3 = new Web3(new Web3.providers.HttpProvider("http://192.168.12.7:8545")
            );
4
5       if (!web3.isConnected()) {
6           console.log("Connection failed - Node not working!");
7       } else {
8           console.log("Connected to: " + web3.currentProvider.host);
9       }
10
11      console.log(web3.version.network);
12
13      var web3Version = web3.version.api;
14      console.log("Web3 version: " + web3Version);
15
16      this.mainUser;
17      this.accounts = web3.eth.accounts;
18      web3.eth.defaultAccount = this.accounts[0];
19
20      console.log("Main account: " + this.mainAccount);
21
22      var insurancePoliciesContract = too large text
23
24      var usersContract = too large text
25
26      var usersAddress = "0x5576318acef62b6c4cd90dbfb042f4bc56f9fd30";
27      this.usersContractInstance = usersContract.at(usersAddress);
28
29      this.CreateInsurancePolicyEvent = this.insurancePoliciesContractInstance.
            CreateInsurancePolicy();
30      this.ChangePolicyHolderStateEvent = this.insurancePoliciesContractInstance.
            ChangePolicyHolderState();
31      this.ChangeInsuranceCompanyStateEvent = this.insurancePoliciesContractInstance.
            ChangeInsuranceCompanyState();
```

```
32      this.UpdateInsurancePolicyStateEvent = this.insurancePoliciesContractInstance.
            UpdateInsurancePolicyState();
33
34      this.changeMainUser = function(accountNum){
35          this.mainUser = this.usersContractInstance.getUserId(web3.eth.accounts[
                accountNum]);
36      }
37
38      //web3.personal.unlockAccount(web3.eth.defaultAccount,"1234", 0);
39  }]);
```