POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Development and Evaluation of Cryptocurrency and PSD2 Payment-Methods in an Ethereum-Based Loyalty Point System

Relatore
prof. Antonio Lioy

Riccardo Persiani

Academic year 2017-2018

To my beloved Clara
To Mum and Dad

Summary

Currently, the blockchain technology aim to bring a pragmatic change in the actual model of electronic transactions. The blockchain technologic assets can potentially revolutionise the fintech and non-fintech world in term of operations speed, cost reduction, fee avoidance, transparency, immutability, intermediaries remotion and security. Nowadays, hundreds of different blockchains are on the market; however, the most relevant one is by far Ethereum. In the last few years, Ethereum has evolved under many aspects: becoming increasingly efficient, scalable and sustainable. Ethereum was the first network to introduce smart contracts, programs where currencies and assets can be transferred into and which use their code for automatically validating conditions, determining if an asset should go to one person or get back to the sender. Despite the destructive aspect of the current economical architecture, the world juridical framework has been in favour of new payment methods and financial entities. Laws and directives on digital payments has been adopted during these years, and, as a matter of facts, PSD2 is the most groundbreaking for Europe. With its compliance, banks have the obligation to introduce API for new actors, basically, third party providers of financial services; with the purpose of guaranteeing them a full access to customer's bank accounts for performing operations. Both opportunities arising from PSD2 and Ethereum state-of-the-art technologies can be combined to develop distributed blockchain-based application, finalised to transact electronically in a fast risk-less way. The distributed application implemented in the thesis work is an example of decentralized application built on top of Ethereum for performing transactions and transfer currency and assets. This project contains three different modern payment methods: an Ethereum payment, in Ether, the native cryptocurrency of the blockchain; a PSD2 payment, a pan-european instant payment in euro; and a tokens transfer, in points, programmed in a smart contract, following the ERC20-token Ethereum standard. The three kind of payments introduced are perfectly complementary with each other in the system, proving both intra and extra blockchain payments, crypto and non-crypto currency, and using money and tokens. The solution developed involves multiple transactions with different payment methods; furthermore, it is possible to interact with through a user-friendly front end. This thesis work concerns about understanding deeply the blockchain, specifically Ethereum; if this could be a next breakthrough technology and if non-strictly-financial applications can be built on its top. The present thesis work goals are to evaluate the development potential of smart contracts, analysing the system risk and vulnerabilities; to test the advantages on developing distributed applications and to compare costs, fees and speed of these modern methods with the current ones.

Acknowledgements

I would fist like to express my sincere gratitude to my thesis advisor, prof. Antonio Lioy of the Technical University of Turin, for the very accurate and rigorous comments, without whom the thesis would not have been at the same technical and scientific quality.

I would like to give a special thanks to the eng. Luca Tomassini, chairman and CEO of Vetrya S.p.A., and his wife, dott. Katia Sagrafena, co-founder and general manager, for offering me the opportunity to research and develop with state-of-the-art technologies, and furthermore, for bringing me an improvement in my professional skills inside an highly innovative company environment.

Moreover, I would like to properly thanks dott. Mauro Patrignani, CTO, and eng. Valentino Stopponi for providing precise and detailed directives in the development of the thesis project, and for giving me important implementative guidelines concerning the architecture of the loyalty point system.

Contents

List of Figures

Chapter 1

Introduction

To determine the best course of action for finding out modern electronic-payment systems, eng. Luca Tomassini, Vetrya S.p.A CEO and Chairman, asked me to study the blockchain technology, determine the blockchain real applications, examine the software development process of a cryptocurrency, and analyse the implications for the Payment Service Directive 2.

Currently, the most common e-payment methods in Europe are PayPal, Visa/MasterCard and the bank transfer. However, these methods have well-known drawbacks: amount restrictions, risks of being hacked, fees, lacks of anonymity, and the presence of multiple intermediaries. Because of these significant disadvantages, the traditional financial system must be overcome for a more secure and trustless decentralised model. Nowadays, blockchain is the only technology for answering to all of these needs. It is a very secure technology for exchanging currency without a trusted central counterparty; it is a stand-alone economic environment, separate form the current bank system; it guaranties transaction transparency with high performance speed at low costs. In addition, blockchain allows developers to build financial and non-financial applications on top of it.

Therefore, with both the corporate supervisors, dr. Mauro Patrignani, C.T.O, and eng. Valentino Stopponi, we study to determine the best way to realize innovative payment solutions on the blockchain. As a result, we decided to experiment crypto and PSD2 payments, inside a real implementation of a fidelity points system, built on the Ethereum blockchain. Specifically, the thesis project required me to perform seven progressive tasks:

1. determine the math and cryptographic background of the blockchain technology;
2. determine the legal framework for e-payments;
3. determine the architecture for the loyalty point system;
4. develop the loyalty point system as an Ethereum application;
5. establish criteria to evaluate system risks and vulnerabilities;
6. develop Ethereum and PSD2 payment solutions integrated in the loyalty point system;
7. analyse payment and infrastructure costs and performance, compared to traditional payment systems.

I found that the blockchain technology is increasing quickly thanks to its properties; moreover, professionals from different background are finding many ways to realise blockchain solutions to improve security and save time and money. Among the Ethereum developers, a great portion is going towards the implementation of non-financial solutions for reducing service costs, improving operations speed, and making data immutable and transparent. My research on the Ethereum-based loyalty point system also found these advantages; furthermore, follows the trend of searching the perfectly suitable application for the blockchain. The thesis work found also the advantage of having a distributed application with reduction of maintenance and launch cost.

My principal findings are, on one hand, regarding Ethereum payments in term of security, transaction immediacy, near zero fees, transparency, immutability and avoidance of intermediaries. On the other hand, regarding PSD2 payments in term of using euro, contained fees, transaction speed, and limited intermediaries. These two methods are used both to give two different and complementary payment options to the client of the loyalty point system.

In the following sections, additional details are provided about the thesis objectives, the company where the thesis project is realised, and the thesis document organisation.

1.1 Thesis objectives

The present thesis work is motivated by the will to clarify the power of the blockchain technology, which is has not been tested in every facet. It is motivated by the idea of experimenting the effects of this technology in a different financial environment, which lives parallel to the convention financial environment. It is motivated by the purpose of using a state-of-the-art technology to improve to the traditional system, which lacks of risk management, immutability, transparency, and privacy. It is motivated by the desire to discover why companies and businesses are investing in blockchain to answer the ever greater demand of performance and security.

The development of the thesis project has been done in a company in order to better understand the dynamics of a prototype development in a corporate environment.

1.2 Thesis in a company

The thesis has been developed at Vetrya S.p.A. The main reason who drove me to ask for a collaboration with Vetrya was the symbol that this company represent as concerns innovation, and the expert knowledge of the development team.

A description of the company is provided in the following section.

1.2.1 Vetrya S.p.A

Vetrya [VTY.MI] is an Italian group recognised leader in the ICT market with about one hundred of employee and the total annual turnover of EUR 59 billion. The Group is listed on the Alternative Capital Market AIM Italy organised and managed by the Italian Stock Exchange.

Vetrya is present throughout the world: headquarters are in Orvieto, where is the corporate campus; furthermore, through Vetrya Inc., a company based in Palo Alto (CA), is in the U.S.A; through the company Asia Pacific Sdn. Bhd., based in Kuala Lumpur, Malaysia, Vetrya is in the Southeast Asian market; through Vetrya do Brasil is in Rio de Janeiro; and Vetrya Iberia in Madrid, for the Iberian peninsula.

Moreover, the company has won multiple prestigious awards, in particular: the prize as “best workplaces 2017” from Great Place to Work 2017, third classified in the medium companies category (with employee number between 50 and 500); the national prize for innovation; the prize “company for innovation 2017”.

The group promotes the success of its customers by introducing innovation across the value chain, with a wide range of cloud multi screen platforms for broadband telecommunications networks, media asset management, mobile entertainment, mobile commerce, value-added services, Internet TV, broadcasting, digital advertising and content production[1].

Precisely, Vetrya projects focus on five principal thematic areas:

Broadband services

Technological innovation is at the basis of Vetrya Group development and the Group has always been pursuing the aim of promoting market innovative solutions, so as to ensure its customers the needed tools to boost flexibility and efficiency. Multimedia broadband innovation centre (BIC) is the centre of excellence dedicated to the expansion of services, solutions, applications and cloud platforms, and is able to support the growing broadband services market. BIC assists its clients to fully exploit the broadband networks potential and develop services, end-to-end solutions, and innovative applications relevant to all Vetrya expertise.

Content delivery

Vetrya have the ability to plan, develop, and manage increasingly integrated innovative services, at high interactivity and with a high customisation level. Through its IP broadband distribution platforms, the company covers all of our customers needs, for any network and on any device. The cloud platforms as a service Eclexia, Visidea, Xivin and Wonda mobile hub manage the end-to-end over IP distribution and allow to convey the any type contents and on any device modalities (smartphone, tablet, connect TV, web, game console, smart watch, and set top box) together with CDN services in partnership with the major market player.

Mobile payment

Mobile devices are more and more important in the consumers' daily lives. The diffusion of alternative payment methods different from cash, is one of the main market segments that will grow significantly in the coming years. Vetrya has platforms able to provide multi-channel payment solutions (credit cards, phone credits, PtoP transfer, dedicated payment circuits) for the delivery of mobile payment services applied to goods and services (m-commerce, m-ticketing). A part from the platform dedicated to phone credit (digital pay), Vetrya provides Gateway Provider services, Payment Gateway and Service Provider.

E-Commerce

The significant development of e-commerce services has radically transformed the relationship between businesses and consumers. Companies, distributors and commercial businesses extend their sales model through the use of multi-channel strategies that give the Customers several points of contact for their product purchases. For this significant market, Vetrya has developed solutions for both B2B and B2C companies. Platforms that provide end-to-end management of the sales cycle of products and services: from repository management products, to promotions, pricing and promotional campaigns, storehouse inventory management and logistics, customer engagement in both physical store and in digital channels .

Mobile applications

Thanks to the relevant and consolidated expertise on telecommunications and mobile devices world, Vetrya ensures the development of services, applications, user experience and platforms that flank both customers Telco/Media and the OTT market. It offers proved ability of solutions development and end-to-end services, mobile application development, user experience, testing, internet of things and related products, as well as related devices and embedded softwares.

1.3 Thesis structure

The thesis is organised in the following chapters: to be provided at the end of the thesis

Chapter 2

Technical analysis of the blockchain technology

Blockchain is a technology that wants to radically change the world of financial transactions in terms of security, efficiency, costs, intermediation, privacy, transparency, and immutability. Thanks to its features, this technology create a new financial model compared to the traditional one; as a result, several professional sectors are going to be transformed about security, payment methods, performance, and costs.

One of the most important feature introduced concerns the relationship between blockchain users and traditional intermediaries, like notaries, banks, and copyrights managers; hence so, in a potential scenario where blockchain is adopted worldwide, intermediaries role should be completely redefined or even destroyed. Because of that power of avoidance of intermediaries, blockchain is sometimes associated to the concept of destructive technology.

Another important feature is about the financial world; in fact, money, stocks, bonds and derivates can be managed and exchanged through the blockchain technology. Consequently, more than the already cited advantages, also dematerialisation will be present, with a definitive overcoming of the printed paper in favour of completely digital ecosystem. As a result, the cost saving will be relevant both for the provider and the user of a blockchain-based service.

The goal of this

In the following sections are outlined the general concepts of the blockchain to illustrate the historical background, the definitions concerning the blockchain, and the technical reasons because the advantages discussed are possible. Also, a section on the current blockchain market and the different types of blockchain technologies is presented.

2.1 The general concepts of the blockchain

It is important to define the background history of the blockchain in order to understand which improvements have been done during years.

2.1.1 Background History

Blockchain referred to the technology that runs Bitcoin; in other words, it is impossible to discuss the history of blockchain technology without first starting with a discussion about Bitcoin.

Bitcoin beginnings

Blockchain technology make his public debut in 2008, when an unknown, with the pseudonym of Satoshi Nakamoto, released the whitepaper Bitcoin: A Peer to Peer Electronic Cash System.

Shortly after Nakamoto's whitepaper was released, Bitcoin was offered up to the open source community in 2009.

Blockchain separates from Bitcoin

Even today, there are many who believe Bitcoin and blockchain are one and the same, even though they are not. In contrast, those who started to realise, around 2014, that blockchain could be used for more than cryptocurrency, started to invest in it. As a result, several new blockchains started to appear on the market with the purpose of revolutionising supply chains, healthcare, insurance, transportation, voting, contract management and more. In addition, also the financial world started investment plans; currently, nearly 15% of financial institutions are using blockchain technology.

Ethereum rises: smart contracts

Vitalik Buterin, co-founder of Ethereum and Bitcoin magazine, was also an initial contributor to the Bitcoin codebase, but became frustrated around 2013 with its programming limitations and pushed for a malleable blockchain. Met with resistance from the Bitcoin community, Buterin set out to build the second public blockchain called Ethereum, launched in 2015.

The largest difference between the two is that Ethereum can record other assets such as loans or contracts, not just currency. Ethereum can be used to build "smart contracts", that can automatically process based on a set of criteria established in the Ethereum blockchain. This technology has attracted the attention of main corporations who are intrigued by the potential of the smart contract functionality to save time and money.

Transition from proof-of-work to proof-of-stake

Today, blockchain generally operates on the proof of work concept, where an expensive computer calculation or "mining" is done in order to create a block (or a new set of trustless transactions). Currently, when the sender initiate a transaction, it is bundled into a block. Then, miners which verify the transactions are legitimate within that block by solving a proof-of-work problem, a very difficult mathematical problem that takes an extraordinary amount of computing power to solve. The first miner that solves the problem gets a reward, and then the verified transaction is stored on the blockchain. Consequently, Ethereum developers are interested in changing to a new consensus system called proof of stake.

Proof of stake has the same goal as proof of work, to validate transactions and achieve consensus in the chain; but it uses an algorithm with a different process. With proof of stake, the creator of a new block "is chosen in a deterministic way, depending on its wealth, also defined as a stake." Since in a proof of stake system, there is no block reward; but, the miners, known as forgers, get the transaction fees. Proponents of this shift, including Ethereum co-founder Buterin, like proof of stake for the energy and cost savings realised to get to a distributed form of consensus.

Blockchain scaling

Since nowadays, every computer in a blockchain network processes every transaction, it can be very slow. A blockchain scaling solution would determine how many computers are necessary to validate every transaction in a way that does not compromise security.

Today, Bitcoin is just one of the several hundred applications that use blockchain technology. It has been an impressive decade of transformation for blockchain technology and it will be intriguing to see where the next decade takes us.

2.1.2 Definition

The great quantity of documentation about the blockchain technology has led to confusion regarding the definitions of blockchain; however, the major competitors of this technology agrees on the following one:

Blockchain shared distributed ledger, cryptographically immutable where transactions are permanently recorded by appending blocks.

Moreover, it is important to focus on a point that leads to several mistakes: a blockchain is not a database. It is not a database in the sense that everything cannot be stored on it; storing images or PDF files is not admitted. The ledger is distributed and everyone on the blockchain network has a copy of the ledger; so, the blockchain must have a relatively contained dimension to be scalable.

In conclusion, the thesis document will refer to the definition provided; if the term will be used with different sense, it will be expressly declared.

2.1.3 Environment: a distributed peer-to-peer network

The blockchain lives in an particular distributed environment, a peer-to-peer network. Deepening, this network is composed by individual machines, called nodes, which provide computational resources directly for all the other members of the network, without the presence of a central authority. Figuratively, it is similar to a fully democratic ecosystem, where every node is equal to every other for rights and roles; as a result, potentially, every node can choose to be a consumer and a producer of resources, without any limit.

Peer-to-peer systems must satisfy some requirements in order to work properly:

- the coordination between and among members;
- the use of a communication protocol fro sending, receiving and processing messages;
- the existence of a network; hence so, of a communication protocol;
- the proper computational capacity of the members to solve a computing problem;
- the guarantee to answer adequately to several security problems.

The advantages of using a peer-to-peer network are:

- the interaction with contractual partners uses less time and less money;
- the removal of a central governative authority, which is a single-point-of-failure.

2.1.4 Analysis of blockchain features

Blockchain is a ledger where transaction are saved inside blocks that, once validated by network nodes, are attached in temporal order to the main chain. All of the advantages that this technology aim to bring are possible because the blockchain is a ledger, is distributed and shared, and is cryptographically immutable.

Ledger

Blockchain is a digital ledger, which is analog to a physical ledger, where all the transactions are written. To summarise, the blockchain serves as a historical record of all transactions that ever occurred, from the genesis block to the latest block, hence the name blockchain. Ledger means that data are stored like a list in temporal order, like in a written register or like an Excel sheet chronologically ordered.

Distributed and shared

Distributed and shared are two properties used to describe the ledger, on one hand, as distributed in every single node. On the other hand, to describe the ledger as shared between every node; which means that, theoretically, in any time instant, the register is equal for every node of the network.

Cryptographically immutable

Cryptographically immutable property refers to the fact that every time a new block of transactions is added to the chain, the block, thanks to a cryptographic algorithm, is made immutable, guaranteeing data integrity e making the data in the ledger completely reliable.

Integrity and reliability are two sides of the same coin. On one side, integrity of the system means assuming that the system perform its functions without being compromised, free from not authorised manipulations. On the other side, the reliability of a system is gave in advance and it change depending on the outcome of the interactions with it. In conclusion, in peer-to-peer networks the user trust cannot exist without preserving the global integrity of the system.

Acquire and maintain data integrity depends from the following factors:

- the knowledge of the node numbers;
- the knowledge of the reliability of the nodes;
- the number of technical failures, components that fails or product wrong results;
- dishonest peers, which intentionally attack the system, lowering the grade of reliability.

The core problem to be solved in the blockchain is acquire and maintain data integrity in a peer-to-peer system composed by an unknown number of peer with unknown reliability. This class of problems is widely discussed in the Internet technology community; it is known with the name of Byzantine General problem.

2.1.5 Property Management

Before adding a block of transactions to the chain, the property of good or a service must be proved. The property proof involves three parameters:

- the owner identification;
- the object owned identification;
- the mapping the owner with the owned object.

The mapping is obtained by the utilisation of a ledger, which is inherent in a blockchain technology. The blockchain itself link the owner to the object owned and guarantees the data integrity.

Firstly, this implies the implementation of the following two properties:

- authentication;
- authorization.

Authentication

In the context of computer systems, authentication is a process that ensures and confirms a user's identity. Authentication is a warranty that the user is who it tells to be, confirming an attribute.

Authentication begins when a user tries to access information. First, the user must prove his access rights and identity. When logging into a computer, users commonly enter usernames and passwords for authentication purposes. This login combination, which must be assigned to each user, authenticates access. However, this type of authentication can be circumvented by hackers.

Lose a password in the blockchain Ethereum means losing everything the digital wallet; it is not possible to recover it.

Authorization

Authorization is a security mechanism used to determine user/client privileges or access levels related to system resources, including computer programs, files, services, data and application features. Authorization is normally preceded by authentication for user identity verification. During authorization, a system verifies an authenticated user's access rules and either grants or refuses resource access.

Completing a financial transaction is an operation that requires an high level of security; because of that an access policy must be defined. Authorization in blockchain systems is acquired through the use of a digital signature.

2.1.6 Validation Process

Nodes are the entities in the blockchain representing users. Nodes act likes witnesses, declaring if a transaction has been performed truly by the x sender to the y receiver with the z amount. The witness behaviour guarantees that no fraud happens in the blockchain network. As a result, greater is the number of nodes and more the anti-fraud procedure is proper and reliable.

The validation process of the transaction, which involves all the node-witnesses , is possible thanks to the blockchain algorithm/protocol. The blockchain protocol is responsible for involving all the nodes in a decision, for example declaring if a transaction is right or not. The protocol coordinates every node in the blockchain network in checking their own ledger; hence so, if the majority of the node agrees on the state of a transaction, the successful transaction is stored permanently in the blockchain. The state of a transaction can be success or rejected. The ledger that every node is checking for the transaction validation is perhaps the blockchain itself, where is the mapping between objects and owners.

For example, in a financial transaction, node must verifies that who is sending moneys, after the authentication and authorization process, owns an amount of money greater than amount that he/she is sending. Every node can check if what the sender declare is right or not, issuing a decision/verdict on the sender data. If the decision is positive, the transaction status is set to success and the transaction will be added in a block and stored in the blockchain. If the decision is negative, the transaction status is set to rejected and the transaction will not be added in a block and will not be stored in the blockchain. This example is a "use case" where the blockchain perform a function of reliability, integrity, and property management of a specific good.

2.1.7 Data hashing

A blockchain based system depends heavily on the cryptographic hash functions. These functions, if properly chosen, are:

- applicable to every kind of data;
- deterministic;

- fast;
- pseudo-random;
- one-way;
- collision resistent.

The goal of these function is identify transaction, comparing the result of the hash function, without comparing data directly. In fact, comparing data directly is a more long and computationally heavy process. The process of verifying the trust of data is a core element; if data are 100% trusted, user are protected from every kind of frauds or attacks.

2.1.8 Block

A block is a group of transactions in chronological order, or the best chronological order that the miner nodes can agree and organise the transactions in. Every block has, as its data, the hash of the previous block. Each block is made of a "block header" and a "block body". Every blockchain has its own block structure, but the main features of all of them are the same.

Regarding the blocks topic, the block header and body structure must be discussed and, furthermore, there is an important difference between the process of block creation and block protection.

Block header structure

Block fields are illustrated in the figure 2.1:

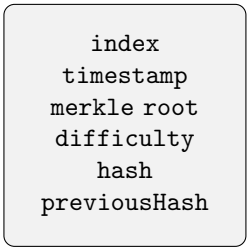

Figure 2.1: The information in a block header.

- `index`: the number of a block in the chain, `index=0` value is reserved for the root of the blockchain;
- `timestamp`: the time instant when the block has been added to the blockchain;
- `merkle root`: the hash of the root of the merkle tree of all the block's transactions;
- `difficulty`: the difficulty level for miner of the block;
- `previousHash`: the pointer to the previous block, which is the hash function result of the previous block; this is illustrated in the figure 2.2;
- `hash`: the result of cryptographic hash function on all the previous described informations.

Block body structure

The body of the block are transactions data, saved inside a block with a merkle tree structure. A merkle tree is a tree constructed by hashing paired data (the leaves), then pairing and hashing the results until a single hash remains, the merkle root. The merkle root hash value is saved in the block header. In Bitcoin, the leaves are almost always transactions from a single block.

The merkle tree verifies a group of transactions, as is possible to see in the figure 2.3.

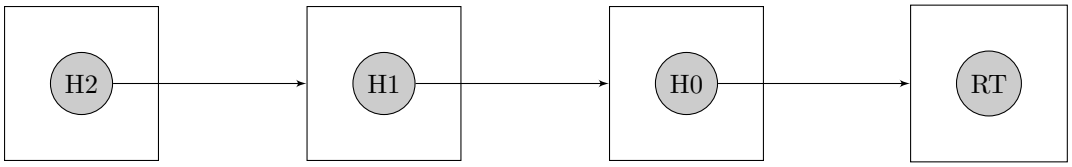

Figure 2.2: Chain structure with block pointers to the previous block hash.

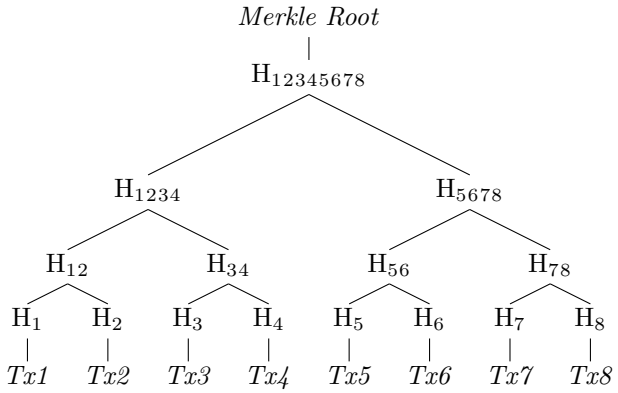

Figure 2.3: A block body: a merkle tree structure.

Block addition process

The process of adding a block to the blockchain is the following [2]:

1. new transaction and/or new blocks are forwarded to the nodes of the blockchain network in "gossip-fashion";
2. every node saves transaction and blocks received in inbox;
3. every node process blocks following grade of priority;
4. every node process new transactions, declaring them valid or invalid for authorization, and formal and semantic reliability;
5. every node collects only valid transactions in a merkle tree and start finding a new block resolving the hash puzzle;
6. when a node solves an hash puzzle, it sends the new block to the other nodes;
7. every node process new blocks verifying the hash puzzle solution and verifying the transaction contained in the block;
8. every node takes the valid blocks and add them to the local blockchain;
9. if a new block is invalid, the block is deleted and nodes would continue to process transaction or finishing hash puzzle of a new block;
10. if a new block is valid, the node removes transactions contained in the new block from the inbox and starts processing transactions and creating a new block;
11. if the block added to the local blockchain is declared invalid or useless later, that block is going to be removed from the blockchain and the transactions would be moved again to the inbox and processed again later;
12. the node that emits an accepted block will receive as reward the fee for all the transactions contained in the block;
13. if a block, previously added to the blockchain, is removed, the reward given in the past is retreat from the node.

Block protection

After a block is added to the blockchain, it cannot be changed. For example, an attacker can change data of the transaction Tx'$_3$ and declaring a different amount of money sent in his/her wallet regarding a true transaction Tx$_3$. Anyway this behaviour implies the generation of a different hash, as is possible to see in the figure 2.4. As a result, every superior hash back to the root should be modified in order to create a valid block. This require too much work to be performed by a single attacker.

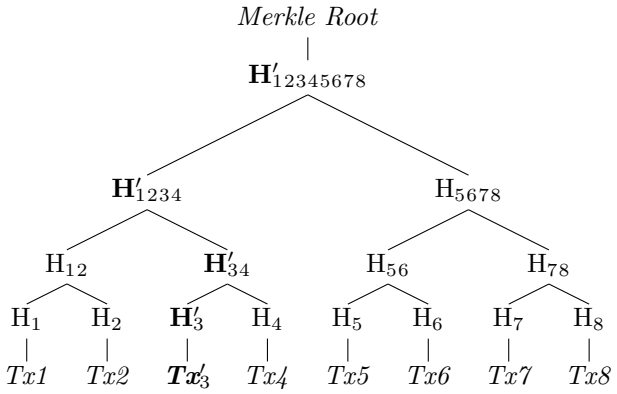

Figure 2.4: Result on a block merkle tree of a transaction attack.

To validate the block the attacker must modify not just the hash of the new transaction but also every other upper hash; so that other nodes, checking in parallel the validity of a block, did not notice the transaction modification. The computational power requirement for a node to perform such an operation is unreasonable, compared with the power of all the other members in the network, that work in parallel.

2.1.9 Chain conflicts

In addition to the block protection mechanism, is important to take into account the problem of chain conflicts. When different chains are in conflict, the main rules is that the longest chain is always accepted has the correct one, see the figure 2.5.

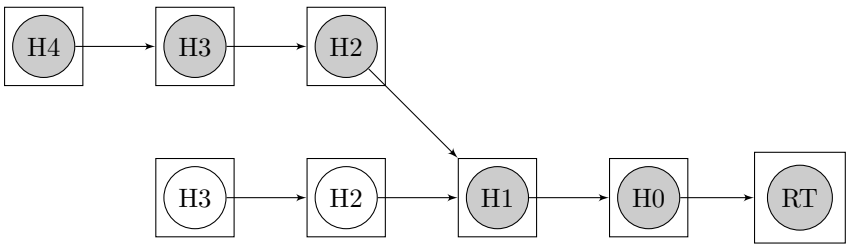

Figure 2.5: Conflict between two chains.

If an attacker success in the validation of a block after a long amount of time, the presence of a longest blockchain would make the attacker work completely useless. Hence so, the attacker not only would not succeed in the attack but it also would lost a significant amount of time and energy, which means losing money.

2.1.10 Hash puzzle

Other than data comparing, data pointing and secure data saving, cryptographic hash functions are used to create "puzzles". These puzzle must be resolved by some special nodes in order to avoid frauds.

Hash puzzles has the following characteristics:

- need time to be resolved;
- have a difficulty level that can be set;
- check their solutions is simple.

Specifically, an hash puzzle is a challenge with a certain grade of difficulty. The grade of difficulty is expressed as an integer, which represents the number of zero that must be present in the hash generated. As is said before, an hash is a one-way function, so the same data in input are going to generate the same output hash value; as a result; however, in order to obtain as output of an hash function, a string with x number of 0, the hash function have to be applied not only on the given data, but also on a `nonce`. The `nonce` is an integer which is changed until the hash puzzle solution is founded. Find a solution uses a lot of computational resources, electricity and time.

This process of solving the hash puzzle is called "mining" and will be explained in the following section.

2.1.11 Mining

Because of the great amount of compute power, energy and time to involve in "mining", not every node of the network tries to solve the hash puzzle. Nodes who try to solve the hash puzzle are called "miners". A miner of the blockchain network try to find a solution for the hash puzzle before other miners. The first miner which can solve the puzzle is the winner; hence so, it is rewarded with `gas`, a synonym of moneys for its work.

Currently, the mining process is used in pretty much every blockchain network; however, just the main principle of mining have been explained and every blockchain has its own mining implementation. For example, the Bitcoin blockchain gives an elegant mining implementation; in fact, miners uses on average one-hundred-billion-billion attempts. This huge amount of attempts is justified by the fact that miners can work in parallel in order to solve the hash puzzle.

Rewards are an instrument used for accelerating and improving the transaction validation process and the block creation process. However, the blockchain need also an instrument to punish bad peer behaviour and to punish who attack the system security. Rewards could be retread if a block accepted in the past is declared invalid or useless; or the absence of a reward can be used if a node is mining a block already added.

Going deeper in mining the concept of proof-of-work and proof-of-stake appears.

Proof of work

Proof of work is basically the mining process described before. Proof of work is a protocol that has the main goal of deterring cyber-attacks such as a distributed denial-of-service attack (DDoS) which has the purpose of exhausting the resources of a computer system by sending multiple fake requests.

Going deeper, proof of work is a requirement to define an expensive computer calculation, also called mining, that needs to be performed in order to create a new group of trustless transactions (the so-called block) on a distributed ledger called blockchain.

Mining serves as two purposes:

- to verify the legitimacy of a transaction, or avoiding the so-called double-spending;
- to create new digital currencies by rewarding miners for performing the previous task.

All the network miners compete to be the first to find a solution for the mathematical problem that concerns the candidate block; a problem that cannot be solved in other ways than through brute force, so that essentially requires a huge number of attempts. When a miner finally finds the right solution, he/she announces it to the whole network at the same time, receiving a cryptocurrency prize (the reward) provided by the protocol.

From a technical point of view, mining process is an operation of inverse hashing: it determines a number (nonce), so the cryptographic hash algorithm of block data results in less than a given threshold.

This threshold, called difficulty, is what determines the competitive nature of mining: more computing power is added to the network, the higher this parameter increases, increasing also the average number of calculations needed to create a new block. This method also increases the cost of the block creation, pushing miners to improve the efficiency of their mining systems to maintain a positive economic balance. This parameter update should occur approximately every 14 days, and a new block is generated every 10 minutes.

Proof of work is not only used by the bitcoin blockchain but also by ethereum and many other blockchains. Some functions of the proof of work system are different because created specifically for each blockchain. The important thing you need to understand is that now Ethereum developers want to turn the tables, using a new consensus system called proof of stake.

Proof of stake

Proof of stake is a different way to validate transactions based and achieve the distributed consensus. It is still an algorithm, and the purpose is the same of the proof of work, but the process to reach the goal is quite different. Proof of stake first idea was suggested on the bitcointalk forum back in 2011, but the first digital currency to use this method was Peercoin in 2012, together with ShadowCash, Nxt, BlackCoin, NuShares/NuBits, Qora and Nav Coin.

Unlike the proof-of-Work, where the algorithm rewards miners who solve mathematical problems with the goal of validating transactions and creating new blocks, with the proof of stake, the creator of a new block is chosen in a deterministic way, depending on its wealth, also defined as stake.

There are no block rewards. Also, all the digital currencies are previously created in the beginning, and their number never changes. This means that in the PoS system there is no block reward, so, the miners take the transaction fees. This is why, in fact, in this PoS system miners are called forgers, instead.

2.2 Types of blockchain

Currently, on the market there are multiple blockchains and each of these have its particular characteristics. Blockchain technologies can be categorised following several criteria, the main are: [3]:

- network type: public, private, and consortium or permissionless and permissioned;
- community size;
- price: open source o payable;
- programming language supported.

Before discussing the criteria presented, it is important to present which blockchains are actually used nowadays.

2.2.1 Market overview

In the following sections are discussed the main blockchain technologies on the market.

In the table 2.1 are listed the main technologies that will be discussed later:

name	*network type*	*community*	*pricing*	*language*
Bitcoin	public, permissionless	highly active	Bitocoin core open	C++
Ethereum	public or private	highly active	open source	Solidity
Fabric	private, permissioned	highly active	open source	Phyton
Corda	private, permissioned	highly active	open source	Kotlin, Java
IOTA	public, permissioned	medium active	open source	C, Phyton, JS
MultiChain	private, permissioned	medium active	open source	C#, JS, Ruby
HydraChain	private, permissioned	low active	open source	Solidity, Phyton
Chain	public	medium active	enterprise licensing	Java, Ruby, Node JS

Table 2.1: More used blockchain on the market.

Bitcoin

The Bitcoin blockchain appeared on the market in 2008, created by the unknown Satoshi Nakamoto. This blockchain is the first public permissionless blockchain ever created. The Bitcoin blockchain is open source, the design is completely public; in addition, the infrastructure and nodes are not controlled by a centralised authority [4]. Fast peer-to-peer financial transaction, global payments, low fees are the main feature offered. Bitcoin core is released under the term of MIT license [5].

Ethereum

Ethereum was born in the august 2015 when the three founders, Vitalik Buterin, Gavin Wood and Jeffrey Wilcke, decided to create an updated blockchain model introducing smart contracts. Ethereum is a blockchain platform smart-contract-based open, decentralised, with no central authority [6]. Ethereum is used for the development of distributed application through the Solidity programming language. Ethereum is refereed as a blockchain 2.0 technology in comparison with Bitcoin which is the blockchain 1.0. The main blockchain is open but is also possible to create private blockchains.

Hyperledger Fabric

IBM's Fabric is a blockchain open-source utilised for the transversal development of the blockchain in the corporate environment. This blockchain was born in collaboration with the leader business in financial, IoT, manufacturing, technology field; moreover, the project is coordinated by the Linux Foundation. The goal of the linux foundation is to create a standardised, enterprise-based, open-distributed-ledger framework [7].

Corda

Corda is a distributed ledger platform developed by R3 in order to register and process financial transactions. Corda. is heavily inspired to blockchain systems without the design qualities that make traditional blockchain inappropriate for several financial scenarios [8]. Corda is a decentralised database with minimal trust between nodes would allow for the creation of a global ledger. Such a ledger would have many useful applications in finance, trade, supply chain tracking and more.

IOTA

IOTA is a revolutionary new, next generation public distributed ledger that utilises a novel invention, called a Tangle, at its core. The Tangle is a new data structure based on a Directed Acyclic Graph. As such it has no Blocks, no Chain and also no Miners. Because of this radical new architecture, things in IOTA work quite differently compared to other Blockchains. The major difference that is worth mentioning (apart from the DAG vs. Blockchain) is how IOTA achieves consensus and how transactions are made []. IOTA is also a cryptocurrency which focuses on furnishing communication and a secure payment way for machines in IoT; in these field, it is uses for the execution of micro-transactions without fees and the creation of secure communication channels between devices [9]. It should be noted that IOTA is currently still in a beta version.

MultiChain

MultiChain is an open platform for the development of blockchain application private/permissioned in-corporate or intra-corporate. It is presented as a upgraded version of the Bitcoin blockchain; in addition, it gives privacy and control on the peer-to-peer network. MultiChain offers to its users a fast and simple mechanism of creating a new blockchain or connecting to an existing blockchain, controlling every aspect of the blockchain, and supporting some features to improve the whole security, like external private key, multi-signature, cold node and "admin by consensus" [10].

OpenChain Project

OpenChain is an open source distributed ledger technology manage digital assets of the companies in a secure, reliable, and scalable way. However, OpenChain is not a real blockchain in the sense that it does not uses blocks to save operations; actually, transactions are directly stored and concatenated one with the other, without using blocks of transactions. As a result, this mechanism allows to reduce the delay for collecting transaction in a block increasing performance of the blockchain technology. Transaction are added to the main chain when they are completed, and as a result it offers transaction verification in real time. OpenChain defines itself a "transaction chain" rather than a blockchain [11].

Chain

Chain is a blockchain platform finalised to industrial and financial application. Chain is based on "Chain Core", an enterprise software designed for operate and participate in blockchain permissioned network. Every network has its own shared ledger, secured by cryptographic algorithms; as a matter of fact, users of this product can transfer digital goods without using intermediaries [12]. Chain main features are: the "1-second consensus algorithm", which allows third party to read encrypted information on their needs; and smart contracts, saved in readable form. The "Chain Core developer edition" has a license under the terms of GNU Affero General Public License Version 3 (AGPL). The Chain Java Software development kit SDK has a license under Apache License versione 2.0 citehandblock.

HydraChain

HydraChain is permissioned distributed ledger based on the extension of Ethereum, its main applications are setups for private and consortium chains. The HydraChain whitepaper says that this blockchain is perfectly suitable with the Ethereum API, allowing the development and the deploy of smart contracts and distributed application.However, the main difference is in the consensus protocol which does not depend on the proof of work but uses a restricted group of validators for the transaction validation process. The core software is an open source and available under the MIT license [13].

2.2.2 Public, consortium and private blockchains

Blockchain can be categorised regarding the access rights to the network, the level of decentralization, the censorship resistance; these parameters divides the blockchain in:

- public;
- consortium;
- private (or totally private).

Public blockchain

Blockchain with open access, where every entity can perform a transaction and participate in the consensus process, in a network where there is no central authority and every type of censorship is rejected. Public blockchain are considered "fully decentralized" [14] and their efficiency relies on the combination of resources used and on the cryptographic verification mechanisms, like proof-of-work or proof-of-stack. The most diffused example of public blockchain is the Bitcoin blockchain, an open source, decentralized platform where network nodes are build to be free and censor resistant; this blockchain is used as a register for every transaction; these transaction once stored are immutable in time and data.

Consortium blockchain

Consortium blockchain indicated a blockchain where the consensus process do not involve all the nodes of the network but only specific nodes elected by a consortium of companies. The right to read is given to the not elected nodes of the network by a central authority or by a group of particular authorities. Consortium blockchains are similar to the private blockchain but they work under the control of a group of several entities rather than a single entity. These blockchains are a sort of hybrid between the public and the provate, sometimes documentation refers to them as "partially centralised". Vitalik Buterin, founder and father of Ethereum, clarifies the concept of consortium blockchain with the following declaration: "So far there has been little emphasis on the distinction between consortium blockchains and fully private blockchains, although it is important: the former provides a hybrid between the "low-trust" provided by public blockchains and the "single highly-trusted entity" model of private blockchains, whereas the latter can be more accurately described as a traditional centralized system with a degree of cryptographic auditability attached".

Private blockchain

Private blockchain are fully centralized blockchains where a single entity have the right of writing and work as orchestrator for the rights of read; as a result, this blockchain fits only with a more traditional model of business and governance. Only the company who owns the blockchain can verify the correctness of every operations; so, this technology is very efficient and transactions are fast respect to the other blockchain types. In addition, the adoption of a private blockchain guarantees a better privacy than a public blockchain because the transaction history can be made not accessible to every node of the network

2.2.3 Advantages of public and private blockchains

In the following sections the advantages of public and private blockchains are discussed. Consortium blockchain are incorporated in the private blockchain section.

Public blockchain advantages

In primis, the disruption of a central authority gives to the nodes of a public blockchain network a state-of-the-art level of trustness. On a market side, this feature of avoiding central authority is what is bringing users and investors in the blockchain world. On one hand for the cost and fee reduction; on the other hand, even more important, for the search for high security measures and "censorship resistance". In addition, public blockchains are fully open to every user, so can be are used freely by everyone in the world; in fact, is possible to have on these kind of blockchains users or companies that are using different services, but still on top of the same blockchain. A similar situation is impossible to realise, in the same way, in other blockchain types.

Private blockchain advantages

Private blockchain have several advantages respect to the public blockchains, because transaction are no more immutable in this kind of network; as a consequence, the entity or the consortium who owns the blockchain can easily perform restore operations or error resolution. Otherwise, sometimes, restore can also be performed on a public blockchain, giving a sort of backdoor key in a smart contract; but this will be a nonsense because in this case is better to use a private blockchain.

Another possible advantage is in having only some nodes validating transactions; perhaps, these nodes are known and controlled, avoiding some particular attacks (like the 51% attack). Malicious entities, if controlling the majority of miners, could stop transaction or invert them; because of that, this attack is considered one of the most dangerous in a public network. It is still true, that if the company controlling the private blockchain has a malicious behaviour it would be impossible to notice for the common user of a service on top of this blockchain.

An additional advantage is that transaction cost less and energy waste is reduced significantly; indeed, just few are the "trusted" nodes for validating a transaction and thousand of miners are no more needed. It is known that mining a big problem of the public blockchain in term of energy waste. Energy waste of the most famous public blockchains is considered unsustainable from an environmental point of view; int fact, the energy consumption is about ** for Bitcoin and ** for Ethereum.

Moreover, transaction are faster respect to a public network; as a matter of fact, despite improvements in the public blockchain are leading to more performant operations, obtaining the same parameters of a private blockchain is impossible right now.

At last, permission of read are limited to just the nodes elected by the central authority; so, a node cannot read transactions performed by another node.

2.2.4 Permissionless and permissioned blockchains

About the design part, R. Sams declared that is important to consider 3 negative behaviours which compromise a blockchain effectiveness

- "Sins of commission", the forgery of a transaction;
- "Sins of omission", the censorship a transaction, which refers to the possibility to make a double spend attack, not registering one or more transaction in the blockchain;
- "Sins of deletion", the reversal of transactions, after adding it to the ledger [15].

Furthermore, Sams claims that is not possible to avoid simultaneously all these three behaviours, but, like a CAP theorem, only the couple forgery-censorship or forgery-reversal can be avoided. The "sins of commission", that refers to forgery, cannot be avoided and it simple implementation is easy using some crypto algorithms.

On one side, permissionless blockchain like Bitcoin optimise the elimination of censorship rather than reversal; on the other side, permissioned blockchain can perform actions towards an attacker who is committing an omission or commission sin.

So, the blockchain types can also be divided in permissionless blockchains and permissioned blockchains; it must be noticed that there is a strong analogy between permissionless-public and permissioned-private.

In order to introduce these categories is important in this context to define a permit:

Permit refers to the authorization for the transactions verification and for subscribing to the network.

Permissionless blockchain

Permissionless are blockchains that do not need to authorise nodes for the network subscription or for allowing them in network validation.

Every node can perform a verification process, so each single node con participate in the main mechanism of the blockchain; besides, in this kind of network the peer participation in the verification in encouraged. The main rules of these network is, bigger is the number of nodes and less is the probability to receive a 51% attack.

In permissionless ecosystem the consensus is reached with the proof-of.work, with the resolution of puzzle, towards a display of usage of computational resources for the total system.

Bitcoin was defined before as a public blockchain in the section 2.2.2, but is also permissionless. The same applies for Ethereum which gives the access to smart contracts to a big community was designed as public and permissionless [6].

Permissioned blockchain

Permissioned blockchains are composed by a group of trusted nodes used for the verification process; these nodes can be authorised to perform some specific tasks exclusively by a single or multiple central authority. This central authority hash the right to distribute permits to the nodes.

Permissioned blockchains are different from the permissionless one for their dimension and for the number of total operations perform in the network. These restricted blockchains are purpose build for a business service and they are created to main a certain grade of compatibility with traditional applications [16].

2.2.5 Advantages of permissionless and permissioned blockchains

In the following section the man advantages of permissionless and permissioned blockchain are discussed.

Permissionless blockchain advantages

Advantages of permissionless blockchains can be associated with ones of the public blockchains. For instance the proof-of-work mechanism permits to the network to be secured from malicious attacks to the system reputation, like the Sybil attack. This kind of attack is caused from what is knows as “cheap pseudonym problem”, a threat to one of the core element of the blockchain: reliability.

Another advantage is related to the open access system of encouraging the peer participation in validating transaction removing every type of constraint.

Permissioned blockchain advantages

Firstly, the main advantage of the permissioned blockchain is in scaling capacity; currently, scalability is one of the searched feature in the blockchain world and it must be considered in every blockchain-based service, especially when there are thousands of nodes and users. In a permissionless blockchain, data are saved in every computer of the network and nodes can very every transaction. So, it is obvious that every time the number of transactions grows up, users who can mine became less, bringing centralization in favour of the remained pools of miners. This situation appears in the Bitcoin network where actually there is a sort of chinese monopoly regarding mining.

Moreover, the advantage of the permissioned blockchain is that if a small amount of nodes can validate they could scale easily with the increment of transactions. However, because of the expected small number of this kind of network, it is easy for group of nodes to collaborate and alterate rules or invert transactions. Anyway, in general, is easy in permissionless blockchain to reject a transaction in the sense that these blockchains are not censorship resistance like the permissionless one.

Respect to the permissionless, furthermore, permissioned blockchains do not need a mechanism for surviving to the Sybil attacks, because the governance process who selects validator makes the validation process restricted just to trusted selected nodes and that cannot be used by pseudonyms.

An example of blockchain permissioned is Fabric.

Chapter 3

Regulatory framework for E-payments in Europe and Italy

The description of the current legislative framework for e-payments follows in this chapter. Concerning the thesis project, an analysis of the European and Italian digital payments regulation was needed, in order to understand normative strengths and weakness of the current payment methods, open banking and blockchain.

3.1 Abbreviations

The following table supply a list of the legislative and computer science abbreviations used in this chapter e in the text in general:

abbreviation	*description*
EU	European Union
EC	European Council
PSD2	Payments Systems Directive 2
TPP	Third Party Provider
PISP	Payment Initiation Service Provider
AISP	Account Information Service Provider
CISP	Card Issuer Service Provider
CBPII	Card Based Payment Instrument Issuers
PSP	Payment Service Provider
ASPSP	Account Servicing Payment Service Providers
BS	Bank Service
EBA	European Banking Authority
RTS	Regulatory Technical Standards
OBP	Open Bank Project
CA	Certification Authority
SCA	Strong Customer Authentication
PKI	Public Key Infrastructure
OTP	One-Time Password
CA	Certification Authority

Table 3.1: Legal abbreviations.

3.2 Directive 2015/2366/EU of the European Parliament

The `Directive 2015/2366/EU` of the European Parliament and of the Council of 25 November 2015 on payment services in the internal market, amending `Directive 2002/65/EC`, `Directive 2009/110/EC`, and `Directive 2013/36/EU` and `Regulation (EU) No 1093/2010`, and repealing `Directive 2007/64/EC` is known in the open banking community as PSD2.

The directive seeks to improve the existing EU rules for electronic payments, taking into account emerging, innovative and complex payment services, such as internet, mobile payments and blockchain. Indeed, recital 10 says [17]:

> This Directive introduces a neutral definition of acquiring of payment transactions in order to capture not only the traditional acquiring models structured around the use of payment cards, but also different business models, including those where more than one acquirer is involved. This should ensure that merchants receive the same protection, regardless of the payment instrument used, where the activity is the same as the acquiring of card transactions.

PSD2 also aims to pay attention to the future development of new e-payment methods, setting up the conditions for an open banking environment.

Open banking is a financial ecosystem, technologically neutral, that allows the development of new types of payment services, with equal operating conditions, for both existing and new payment service providers, as shown in figure 3.1.

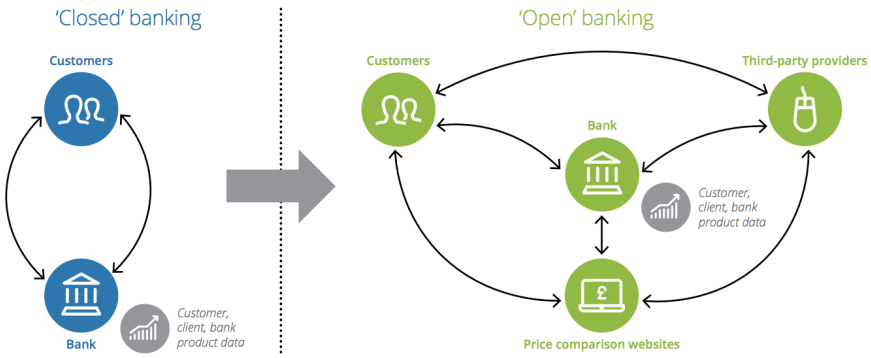

Figure 3.1: Transition from a current banking to an open banking model (source: Deloitte p.9).

As a consequence, the safety is fundamental, and the directive concentrate on several cybersecurity aspects. The recital 7 says [17]:

> In recent years, the security risks relating to electronic payments have increased. This is due to the growing technical complexity of electronic payments, the continuously growing volumes of electronic payments worldwide and emerging types of payment services. Safe and secure payment services constitute a vital condition for a well functioning payment services market. Users of payment services should therefore be adequately protected against such risks. Payment services are essential for the functioning of vital economic and social activities.

Objectives which PSD2 wants to achieve are clear: the release of the legal framework for the further development of a better integrated internal market for e-payments within the EU; the increase of consumers choice and costs reduction through the use of different payment services provided by different operators; the lowering of the bank monopoly, with by the introduction of third parties. The directive puts in place comprehensive rules for payment services, with the goal

of making international payments (within the EU) as easy, efficient and secure as payments within a single country [18].

In the following sections is going to be discussed the market context, technologies involved by PSD2, and the key points which the directive introduces.

3.2.1 Market Context

PSD2 arrives in a period of time where the banking system is at a radical change. Currently, banks are stressed from the market and their emerging operators who have introduced several state-of-the-art products; in this environment, banks have to decide whether to compete in order to maintain a direct relationship with their customers or to limit their role to that of just a provider of banking services.

Actually, the core problem for banks is generating revenue, not only because of low margins in a "lower-for-longer" interest rate environment [19], but also for the outcome on the market of technologies like blockchain. Analysts estimate that bank profits will continue to lag behind cost of equity until 2019:

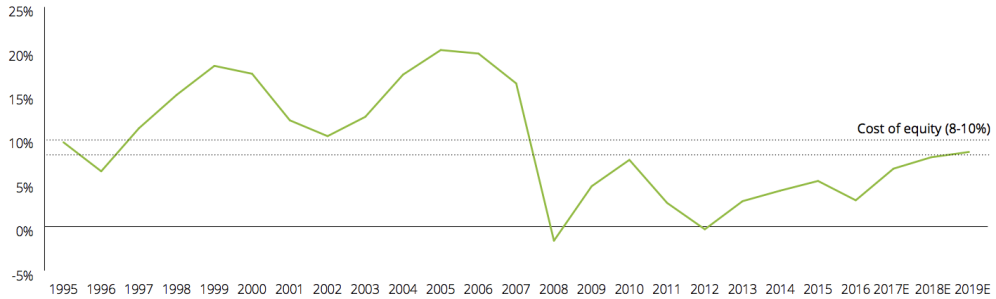

Figure 3.2: Chronological estimation of bank profits (source: Deloitte p.4).

3.2.2 Technologies involved by the directive

An incredible number of technologies is going to be involved by the directive. On one hand, e-payments represent obviously the main component of the PSD2; but, one the other hand, is opportune to notice that fields such as insurance, deposits, capital raising, investments, and market provisioning are part of the destructive revolution of the traditional system. The figure 3.3 shows the technologies for every sector of involvement.

3.2.3 PSD2 Key points

PSD2 seeks to focus on:

- the introduction of new institutions, which contribute to a more integrated and efficient European payments market;
- strict security requirements for electronic payments and the protection of consumers' financial data, guaranteeing safe authentication and reducing the risk of fraud;
- the release of an API layer from banks, which de facto starts the open banking era;
- the base for the blockchain involvement.

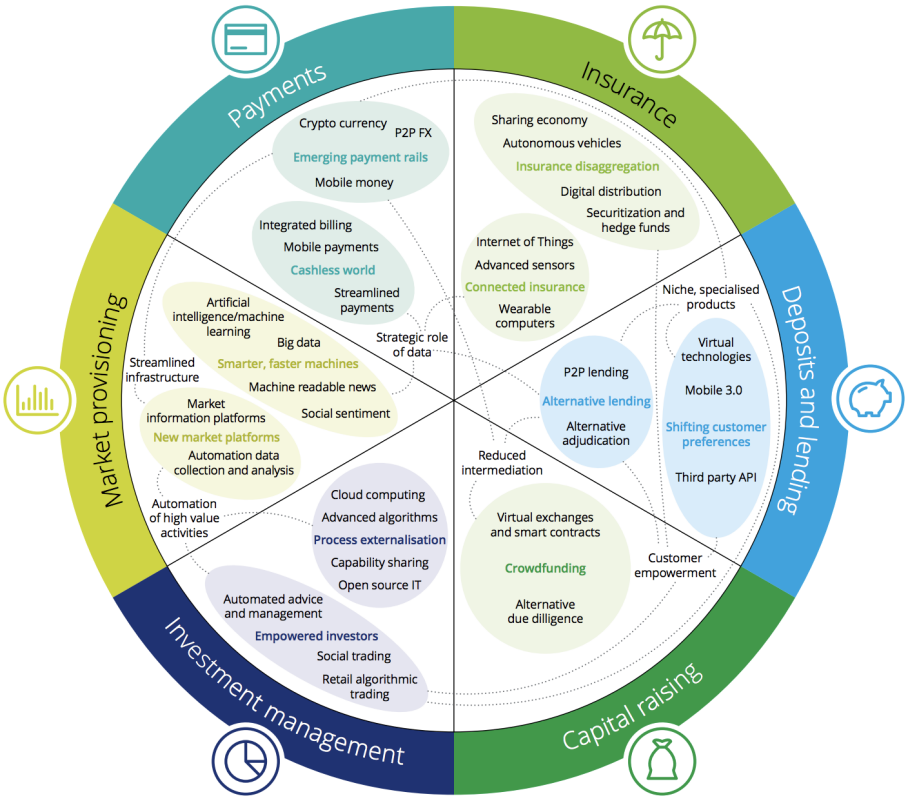

Figure 3.3: Technologies involved in respect to a fintech field (source: World Economic Forum p.12).

3.2.4 New third party institutions

One of the main feature of the directive is the recognition of new institutions. The introduction of new operators and providers gives customer the possibility to choice a better service with less cost; in relationship with the current situation where banks are central authorities.

Current situation

Nowadays, the only way for customers to access their bank accounts to make payment is through products and channels provided by their bank, as shown in the figure 3.4.

Figure 3.4: Current banking architecture (source: Deloitte p.4).

The majority of these new institutions relies to the category of TPPs which are divided in: PISP and AISP [20]. Moreover, also CISP and ASPSP represent institutions, even if they are not in the TPPs category.

PISP

Payment Initiation Service Providers are third parties provider that can start a payment operation online (payment initiation) from a user bank [21], with the right authorization; furthermore, PISP have the faculty to offer new payment methods, as alternative to traditional ones. PISP applications can initiate payments directly from customer payment accounts, so long as they have the customer's consent, as is possible to see in the figure 3.5.

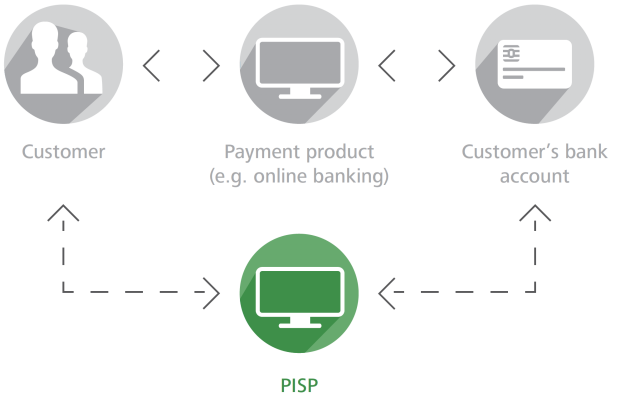

Figure 3.5: Open banking architecture with PISP (source: Deloitte p.5).

AISP

Account Information Service Providers are third party providers that can access informations of the users bank account informations [21]. Regulated third-party AISPs can access customer data (with the customer's consent) to provide an overview of a customer's payment accounts with different banks in one place (e.g. a mobile app). The functions that AISP can perform shows how the directive is not going to discriminate TPP respect to banks. In the figure 3.6 is outlined an AISP model.

Figure 3.6: Open banking architecture with AISP (source: Deloitte p.5).

CISP

PSD2 refers to PSPs issuing card-based payment instruments but does not separately define these Card-Based Payment Instrument Issuers (CISPs). Recital 67 provides some context [17]:

> The issuing of a card-based payment instrument by a payment service provider whether a credit institution or a payment institution, other that the servicing the account of the customer, would provide increased competition in the market and thus more choice and a better offer for consumers.

Consequently, referring also to the annex I, the term Card Based Payment Instrument Issuer therefore means PSPs licensed either as credit institutions or as payment institutions offering services for the execution of payment transactions, including transfers of funds on a payment account with the user's PSP or with another PSP, the execution of payment transactions where the funds are covered by a credit line [...], or the issuing of payment instruments and/or acquiring of payment transactions [22].

There should be a clear distinction between the provisions related to PISPs and AISPs and CBPII as there are different communication and process requirements. CISP are going, for instance, to allow funds checking and other business functions for commercial operators which wants to became payment institutions [23]. An example of check funding performed by a CISP is shown in the figure 3.7.

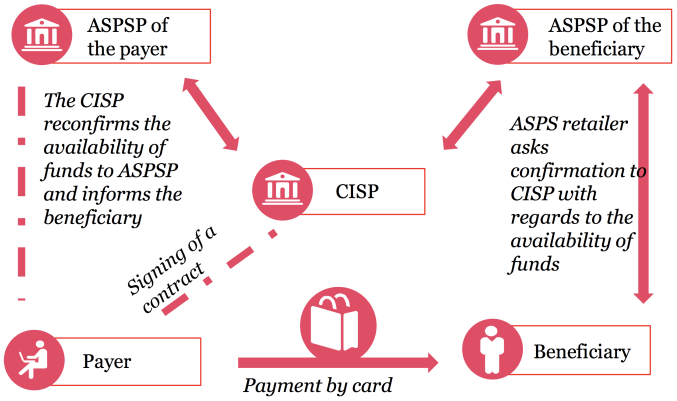

Figure 3.7: Open banking architecture with CISP (source: PWC p.6).

ASPSP

Account Servicing Payment Service Providers are banks and financial institutions; are the provider of user accounts which have to release API to the PISP and AISP, in order to interact with user data [21]. ASPSP can introduce fee for the PISP transactions, but these fees have not to be higher than the fees of the ASPSP products.

3.2.5 Security for e-payments

The growing complexity in payments methods and the need to ensure greater security to payers are so fundamentals that PSD2 requires secure standards for dialogue between TPP and BS, and strengthening of the authentication processes.

Secure standards for dialogue between TPP and BS

Standards for the secure dialogue between TPP and BS payment providers, authorised by final customers, must enable access to online accounts through interfaces that are easy to integrate. The principle of the new regulatory framework represents both a market opportunity and a matter of great concern for more traditional banks, which risk disintermediation from their customers.

The most significant impact on a technical level is the request by the Directive to facilitate operations that access accounts from external providers, in order to collect information or process a payment. Contrasts among the potentials arising from the development of a common language between banks and third parties involved in payment operations are evident, and there is risk of defining overly rigid standards that create barriers for future innovation. EBA was assigned the task of creating standard communication that allows innovation which will be published in (RTS). This will allow for dialog between parties with the uniform and certified criteria.

In this regard, the final version of RTS in the field of "Strong Customer Authentication and Secure Communication" was released by 13 January 2017, while a consultation paper was published in August 2016 [24]. Whatever technology is adopted to define the standard conversation between the parties, the choice that every bank will take is about the project approach. It will be necessary to decide whether to wait for the regulatory and market changes (reactive approach) or anticipate them, interpreting the directive as an opportunity to develop the business (proactive approach) [25].

Strengthening of the authentication processes

The other main innovation of PSD2 is the introduction of a needed implementation of a strong customer authentication in all banks. As a matter of fact, strengthening of the authentication processes, the use of strict safety standards, in compliance with the ECB provisions, becomes mandatory and it requires identity verification through two or more authentication tools, strengthened by the use of dynamic links which certify the uniqueness of the transaction.

Deepening, the user identity must be verified by two or more authentication tools classified as:

- knowledge, something that only the user knows (such as a PIN);
- possession, something that only the user has (such as Token);
- inherence, something that only the user is (such as a digital fingerprint).

EBA, in order to limit the risk of compromising the authentication requirements, is focusing on the issue of the interdependence of the individual elements to ensure that the violation of one authentication does not affect the others. The directive also anticipates that the payment operations with increased security thanks to "dynamic linking" mechanisms will contain at least an amount and a specific beneficiary. In fact, the goal is to ensure that authentication for a remote transaction is not used for any other purpose than the one originally foreseen by the payer.

3.2.6 API for the Open Banking

The legislation does not show the technology that the Banks must adopt to dialogue with third parties, delegating the task to EBA. On one side, the latter intends to orientate their indications in order to preserve innovation and cooperation, avoiding the introduction of unnecessary rigidity. On the other side, the definition of a standard, addressed either by the regulator or the market, will have to be introduced in order to not disperse unnecessary energy of the industry that attempts to reconcile the different interfaces implemented autonomously by the different Institutes. Even in the presence of these areas of uncertainty it is the common view, among financial institutions and fintechs active in the sector, that the API may be a desirable technology to adopt [25].

APIs represent a specific architectural approach that ensures: scalability, security and code reusability. This solution would allow Banks to reduce integration costs, increasing speed and making an innovation platform also available to developers and fintechs. In the figure 3.8 is clearer where is the position of the API in an open banking environment.

Multiple projects on the fintech world are technologically based on APIs, used to open systems to parties included in the ecosystem by increasing the value of the service for the final customer. In the blockchain worlds a lots of project expose APIs to deliver their services. For example, the main players active in the field of oracles have adopted APIs for delivering to distributed blockchain-based application values which are outside the block, such as the price of gold.

With the introduction of APIs, a PISP perform payments, while an AISP provides access to bank account information. As a result, both are competitors to banks, acting as the interface for the customer to access all banks and accounts. Notably, one of the declared targets of PSD2 is fostering competition and innovation, so this change in the competitive landscape is not by accident, but by design.

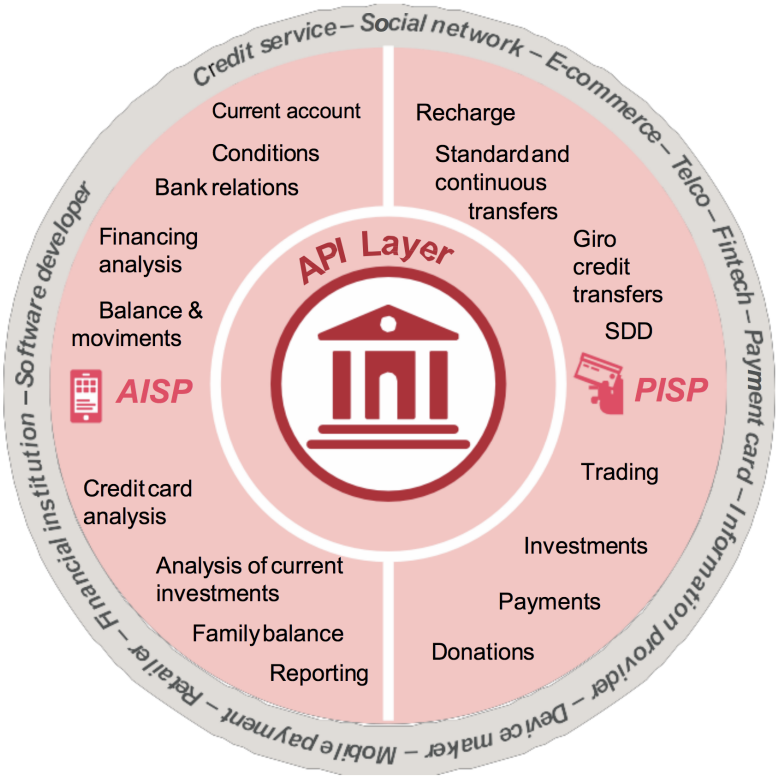

Figure 3.8: API in the open banking model (source: PWC p.4).

On the traditional bank side, this change has two consequences: one is that banks must provide interfaces, such as application program interfaces to the TPPs; notably, PSD2 explicitly prohibits the use of screen scraping.

However, due to the change in the competitive landscape with TPPs becoming the potential interface to the customers, banks are also urged to rethink their business models. Do they want to lose customers and business to others? Or do they want to stay in control of the value chain, acting as both a bank and a TPP? Given that APIs must be provided, one request to the finance industry is to show significant progress in standardising APIs. There are initiatives underway both in the finance industry and the open source community, notably the Open Bank Project [26]. OBP is the sandbox used for performing a PSD2 payment in the loyalty point system developed in the thesis project.

In conclusion, banks themselves must enable such APIs, requiring the management and security for APIs, but also creating an architecture which ensures the interfaces between external TPPs, the APIs, new types of own applications, and the stable core banking systems are well defined, secure and scalable. Having a "bimodal" or "multi-speed" IT now becomes mandatory. From a security perspective, a thought out, multi-layered security approach, including API security management, is inevitable.

3.2.7 Blockchain possible involvement

In the following sections some possible applications of combining the blockchain technology with the opportunity offered by the reception of the PSD2 directive.

Strong Customer Authentication

Many blockchain technologies manage the authorisation of transactions through an infrastructure based on the asymmetric cryptography. In the blockchain network every public key is associated

to a device and the related private key allows members to validate the transaction. The process is very similar to that of the digital signature, without the obligation to use an accredited CA and allows the use of a self-generated key pair, solely stored on a mobile device.

In this way, it offers the possibility of considering the device to be an ownership element to which only the user has access, just like the OTP devices currently in use. It is possible to implement the possession factor of SCA, using the blockchain technology to bind a particular user to a specific mobile device. StrongAuth is an implementation of user-friendly SCA solution that exploits blockchain technology; in fact, StrongAuth is a mobile authorisation system that connects a traditional application of PKI with immutability and security of the blockchain.

Timestamping

Currently, timestamping (with gambling) is actually considered the best example of blockchain application. The timestamp value given by a blockchain is guaranteed and validated by nodes of the network and indeed is secure and reliable.

StrongAuth also exploits from a blockchain is the timestamping: every signature from users is timestamped on the immutable ledger, thus providing what we use to call eAES[1] that adds the trustless time evidence to the well known and defined AES used also for the electronic eIDAS[2] framework.

Timestamping is performed by StrongAuth using the public Open timestamp protocol: the Timestamp server aggregates all the responses created during the interval of time and it assembles them into a single hash that will be stored into the blockchain. This information is needed to prove that provided response signed by particular user existed at the specific time it was performed. With this mechanism StrongAuth notarises efficiently and immutably great amount of data using a single transaction and allow external actors to verify responses in a SelfService mode, thus reducing auditing effort and costs [27].

3.2.8 Complementary regulation

The PSD2 directive is contextual to Regulation (EU) 2015/751 which puts a cap on interchange fees charged between banks for card-based transactions. This is expected to drive down the costs for merchants in accepting consumer debit and credit cards. As written in the XXX [17]:

> The revised union legal framework on payment services is complemented by Regulation (EU) 2015/751 of the European Parliament and of the Council. That Regulation introduces, in particular, rules on the charging of interchange fees for card-based transactions and aims to further accelerate the achievement of an effective integrated market for card-based payments.

[1] enhanced Advance Electronic Signature.

[2] Identification Authentication and Signature.

Chapter 4

Ethereum: a next-generation smart contract and DApp platform

The goal of this chapter is to provide an analysis of the Ethereum blockchain technology, which is the platform used for the development of thesis project. In the following section, the chapter starts analyzing several definitions; these are given to the reader in order to clarify the reading of the present document.

4.1 Definitions

In the official documentation, the name "Ethereum platform" and "Ethereum protocol" refers to the "Ethereum" blockchain technology itself, follows its definition:

Ethereum is a public, open source, blockchain-based platform that allows to build and run smart contract and DApps[1] without any downtime, fraud, control or interference from a third party.

The Ethereum project code is entirely available on `https://github.com/ethereum` divided in several repositories; the researches on which the Ethereum foundation is currently working on are at `https://ethresear.ch/`. Anyway, is opportune to specify that the name "ether" has a different meaning:

ether (ETH) is a cryptocurrency generated by the Ethereum platform; it is the main internal crypto-fuel; indeed, it can be transferred between accounts, it is used to pay transaction fees and to compensate participant mining nodes for computations performed;

cryptocurrency is a digital currency that secures transactions with cryptographic code, which is solved through hardware computational power, known as mining or proof-of-work, or other less energy-intensive ways such as proof-of-stake [28].

4.2 Objectives

The Ethereum founder, Vitalik Buterin, clarify the objectives of its blockchain in the Ethereum whitepaper [29]:

[1] Distributed Applications.

> What Ethereum intends to provide is a blockchain with a built-in fully fledged Turing-complete programming language that can be used to create "contracts" that can be used to encode arbitrary state transition functions, allowing users to create any of the systems described above, as well as many others that we have not yet imagined, simply by writing up the logic in a few lines of code.

4.3 Main features of the Ethereum protocol

The Ethereum protocol is focused, by design, on building decentralized applications with smart contracts. Vitalik Buterin outline five important principles on whom the Ethereum protocol is based [29]:

Simplicity

Keep the technology as simple as possible means helping new developers to create distributed applications, without knowing the entire specifications. Simplicity in Ethereum is core, the founders said that was even necessary to sacrifice costs and time to obtain it. The white paper confirm that simplicity is a key concept asserting that any optimization which adds complexity should not be included, unless that optimization provides very substantial benefit:

> The Ethereum protocol should be as simple as practical, but it may be necessary to have quite a high level of complexity, for instance to scale, to internalize costs of storage, bandwidth and I/O, for security, privacy, transparency, etc. Where complexity is necessary, documentation should be as clear, concise and up-to-date as possible, so that someone completely unschooled in Ethereum can learn it and become an expert.

Abstraction

The concept of abstraction in Ethereum refers to the multitude of possibilities of development in such a platform. It is a universal platform completely open source, where developers can develop any kind of program without any type of limitations. Developers are allowed to create any kind of smart contract, transactions, financial assets, cryptocurrency and tokens as well:

> A fundamental part of Ethereum's design philosophy is that Ethereum does not have "features". Instead, Ethereum provides an internal Turing-complete scripting language, which a programmer can use to construct any smart contract or transaction type that can be mathematically defined.

Modularity

Ethereum has been designed as modular, this means the protocol itself it is divided in separate parts which allows the platform to continue working if some updates have to added. It would have been pretty inefficient to shut down the entire system even for a small protocol modification. The white paper reports a clear example:

> Over the course of development, our goal is to create a program where if one was to make a small protocol modification in one place, the application stack would continue to function without any further modification. Innovations [...] should be, and are, implemented as separate, feature-complete libraries. This is so that even though they are used in Ethereum, even if Ethereum does not require certain features, such features are still usable in other protocols as well. Ethereum development should be maximally done so as to benefit the entire cryptocurrency ecosystem, not just itself.

Agility

Details of the Ethereum protocol are settled. The founders have not designed Ethereum as a permanent structure but as an agile platform, even if they make clear that:

> Although we will be extremely judicious about making modifications to high-level constructs [...].

Non-discrimination & non-censorship

As every public blockchain well-designed, censorship resistance is one of the key concepts. Every kind of usage of the platform is allowed and there are no restrictions to any specific category:

> All regulatory mechanisms in the protocol should be designed to directly regulate the harm and not attempt to oppose specific undesirable applications. A programmer can even run an infinite loop script on top of Ethereum for as long as they are willing to keep paying the per-computational-step transaction fee.

4.4 Clients applications on Ethereum

Currently in Ethereum there are two essential types of applications: wallets and full nodes:

- wallets are just lightweight nodes that use the Ethereum platform just for sending and receiving cryptocurrencies, mainly ETH;
- full nodes are CLI that can perform the full set of operations available on the network.

4.4.1 Wallet

For sending and receiving crytopcurrencies on Ethereum all needed is just a wallet application. A wallet is simply an hardware or a software application that holds keys to the EVM. These keys corresponds to an account, which is referred to by an address.

The most powerful and used on Ethereum is the official wallet: Mist browser; its code is available in the GitHub repository of the Ethereum organization. It is a user-friendly wallet that can also execute smart contracts and even access web-app-like programs, that is why is called browser [30].

4.4.2 Full nodes

Operating a full node on Ethereum means having the possibility to create and deploy smart contracts on the network.

To clarify the functions of wallet and full nodes it must be done an introduction to Ethereum address and accounts, and transactions.

4.5 Accounts

An account is a data object: an entry in the blockchain ledger, indexed by its address, containing data about the state of the account, such as its balance. This concept is linked to an address is a public key belonging to a particular user; it's how users access their accounts. The address is technically the hash of a public key, not the public key itself [31].

In Ethereum, accounts do not store personal information such name, surname, and so on; nut are just pseudonym. Anyone can generate as many address as he/she likes. Wallet application often ask a password to protect the keys with encryption, the private key must be kept absolutely secret. Ethereum accounts are so represented by long hexadecimal addresses, such as the following one:

`E82d7CD3186212819D152b6d27ac88762B147F55`

The address is derived as the last 20 B of the public key controlling the account. The address is often indicated explicitly by prepending 0x to the address. Since each byte of the address is represented by 2 hex char, a prefixed address is 42 characters long.

It is important to clarify that ETH in Ethereum are not contained in a particular machine or application. The ether balance of every node can be queried, and ether sent and received by any computer running an Ethereum full node or a wallet. That is because Ethereum is a public blockchain, hence so, it is fully transparent. Every transactions ever made on the Ethereum main network can be consulted on the ethereum explorer website [32]: Etherscan. Even if the computer where a wallet is gets destroyed all is needed is just the private key and ether can be accessed in any node.

On the other side if someone lose the key, his/her ether are lost forever. Furthermore, if someone hand over another's private key, that person can access the EVM and pull the money out without knowing who he/she is.

Two measure need so to be taken:

- backup the private key;
- do not give the private key to anyone.

4.6 Transactions

A transaction T is a single cryptographically-signed instruction constructed by an actor externally and recorded in an Ethereum block. A human is assumed as the very external actor; otherwise, software tools will be used in its construction and dissemination.

There are two types of transactions: those which result in message calls, and those which result in the creation of new accounts with associated code, known informally as "contract creation". Both types specify a number of common fields, widely discussed in the Ethereum yellow paper, the most reliable document regarding transactions in this network [33]. Follows a list of the most important ones:

- `nonce`: a scalar value equal to the number of transactions sent by the sender, formally T_n;
- `gasPrice`: a scalar value equal to the number of wei to be paid per unit of gas for all computation costs incurred as a result of the execution of this transaction; formally T_p;
- `gasLimit`: a scalar value equal to the maximum amount of gas that should be used in executing this transaction; gasLimit is paid up-front, before any computation is done and may not be increased later, formally T_g;
- to: the 20 B address of the message call's recipient, or $\varnothing$, for a contract creation transaction, used here to denote the only member of $B0$; formally T_t;
- `value`: a scalar value equal to the number of wei to be transferred to the message call's recipient or, in the case of contract creation, as an endowment to the newly created account; formally T_v;
- `v`, `r`, `s`: values corresponding to the signature of the transaction and used to determine the sender of the transaction; formally T_w, T_r and T_s.

Additionally, a contract creation transaction contains: `init`, an unlimited size byte array specifying the EVM code for the account initialisation procedure, formally T_i. `init` is an EVM code fragment; it returns the body, a second fragment of code that executes each time the account receives a message call (either through a transaction or due to the internal execution of code). `init` is executed only once at account creation and gets discarded immediately thereafter.

In contrast, a message call transaction contains: `data`: an unlimited size byte array specifying the input data of the message call, formally T_d.

For sure, one of the most singular aspect of an ethereum transactions is the gas.

4.7 Gas

Gas is the fundamental network cost unit [33]. Basically in Ethereum ETH can be converted freely in and from gas as required. Gas does not exist outside of the internal Ethereum computation engine; its price is set in gwei[2] by the sender of the transaction and miners are free to ignore transactions with low gas prices to mine. Indeed gas price higher means an higher revenue for the miner of the transaction.

The Ethereum network needed gas in order to avoid issues of network abuse; hence so, every operation is subject to fees in this environment. The fee schedule is specified in units of gas; thus, any given fragment of programmable computation has a universally agreed cost in terms of gas, which includes creating contracts, making message calls, utilising and accessing account storage and executing operations on the virtual machine.

It is important to take into account that gas does not exist outside of the execution of a transaction. Every transaction has a field, which is a specific amount of gas associated with it: `gasLimit`. This is the amount of gas which is implicitly purchased from the sender's account balance. The purchase happens at the `gasPrice` specified in the transaction by the sender; obviously, the transaction is considered invalid if the account balance cannot support such a purchase. It is named `gasLimit` since any unused gas at the end of the transaction is refunded, at the same rate of purchase, to the sender's account. Thus for accounts with trusted code associated, a relatively high gas limit may be set and left alone.

In general, ether used to purchase gas that is not refunded is delivered to the beneficiary address, the address of an account typically under the control of the miner. Transactors are free to specify any `gasPrice` that they wish; however, miners are free to ignore transactions as they choose. A higher gas price on a transaction will therefore cost the sender more in terms of ether and deliver a greater value to the miner, and thus will more likely be selected for inclusion by more miners.

Miners, in general, will choose to advertise the minimum gas price for which they will execute transactions, and transactors will be free to canvas these prices in determining what gas price to offer. Since there will be a weighted distribution of minimum acceptable gas prices, transactors will necessarily have a trade-off to make between lowering the gas price and maximising the chance that their transaction will be mined in a timely manner.

4.7.1 Estimated gas unit for operations

In order to clarify, EVM code and EVM assembly have to be defined:

EVM code is the bytecode that the EVM can natively execute and it is used to formally specify the meaning and ramifications of a message to an account; its the EVM own language, to which your smart contracts compile.

EVM assembly is the human-readable form of EVM code.

[2] GigaWei, a subdenomination of ether.

Transactions need to provide enough initial gas to pay for all computation and storage. For instance, every time a transaction is performed, the sum of its operations costs for everyone in the network the price of 21000 unit of gas, according to the costs of the EVM assembly operations [34]:

value	*name*	*gas used*	*description*
0x00	STOP	0	halt the execution
0x01	ADD	3	addition operation
0x02	MUL	5	multiplication operation
0x06	MOD	5	modulo remainer operation
0x08	ADDMOD	8	modulo addition operation
0x16	AND	3	bitwise AND operation
0x20	SHA3	30+6*word	Keccak-256 hash
0x31	BALANCE	400	get the balance of the given account
0x56	JUMP	8	after the program counter
0x54	SLOAD	200	Paid for a SLOAD operation
0x55	SSTORE	20000	to non-zero from zero
0x55	SSTORE	5000	remains unchanged or set to zero
0xF0	CREATE	32000	create new account with associated code

Table 4.1: EVM assembly most significant operations.

4.8 Currency

As said, the Ethereum network includes its own built-in currency, ether, which serves the dual purpose of providing a primary liquidity layer for efficient exchange between different types of digital assets and, more importantly, for paying transaction fees. The smallest sub-denomination of ether is the wei, and thus, the one in which all integer values of the currency are counted. Specifically, 1 ether is defined as being 10^{18} wei. Compared also with EUR and USD, the following are all the ether subdenominations:

multiplier	*name*
10^{18}	Wei
10^{15}	Kwei
10^{12}	Mwei
10^{9}	Gwei
10^{6}	Szabo
10^{3}	Finney
1	Ether
10^{-3}	Kether
10^{-6}	Mether
10^{-9}	Gether
10^{-12}	Tether
455.399	USD
388.741	EUR

Table 4.2: ether sub-denominations with prices 26-06-2018 (source Ethereum Converter).

Sub-denominations are similar to the concept of “dollars” and “cents” or “BTC” and “satoshi”. In the near future, we expect ether to be used for ordinary transactions, finney for micro-transactions and szabo” and wei just for technical discussions around fees and protocol implementation. Gwei is currently used as unit for gas price. Nowadays, the remaining denominations should not be included in clients.

4.8.1 Gas price in GWei

As said, gas price is measured in Gwei. The gas price can be set with no limitations by the sender of a transaction; higher is the gas price setup from the sender and faster will be the transaction to be processed by the miner. Usually, three prices are defined to be useful for senders [35]:

name	*price (gwei)*	*execution time*	*cost in dollars*
safe low	1	<30min	$ 0.009
standard	2	<5min	$ 0.018
fast	18	<2min	$ 0.162

Table 4.3: Prices taken 18:24 2018-06-27 (source Ethereum Gas Station).

4.9 EVM: Ethereum Virtual Machine

The EVM can be seen as a worldwide computer that anyone can use, for a small fee, payable in ether. The EVM is a single, global 256-bit "computer" in which all transactions are local on each node of the network, and executed in relative synchrony. It's globally accessible virtual machine, composed of lots of smaller computers [31].

This big computer, which anyone who has a node or wallet application can access, makes it simple to move arbitrarily large amounts of value nearly instantly. Although anyone can use this global virtual machine, none can forge fake money inside it, or move funds without permission. If it seems wasteful to have the entire EVM, all those nodes, replicating the same transactions and slavishly maintaining the same state among thousands of individual computers, it's important to have proper basis for comparison for how financial services IT works today.

The EVM is a paragon of simplicity and efficiency by comparison. More importantly, all that work isn't for naught; In fact, it's the evidence of this work that actually secures the network (proof-of work, see section 2.1.1). By now, the EVM is a generalized, secure, ownerless virtual machine that offers cheap Fedwire-like functionality with other functions on top.

The EMV is a transaction singleton machine with shared state. In computing, this means it behaves like one giant data object, rather than what it is: a network of discrete machines, themselves singletons, in constant communication. For the perspective of a software developer, the EVM is also a runtime environment for small programs that can be executed by the network (smart contracts, see section 4.10). The EVM can run arbitrary these computer programs, today mostly written in Solidity language. These programs, given a particular input, produce always the same output in the same way, with the same underlying state changes. Because of that, Solidity programs are fully deterministic; moreover, it is guaranteed to execute them provided the user have paid enough for transactions.

Solidity programs are capable of expressing all tasks accomplishable by computers, making them theoretically Turin complete. That means that the entire distributed network, every node, performs every program executed on the platform. When one user uploads a smart contract through their ethereum node, it is saved in the latest block and propagated around the network, where is sorted on every other node in the EVM to run the same code, as part of the block processing protocol. The nodes go through the block thy are process and run any code enclosed within the transactions, each node does this independently; it is not highly parallelized, but highly redundant.

The EVM is so a state machine. State machines are machines with memory that can be thought of as beings who never sleep. As a state machine the EVM has a constant history of all transactions within their memory banks, leading back to the very first transaction. A computer's state is the specific outcome of every single state-change that has taken place inside the machine, since it was first switched on. The latest version of the machines's state is the machine's canonical "truth" about reality as it stands right now. In Ethereum this truth deals with account balances and the series of transactions performed. Transactions, therefore, represent a kind of machine narrative,

a computationally valid arc between one state and another. As Gavin Wood's Ethereum yellow paper says [33]:

> Ethereum, taken as a whole, can be viewed as a transaction-based state machine: we begin with a genesis state and incrementally execute transactions to morph it into some final state. It is this final state which we accept as the canonical "version" of the world of Ethereum. The state can include such information as account balances, reputations, trust arrangements, data pertaining to information of the physical world; in short, anything that can currently be represented by a computer is admissible. Transactions thus represent a valid arc between two states; the "valid" part is important, there exist far more invalid state changes than valid state changes. Invalid state changes might, e.g., be things such as reducing an account balance without an equal and opposite increase elsewhere. A valid state transition is one which comes about through a transaction.

EVM is the most trustful and reliable machine that any global network has today. For every instruction the EVM executes, there must be a cost associated, to discourage useless contracts deployment (for that testnets exist). Every time an instruction executes an internal counter keeps track of the fees incurred, which are charged to who is performing them. Each time a sender initiates a transaction that's user wallet reserves always a small portion to pay fees. After a transaction has been broadcast to the network from a give node, the network propagates the transaction around so that all the nodes can include it in the latest block.

4.9.1 Implementations

The following list describe the nowadays implementations of the EVM live on the main network [36]:

- go-ethereum, a popular Ethereum client with its own EVM implementation (core/vm directory);
- Parity in Rust, another popular Ethereum client with its own EVM implementation (ethcore directory);
- cpp-ethereum, an Ethereum client that generates the consensus test suite (libevm/VM.cpp);
- Pyethereum in Python, a mostly deprecated client (ethereum/vm.py);
- Py-EVM, a Python implementation designed to be highly configurable and modular and compliant with the Ethereum test suite, work is in progress on it to run a full node and develop sharding;
- EthereumJ in Java, a client with its own EVM implementation.

4.9.2 Programming languages

The following list describe the different programming languages who compile into the EVM:

- Solidity, the most popular programming language for Ethereum contracts;
- Vyper, a language strictly focused on security, with overflow-checking, numeric units but without unlimited loops;
- Flint, a language with several security features: e.g. asset types with a restricted set of atomic operations;
- HAseembly-evm, an EVM assembly implemented as a Haskell DSL;
- Bamboo (experimental), a language without loops but with explicit constructor invocation at the end of every call.

4.9.3 Debuggers

The following list describe the main available debuggers for Ethereum developers:

- Remix, an IDE containing an EVM code debugger;
- `debug_traceTransaction` method, an instruction-wise trace information provided by go-ethereum.

4.10 Smart contracts

Smart contracts has been thought by Szabo and Miller in 1997. Around the 1990s, it became clear that algorithmic enforcement of agreements could become a significant force in human cooperation. Though no specific system was proposed to implement such a system, it was proposed that the future of law would be heavily affected by such systems. In this light, Ethereum may be seen as a general implementation of such a crypto-law system.

4.10.1 Why the need of introducing smart contracts

EVM applications are referred with the name of smart contracts [31]. On one hand, in other context, the term "contract" refers to a specific kind of contract: a financial one. Financial contracts are agreements to buy and sell something, usually at a specifies price. In the Ethereum context, smart contracts are agreements between accounts, to render a transfer of ether when certain condition are met.

The reason these contracts are "smart" is they are executed by machine, and the assets (ether and other tokens) are moved automatically. These contracts could be enforced even hundreds of years after the've had written, assuming the network is still running then, and even if a lot of bad actor try to interfere. The EVM is totally sandboxed and free from interference, and isolated from other networks too; these conditions make impossible for a party to back out of a smart contract. In practical terms, this is because smart contract are empowered to hold assets in escrow and move them when the terms of the contract are met Smart contracts can be seen as cryptographic "sboxes" that contain value and only unlock it if certain conditions are met.

4.10.2 Solidity: a smart contracts language

Solidity is an high level new programming language; it is used to write programs called smart contracts [37]. Solidity is a statically typed language, which means that the type of each variable (state and local) needs to be specified (or at least known) at compile-time [38].

It was influenced by C++, Python and JavaScript and is designed to target the EVM. Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features. This new language use basically a set of conventions coming from the world of networking, assembly, web development and cybersecurity.

4.11 DApps: Distributed applications

DApps are applications, typically a web application, that run in a browser and interact directly with smart contracts on the blockchain. A traditional web application would have a web client that makes API requests to a backend server and persists. Data would only be stored on a database fully owned by the application and all business logic would occur in the API Layer. In a DApp, the web application mostly reads data directly from the blockchain and writes data via transactions back to the blockchain. The most of the business logic occurs in a smart contract that is deployed to the blockchain.

4.11.1 DAOs

One particular kind of application that is particularly intriguing are DAOs[3]; this includes entities as large as, or even larger than nation-states, social networks, multinational public companies, etc... These applications are born on the idea that is not a reasonable design choice to have a complete autonomous environment/entity. In fact, the code may not be able to handle new issues that arise, so human intervention have to always be an option; but, preferably in the hands of a small, non-profit entity or some decentralized solution.

However, the first DAO, known as The DAO, resulted in many funds being stolen, and Ethereum hard forking into Ethereum and Ethereum Classic. The DAO, which launched with $150 million in crowdfunding in June 2016, was immediately hacked and drained of US $50 million in cryptocurrency. This hack was reversed in the following weeks, and the money restored [28]. This decentralized bailout was made possible by a majority vote of the blockchain's hash rate.

[3]Decentralized autonomous organizations.

Chapter 5

Open Bank Project API

The Open Bank Project, led by Berlin-based software company TESOBE, is an open-source RESTful API solution for banks and fintech industry, which empowers financial institutions and developers to securely and rapidly enhance their digital products. OBP brings developers an entire environment for experimenting with e-payments and for deploying open banking platforms; this is done by providing access to more than 160 standard banking APIs for transactions and for payment initiation, used by over 8000 fintech community worldwide.

5.1 APIs Objectives

APIs comes out in several sandboxes, environments that offer a shared and controlled development. The sandbox itself gives to financial institutions several ecosystems of third party applications and services, and contains multiple test data which emulates real transactions and offers [39].

OBP enables developers to use a wider range of software applications and services for securely connect to bank accounts, never calling for login credentials to users. As a result of using APIs, OBP is an abstract layer that has the objective of facilitating and securing the developers work in building fintech applications interacting with financial institutions. OBP aim to enable third party apps to connect to bank accounts on desktop, mobile or any other platforms, analyse transaction data, and promote and leverage open data.

5.2 Architecture

The Open Bank Project APIs abstract away the differences in banking systems and provides a uniform technical interface that software developers can easily use to build secure services on top of the bank [40]. As the figure 5.1 shows, the applications using these APIs are very secure because customers, the account holders, always have to log into the bank; no other entities see their username and password. Furthermore, since it is open source technology, there is no vendor lock-in; in fact developers can "fork" the code from OBP GitHub and use commercial or open source licenses. Moreover, teams, from OBP or partners can extend the Open Bank Project API as required.

With the Open Bank Project is possible to: offer a wider range of web and mobile applications to customers; reduce integration and maintenance costs for new applications; achieve better data control and security for the end-users; leverage a growing community of developers and deploy new apps instantly. Inside sandboxes, approved fintech solution providers can access services from participating in building modern payment solutions and can access different kind of data and metadata, such as customers informations, accounts, transactions, payments, but even comments, URLs, geo tags, as is possible to see in the figure 5.1.

Here is shown an high level overview of the OBP API:

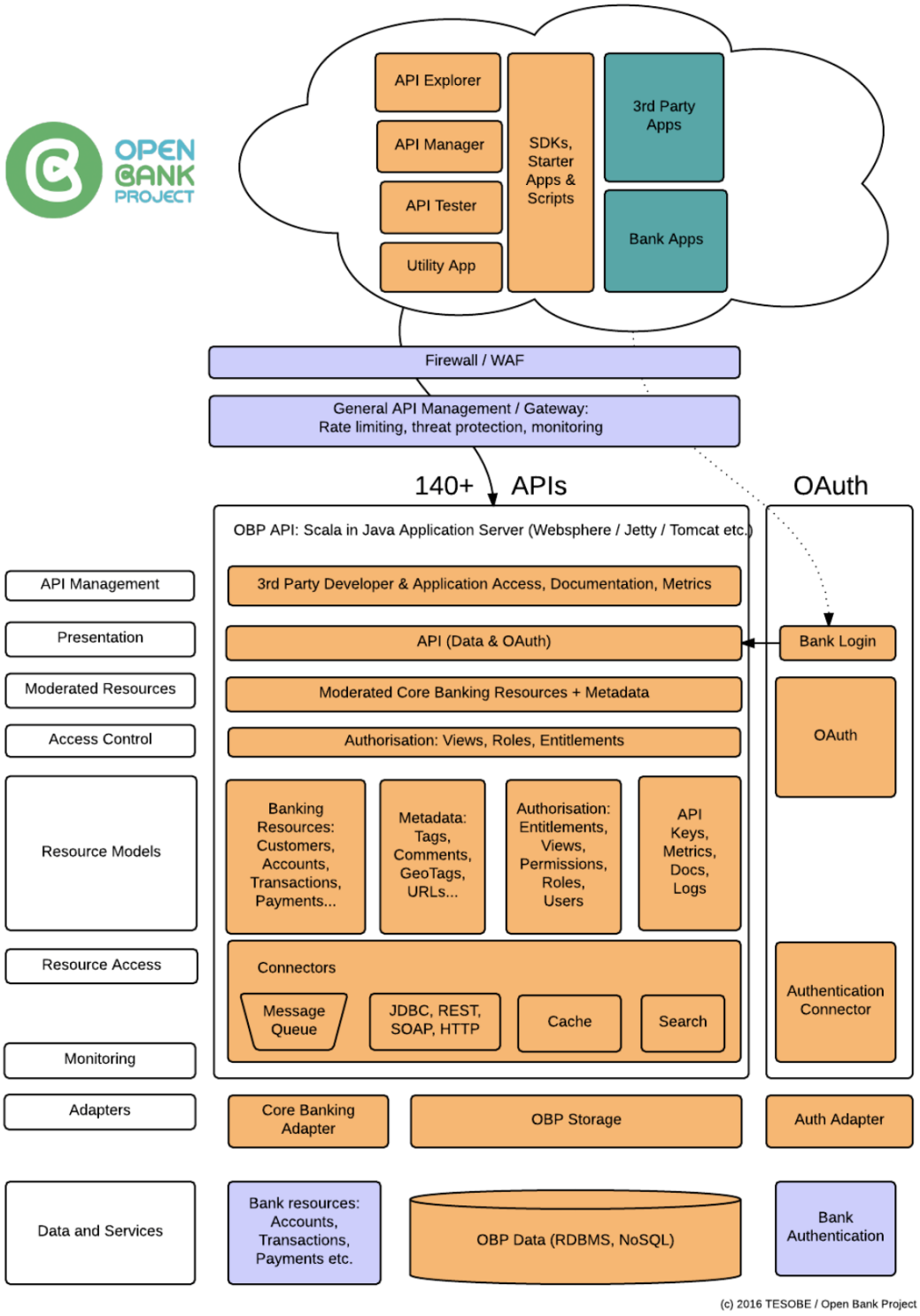

Figure 5.1: Open Bank Project architecture (source: OBP Architecture).

The OBP APIs has been implemented thinking about four classes of possibles users:

- banks, to quickly deliver a greater range of apps and services;
- developers, to build next-generation fintech apps and deploy to a wide range of banks;
- customers, to enjoy a large offer of banking applications;

- society, to raise the bar of financial transparency and unlock an untapped innovation potential.

5.3 Banks

Like online banking today, every financial institution will offer an API tomorrow. According to Gartner, by 2016, 75% of the top 50 global banks will have launched an API platform and 25% will have launched a customer-facing app store. OBP aim to equip these institutions with the right technology and expertise to embark on the open banking journey [41].

The goal of Toledo is to transform the bank following the "bank as a platform" principle; in order to make this technically possible, the open source API were developed allowing banks to make different kind of applications available to it's customers at low cost. It is a technical layer that sits on top of any bank allowing the applications to securely interact with the bank, logging in it directly.

5.4 Developers

With the Open Bank Project, developers via RESTful API can perform transaction, integrate bank account information, use OAuth authentication, write once and run code everywhere, and leverage a consistent and bank-agnostic interface.

Developers can use the following environments [42]:

- the general Open Bank Project sandbox;
- the PSD2 sandbox;
- the Emirates NBD sandbox;
- the UK Open Banking Working Group OBP sandbox;
- the Ulster Bank sandbox;
- the RBS OBP sandbox;
- the BNP Paribas sandbox.

Using these API sandboxes is easier to build a company solution that offer a modern and standards-based technology stack, tested and approved by tier 1 banks, and is simple to create a growing ecosystem of third party developers and applications that enable customers enjoy a true app store experience from day one, generating a product more cost effective to maintain. Access account information and transaction history, enrich transactions with metadata like tags, comments, pictures and geolocation, create and access different views on accounts, transfer funds are some of the feature available for developers.

5.4.1 OAuth

As is possible to see in the figure 5.1, OBP uses OAuth to protect resources; indeed, for developing with one of the sandboxes is required to get a consumer key and consumer secret for the calls and then use OAuth authentication. This was a design choice made by OBP because OAuth 2 it is considered as stable as OAuth 1, which is an RFC and used by important enterprises like Twitter and Mastercard.

The best way to get started with OBP and OAuth is probably to fork one of the several Starter SDKs which take care of the basic OAuth flow; so, developers can choose between the followign ones: Node, Python, C#, Mac, Android, Scala DJango, iOs, Android, and PHP.

5.5 Sandbox PSD2

This Open Bank Project PSD2 sandbox demonstrates a PSD2 API solution. The API provides a secure avenue that allows bank account holders to access their banking data and services via approved third party applications, following the consent of both the bank and customer. This sandbox can be used to explore the PSD2 API catalogue, test an example API powered application and register for a developer key to start building applications.

The process to get started with the sandbox PSD2, and with every sandbox in general, is the following [43]:

1. get the API key, the consumer registration;
2. create a free developer account, and submit basic information about the application at the stage;
3. connect the application to the API using OBP SDKs with the developer key, which was provided during account creation;
4. test the application, using the whole set of functions and available data.

Inside the PSD2 sandbox, applications can access the user's list of accounts and account information, provide fine-grained accounts, transaction access to guests (auditor, accountant or public), explore banks, branches, ATMs, products, access open data related to banks, including geolocation and opening hours. It is possible to Initiate transaction requests and explore transactions; for instance, access the transaction history and transaction metadata, enrich transactions and counterparties with metadata including comments, pictures and tags. The following list shows the resources and the features that the sandbox PSD2 currently support:

- banks and their branches, ATMs and products;
- accounts, including any public accounts;
- transactions, for the specified account;
- transfers, with a list transfer methods.
- create a transfer, results in transaction only on success;
- cards, held by a user;
- metadata management, including urls, tags, comments and geographical information tags;
- entitlements management, such as manage views/roles on accounts to reveal a subset of account information and only allows certain actions;
- challenges and responses, in addition to primary authentication, the API provides a generic challenge-response mechanism for operations that require further security checks;
- OAuth user authentication, so the bank remains the gate-keeper;
- monitoring and control, displaying functions usage metrics and revoke or accept access of third party applications;
- starter SDK's, with hello-world style applications on GitHub that already contain the OAuth flow.

Developing the OBP PSD2 API encourages a community of developers to grow around the bank and enable them to build innovative products and services for customers based on the "bank as a platform" principle. Services may be offered via an application store and the best ideas can be cherry picked for use in branded interfaces.

5.5.1 API Explorer

API Explorer for OBP API is a Scala and Liftweb application that consumes the OBP-API resource documentation so that developers can browse and interact with the OBP REST API endpoints [44]. A list of the possible operations in the PSD2 API sandbox are shown in the table 5.1.

type	*function*
API	Get API Configuration
API	Get API Info (root)
API	Get Adapter Info
Doc	Get API Glossary
Doc	Get Message Docs
Doc	Get Resource Docs
Doc	Get Swagger documentation
Account	Get Account by Id (Core)
Account	Get Accounts Held
Account	Get Accounts at Bank (IDs only)
Account	Get Accounts at Bank (Minimal)
Account	Get Accounts at all Banks (My)
Bank	Get Transaction Types at Bank
Counterparty	Get Counterparties (Explicit)
Counterparty	Get Counterparty by Counterparty Id.(Explicit)
Counterparty	Get Other Account by Id
Counterparty	Get Other Accounts of one Account
Transaction	Get Transaction by Id
Transaction	Get Transactions for Account (Core)
Transaction Req	Answer Transaction Request Challenge
Transaction Req	Create Transaction Request (COUNTERPARTY)
Transaction Req	Create Transaction Request (FREE_FORM)
Transaction Req	Create Transaction Request (SANDBOX_TAN)
Transaction Req	Create Transaction Request (SEPA)
Transaction Req	Get Transaction Request Types for Account
Transaction Req	Get Transaction Requests

Table 5.1: PSD2 sandbox API overview.

Chapter 6

Project: solution of an Ethereum-based loyalty point system

This chapter concerns the project work of the thesis. In the following sections, the architecture of the developed solution and the main functions implemented are going to be discussed; furthermore, this document includes the developer's guide (see appendix A), where are described the core modules, the interface, and the data structure; and the user manual (see appendix B), displaying how to install and use the program and its tools.

6.1 Premise

In this section is explained why makes sense to build up a loyalty program application on Ethereum, and is provided an analysis of the advantages of using a loyalty point blockchain-based system, in comparison with the traditional loyalty schemes.

6.1.1 What are loyalty programs

Loyalty programs are structured marketing strategies designed by merchants to encourage customers to continue to shop at, or use the services of businesses associated with each program [45]. These programs are exploited by a significant multitude of companies in several commercial fields, each one having specific features, but always a reward-schema "points for money".

In marketing generally, and in retailing more specifically, the most common reward schema uses a card that could be: a loyalty card, a rewards card, a points card, an advantage card, or club card [46]. This is a plastic or paper card, visually similar to a credit card, debit card, or digital card that identifies the card holder as a participant in a loyalty program. Other loyalty program can be totally digital. By presenting a card or using a digital service, purchasers typically earn the right either to a discount on the current purchase, or to an allotment of points that they can use for future purchases. Hence so, the card or the digital account is the visible means of implementing a type of what economists call a two-part tariff.

Application forms for cards and services usually entail agreements by the store concerning customer privacy, typically non-disclosure of the store of non-aggregate data about customers. The store uses aggregate data internally, and sometimes externally, as part of its marketing research. Over time the data can reveal, for example, a given customer's habits, or customers' most bought product.

One can regard loyalty programs is so already a form of centralized virtual currency, one with unidirectional cash flow, since reward points can be exchanged into a good or service but not into

cash. As a results, such a schema fits perfectly with a total conversion to a digital decentralized service.

6.1.2 Disadvantages of traditional loyalty schemes

Currently, in the field of marketing, modernisation of traditional loyalty schemes is a crucial topic. On one side the customer expectations are changing, but on the other they have also become expensive to run and difficult to unwind.

Securing customers' loyalty goes beyond having a basic loyalty program. Loyalty is the brand's ability to be top of mind in a customer's head as well as to secure a sense of allegiance from consumers [47]. Allegiance is much harder to achieve at a time when every consumer has different expectations and responds to different triggers when it comes to engaging with a brand.

The main disadvantages of nowadays loyalty schemes are the following:

- they have become a tired concept that needs to be reinvented;
- the customer experience requires to be more personalised, relevant and exclusive;
- they cannot meet rapidly changing customer expectations;
- loyalty schemes can become a financial liability for businesses;
- loyalty solutions need to be agile to build and enhance the offering.

6.1.3 Advantages of blockchain-based loyalty schemes

Building loyalty rewards programs on top of a blockchain, for sure can bring an exiting customer experience. The main advantages of using a blockchain-based solution for a loyalty program are the following [48]:

Cost reduction

One of the blockchain main features is to reduce cost thanks to the fact that smart contracts are deployed in the network and deployers does not need to buy a physical infrastructure or to rent virtual machines. The trade-off cost savings is going to regard system development, system management, electronic transactions, and product acquisition. In addition, the costs for fraud and errors are significantly reduced as a result of the cryptographic properties of the blockchain.

Creation of a standard system

Supermarket points of a customer should be potentially in the same wallet from which he/she redeems points for communication providers. Through a trustless smart contract standard, blockchain-based applications can centralise the multiple customer's loyalty programs. Loyalty providers decide how and with whom the customer uses these rewards, but from a consumer perspective, his/her ability to access and manage them is practically frictionless.

Fast transaction speed

Transactions in blockchain are incredibly fast and accessible; hence so, providing a loyalty program with a speeded up transactional process for transferring points will improve performance and the overall customer experience.

Improved security

Blockchain creates an immutable and time-stamped distributed ledger record of every single transaction ever made, making transactions easily traceable, but also making them irreversible, and preventing double spending, fraud, abuse, and any other type of manipulation.

Creation of new opportunities

Building a loyalty program on platform such as Ethereum means creating smart contracts that provide all the main advantages of the public blockchains. Currently, the power of using efficient and secure structures such as smart contracts has not fully discovered yet, and these could be an added value for those businesses that will be able to find new type of applications.

6.2 Entities

The loyalty program system developed in the thesis project is an Ethereum-based web application where different entities can interact with each other through a front-end. The fidelity points used are Ethereum tokens, designed following the ERC-20 standard (see section 6.6). Concerning who tries to interact with the application, the front-end shows different functions.

Once the smart contract with the business logic application has been implemented, it has been deployed to the Ethereum testnet, Rinkeby (see section 6.5.1). The system is designed for recognising three different kind of accounts, that represent the following entities:

- admin;
- users;
- shops.

Every entities of the system must have an Ethereum address in order to interact with the application, because without an address is not possible to send and receive points, is not possible to call the functions of the application, and it is not possible to be payed in ether. Ethers, as already said, are the gas of the system, so every entity of the application need to have the right account balance in order to pay the functions called. The effective costs are analysed in the chapter 7. In order to collect points and ether, every entities need to have a wallet; and the application was designed to work specifically with the wallet/extension Metamask (discussed in section 6.5.2).

Admin

The admin was designed to represent the service provider; but the service provider could also delegate someone else to be the "manager" entity in the fidelity point application. The admin is the entity who can generate points inside the system and distribute them both to users and shops. Concerning the points generation, when the application is set up the admin instantly receive freely a very big amount of points. If the will run out of points, it will have to pay the smart contract deployed on Ethereum to generate new ones calling a specific function.

User

The user is the main receiver of the points, which are stored in his/her personal wallet, and he/she can use them to redeem physical or digital prices from official shops of the system. The user in order to use the system and receive points have to be logged and authenticated in the web application. The loyalty points service provides the customer with a front-end function for signing up, and so registering to the system. The authentication is managed off-chain with the Firebase authenticator (see section 6.4.4).

Shop

Shops can be on one side external shops and on the other side also the service provider can declare itself as an official shop, offering customers its products and services as rewards. Shops which are external have to make a request to enter in the circuit of official "price-givers" for the users; obviously, the official provider shop does not. Shops in order to be listed in the "shop page" have to perform a free blockchain request to the admin; then, the admin, examining the shop informations (such as bank, Ethereum account, etc...), can decide to approve or reject them. Only If they are approved, shops will have their own selling page, which will be created by the fidelity point company, to offer goods and services for points.

Shops receive a certain amount of points from users who redeem prices, so they have to provide the service to the user or ship the product chosen. A complete set of pages for checking and managing the status of the expedition has been implemented in the project. Moreover, shops after sending the product and collecting points can ask the admin to be refunded of goods sold to the users[1]. In order to do that, shops which owns a consistent amount of points have to make a payment request the admin in ether or euro. The application form for performing the payment request in the front-end, gives to shops the possibility to be payed in ether or in euro via psd2; the Ethereum request is stored on the blockchain, on the other hand the psd2 payment request is stored on the Firebase realtime database as pending. Shops can see from its payment request page, if its requests is actually: pending, in white; rejected, in red; or succeeded, in green (see figure 6.1). The admin, analysing the request, can decide to approve or reject it.

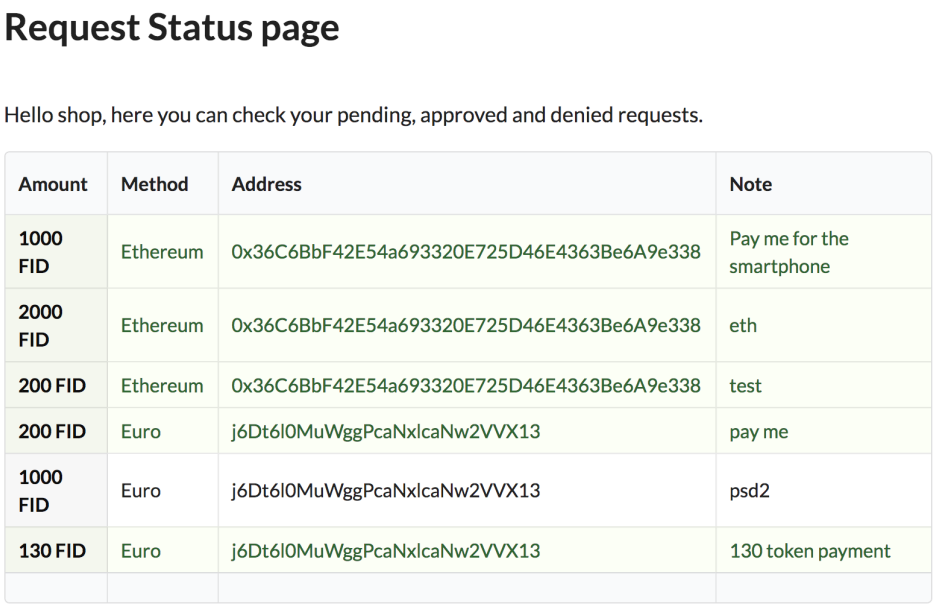

Figure 6.1: Request status page of a shop.

6.3 Architecture overview

The figure 6.2 describe the architectural overview of the loyalty point system. The schema focus on a set of infrastructural components, existing outside the blockchain, which interact through the front-end with the blockchain.

[1]Note that the financial aspect of billing product bought with points has not been analysed

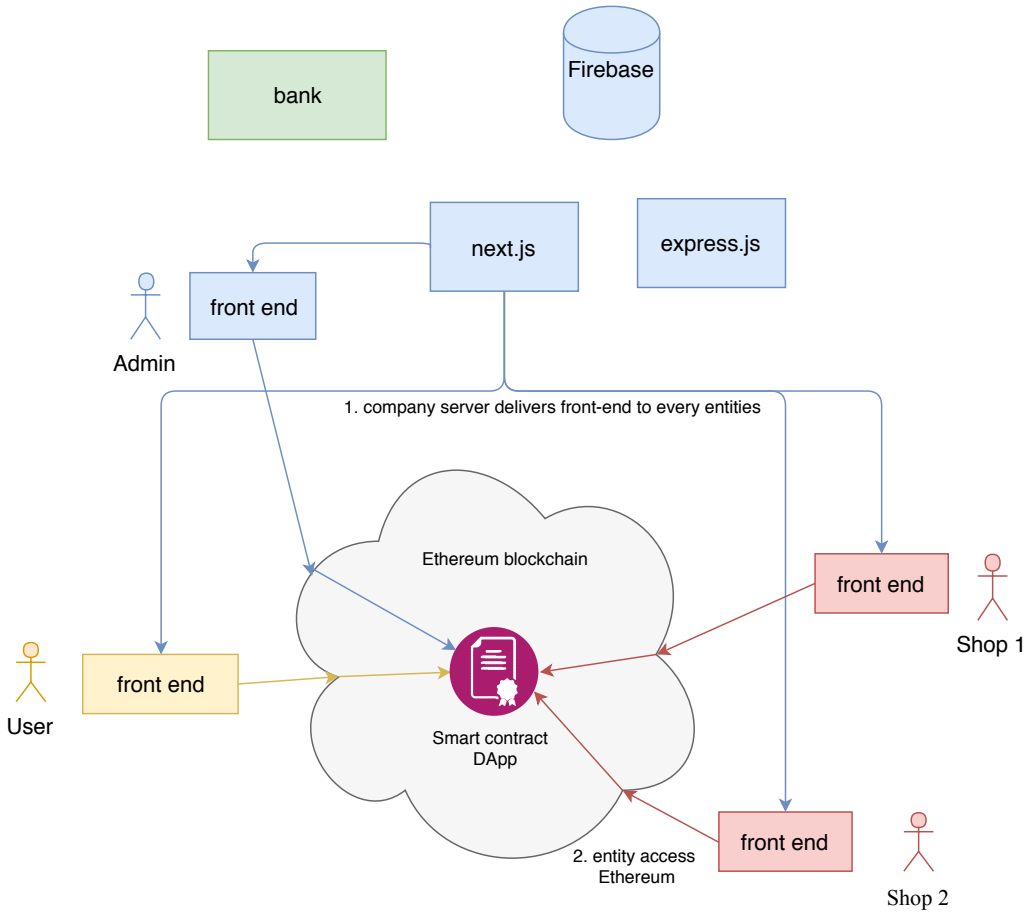

Figure 6.2: Architecture overview.

6.4 Components outside the blockchain

Outside the blockchain there are three entities, the ones previously described: the admin, users and shops (see section 6.2) and furthermore there is the bank, which is used just for the psd2 interaction.

First, the loyalty point system provider has the principle infrastructural components:

- next.js, the server which delivers the front-end web application to everyone, for the blockchain interaction;
- the front-end, the react.js application which allows the entities to interact with the smart contract;
- express.js, the server used for building up the interaction with a bank, through a psd2 transaction;
- Firebase, the platform used for both authenticating user e storing psd2 payments data;
- the admin entity, the physical person who interact with restricted functions of the web application (discussed previously in section 6.2).

6.4.1 Next.js server

Next.js is a minimalistic and lightweight framework and it is used in the project for server-rendering the react.js application. Like web applications created with php, where some .php files are just

written and deployed, in next.js the developer does not need to worry much about routing, and the application rendering lays on the server by default. With next.js, the applications are written in javascript, node.js and react.js.

6.4.2 Front-end

In order to deliver a user friendly interface and make the entities interact with the blockchain a front-end of the application has been implemented, and this is provided to the users, shops and even the admin, from the next.js server. Incoming and outgoing data are converted to graphical interface for the entities to view and interact mainly with the blockchain through digital interaction using html, css and react.js. There are several tools available that can be used to develop the front-end for this kind of web applications, but react.js was chosen because it is a widely used modern javascript framework, which allows developers to build client-side application really fast and creating a responsive web design.

It must be specified that every function that allows to interact with the blockchain is client-side; hence so, the user totally controls the operation called from the contract. The interaction with the smart contract deployed is managed through the web3.js library, developed officially by Ethereum.

React.js

React enables developers to declaratively describe their user interfaces and model the state of those interfaces. This means, instead of coming up with steps to describe transactions on interfaces, developers just describe the interfaces in terms of a final state (like a function). When transactions happen to that state, react takes care of updating the user Interfaces based on that; react is just javascript, there is a very small API to learn, just few functions [49].

web3.js

The web3.js library is a collection of modules which contain specific functionality for the Ethereum ecosystem. The `web3-eth` is for the ethereum blockchain and smart contracts. The `web3-utils` contains useful helper functions for DApp developers. To help web3 integrate into all kind of projects with different standards are provided multiple ways to act on asynchronous functions; moreover, most web3.js objects allows a callback as the last parameter, as well as returning promises to chain functions [50].

This is done because, Ethereum as a blockchain has different levels of finality and therefore needs to return multiple "stages" of an action. For instance, to cope with requirement a `promiEvent` for functions like `web3.eth.sendTransaction` or contract methods. This `promiEvent` is a promise combined with an event emitter to allow acting on different stages of action on the blockchain, like a transaction. The `promiEvent` work like a normal promises with added on, once and off functions. This way developers can watch for additional events like on "receipt" or "transactionHash".

6.4.3 Express.js server

Express is a minimal fast and flexible node.js web application framework that provides a robust set of features for web and mobile applications [51]. With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy. Express provides a thin layer of fundamental web application features, without obscuring node.js features. Many popular frameworks are based on express.js and this service is the main choice for using the PSD2 API Sandbox of OBP.

Through express.js, the admin from the front-end can call the OBP API and perform a transaction to the shop, using its own bank account bank and calling itself for the money transfer.

6.4.4 Firebase

Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014. The platform offers a set of several services, in these project has been used just two of them: Firebase authentication and realtime database. To add Firebase to an application, a Firebase project have to be created and a short snippet of initialisation code, with few details, need to be added in the project. After creating a new application in the Firebase console, the app can use the services that it needs; moreover, Firebase services are free in development phase.

Authentication

Firebase Auth is a service that can authenticate users using only client-side code. It supports social login providers Facebook, GitHub, Twitter and Google (and Google Play Games). In the project it was included a user management system to enable the user authentication with email and password login, stored with Firebase [52].

Concerning the loyalty point service, Firebase authentication allow users to login in the web application using the email address and password sign-in methods, as shown in figure 6.3. To use it is just necessary to install the Firebase SDK and paste the configuration code into your web page as described. In addition, for signing up new customers or shops, a form that allows new users to register has been added. When a user/shops completes the form, validate the email address and password provided by the user; then, the authentication account is created.

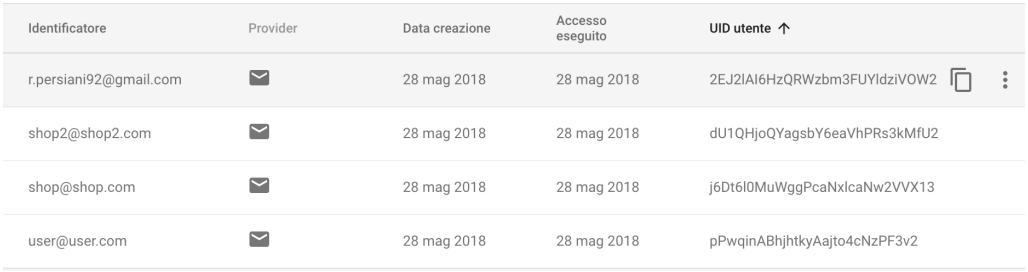

Figure 6.3: Authentication in Firebase.

Realtime database

Firebase provides a realtime database and backend as a service; It provides application developers an API that allows application data to be synchronised across clients and store on Firebase's cloud. The company gives client libraries to enable integration with android, ios, javascript, java, objective-c, swift and node.js applications. The library written in node,js is the one used by the thesis application.

The Firebase realtime database is a cloud-hosted database where data are stored as JSON and are synchronised in realtime to every connected client. When building cross-platform application with javascript SDK, all the clients share one realtime database instance and automatically receive updates with the newest data [53].

The Firebase Realtime database is used in the project to store all that data that does not need to be on the blockchain, such as shop bank details and PSD2 payment request performed by shops, which need to be approved by the admin.

6.4.5 OBP PSD2 Payment

The PSD2 OBP API is the interface the admin use to perform a transaction through an application built on the express.js middleware. It is assumed that both admin and shops have their own bank account; as a matter of fact, banking data are required for shops at the moment of the subscription.

Basically, in the system, the admin call the `create transaction request` operation exposed by the API, which initiates a payment via transaction request. This is the preferred method to create a payment and supersedes the old method `makePayment`. This operation is exactly a third party access to payments, which is on of the core tenet of PSD2; in fact, this call satisfies that requirement from several perspectives:

- a transaction can be initiated by a third party application, the loyalty point system;
- the customer, which is the admin him/herself in the project, is informed of the charge that will incurred;
- the call uses delegated authentication via OAuth.

Account data can be easily created on the OBP PSD2 Sandbox webpage (see figure 6.4), filling the form with bank name, account id, desired currency, and with the initial balance of the account. In the project, in order to be payed in euro through psd2, a shop must provide the name of its bank and the bank account; otherwise, its registration in the system would not be allowed an it would not ever become an official shop.

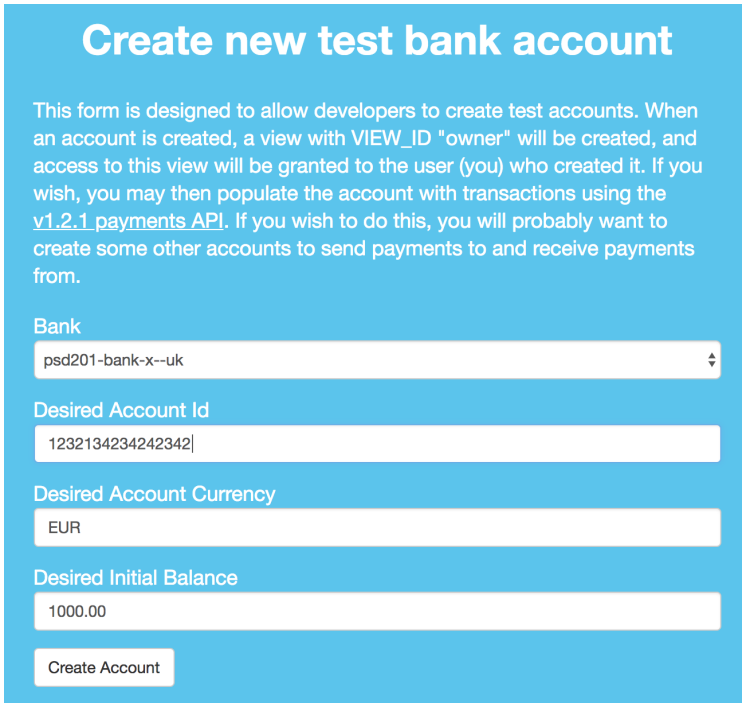

Figure 6.4: Create new bank account in the OBP PSD2 sandbox.

6.4.6 Application

The admin perform the payments for an application which is build up on express.js; so, there is a link that redirect him/her from the next.js front-end to the middleware interacting with the bank. This application is really "one function" and just allows the admin to perform the transfer without changing the parameters of the form, which are pre-calculated. The template of the application is written in pug.

Pug

Pug is the middleman and a template engine for node.js. It is a template engine that allows to inject data and then produce html, often used for rendering when working with express.js [54]. In short, at run time, pug (and other template engines) replace variables in files with actual values, and then send the resulting html string to the client.

6.5 Components used for smart contracts interaction

This sections shows the tool which are related with the Ethereum blockchain; it must be specified that the project is using an Ethereum testnet, Rinkeby; otherwise, every transaction would have been payed.

6.5.1 Rinkeby and testnets

Rinkeby, with Ropsten and Kovan, is one of the official test networks of Ethereum, and is the testnet used for the project development. Testnet stands for a network that is not a real blockchain network, but just emulates the behaviour of a blockchain for testing purposes.

As a result, the network cannot guarantees advantages like immutability, security, and intermediaries avoidance. As a matter of fact, the Rinkeby testnet is a proof-of-authority network, so uses a different consensus mechanism than the main Ethereum, where there is no mining at all and so it is not fraud proof. However, Solidity developers does not care about security in testing phase in Rinkeby.

On the other side, the Ropsten testnet is a proof-of-work network, so more similar to the public main net; but, it is a sort of weak mining and this network is often slower thank Rinkeby, because of developers using a ridiculously high gas price to speed up their operations. In these testnets, in fact, gas does not have to be payed, and developers can use an illimitate amount of ether. In addition, in Ropsten, the mining is done just for helping the developer community, there is no real financial incentive as opposed to main net. Against this background, every transaction performed on the testnet happens with the same pattern as the Ethereum main network.

6.5.2 Metamask

Metamask is the tool used for collecting points in the project. MetaMask is a bridge that allows to visit the distributed web through the browser; It a great tool for development because allows to run Ethereum DApps right in the browser, without running a full Ethereum node.

The figure 6.5 shows the Metamask interface of the admin of the loyalty point system (account 1). On the left-top of the Metamask interface, the dropdown menu allows to select the network; currently, the interface is in the Rinkeby network. The admin ethereum address is showed partially under the account name; furthermore, it is displayed the balance in ether and even in the corresponding conversion in dollars.

In the image on the left is possible to check the latest transactions; actually, there are starting from the top to the bottom:

- a point transfer transaction, in fact the ether transferred are zero, but the transfer succeeded;
- a contract deployment transaction, when a smart contract is deployed in the blockchain network;
- a ether transfer transaction, in fact the ether transferred are different from zero and the transfer succeeded;
- a rejected transaction, which could both be refused by the sender through Metamask or not mined by the network due to insufficient gas.

In the image on the right is possible the set of different tokens owned admin (they are different even if they have the same name); furthermore, the orange button “add token” allows the Metamask user to set new tokens, adding the contract address that creates them.

MetaMask includes a secure identity vault, providing a user interface to manage different accounts on different sites and sign blockchain transactions. MetaMask add-on can be installed in

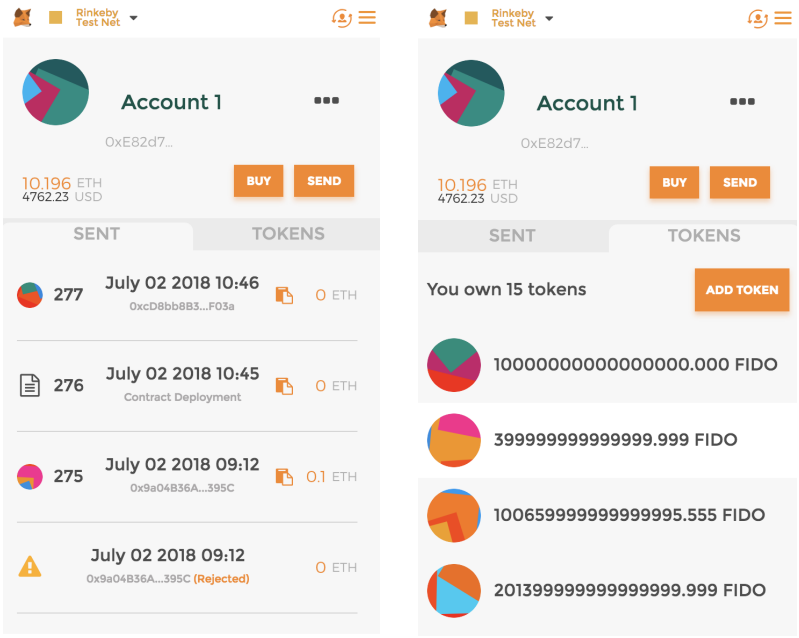

Figure 6.5: Metamask interface.

Chrome and other main browsers. The goal of this tool is to make Ethereum easy to use for many people as possible [55].

The MetaMask browser extension, enables browsing Ethereum blockchain enabled websites. The extension injects the Ethereum web3 API into every website's javascript context, so that the loyalty point DApp can read from smart contracts deployed in Ethereum (Rinkeby). MetaMask also lets the user create and manage their own identities; in fact is possible to have multiple accounts in the same instance of the extension. Indeed, in the project, was possible to test admin, shops, and users operations from the same instance.

Moreover, when, for instance, a user wants to exchange points for a price, he/she writes into the blockchain through a secure interface to review the transaction, before approving or rejecting it. Metamask compute a pre-calculus of the fee costs and enables user to manage the gas price and the gas limit. Because it adds functionality to the normal browser context, MetaMask requires the permission to read and write to any webpage. The source of MetaMask is open on GitHub [56].

6.5.3 Etherscan

Etherscan is the web application used to inspect in depth every transaction happened in the network [32]. Every network have its own Etherscan to inspect transactions and stats, including, on one hand, the Ethereum main network, but on the other hand, also Rinkeby, Ropsten and Kovan testnets. In the figure 6.6 are listed the latest transactions executed or reject with the loyalty point smart contract.

The contract address: `0x3735543206B4bbA0fC09E2Bd2E4906D2143AeB9B` is displayed on top of the page; instead, bottom, is shown the full timeline of the operations performed with the actual value, the fees, the block number of the blockchain, and incoming or outgoing status.

In addition, we can see that this is a "token contract", and that the token created in is named "Fido Coin(FID)"; moreover, also the creator address is displayed, in fact the address `0xe82d7...` match the admin address, which have also performed the last transaction (still "pending"). It is clear that Etherscan shows exactly what it is the fully transparency of a public blockchain; even the code and the bytecode of the contract are public.

The figure A.4, analyse the last transaction of the figure 6.6, the one which was pending. However in the new image describe a step forward because the status is "Success". This means

that it is successfully performed. Concerning the other informations, it is showed how the gas limit was set and how many gas has been used with the respective gas price decided by the sender. The actual cost fee must be calculated. Anyway the price shown is 0$; in fact, no expenses happens in a test network, but if this transaction would have been performed in the main network, the fee would have to be payed.

Furthermore, the image provides a description of how many points are transferred from the admin to an user, 10,000 points in this use case. Every address is a link to another page where are displayed more information; it could be a user account, a contract, a token contract, a block. Everything is described in Etherscan.

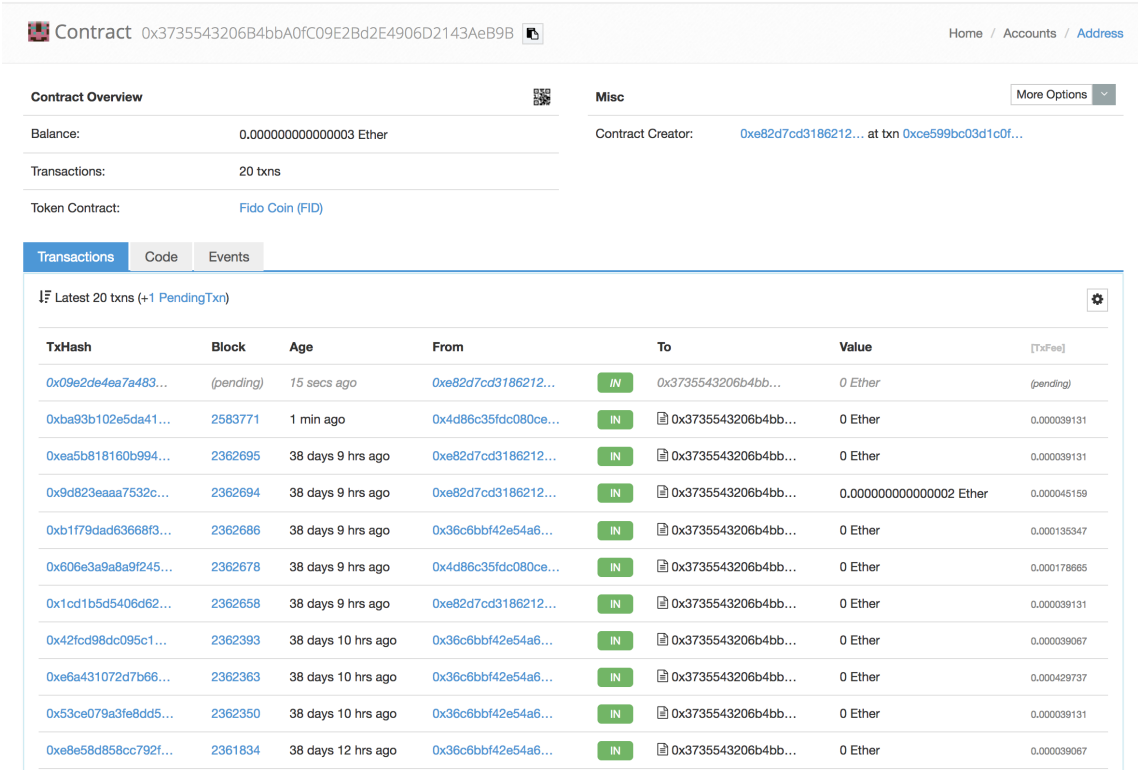

Figure 6.6: Loyalty point smart contract.

6.5.4 Remix IDE

Remix is a powerful, open source tool that helps developers to write Solidity contracts directly from the browser. Written in javascript, Remix supports both usage in the browser or locally [57]. The online Remix IDE is a useful tool for testing, debugging and deploying smart contracts and much more in browser-based ecosystem. It is stable, and is updated at almost every release. Concerning the project, Remix was used to the test efficiently the smart contract functions, because allows to call functions straight, without the need of a front-end application.

6.6 Points

Points in the actual loyalty points system are Ethereum tokens. An Ethereum token is a currency used in a particular DApp; however, a point in the system has a value extremely low, respect to an ether; as a consequence, this makes the token value not qualifiable ($1\ ether = 10^{18}$ points). So, the idea is that Ethereum not only has its own currency, but also has tokens on top of it which can act as currency themselves.

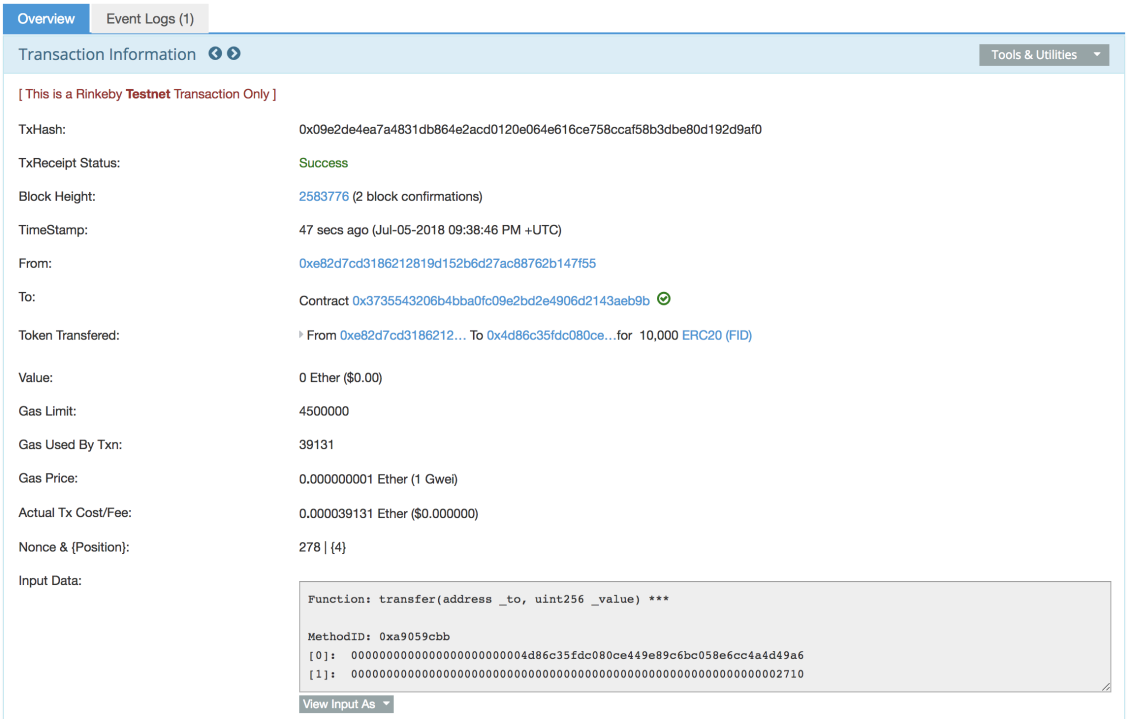

Figure 6.7: Point transfer transaction.

6.6.1 Tokens

Tokens in the Ethereum ecosystem can represent any fungible tradable good: coins, loyalty points, gold certificates, IOUs, and even in-game items. Between 2.9 and 5.8 millions of private as well as institutional users actively exchange tokens and run the various transaction networks. In May 2017, the whole market capitalisation of active cryptocurrencies and tokens surpassed $91 billion [58]. Since all tokens implement some basic features following a standard, this also means that a token will be instantly compatible with the Ethereum wallet and any other client or contract that uses the same standards [59].

As a metaphor, Ethereum can be seen as the internet and all the DApps as websites that run in it. DApps are all decentralized and not owned by an individual, they are owned by people once the Dapp smart contract is deployed; therefore, customers have to buy certain tokens of that DApp in exchange of your ether to use their functions. It must be specified, that a token is not needed in every DApp, it depends if it may be useful or not. These tokens are distributed by the DApp deployers in what is usually called “ICO”, a sort of crowd sale (see section 6.6.2).

These tokens are usually of 2 varieties:

- usage tokens, the tokens that act like native currency in their respective DApp, like points used in the project;
- work tokens, tokens that identify the possessor as a sort of shareholder in the DApp; a perfect example is the DAO tokens, where a DAO token holder had the right to vote on whether a particular DApp could get funding from the DAO or not.

Regarding all the DApps which live in the Ethereum network, then why don’t users simply use ether to pay for every transaction? Why a native currency is needed for them? The answer is that even in real life there are tons of services where form of token are used over cash, such as tickets.

By using points to execute certain functions in the smart contract of the DApp, the process became much simpler and seamless. Points are easy to send, and without owning them is impossible to performe requests like asking for a payment to the admin or buy a product from a shop.

Currently, the most important and used standard for realising token in Ethereum is ERC-20.

6.6.2 ERC-20 standard

The ERC20 standard is basically a specific set of functions which developers must use in their tokens to make them ERC20 compliant. While this is not an enforced rule, in fact the loyalty system points just implements some of them, most DApp developers are encouraged to follow the standards to ensure that their tokens can undergo interactions with various wallets, exchanges and smart contracts without any issues. This was great news for everyone because now they at least had an idea of how future tokens are expected to behave. ERC20 tokens have gotten widespread approval and most of the DApp sold on ICO's have tokens based on the ERC20 standard.

So, a token need to have to be ERC20 compliant needs essentially to declare some variables and a set of six functions that can be recognised and identified by other smart contracts. When executed, the following four basic activities are what all the ERC20 tokens required to do:

- get the total token supply;
- get the account balance;
- transfer the token from one party to another;
- approve the use of token as a monetary asset.

ICO

Initial coin offerings are public offers of new cryptocurrencies in exchange of existing ones, aimed to finance projects in the blockchain development arena. In the last eight months of 2017, the total amount gathered by ICOs exceeded 4 billion US$, and overcame the venture capital funnelled toward high tech initiatives in the same period. A high percentage of ICOs is managed through smart contracts running on Ethereum blockchain, and in particular to ERC-20 token standard contract [60].

Usually, when a new and exciting DApp comes along, the tokens are sold through the ICO; however, the application developed in the thesis does not required to sell tokens, and indeed it does not even needed to do an ICO. The fact is that points should be just used for redeeming purpose and not as exchange money; in addition, nowadays ICO are not good views because of many fraud and scams happened.

6.7 Transaction Types

Three different types of transactions are possible inside the application: point transfer, ether payment and PSD2 payment.

6.7.1 Point Transfer

The point transfer stands for the operation of transferring points from a sender to a receiver in the loyalty point system. Basically, the possible use case are four:

- user sending tokens to a shop, to redeem a price;
- shops sending tokens to the admin, asking to be payed in recognised cryptocurrency or euro;
- admin sending tokens to a user, following a fidelity point policy (e.g using a service and receiving point as prize for the usage);
- admin sending back token to a shop after a payment request is rejected for some reason.

In the first three use cases, the operation of point transfer is performed by filling the form in the figure 6.8, inserting the value and the address of the receiver, and clicking on the button "Transfer". Before the mining of the transaction starts, the Metamask extension begins the pre-calculus of the operation, looking if the gwei amount chosen by the sender are enough to pay miners, and asks for a confirmation of the price fee calculate (see figure 6.9). Moreover, Metamask display also if the transaction is going to fail or succeed.

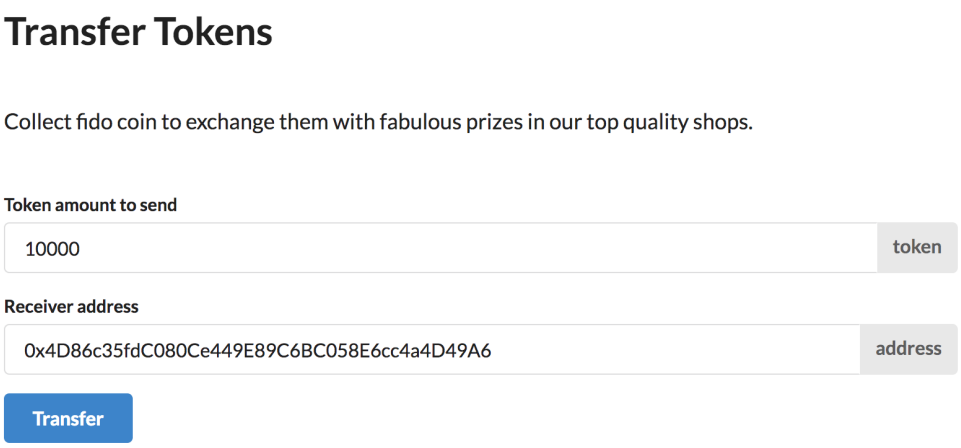

Figure 6.8: Point transfer form.

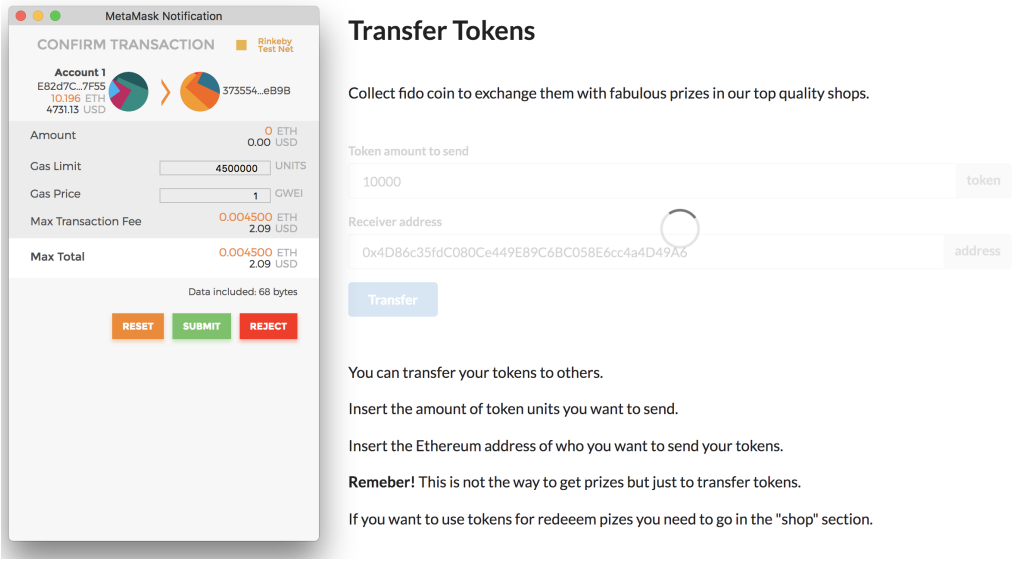

Figure 6.9: Metamask pre-calculus.

After the Metamask confirmation, the transaction is sent to the blockchain and after the appliance of the proof-of-work consensus protocol, the transaction is stored on the blockchain of the Ethereum testnet, Rinkeby. As shown in the figure A.4 no ether are moved from an account to another during the point transfer; the only ether used are those for paying the miners, so the Ethereum fees.

6.7.2 Ethereum Payment

The Ethereum payment is the operation which allows the shop to be payed for the prices given to the users in ether. The Ethereum payment is done just when the admin click on the approve button of the request. The price is pre-decided by the smart contract system. Again, the admin

need to confirm also on Metamask the decision of performing the transaction, specifying gas limit and gas price, remembering that higher is the gas price is set, higher is the mining speed.

It must be clear that before performing the payment in ether (and also PSD2) the admin need to receive a request from a shop with the corresponding point amount it wants to redeem.

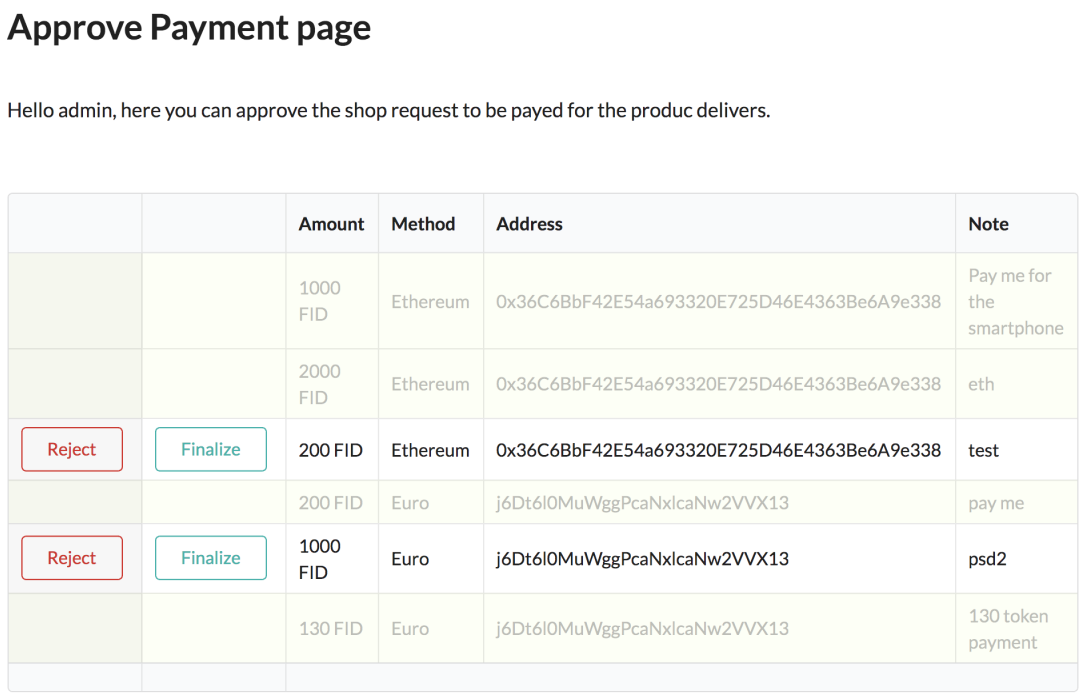

Figure 6.10: Admin page for the payment approbation.

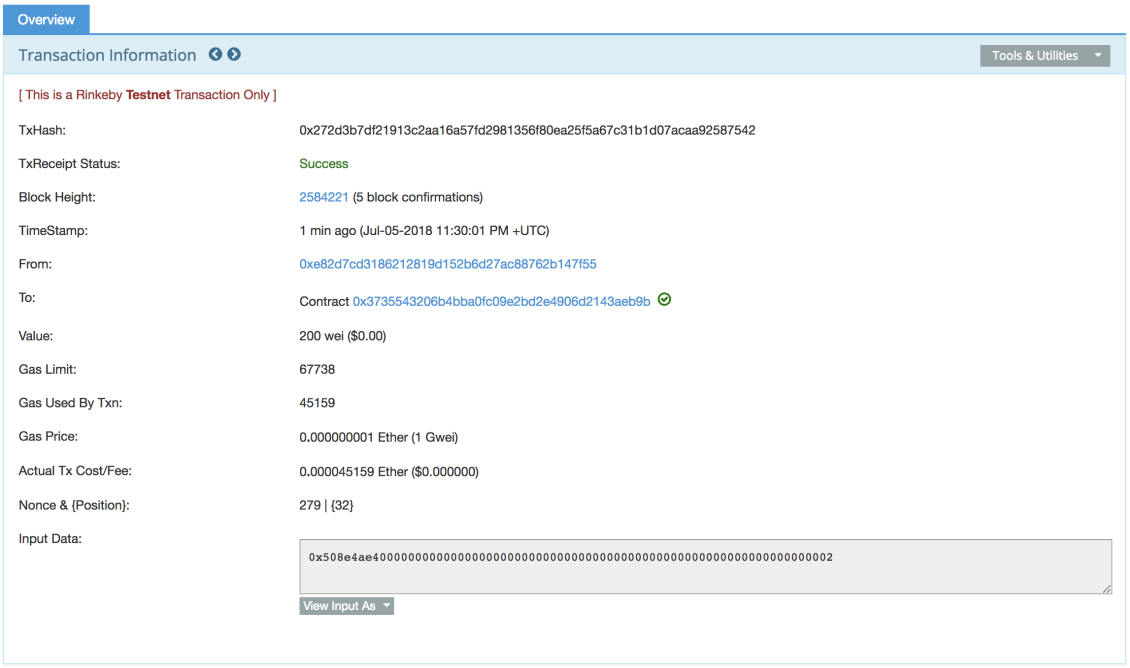

Figure 6.11: Ethereum transfer transaction.

6.7.3 PSD2 payment

The PSD2 payment is the operation which allows the shop to be payed by the admin in euro, for the prices given to the users. The shop requests are stored in the Firebase realtime database, as shown in figure 6.12. The table shows if the pending payment has been completed or reject, showing the timestamp of when the request was made, and the token amount for which the shop was asking for a refund.

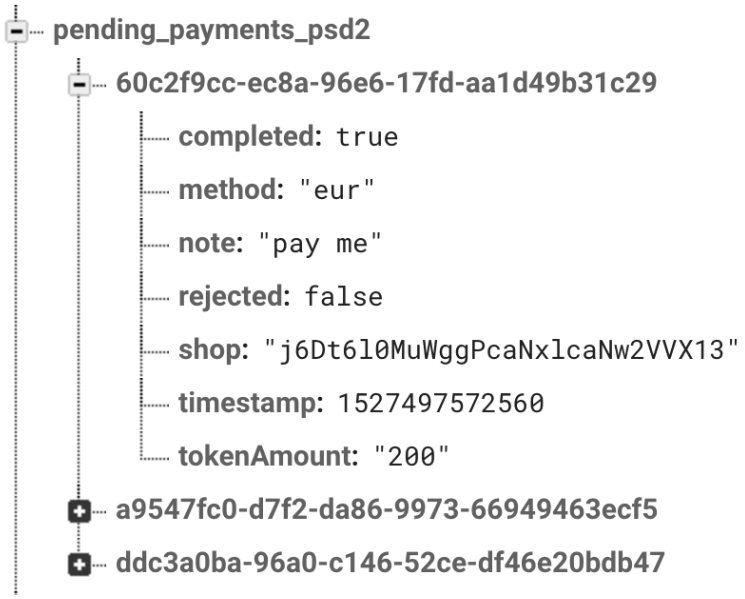

Figure 6.12: Psd2 pending payment database structure.

From a web page of the front-end next.js application, the admin can push a button for approving PDS2 payment request and he/she is redirected with a PID to the express.js and it is authorised through OAuth and have to login in the OBP sandbox (see figure 6.13). After that he can transfer the pre-calculated amount of euro to the shop through a pre-filled form, such as happen in the figure 6.14.

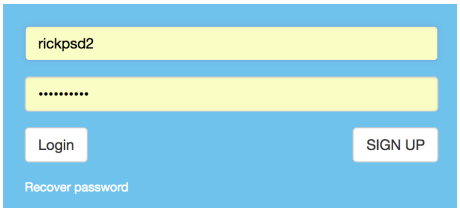

Figure 6.13: OBP Authentication.

After performing the payment, both shop and admin checking their bank account can see their changed balance. This can be done thanks to the OBP dashboard (displayed in figure 6.15) that comes up in the API Explorer section of the PDS2 sandbox. The dashboards allows to check balance and transactions selecting them starting from the left with the dropdown menu of banks, then for every bank all the accounts created are showed and then other informations including all the ids of the transactions performed by an account holder. The data are in JSON, in the figure 6.16 are showed partial data of a transaction from the owner of the account `12456734234` of the `psd201-bank-x--uk`.

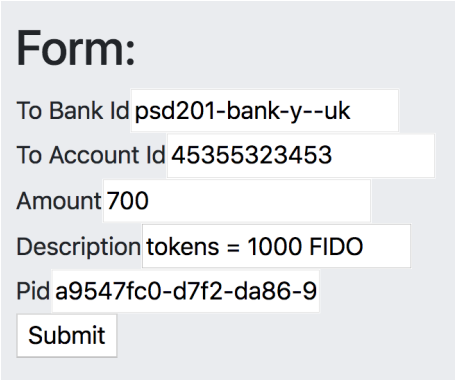

Figure 6.14: Psd2 payment form.

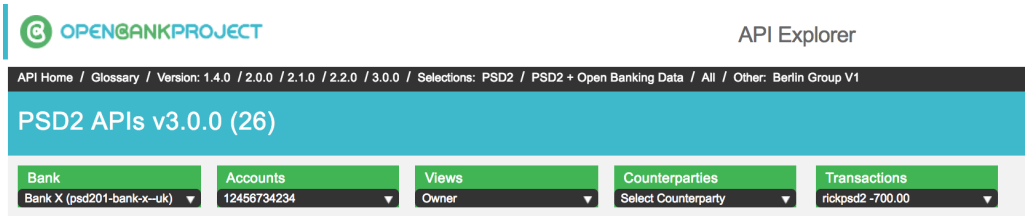

Figure 6.15: OBP PSD2 sandox dashboard.

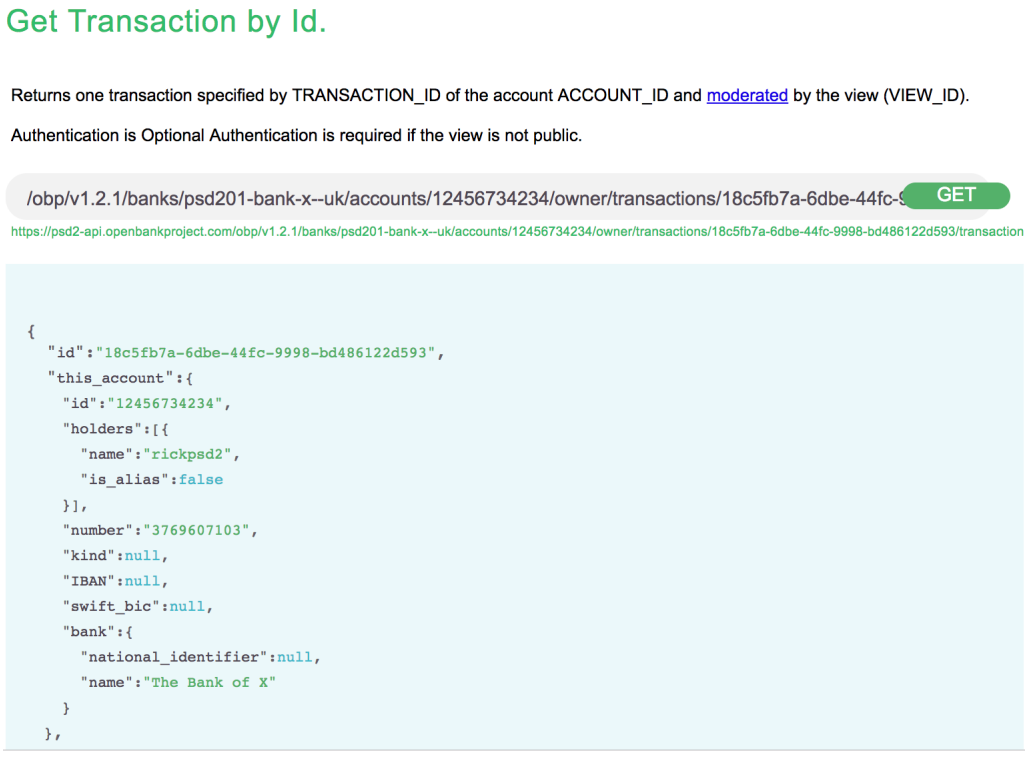

Figure 6.16: Transaction inspection from the PSD2 Sandbox web application.

Chapter 7

Process and experimental results

This chapter is related to the results acquired in the project. What is evaluated are the performance and the cost of the three payment methods used in the loyalty point system: the points transfer, the Ethereum payment, and the PDS2 euro payment. A description of how these methods works is provided in the chapter 6 (subsection 6.7). The analysis of the result is divided in two main parts: the first concerning an overview of the current costs of the traditional payment methods; the second one regarding the value of the performance of the methods experimented in the project.

7.1 Traditional e-payment methods

Several years ago, with the advent of computers and electronic communications a large number of e-payment systems have emerged. These include debit cards, credit cards, electronic funds transfers, Paypal. Standardisation has allowed some of these systems and networks to grow to a global scale, but there are still many country-specific and product-specific systems. The dated meaning of the term "e-payment" referred to a payment made from one bank account to another using electronic methods and forgoing the direct intervention of bank employees [61].

Here are described the disadvantages regarding the most important e-payment methods used online in Italy:

Credit cards

Mainly, for a potential user a credit card payment is a secure and flexible way to purchase goods and services, and can be a good way to spread the cost; but, if used only for minimum payments, a credit cards can be costly. The credit card allows to charge the seller for the money of a product sold; in fact these amount would be ducted only at the end of the month from the buyer account.

The main costs and disadvantages are the following:

- high interest payments, if user does not clear the balance at the end of each month;
- the debt spiral, if the user miss just one payment, the interest will start to add up;
- additional fees, especially penalties for exceeding credit limit;
- annual cost, specifically for the credit cards which offers additional benefits;
- fee for the seller, for the POS hire and for the commission applied.

Furthermore, another limit is the more expensive use abroad; however, this very much depends on the card. Some are designed for travellers, others are more expensive when it comes to fees and other charges depending upon whether you use the card for purchases or cash withdrawals.

Prepaid Card

Prepaid cards offer the convenience of a credit and debit card without the borrowing; as they do not require a credit check, they could be useful to those with bad credit or who have been refused credit and want to get back into positive financial management habits.

The following are the main issues with e-payments performed with prepaid cards:

- activation and loading fees, applied every time money are transferred to the prepaid card;
- low prevalence, which makes their use very restrict; in fact, the success of a payment does not relies on the bank circuit that is used, because prepaid card are treated differently in several goods and service merchants.

PayPal

PayPal describe itself as a faster, safer and more user friendly e-payment method; the service lets the user pay, send money, and accept payments without having to enter your financial details each time. 173 million people use PayPal to shop on millions of sites worldwide, in 202 countries and with 21 different currencies [62].

However, there are drawbacks in this service [63]:

- high fee for selling, the sender occurs up to 3,4% + 0,35€ for transactions;
- high fee for sending invoices, up to 3.4% + 0,35€ per invoice transaction, based on total sales volume;
- additional fees for transaction outside the euro zone.

SEPA Transfer

The Single Euro Payments Area (SEPA) is a European Union (EU) payments integration initiative aimed at harmonising electronic euro payments in Europe. The main disadvantage for the SEPA transfer is just one but it is extremely relevant. The fact that it does not apply fees both to the sender and the receiver makes the SEPA a really used e-payment but the long waiting time for the payment finalisation really limits its usage. As a matter of fact, the time for the amount accreditation have a median wait of two days.

7.2 Point transfer results

The results displayed refers to the figure A.4. The transaction used 39131 units of gas and the gas price was set with an absolutely low value: 1gwei. Indeed this value coincide with the safe low gas price; hence so the result expected was a transaction mined under 30 minutes and with the lowest price possible. The mining cost for this transaction has been of 39131 gwei which is around 0,01 dollars cent; it must be noticed that the amount transferred does not influence the gas units used. The time for completing it was 5 minutes. In case of the same point transfer but performed with the fast gas price, the cost should have been of 704.376 gwei, which are 0,33 dollars cent, with a finalisation of the operation in 30s.

7.3 Ethereum payment results

The results displayed refers to the figure 6.11. The transaction used 45159 units of gas and the gas price was set with an absolutely low value: 1gwei. Indeed this value coincide with the safe low gas price; hence so the result expected was a transaction mined under 30 minutes and with the lowest

price possible. The mining cost for this transaction has been of 45159 gwei which is around 0,02 dollars cent; it must be noticed that the amount transferred does not influence the gas units used. The time for completing it was 6 minutes. In case of the same point transfer but performed with the fast gas price, the cost should have been of 812.862 gwei, which are 0,38 dollars cent, with a finalisation of the operation in 35s.

7.4 PSD2 payment results

The PSD2 payment in euro happens in real time; as a result, they are instant payment in the moment when the admin authorise the third party application (which is its own system) to complete the operation. Regarding the fees applied, no cost appears in the sandbox provided by OBP; however, the bank which the admin is interacting with, in order to finalise the payment can decide to apply a fee which must not be superior of the bank fees applied on their services. For sure, the bank will decide to apply these fees that, like other traditional payments analysed, will results in a expected 2-3%.

Chapter 8

Final considerations and conclusions

The blockchain technology, especially the public blockchains like Ethereum, are attracting a lot of attention for not just a speculative aspect of investing in cryptocurrency; but, on the other side, for the the massive multitude of possibilities offered by its properties, by the smart contracts, and by the distributed applications. During the thesis work, I had the possibility to attend multiple conference; specifically, in EDCON2018, the Ethereum developer conference happened in Toronto (CA) showed me how much the Ethereum community is growing up.

What I noticed was that even if the target of the conference were developers, the attenders were a blend of professionals from every background. Moreover, the speech of the founders showed how big is the research activity on the Ethereum network. Plasma chain, proof-of-stack, sharding are the new big features coming in the Ethereum world.

I found that the blockchain technology is growing up with applications that will improve security and increase speed and save money. In my opinion, the real impact of the blockchain still have to come because even if the technology is already on point, it will need several years to penetrate in the mindset of the people, as was for the internet technology.

My research on the Ethereum-based loyalty point system founded, specifically, advantages regarding avoiding intermediaries, because ether payments does not involve the bank; and advantages in term of speed and costless execution for a company and for customers. I think that also PSD2 has several limitations compared to Ethereum payments, because relies too much on the presence of a bank, which, if it wants, can deny the access to its accounts. In addition fees are higher than Ethereum, and relies on one currency which is just continental. Ether potentially have no limitations, and even if the word legal framework will not agree on a politics of acceptance of blockchain payments, in few years it will change radically the way of performing e-payments.

Appendix A

Developer's guide

This appendix describes the most important parts of the code written in the thesis project, to develop the latest version of the loyalty point system application. Here is provided a description of the loyalty point smart contract, of the web3.js configuration, and of the express.js middleware configuration and interaction.

A.1 Loyalty point smart contract

In this section is described the structure of the solidity smart contract deployed on Rinkeby.

The address of the contract is the following:

0x3735543206B4bbA0fC09E2Bd2E4906D2143AeB9B

Furthermore, the contract can be consulted on the Rinkeby Etherscan at the following link: https://rinkeby.etherscan.io/address/0x3735543206B4bbA0fC09E2Bd2E4906D2143AeB9B.

The file with the code is at the following path in the project:

`fidelity-points-system-thesis/ethereum/contracts/FidelityPoints.sol`

Inside this file there are:

- the SafeMath library;
- the IERC20 interface;
- the owner restriction;
- the main contract FidelityPoints, which manages the fidelity point system.

A.1.1 SafeMath

SafeMath is a library created by OpenZeppelin in order to give math operations safety checks, throwing errors. Currently, this library is considered core in pretty much every contract that uses mathematical operations. The main reason of its usage is that it provides protection against overflow and underflows. The code related to the library is displayed in the figure A.1.

Concerning the function `div(uint256 a, uint256 b)`, Solidity automatically throws an error when dividing by 0, so the check is not needed.

The `pure` modifier, is a very restrictive attribute which indicates that the function will not alter the storage state in any way; moreover, it won't even read the storage state.

The `internal` modifier can be better compared with "protected" in object-oriented programming languages. Internal functions of the contract C are visible to the code running at the current address (i.e. the current contract instance) but also to contracts derived from C.

```
library SafeMath {
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) { return 0;}
        uint256 c = a * b;
        assert(c / a == b);
        return c;
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a / b;
        return c;
    }

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        assert(b <= a);
        return a - b;
    }

    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        assert(c >= a);
        return c;
    }
}
```

Figure A.1: SafeMath library.

```
interface IERC20 {
    function totalSupply() public constant returns (uint256 totalSupply);
    function balanceOf(address _owner) public constant returns (uint256
        balance);
    function transfer(address _to, uint256 _value) public returns (bool
        success);
    event Transfer(address indexed _from, address indexed _to, uint256 _value);
}
```

Figure A.2: ERC20 token interface.

A.1.2 IERC20

The interface in the figure A.2 refers to some functions that must be implemented to make a token ERC20 compliant.

The `totalSupply()` function must return the total amount of tokens created.

The `balanceOf(address _owner)` function must return the token balance of an account `_owner`.

The `transfer(address _to, uint256 _value)` function must transfer an amount `_value` of points to the account `_to`.

The `Transfer()` event writes a log that will be readable from the etherscan page of the corresponding transaction involving.a `transfer(address _to, uint256 _value)` function.

```
contract Owned {
    address public owner;

    function Owned() public {
        owner = msg.sender;
    }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

    function transferOwnership(address newOwner) public onlyOwner {
        owner = newOwner;
    }
}
```

Figure A.3: Owner restrictor code.

A.1.3 Owner restrictor

Basically, the restrictor code (shown in figure A.3) is used to: restrict the usage of some functions to the owner of the contract, with `onlyOwner`; makes the deployer of the contract the current owner, with the constructor `Owned()`.

Furthermore, only the owner can decide to transfer his/her contract ownership to another account `newOwner` invoking the function `transferOwnership(address newOwner)`.

A.1.4 FidelityPoints contract

The contract is defined with the following code:

```
contract FidelityPoints is IERC20, Owned
```

and after the definition it calls the library SafeMath as follow:

```
using SafeMath for uint256
```

With these two lines of code the contract is declared importing also the token interface, the owner restrictor and the library SafeMath. Moreover, in the contract are also defined other modifiers to restrict the access of some functions to users and shops. Remember that the owner, which is the admin, is also a shop because the service provided have its own page to sell products to the users.

The contract manages several modules which can be divided in: Token operations, Ethereum payment request, buying a product request.

Token creation

First of all, for creating an ERC20 token the following variables must be declared:

```
string public constant symbol = "FID";
string public constant name = "Fido Coin";
uint8 public constant decimals = 18;
```

```
uint256 public constant RATE = 1000000000000000000;
uint public constant INITIAL_SUPPLY = 1000000000000;
uint public _totalSupply = 0;
mapping (address => uint256) public balances;
```

Starting from the top to bottom is possible to see: the token symbol definition; the token complete name; the number of decimals of the token; the exchange rate with one ether; the initial supply of tokens, the variable which will contains the total amount of token generated from the the contract creation; and for last, the mapping between the Ethereum address and its balance.

After the token parameters definition, the functions defined in the IERC20 interface have to be implemented (see section A.1.2). The functions `totalSupply()` and `balanceOf(address _account)` just returns the value of the variables `_totalSupply` and `balances[_account]`, like two standard getters.

The `event Transfer()` implementation is easily as the declaration in the interface:

```
event Transfer(address indexed _from, address indexed _to, uint256 _value)
```

It simply print this values in the log of the transaction.

More interesting is the `transfer(address _to, uint256 _value)` function, implemented in the figure A.4, which works in the following way:

1. check if the sender has enough;
2. check if the amount transferred is greater than 0;
3. prevent transfer to 0x0 address;
4. check for overflows;
5. check for underflows;
6. subtract the token amount from the sender;
7. add the token amount to the recipient;
8. emit the Transfer event;
9. asserts that are used to use static analysis to find bugs in code. They should never fail.

Ethereum payment request

The figure A.5 shows the structure of a payment request done by a shop asking money from the ISP.

The fields are the following:

- the Ethereum address of the shop, sender of the request;
- the additional notes for the admin;
- the amount of ethereum to be transferred;
- the Id of the shop who is making the request in order to get its data from the database;
- the flag regarding the admin execution of the payment;
- the flag regarding the approbation status of the request.

The Ethereum requested is created through this function:

```
function transfer(address _to, uint256 _value) public returns (bool success) {
        _value = _value * 10 ** uint256(decimals);
        require(balances[msg.sender] >= _value);
        require(_value > 0);
        require(_to != 0x0);
        require(balances[_to] + _value > balances[_to]);
        require(balances[msg.sender] - _value < balances[msg.sender]);
        uint previousBalances = balances[msg.sender].add(balances[_to]);
        balances[msg.sender] = balances[msg.sender].sub(_value);
        balances[_to] = balances[_to].add(_value);
        emit Transfer(msg.sender, _to, _value);
        assert(balances[msg.sender].add(balances[_to]) == previousBalances);
        return true;
    }
```

Figure A.4: The `transfer(address _to, uint256 _value)` function code.

```
struct EthereumPaymentRequest {
        address shop;
        string note;
        uint value;
        string shopId;
        bool completed;
        bool rejected;
}
```

Figure A.5: The EthereumPaymentRequest struct.

```
function createEthereumPaymentRequest(uint _value, string _note, string _shopId)
                        public onlyShop returns (bool)
```

This function stores the new Ethereum request in the array of the current Ethereum requests:

```
EthereumPaymentRequest[] public ethereumPaymentRequests
```

The admin can finalise or reject the request and pay the shop invoking these functions:

```
function finalizeRequestEthereum(uint _index) public onlyOwner payable
    function rejectRequestEthereum(uint _index) public onlyOwner
```

With the function of finalisation of the payment, the ether amount is sent and the flag completed is set to true.

Buying request

The figure A.6 shows the structure of a buy request done by a user for buying a product from a shop.

The fields are the following:

- the Ethereum address of the user, sender of the request;
- the Ethereum address of the shop, receiver of the request;

```
struct BuyingRequest {
    address user;
    address shop;
    string product;
    string shopEmail;
    string userId;
    uint value;
    bool shipped;
    bool rejected;
}
```

Figure A.6: The BuyingRequest struct.

- the Id of the product object to be bought;
- the email of the shop, used for displaying the request to its belonging shops;
- the Id of the user, used for showing him every request he has performed;
- the tokens used for buying the product, they are sent with the request;
- the flag regarding the shipment status of the request;
- the flag regarding the approbation status of the request.

The Buying requested is created through this function:

```
function createBuyingRequest(string _product, string _shopEmail, address
 _receiver, uint _value, string _userId) public onlyUser returns (bool)
```

This function stores the new buying request in the array of the current buying requests:

```
BuyingRequest[] public buyingRequests
```

The shop, including the admin, can finalise or reject the request and send the product to the user with these functions:

```
function finalizeUserRequestBuy(uint _index) public onlyShop
 function rejectUserRequestBuy(uint _index) public onlyShop
```

After the function of finalisation the shop have to ship the product or make the service offered available to the user.

A.2 Web3

With the code in the figure A.7, web3.js is imported and linked to the network Rinkeby. First there is a check if Metamask is available or not; with typeof used to see if a variable is defined. If is the variable is defined, the user is in the browser with Metamask is running; otherwise, the user is on the browser or the user is not running Metamask. This configuration is the standard for running web3.

After importing web3, it can be used to interact from the fronted with the Rinkeby testnet, follows an example where the admin finalise an Ethereum payment:

```
import Web3 from 'web3';
let web3;

if (typeof window !== 'undefined' && typeof window.web3 !== 'undefined') {
    web3 = new Web3(window.web3.currentProvider);
} else {
    const provider = new Web3.providers.HttpProvider(
        'https://rinkeby.infura.io/bagPaFcte2OtdWeyGjdB'
    );
    web3 = new Web3(provider);
}
export default web3;
```

Figure A.7: web3.js configuration.

```
const accounts = await web3.eth.getAccounts();
await fidelityPoints.methods.rejectRequestEthereum(this.props.id).send({
          from: accounts[0],
          gas: '4500000'
})
```

As it shown, thanks to web3, the address of the admin is fetched from Metamask calling `accounts[0]` and it is also possible to preset a gasLimit; anyway, the user of the function can always decide to change these parameters.

A.3 Express.js

Express.js is the middleware used for interacting with the PSD2 OBP sandbox.

The file where express.js and the calls to the OBP API are performed is at the following path in the project:

`fidelity-points-system-thesis/psd2/oauth.js`

In the figure A.8 is shown the configuration of the middleware with several modules imported like express.js itself, the session manages of express, oauth authentication, pug for the template etc.. Moreover, the `var consumer` is the configuration of the oauth for the admin of the service. The Key and the secret of the admin are saved in the `config.json` file.

The most relevant function in this file is the `createTransactionRequest` which allows to send the form data, filled with the euro amount, to give the shop the value which asked for the redeeming points.

The code related to this function is displayed in the figure A.9. This is a get function, this means that data are only prepared to be sent. Here first is set the front-end template from where is possible to perform this operation; then, from Firebase the data related to the `pending_psd2_payment` are fetched in order to fill the form statically.

Only after pressing the button "submit" the data will be sent to the endpoint `/createTransactionRequest`, with a post request, to the bank and the transaction will be performed. The post request basically contact the endpoint with the following code:

```
var postUrl = apiHost + "/obp/v2.1.0/banks/" + fromBankId + "/accounts/" +
      fromAccountId + "/" + viewId + "/transaction-request-types/" +
            transactionRequestType + "/transaction-requests"
```

```
var express = require('express', template = require('pug'));
var session = require('express-session');
var util = require('util');
var oauth = require('oauth');
var firebase = require('firebase');
var app = express();
var pug = require('pug');
var config = require('./config.json');
var bodyParser = require('body-parser')
var urlencodedParser = bodyParser.urlencoded({ extended: false })
var _openbankConsumerKey = config.consumerKey;
var _openbankConsumerSecret = config.consumerSecret;
var _openbankRedirectUrl = config.redirectUrl;
var apiHost = config.apiHost;

var consumer = new oauth.OAuth(
    apiHost + '/oauth/initiate',
    apiHost + '/oauth/token',
    _openbankConsumerKey,
    _openbankConsumerSecret,
    '1.0',
    _openbankRedirectUrl,
    'HMAC-SHA1'
);
```

Figure A.8: express.js configuration.

```
app.get('/createTransactionRequest', function(req, res){
    var template = "./template/createTransactionRequest.pug";
    var pid = req.session.pid;
    console.log("SIGNED IN pid", pid);
    if (!firebase.apps.length) {
        firebase.initializeApp(configFirebase);
    }
    var refPsd2Payment = firebase.database().ref("pending_payments_psd2/" +
        req.session.pid);
    refPsd2Payment.once("value").then(function(snapshot) {
        var tokenAmount = snapshot.child("tokenAmount").val();
        var euroAmount = tokenAmount / 1000 * 700;
        var shopId = snapshot.child("shop").val();
        var refShop = firebase.database().ref("shops/" + shopId);
        refShop.once("value").then(function(snapshot) {
            var toBankId = snapshot.child("bankId").val();
            var toAccountId = snapshot.child("accountId").val();
            var options = {
                "title": "Create Transaction",
                "pid": req.session.pid,
                "shopId": shopId,
                "toBankId": toBankId,
                "toAccountId": toAccountId,
                "tokenAmount": tokenAmount,
                "euroAmount": euroAmount,
            };
            var html = pug.renderFile(template, options)
            res.status(200).send(html)
        };
    });
});
```

Figure A.9: Performance of a PSD2 payment calling OBP API.

Appendix B

User manual

This appendix describes: the instructions on how to install the program (section B.1); the graphical interfaces (section B.2); and the input and output data (section B.3) used in the loyalty point system, developed in the thesis project.

B.1 Installation

The project is located in a public repository of my personal GitHub, at the following link: https://github.com/riccardopersiani/fidelity-points-system-thesis.

B.1.1 Download

The first step is to download the project. Open the terminal and through SSH, the most secure method, use the following command in a workspace:

```
git clone git@github.com:riccardopersiani/fidelity-points-system-thesis.git
```

However, also the HTTPS method can be used, using the following command in a workspace (as shown in figure B.1):

```
git clone https://github.com/riccardopersiani/fidelity-points-system-thesis.git
```

Using both the two commands shown, the project will comes up already in its main folder:

```
fidelity-point-system-thesis
```

```
➜ Desktop git clone https://github.com/riccardopersiani/fidelity-points-system-
thesis.git
Cloning into 'fidelity-points-system-thesis'...
remote: Counting objects: 32025, done.
remote: Compressing objects: 100% (22900/22900), done.
remote: Total 32025 (delta 7426), reused 31926 (delta 7331), pack-reused 0
Receiving objects: 100% (32025/32025), 68.33 MiB | 897.00 KiB/s, done.
Resolving deltas: 100% (7426/7426), done.
```

Figure B.1: Cloning the project from the GitHub repository.

After downloading the project, it will be possible to read it and to work with it, through one of the text editor such as VSCode, Atom, IntelliJ or whatever. Personally, I advice for VSCode

usage, which allows to install several add-ons that are really useful for managing the project and programming in Solidity and Javascript. VSCode will be assumed as default for the following instructions, but other editors are fine as well.

Here is the link for downloading VSCode: https://code.visualstudio.com/download.

To open the project with VSCode, start the terminal inside the main folder, which is the project folder, such as:

```
/Users/"username"/Desktop/fidelity-points-system-thesis
```

Then, to open the entire project, type the following command :

```
code .
```

From the terminal, typing `ls`, will be possible to check if the folders inside are the ones showed in the figure B.2.

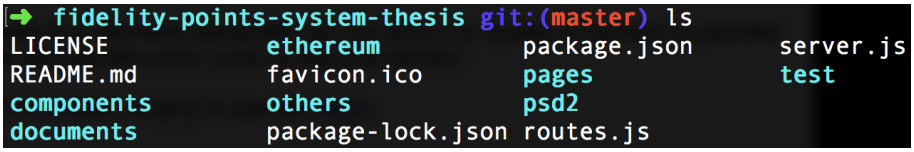

Figure B.2: Project folders and files.

It must be noticed that in this project is required the installation of "npm" (node package manager) and "node"; however, npm is distributed with Node.js, which means that when node.js is downloaded, automatically includes npm installed. The project uses the npm version 6.1.0 and node v8.9.4, but also the latest releases should not give problems. Download node.js at: https://nodejs.org/en/.

Furthermore, Firebase is already configured in the following main component:

```
fidelity-point-system-thesis\components\template\header.js
```

The configuration code is outlined in the figure B.3, all the parameters are provided in the moment of the registration on the platform.

Metamask

Metamask must be installed on the browser, in the testing phase of the project the Chrome browser was used. Metamask can be downloaded at the following link: https://metamask.io/. After adding the extension, and configuring the initial password, the Rinkeby network must be

```
var config = {
      apiKey: "AIzaSyB7-H-6t5kb5D8XB9jf33SVkpjgmeJqATg",
      authDomain: "test-3ff4d.firebaseapp.com",
      databaseURL: "https://test-3ff4d.firebaseio.com",
      projectId: "test-3ff4d",
      storageBucket: "test-3ff4d.appspot.com",
      messagingSenderId: "1059441748413"
    };
```

Figure B.3: Firebase configuration.

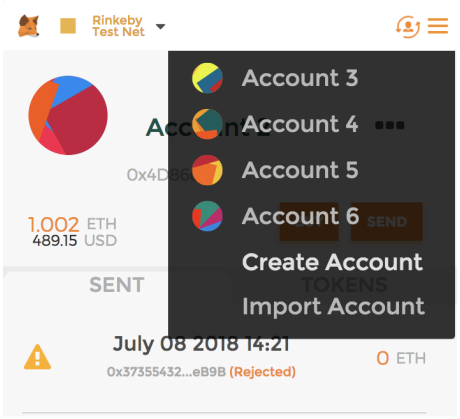

Figure B.4: Metamask account creation.

selected. In addiction accounts must be created, clicking in the icon figuring a person on the right top of the Metamask GUI (as shown in the figure B.4).

Now, the account must be filled with Ethereum, to do that the official procedure is the one described in the link `https://faucet.rinkeby.io/`. To complete the procedure is needed or a Twitter, or a Google+, or a Facebook account.

B.1.2 Build

The second step, after downloading and opening the project, is the project building. First of all, all the node packages must be installed; as a result, the following command mast be typed in the main folder:

```
npm install
```

Then the Ethereum contracts needs to be compiled with the following command starting from the main folder:

```
cd ethereum
node compile.js
```

Moreover the building of the next.js server is needed; as a result need to be typed from the main folder, the command:

```
node server.js
```

Note, that the latest command described also launch the next.js server.

B.1.3 Start

After the correct building of the project, the third and last phase is the launch of the two servers.

Start the next.js server from the main folder using the following command (do not do this, if already done in the building phase):

```
node server.js
```

Start the express.js server from the main folder psd2 folder using:

```
cd psd2
node oauth.js
```

The frontend application for the loyalty point system is reachable at `http://localhost:3000/`. The frontend application for the psd2 payment is reachable at `http://localhost:8085/`, but this require the authentication to the OBP PSD2 Sandbox.

B.2 Graphical Interface

This section provides an idea of the multiple interactions with the loyalty point system through its interfaces of the three entities: users, shops, and the admin. However, before the login and even after some pages are shared and are the same for every entities.

B.2.1 Without login

In the figure B.5 is shown the home page, which provides a description about the project.

In the figure B.6 is shown the statistic page, which provides some information and stats about the loyalty point system, such as the owner contract and the name of the token.

In the figure B.7 is shown the user signup page.

In the figure B.8 is shown the shop signup page.

The admin comes out already registered, potentially during an eventual setup phase of the loyalty program application.

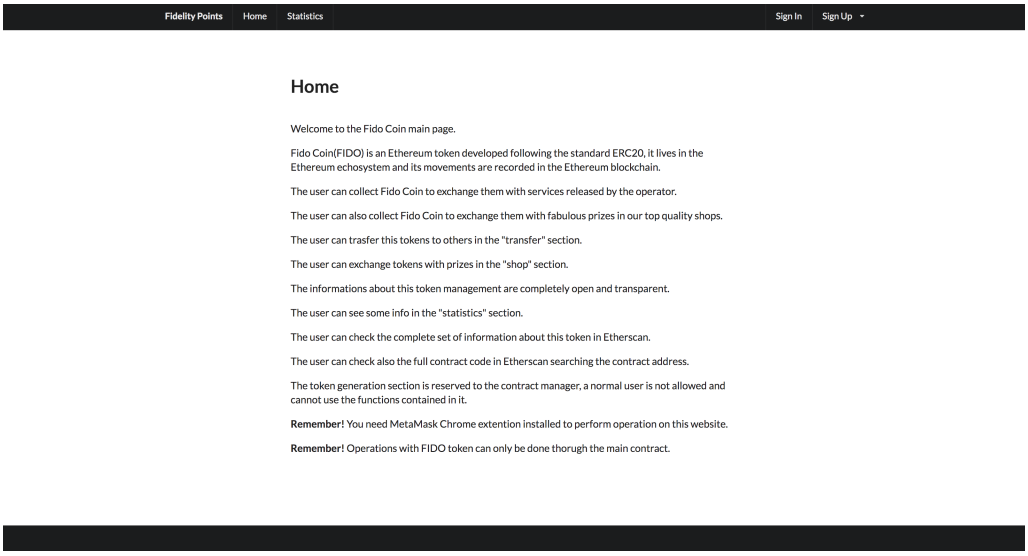

Figure B.5: Home page.

B.2.2 User

Is possible to signin as one predefined user with:

`Email`: user@user.com
`Password`: 11111111

To be clear, the password is eight times the number 1. it is important to understand that is possible to access to the front-end with this credentials; but, the operations performed must be done with his/her account, which is unique and available in my own instance of metamask, with the address:

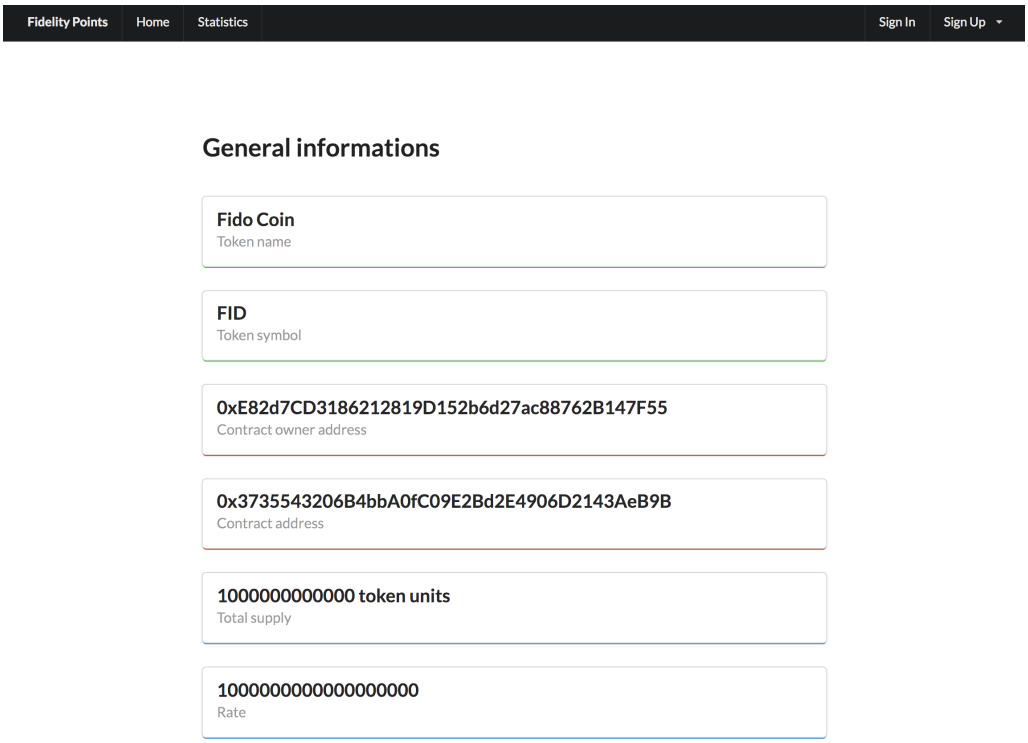

Figure B.6: Stats page.

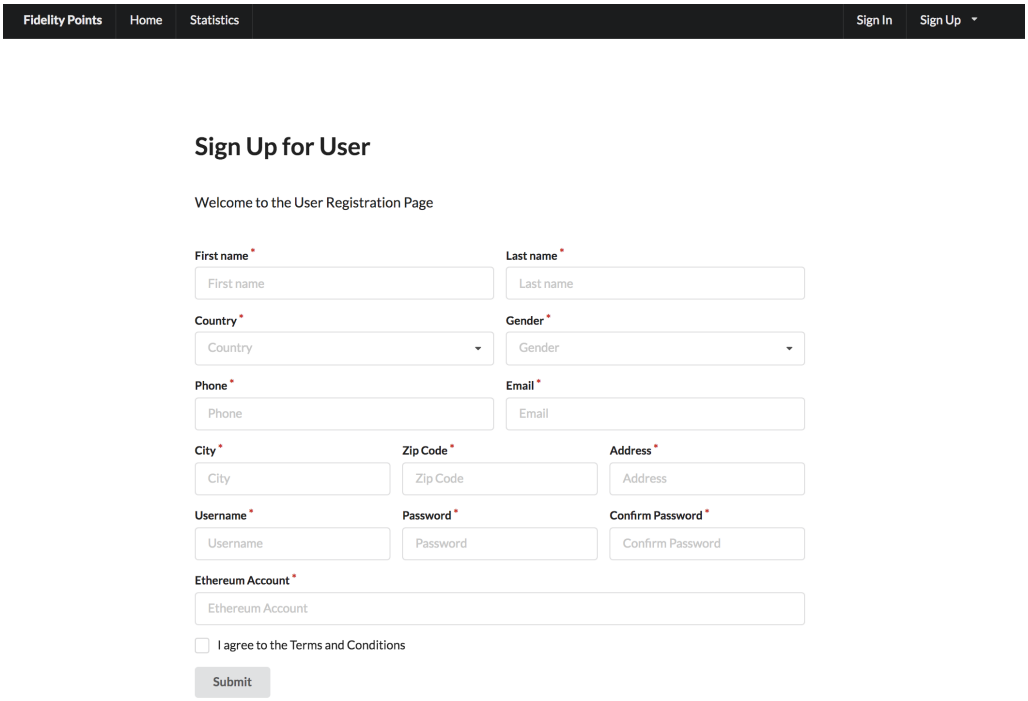

Figure B.7: User SignUp page.

`0x4D86c35fdC080Ce449E89C6BC058E6cc4a4D49A6`

As a consequence, operation performed with other accounts will result useless; this is valid also for shops and the admin. So in order to test it, a new signup is required with the personal address generated on Metamask.

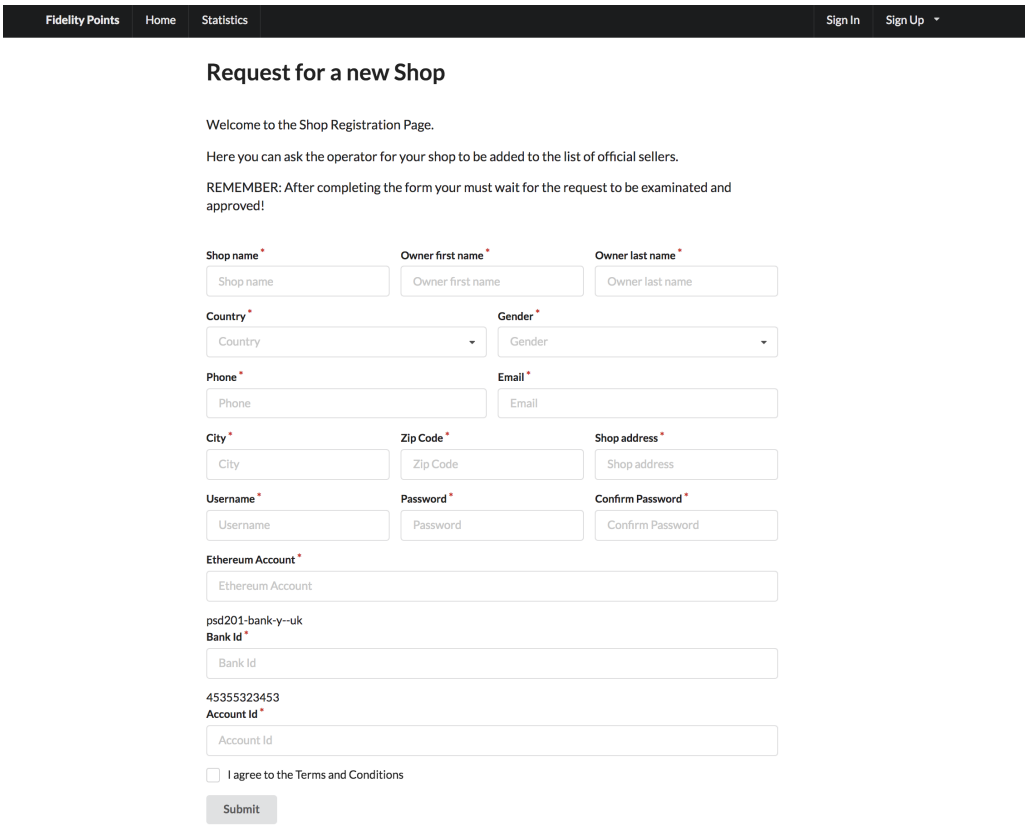

Figure B.8: Shop SignUp page.

The most important operation that the user can perform is the buy of a product with the points collected. An example of a user buying a product from the service provider page is shown in the figure B.9. Please, notice that Metamask is fundamental to buy the product, and the user must use his/her Ethereum account which typed to register with.

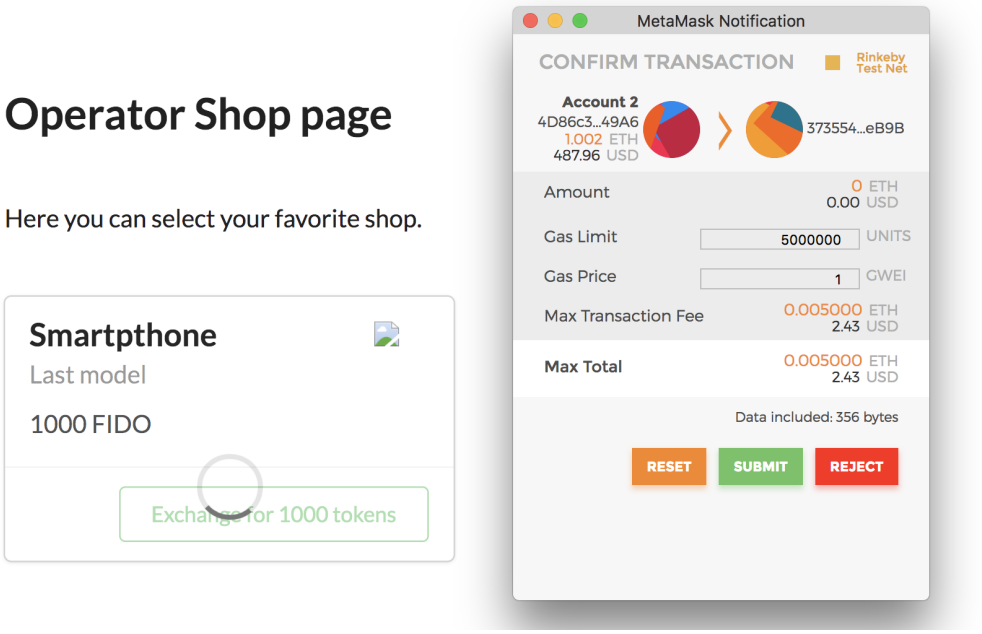

Figure B.9: User buy page.

User in his/her front-end can check the complete, pending and rejected orders (see figure B.10) or transfer some points to whoever he/she wants.

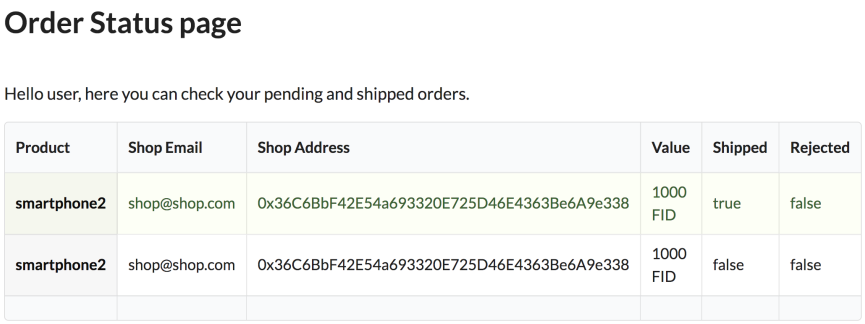

Figure B.10: User check order status page.

B.2.3 Shop

Is possible to signin as one predefined shop with:

`Email`: shop@shop.com
`Password`: 11111111

The most important function performable by the shop is the ask payment function, where it have to send to the admin the amount of points it wants to redeem and choose the method thought it want to be payed with; see the figure B.11.

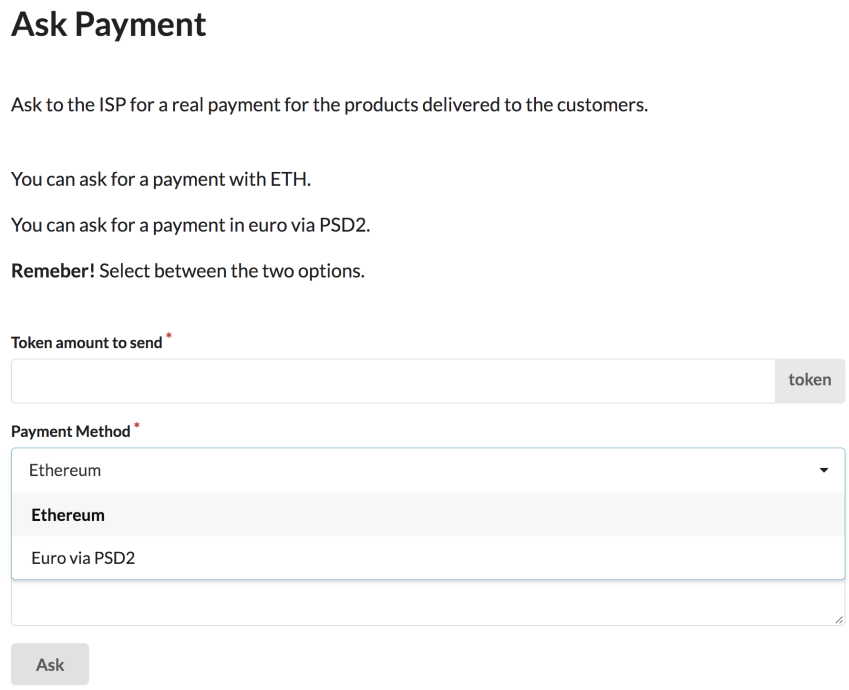

Figure B.11: Shop ask payment page.

Then, there are other pages which the shop can use are the ones to complete orders asked by the users, transfer points, and check the request status of the payment asked.

B.2.4 Admin

Is possible to signin as the admin with:

`Email`: r.persiani92@gmail.com
`Password`: 11111111

One of the most important function of the admin is the token generation where ether can be converted to tokens. To call that function he/she has to go to the page showed in the figure B.12.

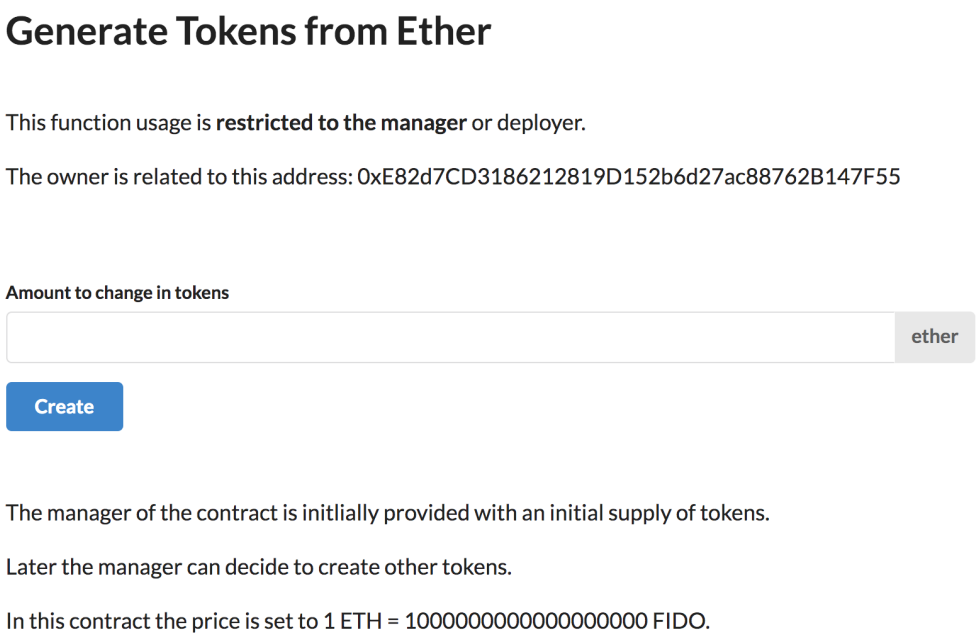

Figure B.12: Admin token generation page.

Then, the other pages which the admin can use are the ones to approve shops request, approve payments, complete orders, and transfer points.

B.3 Data

In this section the data contained in the firebase database are displayed.

B.3.1 Firebase

The firebase database is composed by three tables:

- the registered users (which includes also the admin), see figure B.13;
- the shops registered, see figure B.14;
- the pending PSD2 payments, see figure B.15.

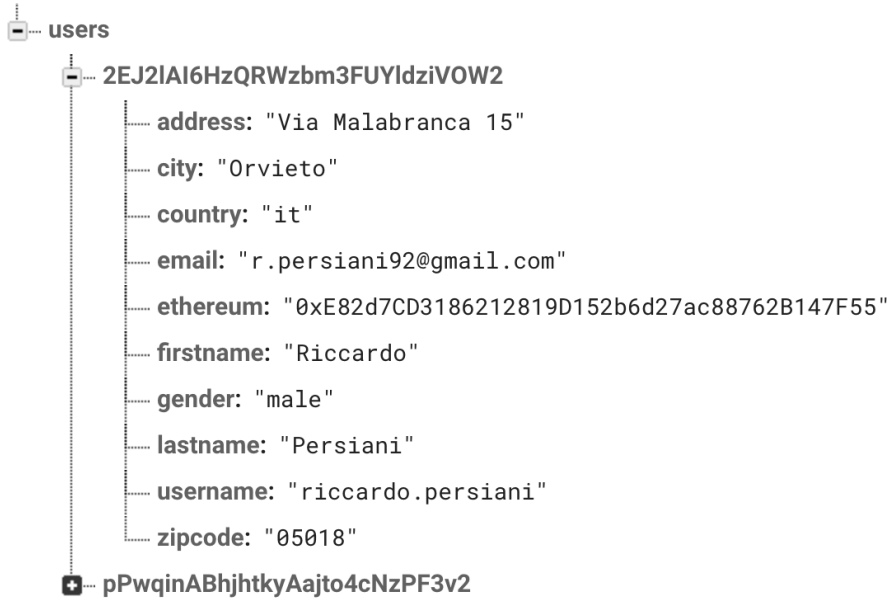

Figure B.13: Users registered table structure

Figure B.14: Shops registered table structure.

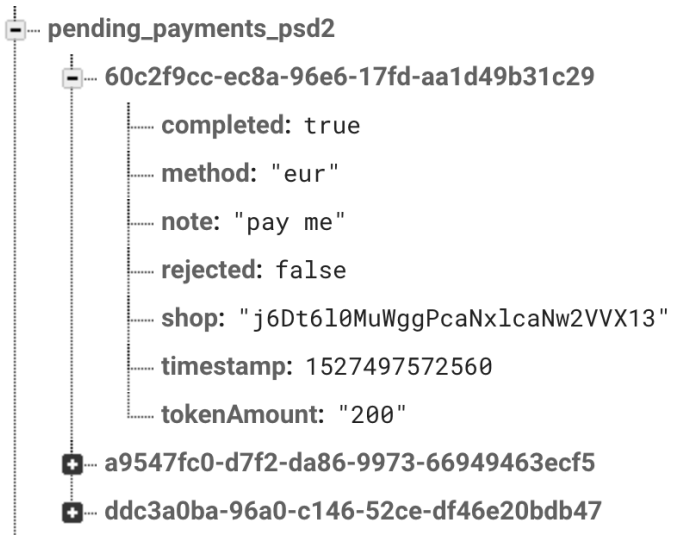

Figure B.15: Pending PSD2 payments table structure.

Bibliography

[1] Vetrya S.p.A. website, http://www.vetrya.com/

[2] D.Drescher, "Verifying and adding transactions", Blockchain Basics: A Non-Technical Introduction in 25 Steps (T.Green and S.McDermott, eds.), pp. 159–162, Apress, 2017, DOI 10.1007/978-1-4842-2604-9

[3] Eight Blockchain platforms for rapid prototyping, http://radiostud.io/eight-blockchain-platforms-comparison/

[4] The Bitcoin project, https://bitcoin.org/en/

[5] R.Lai and L.Chuen, "Blockchain – from public to private", Handbook of Blockchain, Digital Finance, and Inclusion, Volume 2 (L.Chuen and R.Deng, eds.), pp. 145–177, Academic Press, 2017, DOI 10.1016/B978-0-12-812282-2.00007-3

[6] The Ethereum project, https://www.ethereum.org/

[7] The Hyperledger project, https://www.ibm.com/blockchain/hyperledger.html

[8] The Corda project, https://chain.com/

[9] The IOTA project, https://iota.org/

[10] The MultiChain project, https://www.multichain.com/

[11] The OpenChain project, https://www.openchainproject.org/

[12] The Chain project, https://chain.com/

[13] The HydraChain project, https://github.com/HydraChain/hydrachain

[14] Public and private blockchains, https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

[15] Blockchain Finance, https://www.slideshare.net/rmsams/blockchain-finance

[16] G.W.Peters and E.Panayi, "Understanding Modern Banking Ledgers Through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money", November 2015, pp. 4–6, DOI http://dx.doi.org/10.2139/ssrn.2692487

[17] PSD2 Directive, https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32015L2366&from=EN

[18] Euro Lex Europa, https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=LEGISSUM:2404020302_1&from=EN&isLegissum=true

[19] Deloitte, https://www2.deloitte.com/content/dam/Deloitte/cz/Documents/financial-services/cz-open-banking-and-psd2.pdf

[20] Pwc, pillola di PSD2 n.3, https://www.pwc.com/it/it/industries/banking/assets/docs/psd2-pillola-n03.pdf

[21] A short introduction to PSD2, https://hernaes.com/2016/08/25/a-short-introduction-to-psd-2/

[22] Guidance for implementation of the revised Payment Services Directive, https://financedocbox.com/Credit_and_Debt_and_Loans/72599769-Guidance-for-implementation-of-the-revised-payment-services-directive-psd2-guida
html

[23] Pillole di PSD2, https://www.pwc.com/it/it/industries/banking-capital-markets/psd2.html

[24] Regulatory Technical Standards on strong customer authentication and secure communication under PSD2, https://www.eba.europa.eu/regulation-and-policy/payment-services-and-electronic-money/regulatory-technical-standards-on-strong-customer-authentication-and-secure-communication
-/regulatory-activity/press-release

[25] PSD2 in a nutshell 2, https://www.pwc.com/cz/en/bankovnictvi/assets/psd2-nutshell-n02-en.pdf
[26] PSD2: time to open and secure APIs and rethink business models, https://www.computerweekly.com/opinion/PSD2-time-to-open-and-secure-APIs-and-rethink-business-models
[27] PWC Strong Auth, https://www.pwc.com/it/it/industries/banking-capital-markets/StrongAuth.html
[28] Ethereum Wiki, https://github.com/ethereum/wiki/wiki/Ethereum-introduction
[29] Ethereum white paper, https://github.com/ethereum/wiki/wiki/White-Paper
[30] The Mist browser project, https://github.com/ethereum/mist
[31] C. Dannen, "Introducing ethereum and solidity: Foundations of cryptocurrency and blockchain programming for beginners", Apress, 2017, ISBN: 978-1-4842-2534-9
[32] The Etherscan project, https://etherscan.io/
[33] Ethereum yellow paper, https://ethereum.github.io/yellowpaper/paper.pdf
[34] The Ethereum gas costs, https://docs.google.com/spreadsheets/d/1n6mRqkBz3iWcOlRem_mO09GtSKEKrAsfO7Frgx18pNU/edit#gid=0
[35] The Ethereum Gas Station project, https://ethgasstation.info/
[36] The Ethereum EVM implementations, https://github.com/ethereum/wiki/wiki/Ethereum-Virtual-Machine-(EVM)-Awesome-List#evm-implementations
[37] The Solidity Documentation, https://solidity.readthedocs.io/en/v0.4.24/
[38] The Solidity Documentation, https://solidity.readthedocs.io/en/v0.4.24/types.html
[39] The Open Bank Project website, https://openbankproject.com/
[40] The Open Bank Project website, https://openbankproject.com/for-customers/
[41] The Open Bank Project website, https://openbankproject.com/for-banks/
[42] The Open Bank Project website, https://openbankproject.com/for-developers/
[43] The Open Bank Project PSD2 sandbox, https://psd2-api.openbankproject.com/
[44] OBP API Explorer PSD2, https://psd2-apiexplorer.openbankproject.com/?version=2.0.0&psd2=true?ignoredefcat=true&tags=Account
[45] B.Sharp and A.Sharp, "Loyalty Programs and Their on Repeat-Purchase Loyalty Patterns", International Journal of Research in Marketing, vol. 14, December 1997, pp. 436–486, DOI 10.1016/S0167-8116(97)00022-0
[46] Glossary L, Loyalty Program, https://www.elect-mer.com/glossary-l.html
[47] Deloitte UK consumer review customer loyalty, https://www2.deloitte.com/content/dam/Deloitte/uk/Documents/consumer-business/deloitte-uk-consumer-review-customer-loyalty.pdf
[48] Deloitte making blockchain real customer loyalty rewards programs, https://www2.deloitte.com/us/en/pages/financial-services/articles/making-blockchain-real-customer-loyalty-rewards-programs.html
[49] The React project, https://reactjs.org/
[50] The web3.js library documentation, https://web3js.readthedocs.io/en/1.0/getting-started.html
[51] The express.js project, https://expressjs.com/
[52] Get Started with Firebase Authentication on Websites, https://firebase.google.com/docs/auth/web/start
[53] Firebase Realtime Database, https://firebase.google.com/docs/database/
[54] The Pug project, https://pugjs.org/api/getting-started.html
[55] The Metamask project, https://metamask.io/
[56] Metamask GitHub repository, https://github.com/MetaMask/metamask-plugin
[57] The Remix documentation, https://remix.readthedocs.io/en/latest/
[58] A. Elbahrawy, L. Alessandretti, A. Kandler, R. Pastor-Satorras, and A. Baronchelli, "Evolutionary dynamics of the cryptocurrency marke", Royal Society Open Science, vol. 4, May 2017, DOI 10.1098/rsos.170623
[59] The Ethereum tokens, https://www.ethereum.org/token
[60] G. Fenu, L. Marchesi, M. Marchesi, and R. Tonelli, "The ico phenomenon and its relationships with ethereum smart contract environment", 2018 International Workshop on Blockchain Oriented Software Engineering, Campobasso (Italy), March 20-22, 2018, pp. 26–32, DOI 10.1109/IWBOSE.2018.8327568

[61] P. Schueffel, "The concise fintech compendium", School of Management Fribourg, 2017
[62] The PayPal project, https://www.paypal.com/it/webapps/mpp/home
[63] The PayPal fees, https://www.paypal.com/it/webapps/mpp/paypal-fees