UNIVERSITY OF PISA
Department of Computer Science
Master's degree in Computer Science
ICT Solutions Architect

THESIS

Decentralized accounting system:

Efficient document management system implemented on top of blockchain secure transactions environment

SUPERVISOR

Prof. Laura Ricci

EXTERNAL SUPERVISOR

Prof. Giacomo Poretti (SUPSI)

CANDIDATE:

Vincenzo Di Perna

Matr. 606933

Academic Year 2021/2022

"Until you make the unconscious conscious,
it will direct your life
and you will call it fate"
~ C.G. Jung

Preface

Passing in front of the window of a restaurant projects us into a scenario made of romantic dinners, friendship, or work, in a welcoming environment that gives emotions through the pleasure of enjoying good food. However, it is difficult to perceive what is happening "behind the scenes", the need for coordination between employees, orders, payments, and the warehouse. This is what a restaurateur has to manage to make sure that the company will be as efficient as possible. Externally the company is not possible for the restaurateur to have a part in the accounting process, which is forced to integrate a specific system, often managed by third parties. These systems are often antiquated and almost manual, especially during the revision phase (invoices). It may happen that the operator at the checkout has typed the wrong amount for the customer's payment, producing an incompatibility between the order received and the payment made. Or even other operations such as the reversal, or the “simple finding” of the necessary invoices, can lead to an inefficiency in the activity.

It has happened to me, on several occasions, to be involved in environments related to the Catering and Bathing sectors, from where we had the opportunity to analyze mainly the service mechanisms offered and the coordination between the parties, rather than the accounting aspects. Here, we wondered how this environment could have been made better - staff always in a hurry, queues to grab a place, and still lines for payments at the checkout. In short, we noticed the need for synchrony between these parts. This becomes more understandable if we consider that the problem of accounting systems can affect up to 30% of the revenues of an activity. Revenues that if better re-distributed can have a positive impact on innovation and the livelihood of the company's internal activities.

The aim is to make accounting a more efficient mechanism, but above all less manual and subject to continuous revisions. The introduction of the Blockchain[1] mechanism is certainly one of the first steps to be activated as its characteristics of a decentralized system, transparency, and security, would make payments safer and transactions available in a more direct, simple, and automated, avoiding manual revisions. At the same time, there is also a need

for privacy due to the presence of accounts, amounts, and customer data. Of course, we could continue to click "I agree", but it is increasingly evident that the information lives in an almost borderless context. In addition, delegating the custody of this information to third parties seems that also make it difficult to find, even by the owner himself! Hence, we express the need to be able to have our storage for our data, which in the case of the restaurant we can identify as invoices, active orders, supplier data, etc... Access protection allows us to authorize only those we want. This can have a huge advantage for the accounting system that will no longer have to be forced to contact the restaurateur (or third-party mechanisms) to find accounting information, but the restaurateur himself can guarantee direct access to this information.

Drawing inspiration from these activities, the purposes are precisely to improve an accounting system that is now unsuitable and take advantage of the profit re-generated, distributing it among the various internal activities of the company that can be the recruitment of new staff or the purchase of new work material. Above all, do not incur situations of tax offense due to "missing" items in an invoice or typing errors, which must be remembered to be human error and therefore as safe as it is predictable.

Index

1 Introduction 1

1.1 Problem report 1

1.2 Scenario analysis 3

1.3 Thesis's structure 5

2 Background 7

2.1 Concepts evolution 7

2.1.1 Blockchain 8

2.1.1.1 Ethereum 10

2.1.2 Solid Library 12

2.1.3 Web3 16

2.1.4 Coin Wallet 20

2.1.5 ERP System. 23

2.1.6 eBill 25

2.1.7 Stable coin 29

2.2 Related Work & Literature. 30

2.2.1 Multi Agent System. 30

2.2.2 Decentralized Storage. 32

2.2.2.1 Solid context. 32

2.2.3 Blockchained ERP system 33

2.2.4 Blockchain privacy transactions. 33

2.2.5 Blockchain notifications. 34

3 Proposal 36

3.1 Use-case: Restaurant 36

3.1.1 Intervention. 42

3.2 Development 43

3.2.1 Environment & Implementation 43

3.2.2 Server Backend. 46
3.2.3 Solid Pods. 48
3.2.4 WebApp. 55
3.2.5 Blockchain system. 59
3.2.6 MetaMask. 64
3.2.7 QR-code system. 66
3.2.8 eBill system.. 69

4 Conclusions 73
4.1 Considerations 73
4.2 Perspectives 74
4.3 Engagement 76

4 Acknowledgments 77

Bibliography & Sitography 78

Appendix A – Infrastructure directories.. 82
Appendix B – Smart-contract 83
Appendix C – Solid Interface 84

List of Figures

2.1 Blockchain linked blocks……..……...……...……...9
2.2 Solid's Architecture……..……...……...……...15
2.3 The technological layout of Web 3…..……...……...……...18
2.4 Three Layers of Webpage Annotations……...……..19
2.5 ERP domain.....……...23
2.6 Payment network process flow control……...……25
2.7 Various invoice formats graph……...……26
2.8 eBill customer execution flow……...……..27
2.9 Overview of the ranking from the perspective of the invoice recipient/issuer..……28
2.10 Automatic insertion of additional costs (e.g. taxes).. ……31
2.11 Example of incoming Request.....……31

3.1 General restaurant incoming/outgoing flow …….....37
3.2 Restaurant's steps timeline.....……..…….....38
3.3 Infrastructure Components…...……...…….....……..…….....44
3.4 Infrastructure Layers…...……..…….....45
3.5 e-Wallet-Blockchain structure…..…….. …...……..…….....46
3.6 generate-oidc-token utility…….....…….....50
3.7 Solid graph interactions…..…….....…….....55
3.8 Menu & Cart switches…….....…….....56
3.9 List of products……...…….....…….....56
3.10 Product Description...…….....…….....57
3.11 Quantities Update…...…….....…….....57
3.12 Get Bill phase……..…….....…….....58
3.13 Billing phase……...…….....…….....58
3.14 MetaMask experience…….....…….....65
3.15 Table Identification through the QR-code.…….....67
3.16 QR-code on a digital bill…….....…….....68
3.17 Pre-invoice……..…….....…….....70

List of Tables

2.1 Comparisons of Web versions.....18

Acronyms

ABI - Application Binary Interface. 59, 61, 63

ACP – Access Control Policy. 15, 49, 50

ACR – Access Control Resources. 15, 48

CBDC - Central Bank Digital Currency. 28

CRM – Customer Relationship Management. 16

dApp - Decentralized App. 20, 21, 57

DeFi - Decentralized Finance. 33

ERP - Enterprise Resources Planning. 2, 4, 7, 22, 23, 24, 32, 51, 72, 73

EVM - Ethereum Virtual Machine. 11, 12

IPFS - International Planetary File System. 32

LDP – Linked Data Platform. 15

MAS - Multi-agent System. 29, 30

NLP - Natural Language Processing. 18

P2P - Peer to peer. 8, 29, 31, 40, 72

PoW - Proof of Work. 9, 62

RDF – Resource Description Framework. 14, 17, 48

SHA - Secure Hash Algorithm. 31

SIX – Swiss Exchange. 26

SME - Small & Medium enterprises. 5, 24, 40, 71, 72

WAC – Web Access Control. 15

WOL - Web Ontology Language. 17

Abstract

The accounting system of a company is a necessary resource, which now takes on more and more depth in SMEs. In this regard, it is required that these systems (often still manual) combine less and less impact on costs, productivity, and efficiency of the activity. Accounting is one of the items at the top of the list of recorded expenses, going to cancel a good part of the margins produced during the financial window. According to one estimate, 20 to 30% of an SME's revenue is drained for accounting. Other problems of a practical nature surround the accounting system such as the compliance of invoices, their values, the use (often) of prehistoric systems that require many manual interventions, etc. It must be remembered that totalitarian trust must be placed in an accounting system. Consequently, an error of any kind may result in losses, fines, or serious taxes on the company. Considering all these facets, the proposal that we wanted to advance through this thesis was the implementation of a system that manages and automates accounting procedures within the company, allowing recovery and re-use of margins to promote sustainability and renewal of the company. In particular, attention has been paid to transparency, security, and reliability through the integration of the Ethereum Blockchain. Depth and importance have also been given to data privacy through Solid and the concept of Pod, a promising method of storing data on the network that uses access rules and authorization mechanisms to dialogue and find information in the Pod. The use-case considered concerns a Restaurant. Being synonymous with quality and the tech movement, Switzerland is referred to as the next Silicon Valley for crypto and blockchain. The thesis' work is then only a draft of a larger design involving real companies, consultants, and Swiss innovation bodies (InnoSwisse).

Chapter I

INTRODUCTION

1.1 Problem report

From left to right in just a few comments:

- *"There was a time, a restaurant in that alley, where I got to taste some delicious desserts"*
- *"…we had a wonderful evening, great and nice food, quality wine list and very nice staff…"*
- *"…Luckily, we had some cash, but we would have preferred to pay by card when we were told that they take cards."*
- *"It will take at least 20 minutes, maybe even more, and anyway there are other tables before you"*
- *"…several times during dinner I had to get up to ask for something…"*

For a restaurateur, the need to keep their business in line with the times is an indissoluble request. In particular, the above example to which we wanted to mention, highlights some of the needs through the review mechanism activated by TripAdvisor platform, which acts for restaurateurs both as a showcase and as a tool for comparison with customers. And this comparison is important because it is a part of the renewal process, thanks above all to the feedback released by customers, which highlights their strengths and weaknesses. It is necessary to study these points, understand the amount of discomfort, and design an efficient and inexpensive solution.

With these first words, we tried to introduce the reader to the concept of necessity. A need that pushes us to take care of and safeguard our efforts, our investments. To ensure that our ideas can last over time. A family-run company, which persists over the years, provides a mark of guarantee and quality, also thanks to this longevity of the service. Undoubtedly, it provides details on the renewal process with which the owner has to deal with (for years) to give always new light to the business.

The work of the Thesis has been allowed us to analyze the reflections of a restaurateur regarding the activity. In particular, they highlighted the following "shortcomings":

- *reduction of administrative costs*,

  constantly increased due to new laws or tax impositions. However, it must be said that these costs are mainly related to accounting needs. The restaurateur, with very accurate analysis, was able to quantify this expense around 20-30% of his income. Regardless of whether this aspect is accompanied by the hiring of a physical accountant, or for the subscription to an Enterprise Resource Planning service, it is still a point on which to work and obtain improvements. For companies with a background already started, the latest solution (ERP) is considered a must-have (as well as complex system), and the costs tend more towards the maximum value (30%).

- *inconsistencies between invoices*,

  where the obligation of uniformity should reign. Whether it is a typo by the cashier, by the administrator, or related to a bug in the software, it does not matter. The conclusion is that the error may exist. However, this error may result (in the worst case) in the payment of a fine due to the untruthful declaration of the amount of an invoice. This inconsistency must not exist at the time of entry into the accounts by the administrator, but above all at the time of verification by the authority.

- *transparency*,

  in a context where legality is a must. Without doubt, the transparency in accounting and non-accounting operations guarantees greater control by the state bodies used, greater correctness and verifiability, as well as the impossibility of denying the occurrence of events within the activity. Often, this is not yet

guaranteed because these actions are implicit in the activity itself, and do not always require pro-transparency adaptation.

- *decentralized documents*,

  in a context where the centrality of the flow of documents, information, invoices, payments, continues to be the bottleneck that does not allow the creation of a referential network within the company. A decentralization mechanism would allow the distribution of the business flow between several parties, ensuring the availability of information, uniquely, from each of these.

- *cash removal*,

  mainly due to the entry of both new payment methods and new forms of currency. In addition, the presence of physical cash has different implications for the business and customers, e.g. the physical presence of cash in the cash register that can be the cause of theft, or the usability and therefore the limitation of physical cash for a customer who may not have the right amount at the time of payment.

All these aspects make a strong reference to a process of digitalization that should be started to make the current activity updated infrastructurally to today's needs. Not surprisingly, the thesis offers to prototype an infrastructure that can subvert the problems discussed.

1.2 Scenario analysis

The previous highlighted points summarize a scenario with a strong need for adaptation to increasingly cutting-edge digital techniques that will allow us to obtain a completely renewed infrastructure ready for these new needs. As an advance on the chapter of the technological background, we will introduce some of the technologies that will later be better framed in our context. With this regard, we want to introduce the following entities:

- ***Blockchain***,

  a mechanism that allows multiple users to verify and confirm the information shared within a network. This shifts the focus from a centralized context to a permissionless one, where the strength is precisely the network of nodes that

constitutes it. Relative to the information it can handle, it is authenticated and distributed across all nodes.

- ***Solid library***,

  a new technological reference that returns to the main user the control over their activities and information, going to abstract the latter from the various Web platforms where they previously resided. With the implementation of this library, we aim to be able to keep restaurant information such as invoices, daily menus, etc. in a secure context, and record only a small part of these on the blockchain.

- ***ERP systems***,

  allows the centralized management of enterprise's activities and provides an excellent environment for the management of accounting, thanks also to the fact that the system integrates the information coming from the different sections of the company and from which we can make decisions based on evaluations, merging and filtering on this information. About the accounting aspect, it is immediately evident that such a system subverts the problems of uniformity related to a previous manual accounting, or the manual retrieval of the related information.

- ***Web3***,

  the new frontier of the Web that exploits a more accurate semantics of the information to enable the machine to understand and deal more naturally with this information, allowing it to guide the new user experience on the network. It enables the web to the concept of decentralization and gives access to increasingly personalized content.

- ***MetaMask***,

  a virtual wallet that enables the restaurateur, and therefore the customers, to make cashless payments. MetaMask, in particular, allows us to select the network on which to carry out transactions and to choose the currency among those offered by the service itself. Intrinsically to its concept of a web wallet, it introduces the concept of cryptocurrencies such as Ethereum.

1.3 Thesis's structure

The main goal of the Thesis will be to activate a mechanism of decentralized management of accounting documents within a context such as a restaurant. For this, the implementation of a blockchain and a library for document management will be required. In the end, further particularities will be introduced that could serve as a reflection for future developments. Then, the structure of the thesis will follow this path:

- *Chapter II*:

  development of the state of the art through a sequence of steps that show the evolution of an accounting system for an SME; overview of the general form of the blockchain mechanism before the illustration of the various methodologies and techniques applied to it in the last decade; focus on the innovations introduced by the latest studies.

- *Chapter III*:

  considerations on the implementation choice, also based on the various cases present in the collected material, and evaluations for the various phases that make up the decentralization process; description of the practical implementation, and the steps performed for the execution of the application. All concerning the use-case of a restaurant. In this regard, the implicit activities at the restaurant will be analyzed in detail, then the reflections achieved.

- *Chapter IV*:

  final considerations on the topic of blockchain, on Solid and therefore of the thesis work. Notes on the prospects regarding the process of decentralizing an accounting system and the phases that compose it.

Chapter II

BACKGROUND

"To ask the right question is already half the solution of a problem."

~ C.G. Jung

The chapter shows the conceptual and technical background needed. This will introduce us to the context of a restaurant, whose proposal is to try to adapt its core business towards a document-management system on blockchain that facilitates the mechanisms of billing management by an ERP system. The focus will be introduced on the novelties of the latest studies, then the state of the art related to the most used techniques for this context.

2.1 Concepts evolution

The path for the Thesis development was characterized by different aspects, linked between them, which allowed us to faithfully represent the use-case of a restaurant. Although the focal point of the design was the integration of the blockchain, importance will be given to the Solid library too. The latter, more recently, required a huge amount of work to learn its concepts and functionalities. Further space was given to different design components such as the QR code, the Wallet, Web-server, and more, necessary to prototype a digital infrastructure for the services offered by the restaurateur. Another aspect considered was the ERP system to which the restaurateur makes faith to take care of his administrative and accounting part. From here, the main analysis for the reduction of relative cost started. The characterization and study of other related works have allowed us both to understand new concepts and implementations types, identifying further points of disadvantage or re-design.

2.1.1 Blockchain

When we talk about blockchain, the term is often compared to Bitcoin cryptocurrency and its use. However, it is not correct to generalize in this way, and therefore we need to provide more details. Its main description is that it is a distributed ledger that has its data cryptographically signed and grouped in blocks that will be chained by hash pointers pointing to the previous block of data [1]. First of all, the blockchain was born recently as one of the main novelties since the beginning of the 21th century, and it is a mechanism that guarantees transparency, authentication (security for the identity of the seller), integrity (not tamper transactions), and non-repudiation (transactions cannot be denied) making it suitable for record management activities, i.e. financial transactions or identity management [2]. It must be said that a blockchain can be both permissionless (e.g. Bitcoin or Ethereum) or permissioned (e.g. Hyperledger[1]): in the former, anyone can join or leave the blockchain network which may raise security risks; in the last case we have a balancing towards greater controls and protections between the parties, with the presence of nodes with a higher degree of control over the mechanisms of the blockchain, a quasi-decentralization [2]. Key aspects are decentralization and chaining. The first ensures the removal of a single point of coordination to favor the distribution of information among nodes, through a structure similar to P2P. Through the concept of chaining, it was possible to guarantee reliability and integrity on the distribution, communication between multiple nodes, and synchronization of the content of the chain. Not surprisingly, a strong point of the blockchain is precisely the irreversibility of the blocks that characterize it.

The Blockchain allows saving blocks of information on the chain by users who request them. It should be said that each node is identified on the chain by an address in base-58 format, that corresponds to the public key part of the cryptographic key pair associated with the user.

1K31KZXjcochXpRhjH9g5MxFFTHPi2zEXb

Each transaction is then inserted into a block that will be "mined" by specific nodes (miners) whose purpose is to provide computational resources to the blockchain to be able to

[1] Hyperledger Project is a suite of open source permissioned blockchains and related tools, started in December 2015 by the Linux Foundation.

save uniformly the information within, also guaranteeing order in the transactions. Then, transactions not yet in a block are called "unconfirmed".

This offers a starting point to talk about the computational effort to be able to "mine" a block. The network of nodes (the miners, which we'd like to remember that do not necessarily trust each other) chooses the next block to be mined through the resolution of a mathematical problem, where the hash of the entire block plus a nonce (random number) is calculated until the result is less than a certain value. Here, the computation problem can only be solved by a brute force method [1]. This process of validating a transaction is called the Proof Of Work[2] mechanism [3]. We have to state that blockchain transactions are expensive regarding computational efforts. Nowadays blockchains operates at the speed of a few transactions per second [1]. The usage of encryption, in addition to maintaining the secrecy of data, turns out to be used to make it impossible to spend coins from another user's wallet or alter the Blockchain. An example of hash functions includes the family of Secure Hash Algorithms[3] [2]. Furthermore, during each transaction, the hash of the previous block is digitally signed so that the modification of a transaction will not be possible otherwise, we could invalidate the hash. This means that to modify the data in previous blocks, it is necessary to change all the hash pointers of all the subsequent blocks to maintain consistency [1].

Figure 2.1 – Blockchain linked blocks

However, this guarantees control of the various changes of ownership of the coin by verifying the signatures present in each transaction. Generally, blockchain uses asymmetric cryptography, having two keys: a public key to receive coins, or in any case, the hexadecimal address used to identify a node that is obtained as the hash of the public key; and the private key, used to sign transactions (which allow us to use coins).

[2] PoW consensus happens with a lottery where the miner deciding the next block of transactions to be added to the chain is the one solving the PoW, the solution of a hash puzzle find a nonce that the hash starts with n zeros. The mechanism that allows a party to prove to another that computational resources have been utilized for time.
[3] SHA indicates a family of five cryptographic functions (SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512). SHA-1 is the most widespread.

An alternative for controlling transactions, blocks, and coins is through queries on the blockchain (thanks to its transparency) by dedicated explorers (Etherscan[4]) [4].

It must be said that miners are those entities that need space for blockchain synchronization which is the case of 600GB+[5] looking at Ethereum's ledger by early 2022 [1]. Upon completion of a block hash, miners are compensated with a reward - which is why solving blocks is called mining, although its real purpose is to verify transactions and safeguard the blockchain. On average it would take several years to solve a PoW with a normal PC, thus the possibility of an attacker to resolve several blocks in a row before the rest of the network is very unlikely.

In the Thesis scenario, the blockchain assumes a fundamental role for these reasons:

- ensure reliability in a context where transactions identify payments for a service;
- ensure the resolution of the double-spending problem[6] that characterizes digital money;
- encourage the removal of cash to facilitate contactless payments;
- allow additional information such as the invoice number or its amount;
- the last aspect, linked more to design requirements, is the usage of a dedicated blockchain (LuganoChain[7]) that will guarantee advantages to those who use it, such as a single mechanism recognized within the context of Lugano, where we will have the opportunity to make purchases (as a user) in all the other activities associated with the chain, being able to use contactless payment with the LVGA points[8].

2.1.1.1 Ethereum

We have already indicated Ethereum as a permissionless blockchain, similar to Bitcoin, where the consensus mechanism does not include nodes with greater authority and priority. To describe it better, we will start by saying that Ethereum offers a different idea on the

[4] Etherscan is a blockchain explorer for the Ethereum network that allows us to search on-chain data. It is one of the most popular explorers for the Ethereum network and is free.
[5] https://ycharts.com/indicators/ethereum_chain_full_sync_data_size
[6] Double-spending is the risk that a cryptocurrency can be used twice or more.
[7] Better known by the name 3Achain, it is the blockchain that introduces the city of Lugano as a promoter of this technology. Based on PoA and Quorum, it hooks onto Ethereum. Its characteristics, intrinsic also in the name, are Accessibility, Authoritativeness and Reliability.
[8] Token related to the 3Achain and to the MyLugano App.

blockchain in computing terms. It opens up to the world of Smart Contracts and decentralization, no longer exclusively as a distributed ledger, but also computational in terms of resources available to the blockchain. These terms of computation refer to code inserted as Smart Contract within the blockchain, where the nodes are rewarded for the computation of the functions offered by the contract. This must be both immutable and atomic in operational terms, where the parameters decide the integrity and dynamism of the contract [1]. Therefore, Ethereum must be interpreted as a solution in the field of blockchain that allows a wide option of use-cases [5].

Smart Contract is a transaction protocol that executes its terms by minimizing malicious exceptions and the need for trusted intermediaries, lowering fraud loss. However, Ethereum does not exclusively consider Smart Contracts, but includes an entity called "Externally owned account", that have no code associated, but can transfer Ethers, similarly to the Bitcoin system, or trigger smart contracts. To trigger a contract, it needs to be funded with some Ethers in a kind of jukebox-manner, causing its execution in the EVM, a single-threaded machine with halting characteristics in case of never stopping code, that permits to underly both the hardware or OS of a node. A particularity is that if the activator does not provide the necessary amount of reward to execute a transaction, then this is considered invalid, reversing the state changes, but without refunding the activator.

As we said before, Ethereum has a consensus mechanism that assures that each node changes its version of the ledger with the result of the smart contract evaluation going to change the global state of the ledger. In this scenario, the contract doesn't have a private key and cannot initiate a transaction, and might call another contract, but not themselves. Their execution is triggered via transactions [5].

About the cost, and therefore the type of coin identified by Ethereum, we can introduce Ether which therefore enables a range of applications including that of cryptocurrencies, management of tokens, identity management. Also, we could exploit contract features to run (decentralized) applications on this network of computers [6]. The main concept is that Ether can be used to transfer value in transactions and to pay "computational fees" expressed in *gas*. This particular kind of value is managed as a unit of measurement where looking at the amount

of Ether we are willing to spend on every unit of gas, this is measured in *gwei*, where 1 *gwei* is equal to 1 billion of *wei* [9].

Also, Ethereum has a nonce, a scalar value equal to the number of transactions sent from the address or, in the case of accounts with associated code, the number of contract-creations made by this account.

The way we deal with Smart Contract is by using the Application Binary Interface[10]. Then, Smart Contracts are written in a domain-specific or a general-purpose programming language [7]. For instance, Solidity is a contract-oriented high-level language with a syntax similar to JavaScript and with a static type system where the type of the variable must be specified.

Thanks to its computational characteristics, and therefore concerning Smart Contracts, Ethereum is also an excellent opportunity for context-storage and its management in a distributed way within the chain. In particular, there are also the logs generated by the EVM that cannot be accessed by any entities, but they are cheaper than the normal storage feature. The main reason to adopt this scenario is that, as for Bitcoin, are maintained all those characteristics of immutability, decentralization, and distribution that permit us to not tamper with all the info traveling on the chain, guaranteeing persistency during time [6].

2.1.2 Solid Library

Nowadays, it is possible to notice among the Web platforms a centralized control over the information that the user process. There is always this aspect of *Who* owns what, *Where* owns it, and *What* mechanisms use to find information.

This brings some problems, where the main one faces the privacy for the user, or simply if we think about how it may be difficult to move these data (on user request) from one platform to another. And still, just think about the restrictions imposed by APIs regarding how it is possible to access or modify the information. An incisive example is certainly the use of Social

[9] 1 Ether == 10^8 wei
[10] In Ethereum, it's basically how we can encode Solidity contract calls for the EVM and, backwards, how to read the data out of transactions.

Media, where the registered user shares personal information with the rest of the world, actively managed above all by the platforms themselves [8].

From these evaluations arises the desire to self-manage our information, our data within a network or in a common application such as Facebook or Instagram. Related ideas were designed and realized by important names in the Information Technology sector such as Tim Berners Lee or institutions such as the MIT in Boston. The original vision of Web by Lee has evolved into something different from openness, cooperation, and creativity. For instance, data are present on servers of companies like Google and Facebook, where we can delete our histories (are we sure?), but we still can't easily move our data between services. Lee decided to not wait for the current tech giants to switch to an open and decentralized model of data handling, or decoupling the user data from their respective servers. As a direct consequence, he opted to "re-orient" its original vision of the Web [9].

We have to remember that companies themselves use these data to selective make predictions and actions to obtain better results from their products. For this, we might observe a no-sense of these giants to open to a return of control to the users themselves [10].

Imagine a data leaks event, like in the case of the Facebook scandal - Cambridge Analytica[11] - of the previous years. What users (and developers) can do to overcome the data privilege scheme requested by the entire Web? [11]

And here comes Solid, an ecosystem that offers data independence and functions for data management. The main concept to learn is Pod, or containers for user data, accessible as an online resource. These Pods can be hosted on local or public (dedicated) servers. Briefly, Solid has a strong focus on decoupling data from applications, ensuring that apllications could use a defined way to access data stored in the Pods [8]. In this context, each user will be able to control and link their identity across sites, placing them in a Web of relationships.

There are other realities, such as Cozy[12], developed in France, which adopt the same principles as Solid. On the other hand, Solid is still an emergent environment, without an interesting UI that helps users in its workflow. Since this early stage, we cannot find a perfect guide to its usage, and not all of the features covered in the documentation were actively

[11] Cambridge Analytica was a company that specialized in collecting data from social networks. It became famous following a scandal related to data management to influence electoral campaigns through data mining techniques, with which it could identify the psychological profile of users and package personalized messages.

functional, especially differentiating a local Solid server or those offered by different providers [12].

Few use-cases applicative, that try to implement this new orientation of decentralization and privacy-decoupling, comes from the Internet [12]:

- *Mastodon*, a social network project similar to Twitter, where a user could obtain its server;
- *PeerTube* a decentralized alternative to YouTube;
- *Diaspora*, another decentralized social web that has no way to move e.g Facebook data in any format to an account on a Diaspora pod.

Pods use LDP[12] to organize data into containers and resources, giving each their own URI. In addition, Solid is based on a mechanism of linking between data that allows the application that integrates it to be able to obtain data from different users, from different Pods. This guarantees a separation from Web platforms, allowing the user to move in different contexts considering that their data are now under its complete management, available online regardless of platforms. Going deeper, Solid adopts a particular architecture where [8]:

- the user must register with an identity provider (e.g. Inrupt[13]) that saves a user's WebID[14] and cryptographic keys
- data is managed with REST and can be distinguished into structured and unstructured data, where the former are represented with RDF[15] and subsequently managed with common formats such as JSON, while the others should be common files such as images or documents.

[12] Linked Data Platform defines a set of rules for HTTP operations on web resources to provide an architecture for read-write linked data on the web.
[13] Inrupt is a start-up build on the Solid platform that serves as starting project to deal with Solid.
[14] WebID is a URI that refers to a person (or agents) that uniquely identifying the user by his relation to a public key.
[15] RFD is a standard designed as a data model for metadata. It is represented a s a direct graph composed of triple subject-predicate-object.

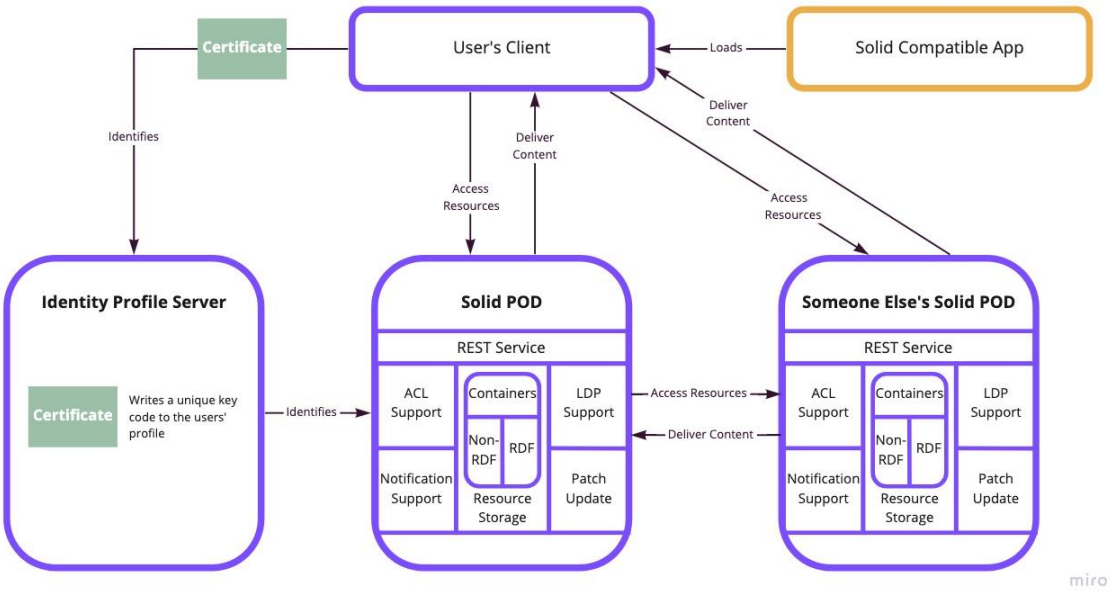


Figure 2.2 – Solid's Architecture

About the WebID, Solid implements global identity management architecture based on the concept of decentralized identity providers [8] [13].

The idea is that any kind of user creates their own identities by linking a unique identifier (in the form of an URI) to a profile document. This "Web of trust" may be used to make authorized decisions, allowing access to resources based on the properties of a user [8]. Then, when we want to stop using an app, we just revoke its access.

The main mechanism guarantees the reading and writing of data on a specific Pod through a user configuration of the accesses to the Pod itself. These access mechanisms are [14]:

- WAC, describing access control at the level of a container or resource with 4 different types of access modes (Read, Write, Control, and Append) where each resource can have an associated ACL, otherwise it inherits the authorization from the parent container;
- ACP, where resources have an associated ACR that determines the access to them by specifying an Access Policies with Access Rules and Access Modes for the resource.

Despite the mechanism we chose, Solid guarantees privacy through the activation of one of them. Privacy in the sense that data now are ready to be used only when the process of authorization is completed, and not simply by requesting them as in the case of a Web page construction. Furthermore, the confidentiality of information is represented by the relative

access policy implemented. Particularly, the "Who" presented by the user desired to access the data permits us to identify uniquely the user in the access policy. In this way, the information will be available just in case of user-granted access.

Actually, the Inrupt servers are available only with ACP mechanisms since WAC is still under the testing phase, so their APIs are not guaranteed to work as well. One of the main differences between those 2 methods is that WAC opens to 4 access modes that are Read, Write, Control, and Append, while ACP shows only Read, Write and Append. The ACP access mechanism uses access policies configured to permit the user within to act as predefined.

In the 3rd chapter, the application methodology for the thesis work will be introduced, which will explain the fact that Blockchain is not a general-purpose database otherwise, it will quickly scale to an unmanageable size. Solid guarantees privacy in a context where, before its implementation, information was available to everyone. If we consider the desire to be able to save information on a blockchain, which is a transparent ecosystem, the intervention of Solid can allow to further decentralize the blockchain going, e.g. to save just a part of the information in Blockchain, and leave the main part(s) on our Pod. This becomes very important with sensitive information such as business invoices, credits or debts, and more, guaranteeing the transparency of a blockchain and the selectivity of data access in Solid.

From this point of view, it is clear that the architectural components of Solid are an incentive for entering the context of Web 3.0. Factors such as OWL, RDF and general semantic aspects, together with the concept of decentralization, user's control of own data, and custom content creation help us in framing the transition from Web 2.0 to the new concept of Web 3 (next section).

2.1.3 Web3

Before diving specifically into the Web3.0 context, we'd like to present the previous (1.0 and 2.0) versions. Surely, version 1.0 is the first-ever interaction with the web, where data are static, without interaction and editing by the audience. In certain sense, here we deal with simple links and texts.

Passing through Web 2.0, we have seen the concept of sharing, by "setting the foot" on social media [15]. Here the content is more dynamic with a lot of attention on user-required content, highlighting the opportunity for interaction. It opens to the possibility of asking questions on forums or using comments to avoid misinterpretation regarding a specific topic [16]. A term used behind Web 2.0 is "mashup", which conceptualizes the composition of online services from multiple sites. This was helped also by the arising of APIs that enable a point of contact to deal with information on the considered Web site [17].

Now, coming to web 3.0, we aim to create more intelligent, linked, and open websites, trying to offer more flexibility to the users. In contrast to Web 2.0, which is understood as being based on users' participation, Web 3.0 will be based on users' cooperation [18]. Especially, the opportunity to offer something customized to the users, based on both its behavior on the Web and its preferences. Also, the emergent field of artificial intelligence leads us to see more relevant adverts that are more likely to be valuable to us [19]. And still, a lot of decentralization will be posted as a question to the repetitive demand for renovation.

However, there are counterparts to all these benefits, in terms of access to the "new Web", that will enlarge its window of adoption. For instance: interoperability between older sites; new regulation becoming; necessities of faster hardware and modernization [19].

We could observe an increment of small businesses, and the adoption of new technologies that enhance features such as payment, cyber-security, data processing or augmented reality. Indeed, Web 3.0 will change the way how companies use the information to market and sell their products and operate their businesses. Looking at the CRM[16], in [20] we realize that Web 3.0 is more pendant to the maintenance stage of the CRM, surely considering its big data characteristic, highlighting the fact that the increasing volume of data on the Web requires more critical strategies [21] [22].

" Web 3.0 is where the computer, rather than humans, generate new information. " - Wolfram

[16] Customer Relationship Management is a process in which a business administers its interactions with customers, typically using data analysis to study large amounts of information.

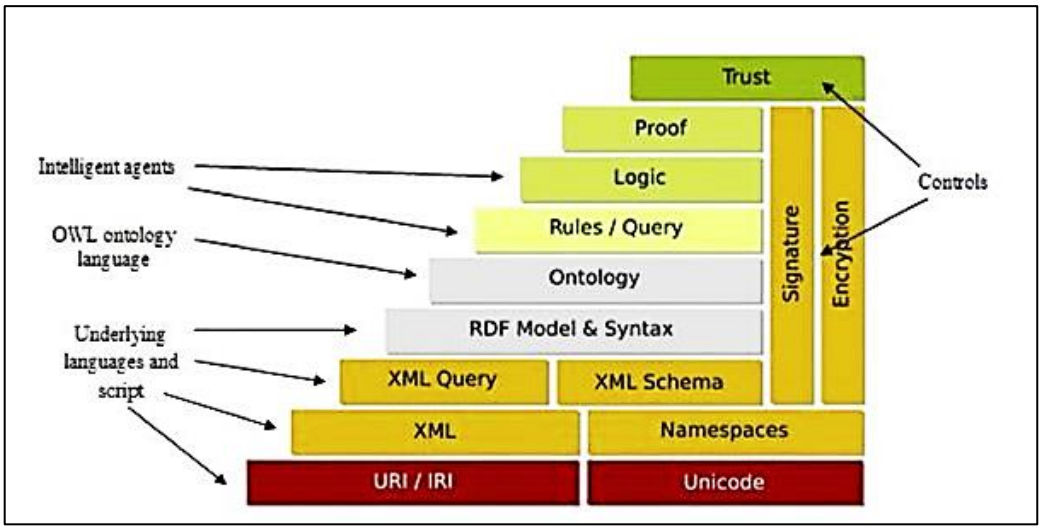


Figure 2.3 – The technological layout of Web3

Stage	Main feature	Hardware	Software	Software development	Theoretical frameworks	
Web 1.0	Web site publishing static information	Desktop computer, server, wired internet	Static HTML pages published by web author	Integrated single document	Standard broadcast publishing "democratised" – first big expansion of publishing since printing press (utopian and/or restrictive technical in scope)	CMC/socio-technical/McLuhan/Fisk (passive audience receiving broadcast model)
Web 2.0	Web site presenting user-generated content	Desktop or laptop computer, server, wired/wifi internet	Database-driven web site with content uploaded by users employing HTML, CSS, PHP, Javascript	Separation of form and content of document	Social network theory (instrumental and lacking sociological depth)	Song – Bourdieu's field/habitus (Bergson/Deleuze critique of notions of habit/memory) Orlikowski – sociomateriality. (Bourdieu habitus constrained) discursive practice
Web 3.0	Application using crowd-sourced data, Internet of Things	Desktop or laptop computer, smartphones, server, wired/ wifi/mobile internet	Mashup of HTML, CSS, PHP, JavaScript, APIs and public micro-blogging IM service	Distributed components and services mashup beyond the document model – document that hits the browser is no longer the centre-piece	Latour (1992, 2004) – actor network, seamless web of heterogeneous interconnections (lacking understanding of power)	Bricolage/mangle/ complexityBergson/Derrida/ Deleuze – material-discursive; temporal seamless flow and decentred knowledge/power bricoleur discourse – flaneur

Table 2.1 – Comparisons of Web versions [23]

The main mechanism that makes richer Web 3.0 is its matching to the term "semantic web", mainly the RDF mechanism that creates statements about particular resources on the Web through a triple expression in the form of subject-predicate-object. As stated by Tim Berners-Lee, *"The Semantic Web is not a separate Web, but an extension in which information has a well-defined meaning, enabling cooperation"*. Technologically, here we see markup languages providing a digital description of concepts in the form of an ontology (Web Ontology Language[17]), which defines a vocabulary that can be used to create data models of a domain,

[17] WOL is a family of knowledge representation languages for authoring ontologies, a formal way to describe taxonomies and classification networks by defining the structure of knowledge for various domains: the nouns representing classes of objects and the verbs representing relations between the objects.

simplifying automatic processing. In this way, we obtain a third layer for documents, above the annotation layer of the structure (XML) and the annotation layer of the layout (HTML).

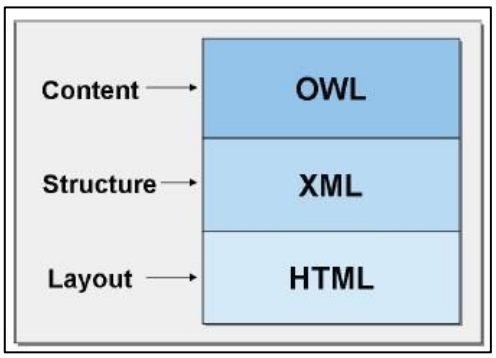


Figure 2.4 – Three Layers of Webpage Annotations

Then, through the use of ontologies, the hyperlinks can be semantically classified, thus enabling semantic surfing and semantic-driven retrieval [17].

Considering search engines, they lack coherence in the reliability of resources and relevancy of the information found (because the information is linked via hyperlinks). New semantics technologies and Natural language processing[18] will enable a search engine to organize information based on the context within the document, and not just by the recognition of phrases [21] [24]. Another example of Web 3.0 is the mobile personal information assistant where users make queries using natural language, and the assistant answers by extracting and combining information from the web, evaluating the information found while applying Semantic Web technologies [17].

The increasing number of mobile devices used by people to read the news while taking a coffee, or share a picture during a visit to a museum, helps us to glimpse the consequences of web 3.0 entrance, where business models will be influenced [25].

Looking at the Thesis work, it offers an interface to the Ethereum network that helps us to the adoption of crypto-currency payments. In this way we could redefine our websites in a way that they can implement wallet and multi-payment forms, enabling in-app payments. But also, Web 3.0 could be the starting point to undertake further optimization from the business viewpoint, maybe looking to custom advertisements and better offerings to clients, optimizing

[18] NLP refers to the branch of computer science - and more specifically, the branch of artificial intelligence - concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

cross-selling and improving the customer shopping experience, taking advantage of integration with social networks, and geo-referencing [25].

Enhancing e-payments, we highlight faster and safer payment at any time, anywhere. The idea is to get revenues from micro-payments for the individual use of data sets [26].

2.1.4 Coin Wallet

We associate the concept of wallet in everyday life with the collection of cash, credit cards, and the several electronic cards we are equipped with (personal cards, health cards, identity card), whose purpose is to have access on specific requests. Although the portability guaranteed by the physicality of the object (and its contents) is convenient and at the same time limitable in a context such as that of available cash (as no one carries large amounts of money with them), some limitations make it an increasingly unreliable and insecure choice. Examples are: limitation of the resources that we can insert, especially in terms of cash; the danger of theft, often due to a poorly check of the object, where the pockets of the trousers are too exposed, and even a backpack (placed behind us) does not guarantee security; always dangers of theft, but with more sophisticated techniques that allow, with a simple approach to the wallet, to be able to deal (and even clone) with the electronic cards inside it. Therefore, it is immediately understandable that the always better techniques of theft go to make this object not very useful and effective in everyday life [27].

To overcome this adversity, the e-Wallet was born, precisely a wallet no longer physical, but digital and with more functionality and reliability than its physical counterpart. It allows users to store and control information like credentials, passwords, shipping addresses, and credit card details [28]. In some ways, they are the most suitable tools for those who make numerous online transactions [27].

A valid example of an e-wallet is undoubtedly PayPal where users send money without sharing financial information. It acts like an intermediary entity between the users that want to make a transaction and the bank. In this way it splits the original payment process, guaranteeing more security to the correct execution of a transaction [28].

The transition to this type of Wallet is duplex:

1. relatively the owner, who will have to become familiar and learn how to use it, and at the same time choose the one best suited to his needs;
2. about those who request payment, which will have to adapt its infrastructure for payments in this new way.

Regarding this last statement, we must not confuse e-wallet payments and POS payments. The latter still requires physical possession of the credit card, while the purpose of the digital wallet is precisely to remove the physical possession of cash and all its variants (such as credit cards). To overcome this "evolutionary" need, it is necessary to conduct evaluations on the type of wallet supported by the seller, the currencies with which it can be used, and its implementation [29].

We could note that each country has its preferences in terms of e-wallet adoption: for instance, WeChat Pay in China is almost the main applicative to doing this; or Touch'n Go as one of the most popular in Malaysia [30] [31].

An important aspect is that of providing an ease of use wallet implementation, also in terms of convenience when dealing with a not-expertized user otherwise, they should tend to refuse its adoption. [30]

An example of an eWallet is MetaMask, an Ethereum wallet implemented as a browser extension that permits us to create free accounts for different networks. Born from the idea of Aaron Davis and Dan Finlay, it allows, through browsers and web3.js (Ethereum API), to interact with dApps and especially active blockchains [32].

An e-Wallet also allows us to manage and monitor digital transactions with a single tool, whether they are device-based or Internet-based (both client-side or server-side): the former need a physical device (usually a pen-drive); the others, on the other hand, require an Internet connection, subject to either data on the device used by the user, or on an external server, which often relies on mobile apps (e.g. GooglePay[19], Satispay[20], etc.).

[19] Google Pay is a quick and easy way to pay on sites, apps, and stores using cards saved in our Google Account.
[20] Satispay is a dedicated app that allows users to buy in affiliated stores, make phone top-ups and exchange money with friends.

The available operations are [28]:

- receipt and sending payments;
- availability of different currencies;
- digitization of documents (identity card, passport, travel tickets);
- finances monitoring;
- trading;
- notifications after a transaction;
- complete control;

Implicitly, they allow to identify the user, often bound in the circulation of coins by government, political or structural regulations such as the acceptance of purchase with an amount of money above a threshold (if adult). Therefore, it allows us to eliminate the re-insertion of personal and confidential information [27] [33].

It should be noted that an e-wallet does not contain the relative currency, but only the information necessary to carry out transactions, which in the case of cryptocurrencies are public and private keys [34].

With MetaMask, the user can control interactions with the specific dApp on how to perform allowed actions and events, and therefore the use of tokens. Considering that each action has its own cost, then this is managed according to the token selected [32].

As already highlighted, an important aspect is the mechanism by which the user authenticates, such as using the fingerprint with a password.

Cash removal is an enormous point of advantage for a seller, as well as for the owner. For both of them, there is certainly an aspect of security towards theft that should be considered. In addition, we note both needs and incentives in the removal of physical money. Incentives to companies that adopt digitized payments, and needs such as the lack of cash from the customer at the time of payment. Another important aspect is the traceability compared to cash, also ensuring better accounting and reduction of illicit activities.

Considering also the pandemic that has affected all countries since 2020, and therefore the need (and obligation) to have to decrease, if not remove completely, physical contacts, makes it mandatory for a seller to implement an e-wallet, thus having to update the business

to the new payment regulations. As mentioned earlier, there is an incentive from governments on the renewal and adaptation of activities (e.g. Payback & Cashless). As result, during the period of lockdown usage of eWallet increased by 44% [35].

By doing so, many companies have introduced electronic wallet services, also to facilitate the process of transactions within their application [27]. The practicality of these new payment methods is found above all in their speed, in their reduced management costs, and in their security and privacy that distinguish them from physical wallets [33].

By considering also commission costs, these can be almost zeroed. In the context of the Thesis, the implementation of the Lugano Blockchain will also reduce the costs for the confirmation of (occurred) transactions.

2.1.5 ERP System

As the term refers, Enterprise Resource Planning[36] is an entity implemented in an industrial environment to manage data from the creation of a product to the customer handover. Its usage enables companies to maximize resource utilization, improve planning processes, obtaining customer satisfaction. It controls the inner enterprise processes to make better choices for the future. This integrated view of management requires that each unit understand how its decisions affect the functions of other units [36] [37].

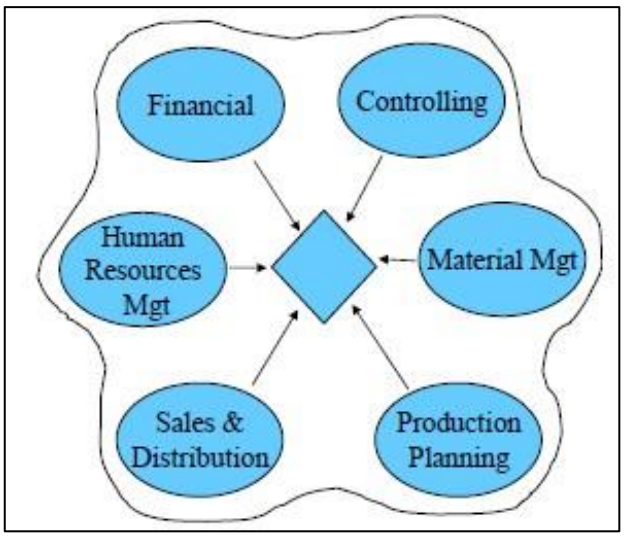


Figure 2.5 – ERP domain

Looking at its complexity, ERP systems are difficult to use, with complex UI designs. Often, they fail to fulfill the increasing demand for adding features. But above all, it can be very expensive both in the implementation phase and in the maintenance phase [38].

They allow data from all departments to be integrated into a single system, optimizing business processes throughout the organization [39].

Technology has been viewed as an additional cost of doing business, rather than an investment [37]. Management information systems, like ERP systems, are often perceived by organizations as very complex and large systems they have to work with, in terms of financial resources invested, people involved, and the scale of implementation [36]. With a successful implementation, the operating costs will be reduced, and improved access to enterprise information will push more Agile decision-making [37].

The amount of data generated in the industry is very huge, where management, validation, and backtracking become time-consuming. Pushed by these problems, we decided to design a data flow model by using blockchain in such a way that generating, processing, and storage of these data become more smooth, transparent, and faster. Instead of hiding data, blockchain abstracts them, enabling efficiency compared to the older approach of ERP. This "new" back-tracking capability will help in case of information discrepancy [3]. By using blockchain, companies can integrate the data of all transactions within the ERP network, ensuring that data cannot be manipulated [39].

Furthermore, blockchain will not replace an ERP system, but be a complementary application that can simplify integration and reduce vulnerability. As said by Brigid McDermott, vice-president of Blockchain Business Development at IBM "*What blockchain does is provide a trust system of record between companies*". Then, Blockchain confirms authorized personnel by using digital signatures based on public-key cryptography [40] [41].

And still, integrating cryptocurrencies with the ERP systems would enable to complete transactions faster and at reduced costs [39].

Although we'd like to mention the next statement in the Blockchain Chapter, we preferred to state it here since it shows importance related to the concepts treated in the ERP nowadays. By adopting blockchain with an ERP system, we should lead to coordination and verification of data between nodes. Just imagine each time the bills accounting manager might fight with incongruence between an entrance and a "simple" human error during digit of

payment. Trying to resolve, we can incur in an intensive job. Thanks to the blockchain and its transparency and immutability, this can be overwhelmed in a few. In this context, it is critical to maintain coordination and verification between data, otherwise we may incur discrepancies, which cause the ERP's client to effort an administrative penalization [42].

2.1.6 eBill

The current invoicing system is not able to boast a substantial number of companies that have switched to electronic invoicing, indeed the evolutionary aspect is delaying compared to the timing predicted by supporters. Although there are countries already included, the majority have not decided upon yet [43] [44].

The passage to an eBilling system will allow us to pay invoices comfortably online, deciding which suppliers we might want to receive electronic invoices from. Then, by logging in the eBill portal, we can verify, print out and save the invoice, and either approve or decline a payment [45].

Some papers and investigations relate the above to a global expectation within 2025. Even if Europe has a good history about the optimization of the (electronic) business process in a step-to-step way, then it should work for a lot of countries with a huge number of SMEs, with a lot of active legislations and languages [44].

EU legislation asks to archive the original (electronic) e-invoices format, even if it were printed. Indeed, the originals are the only relevant invoices for the authorities, while the paper copies could be seen as representations, and not as legal versions [44] [46].

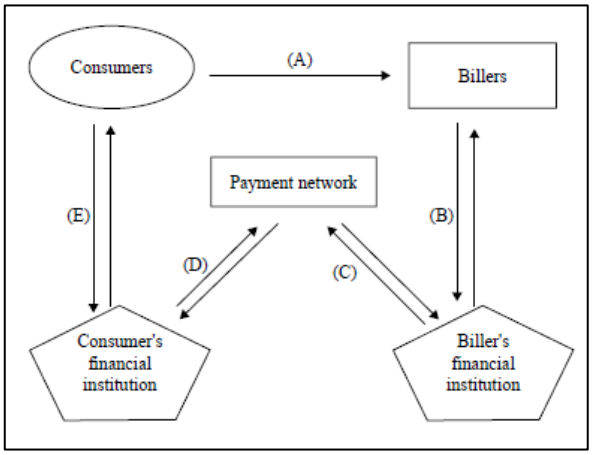


Figure 2.6 – Payment network process flow control

In a certain sense, both invoices and receipts are ways of tracking purchases, with content including the amount of sale, taxes, discount, etc. Its main format is an image-based PDF that could be cheaper and archivable-enhanced. In recent years, this was changed with a structure of PDF with XML data embedded [44].

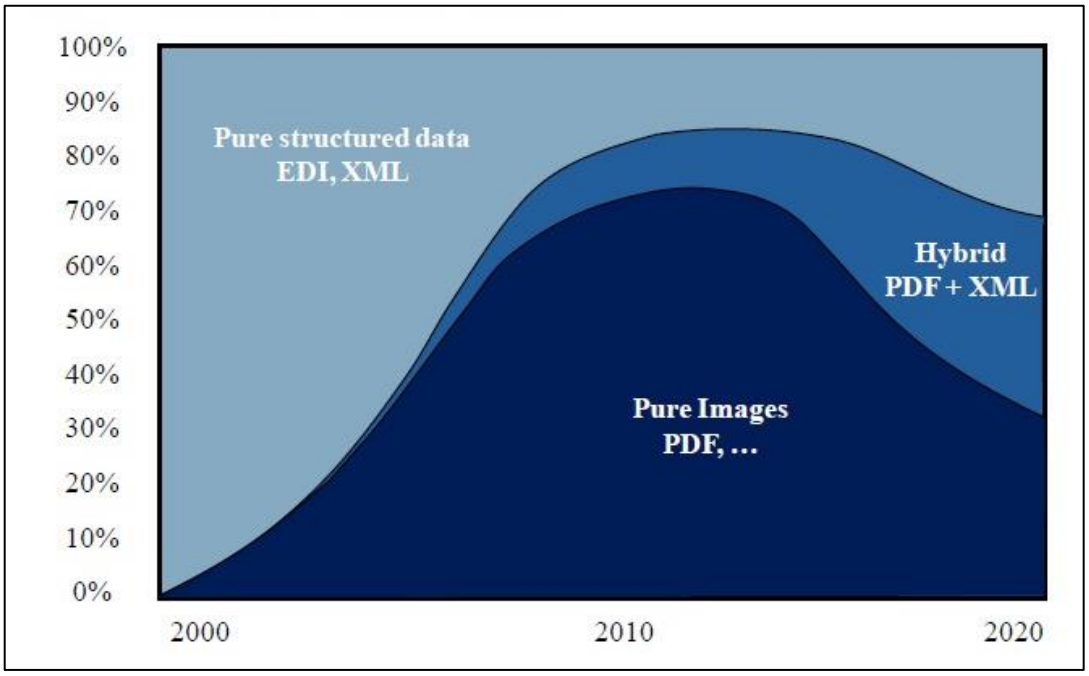


Figure 2.7 – Various invoice formats graph

Sometimes, organizations prefer to insert directly in the billing process the authority check, also because they are required to exchange invoices to them.

The billing aspect is really important in Switzerland, where it is ingrained in the Swiss mentality too. Indeed, in [47] was conducted a survey that confirm the fact that 99% of these study cases were familiar with the billing system.

A supplier distributes its invoices by sending via email, SMS, or Apps, but also providing them in their portal where the user can retrieve them [44]. In line with the ISO 20022 [21], the current form of payment is in the process of being replaced with the payment including a Swiss QR code, where the consumer scans the QR code and passes the billing information to its banking system that could release the payment. Another (Swiss) solution is

[21] ISO 20022 is a standard for electronic data interchange between financial institutions. It describes a metadata repository containing descriptions of messages and business processes.

eBill where the invoice is notified within the incoming bills section of the user's e-banking system.

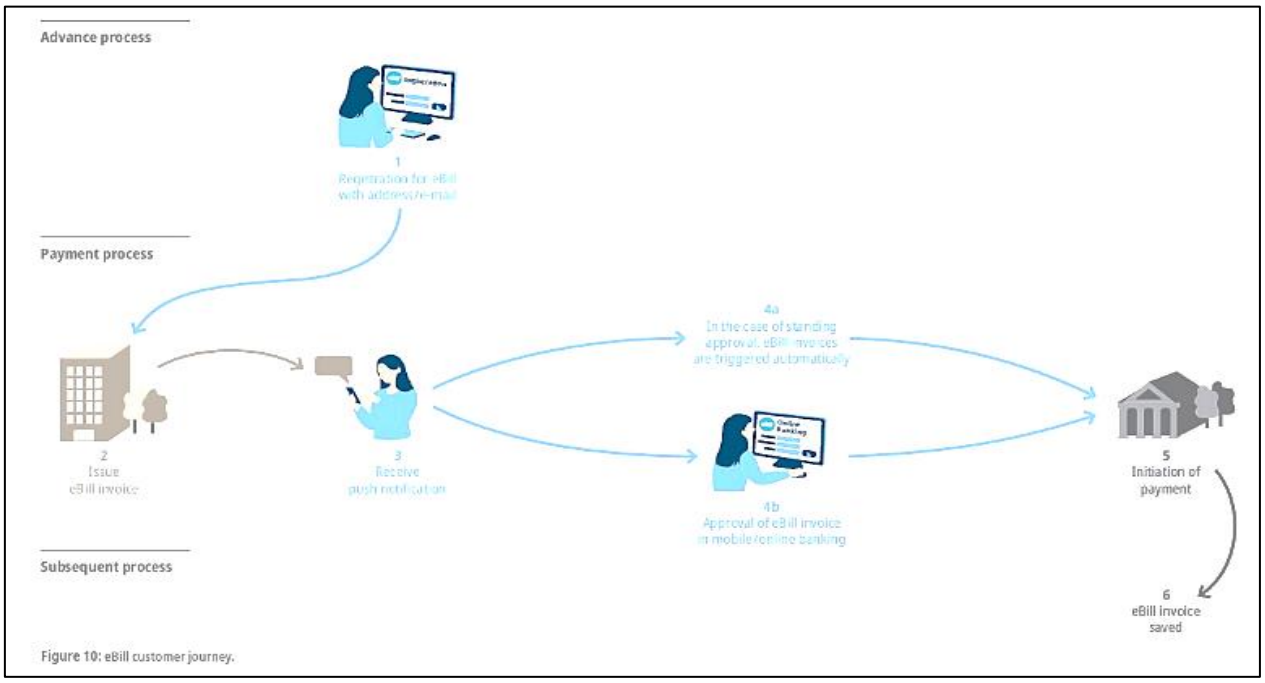


Figure 2.8 – eBill customer execution flow

Also, considering the SIX[22] White Paper [47], it proposes a particular kind of billing system (Direct Debit LSV) able to be seen as a debit authorization where the issuer's account is credited by financial institutions without need of confirmation for each bill occurrence, in such a kind of payment method by installments (rate).

An evaluation that considers factors such as political and environmental dimension, economy, social aspects, and technologies, shows as the QR-code method and (mainly) the eBill system are those showing best feedbacks, and for this, ranked most [47].

[22] SIX operates the infrastructure for the Swiss financial center. The company provides services relating to securities transactions, the processing of financial information, payment transactions and is building a digital infrastructure.

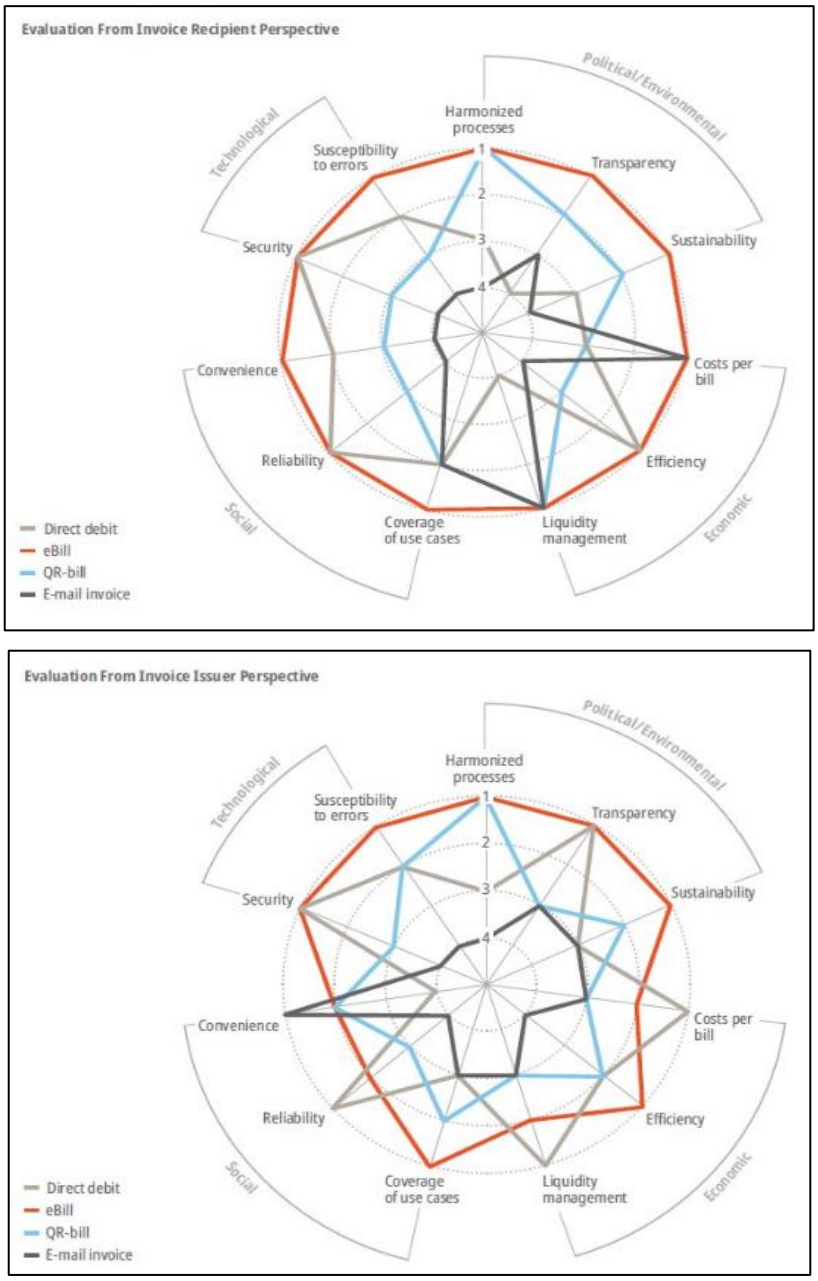


Figure 2.9 – Overview of the ranking from the perspective of the invoice recipient (above) and of the issuer (below).

By implementing an electronic billing mechanism, we should obtain an immediate receipts report to the consumers. In a context where we buy goods on the Web, and for this we exploit the pseudo-virtualization of the client in the Web, an electronic billing mechanism surely exist because we cannot have an exchange of paper report between the parties. But, if we try to imagine a physical context, where the client needs to pay (at least) at checkout, waiting for both the final amount calculation and the closing of the activities conducted by the client, this makes it difficult for the company to manage consumers. Difficulties also in the management of the queue of customers, who are waiting for the completion at the checkout of the different steps for the closure of the customer's account.

2.1.7 Stable coin

Nowadays, a lot of digital currencies have been issued as cryptocurrencies. However, still online payment systems have not employed a cryptocurrency as an alternative to a FIAT currency due to the high price volatility [48]. Here, stablecoin could be defined as *"a digital token that will have low price volatility as a result of being pegged to some underlying FIAT currency"*, where the word "stable" is related to the price volatility [49]. The same term is also referred to as "central bank digital currencies" due to the fact that their release is determined by central banks.

For instance, Tether is an interesting stablecoin that gained importance in 2017, where it increased from $10 million to $2 billion [50].

The question is:

Should central banks issue central bank digital currency?

Central banks are familiarizing themselves with the bits and bytes of digital money [51]. On the positive side, CBDC can be used to alleviate some of the societal ills and eliminate the costs of handling physical cash. While, on the other side, it can give central authorities too much power over the economy and privacy. CBDCs can offer more resilience, more safety, greater availability, and lower costs than its private forms. For example: in the Bahamas, we have the *Sand Dollar*, or in China, the digital *renminbi* (called *e-CNY*). It will be pointed out that, while CBDC is stable with respect to the underlying FIAT currency, it does not make the FIAT currency stable in itself [52].

Following the advices of [53] and [54] we should appreciate these facts:

1. Since each economy is different, there is no universal case for CBDCs, and central banks should tailor on their needs;
2. Central banks have to minimize the impact of CBDCs. In certain cases, countries placed limits on holdings of CBDCs to prevent sudden outflows of bank deposits;
3. CBDC needs a balance between developments of both the design and the policy fronts.

2.2 Related Work & Literature

Before moving to the 3rd Chapter, we will present a review of works related to the topics of the Thesis. It is an analysis of the literature collected from academic articles, papers, conference proceedings, and scientific journals in general, regarding mainly Blockchain contexts, with both Ethereum and Solid. Then, these observations will be used both as ideas for different aspects that will characterize the practical work of the Thesis, as means of reasoning to better understand the context to which we will try to introduce ourselves, and also to better understand the ideas that can arise from these technologies.

2.2.1 Multi Agent System

Multi-Agent Systems[7] consider agents as entities in P2P networks, with expendable knowledge, able to observe the surrounding environment. Here, a fair scheduling and general protection of the whole cluster against malfunctioning is one of the main challenges. It has a role in the development of intelligent distributed systems where accountability and trusted interactions among agents have become mandatory aspects.

> *"… bookkeeping is not just about keeping books or recording profits and losses. It's about the business's narrative that maintains the integrity of the business itself and its connection to the world."*
>
> ~ Dan Palanza

Looking at the bookkeeping accounting system, Kapil Rana talks in [55] on the fact that passing from primitive societies that just write down transactions, to the single-entry bookkeeping, along with the usage of currency, it was possible to record the exchange of goods between a wider range of people, but with impossibilities to examine accountability.

Then, in the double-entry method, every transaction was both an asset and a liability.

Nowadays, Blockchain is a revolution because it is a triple-entry system where a large number of nodes need to confirm that a transaction is valid. As stated by [7], [1] and [56], the combination of blockchain and MAS can provide flexibility and security to these entities. But also, using MAS may solve the scalability issues of blockchain. Considering the healthcare

domain, every actor can be modeled as a MAS agent, and Smart contracts could be used for managing insurance or payments for medical visits.

Request [57][58] can be seen as a layer on top of Ethereum, aiming to become the backbone of world payments where anyone can create a Request (for Payment) for which the recipient can pay securely. All of the information is stored in a decentralized ledger that supports transactions, regardless of currency, legislation, or language.

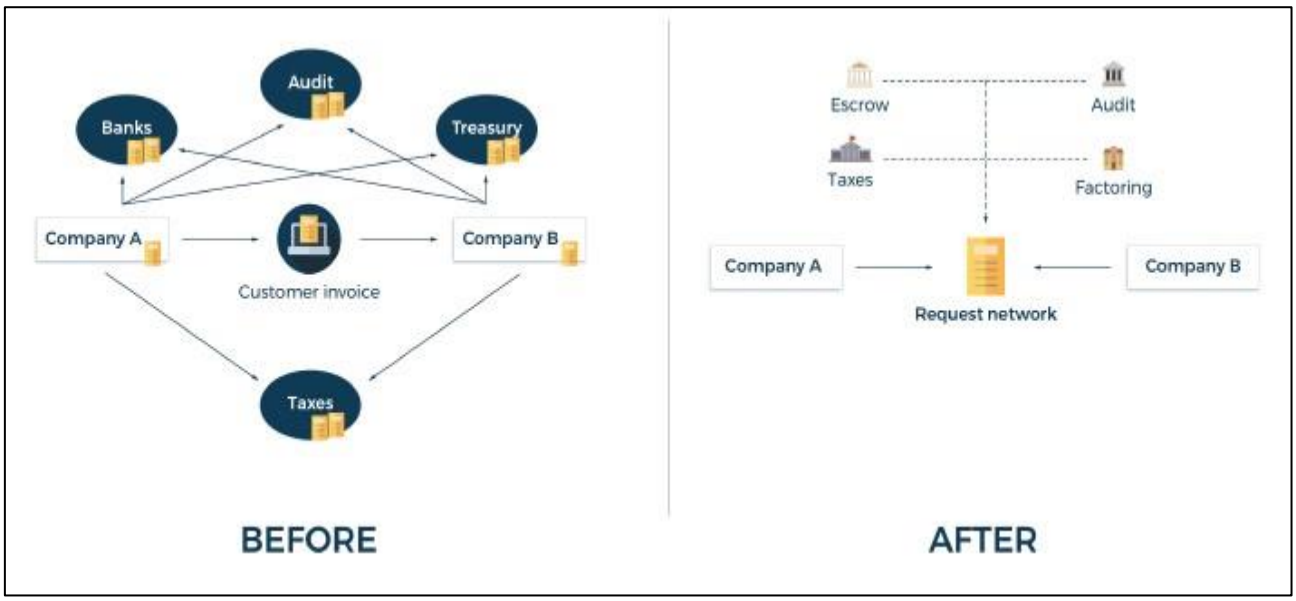


Figure 2.10 – Automatic insertion of additional costs (e.g. taxes)

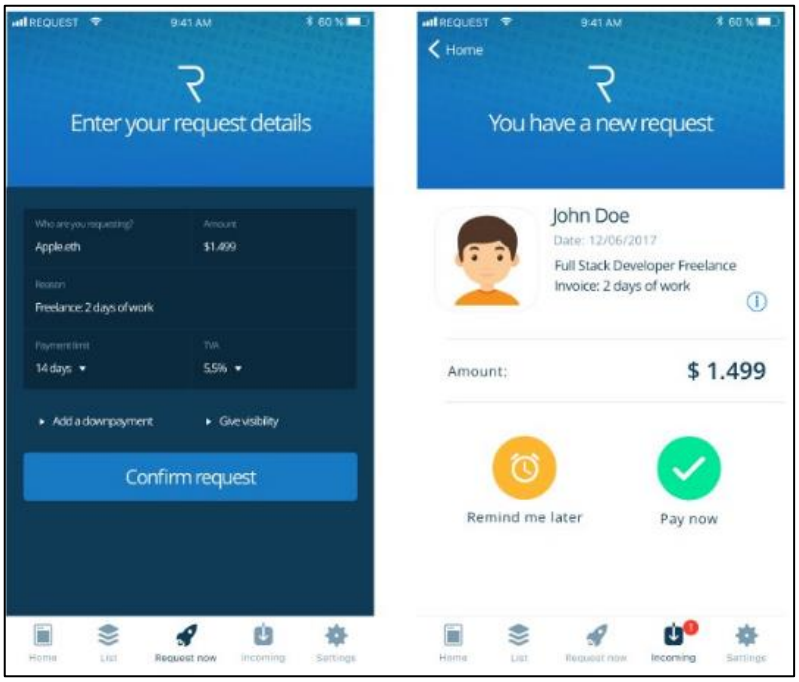


Figure 2.11 – Example of incoming Request

The Request can be detected by the recipient monitoring the network (via a wallet or via a financial application). When a Request is created, the trade laws that apply to its specific case are considered, and taxes are applied. For instance, if Bob was a seller on Amazon and Alice was making a purchase, then Amazon would create a Request on the blockchain that Alice actively analyzes by its phone (notification), and agrees to pay.

2.2.2 Decentralized Storage

[59] preserve a cloud storage model that is not based on trust between client and host, where client data must be encrypted with the SHA-256 hash before a transfer takes place to the cloud. MetaDisk allows decentralized storage by modularity with open-source projects. Also, it can use its cryptocurrency as a means to pay for and exchange storage space and bandwidth.

PingER in [2], is a worldwide end-to-end Internet performance measurement framework with Monitoring Agents spread over the globe that gathers (locally) ping statistics by probing remote sites.

The P2P cloud storage is an interesting application of blockchain as it provides a decentralized data storage facility without involving any trusted third party. The decentralization will help to eliminate data failures by increasing the control over them. One advantage of adopting a blockchain mechanism is that it might add near real-time data collection and analysis across the blockchain members.

2.2.2.1 Solid context

With centralized data management, companies close the control of the user to use and profit from their data. [60] allows for trading access to Solid Pod resources via the blockchain. It showcases an implementation that automates access control based on payment, with specialized modules to listen for transactions and modify the ACLs accordingly. In the virtual world, if our web server uses some payment service on the web, we need to show the respective financial service provider that we adhere to security and privacy guidelines. Just as there is no

certification process to go through before a web server implementation may be connected to the internet, there is no certification process for Solid Pod software.

[61] developed multiple configurations of blockchain mechanism for Solid:

- pod stored files as hashes, with the possibility of using both file hash, or a Merkle tree hash;
- pod-stored files as Smart Contracts, deciding to use the triple subject-predicate-object to facilitate the machine-readable way of linking data;
- pod-stored files as Blockchain key store, adapting Solid pods to behave like a wallet.

Thanks to this work, [61] shows 2 use-cases: considering fitness and education domains.

[11] state that Solid data can be stored on an IPFS[23] while a pointer to the storage address will be stored on blockchain separating web applications from the user data.

2.2.3 Blockchained ERP system

ERPs were previously based on databases where volume is huge and synchronization with blockchain is not so easy. The proposed design in [3] provides more transparency, smoothness, security, and an easy approach to organizing enterprise data by using blockchain technology. This will help in tracking back any product, understanding the flaw points in case of any specific product-related discrepancy. This can open a view on homegrown ICOs which might be used for validation in intra-network Blockchain.

2.2.4 Blockchain privacy transactions

Zhiyuan Sun in [62] investigates on ConsenSys Rollups, a software service that aims to protect blockchain transaction elements such as wallet addresses and transaction amounts. This is a popular cryptographic technique since users fear their crypto transactions are at risk of being tracked by blockchain forensic firms.

[23] The InterPlanetary File System is a protocol and peer-to-peer network for storing and sharing data in a distributed file system.

2.2.5 Blockchain notifications

For Ashvarya Kharoo, in [63], push notifications will be a revolutionary aspect of DeFi, that Ethereum Push Notification Service[24] intends to solve by aggregating them in one place.

[24] EPNS is a notification protocol which enables users (wallet addresses) to receive notifications in a platform agnostic way (mobile, tablet, web, fav. wallets, etc).

Chapter III

PROPOSAL

"In all chaos there is a cosmos, in all disorder a secret order."

~ C.G. Jung

The chapter aims to describe the implementation solution designed. An analysis of the intervention, the available implementation solutions, and the improvements they could bring will be carried out, and then a final proposal will be conducted. Subsequently, the methodological-practical part will be described where the development environment and the necessary programming will be analyzed. In particular, we will immerse ourselves in the context of a restaurant that has decided to renew its core business through innovative and efficient mechanisms.

3.1 Use-Case: Restaurant

The first goal we set was to analyze the use-case in question, therefore the dynamics of a restaurant, and above all the points of view of the restaurateur himself. Having made this consideration, we will first introduce a description of the dynamics of a restaurant, also trying to integrate the notes and advice of the restaurateur. This will help us to identify problems and needs, subsequently used as food for thought.

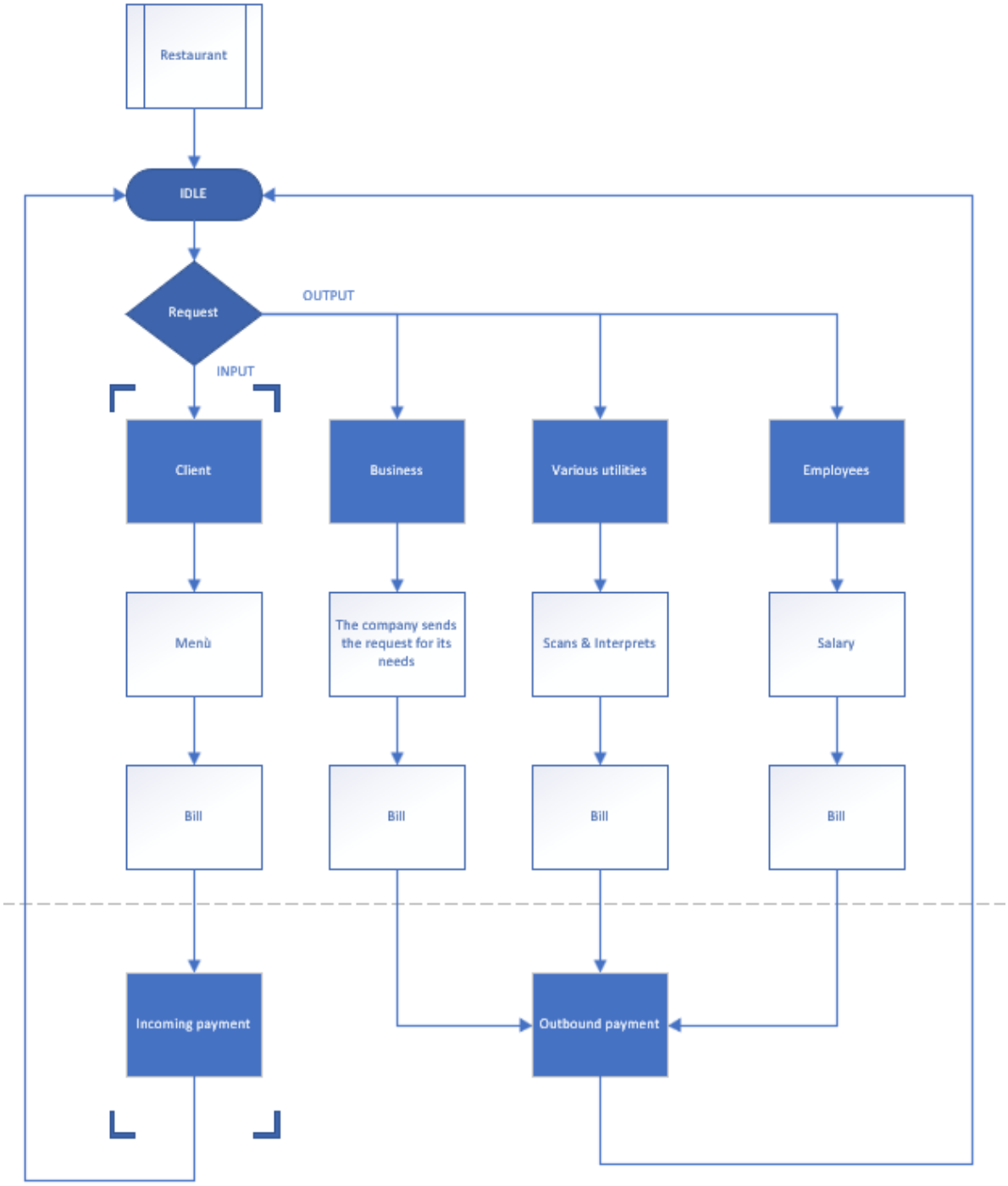


Figure 3.1 – General restaurant incoming/outgoing flow

The flowchart offers a description of the steps that are usually faced in a restaurant, but also what the activity faces to offer a service and manage the flows of different agents. What we notice is that the company reacts according to the request and the agent it must satisfy, that is, we can have an input request such as the case of the customer who requests a service, makes selection, receives an invoice, and makes the payment; the alternative is the outgoing requests,

where the company issues payment and an invoice for service received. In the latter case, the figures identified are those on the business side, therefore suppliers, those of utilities, and that of employees.

This vision allows us to have a first abstraction on the steps of a restaurant or, if desired, of a company in general. To go into even more detail, we will try to rely on a new chart based on timelines, trying to highlight the different parties involved, together with the internal mechanisms of the restaurant's activities.

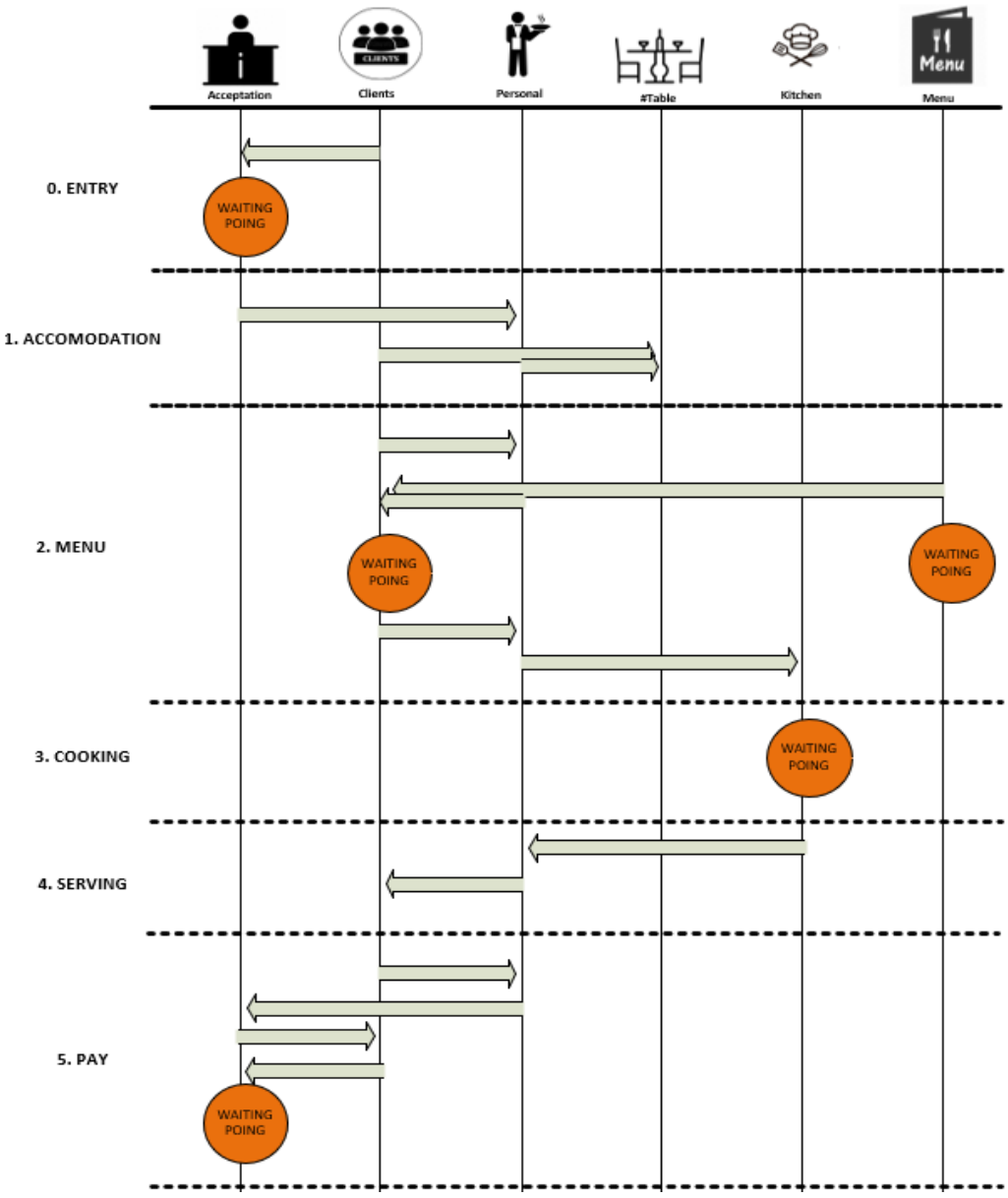


Figure 3.2 – Restaurant's steps timeline

The entire representation now offers a complete view of the restaurant's use case.

Trying to analyze the representation alongside the reflections of the restaurateur, we can make the following assessments:

- point 0 highlights the first deficit that the restaurateur faces, namely the waiting of the incoming customer, undoubtedly an annoying wait for the customer who wants to get to the restaurant and be able to sit down without having to wait outside the room or at the bar at the entrance - where drinks can also influence the culinary experience of the customer. Often, the wait can also result in more than 30 minutes without being able to sit at the table, especially in periods of high tourist activity, causing loss of customers, who not only will not be able to consume the dishes of the restaurant, but also share negative feedback due to the lack of experience lived.

- point 1 highlights the impossibility of being able to sit freely at a table given (possibly) the absence of booking service, as well as the wait for the queue to flow. It must be considered that once the user is made to sit down, then may not even have access to the preferred table, which therefore could annoy the customer. Focusing only on the restaurant, having to appoint staff to accommodate customers because of one of the previous items, can result in an unnecessary waste of company resources, which could be dedicated to other tasks.

- point 2 is one of the main bottlenecks of the entire customer entry-exit process. A triple deficit is evident such as:
    - the need to request the menu, and therefore possibly also the wait for the waiter who brings them to the table;
    - waiting for the confirmation of the order by the customer (who will have to choose) and the waiter (who will have to deliver the order to the kitchen);
    - the possible unavailability of the selections made by the customer, and consequently having to ask in the kitchen and the chef for availability, any counter-proposals, and return to the customer to recommend alternatives to the selection made by him.

- point 3 seems to be the only one "free" of inefficiencies, if not related to optimization of coordination within the kitchen.

- point 4 should coordinate kitchen staff for customer service, and it is evident that there is "repetitiveness" in kitchen staff calls.

- points 2-3-4 also bring out the possibility that the customer may request an addition to the command which will then involve the repetition of points from 2 to 4.

- the fifth point shows the client's desire to want to end the experience through the account request. The series of steps that need to be followed shows how further waiting is necessary due to the preparation and closure of the account at the checkout. Any inability to pay at the table due to the POS at the cash desk could lead to further waiting for other customers during payment. Around the corner, there is also the impossibility for the restaurateur to be able to guarantee all the payment methods that the user may request.

The needs we have sought have been based very much on the wishes of the restaurateur, as well as in consideration of the points mentioned above. These can be summarized with:

- ❖ the desire to reduce waiting times for the customer;
- ❖ the possibility of offering more freedom to the customer using the structure and therefore table, menu, etc.;
- ❖ activate a model for personnel management that allows making their use more efficient;
- ❖ greater coordination between staff, e.g. kitchen-waiter.

All these details allow a developer to put forward different proposals to the restaurateur, who will then be offered the possibility, but above all the opportunity, to digitalize the business. For example, on the part of the customer, there may be a need to have the invoice at hand, anywhere, on their mobile device.

The project aims to bring the business management functions within "autonomous agents" (smart contracts) and decentralized (blockchain) able to control and automate most of the company's administrative activities: payment flows, the management of receipts, and the management of employment contracts and payment of wages. Not only that, the set of smart contracts want the creation of an open network of companies able to manage B2B transactions in direct form going to optimize and further automate the intercompany input/output processes.

The ambition is to create a completely decentralized management system, with accounting and budgeting that can appropriately manage the new forms of digital currency and direct payments. Fundamental in this context will be the ability to produce, store and share the supporting documents relating to individual transactions (bills, invoices and receipts) necessary to meet the legal needs of keeping and auditing accounting.

The risks associated with this project are to be understood in the brake on these new forms of digital currency imposed by states and central banks and which will slow down its spread. A further risk is a high standard in the Swiss and European banking system, which can be a brake on the growth of new financial models, but which we believe will establish themselves in many economic sectors in the medium term. The study, deepening and practical implementation of decentralized web technologies integrated into the Distributed Ledgers world have an interest that goes beyond the thesis work here.

In a P2P direct payments model, even invoices, receipts and payment confirmations, now produced in paper form, require completely digital management, completed with archiving and sharing functions to ensure transparency, audit, control and the production of reports and statistics, including the company balance sheet.

From a scientific point of view, the project aims to allow a better adoption and dissemination of blockchain technologies at the level of SMEs. DLT and Blockchain are, among the emerging, disruptive and promising technologies, those that more than others seem to have the potential to significantly influence the accounting and auditing sectors. From a scientific point of view, the project aims to integrate the world of decentralized transactions with the paradigm of "decentralized web" and therefore of a private area for storing and sharing data under full control of the owner of the data.

The purpose of the development work should be mainly to:

1. guarantee a form of contactless payment, with openness to the use of innovative payment methods, including cryptocurrencies, and try to open the way for "stable-coins";
2. make electronic invoicing our own, and specifically the eBill system;
3. allow smart accounting to reduce administration costs;
4. use of smart contracts to manage multiple purposes;
5. security and reliability thanks to the blockchain.

3.1.1 Intervention

The intervention proposal, therefore, provides that the entire Business Core of the restaurant will be renewed. The infrastructure will be further digitized and adapted to new technologies that will allow it to start a process of reducing administrative and non-administrative costs.

In particular, the intervention provides that:

- develop a Web platform to be able to interact with the different phases of the culinary experience, and therefore that allows the selection of products from an interactive menu, update the command, close the account, pay and receive the invoice on our device;
- the implementation of the Solid ecosystem, and therefore of Pods, adapted to the different needs of the restaurant such as invoices paid and to be paid, or to indicate the orders concluded, or even the products in stock;
- implement a QR-code system that allows us to identify the user in the process of using the WebApp based on the table at which decides to sit. In this way we will be able to activate a mechanism for sending the control directly to the kitchen, which will know the table to be served;
- activate an eWallet for payments in cryptocurrencies, and therefore the implementation of MetaMask as an integral part of our system. This will allow us to manage transactions, and therefore able to communicate correctly with the blockchain;
- configure the wallet to deal with (and activate) the Lugano blockchain;
- configure the new electronic invoicing system. In particular, invoicing will integrate a QR-code system that will allow us to identify the location of the invoice faster. In fact, it was thought that this system can speed up the queue at the checkout thanks to the fact that if an invoice is paid, it will be immediately moved to the dedicated Pod, so the scan at the QR checkout on the invoice downloaded by the customer will guarantee the exit from the restaurant.

Entering into the merits of the intervention to be proposed, it is necessary to replace the previously identified points of disapproval (Figure 3.2) and introduce new schemes with the required functionalities.

The introduction of these improvements should allow both a return on the costs of administration of the activity, and speed up and optimize the entire executive process of the restaurant. The option of being able to reduce administrative costs would ensure that these funds can be re-invested in other activities related to the company such as dedicating them to a continuous renewal of the infrastructure, the recruitment of new staff, or any expansions. In this regard, the Core Business: the possibility of reducing, if not even eliminating, the customer's wait, as well as allowing to pay with cryptocurrencies or, more generally, use mobile device to interact with the restaurant itself, will guarantee an innovative and more immediate experience.

3.2 Development

This section will be dedicated entirely to the description of the practical work carried out. We will describe the development environment and the ecosystem designed for the restaurant, identifying the infrastructural layers, also graphically. These will be thoroughly explored in the respective subsections through the interpretation of the most interesting and fundamental parts of the code.

3.2.1 Environment & Implementation

The environment in which we started the development of the application, and therefore the coding of libraries, interfaces, WebApp, is mainly based on the Visual Studio Code editor. To this, we could add the Google Chrome browser for browsing between web pages, and Heroku[25] for the transition to the "production" phase and then the publication of the application.

It is the most appropriate environment for the fact that:

- allows us to quickly interact with GitHub,
- allows us to work instantly from the CLI,
- ensures excellent management of saves and backups,
- highlights the most interesting pieces of code,

[25] A cloud platform as a service, developed since June 2007, supporting many programming languages. It has features for a developer to build, run and scale applications.

- allows us to integrate any functionality through the selection of specific extensions.

The image below represents the different components touched upon for the design of the entire ecosystem.

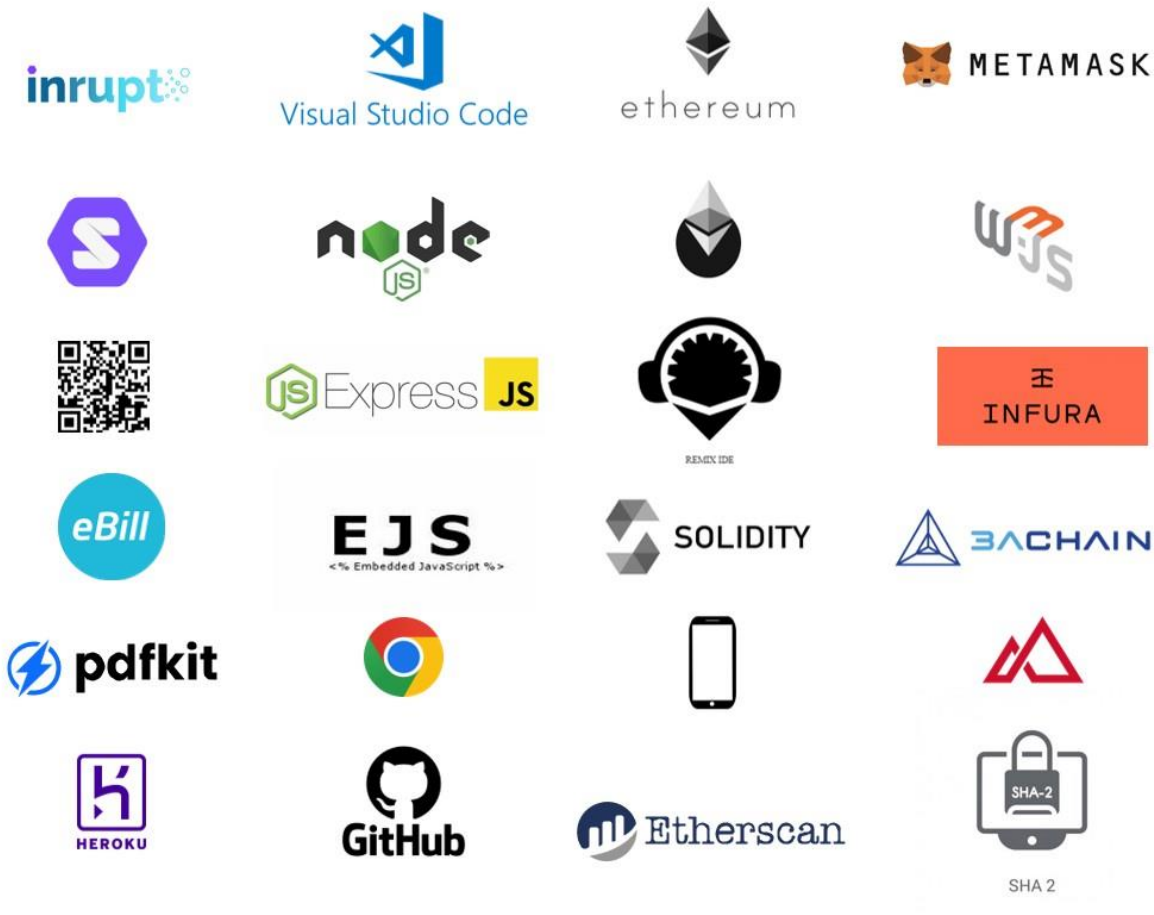


Figure 3.3 – Infrastructure Components

The application can be started with the *-i* option that activates the initialization of the Pods within our Solid-context, thus eliminating the previous implementations. After that, by connecting to the localhost on port 8080 (during test phase) we have access to the WebApp. It is necessary to indicate in the URL the table from which the order is being worked. The final version allows us to directly access the table by scanning the respective QR-code. The first connection returns to the menu taken from the dedicated Pod, dynamically considering the quantities available for the products thanks to the consultation of the store. At the end of the experience, the MetaMask interface allows us to pay and close the account.

Going deeper, here is a brief introductory description of some implementing peculiarities:

- the number of tables assigned will be managed according to the occupation of the same. An array ensures that if all cells contain a "1", we are redirected to a page indicating that there are no tables available;
- The *utils* folder contains the files related to the store. This, structured in the form of JSON, shows availability in terms of quantity, along with other information such as a brief description of the product, or the period of presence on the menu;
- To help in stock management, a quantity increase function when below a threshold will be discussed;
- the users who will be impersonated during the executions of the application are:
  - admin, with total control over the code as well as the ownership of Pods created in the personal space in Solid;
  - restaurant, understood as the activity itself, from which we request menus or interact in different ways

The following images allow us to glimpse the generic infrastructure and the implementation of the blockchain within the application.

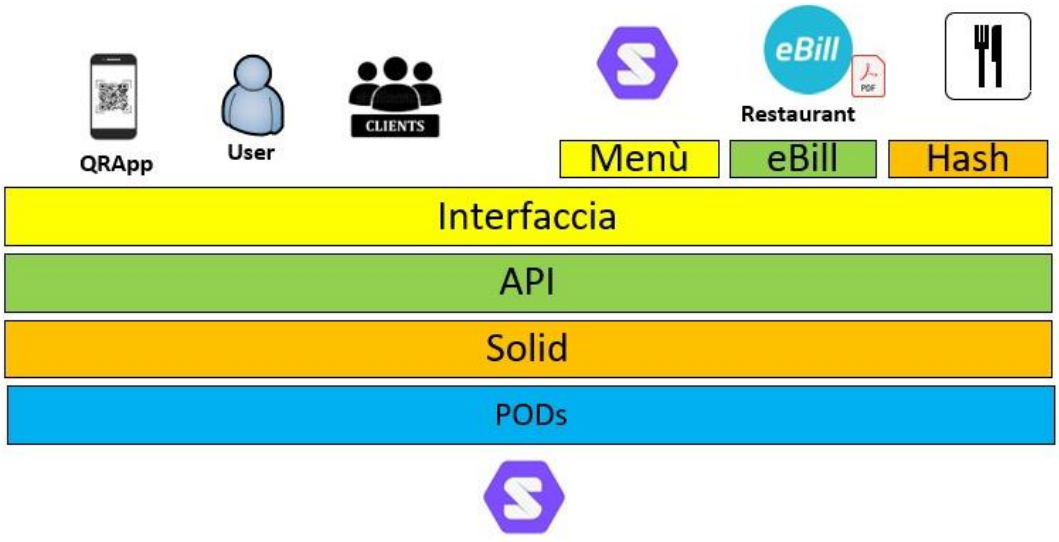


Figure 3.4 – Infrastructure Layers

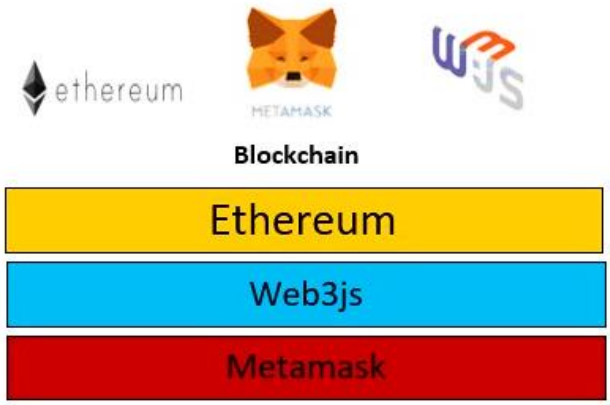


Figure 3.5 – e-Wallet-Blockchain structure

3.2.2 Server Backend

For the server part, we decided to use the NodeJS engine obtaining simplicity of use and modularity through the package manager (npm). In fact, the main example is the Express framework, which with a few command lines allows us to expose a Web interface. In the case of the restaurant's WebApp, the web interface is displayed locally on port 8080. The version of NodeJS active on the machine is 14.17.6.

Other setup libraries necessary to ensure the functionality necessary for the interface are those indicated in the following piece of code:

```
$ npm install buffer express webpack webpack-cli webpack-dev-server css-loader
style-loader gulp turbo-cli
```

To help the dynamism of the web pages we opted for the use of EJS or Embedded JavaScript as an instant framework and to promote adaptability and interoperability with the underlying NodeJS ecosystem.

```
$ npm install ejs
```

It allows us to use JavaScript code when loading web pages. Like other interoperability frameworks, it allows us to invoke static components to the page, and also define any directories with different service functions.

To ensure the opportunity to manage multiple connections at the same time, and thus ensure the management feature of free tables, we felt it necessary to use the "cluster" library. In this way, we can manage multiple connections from the same table when we have more seats, and also more connections from multiple tables.

The source code files (Appendix A) and all stuff needed for the execution are distributed as follows:

- ***./src*** folder containing the custom implementation of the Solid library, the web-app functions and the custom pdf-kit version that contain the beautify template for the bills;
- ***./static*** folder contains the style-sheet (.css) for the WebApp;
- ***./utils*** folder contains temporary files for the figure of the "store", the relative product images, the code of the smart contract used during the project, and the temp file used to store a pre-bill state;
- ***./views*** contain the pages served by the different steps of our WebApp, in .ejs format (see Section 3.2.4);
- ***restaurant.js*** containing the main of the Node.js server, used to start the context and the Restaurant WebApp;

At the end of the development and testing phase, we decided to rely on an account on the Heroku platform for publishing the application, favored by sharing the repository on GitHub, which will keep the public version on Heroku constantly updated.

3.2.3 Solid Pods

This section deserves a dedicated in-depth analysis as it was both an interesting and innovative aspect. In some ways, it struck us strongly and deviated from the context of the blockchain, the cornerstone of the thesis and the project.

We want to mention how this technology is recently developed, and therefore required several learning sessions, participation in Workshops and meetings between Solid developers, as well as active participation in the discussions on the different Solid Community Forums. This last aspect has also served us a lot to have the point of view of other more experienced developers, as well as being able to know their creations in a Solid context.

It is necessary to make a clarification to introduce our approach to Solid: in the first phase, we started by going to investigate, not so much on the Solid library, but trying to identify the best context where we can activate our Pods, "my Solid space". The choice was between a local activation (NGINX server or Docker approach) or relying on external services that, in fact, allowed to activate personal Solid servers. This investigation led us down the road that then allowed us to get to the Inrupt APIs and its context of opening a Pod (on the Internet) on their servers. Recall that Inrupt is the start-up started by Tim Berners Lee, one of the proponents of Solid.

Subsequently, after analyzing the library itself, we decided to implement a (further) interface to this library that would allow us to customize some aspects and parameters of the original calls to the functions exposed by the APIs, obtaining an interface more suited to the needs of our application (code in Appendix C).

It was necessary to import part of the modules of the Solid library, within our NodeJS code, and in particular:

- ***solid-client***, a client library for accessing data stored in Solid Pods.
- ***solid-client-authn***, a set of libraries for authenticating to Solid identity servers:
    - ***solid-client-authn-node*** for authenticating to Solid identity servers, specifically with NodeJs.

- ***vocab-common-rdf***, A library that provides convenience objects for many RDF-related identifiers, such as the Person and familyName identifiers from the Schema.org vocabulary from Google, Microsoft and Yahoo!

```
$ npm install @inrupt/solid-client @inrupt/solid-client-authn-node
      @inrupt/vocab-common-rdf @inrupt/generate-oidc-token
```

These modules allow us to have the main features of Solid, and therefore the management of information in our Pods, the management of access to information, and the facilities to work in RDF-context. This last item did not have wide use in the application as it was decided to work mainly with files and documents, rather than managing information in unstructured databases.

Among the functions made available, we went to re-implement only a part of these, and in particular those for:

- identify and log in the user, including the *Session* class or the *getSessionFromStorage* function;
- creating, uploading, and removing information and containers, including the *createContainerAt*, *getFile*, *getSolidDataset*, or *deleteContainer* functions;
- accessibility and authorization to information through the ACL policy (section 2.1.2), including *get/setAgentAccess*, *createRule*, *createPolicy* or *addMemberAcrPolicyUrl*;

Before introducing the initialization mechanism of our Solid context, we wanted to deepen the authentication phase. Mainly, this happens when a user has to access a Pod (own or external), often due to a loading of a Web page that must find information from the Pod to render 100% of a page, or even when we need to send or download a file from a Pod. On these occasions, the browser shows the login window of the provider on which the requested Pod resides, and where the user enters their credentials and starts access - which will be based on the policies set by the Pod admin. This solution, therefore, requires interaction and a break in execution to enter **manually** the login credentials, resulting in an often-inconvenient choice. In this regard, the previous introduction on the implemented libraries and the specific selection to use the one for NodeJS allows to authorize access to the Pod through a different mechanism: each user uses a WebID and the provider from which his WebID was registered, to request

temporary access tokens through the *generate-oidc-token* utility. Thanks to this tool, we can use NodeJS without having to manually perform the authentication process.

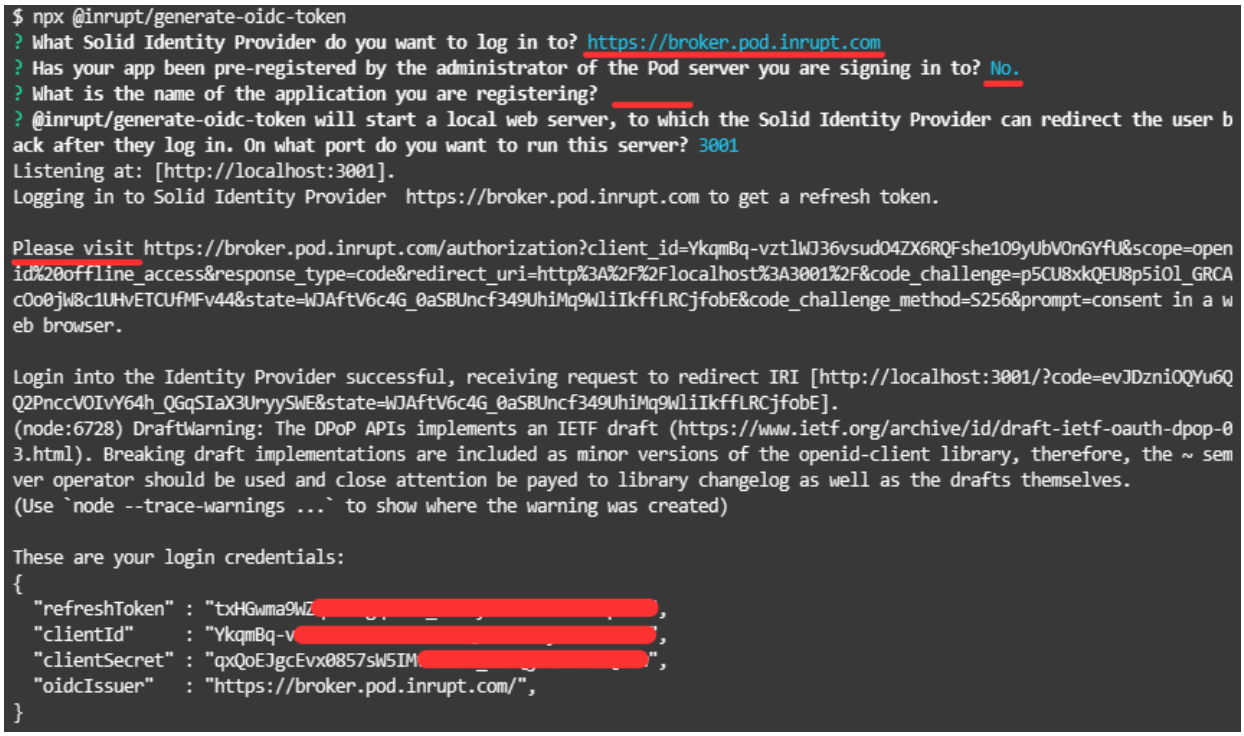


Figure 3.6 – generate-oidc-token utility

We can now use these tokens to authenticate ourselves "before-execution".

To enhance the execution of the application and the context, we decided to activate an option for the NodeJS script that would allow us to initialize our Solid context to Inrupt, and have everything ready for a first execution. This script will then allow us to delete the previous configuration and start again with a clean installation.

To make the initialization work properly, it will be necessary to generate tokens for the WebID of the Pod admin (potentially our restaurateur), the only one able to maneuver the Pods at will.

```
initialize(adminToken);
…
…
async function initialize(token) {
        //login as admin
        const session = await login("Admin-", token);

        await createDatasetACP(obj.acp, session);

        if(session.info.webId != obj.CEOWebID){
                await createAgentRuleACP(obj.acp, obj.CEOWebID, "restaurantCEO", "adminRoot", session);
                 await createPolicyACP(obj.acp, { read:true, write:true, append:true }, "restaurantCEO",
"adminRoot", session);
                 await createMemberRulesPolicies(obj.adminRoot, obj.acp, "restaurantCEO", "adminRoot",
session);
        }

        //!!INFO: TO DELETE WE NEED TO HAVE AN EMPTY CONTAINER
        //AT THE MOMENT THIS CAN BE DONE BY MANUAL DELETION
        await deleteContainerFromPod(obj.root, session);
        await createContainer(obj.root, session);

        createStoreSection(session);
        createBillingSection(session);
        await createActiveOrderSection(session);
}
```

From the code we can see that:

1) the login through the credentials of the admin;
2) the creation of a Dataset for the management of control policies;
3) the explicit insertion (to overcome any execution bugs) of the "restaurateur" user within the dataset for a policy with full access;
4) the elimination of the previous execution, and then of the containers created;
5) re-initialization of the new Pods;
6) creation of specific directories within Pods.

As a Pod, we decided to create some to store:

- information relating to the warehouse;
- invoices paid;
- active orders.

Respectively, the initialization of each of these sections includes the creation of the container, the explicit setting of permissions for the restaurateur on the container (always to ensure the absence of bugs during execution), the upload of any files, and the setting of access to the container and the initialization and configuration files.

The container related to the warehouse will contain the JSON file that simulates the products and quantities present in the warehouse, along with images that will accompany the rendering of each product on the home page. Regarding the access permissions, these will be set to allow reading for anyone. This is because it was decided to make the menu a public resource, or give the possibility to the customer who sits at the table to be able to access it, without authorization mechanisms by the customer or the restaurant for finding. Although it is necessary to guarantee more security to the information of the warehouse, with this access policy we can avoid any change, and only allow read access to the store that will help in the initialization of the menu.

The section for invoices includes 2 Pods, one for bills to be paid and one for those paid. The initialization involves the upload of a template for invoices, and the setting of accesses to the Pod that will be divided as follows:

- the figure of the restaurant can always have full access, subject to read-only for the invoice template;
- while the supervisory authority and the ERP system will only be able to read the container for invoices paid.

Finally, the Pod for active orders will need the upload of a template for orders, the setting of full access for the figure of the restaurant, the only one to be able to access this area, and the cancellation of any active orders for the tables of the restaurant. In this regard, we want to remind that each table will have its file for order management so that we do not have to request any session management mechanism.

As mentioned above, it is not necessary to specify an entry for the Pod administrator within the policy dataset, as being an administrator guarantees all rights to all resources. However, being in the testing phase, we decided to make these permissions explicit both to

highlight this feature and to prevent external accesses from revoking the status of "owner" to the admin.

By interacting with Solid, each function of the application activates a different characteristic. Just to highlight some of them:

- *makeOrder* and *updateStore* are two functions that work together to allow, at each confirmation of the sending of an order, to send replenishment requests to the products of the warehouse that are about to run out of availability. The first takes the order selection from the menu page and turns it into a JSON file, also assigning an order ID and a timestamp. After that, go to overwrite the order present for the relative table. After confirming the order, the second function is activated that updates the quantities ordered for the products, subtracting them from the total available, and increasing them if below a predetermined threshold. This mechanism simulates what in reality is seen as sending a replenishment order for companies of specific products.

```
async function makeOrder(req, session) {
    //get info/params for order
    var order = JSON.parse(req);
    if(order["id_order"] == "orderID"){
        let orderID = "order-";
        orderID += Math.random().toString(36).substr(2, 9);
        order["id_order"] = orderID;
    }

    var today = new Date();
    var date = today.getFullYear()+'-'+(today.getMonth()+1)+'-'+today.getDate();
    var time = today.getHours() + ":" + today.getMinutes() + ":" + today.getSeconds();
    var result = date+'-'+time;
    order["time"] = result;

    var total = 0;
    Object.keys(order.products).forEach(function(k){
        order.products[k].amount = order.products[k].quantity * order.products[k].price_per_unit;
        total += order.products[k].amount;
    });
    order["total"] = total;

    //insert as file .json
    var table = order.table_number.toString();
```

```
    order = JSON.stringify(order);
    await uploadJSON(order, obj.activeOrderContainerURL, `order-table-${table}.json`, session);
    await createResourceSpecificPublicRulesPolicies(obj.activeOrderContainerURL+`order-table-
${table}.json`, "activeOrder", { read:true }, session);

    return JSON.parse(order);
}
…
…
async function updateStore(json, session) {
    //get the store to update
    var store = await readFileFromPod(obj.storeFileURL, session);

    //take the json (the order) and subtract each field to the relative in store
    Object.keys(json.products).forEach(function(k){
        store.products[k].quantity_available -= json.products[k].quantity;
        if (store.products[k].quantity_available <= store.products[k].max_capacity*0.15){
            var units = store.products[k].max_capacity - store.products[k].quantity_available;
            store.products[k].quantity_available = store.products[k].max_capacity;
            console.log("Product: ", store.products[k].name, " refunded of ", units ," units");
        }
    });

    //update the file
    store = JSON.stringify(store);
    await uploadJSON(store, obj.storeContainerURL, "store.json", session);
}
```

The graph below shows the final interaction through the application process that involves the related entities. Particularly, it offers a viewpoint on how the Solid component has been involved. Indeed, as main component, It needs to be open and correctly implemented with all other entities, otherwise the flow of execution should not guarantee all the features explained in the 2nd chapter.

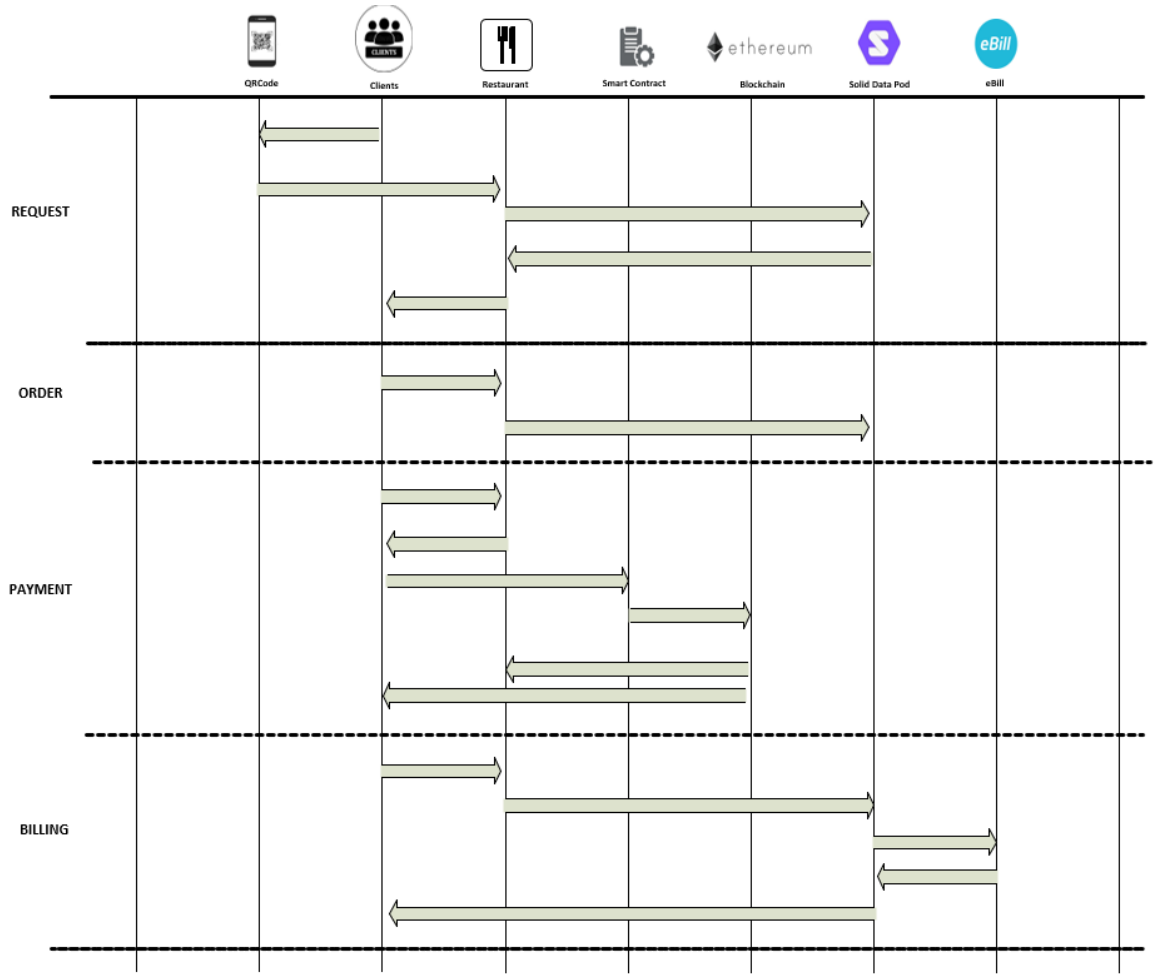


Figure 3.7 – Solid graph interactions

3.2.4 WebApp

During the development of the graphical interface, we tried to take care of the flow and the UX, rather than the graphics themselves. The WebApp is very simple and effective to highlight the simplicity of the steps to be faced.

Once we have receipted the Menu and logged in as "Table #X" we will immediately check if there is an order for the table or an empty order in the Order Pod. The use of these containers (Pods) allows us to not implement any cookie mechanism and session management, because at the first occupation of the table, and therefore at the first sending of the order, the system creates an order for that table. Then, the refresh of the page, or in any case a new scan of the QR-code, will return to us any active order or will initialize a new one.

It is possible to switch between the menu and the orders made, going to change the order from both the two sections.

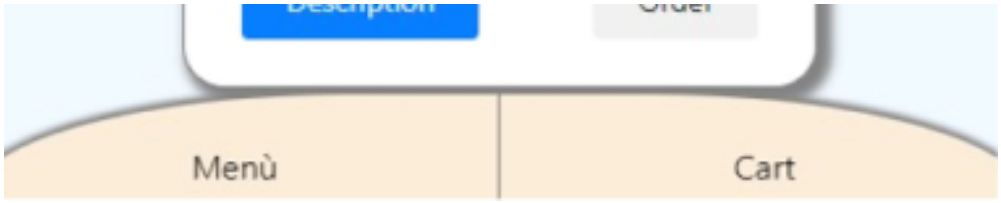


Figure 3.8 – Menu & Cart switches

The main section offers the list of products on the menu. It is necessary to remember that the render of the menu includes a phase of verification of the products available within the warehouse, a JSON file.

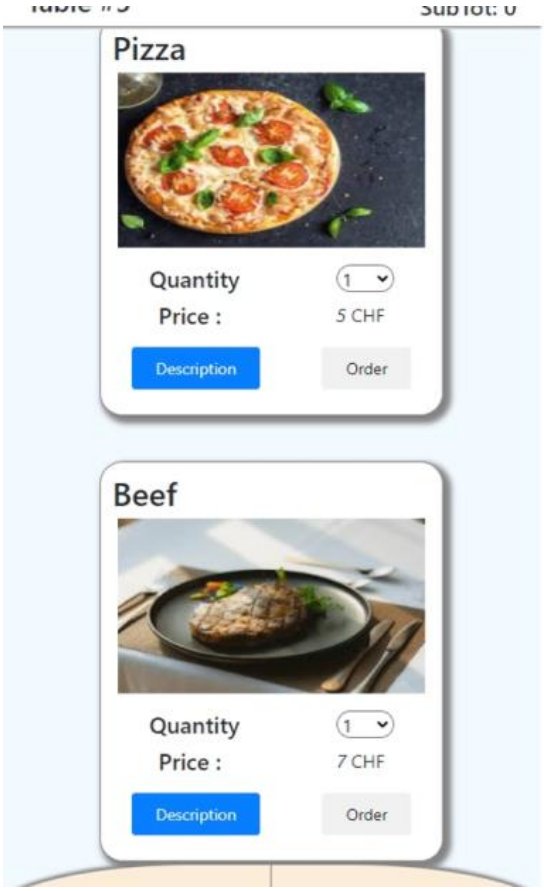


Figure 3.9 – List of products

We will be able to select the quantity we want to order for a product, and immediately view a subtotal of the selection. In addition, each product has an interactive description by clicking on the "Description" button, which will open a pop-up.

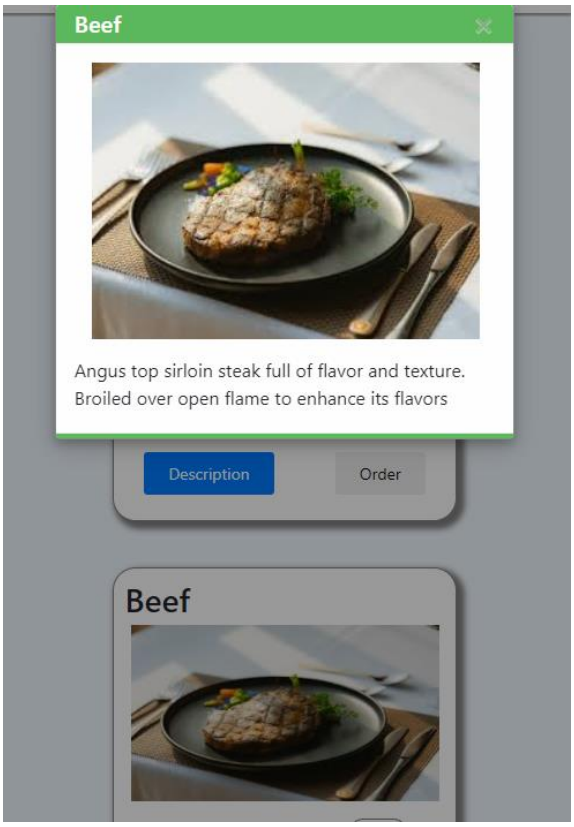


Figure 3.10 – Product Description

The cart offers the possibility to update the order by dynamically increasing the quantities for the products already selected and those already ordered. Obviously, guaranteeing that we cannot decrease this value or remove the product from the order after having already ordered it.

Figure 3.11 – Quantities Update

In addition, from the cart we can close the order, asking for the invoice. In this case, a statement will be shown in pdf format, from which the customer can first make the transaction and then wait for verification by the restaurant.

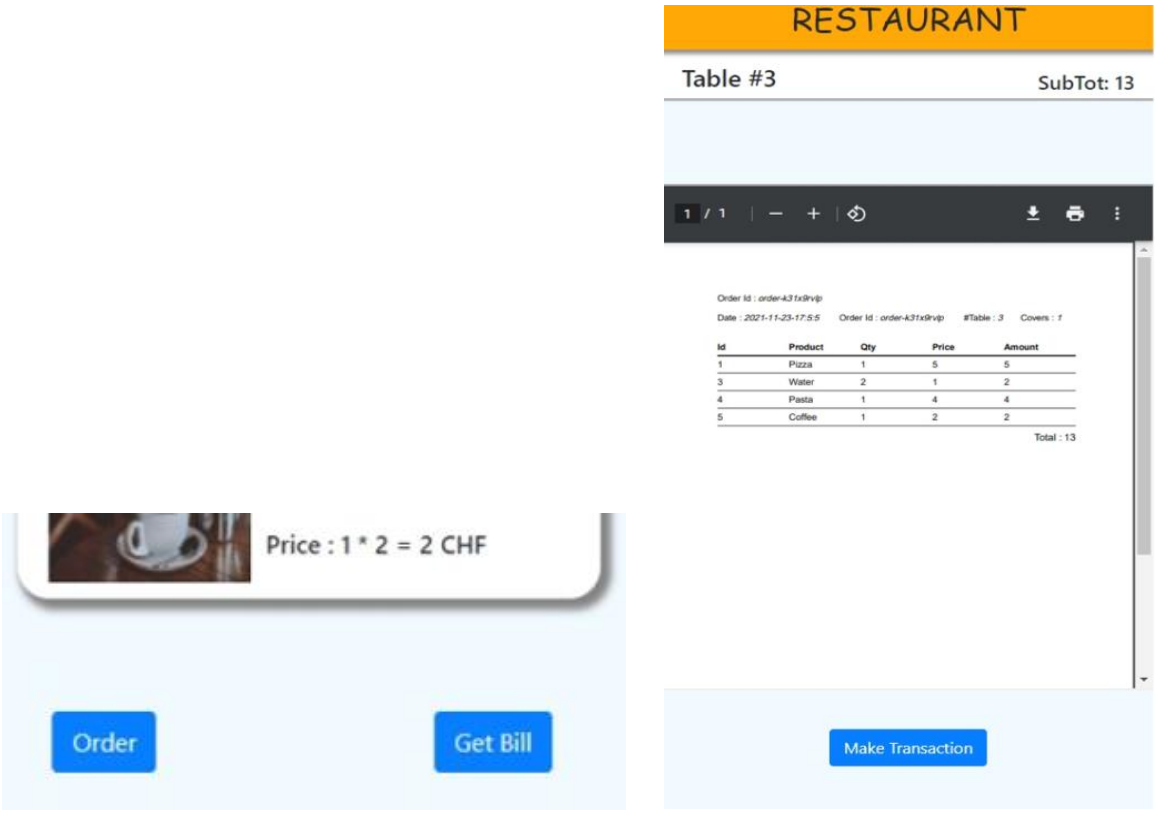


Figure 3.12 – Get Bill phase

In the end, the restaurant will produce a final invoice in pdf format which, thanks to the presence of a QR code, guarantees the customer's exit with a simple verification of the (final) QR code at the checkout, avoiding queues for payment.

Figure 3.13 – Billing phase

Although the cover is offered free of charge to everyone in Switzerland, we preferred to include this item as a point of attention to know how many customers are sitting at the table. This will help in the preparation of the table.

A constraint related to the use of the interface is that it is currently necessary to use a Google Chrome browser that allows the MetaMask extension for payment with cryptocurrencies and blockchain. In particular, this configuration is linked to a local context, while in the production environment (and therefore when the user uses their smartphones to interact) there is a need to use the direct MetaMask browser that can guarantee the use of the e-wallet.

3.2.5 Blockchain system

The blockchain system allows the restaurant to ensure that its payments take place within a public ledger, therefore transparent, secure, and confidential (at the level of privacy). It also provides a linking system between the invoices issued. In fact, every transaction activated by the client goes to invoke a smart contract that turns the money to the reserved account of the restaurant, and at the same time saves in a data structure (hash table on the blockchain) the hash of the order and the link where it is possible to find the invoice in pdf format.

Access to the blockchain, or more commonly the openness to the development and integration of dApps in a browser context, is facilitated by the use of the web3js library.

```
const Web3 = require("web3");
```

In particular, this library allows us to dialogue with the blockchain and then invoke its methods and functions. These can be activated after its establishment, where we have to connect to an Ethereum node. In our case, we decided to use Infura[26] as a service for accessing

[26] Infura runs Ethereum nodes and provides access to them via api to eliminate the need to run and update our infrastructure.

Ethereum nodes. After obtaining an API key, and selecting the Kovan[27] testnet (for the test phase), we used the (provided) URL to access the network and the nodes configured by Infura.

```
var web3 = new Web3(`https://kovan.infura.io/v3/APIkey`);
```

The choice of a smart contract was made considering several aspects. First of all, the need to save external data on the blockchain (the transaction itself) required that a smart contract be present. Although this figure weighs down the flow of execution of the restaurant and payment, it turned out to be the only solution for our purpose. As if that were not enough, a call to a smart-contract is more expensive than a direct transaction. In this, we will exploit the blockchain of Lugano, which will guarantee no-commission costs for transactions (fees), if not for the transaction intrinsic to the smart contract that sends the money to the restaurant wallet.

Relying on the Remix platform, we were able to carry out several tests on the development of our smart-contract (Appendix C) and then publish it on the blockchain. As anticipated, the smart contract works in the following way:

- contains the address of the restaurant marked as "payable";
- contains a hash table data structure that allows, in addition to constant access to the data inside, to save the information necessary for our application, or given the hash of an order activate a link to the invoice issued for that order. The assumption here is that the presence of the hash, and therefore of the link for the invoice, are verified by the fact that the payment has taken place, otherwise the smart-contract would not have had the opportunity to save those entries in the data structure;
- allows us to receive and send money;
- activates a policy of reading the data structure only for the restaurateur's account;
- therefore, it allows us to receive money (temporarily guarded by the smart-contract) and forward it to the restaurateur's account.

[27] A public PoA Ethereum testnet that use Parity to provide a stable, secure testnet environment for Ethereum developers.

```
function forwardPayment(uint256 amount, string memory _orderHash, string memory _data) public payable {
    require(msg.value == amount);
    (bool sent, ) = _owner.call{value: msg.value}("");
    require(sent, "Failed to send Ether");
    hashTable[_orderHash] = _data;
}
```

After the development and testing phase of the smart-contract, and after publishing the same on the Kovan testnet, we moved on to the development phase of the interactions with the blockchain through our NodeJS backend. As a consequence of the instantiation of the Web3js library, it was also necessary to report the JSON encoding of the smart-contract, or rather, through Remix we could report the ABI of the smart-contract, and therefore the interface containing its methods, specification of parameters to functions, data structures, etc. . This was necessary because we were able to activate the variable that identifies the smart-contract through the address where it was published, and its ABI.

Once connected to the blockchain, and established the interface for the contract we developed, we start to insert the terms for the creation of the mechanism of transactions that will flow into from the restaurant system.

The *getPayment* method of our application activates the payment process of our customer who will then have to interact with MetaMask (next section) and his wallet through the browser. This method manages the wait for confirmation of payment, and then confirmation of receipt of the transaction on the blockchain. Then, activate the "invoice completion" mechanism discussed above, that is, use the template for paid invoices and enter the user's payment details and the QRcode with the link to the Pod for paid invoices where the invoice will be moved.

```
async function getPayment(bill, session) {
    bill = JSON.parse(bill);
    const done = await checkPayment(bill.order.hash, bill.blockchain_transaction_id);
    if (done == false) {
         // console.log("Payment not executed");
        return false;
    }
    console.log("Payment executed");
    await deleteFileFromPod(obj.activeOrderContainerURL + `order-table-${bill.order.table_number}.json`,
session);

    bill["payed"] = true;
    bill.src =  obj.billPayedURL + bill.order.hash + ".pdf";
    bill["hash_bill"] = hashCode(JSON.stringify(bill));

    //create QRpayed
    //create link request to encode
    var qrCodeText = new URL(obj.billPayedURL + `${bill.order.hash}.pdf`).toString();

    await qr.toFile(`./utils/img/QR-tables/temp.png`, qrCodeText, {
        errorCorrectionLevel: 'H',
        // version: "",
        type: 'image/png',
        quality: 0.95,
        margin: 3,
        color: {
         dark: '#000000ff',
         light: '#ffffffff',
        },
    });

    //insert in bill container
    makeBillPDF(bill);
    deleteFileFromPod(obj.billToPayURL + `${bill.order.hash}.pdf`, session);
    await uploadFileFromPath(`./utils/temp.pdf`, "application/pdf", obj.billPayedURL, bill.order.hash +
".pdf", session);
    await createResourceSpecificPublicRulesPolicies(obj.billPayedURL +`${bill.order.hash}.pdf`,
"billPayed", { read:true }, session);

    return [true, bill.src];
}
```

Relative to the payment process itself, given the hash of the order considered, the *checkPayment* function will use the instance of the smart-contract (created as before) to go to call the *getData* method of the smart-contract which, using the address of the restaurant wallet, will verify that the hash of the order passed as a parameter to the *checkPayment* function returns true or false. This, depending on whether the string or the link of the invoice paid linked to that order exists and linked to that hash, so it exists in the Pod.

```
async function checkPayment(hashOrder, transactionID){
    try {
         // Instantiate web3 with HttpProvider
        var web3 = new Web3(`https://kovan.infura.io/v3/09025260fc864cd09d057f68852e45ea`);

         //################################## Contract Case ##################################
        var ERC20ABI = [ …
        ];
        var instanceContract = new web3.eth.Contract(ERC20ABI,
"0x396DC917E64909Dfd3081FE1Ac461c14b87Dc6a8");
        await instanceContract.methods
            .getData(hashOrder.toString())
            .call({ from: "0x684F22798FEf8dDcaCB8278447703787293cEe07" }, function (err, res) {
                if (err) {
                    console.log("An error occured", err)
                    return
                }
                return res.toString() != "";
            });
    }
    catch (error) {
        console.log(error)
    };
}
```

```
    function getData(string memory _orderHash) view public onlyOwner returns(string memory data) {
        return hashTable[_orderHash];
    }
```

The Kovan testnet was chosen as one of the most common and reliable. However, the real purpose was to connect our system to the blockchain movement started in Switzerland, and in particular in the city of Lugano.

The newly designed Lugano Blockchain will be part of our project offering the following advantages [64][65][66]:

- non-profit;
- low energy expenditure;
- zero commissions for citizens and traders;
- an institutional network that promotes, therefore, Proof of Authority compared to PoW;
- affordable and advantageous pricing plans;
- one node for each partner to ensure governance deployment;
- the use of (pre-existing) LVGA points payment tokens.

3.2.6 MetaMask

The use of MetaMask, through the installation of the appropriate extension for the browser, therefore allows us to accept and make transactions, payments. The connection to the Kovan testnet network allowed us to test our smart contract, and to be able to test and learn the mechanism of transactions.

The web page script again uses the web3js library, which once again allows us to invoke the method of a smart-contract. In our case, allow the sending of money to the restaurant with the *forwardPayment* method of the smart-contract.

```
window.web3 = new Web3(ethereum);
try {
    await ethereum.enable();
    var accounts = await ethereum.request({ method: 'eth_requestAccounts' });
    var ERC20ABI = […
    ];
    var instanceContract = new web3.eth.Contract(ERC20ABI, "0x396DC917E64909Dfd3081FE1Ac461c14b87Dc6a8");
    await instanceContract.methods
        .forwardPayment(wei, bill.order.hash.toString(), bill.src.toString())
        .send({ from: accounts[0], value: wei }, function (err, res) {
            if (err) {
                console.log("An error occured", err)
                return
            }
            console.log('Payment successful', res);
```

```
            console.log(bill.order.hash);
            bill.blockchain_transaction_id = res;
            document.getElementById("paymentSending").value = JSON.stringify(bill);
        })
} catch (error) {
    console.log(error);
}
```

As mentioned in the previous section, every participant in the blockchain, every activity, must promote the chain by activating a node. Once the SUPSI node was activated, we were able to proceed with its configuration in MetaMask, through the "Add a network" item that allows us to set the configuration parameters of the node considered, and dialogue with the Lugano blockchain.

Note: during the test phase, we relied on online interfaces (Ethdrop, Gitter) that allowed the free retrieval of ETH, with which we were able to carry out real simulations of our application through our wallet.

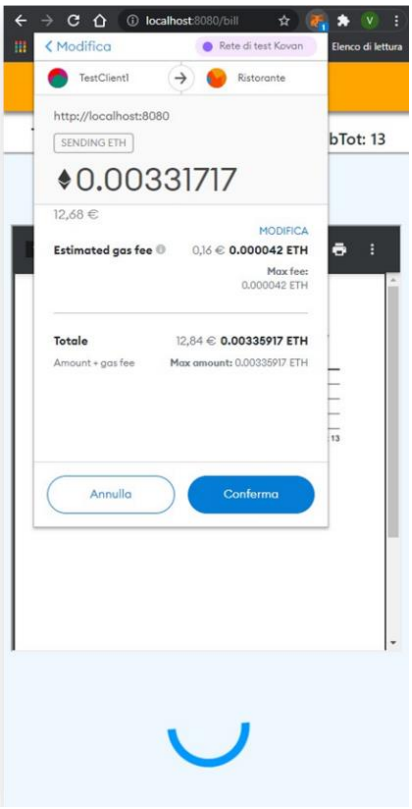


Figure 3.14 – MetaMask experience

3.2.7 QR-code system

By implementing a QR-code system, we are going to guarantee 2 advantages to the restaurant:

1) allow the user to log in without activating the "protocol" of the waiter's call. This, especially to give a new dimension to the restaurant. Thanks to the QR code we can access the web interface of the restaurant that will return from the specific Pod the menu, with all the products currently available, but above all logging in as "Table #3" so that we can connect the orders to the table from which the customer sends the command, or where he is sitting. Previously, this happened with the entry of the customer who with the help of the staff, was made to sit at the first free table, after which he had to ask for the "card of the day" and make orders to the waiter, or call him back to add or remove drinks from his order;

2) guarantee the verification of the payment at the exit by the customer. In particular, when the customer concludes the payment on the WebApp, this will return a pdf representing the order at the restaurant, and then "the receipt", with also a QR-code that points to a Pod containing the invoices paid. If the invoice exists in the Pod, it means that the payment has been successful, it has been received, and then the system has inserted the invoice paid within the relative Pod. In this way, through a scan of the QR-code on the electronic invoice in .pdf format, the cashier will be able to verify that the customer has concluded and paid the correct amount for his experience.

In the development phase, this resulted in the need for a link to be inserted in the QR code. The support library is *qrcode*, available as an add-on module for NodeJS. This allows through the *toFile()* function to be able to save in .jpeg format or .png the QR-code, set through the parameters of the function. These are precise:

1) the path where to save the QR code;
2) the link to be included in the QR code;
3) a series of parameters related to the QR code such as the colors to be used, the level of image quality, the margins, and more.

```
const qr = require("qrcode");
…
…
var qrCodeText = `https://rwa-test-heroku.herokuapp.com/${i+1}`;

qr.toFile(`./utils/img/QR-tables/qr-table${i+1}.png`, qrCodeText, {
        errorCorrectionLevel: 'H',
        type: 'image/png',
        quality: 0.95,
        margin: 3,
        color: {
                dark: '#000000ff',
                light: '#ffffffff',
        },
});
```

By setting in a for loop the number of tables of the activity, we can get as many QR-codes as there are tables, and then apply them to the tables themselves. The scene that will present itself will then be the customer(s) who once seated at the table will be able to access the menu and place orders (for the table) simply by scanning the QR-code.

Figure 3.15 – Table Identification through the QR-code

Implicitly, this allows both the restaurateur to be able to eliminate paper menus and avoid continuous calls to staff, and to offer the possibility to the customer to be able to use their mobile devices for interactions with the restaurant such as ordering from the menu. We have tried to take advantage of the fact that the mobile device is now a must-have, as well as

technological advancement allows more interactions, and in our case in particular electronic payments. It turns out to be a very efficient solution if compared with another hypothesis: just imagine a scenario where each table implements a monitor to activate the same functionality now offered by the customer's mobile phone, with the evidence of the fact that the monitor can be expensive, but above all expensive in the case of an activity where the number of tables (and therefore the monitors installed) is very high. Therefore, it is very efficient a single device supplied "natural" to the customer.

Regarding the second point of advantage, it should be noted that the imposition of a QR code on the final invoice sent by the restaurant to the customer's device, allows to speed up the customer's exit from the restaurant. In other words, this mechanism avoids the formation of queues at the exit, facilitating and speeding up the verification on the receipt of payment (which we remember to be in blockchain, and therefore the need for a few moments for the confirmation of the transaction by the nodes of the network). A simple scan will avoid different customer-cashier interactions. In a further futuristic scenario, the figure of the cashier could be literally removed, going to ask the user to show the QR on the bill to a scanner placed near the exit door, guaranteeing its opening in case of confirmed checkout.

Figure 3.16 – QR-code on a digital bill

3.2.8 eBill system

To insert the ability in the application to be able to produce a digital invoice based on the order placed by the customer, from his data and those of the restaurant, it was necessary to use the *pdfkit* library as an additional module to NodeJS. We did not rely directly on the Swiss eBill system, as it was less suitable for the context of a thesis and more complex. However, it turned out to be just as interesting, easier to use, and with features really similar to the eBill context (Section 2.1.6).

From the *src* folder, we can find the file *pdf-kit-ext.js* as an extension of the *pdfkit* library, or rather we have tried to set up a template (as in Figure 3.11) for the billing system, avoiding starting each time from the structuring of the final document.

```
const PDFDocument = require('./pdf-kit-ext');
…
…
function makePreBillPDF(bill){
    let pdfDoc = new PDFDocument({size: 'A4', modifying:false });
    pdfDoc.pipe(fs.createWriteStream(`./utils/temp.pdf`));

    //Structure
    pdfDoc
        .fontSize(11.5)
        .text("Order Id : ", { align: 'left', continued:true })
        .text(bill.order.id_order, {oblique : true});
        …
        …
    pdfDoc.end();
}

function makeBillPDF(bill){
    //insert in pdf
    let pdfDoc = new PDFDocument({size: 'A4', modifying:false });
    pdfDoc.pipe(fs.createWriteStream(`./utils/temp.pdf`));
        …
        …
    //insert the QR Code within the PDF
    // Fit the image in the dimensions, and center it both horizontally and vertically
    pdfDoc.image(`./utils/img/QR-tables/temp.png`, {align: 'center', valign: 'center'});
```

```
    pdfDoc.end();
}
```

In the context of the restaurant application, the call to the library and the creation of the digital document takes place in 2 moments:

1) when the user decides to conclude the order, and requests the account;
2) when the user completes the payment.

In the first case, we have the *makePreBillPDF()* function that maintains a simpler document structure than that of the final invoice, as the customer's need at the time of requesting the account for the table is simply to have a summary of the products consumed, with the display of the total to be paid.

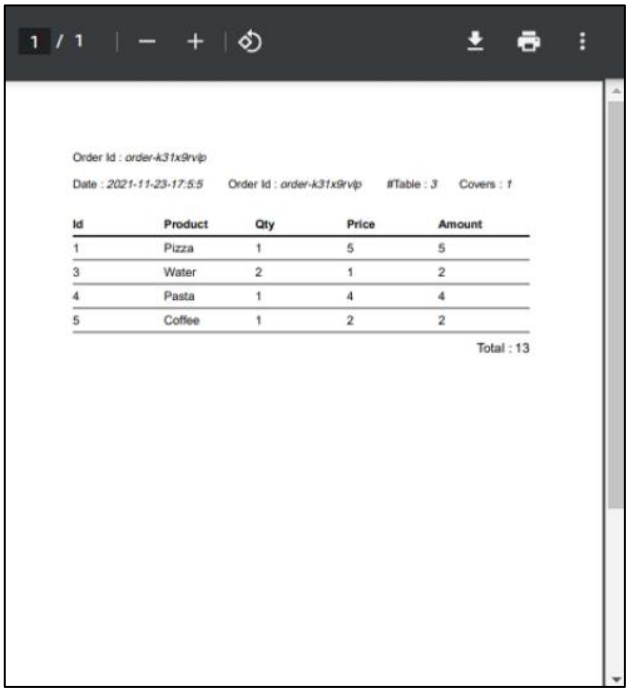


Figure 3.17 – Pre-invoice

Instead, with the final invoice obtained with the *makeBillPDF()* function, we have an official document, complete with all the necessary information such as:

- restaurant data, such as an address, VAT number, etc.;
- invoice hash;
- order hash;
- covers;
- table number;

- selected payment method;
- any details related to the transaction on the blockchain;
- the QR code for verification of payment.

Given these considerations, it is necessary to include a final aspect that links the invoices and digital documents produced to the context of Solid. In fact, the management of the documentation takes place within specific Pods. In particular, when the "pre-invoice" is issued, it is inserted into the Pod of the "invoices still to be paid", while the final invoices, which should be remembered can be issued only after receiving the payment, are saved in a Pod for the "paid invoices".

Chapter IV

CONCLUSIONS

"It's better to do less than you hoped than nothing at all.
No zero days."
~ James Clear

4.1 Considerations

The pandemic is forcing the SME sector to rethink how its productivity and profitability are ensured. For instance, "take away" is proving to be a model that is likely to remain in post-covid too. Also, accounting and financial reporting regulations force the correct maintenance of all paper receipts relative to financial transactions, which is completely missing in the emerging crypto world. This transformation requires technology to help minimize administrative costs and, thanks to new forms of automation, ensure the survival of the sector.

Through a shared transactional management, the level of digitalization of processes is increased, reducing the bureaucratic and administrative activities of the company and therefore the costs related to manual back office activities, increasing efficiency, transparency, usefulness and competitiveness of SMEs. The challenge of the research is related to the integration of decentralized web principles with blockchain environments and decentralized cryptographic architectures, to improve the problems related to the efficiency of the shared management of a large amount of information (in our case the digital copies of the receipts of each individual transaction, the details of customers and warehouse items). Companies are constantly looking for solutions to take advantage of new technologies and optimize "collaboration" and

"coordination" between entities within the company, as well as between entities within their supply chain.

The study conducted aims to investigate how to integrate the world of decentralized crypto transactions (smart contract and stable coins) with the accounting and financial reporting regulations using emerging paradigms of a "decentralized web" and therefore of a private area for storing data under full control of the owner. Specifically, the objective of the thesis is to evaluate a way to enrich traditional blockchain payments with decentralized document management features capable of storing any form of proof of transactions.

The presentation of the thesis work sought to clarify the concept of using a shared DLT platform that automates corporate accounting flows (liquidity, creditors, debtors and wages), integrating information deriving from movements in traditional currency (FIAT) and cryptographic currency (stable coins), and adding these transactions in the balance accounting systems. Emerging projects are proposing themselves as solutions for decentralized management of private data. The one we considered is Solid, sponsored by Tim Barnes Lee, where the idea was to evaluate if these systems can be linked and integrated with blockchain payment transactions. In a few years, paying in real-time will be the norm and digital currencies will allow direct (P2P) and paperless operations, further transforming payment methods and consequently business management tools. This study is the first step for the future development of a decentralized ERP system for typical corporate transactional management.

4.2 Perspectives

The project aims to provide SMEs with a tool able to harmonize document and accounting transactions and automate repetitive manual processes, so that management can focus its resources on the added value of the company and grow organically with a greater Capex available. The goal is to take advantage of the new forms of transactions and direct payments, allow accounting management of new digital currencies, reducing many data entry and control activities today carried out manually. Also, the project will allow to consolidate the B2C model, expanding to B2B activities related to the integration of the central functions of the system with other systems, through the development of special APIs.

The product is expected to have a direct impact on the secretarial, back office, budgeting and reporting costs, with a reduction in operational staff and administrative audit costs. The quantifiable goals are:

- implementation of direct payment functions with related management of intra-blockchain digital receipts;
- off-chain management of private corporate data (transactions, receipts and warehouse) within private decentralized agents in sync with individual on-chain transactions;
- 60 % reduction in paper documentation;
- ability to produce reports and financial statements at the transaction level.

The space for extensions and future works considers a premise: the thesis work in which we took part, is a "prototyping" of a larger project that will be activated in Switzerland. In fact, the Thesis, and my path in Switzerland, concerned not only an Internship between universities, but also a real project that will be proposed to the specific Swiss bodies. In particular, the project has created a dialogue between the needs of a restaurateur and the ERP system to which has entrusted the task of accounting system, and from which he will try to take advantage for the company and its revenues. This premise is necessary because several meetings have followed that have given way to make future evaluations. For example, some points of development could be:

- the constitution of a modular system, able to reason in "modules" and therefore frame the context as a set of independent blocks;
- insert the possibility of adding notes to the order placed, how to customize the chosen dish or indicate intolerances and allergies;
- tracking of products, authenticating all the paths they have taken from the manufacturer to the supplier, arriving at the product served;
- the choice on the inclusion of a product within the menu based on definable periods, taking advantage of the periodicity of the products offered by the restaurant;
- register the customer, to identify him in the system guaranteeing services suited to his preferences;
- implement a multi-screen system, which allows the different sections of the restaurant (kitchen, dining room, bar) to always have a view of the workflow to follow;
- offer the possibility of being able to divide the subtotal among the customers of the table.

The idea that led to the reasoning conducted on different points to be evaluated to be able to implement and feel directly was shown. This assessment, carried out in collaboration with the professors and the restaurateur, was also welcomed for the emerging realities in the Swiss ecosystem.

4.3 Engagement

We would like this thesis to be the descriptive means by which we were able to conclude the Master's degree at the University of Pisa having managed to live a foreign experience (Switzerland - Lugano) that allowed me to live in a school and work context very different from those we have met so far. We tried to reconcile the thesis work with the objective of the Master's Degree course we undertook in Pisa by choosing the "ICT Solutions Architect" study plan. On the other hand, it was necessary to know, study and implement different components for the prototype of the Thesis.

Acknowledgments

I'd like to thank the *University of Pisa* for facilitating this wonderful experience in Switzerland by awarding me a scholarship for mobility for Theses abroad.

I want to thank *SUPSI* for welcoming me into a new environment without missing anything, and allowing me to start new school-work relationships with people of substance in the IT sector.

I would like to thank *Prof. Laura Ricci* for giving me freedom of choice on the conduction of the Thesis work. It was a great gesture of trust in me.
And I thank *Prof. Giacomo Poretti*, for having welcomed me as a student, for having introduced me to a reality greater than the one I was used to, and for having guided me on the most important steps that characterized the work.

I have to thank my family for supporting me in all aspects related to this school path made of ups and downs. Regardless of how these efforts will then be realized, I am very happy to know that I have you to rely on.

I'd like to thank all my friends. They gave me moments of joy when I needed it most, to give me strength and never leave this path.
Although life is full of misunderstandings, I am sincere in telling you that the esteem I feel for you is very great and I know that I have, in you all, a point of reference, a family.

BIBLIOGRAPHY & SITOGRAPHY

[1] F. G. Papi, J. F. Hübner, and M. De Brito, “Instrumenting accountability in mas with blockchain,” *CEUR Workshop Proc.*, vol. 2051, pp. 20–34, 2018.

[2] S. Ali, G. Wang, B. White, and R. L. Cottrell, “A Blockchain-Based Decentralized Data Storage and Access Framework for PingER,” *Proc. - 17th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. 12th IEEE Int. Conf. Big Data Sci. Eng. Trust. 2018*, pp. 1303–1308, 2018, doi: 10.1109/TrustCom/BigDataSE.2018.00179.

[3] V. Ranade, S. Shrivastava, and S. Sharma, “Generalised Design of Efficient Supply Chain Management System and Enterprise Resource Planning [ERP] System , Using Two Layer Blockchain Setup on Hyperledger Fabric and Ethereum,” *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 6, no. 7, pp. 6713–6721, 2018, doi: 10.15680/IJIRCCE.2018.0607022.

[4] D. Pramulia and B. Anggorojati, “Implementation and evaluation of blockchain based e-voting system with Ethereum and Metamask,” *Proc. - 2nd Int. Conf. Informatics, Multimedia, Cyber, Inf. Syst. ICIMCIS 2020*, pp. 18–23, 2020, doi: 10.1109/ICIMCIS51567.2020.9354310.

[5] S. Ferretti and G. D’Angelo, “On the Ethereum blockchain structure: A complex networks theory perspective,” *Concurr. Comput. Pract. Exp.*, vol. 32, no. 12, 2020, doi: 10.1002/cpe.5493.

[6] M. BASILE, “A blockchain-based approach for digital archives: a decentralized Ethereum application for the safeguarding of the cultural heritage.,” 2018, [Online]. Available: https://etd.adm.unipi.it/t/etd-11212018-010903/.

[7] D. Calvaresi, A. Dubovitskaya, J. P. Calbimonte, K. Taveter, and M. Schumacher, “Multi-agent systems and blockchain: Results from a systematic literature review,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10978 LNAI, no. September, pp. 110–126, 2018, doi: 10.1007/978-3-319-94580-4_9.

[8] A. V. Sambra *et al.*, “Solid: A Platform for Decentralized Social Applications Based on Linked Data,” *MIT CSAIL Qatar Comput. Res. Institute, Tech. Rep.*, 2016, [Online]. Available: http://emansour.com/research/lusail/solid_protocols.pdf.

[9] H. Leith, “The father of the internet just created a Solid alternative. Here’s how it works | Stuff.co.nz,” Oct. 04, 2018. https://www.stuff.co.nz/technology/107587496/how-sir-tim-bernerslees-alternative-to-traditional-internet-would-work (accessed Jan. 28, 2022).

[10] K. G. Orphanides, “How Tim Berners-Lee’s Inrupt project plans to fix the web | WIRED UK,” *Wired UK*, no. March/April, p. 3, Feb. 2019, Accessed: Jan. 28, 2022. [Online]. Available: https://www.wired.co.uk/article/inrupt-tim-berners-lee.

[11] T. Cai, Wuhui Chen, and Y. Yu, “BCSolid: A Blockchain-Based Decentralized Data Storage and Authentication Scheme for Solid,” in *Communications in Computer and Information Science*, 2020, vol. 1156 CCIS, pp. 676–689, doi: 10.1007/978-981-15-2777-7_50.

[12] S. Shelby, “Self-Hosted Personal Data is Key to Four Promising Facebook Alternatives | ProgrammableWeb,” Jul. 02, 2019. https://www.programmableweb.com/news/self-hosted-personal-data-key-to-four-promising-facebook-alternatives/analysis/2019/07/02 (accessed Jan. 28, 2022).

[13] A. Sambra, H. Story, and T. Berners-Lee, “WebID 1.0,” 2014. Accessed: Oct. 26, 2021. [Online]. Available:

https://www.w3.org/2005/Incubator/webid/spec/identity/.

[14] SF Examiner, “Inrupt JavaScript Client Libraries — Inrupt JavaScript Client Libraries.” https://docs.inrupt.com/developer-tools/javascript/client-libraries/ (accessed Oct. 26, 2021).

[15] O. Lassila, “Embracing ‘Web 3.0,’” 2007.

[16] “An overview of Web 1.0, Web 2.0, and Web 3.0 - The San Francisco Examiner,” 2022. https://www.sfexaminer.com/marketplace/an-overview-of-web-1-0-web-2-0-and-web-3-0/amp/ (accessed Feb. 12, 2022).

[17] B. W. Wahlster and A. Dengel, “Technology Radar Feature Paper Edition II / 2006,” no. June, 2006.

[18] V. Barassi and E. Treré, “Does Web 3.0 come after Web 2.0? Deconstructing theoretical assumptions through practice,” *New Media Soc.*, vol. 14, no. 8, pp. 1269–1285, 2012, doi: 10.1177/1461444812445878.

[19] Aishwarya Banik, “What are the Advantages and Disadvantages of Web 3.0?,” Jan. 24, 2022. https://www.analyticsinsight.net/what-are-the-advantages-and-disadvantages-of-web-3-0/ (accessed Dec. 16, 2021).

[20] X. Zhang and D. Vogel, “Electronic Commerce in Web 3.0: The Evidence from Emerging Economics,” *Int. J. Comput. Sci. Inf. Technol.*, vol. 3, no. 3, pp. 240–254, 2011, doi: 10.5121/ijcsit.2011.3317.

[21] R. Rudman and R. Bruwer, “Defining Web 3.0: Opportunities and challenges,” *Electron. Libr.*, vol. 34, no. 1, pp. 132–154, 2016, doi: 10.1108/EL-08-2014-0140.

[22] N. Kobie, “Q&A: Conrad Wolfram on communicating with apps in Web 3.0,” *Analysis & Insight*, vol. 2013. 2010, Accessed: Feb. 12, 2022. [Online]. Available: https://www.itpro.co.uk/621535/qa-conrad-wolfram-on-communicating-with-apps-in-web-30.

[23] D. Kreps and K. Kimppa, “Theorising Web 3.0: ICTs in a changing society,” *Inf. Technol. People*, vol. 28, no. 4, pp. 726–741, 2015, doi: 10.1108/ITP-09-2015-0223.

[24] Y. Horban, A. Humenchuk, O. Karakoz, O. Koshelieva, and I. Shtefan, “Application of web 3.0 technologies in distance education (by levels of higher education),” *Laplage Em Rev.*, vol. 7, no. Extra-B, pp. 575–586, 2021, doi: 10.24115/s2446-622020217extra-b974p.575-586.

[25] F. Almeida, J. D. Santos, and José A. Monteiro, “E-COMMERCE BUSINESS MODELS IN THE CONTEXT OF WEB 3.0 PARADIGM,” *Scientist*, vol. VI, no. September 2010, pp. 1–12, 2006.

[26] F. L. F. Almeida and J. M. R. Lourenço, “eCreation of value with Web 3.0 technologies,” *Proc. 6th Iber. Conf. Inf. Syst. Technol. Cist. 2011*, 2011.

[27] A. Upadhayaya, “Electronic Commerce and E-wallet,” vol. I, no. March, pp. 37–41, 2012.

[28] M. Salah Uddin and A. Yesmin Akhi, “E-Wallet System for Bangladesh an Electronic Payment System,” *Int. J. Model. Optim.*, vol. 4, no. 3, pp. 216–219, 2014, doi: 10.7763/ijmo.2014.v4.376.

[29] M. Anshari, M. A. Arine, N. Nurhidayah, H. Aziyah, and M. H. A. Salleh, “Factors influencing individual in adopting eWallet,” *J. Financ. Serv. Mark.*, vol. 26, no. 1, pp. 10–23, 2021, doi: 10.1057/s41264-020-00079-5.

[30] N. I. Mohamad Anuar *et al.*, “The barriers towards the adoption of e-wallet payment system,” *Int. J. Eng. Res. Technol.*, vol. 13, no. 11, pp. 3772–3777, 2020, doi: 10.37624/ijert/13.11.2020.3772-3777.

[31] S. Teng and K. W. Khong, “Examining actual consumer usage of E-wallet: A case study of big data analytics,” *Comput. Human Behav.*, vol. 121, no. October 2020, p. 106778, 2021, doi: 10.1016/j.chb.2021.106778.

[32] “Cos’è MetaMask? Il modo più semplice per utilizzare le dApp.” https://academy.bit2me.com/it/cos%27è-metamask-il-modo-più-semplice-per-usare-dapps/ (accessed Jan. 07, 2022).

[33] J. Condemi, “E-wallet: pagamenti sicuri con il portafoglio elettronico - Pagamenti Digitali,” 2020. https://www.pagamentidigitali.it/instant-payment/e-wallet-pagamenti-sicuri-con-il-portafoglio-elettronico/ (accessed Jan. 07, 2022).

[34] “N26, rated the #1 bank in Italy by Forbes — N26.” https://n26.com/en-it (accessed Jan. 07, 2022).

[35] S. Undale, A. Kulkarni, and H. Patil, “Perceived eWallet security: impact of COVID-19 pandemic,” *Vilakshan - XIMB J. Manag.*, vol. 18, no. 1, pp. 89–104, 2021, doi: 10.1108/xjm-07-2020-0022.

[36] M. K. Chang, W. Cheung, C. H. Cheng, and J. H. Y. Yeung, “Understanding ERP system adoption from the user’s perspective,” *Int. J. Prod. Econ.*, vol. 113, no. 2, pp. 928–942, 2008, doi: 10.1016/j.ijpe.2007.08.011.

[37] V. Botta-Genoulaz and P. A. Millet, “An investigation into the use of ERP systems in the service sector,” *Int. J. Prod. Econ.*, vol. 99, no. 1–2, pp. 202–221, 2006, doi: 10.1016/j.ijpe.2004.12.015.

[38] K. Mudit, “All That You Need to Know About Blockchain and ERP Integration,” 2020. https://blockchain.oodles.io/blog/benefits-integration-blockchain-erp/ (accessed Dec. 16, 2021).

[39] “How Will Cryptocurrency & Blockchain Impact ERP Systems | TEC,” Mar. 2021. https://www3.technologyevaluation.com/research/article/blockchain-cryptocurrency-erp.html (accessed Feb. 03, 2022).

[40] A. Cifu, “Why Integrating ERP Systems into Blockchain Is a Great Idea?” https://www.blockchain-council.org/blogs/why-integrating-blockchain-and-erp-is-a-great-idea/ (accessed Feb. 03, 2022).

[41] L. Mearian, “Blockchain integration turns ERP into a collaboration platform,” Jun. 2017. Accessed: Jan. 22, 2022. [Online]. Available: https://www.computerworld.com/article/3199977/blockchain-integration-turns-erp-into-a-collaboration-platform.html.

[42] V. Portale, “Blockchain for business: applicazioni e sviluppi futuri nelle aziende,” Mar. 30, 2021. https://blog.osservatori.net/it_it/blockchain-business-applicazioni-aziende (accessed Jan. 22, 2022).

[43] I. Training, “Issues & Trends,” no. April, pp. 5–7, 2011.

[44] B. Koch, “E-Invoicing / E-Billing ahead,” 2017.

[45] Y. Own, B. Right, I. N. Your, and O. W. N. Home, “YOUR OWN BRANCH RIGHT.”

[46] O. Vincent, O. Folorunso, and A. Akinde, “On consolidation model in e-bill presentment and payment,” *Inf. Manag. Comput. Secur.*, vol. 17, no. 3, pp. 234–247, 2009, doi: 10.1108/09685220910978121.

[47] S. I. X. W. Paper, “Future of Billing.”

[48] M. Mita, K. Ito, S. Ohsawa, and H. Tanaka, “What is Stablecoin?: A Survey on Price Stabilization Mechanisms for Decentralized Payment Systems,” *Proc. - 2019 8th Int. Congr. Adv. Appl. Informatics, IIAI-AAI 2019*, pp. 60–66, 2019, doi: 10.1109/IIAI-AAI.2019.00023.

[49] H. Kołodziejczyk and K. Jarno, “Stablecoin – the stable cryptocurrency,” *Studia BAS*, vol. 3, no. 63. pp. 155–170, 2020, doi: 10.31268/studiabas.2020.26.

[50] L. Ante, I. Fiedler, and E. Strehle, “The influence of stablecoin issuances on cryptocurrency markets,” *Financ. Res. Lett.*, vol. 41, no. June 2020, p. 101867, 2021, doi: 10.1016/j.frl.2020.101867.

[51] K. Georgieva, “The Future of Money: Gearing up for Central Bank Digital Currency,” Feb. 09, 2022. https://www.imf.org/en/News/Articles/2022/02/09/sp020922-the-future-of-money-gearing-up-for-central-bank-digital-currency (accessed Feb. 12, 2022).

[52] A. Lipton, T. Hardjono, and A. Pentland, “Digital trade coin: Towards a more stable digital currency,” *R. Soc. Open Sci.*, vol. 5, no. 7, 2018, doi: 10.1098/rsos.180155.

[53] W. Bossu, N. Che, and J. Kiff, *Behind the Scenes of Central Bank Digital Currency: Emerging Trends, Insights, and Policy Lessons*. 2022.

[54] M. Ferrari, A. Mehl, and L. Stracca, “The international dimension of a central bank digital currency | VOX, CEPR Policy Portal,” Oct. 12, 2020. https://voxeu.org/article/international-dimension-central-bank-digital-currency (accessed Feb. 17, 2022).

[55] K. Rana, “Triple entry accounting system: A revolution with blockchain.” p. https://medium.com/dataseries/triple-entry-account, Mar. 27, 2020, Accessed: Jan. 15, 2022. [Online]. Available: https://medium.com/dataseries/triple-entry-accounting-system-a-revolution-with-blockchain-

768f4d8cabd8.

[56] J. I. Ibañez, C. N. Bayer, and P. Tasca, “REA , Triple-Entry Accounting and Blockchain :,” pp. 1–63, 2021.

[57] Request, “Request Network The future of commerce A decentralized network for payment requests,” pp. 1–24, 2018.

[58] Request Network, “Request Network Yellow Paper The concept of smart financial audits on the Blockchain,” pp. 1–26, 2017, [Online]. Available: https://request.network/assets/pdf/request_yellowpaper_smart_audits.pdf.

[59] D. (no date) ‘Accelerating C. R. with B. : A. multi-path file transfer protocol in I. and F. Shawn Wilkinson, J. L. (2014) ‘Metadisk: Blockchain-Based Decentralized File Storage Application’, Liquid Crystals, 14(2), pp. 573–580. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.692.8781.Elattar, E. E. and ElSayed, S. K. (2020), “Metadisk: Blockchain-Based Decentralized File Storage Application,” *Liq. Cryst.*, vol. 14, no. 2, pp. 573–580, 2014, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.692.8781.

[60] H. Becker, H. Vu, A. Katzenbach, C. H. J. Braun, and T. Käfer, “Monetising Resources on a SoLiD Pod Using Blockchain Transactions,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12739 LNCS, pp. 49–53, 2021, doi: 10.1007/978-3-030-80418-3_9.

[61] M. Ramachandran, N. Chowdhury, A. Third, J. Domingue, K. Quick, and M. Bachler, “Towards Complete Decentralised Verification of Data with Confidentiality: Different ways to connect Solid Pods and Blockchain,” *Web Conf. 2020 - Companion World Wide Web Conf. WWW 2020*, pp. 645–649, 2020, doi: 10.1145/3366424.3385759.

[62] S. Zhiyuan, “ConsenSys launches Rollups for privacy-enabled transactions on Ethereum blockchain with support of Mastercard,” 2021. https://cointelegraph.com/news/consensys-launches-rollups-for-privacy-enabled-transactions-on-ethereum-blockchain-with-support-of-mastercard (accessed Feb. 08, 2022).

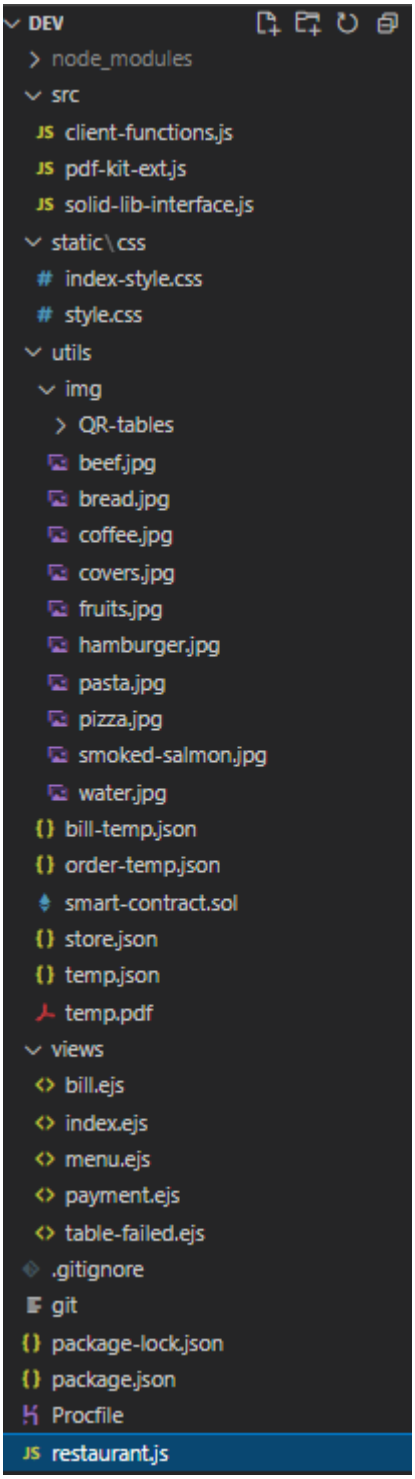
[63] A. Kharoo, “Decentralized finance revolutionized with native push notification service | by Ashvarya Kharoo | Coinmonks | Medium,” Apr. 29, 2021. https://medium.com/coinmonks/decentralized-finance-revolutionized-with-native-push-notification-service-b8948e20a6b1 (accessed Jan. 15, 2022).

[64] Gabriele Testi, “‘3Achain’: così Lugano promuove una blockchain con tripla A,” Nov. 12, 2021. https://innovando.it/3achain-cosi-lugano-promuove-una-blockchain-con-tripla-a/?privacy=updated (accessed Feb. 26, 2022).

[65] “3Achain | La blockchain Autorevole, Affidabile, Accessibile.” https://www.3achain.org/ (accessed Feb. 26, 2022).

[66] A. Tomasicchio, “3Achain: a Lugano presentata la blockchain della città - The Cryptonomist,” Nov. 13, 2021. https://cryptonomist.ch/2021/11/13/3achain-a-lugano-presentata-la-blockchain-della-citta-svizzera/ (accessed Feb. 26, 2022).

Appendix A - Infrastructure directories



Appendix B – Smart-contract

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.0 <0.9.0;

contract RestaurantContract{
    address payable _owner;
    mapping(string => string) hashTable;

    constructor() {
        _owner = payable(msg.sender);
    }

    receive () external payable{}

    modifier onlyOwner(){
        require(msg.sender == _owner, "Not owner");
        _;
    }

    function forwardPayment(uint256 amount, string memory _orderHash, string memory _data) public payable {
        require(msg.value == amount);
        (bool sent, ) = _owner.call{value: msg.value}("");
        require(sent, "Failed to send Ether");
        hashTable[_orderHash] = _data;
    }

    function getData(string memory _orderHash) view public onlyOwner returns(string memory data) {
        // require(msg.sender == _owner);
        return hashTable[_orderHash];
    }
}
```

Appendix C – Solid Interface

```
require("@inrupt/solid-client-authn-node");
require("@inrupt/solid-client");

const fs = require('fs');
const { readFile, writeFile } = require('fs/promises');

################################################################################
--------------------------------LOGIN FUNCTIONS-------------------------------
################################################################################
function createSessionID(){
  var today = new Date();
  var date = today.getFullYear()+'-'+(today.getMonth()+1)+'-'+today.getDate();
  var time = today.getHours() + ":" + today.getMinutes() + ":" + today.getSeconds();
  var result = date+'-'+time+'-'+Math.random().toString(36).substr(2, 9);

  return result;
};

async function logging(sessionID, token){
    sessionID += createSessionID();
    const session = new Session(
      {
        //clientAuthentication: getClientAuthenticationWithDependencies({})
        //public readonly info: ISessionInfo,
        //storage: IStorage;  //set-delete-get
        //tokenRequestInProgress = false,
        //onNewRefreshToken?: (newToken: string) => unknown
        onNewRefreshToken: (newToken) => {
          console.log("New refresh token: ", newToken);
          //token = newToken;
        }
        //defaultMaxListeners: number,
        //errorMonitor: typeof errorMonitor,
      },
      sessionID
    );

    await session.login({
      // Set oidcIssuer to the Solid Identity Provider associated with the credentials.
      oidcIssuer: token.provider,
```

```
      //An already-registered clientId, which identifies your application to the Solid Identity Provider.
      clientId: token.clientId,

      //An already-registered clientSecret, associated to the Client ID. Like a password
      clientSecret: token.clientSecret,

      //An already-registered refreshToken, which your application can use to get an Access Token.
      //Access Tokens allows you to access Resources for which you have been authorized.
      refreshToken: token.refreshToken,

      onNewRefreshToken: (newToken) => {
        console.log("New refresh token: ", newToken);
        //token = newToken;
      }
    })
    .then(() => {
      if (session.info.isLoggedIn) {
        console.log("I m logged in :", session.info.isLoggedIn);

        //The WebID if the user is logged into the session, and undefined if not
        console.log("WebID: ",session.info.webId);
        /**
          * UNIX timestamp (number of milliseconds since Jan 1st 1970) representing the
          * time until which this session is valid.
          */
        console.log("Expiration Date: ",session.info.expirationDate);
      }
      else
        console.log("You are not logged");
    });

    return session;
}

function logOut(session){
  session.logout();
  console.log("Logged out");
}

################################################################################
----------------------------------ACP FUNCTIONS-------------------------------
################################################################################
async function createDatasetACP(resourceURL, session){
  try {
    let myRulesDataset = createSolidDataset();

    if(isDataset(resourceURL)){
```

```
      const savedSolidDataset = await saveSolidDatasetAt(
        resourceURL,
        myRulesDataset,
        { fetch: session.fetch }
      );
      console.log("Rules Dataset created succesfully");
    }
    else
      console.log("The resource is not a Dataset");
  } catch (error) {
    console.log("createDatasetACP");
    console.log(error);
  }
}

async function createAgentRuleACP(resourceUrl, agentOrGroupID, who, resourceName, session){
  try {
    var myRulesDataset = await getDataset(resourceUrl, session);
    // 2. Initialize your new Rules.
    let rule = acp_v3.createRule(`${resourceUrl}#${who}-rule-${resourceName}`);
    // 3. For the rules, specify the Agent to match.
    rule = acp_v3.addAgent(rule, agentOrGroupID);
    // 4. Add your new rules to the SolidDataset.
    myRulesDataset = acp_v3.setRule(myRulesDataset, rule);
    const savedSolidDataset = await saveSolidDatasetAt(
      resourceUrl,
      myRulesDataset,
      { fetch: session.fetch }       // fetch from the authenticated session
    );
  } catch (error) {
    console.log("createAgentRuleACP");
    console.log(error);
  }
}

async function createPolicyACP(resourceUrl, accessModes, who, resourceName, session) {
  try {
    var myRulesDataset = await getDataset(resourceUrl, session);
    let policy = acp_v3.createPolicy(`${resourceUrl}#${who}-policy-${resourceName}`);
    policy = acp_v3.addAllOfRuleUrl(
      policy,
      `${resourceUrl}#${who}-rule-${resourceName}`
    );
    policy = acp_v3.setAllowModes(
      policy,
      accessModes,
    );
    myRulesDataset = acp_v3.setPolicy(myRulesDataset, policy);
```

```
    const savedSolidDataset = await saveSolidDatasetAt(
      resourceUrl,
      myRulesDataset,
      { fetch: session.fetch }      // fetch from the authenticated session
    );
  } catch (error) {
    console.log("createPolicyACP");
    console.log(error);
  }
}

async function createMemberRulesPolicies(resourceUrl, acpUrl, who, resourceName, session) {
  try {
    const resourceWithAcr = await acp_v3.getSolidDatasetWithAcr(
      resourceUrl,  //Container
      { fetch: session.fetch }            // fetch from the authenticated session
     );
    // Add the newly created policy as a Member Policy for the Container
    let changedResourceWithAcr = acp_v3.addMemberAcrPolicyUrl(
    resourceWithAcr,
    `${acpUrl}#${who}-policy-${resourceName}`
    );
    changedResourceWithAcr = acp_v3.addMemberPolicyUrl(
      changedResourceWithAcr,
      `${acpUrl}#${who}-policy-${resourceName}`
    );
    changedResourceWithAcr = acp_v3.addPolicyUrl(
      changedResourceWithAcr,
      `${acpUrl}#${who}-policy-${resourceName}`
    );
    changedResourceWithAcr = acp_v3.addAcrPolicyUrl(
      changedResourceWithAcr,
      `${acpUrl}#${who}-policy-${resourceName}`
    );
    console.log(`acr : ${resourceName}`, acp_v3.acrAsMarkdown(changedResourceWithAcr));
    // 8. Save the Resource with its ACR.
    const updatedResourceWithAcr = await acp_v3.saveAcrFor(
      changedResourceWithAcr,
      { fetch: session.fetch }          // fetch from the authenticated session
    );
  } catch (error) {
    console.log("createMemberRulesPolicies");
    console.log(error);
  }
}

################################################################################
--------------------------------ACP RESOURCE FUNCTIONS------------------------------
```

```
################################################################################
async function createResourceSpecificRulesPolicies(resourceUrl, title, accessModes, agentOrGroupID, session) {
  try {
    // 1. Fetch the SolidDataset with its Access Control Resource (ACR).
    let resourceWithAcr = await acp_v3.getFileWithAcr(
        resourceUrl,
        { fetch: session.fetch }          // fetch from the authenticated session
    );
    // 2. Create the Resource-specific Rule.
    let resourceRule = acp_v3.createResourceRuleFor(resourceWithAcr, title+"-rule");
    resourceRule = acp_v3.setAgent(resourceRule, agentOrGroupID);
    // 3. Create the Resource-specific Policy.
    let resourcePolicy = acp_v3.createResourcePolicyFor(
      resourceWithAcr,
      title+"-policy",
    );
    resourcePolicy = acp_v3.addAllOfRuleUrl(
      resourcePolicy,
      resourceRule
    );
    // 5. Specify the access modes for the policy.
    resourcePolicy = acp_v3.setAllowModes(
      resourcePolicy,
      accessModes,
    );
    // 6. Add the new Rule to the Access Control Resource.
    resourceWithAcr = acp_v3.setResourceRule(
      resourceWithAcr,
      resourceRule,
    );
    // 7. Add the new Policy to the Access Control Resource.
    resourceWithAcr = acp_v3.setResourcePolicy(
      resourceWithAcr,
      resourcePolicy,
    );
    // 8. Save the Resource with its ACR.
    const updatedResourceWithAcr = await acp_v3.saveAcrFor(
      resourceWithAcr,
      { fetch: session.fetch }          // fetch from the authenticated session
    );
  } catch (error) {
    console.log("createResourceSpecificRulesPolicies");
    console.log(error);
  }
}

async function createResourceSpecificPublicRulesPolicies(resourceUrl, title, accessModes, session) {
  try {
```

```
    // 1. Fetch the SolidDataset with its Access Control Resource (ACR).
    let resourceWithAcr = await acp_v3.getFileWithAcr(
        resourceUrl,
        { fetch: session.fetch }          // fetch from the authenticated session
    );
    // 2. Create the Resource-specific Rule.
    let resourceRule = acp_v3.createResourceRuleFor(resourceWithAcr, title+"-public-rule");
    resourceRule = acp_v3.setPublic(resourceRule);
    // 3. Create the Resource-specific Policy.
    let resourcePolicy = acp_v3.createResourcePolicyFor(
      resourceWithAcr,
      title+"-public-policy",
    );
    resourcePolicy = acp_v3.addAllOfRuleUrl(
      resourcePolicy,
      resourceRule
    );
    // 5. Specify the access modes for the policy.
    resourcePolicy = acp_v3.setAllowModes(
      resourcePolicy,
      accessModes,
    );
    // 6. Add the new Rule to the Access Control Resource.
    resourceWithAcr = acp_v3.setResourceRule(
      resourceWithAcr,
      resourceRule,
    );
    // 7. Add the new Policy to the Access Control Resource.
    resourceWithAcr = acp_v3.setResourcePolicy(
      resourceWithAcr,
      resourcePolicy,
    );
    // 8. Save the Resource with its ACR.
    const updatedResourceWithAcr = await acp_v3.saveAcrFor(
      resourceWithAcr,
      { fetch: session.fetch }          // fetch from the authenticated session
    );
  } catch (error) {
    console.log("createResourceSpecificRulesPolicies");
    console.log(error);
  }
}

################################################################################
--------------------------------RESOURCE-BASED FUNCTIONS------------------------------
################################################################################
//------------------------------  FILE DATA -------------------------------------
function isFile(resourceURL){
```

```
  const pathArray = resourceURL.split("/");
  const lastIndex = pathArray.length - 1;
  const lastArray = pathArray[lastIndex].split(".");
  if (lastArray.constructor.name=="Array" && lastArray.length-1 == 1){
    return true;
  }
  return false;
}

async function uploadFileFromPath(filepath, mimetype, targetURL, fileName, session) {
  try {
    var data = await readFile(filepath);
    /**
     * to write a file without container (pre-existence) constraints
     * Users with Write access to the given folder/Container may prefer to use overwriteFile.
     * overwriteFile<FileExt>(fileUrl, file, options?): Promise<FileExt & WithResourceInfo>;
     */
    // If the targetFileURL exists, overwrite the file.
    const savedFile = await overwriteFile(
      targetURL + fileName , //"https://pod.example.com/some/container/myFile.txt",
      data, //new Blob(["This is a plain piece of text"], { type: "plain/text" }),
      { contentType: mimetype, fetch: session.fetch } //optional?, mimetype:"text/plain"
    );
    console.log(`File saved at ${getSourceUrl(savedFile)}`);
  } catch (err) {
    console.log("uploadFileFromPath");
    console.log(err);
  }
}

async function uploadJSON(data, url, fileName, session) {
  try {
    await writeFile('./utils/temp.json', data, 'utf8');
    await uploadFileFromPath('./utils/temp.json', "application/json", url, fileName, session);
  } catch (err) {
    console.log("uploadJSON");
    console.log(err);
  }
}

async function readFileFromPod(resourceURL, session) {
  try {
    if (isFile(resourceURL)){
      const file = await getFile(
        resourceURL,               // File in Pod to Read
        { fetch: session.fetch }       // fetch from authenticated session
      );
      console.log(`Fetched a ${getContentType(file)} file from ${getSourceUrl(file)}.`);
```

```
      const json = JSON.parse(await file.text());
      return json;
    }
    else {
      console.log("The URL passed as argument seems to not be a file.")
      console.log("Try to pass a file as URL.")
    }
  } catch (err) {
    console.log(err);
  }
}

async function readPublicFileFromPod(resourceURL) {
  try {
    if (isFile(resourceURL)){
      const file = await getFile(
        resourceURL               // File in Pod to Read
      );
      console.log(`Fetched a ${getContentType(file)} file from ${getSourceUrl(file)}.`);
      const json = JSON.parse(await file.text());
      return json;
    }
    else {
      console.log("The URL passed as argument seems to not be a file.")
      console.log("Try to pass a file as URL.")
    }
  } catch (err) {
    console.log(err);
  }
}

async function deleteFileFromPod(resourceURL, session) {
  try {
    if (isFile(resourceURL)){
       await deleteFile(
        resourceURL,          // File to delete from Pod
        { fetch: session.fetch }   // fetch from the authenticated session
      );
      console.log(`File deleted at/from ${resourceURL}`);
    }
    else {
      console.log("The URL passed as argument seems to not be a file.")
      console.log("Try to pass a file as URL.")
    }
  } catch (error) {
    console.error(error);
  }
}
```

```
//------------------------------  STRUCTURED DATA -------------------------------------
function isDataset(resourceURL){
  const pathArray = resourceURL.split("/");
  const lastIndex = pathArray.length - 1;
  const lastArray = pathArray[lastIndex].split(".");
  if (lastArray.constructor.name=="Array" && lastArray.length-1 == 0 && lastArray[0] != ""){
    //console.log("is a Dataset");
    //console.log(pathArray[lastIndex]);
    return true;
  }
  return false;
}

async function getDataset(resourceURL, session){
  try {
    if(isDataset(resourceURL)){
      let myDataset = await getSolidDataset(
        resourceURL,
        { fetch: session.fetch }          // fetch from the authenticated session
      );
      return myDataset;
    }
    else
        console.log("The resource is not a Dataset");
  } catch (error) {
    console.log("getDataset");
    console.log(error);
  }
}

async function createContainer(resourceURL, session){
  try {
    await createContainerAt(resourceURL, { fetch : session.fetch});
    console.log("Container created succesfully");
  } catch (error) {
    //Throws an error if creating the Container failed,
    //e.g. because the current user does not have permissions to,
    //or because the Container already exists.
    console.log("createContainer");
    console.log(error);
  }
}

//If operating on a container, the container must be empty otherwise a 409 CONFLICT will be raised.
async function deleteContainerFromPod(resourceURL, session) {
  try {
    if (isContainer(resourceURL)){
```

```
      await deleteContainer(resourceURL, { fetch : session.fetch });
      console.log(`Container deleted at/from ${resourceURL}`);
    }
    else {
      console.log("The URL passed as argument seems to not be a Container.")
      console.log("Try to pass a Solid Container as URL.")
    }
  } catch (error) {
    console.log("deleteContainerFromPod");
    console.log(error);
  }
}
```