Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Eng.ª Informática – Computação Móvel

EXPOSURE TRACKER LEDGER

TOMÁS HONÓRIO OLIVEIRA

Leiria, November of 2021

Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Informática
Mestrado em Eng.ª Informática – Computação Móvel

EXPOSURE TRACKER LEDGER

TOMÁS HONÓRIO OLIVEIRA
Número: 2190338

Project conducted under the guidance of the Professor Catarina I. Reis (catarina.reis@ipleiria.pt) and Professor Marisa Maximiano (marisa.maximiano@ipleiria.pt)

Leiria, November of 2021

ORIGINALITY AND COPYRIGHT

ACKNOWLEDGEMENTS

Whit the end of this project, I can only express my gratitude to the guiding teachers Catarina Isabel Ferreira Viveiros Tavares and Marisa da Silva Maximiano. Thank you for all the help and support given during the project implementation and later on the paper and final report.

ABSTRACT

On March 11, 2020, the novel coronavirus (COVID-19) was declared a global pandemic. With no treatment or vaccine available at the time, it was necessary to rely on non-pharmaceutical methods for case identification and contact tracing. This kind of approach has good results in detecting and preventing tuberculosis, sexually transmitted infections, and vaccine-preventable diseases. Contact tracing and keeping safe distances are crucial to containing the spread of COVID-19. Nonetheless, contact tracing is a complex intervention, it involves quarantining and investigating close contacts. Manual contact tracing methods are slow, require a large amount of effort, and, more often than not, rely on the memory or assumptions of individuals. To combat these downsides, contact tracing applications were developed, resulting in quicker and more reliable recognition of infected individuals. However, because of the complex nature of these applications and their lack of transparency, a large portion of the population started doubting the privacy of the data collected. On the other hand, one of the beacons of privacy we saw appear over the years was Distributed Ledgers. More accurately, the appearance of blockchain technology. By merging both contact tracing with a Distributed Ledger we hope to solve many of the issue faced by other applications. That being said, conventional Blockchain technologies, such as *Bitcoin* or *Ethereum*, cannot support a large amount of transactions in any meaningful way. At least for a use case like contact tracing, that may require the interaction of millions of users. However, Blockchain is not limited to only *Bitcoin* or *Ethereum*, and one of the most prominent projects launched in the past five years is actually *Hyperledger*. *Hyperledger* is an umbrella project focused on developing tools, frameworks, and libraries for enterprise-level blockchain implementations. *Hyperledger* is really interesting, but if we want to support a large amount of transaction, we have to choose which one of his underlying project is the most resilient for our use case. That's how we got to *Hyperledger Sawtooth*, a project that seeks to remain distributed while offering a secure environment for Smart Contracts. Perfect for building an application focused on data privacy and transparency. With that said, the development applied throughout the project was the combination of a lot of research both regarding contact tracing, as well as regarding *Hyperledger Sawtooth*. Despite many challenges, the development of a contact tracing application focused on data privacy was achieved.

CONTENTS

Originality and Copyright i
Acknowledgements iii
Abstract v
Contents vii
List of Figures xi
List of Tables xiii
List of abbreviations xvii

1 INTRODUCTION 1

1.1 Objectives . . . 3
1.2 Methodology . . . 4
1.3 Document structure . . . 4

2 BACKGROUND 5
2.1 *Distributed Ledger* . . . 5
2.1.1 Hash Functions . . . 6
2.1.2 Merkle Tree's . . . 7
2.2 Blockchain . . . 8
2.2.1 Consensus . . . 8
2.2.2 Public Blockchains . . . 9
2.2.3 Permissioned Blockchains . . . 10
2.3 Blockchain 1.0 - *Bitcoin* . . . 10
2.3.1 Nodes . . . 11
2.3.2 Transactions . . . 12
2.4 Blockchain 2.0 - *Ethereum* . . . 12
2.4.1 *Smart Contracts* . . . 13
2.4.2 *Ether* . . . 14
2.4.3 *Tokens* . . . 15
2.4.4 On-chain vs off-chain . . . 16
2.5 Blockchain 3.0 - Hyperledger . . . 16
2.5.1 Hyperledger Sawtooth . . . 17

2.5.2 Hyperledger Fabric 18

3 RELATED WORK 19
3.1 Contact Tracing Applications - Centralized Systems 20
3.1.1 Portuguese application 20
3.1.2 Irish application 23
3.1.3 French application 25
3.1.4 New Zealand Application 27
3.1.5 Singapore Application 29
3.2 Contact Tracing applications - Decentralized Systems 32
3.2.1 BeepTrace 32
3.3 Comparison 34
3.3.1 Centralization vs Decentralization 34
3.3.2 Voluntary vs Involuntary 35
3.3.3 Data usage only within the app 35
3.3.4 Data destruction 35
3.3.5 Possible Concerns 36

4 EXPOSURE TRACKER LEDGER - REQUIREMENTS ELICITATION 37
4.1 Privacy Protection 37
4.2 Detection of close contacts 38
4.3 Submission of interactions 39
4.4 Notifying about possible infections 40

5 EXPOSURE TRACKER LEDGER - DESIGN AND IMPLEMENTATION 41
5.1 Detection of close contacts 42
5.2 Interaction registration 43
5.3 Ledger technology and Storage 44
5.4 Storage Model 45
5.4.1 Single address 45
5.4.2 One user per address 46
5.4.3 One user multiple addresses 46
5.5 The implementation of a node 46
5.5.1 Docker 47
5.5.2 Transaction Family 47
5.5.3 Consensus algorithm 48
5.6 Submission into the ledger 49
5.6.1 Fetch and parse the *Payload* 50

5.6.2 Build a Transaction 52
5.6.3 Submit to the Rest API 54
5.7 Listening for Changes 56
5.8 Challenges 60

6 CONCLUSION AND FUTURE WORK 63

BIBLIOGRAPHY 67

Appendix

A APPENDIX A 77

DECLARATION 79

LIST OF FIGURES

Figure. 1 Timeline of task execution . . . 4
Figure. 2 Centralized Systems vs Distributed Systems . . . 5
Figure. 3 Merkle Three illustration . . . 7
Figure. 4 *Bitcoin* [18] . . . 10
Figure. 5 Ethereum [25] . . . 13
Figure. 6 CryptoKitties [34] . . . 16
Figure. 7 Fabric [37] . . . 18
Figure. 8 StayAway Covid Logo [46] . . . 21
Figure. 9 DP-3T Logo [47] . . . 21
Figure. 10 Application adoption, according to INESC TEC ([50]) . . . 22
Figure. 11 Covid Tracker Logo [55] . . . 23
Figure. 12 TousAntiCovid logo [57] . . . 25
Figure. 13 NZ Covid Tracer Logo [60] . . . 27
Figure. 14 Trace Together Logo [63] . . . 29
Figure. 15 Beep Trace Logo [69] . . . 33
Figure. 16 Users within 2m range are recorded . . . 38
Figure. 17 App Summary . . . 38
Figure. 18 Submit contact views . . . 39
Figure. 19 Contacted with an infected user (Notification) . . . 40
Figure. 20 Sawtooth Architecture Diagram [36] . . . 41
Figure. 21 Beacon comparison [79] . . . 42
Figure. 22 Beacon running in the background (Notification) . . . 42
Figure. 23 One address for all data . . . 45
Figure. 24 One address per User . . . 46
Figure. 25 Multiple addresses per user . . . 46
Figure. 26 Simulation of a basic node . . . 47
Figure. 27 Simulation of 5 nodes running at same time . . . 49
Figure. 28 The generated IDs are submitted into the ledger . . . 50
Figure. 29 Validator Broadcast . . . 56
Figure. 30 Event subscription . . . 57
Figure. 31 Event protobuf [91] composition . . . 57
Figure. 32 Fetching only the required amount of information . . . 58

Figure. 33 SDK Maturity [36] 60
Figure. 34 State example [36] 61

LIST OF TABLES

Table. 1 Contact Tracing application compared 78

LISTINGS

5.1 Submit data into state 47
5.2 Fetch IDs and serialize the data 51
5.3 Steps for submitting transaction 53
5.4 Submit transaction into Rest API 55
5.5 Listen for changes 59

LIST OF ABBREVIATIONS

DAG	Directed Acyclic Graph.
DP-3T	Decentralized Privacy-Preserving Proximity Tracing.
ETH	Ethereum (cryptocurrency).
EVM	Ethereum Virtual Machine.
FA	Federated architecture.
LOB	Lines of Business.
MOH	Ministry of Health.
MSP	Membership Service Provider.
pBFT	Practical Byzantine Fault Tolerance.
PoET	Proof of Elapsed Time.
QR-code	Quick Response Code.
SDK	Software development kit.
SMS	Short Message Service.

1

INTRODUCTION

Over the recent COVID-19 epoch, it has become clear that the once acclaimed tracking applications have mostly turned out to be a letdown [1]. That being said, the question is why did it end up like that, and what led this type of application to be abandoned.

The COVID-19 pandemic caused a global health crisis that no one was prepared to face, resulting in millions being infected and the appearance of new, more infectious strains [2]. It is now clear that we need to track the infections, and since this is such a significant process, it needs to be done reliably and in real-time [3]. However, decision-making has proven to be a daunting challenge for both authorities and the public.

To better understand the dynamics of this pandemic and provide effective countermeasures, data with quality is a must. That being said, the search for trustworthy data has led to the appearance of contact-tracing applications, apps that in theory should facilitate the management and tracking of new cases. However, it is apparent that in the vast majority of cases, the result was somewhat of a letdown.

According to [4], the main reason for the low adherence was not technological limitations, but the concern with privacy and the negativity surrounding this type of application. One of the leading factors of this sentiment was the direct connection between these applications and big tech giants such as Google and Apple. A vast portion of contact-tracing applications were based on protocols provided by Google/Apple, giving them power to change them overnight. These being the same companies that come to blows repeatedly with entities like the *EU Court of Justice* because of privacy concerns. If possible, a user should be able to identify where and how their data is being used. We propose to solve the lack of trust by relying on transparent systems, such as solutions based on *Distributed Ledger*, more precisely Blockchain.

The present project intents to explore how a Contact Tracing application based on a distributed ledger (Hyperledger) would work. And what needs to be considered achieving such a goal. With that, the project will comprise the study and development of a Contact Tracing application based on Hyperledger.

This theme is of special interest because of all the growth in the area (Blockchain) over the past few years. A growth that seems to be even more exacerbated because of the current climate.

1.1 OBJECTIVES

The project is thus oriented based on these three essential objectives:

O1 Survey the state of the art regarding Distributed Ledgers;

O2 Survey the state of the art regarding Contact Tracing and why so many of these applications were a letdown;

O3 Analyses and design a platform for contact tracing that leverages Distributed Ledgers / Blockchain to answer potential concerns.

In order to tackle the first objective, some questions need to be answered:

- What is a Distributed Ledger and why do we need it?
- What is a Blockchain and how does it relate with the previous question?
- What is consensus?
- What are permission levels?
- What is Bitcoin / Ethereum?
- What are tokens?
- What are Smart Contracts?

Regarding the second objective, a thorough analysis needs to be done to find out what were the issues that lead to the demise of so many of these applications. We will need to compare some of the most popular contact tracing applications and answer the following questions:

- How does the application work?
- Does it have any outstanding feature?
- How was the adoption rate?

To succeed in the third goal, a proof of concept was implemented. Only by leveraging Distributed Ledger technology is possible to solve the issue discussed on the second topic.

Tasks	Time taken	2020									2021					
		Sept	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	June	July	Aug	Sept	Oct	Nov
Survey on the state of the art (Distributed Ledgers)	11 months															
Survey on the state of the art (Contact Tracing)	11 months															
Survey on app adoption / issues	7 months															
Technologies choice	1 month															
Prof of Concept development	5 months															
Documentation / report	12 months															

Figure 1: Timeline of task execution

1.2 METHODOLOGY

Due to the sheer complexity of the project, it was divided into several stages. That being said, in the Figure 1, it is possible to see how much time was assigned to each task.

1.3 DOCUMENT STRUCTURE

This report has 6 chapters, each one with the corresponding subsections.

Starting with the second one, the “Background” chapters looks at the underlying concepts required for the project. The primary focus is Distributed Ledgers and its many forms, as well as an introduction to contact tracing.

In the third chapter, also known as “Related work”, is done a brief analysis on the existing solutions and their degree of success.

This is followed by the chapter “Exposure tracker ledger - requirements elicitation”, where the requirements are laid out, and the first overview of the application developed in partaken.

The fifth chapter, “Exposure tracker ledger - design and implementation“, is dedicated to the exploration of how the application was developed, and what were the technologies chosen.

At last, the conclusion summarizes what was done, what could have been done and what were the challenges.

2

BACKGROUND

To better illustrate how this kind of application would work, it is critical to comprehend some key concepts. In the following sections, we will approach the basis behind a *Distributed Ledger* and how it relates to known blockchain technologies, like Bitcoin [5] and Ethereum [6]. Ultimately, we will also take a quick look at a new and emerging technology that goes by the name Hyperledger [7].

2.1 *DISTRIBUTED LEDGER*

The concept of Distributed Ledgers has been gaining attention over the past few years, in part because of the emergence/spread of new blockchain technologies based on *Bitcoin* and *Ethereum*. However, to understand the underlying concept of a *Distributed Ledger*, it is necessary to comprehend what are the differences between a centralized architecture when compared with a decentralized one. It can be summarized through the question "Who has control over the network."

In a centralized system, there is a sole authority with absolute control over all aspects of the network. This authority is exercised through a central server that manages data and permissions. On the other hand, a decentralized system works quite differently. As a rule, it makes use of nodes that have independent authority

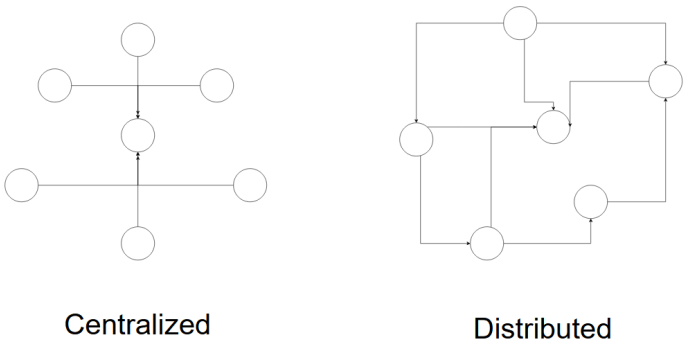

Figure 2: Centralized Systems vs Distributed Systems

and decision-making power, by looking at Figure 2 it is possible to better grasp how the nodes interact. These networks allow for both computational power and work to be distributed equitably across the network.

In short, a *Distributed Ledger* is a shared database, replicated and synchronized between the members of a decentralized network. All participants are governed by a form of consensus that allows them to unanimously update their records. It should be noted that a *Distributed Ledger* stands in direct contrast with conventional (centralized) systems. One of the most highlighted differences remains the fact that conventional systems have a single point of failure.

By default, a database allows four types of operations, CREATE, READ, UPDATE and DELETE (CRUD). And since a *Distributed Ledger* can be seen as a database, it would make sense to support the same CRUD operations. But this is not the case, as one of the fundamental differences between a *Distributed Ledger* and a traditional database is the constraints of performing a DELETE operation. As mentioned in Branden[8], immutability, reduced costs, and security are directly related to how a *Distributed Ledger* makes use of cryptographic algorithms.

There are several ways to implement a *Distributed Ledger*, different data models, and technologies. However, all *Distributed Ledgers* tend to be based on three fundamental characteristics:

- Public key encryption;
- Distributed *peer-to-peer* networks;
- Consensus mechanisms.

2.1.1 *Hash Functions*

The primary use given to hash functions [9] is usually to protect data integrity. They make use of a mathematical algorithm in order to map a “message” (data of arbitrary size) into a *Bit Array* of fixed size. It works because the final output (The Hash) is one-sided, meaning that is computationally infeasible to reverse engineer the original message from a given hash.

Usually a person makes use of Hash in order to validate that a message has the current content. Let’s say, a user wants to download a piece of software, but is not sure about the authenticity. However, this same user has the Hash for this same software, acquired through a trusted source. The user can then generate and

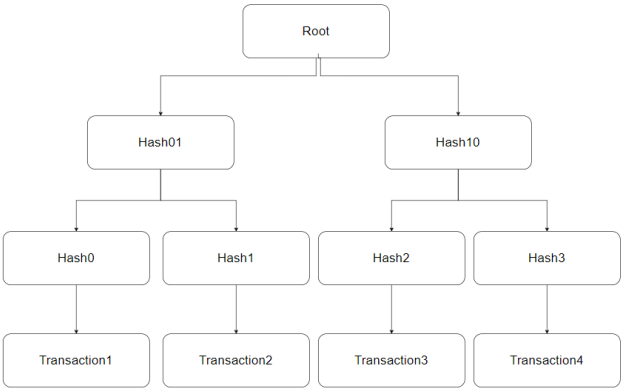

Figure 3: Merkle Three illustration

compare the Hash of the software he was not sure, with the trusted one. Validating with this, the authenticity.

According to [10], a hash function must have the following properties:

- Be deterministic, and the same massage must always result in the same hash.
- Be quick to calculate regardless of the message.
- Being computationally unfeasible, generating a hash that returns a preconceived value.
- Being computationally unfeasible, getting the same hash for two different messages.
- Any change that occurs in a given message must have such an extreme impact that the resulting hash does not bear any similarity to the original hash.

2.1.2 *Merkle Tree's*

Merkle Tree [11] is a data structure often used in conjunction with *Distributed Ledgers*. In bitcoin and other cryptocurrencies, Merkle trees are widely used to encode data in the most efficient and secure way. Often portrayed as trees, they are composed of a bottom row, referred as "leaves", the row in the middle, referred as "branches" and the hash at the top, referred as "root". A representation of the structure just described is seen in the Figure 3

Their primary use is to validate the integrity of data stored, that was manipulated or transferred between computers. Given the way *Merkle Tree's* work, a user can verify a specific transaction without having to search the whole three, it becomes

relatively mundane to validate data integrity and really efficient. It also achieves space benefits by making use of Hash's instead of the original file.

2.2 BLOCKCHAIN

The meteoric rise of Bitcoin and Ethereum made us aware of terms like Blockchain [5], *Distributed Ledger*, Cryptocurrencies, and many, many more. It seems fitting to address what are the key differences between this and a Blockchain since more often than not, both terms are misused.

A blockchain is in his essence a digital ledger of transactions that is duplicated and distributed across a network. It can also be described as a growing sequence of blocks connected as if they were a chain. Each block contains several transactions, and every time a new transaction occurs, a record of the transactions is added to every participant's ledger. With this, consensus is kept across the network, since if one block is changed, it is immediately apparent that someone is tampering with the data.

2.2.1 *Consensus*

The duplication of digital goods is fairly easy, being not far-fetched the idea that someone could issue transactions in parallel and thus transfer the same good to different recipients. This risk is inherent to pretty much every digital good / currency and is commonly known as the "Double spending" issue. The same is much less common when it comes to physical coins we use in our day-to-day life. Since everyone involved in a transaction has visual (immediate) access to the original currency. Distribution of information and lack of consensus is a challenge, especially in the presence of malicious participants.

Being a decentralized system, blockchain systems do not need a trusted third-party. Instead, in order to ensure the reliability and consistency of data/transactions, blockchain adopts a decentralized consensus mechanism, that the various members of the network have to agree on. In other words, a consensus mechanism is the set of rules that keep facts consistent among all network participants.

This topic was first approached by Pease, Shostak and Lamport in 1980 [12], later being popularized with the story of the Byzantine Generals. It can be summarized with the following description:

Let's imagine a Byzantine army that is preparing to attack a city. To start the attack, the generals of each division (which are scattered around the city's outskirts) must agree on a battle plan. However, while some generals wish to attack, others prefer to retreat. To reach consensus, all generals must agree on the same decision.

Nowadays, there are several solutions for this problem. As an example, the two most well know blockchain implementations (*Bitcoin* and *Ethereum*) both make use of *Proof of Work*.

Proof of Work [13] is known for requiring a high computational power during its execution, and the idea behind it is that a given node needs to prove that they made a large amount of effort in order to submit a new block. *Nodes* provide the computational power needed to maintain the blockchain and verify transactions, while ensuring the network's immunity against malicious attacks.

If the block is extracted correctly, it will be attached to the ledger and the first *Node* that was able to solve the "puzzle" given by the *Proof of Work* algorithm will be rewarded. The puzzle is usually a computationally hard but easily verifiable problem [14]. And once solved, it will be broadcasted to other nodes, in order to be verified. Solving the "puzzle" is often equated with mining a precious material/currency, thus, nodes that do this kind of work are often called miners.

In his essence, *Proof of Work* involves looking for a value that, when hashed using the SHA-256 cryptographic hash function, starts with a certain number of bits set to zero. The average work required is exponential and can be verified by running a single hash. Because of its nature, after this effort is spent, the block cannot be changed without redoing all the work, and to change a block, it would be necessary to redo all the blocks that follow it.

2.2.2 *Public Blockchains*

Bitcoin and *Ethereum* are the best-known examples of public blockchains [15]. They were developed with the assumption that everyone is free to enter the network and validate transactions [16].

Transactions are recorded to the *Ledger* only if consensus is achieved among the majority of participants. All *Nodes* own a copy of the data and that makes it almost impossible to tamper.

The public network operates under an incentive scheme, which translates into the adhesion of new participants and encourages them to keep the network agile. This

Figure 4: *Bitcoin* [18]

solution is extremely valuable when the aim is to produce a truly decentralized, democratic, and authority-less system.

2.2.3 *Permissioned Blockchains*

Due to the unique characteristics of immutability and decentralization, blockchain technology has evolved beyond *Bitcoin* and *Ethereum*. Coming to integrate new priorities more focused on the business world, being them the need to track records and assets, compliance and monitoring of agreements, authentication and authorization of participants.

Permissioned Blockchain [17] offer high versatility, and its configuration can change as needed. One option is the possibility of allowing anyone to enter the network after a brief verification of your entity.

Unlike public blockchains, this form of blockchain enjoys the advantages of blockchain technology without sacrificing the authority aspect of a centralized system. Thereby maintaining a layer of control to allow certain actions to be performed only by certain identifiable participants.

2.3 BLOCKCHAIN 1.0 - *BITCOIN*

The first attempts to build a digital currency were strongly linked to the existence of a single central authority responsible for supervising/validating any transaction. However, with the appearance of alternative approaches such as B-money [19],

Karma [20], RPOW [21] and bit gold [22], this task was eventually delegated to cryptographic problems such as Proof of work.

The appearance of new distributed systems led to many opportunities. It's now possible for any individual to be part of the validation process needed for transactions. However, because of these solutions, property registries remained centralized through a single entity. Then, on the October 31, 2008, Satoshi Nakamoto shared with the rest of the world the paper "*Bitcoin: The Peer-to-Peer Electronic Cash System*" [5].

Bitcoin (see Figure 4) carries attributes of a payment system as it facilitates the transfer of value between various parties. However, unlike traditional payment systems, which usually involve transferring value in a sovereign currency (such as the euro), *Bitcoin* has its own value metric called bitcoin.

It also tries to diminish/eradicate the reliance on third-parties such as banks and financial institutions. One way it manages this is by each entity negotiating directly. The owner of a coin has full control over its asset, being able to spend it anytime/anywhere. It is also important to mention that in order to achieve full independence over third-parties, each "account" must be distributed.

Bitcoin is a *token* with no reference to any underlying commodity or sovereign currency. It has no intrinsic value, other than its use to make payments and the speculation it generates.

Bitcoin was the catalyst behind the concept of Blockchain and thanks to this, cryptocurrencies such as bitcoin and related applications are often labeled Blockchain 1.0.

2.3.1 *Nodes*

Generally speaking, Node is the name given to a device connected to the network. They can take different forms, each one playing a different role. With *Bitcoin*, there are four main types of *nodes*:

- *Mining Nodes*;
- *Complete Nodes*;
- *Super Nodes*;
- *Light Nodes*;

Full Nodes, *Super Nodes* and *Light Nodes* all perform similar functions, while *Mining Nodes* deviate a little from the remains nodes [23].

Mining Nodes confirm that a block should be placed in the ledger, the process is known as "mining". After a block is created, it is sent over the network to *Full Nodes*, who validate them and add them to the blockchain. As the name suggests, *Full Nodes* maintain and distribute copies of the entire blockchain record. With the help of them, it is possible to validate transactions up to the *Genesis Block*.

Super Nodes connect with other *Full Nodes* in order to spread the blockchain across the entire network. They ensure everyone has the correct copy of the blockchain.

Light Nodes perform a similar function to *Full Nodes*, but instead of keeping an entire copy of the blockchain, they only contain a small portion of it.

2.3.2 *Transactions*

Transactions are data structures that encode the transfer of value between the participants of the system [24]. They can be created by anyone, even if they are not an (authorized) signatory to the account.

Once created and duly formed, transactions must be signed by the owner(s) of the funds, after which they become valid. In this state, they should contain all information needed to carry out the transferal of funds. This includes one or more signatures displaying the authorization to spend the funds.

Afterwards, the transaction is propagated throughout the network *Bitcoin* until it reaches almost every node. Being each one responsible for validation. Once the transaction is verified by a mining node, it is included into a block of transactions, and then recorded into the blockchain. After this, the remaining nodes will need to confirm the new addition in order to reach a consensus. If this is the case, the new block will then become a permanent part of the blockchain [24]. Allowing for the funds transfer during the transaction to be available for use once again.

2.4 BLOCKCHAIN 2.0 - *ETHEREUM*

Launched in 2015, *Ethereum* (see Figure 5) is an *open-source* platform for decentralized applications that goes beyond the features proposed by *Bitcoin* [26]. It came about through a programmer by the name of Vitalik Buterin.

Figure 5: Ethereum [25]

It is important to understand that, although constantly being compared to each other, *Ethereum* and *Bitcoin* are two different projects with target objectives. *Bitcoin* is the first cryptocurrency and, in essence, a money transfer system. *Ethereum* on the other hand, took the technology behind *Bitcoin* and substantially expanded its capabilities. Currently, *Ethereum* is a complete network, with its own browser, programming language and payment system. Most importantly, it allows users to create decentralized applications on the Blockchain *Ethereum*. The *Ethereum* network can be described as a cluster of *Nodes* connected to each other. The entire network can be seen as a single entity called *Ethereum Virtual Machine* or EVM.

The distinction most often pointed out to *Ethereum* when paired with *Bitcoin*, is the presence of *Smart Contracts*. However, these are not exclusive to *Ethereum*. On the contrary, with *Bitcoin* it was already possible to develop a *Smart Contract*. However, *Smart Contracts* as we know them today, first appeared with *Ethereum*.

With *Ethereum*, the second wave of innovation in *Blockchain* began, expanding the use cases offered by *Bitcoin*.

2.4.1 *Smart Contracts*

In the 1990s, cryptographer Nick Szabo coined the term *Smart Contracts*, defining it as “a set of digital promises, comprising protocols respected by everyone involved”

[27]. Since then, the concept of *Smart Contracts* has evolved, especially after the introduction of decentralized blockchain platforms such as *Bitcoin* (2009).

The term *Smart Contracts* is a testament to their ability to run automatically once a pre-defined requirement is met. It should be noted that *Smart Contracts* are a *Ethereum* account type, have a balance, and can send transactions over the network. However, they are not controlled by a user, instead they are deployed to the network and run as scheduled, being that any user can interact with a *Smart Contract*.

Since rules can be defined and applied automatically through the code, each contract is identified by an *Ethereum* address in order to be used in a transaction. And since every transaction is registered, it is possible to keep everyone involved accountable for their actions.

Smart Contracts offer enormous flexibility to its developers, which can be used in different ways. It is even possible to use a *Smart Contract* to complement another *Smart Contract*. In short, *Smart Contracts* are applications that run on the *Ethereum Virtual Machine*.

In turn, *Smart Contracts* are developed using high-level languages, such as "Solidity" [28]. However, to be executed they must be compiled to byte code, in order to be executed by the EVM. Once compiled, *Smart Contracts* are inserted into the *Ethereum* platform through a special transaction. The contract creator does not get any special privileges at the protocol level (although he can explicitly encode them in the *Smart Contract*).

2.4.2 *Ether*

As was previously discussed, an *Ethereum* network can make use of *Smart Contracts*, and with this, run programs directly inside the network. Unfortunately, this creates an issue, since it becomes possible to create optimized / malicious programs that will slow the *Ethereum* EVM. In order to combat such issue, transaction fees were introduced. *Gas* refers to the fee required to conduct said transactions successfully, they are paid in Ethereum's native currency, Ether (ETH). Like *Bitcoin*, *Ether* is also a digital asset/cryptocurrency, however, it allows for much more possibilities when compare with the original *Bitcoin*.

When *Miners* successfully verify a group of transactions, they receive a small amount of *Ether* as compensation.

2.4.3 *Tokens*

The previously discussed *Ether* and *Gas*, are only two of the many tokens *Ethereum* currently supports. In *Ethereum*, a *token* can represent anything from a physical object such as gold (Digix) to a native currency used to pay transaction fees (Golem) [29]. The properties and functions of each *token* are entirely subject to their intended use. They can have a fixed supply, constant inflation rate or even a supply determined through a sophisticated monetary policy [29].

That being said, the *tokens* must follow a set of rules/standards, in order to guarantee that when a new project issues a *token*, it remains compatible with the existing decentralized exchanges. Among the most frequently used patterns are the *tokens* ERC20 and ERC721.

2.4.3.1 *ERC20*

ERC20, are a pattern of fungible *tokens*, where each *token* is exactly the same with regard to its value. These are the most common type of *tokens* in the *Ethereum* network, and usually are used to pay for goods and services [30]. Some features that should be expected from these are [31]:

- Fungible - Each *token* is exactly the same in terms of its value;
- Transferable - Can be sent from one address to another.
- Fixed Provisioning - A fixed number of *tokens* must be created so that developers cannot issue more *tokens* and increase the provisioning.

2.4.3.2 *ERC721*

In contrast, ERC721 is the standard used for the acclaimed NFTs (Non-Fungible). They are used to identify something / someone in a unique way. It is unique and may have a different value from another *token* of the same *Smart Contract* [32]. This type of *token* is often used to represent collectibles, access keys, tickets, among others. There is a wide variety of ERC721 *tokens* available on *Ethereum*, the most popular being dubbed *CryptoKitties* [33].

One of the best known applications developed using *Smart Contracts* on the *Ethereum* network is *CryptoKitties* (see Figure 6). Calling itself “one of the first games in the world to be developed with blockchain technology”. Essentially, is a collection item stored in the *Ethereum* network. New *CryptoKitties* are generated

Figure 6: CryptoKitties [34]

via “playback”, which involves choosing two basic *CryptoKitties* and spending *tokens Ether* to execute a *Smart Contract*. The contracts use the two chosen cats to generate a new CryptoKitty. These cats and the details of the breeding process are stored on the *Ethereum* network. A user who owns these *CryptoKitties* can then sell them or trade them with other people. The art itself for each CryptoKitty is stored of-chain.

2.4.4 *On-chain vs off-chain*

An On-chain transaction occurs when the blockchain is modified to reflect a transaction that has taken place in your ledger. This implies that the transaction is validated and authenticated by enough participants, recording the transaction details in the appropriate block and transmitting the necessary information to the entire network.

Essentially, each step of a transaction occurs in the blockchain, and the blockchain state is modified to reflect the transaction occurrence and validity.

Off-chain transactions refer to transactions that occur on a blockchain but whose value/impact is felt in a database outside the blockchain.

2.5 BLOCKCHAIN 3.0 - HYPERLEDGER

That being said, blockchain is just one of the many forms a *Distributed Ledger* may have, and while *Bitcoin* and *Ethereum* have been pretty popular, blockchain goes beyond that, example of this, is the Hyperledger [7].

Presented by the Linux Foundation back in December 2015, Hyperledger, since then has seen the contribution of companies like IBM, Intel, and SAP Ariba. With this, Hyperledger can be seen as an umbrella project focused on developing tools, frameworks, and libraries for enterprise-level blockchain implementations [35]. Rather than declaring a single blockchain standard, Hyperledger encourages a collaborative approach. Some of the most well known projects that resulted from it are Hyperledger Fabric, Hyperledger Sawtooth, Hyperledger Grid among others.

2.5.1 *Hyperledger Sawtooth*

Hyperledger Sawtooth [36] is a *Distributed Ledger* developed by Intel, that seeks to remain distributed while offering a secure environment for *Smart Contracts*. The project is impressive, being flexible and ease to use.

Both Bitcoin and Ethereum have several drawbacks, that in hindsight, make them undesirable for a large number of concurrent transactions. On the other hand, Hyperledger Sawtooth solves this and improves privacy by using a permission mode, together with a fine-grained access control.

Other of the advantages is the ability to specify business rules without requiring extensive knowledge of the underlying design. This being possible, thanks to Sawtooth's *core*, as it allows the choice of *transaction rules*, permissions and algorithms in an easily approachable manner. In addition, it does not require mining, since one of the key features of a permissioned model is that everyone involved has a stake in verifying the data.

At last, it has a strong separation between Application Level and Core system. It allows design decisions to be made at the processing layer, thus enabling multiple applications to exist in the same network instance (Transaction Families). This separation is done with the help of transaction families, which allow application developers to write in the languages of their choice. In this context, we can think of a transaction family as the way Hyperledger Sawtooth implements *Smart Contracts*.

The transaction family main components are:

- A transaction processor to define the business logic for your application;
- A data model to record and store data;
- A client to handle the client logic for your application.

Figure 7: Fabric [37]

2.5.2 *Hyperledger Fabric*

Hyperledger Fabric (see Figure 7) is a *Distributed Ledger* developed by IBM, that offers high levels of confidentiality, resiliency, flexibility and scalability [38]. It was designed to support modular implementations of different components and accommodate the complexity that exists across an entire economic ecosystem [39].

One of its distinct features is it being a private blockchain with permissions that offers the ability to create channels, allowing a group of participants to create a separate Transaction Ledger. Something fundamental, since in competitive environments where the various participants compete with each other, secrecy is a fundamental asset.

Hyperledger Fabric makes use of protocols such as "Proof Of work" in order to validate transactions and protect the network, with the members subscribing through a "Membership Service Provider" (MSP).

That being said, Hyperledger Fabric *ledger* is made up of two components:

- World state - describes the state of the *ledger* in a given period. In other words, it's the *ledger* database;
- Transaction log - logs all transactions that resulted in the current value of the World state. In other words, it is the update history of the World state.

Smart Contracts are written in Chaincode, meaning that they can be implemented in several programming languages, currently supported by Go, Node js and Java.

3

RELATED WORK

Contact Tracing can be described as using collected data from people diagnosed with an infectious disease, to recognize and provide support to new infectious individuals [40].

By enabling people to know that they may have been infected, it is possible to monitor their health for symptoms. The World Health Organization (WHO) defines three crucial steps for any form of contact tracing [41]:

- Identification - Upon confirmation of an infected individual, all of his contacts must be identified. This is achieved by analyzing the habits and activities of the infected individual.
- Listing - All people who have come into contact with an infected person must be listed as a "Contact" and informed of their status.
- Follow-up - Contacts should be observed regularly to monitor the onset of symptoms/complications.

According to [42], close contacts who spent over 15 minutes in contact with an infected person are of special interest, since they are more likely to being infected. By testing these individuals, it is possible to delay the progress of the transmission chain.

As a benchmark for a successful contact tracing (COVID-19) operation, WHO suggests locating/quarantining 80% of close contacts within 3 days (after the first case is confirmed). According to Christophe Fraser (Oxford University), transmission is extremely fast and the virus can spread before any action is taken [43]. Even if all cases were discovered/isolated within three days, the pandemic will continue to grow. To prevent such an outcome, 70% of cases need to be isolated on the first day. Only then, can the outbreak be significantly reduced.

Mukhi et al [44], advocates that traditional methods of surveillance and data collection (employing a paper-based method), pose many challenges, such as data loss, duplication, difficulty in managing databases, and lack of timely access to the

data. That being said, the only way to gather such information in such a brief amount of time is by relying on technology.

As mentioned by Mukhi et al [44], COVID-19 pandemic infection/death rates may slow down in countries with robust vaccination coverage. However, on a global scale, new mutations and new pathogens will continue disrupting society for ages to come, and therefore, it is important to keep advancing in this field.

Contact tracing applications allow for a quicker and more reliable identification of newly infected individuals. Thanks to this, the quality of the data collected is also greatly improved, being this the main purpose for the development of such apps in several countries.

3.1 CONTACT TRACING APPLICATIONS - CENTRALIZED SYSTEMS

Starting with an overview of well known centralized contact tracing applications. In this section, we will talk about the applications from Portugal, Ireland, France, New Zealand and Singapore.

3.1.1 *Portuguese application*

Starting with the Portuguese application (see Figure 8), the development was a collaboration between INESC TEC, ISPUP, Keyruptive, and Ubirider, where companies and research institutions joined efforts to develop the application Stay Away Covid [45]. This application has the purpose of detecting (through the use of smartphone) if users were near someone infected with COVID-19. It works by announcing its presence to all nearby devices, using random identifiers. By doing this, the application recognizes who has been in contact with the user. For a contact to be of interest, it needs to be within 2 meters and for at least 15 minutes [45]. In case of infection, the user will receive a code to be inserted into the application, after which they will notify the contacts. One of the key points is that it works without recording personal information, being the sole exception, infected users.

3.1.1.1 *Technical Analysis*

As advertised on the StayAway Covid GitHub [46], StayAway Covid is a React Native project that makes use of the "Decentralized Privacy-Preserving Proximity

Figure 8: StayAway Covid Logo [46]

Figure 9: DP-3T Logo [47]

Tracing"(DP3T) SDK. It makes use of the DP-3T protocol by using the Exposure Notification Framework developed by Apple / Google.

DP-3T (see Figure 9) is an open-source protocol developed in response to the COVID-19 pandemic, intended to facilitate the contact-tracing of infected individuals.

DP-3T makes use of Bluetooth Low Energy in order to track and record encounters. It also keeps data privacy, by ensuring personal data and computation remain entirely on the user's phone. With this, it can be mostly decentralized, with the only exception being if a user is flagged as infected. If that is the case, some information will be transmitted to a centralized server, allowing for contacts to be informed. This information will be kept for 14 days.

According to the paper [48], the following criteria must be always be accomplished:

- "Minimization" of the data. The central server can only observe watches (anonymous) identifiers of users who have tested positive. It should not be possible to get any extra information, other than that provided by the user when notifying him that he was infected;
- Prevents data abuse.
- Prevent user tracking. No entity should be able to track users who have not reported a positive diagnosis.

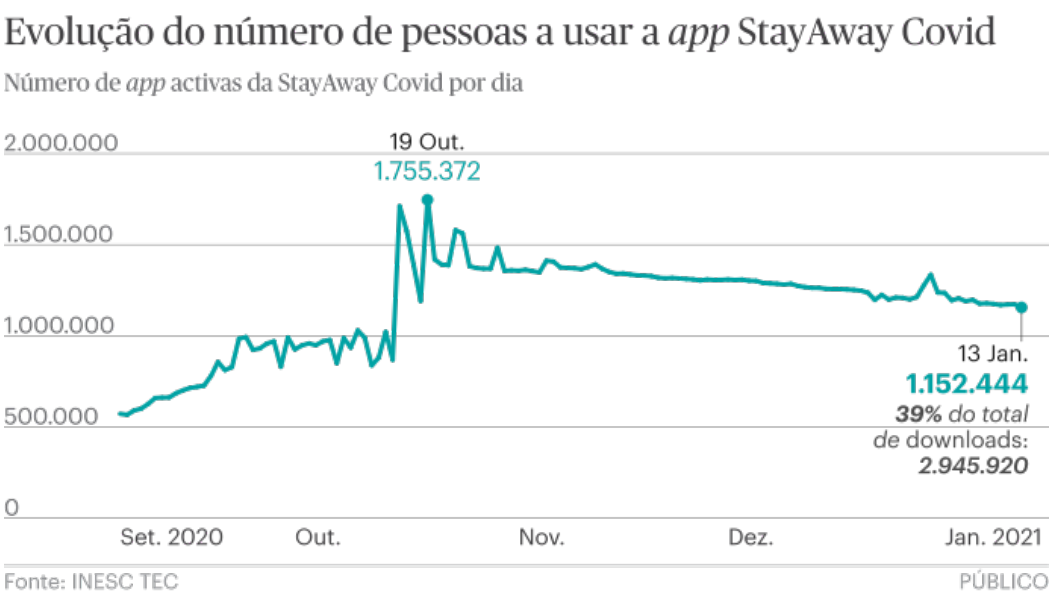

Figure 10: Application adoption, according to INESC TEC ([50])

- Self Dismantling. The system must be dismantled as soon as the application is no longer needed.

According to paper [49], the Bluetooth-based contact tracking system was first proposed by Altuwaiyan et al. (2018). In this way, it is possible to directly detect whether two users have been determined and, if so, to know with some precision at which distance thanks to the strength of the signal. However, it is not a perfect system, as it is impossible to detect indirect contagions, which occurred through surfaces or other infected environments.

Indirect transmission refers to the transfer of an infectious agent from a carrier to a new individual through:

- Air Particles;
- Contact with contaminated objects;
- Contact with animals.

3.1.1.2 *Adoption*

According to "60% have already deleted StayAway Covid" [51], the application has been losing users at an alarming rate, being the peak of users during October 2020. Around the same time, a bill was drafted to mandate the use of the application [52], but later on, end up being rejected. Since then, the number of users has been

Figure 11: Covid Tracker Logo [55]

steadily declining and in January 2021 only 39% of the nearly three million people who installed the application continued to use it.

In hindsight, it is possible to point out 2 main issues: lack of trust/concerns with data privacy and lack of coordination. Lack of trust can be pointed to as the fundamental issue behind the slow adherence and fast decline of the application. There are many reasons for this sentiment, but a glaring one is a reliance on technology developed by Google and Apple. Both of which are known for possessing their fair share of privacy issues [53] [54].

Lack of coordination did also play a big role in the application's failure [51], "in the last five months, only 2708 codes were used."

The problem was caused by a lack of knowledge, since many of the infected did not know where to get the codes needed for the app, and many health professionals did not know where to find them.

Some examples of the overall mistrust are [50]:

- "People are losing confidence in the app because there are no codes";
- "There are no codes because doctors are poorly informed about how the app works and where the codes are found";
- "Since the application was launched, doctors have contacted us asking for help. It shouldn't be like that."

3.1.2 *Irish application*

Another interesting application is the Irish application (see Figure 11), also known as Covid Tracker. The application aims to help the national health system during the COVID-19 crisis. Among the most well-known features provided, are:

- Contact tracing - allows you to notify quickly and efficiently if a user has come into contact with an infected individual;
- Check-In COVID - allows to record and upload a user's symptoms to the HSE (Health Service Executive) on a daily basis, without revealing their identity;
- News and information - provides reliable figures, and relevant information regarding the pandemic and its current status in Ireland;
- The application gives its users the option to delete all data held by the application at any time;
- Other features - if permission is given, the application can collect meta-data. So that the HSE (Health Service Executive), DoH (Department of Health) and public health teams can monitor how the application contributes to the eradication of the virus.

Like many other applications, is entirely voluntary and, according to [55], each service works separately regarding the user's consent. It is possible to give / withdraw consent depending on the application service to be used. Other than that, the application works identically to "StayAway Covid". If the user is diagnosed as being infected, the CTC (Contact Tracing Centre) sends him a code via SMS. With this, the option to upload the keys of all contacts with whom the user has been in the last 14 days is unlocked.

In conclusion, the application also makes use of "COVID-19 Exposure Notifications", developed by Apple and Google. However, when compared with similar application, this one offers more fine grain whit data selection.

October 2020, around 2.1 million records were created with the "Covid Tracker" app, and around 5,800 people did receive close contact alerts. According to Colm Harte (Technical Director of the company that carried out the development of the application), a decentralized model was extremely important. Some of the citations found were [56]:

- "There were many concerns from the media, will it be effective? Will it work? And how will it impact my privacy?";
- "Are you going to inquire about me? Will you know my location? The combination of Bluetooth and the decentralized approach helped answer many of these questions";
- "I think one reason for its success was the fact that we dealt with privacy issues"

Figure 12: TousAntiCovid logo [57]

- "It gave people the confidence that they could trust that this application was doing what it was supposed to do and that it really is just an additional tool to help prevent the spread of Covid-19."

3.1.3 *French application*

Launched on June 2, 2020, the application "TousAntiCovid" (see Figure 12) also aims to detect and inform infected users of COVID-19, being the installation voluntary.

The application was developed by the National Institute for Research in Digital Science and Technology (Inria), and like many others, TousAntiCovid uses Bluetooth Low Energy to build a rogue list with which the user interacts.

If a user is diagnosed with COVID-19, he is given a code or *QR-Code*, which can be inserted into the application in order to share the list of contacts (IDs). If it is detected that the user comes into contact with someone infected, the application sends an alert requesting that they contact a doctor in order to perform a test on COVID-19.

The application provides up-to-date data about the coronavirus, such as the infection rate, the number of patients in intensive care and the number of application users. Users can also access *DépistageCovid*, a map of testing centers, along with up-to-date information on waiting times.

3.1.3.1 *Technical Analysis*

The application uses Bluetooth to measure the proximity between two users, and the data storage mechanism is based on the Robert protocol (Robust and Privacy-Preserving Proxy Tracing). The Robert protocol was developed by the National Institute for Research in Digital Science and Technology (Inria - France) and by the Fraunhofer Institute for Applied and Integrated Security (Germany). Adopting a centralized data storage system based on a *Federated architecture* infrastructure and making use of anonymous (temporary) IDs. Federated architecture (FA) is a standard in enterprise architecture that enables information interoperability between semi-autonomous / decentralized lines of business (LOBs). According to [58], the data collected is encrypted at random, and only data relating to the proximity and duration of the integrations are captured.

3.1.3.2 *Robert vs DP-3T analysis*

In the [58] paper, some comparisons are made between the *Robert* protocol and distributed solutions. It is apparent that the paper primarily refers to the DP-3T protocol (used in applications such as StayAway Covid and Covid Tracker)

Examples of this are:

- "While it may seem attractive in terms of privacy to adopt a fully decentralized solution, such approaches face important challenges in terms of security and robustness against malicious users";
- "Other schemes, qualified as "decentralized", transmit to each App information about the pseudonyms of infected users. This information allows the Application to decode the identifiers of infected users and check if any of them are part of its contact list. Our scheme does not follow this principle, as we believe that sending information about all infected users reveals a lot of information";
- "In this project, scores are calculated on a reliable server and used to notify users. At the same time it offers more flexibility to adapt scoring algorithms as needed, washing out more effective systems."

However, as mentioned in an *issue* present in the Github these statements have some flaws.

Figure 13: NZ Covid Tracer Logo [60]

3.1.3.3 *Adoption*

It should be considered that the application was originally launched under the name of StopCovid, and its name was only later changed to TousAntiCovid. According to [59], this was due in part to the lack of voluntary adoption by the population. Since in its original state (StopCovid), less than 5% of the French population had downloaded it. On October 22nd, a new update was released, with this being changed its name to TousAntiCovid (United Against Covid) and added some features, like for example:

- Updated information on the infection rate, the number of patients in intensive care, among others;
- Covid screening;
- Links to exemption certificates.

According to [59], this new version had over 4.5 million installs in the last week of October, considerably better than the first version.

3.1.4 *New Zealand Application*

NZ Covid Tracer (see Figure 13) is an application provided by the "Ministry of Health of New Zealand", which allows creating a digital diary of places visited by its users. It was developed to help contact individuals who have been exposed to COVID-19.

The use is optional for both individuals and companies, and the most outstanding feature it has is allowing to create a digital diary (App Diary) through the use of *QR-Codes*. For context, the first *QR-Code* system was developed in 1994 by the company Denso Wave, with the purpose of creating bar-code that could encode kanji, kana, and alphanumeric characters. A *QR-Code* is comprised of black squares

arranged on a white background, this can then be read with a camera and analyzed with the help of a smartphone. While standard bar-codes can only be read in one direction (top to bottom), *QR-Codes* can be read top to bottom and right to left, offering greater flexibility and allowing you to store a greater amount of data.

QR-Codes can be generated by companies or organizations, through the website of the "Ministry of Business, Innovation and Employment's Business Connect service".

If a user is identified as a potential carrier of the COVID-19, they are assigned a "tracker". Which will follow up on the individual, providing them with advice and assessing their well-being. It is also at this stage that the "tracker" could request additional information present in the App Diary.

As described on the [60] page, the application is extremely versatile, with some of the offered features, being:

- Digital diary - it is possible to choose which places / activities are stored in the digital diary.
- Contact details - you can choose which details you want to share. These will only be used if it is necessary to contact the user.
- Notifications - the user can show whether he wants to receive alerts in case he has come into contact with an infected individual.
- Bluetooth tracking - you can enable / disable this feature. The user can choose to use the application only as a digital diary if he so wishes.
- Details of your "National Health Index number" (NHI) - you can add your NHI to speed up test access if you visit a COVID-19 testing center.

According to [61], regardless of the alert level, all businesses and services that are not exempt need to display a *QR-Code* in a prominent place or near the entrance.

To register a new location, the user simply points the mobile phone camera at one of the posters that are easily available and waits for the application to confirm their disappearance. It is possible to detail the information that accompanies each *QR-Code*, data such as, who is following the user or what type of activity he is doing. All of this facilitates future investigations that may occur, giving the existence of a digital record of each interaction. This information will remain on the device for 60 days.

The application makes use of the Apple / Google Exposure Notifications, having a similar behavior to the StayAway Covid (Portugal) and Covid Tracker (Ireland) applications, in conjunction with the previously discussed digital diary.

Figure 14: Trace Together Logo [63]

Registration only requires an email address, being that name, phone number and home address are optional. The app can be downloaded and installed by people with registered accounts outside of New Zealand.

Companies opting to take part register on the government's Business Connect website [62]. This requires the company to have an identity certified with the government system (RealMe) and the company to have a New Zealand business number.

After registering the business locations and contact details, the registered company is issued with *QR-Codes* for each managed location, which must be downloaded, printed and displayed mainly on all entries of each location.

3.1.5 *Singapore Application*

For Singaporeans (see Figure 14), the Covid-19 pandemic is closely linked to technology, thanks to the various initiatives launched by the government. First, there is the SafeEntry application, whose aim is to create a digital check-in system, which records identification / mobile phone numbers whenever a user visits places of interest. The application makes use of spaces such as entrances or exits to restaurants, stores or malls, in which users can check-in / check-out, by scanning a *QR-Code*. Once this is done, the user is asked to fill in the identification number (NRIC) and his mobile phone number.

On March 20 (2020), the "Ministry of Health" together with the "Government Technology Agency (GovTech)" launched the TraceTogether application. It is a Contact Tracing application that exchanges Bluetooth signals over a short distance,

in order to detect other users. The information is subsequently stored locally (user phone) for 25 days. If a user is interviewed by the MOH as part of their contact tracking efforts, he/she may consent to submit their TraceTogether data to the MOH.

This facilitates the contact-tracing process and allows trackers to inform TraceTogether users as quickly as possible if an infected individual is located. For those who cannot or do not want to use the application, a TraceTogether token is also available. Small digital key rings that can be easily carried in bags or suitcases. The token exchanges short-range Bluetooth signals with tokens / mobile phones that have the TraceTogether app installed. If the user is infected, the device must be physically handed over to the authorities so that they can extract the data for tracking.

Using the TraceTogether application or its token became mandatory at the end of December 2020 in public places such as restaurants, workplaces, schools and malls.

3.1.5.1 *Technical Analysis*

The development of the TraceTogether application was based on the BlueTrace protocol, developed by the Government Digital Services team (Singapore).

BlueTrace was developed to make the process more scalable and combating limitations, such as the inability to identify all the people who were in contact with an infected individual. It is a protocol that seeks to record encounters between devices while protecting user data. When a user registers in the application, they are asked for a phone number, which is associated with a random UserID. The telephone number should be the only information requested from the user, contacting if prolonged exposure to infected individuals is detected. When two devices come into contact, a message (anonymous) with temporary IDs is played.

According to the paper "BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders" [64], the device in question stores locally a history of all interactions that occurred in a certain period (with TraceTogether the period is 25 days).

If a user is infected or there is a suspicion that he may be infected, he will be asked to share the history of interactions with the authorities responsible for the tracking.

3.1.5.2 *Privacy and Data Protection*

BlueTrace claims to protect users' privacy while giving them control over their information.

The protocol includes the following privacy protections:

- Collection of personal data limited to what is strictly necessary to contact the user (phone number);
- Local meeting history storage. This is only provided to the authorities if the user shares it;
- User identification means (ID's) change over time in order to discourage tracking by malicious users;
- Revocable consent. When consent is withdrawn, all personally identifiable data stored at the health center is deleted.

However, some concern has been expressed by the public after on January 4, 2021. The Minister of Internal Affairs (Desmond Tan) stated that the data collected by the application was under the Code of Criminal Procedure. Unlike previous statements made, this statement means that the police may use data from the TraceTogether and SafeEntry systems in criminal investigations unrelated to contact-tracing efforts.

According to [65], this statement came a day after the announcement that the data had already been used as evidence in a murder case. While Singapore's general attitudes about data privacy may not reflect what happens elsewhere in the world, contact-tracing applications have raised privacy concerns from day one. As mentioned in [66], news of this move by the Singapore government raises the concerns of activists/ethics experts about data misuse.

3.1.5.3 *Adoption*

On October 20, 2020, Lawrence Wong Minister of Education and co-chair of the task-force responsible for dealing with the Covid-29 pandemic, stated that having 70% of the population using the TraceTogether application is one of the main requirements for enter the third and final reopening phase. To achieve these results, measures were implemented, such as the distribution of tokens (started on September 14, 2020), the creation of check-in (SafeEntry) in schools, restaurants and shopping. Token distribution initially began at community centers, but demand soared after the government announced that TraceTogether would become mandatory to enter

the aforementioned locations. According to [67], Lawrence Wong stated that the search for tokens went far beyond what was initially expected, leading to delays in their manufacture and distribution. This led to the postponement of the initially proposed calendar, with the obligation being put into practice only on January 4, 2021. However, according to [68] already on December 21, 2020, the adoption rate had passed the 70% initially stipulated. In this way, it is possible to enter the third and last phase of deconfinement.

3.2 CONTACT TRACING APPLICATIONS - DECENTRALIZED SYSTEMS

These applications are only some examples where social factors remain the driving force behind the lack of adherence. To combat the negative sentiment, some key points need to be taken into consideration they are:

- Data transparency: a user should be able to identify where and how their data is being used;
- Individual control of data: a user should be able to choose what information is being shared;
- Decentralized chain of power: it should not be fully reliant on the goodwill of a company (Google or Apple) to work.

That being said, the proposed way to combat all these issues is by shifting the application from a centralized chain of power to a decentralized system. Where each user has control over the information and can freely interact with it, instead of it being a black box. A more effective system to accomplish this is *Distributed Ledger* or, more accurately, an application that relies on Blockchain technology to store the user information.

3.2.1 *BeepTrace*

According to [70], "engineers from the University of Glasgow outline how a trustworthy contact tracing system could be built on the unique properties of *Distributed Ledger* technology." In the same article, [69] is mentioned how traditional methods pale in comparison with new, more technologically advanced ways of doing contact tracing. And how an application like the TraceTogether helped the government of Singapore to contain their coronavirus outbreak [71].

Figure 15: Beep Trace Logo [69]

Simultaneously, it was proposed a new mobile-based system named BeepTrace [69] (see Figure 15), which hopes to "harness the unique properties of *Distributed Ledgers* to create a decentralized, potentially international system to help break the chains of virus transmission." It works by assigning a randomly generated ID, which changes regularly to prevent tracking or identification. If a test gives a positive for COVID-19, it shares the IDs gathered by the application over the past 14 days with the ledger. Notifying any user that was in contact during the period.

In the same paper, it is also mentioned how the application possesses two different modes, a passive and an active. The passive mode works by relying on GPS information to gather where the user was and who was the user in contact with. The active mode works by relying on Bluetooth to register users in close contact for a prolonged period.

3.2.1.1 *DAG based solutions*

A DAG based solution represents an alternative to the traditional blockchain that aims to improve speed, scalability, and cost. However, it is still a *Distributed Ledger*. It relies on a graph structure with directed edges and where no vertex should lead back to himself. Usually, there are no blocks, being the most considerable difference the way they add transactions to the network [72]. According to [73] because of the adoption of graph structure, it can do the processing of transactions in parallel. This is in direct contrast to the sequential manner used in chain-based solutions.

3.2.1.2 *Chain-based solutions vs Directed Acyclic Graph based solutions*

When comparing Chain-based solutions vs DAG-based solutions, it is possible to recognize some key differences. Chain-based solutions offer transparency and immutability, but lack scalability when it comes to performance since it often relies

on consensus algorithms like Proof of Work [74]. These algorithms not only limit the number of transitions that can be processed in a given amount of time, but also consume copious amounts of computing power and electricity. Establishing them as not the best-suited solution for high volumes of transactions. DAG offers more efficient scaling and the reduction/avoidance of fees, but as a result, this may make it vulnerable to attacks. That is why many of the DAG based solutions have to rely on centralized features like central co-ordinators, pre-selected validators, 'witness' nodes, or completely private network systems.

3.3 COMPARISON

The Table 1 present in the Appendix A shows us a brief comparison between all the described applications and their usage rate. Only exceptions are NZ Covid Trace, that shares *QR-Codes* scanned instead of a number of users, and BlueTrace that is not on the same scale as the other applications.

By looking at the data, we can also see that all of them, in one way or another, rely on Bluetooth for contact tracing, even if, is with some Small divination's.

3.3.1 *Centralization vs Decentralization*

When it comes to centralization versus decentralization, we can find a mix. Applications like NZ Covid Tracer, TousAntiCovid and TraceTogether tend to be inherently more centralized because of the extra feature they pack. For example, NZ Covid Tracer will share anonymous data regarding the amount of QR-codes scanned even if no infection was found.

On the flip side, applications like StayAway Covid and Covid Tracker are inherently decentralized, at least, up to a certain point. All data gathered will stay only on the user's phone, unless it deemed the user infected. If that is the case, the IDs will be sent into a central reporting server [75].

At last, applications like BeepTrace are fully decentralized, and achieve this by relaying on *Distributed Ledger* technology.

3.3.2 *Voluntary vs Involuntary*

Currently, only the TraceTogether application is not voluntary.

And although this is an easier way to increase the number of users, it has too many caveats in order to be applied for every country. To list a few:

- Not every person has a smartphone.
- Not every country has the same number of persons using a smartphone.
- Trust in the government varies widely from country to country, same applies to trust in the application.
- It is not cost efficient to enforce this law.

3.3.3 *Data usage only within the app*

This is one of the fundamental metrics to build trust.

If look at the Table 1 one again, we can see that only TraceTogether does not comply with this metric (the only mandatory application within the comparison).

Since then, information regarding the use of this data by police informant has made the news.

Quoting [63], "TraceTogether data may be used in circumstances where citizen safety and security is or has been affected".

To further emphasize why this is of concern, we can give the example of the South Korea application. In South Korea, surveillance techniques were used to get around the problem of people not being able or willing to reveal close contacts. According to the article [76], in 2015 a law passed in response to the (MERS) outbreak was passed that allows authorities to use credit card and telephone data to track a person's movements and identify other people who may have been exposed to the virus.

3.3.4 *Data destruction*

At last, data destruction. All applications agreed on destroying the data collected once the pandemic is over or the application is of no use.

3.3.5 *Possible Concerns*

Besides all this, there are many flaws that may affect an application usage rate, which end up drastically reducing its effectiveness [76]. Among them are people being infected without knowing, as well as waiting time for test processing, the impossibility of identifying all the contacts an infected person has had, among many others.

As pointed out in the article [77], an intervention of this type raises ethical issues related to access, transparency, protection, and use of personal data, all social factors that need to be considered.

4

EXPOSURE TRACKER LEDGER - REQUIREMENTS ELICITATION

After this brief analysis, is clear that social factors remain the driving force behind the lack of adherence. In order to combat the negative sentiment, some key points need to be considered. First, any application that hopes to succeed needs to be completely transparent with its users. It is of the at most importance for the application to provide a way for the user to verify / validate what data is being shared. Second, the user needs to be in control of what it is shared. Third, the application must not force the user into taking action, it should only advice what is the best course of action. Fourth, the application should be independent from outside interference, it should not be reliance in a private company like Google or Apple to work.

4.1 PRIVACY PROTECTION

Any proposed solution must preserve the privacy of its users. And at the same time, be robust against attacks aimed at decreasing system performance or reliability. If not, users quickly lose confidence in the application, something that has been proven to be one of the main factors that lead to low adoption rates. Only in this way is it possible to avoid abuses of power as described in the article "Singapore's police now have access to contact tracing data"[78]. As per the Criminal Procedure Code, the Singapore Police Force can get any data, including data from TraceTogether [66].

A *Distributed Ledger* that stores the user information is the best way to accomplish this goal. Because of this it was developed a small proof of concept that should mitigate the key point mentioned above while increasing thrust from the end user.

Like any contact tracing application, it should be able to:

- Detect if 2 users are in close contact;
- In case of a user infection, share this information;
- Notify its contacts in case of interaction with an infected user.

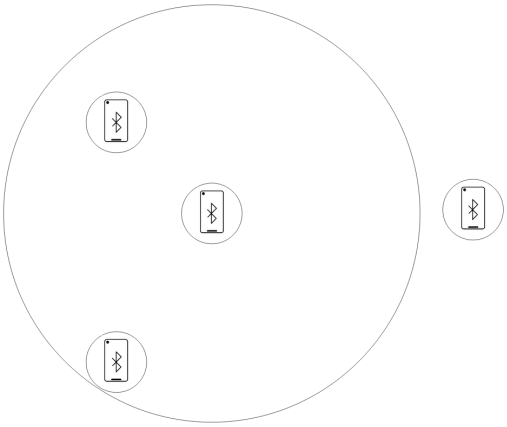

Figure 16: Users within 2m range are recorded

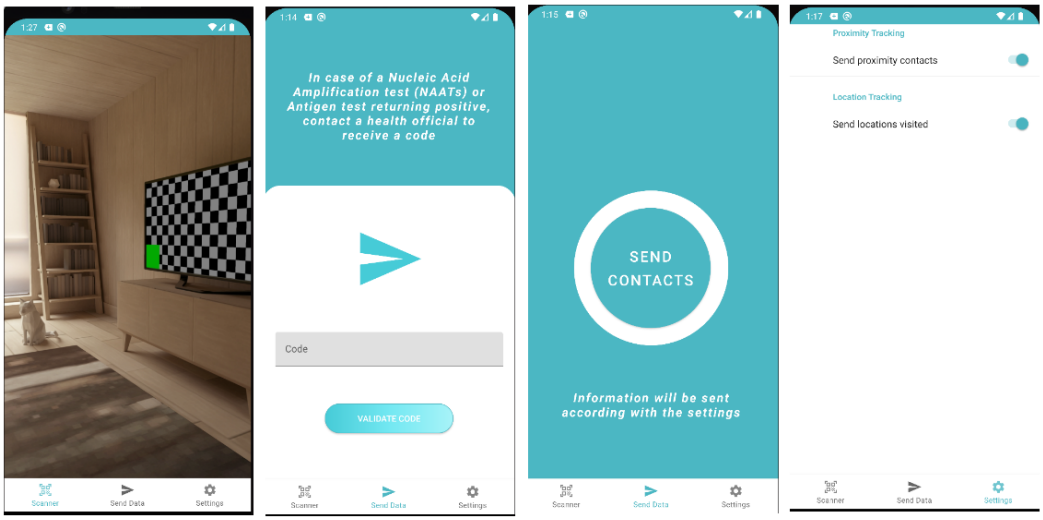

Figure 17: App Summary

However, it should also:

- Only ask for information if strictly necessary;
- Allow for custom submission of information;
- Offer a way to keep track of what was submitted;
- Keep shared information public.

4.2 DETECTION OF CLOSE CONTACTS

As the name entails, the primary function is to contact trace. We propose to achieve this by making use of proximity tracking (see Figure 16).

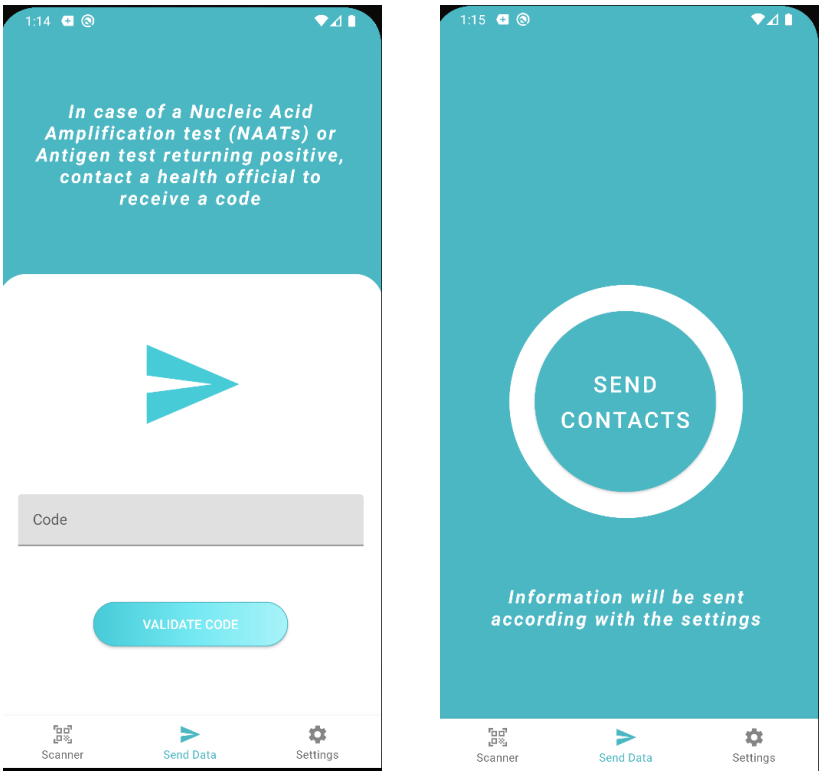

Figure 18: Submit contact views

For a new contact to be recorded, a couple of procedures need to happen. First, the user needs to have the application setup done (see Figure 17 for an overview), this entails, permissions given and Bluetooth turned on. Second, another user with application working needs to enter close contact for an extended period of time. According to the WHO, for a contract to be deemed of interest, it needs to be for an extended period (around 15 minutes) and within 2 meters of distance.

4.3 SUBMISSION OF INTERACTIONS

When a user is deemed infected, a health professional or other entity with similar access, should give him a code. By inserting it into the application, a view containing a submit button will appear (see Figure 18). Once the user presses this button, the application will send all information collected in the past 14 days into the ledger.

For this to happen, the user is required to have a positive test for COVID-19. This requirement is often used to avoid misbehavior from ill intended users.

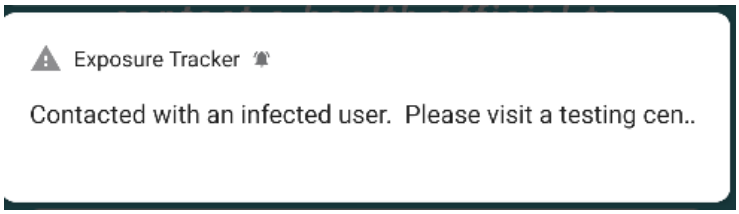

Figure 19: Contacted with an infected user (Notification)

4.4 NOTIFYING ABOUT POSSIBLE INFECTIONS

After everything is set and done, ensuring that everyone that was in contact with the infected user is rightfully notified is of the utmost urgency. Because of that, the application will listen to every submission in order to validate if the user was or not in contact with the infected subject. And if this is the case, the application will send a notification informing the User to get tested (see Figure 19).

5

EXPOSURE TRACKER LEDGER - DESIGN AND IMPLEMENTATION

During the development there were multiple challenges / choices to be made, as such each step will be described in the following manner:

- Detection of close contacts;
- Interaction registration;
- Submission of interactions into the ledger;
- Transaction family;
- Events;
- Notifications;

By looking at the Figure 20, we can see an overview of the architecture. We have 2 clients, in this case 2 smartphones, interacting over Bluetooth. In case of infection, one of them will submit the information gathered into the Node and, after validation, this will be submitted into the ledger. Once this is done, both applications will be notified of the change in state. And all contact will be promptly notified.

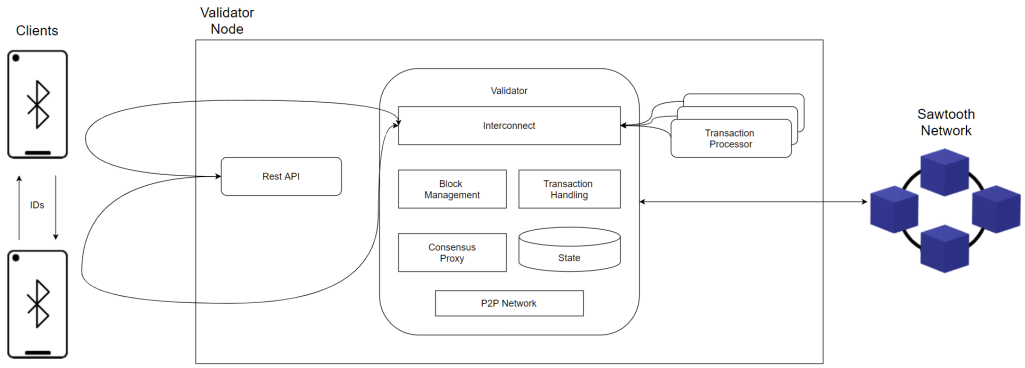

Figure 20: Sawtooth Architecture Diagram [36]

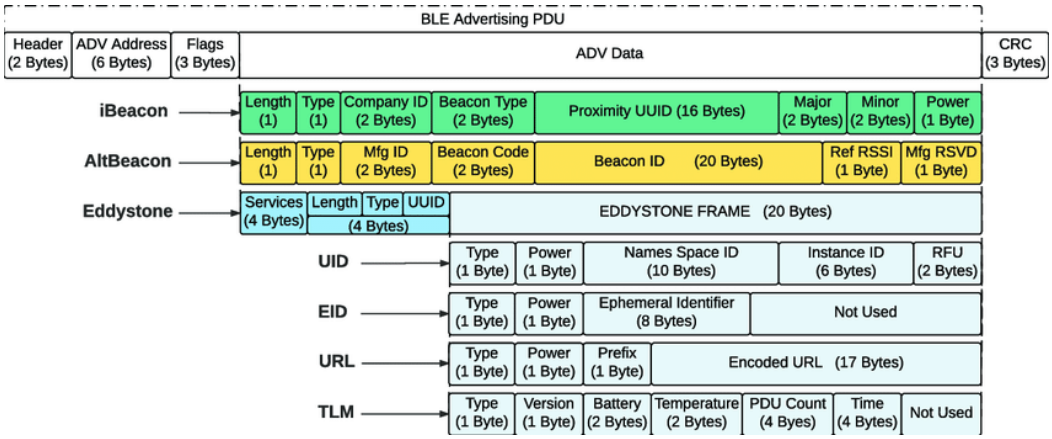

Figure 21: Beacon comparison [79]

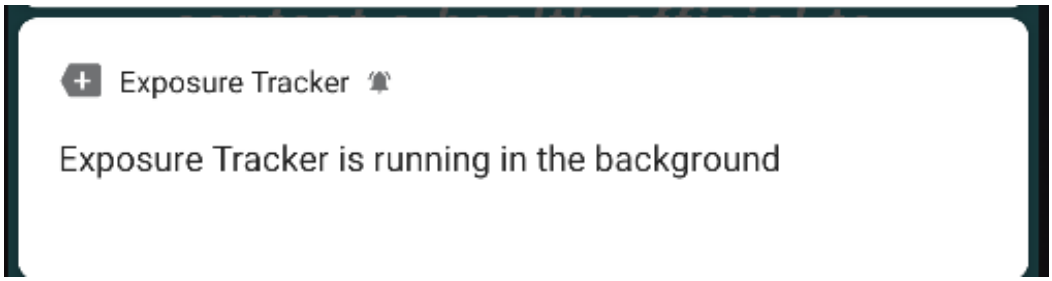

Figure 22: Beacon running in the background (Notification)

5.1 DETECTION OF CLOSE CONTACTS

The detection of close contacts relies on having an active beacon responsible for measuring the distance to any other user. For this to work, the application needs to be running in the background, and because of that, it will display the notification present in the Figure 22.

To achieve this result, a choice of technology had to be made, therefore a couple of beacons were tested (see Figure 21), but in the end, it boiled down to the AltBeacon protocol [80].

AltBeacon is a protocol that defines a format for proximity beacon advertisement, with it, they can signal their proximity to nearby receivers. The contents of the emitted message contain information that the receiving device can use to identify the beacon and to compute its relative distance to the beacon.

The content of the emitted message allows the receiving device to get the user random ID and compute the relative distance between each other [80].

5.2 INTERACTION REGISTRATION

When 2 users interact, a random ID is shared across both of them. From the point of view of the application, there are 2 IDs to consider. The first is the ID gotten through the AltBeacon, and ID represents the User with whom the interaction took place. The second ID is the User current ID. It is stored, because later on will be compared to find if the user was or not in contact with an infected user.

This information is all stored in a local database, in order to preserve privacy. Some advantages achieve with this are:

- All sensitive data is kept on the user smartphone.
- No need to rely on third parties to store the information.
- It is less data being submitted into the ledger.
- In case some sort of data breach happens, where it becomes possible to identify the user by their random generated ID. Only infected user will be affected.

For this, the database implementation chosen was SQLite. SQLite is a library that implements a small, fast, independent and highly reliable SQL Database engine [81]. Unlike most databases on the market, SQLite is not a client-server database, since it records the information directly in files on the disk. As mentioned by ([82]) "A SQL database complete with multiple tables, indexes, triggers and Views, all contained in a single file". Among the major advantages of its use are ([81]):

- Performance - Reading and writing to a SQLite Database tends to be faster than reading and writing files individually to disk.
- Reduced cost and complexity - Database content can be easily accessed / updated via SQL queries. Adding new tables to the damage base does not make old versions of it incompatible.
- Portability - Diverse content that could be stored as a "stack of files" is encapsulated in a single file.
- Reliability - The content can be updated continuously and automatically without major loss of information due to failures.
- Accessibility - Because of the ubiquitous use of this type of database, there is a lot of information / ways to interact with the data.

- Good integration in mobile applications - As SQLite Databases don't require administration, they tend to work well on mobile devices / *Internet of Things* without any kind of specialized support.

5.3 LEDGER TECHNOLOGY AND STORAGE

As for the choice of technology for storage and sharing the information, we went with Hyperledger. This was in fact, the underlying motif for the development. At first, a couple of different technologies were tested. As mentioned in the chapter "Blockchain 3.0 Hyperledger", Hyperledger is home to a lot of different approaches being the most popular Hyperledger Fabric and Hyperledger Sawtooth, these being the ones that divided our attention. As everything they have, their differences, as for Hyperledger Fabric, it "uses fine-grained governance over participation in the network by Membership Service Providers (MSP)" [83]. For this purpose, we can boil down this statement to Hyperledger Fabric requires a permissioned network in order to function.

On the other hand, Hyperledger Sawtooth offers support for both permissionless or permissioned networks, giving it a little more flexibility. Permission level was the defining factor and is often mentioned as the most significant difference between both implementations, and because of that, we went with Hyperledger Sawtooth.

Regarding implementation, other than Hyperledger Fabric. Hyperledger Sawtooth offers a set of features that help tip the scale in his favor, they are [84]:

- Parallel Transaction Execution - While many blockchains use serial transaction to ensure consistent when ordering nodes. Hyperledger Sawtooth relies on a complex parallel scheduler to do transaction in parallel. Leading to a significant boost in performance.
- Separation of Application from Core - it provides a clear separation between the application level and the core system level. It provides *Smart Contract* abstraction in order to allow application developers to write contract logic in any language. Quoting the official documentation [36], "An application can be a native business logic or a *Smart Contract* virtual machine. In fact, both types of applications can co-exist on the same blockchain. Sawtooth allows these design decisions to be made in the transaction-processing layer, which allows multiple types of applications to exist in the same instance of the blockchain network."

```
123456 0000000000000000000000000000000000000000000000000000000000000000:'
{"user":<UserIdentifier_1>,"uuis":<List<UUI>>;"user":<UserIdentifier_2>,"uuis":<List<UUI>>;...}
```

Figure 23: One address for all data

- Custom Transaction Processors - Each application can define the custom transaction processors to meet its unique requirements.
- Dynamic consensus - Hyperledger Sawtooth allows the use of a vast amount of different consensus algorithms, to list a few, Proof of Elapsed Time, Practical Byzantine Fault Tolerance, Devmode, Raft, and PoET CFT. With this, we can also point out the added benefice of supporting Byzantine Fault Tolerance consensus.
- Interoperability with *Ethereum* - Despite it not being used during the implementation, Hyperledger Sawtooth supports running *Ethereum Virtual Machine Smart Contracts*, thanks to the use of the *Sawtooth-Ethereum integration project* (Seth).

5.4 STORAGE MODEL

Like mentioned on Hyperledger Sawtooth [85], when storing data into the ledger, we can choose between 3 alternatives:

- All of our data is stored at a single address;
- Each user information is stored in its own address;
- Each user information is stored over multiple addresses.

5.4.1 *Single address*

As we can see in the Figure 23, the first option is not efficient, since a single address contains all information for the application. If we want to do a comparison, this would be the same as instead of using a database for data storage, all data was stored in a single file. A file that would need to be read / write every time a change was made.

```
// User_1
1234563c9909afec25354d551dae21590bb26e38d53f2173b8d3dc3eee4c047e7ab1c1:'{"<List<UUI>>}
// User_2
1234573c9909afec25354d551dae21590bb26e38d53f2173b8d3dc3eee4c047e7ab1d6:'{"<List<UUI>>}
```

Figure 24: One address per User

```
// User_1
1234563c9909afec25354d551dae21590bb26e38d53f2173b8d3dc3eee4c047e7af0bc:'{"trackingUUIDs":"<List<UUI>>}
1234563c9909afec25354d551dae21590bb26e38d53f2173b8d3dc3eee4c047e7a384f:'{"locationUUIDs":"<List<UUID>>}
// User_2
1234563c9909afec25354d551dae21590bb26e38d53f2173b8d3dc3eee4c0400000002:'{"trackingUUIDs":"<List<RandomUUI>>}
1234563c9909afec25354d551dae21590bb26e38d53f2173b8d3dc3eee4c0400000003:'{"locationUUIDs":"<List<UUID>>}
```

Figure 25: Multiple addresses per user

5.4.2 *One user per address*

For all intends and purposes, currently this is the Data Model in use. Since there is no need to save anything other that the Random UUIs generated for each user. As shown in Figure 24, each user that was able to press the submit button has one address (computed by the node based on a key), and on that address, are stored all random IDs generated.

5.4.3 *One user multiple addresses*

At last, this is the most complete option, especially when storing more than one type of data. As we can see in the Figure 25, if there were 2 distinct data types for each user (trackingUUID and locationUUID), this is the ways to go, since we can have one in each address.

5.5 THE IMPLEMENTATION OF A NODE

In order to have the network up and running, we needed to set up at least one Node. Their required components are:

- A transaction Processor - Part of the Transaction Family responsible for containing the business logic;
- A rest API - In order to communicate with the node;

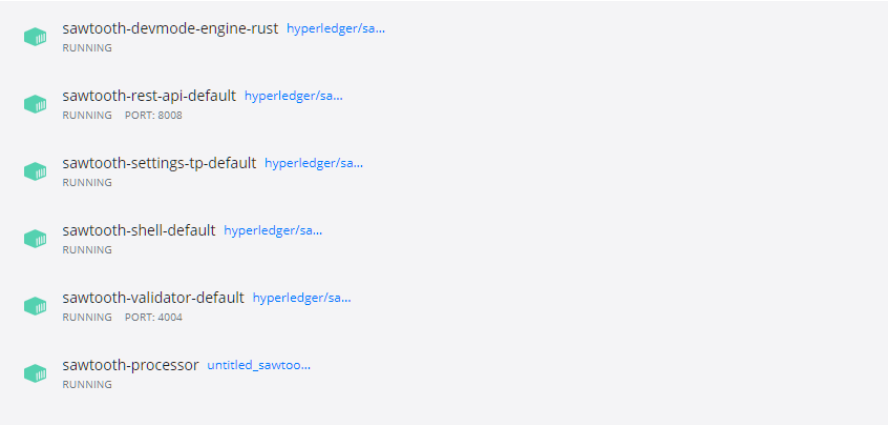

Figure 26: Simulation of a basic node

- A validator - Responsible for validating batches of transactions, combining them into blocks, maintaining consensus with the network, and coordinating communication between clients, other validators, and transaction processors [86];
- Consensus engine - The consensus algorithm used by the network.

Seething all of this up, is made easier by using Docker [87] to build the environment.

5.5.1 *Docker*

Docker being an open-source platform for container-based application development that allows the separation of the application from the infrastructure. And container being the standard software unit, that allows the execution of applications in a loosely isolated environment by encapsulating everything it needs in order to run, while being environment agnostic.

By looking at Figure 26, it possible to see the simulated node up and running.

5.5.2 *Transaction Family*

Listing 5.1: Submit data into state

```
@Override
public void apply(TpProcessRequest transactionRequest,
   State state) throws InvalidTransactionException,
   InternalError {
```

```
    try {
        String action = new
            JSONObject(decodeData(transactionRequest).toString())
                .getString("ACTION");
        if
            (action.equalsIgnoreCase("publish_positive"))
            {
             headerPublicKey =
                transactionRequest.getHeader().getSignerPublicKey();
             String submittedData = new
                JSONObject(decodeData(transactionRequest)
                    .toString()).getString("KEY");
             submitRecord(state, submittedData,
                headerPublicKey);
        }
    ...
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Keeping it simple, this Snippet 5.1 of a custom transaction family validates what type of request was done and where it should be sent. In this case, we validate if the action present on the request is "publish_positive" and if so, we decode the payload and save the information into an address on the ledger state. Being that "publish_positive" is the action reserved to add information about infected users, more precisely the list of random IDs.

5.5.3 *Consensus algorithm*

At first, tests were done with a single node, and because of that, the consensus used was the most simple one, also known as Sawtooth dev mode. Usually is only intended for testing and not production, offering a simplified random-leader algorithm. However, it commits batches as fast as possible, providing quick feedback, ideal for development. That being said, other consensus were tested, to list a few: Proof of Elapsed Time (PoET) and Practical Byzantine Fault Tolerance (pBFT).

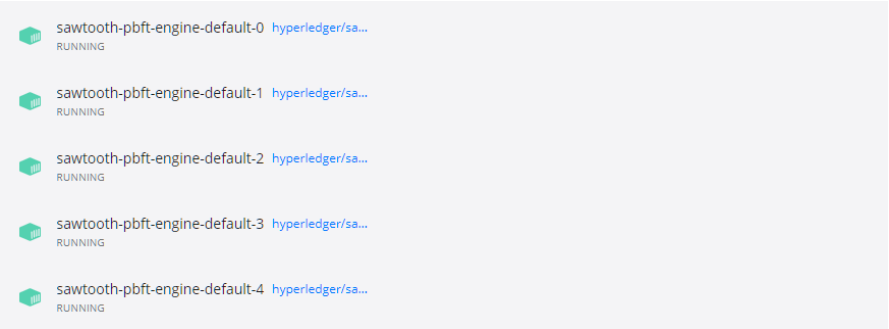

Figure 27: Simulation of 5 nodes running at same time

Starting with PoET, it is a consensus mechanism that prevents high resource utilization and high energy consumption by following a fair lottery system where every single node is equally likely to be a winner, by spreading the chances of a winning fairly across the largest possible number of network participants. The algorithm makes use of a randomly generated elapsed time to decide mining rights and block winners on a blockchain network. The timer is different for each node, being that every participant is assigned a random amount of time to wait, and the first participant to finish waiting gets to commit the next block to the blockchain.

Practical Byzantine Fault Tolerance, on the other hand, is a voting-based consensus algorithm, that like the name entails, provides Byzantine fault tolerant. A pBFT distributed system, does ordered sequentially with one node being the leader. As for the others, they are referred to as secondary or backup nodes. That said, the network safety is achieved even when some nodes are faulty or malicious, as long as a minimum number of nodes stay connected, working properly, and behaving honestly. The goal of an honest node is to agree on the system's state, based on the opinion of the majority [88].

Is also important to know that pBFT does not support open enrollment, being required for each node to be added by an administrator.

With regard to testing the interaction between different nodes, 5 of them were simulated using Docker (see Figure 27).

5.6 SUBMISSION INTO THE LEDGER

In order for the application to submit the generated IDs (see Figure 28), some steps need to be followed.

- Fetch and parse the Payload;

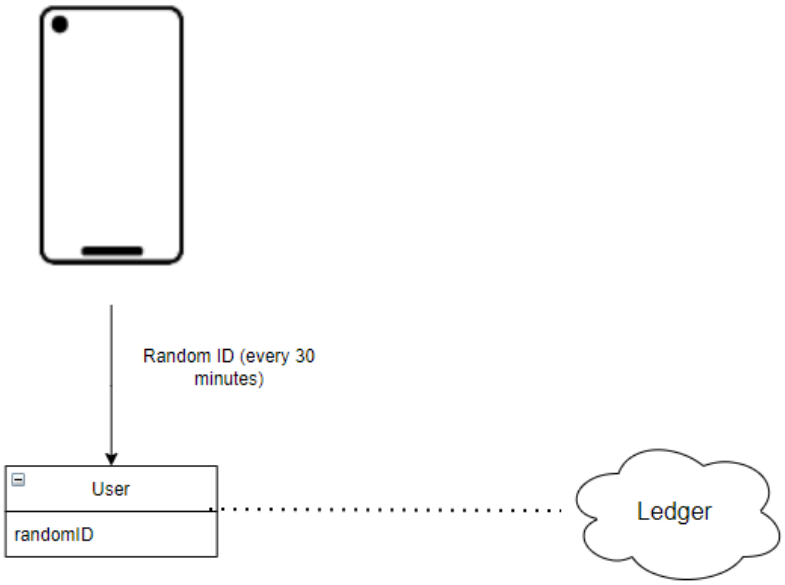

Figure 28: The generated IDs are submitted into the ledger

- Build a Transaction;
- Submit to the Rest API.

5.6.1 *Fetch and parse the Payload*

Listing 5.2: Fetch IDs and serialize the data

```
public void onClick(View view) {
    List<ByteArrayOutputStream> listOfPayloads = new ArrayList<>();
    BlockchainController requestHandler = new
       BlockchainController(BlockchainHandler.getPrivateKeyFromString(userService.getAddress()));
    ByteArrayOutputStream payload = new ByteArrayOutputStream();
    String publicIdJason = new
       Gson().toJson(userTrackingService.getAllIdentifiersGenerated(numberOfDays));
    try {
        new CborEncoder(payload).encode(new CborBuilder()
                .addMap()
                .put("ACTION", "publish_positive")
                .put("KEY", publicIdJason)
                .end()
                .build());
    } catch (CborException e) {
        e.printStackTrace();
    }
    listOfPayloads.add(payload);
    requestHandler.wrapAndSend(Endpoints.URL.value, listOfPayloads);
}
```

As shown in the Snippet 5.2, first we need to fetch all the IDs that were generated in the past 14 days and parse them into a JSON. And once this is done serialize the information with the help of CBOR Encoder. For reference, Concise Binary Object Representation (CBOR) is a data format whose design goals include the possibility of extremely small code size, fairly small message size, and extensibility without the need for version negotiation [89].

5.6.2 *Build a Transaction*

Listing 5.3: Steps for submitting transaction

```
public void wrapAndSend(String baseUrl,List<ByteArrayOutputStream> listOfPayloads){
    List<Transaction> transactions = new ArrayList();
    byte[] payloadBytes;
    for (ByteArrayOutputStream payload:listOfPayloads) {
        payloadBytes = payload.toByteArray();
        // Create Transaction Header
        TransactionHeader header = getTransactionHeader(address, payload);
        // Create Transaction
        Transaction transaction = getTransaction(payloadBytes, header);
        // Add transaction to list of transactions
        transactions.add(transaction);
    }
    List<String> collect =
       transactions.stream().map(Transaction::getHeaderSignature).collect(Collectors.toList());
    // Create Batch Header
    BatchHeader batchHeader = getBatchHeader(collect);
    // Create Batch
    Batch batch = getBatch(transactions, batchHeader);
    ByteString batchListBytes = BatchList.newBuilder().addBatches(batch).build().toByteString();
    submitTransaction(baseUrl, batchListBytes);
}
```

Citing the Hyperledger Sawtooth documentation [36], "Transactions are the basis for individual changes of state to the Sawtooth blockchain.They are composed of a binary payload, a binary-encoded TransactionHeader with some cryptographic safeguards and metadata about how it should be handled, and a signature of that header".

Like mentioned in the Snippet 5.3, the steps to be followed are:

- Create Transaction Header - Contains information for routing a transaction;
- Create Transaction - Is composed of the Transaction Header as Bytes, signature and the payload;
- Create Batch Header - Contains the public key of the signer and the list of Transaction ID;
- Create Batch - Batches are the atomic unit of change in Sawtooth's state, and they are required no matter the amount of transactions you may want to do. For simplicity's sake, they may be seen as a list of transactions.

5.6.3 *Submit to the Rest API*

Listing 5.4: Submit transaction into Rest API

```
private void submitTransaction(String baseUrl, ByteString batchListBytes) {
    try {
        Request request = new Request.Builder()
                .url(baseUrl + "batches")
                .post(RequestBody.create(MEDIA_TYPE_MARKDOWN, batchListBytes.toByteArray()))
                .build();
        client.newCall(request).enqueue(new Callback() {
            @Override public void onFailure(Call call, IOException e) {
                e.printStackTrace();
            }
            @Override public void onResponse(Call call, Response response) throws IOException {
                try (ResponseBody responseBody = response.body()) {
                    System.out.println(responseBody.string());
                }
            }
        });
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

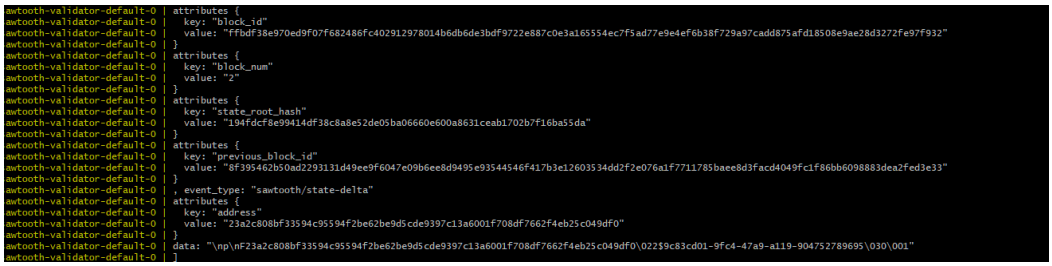

Figure 29: Validator Broadcast

After the transaction is ready, and the Batch fully formed, we can now make our request to the node. To achieve this, a request has to be made into the Hyperledger Sawtooth Rest API (see Snippet 5.4). Once that is achieved, the validator will ensure everything on the request is according to specification and, if so, it will be passed into the Transaction Handler. This is the same Transaction Handler mention in the subsection “3.5.5.2 - Transaction Family” , that will validate what type of request was done and where it should be sent.

5.7 LISTENING FOR CHANGES

Like was said in the chapter “3.4 - Notifying about possible infections” it is paramount that everyone that was in contact with the infected user is rightfully notified. In order to achieve this, Sawtooth events were used. Sawtooth events occur when blocks are committed and are broadcasted by the Validator when the commit succeeds (see Figure 29). One of their major advantages is that they can be be filtered or queried in order to customize the information gathered.

As for the event subscription Snippet 30, this is in part abstracted by the Sawtooth SDK. However, the underlying implementation is based on ZeroMQ [90].

As for composition, an event has three parts [36] (see Figure 31):

- Event type;
- Payload;
- List of attributes.

By default there are 2 distinct types of event:

- Block-commit - This event occurs when a block is committed and contains information regarding the block (block ID, block number, state root hash, and previous block ID).

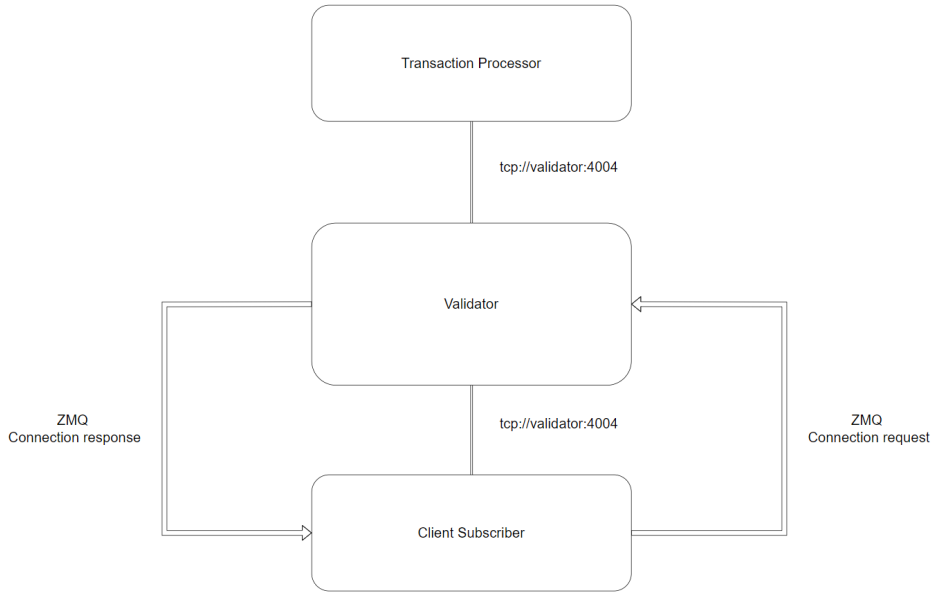

Figure 30: Event subscription

```
message Event {
  // Used to subscribe to events and servers as a hint for how to deserialize
  // event_data and what pairs to expect in attributes.
  string event_type = 1;

  // Transparent data defined by the event_type.
  message Attribute {
    string key = 1;
    string value = 2;
  }
  repeated Attribute attributes = 2;

  // Opaque data defined by the event_type.
  bytes  data = 3;
}
```

Figure 31: Event protobuf [91] composition

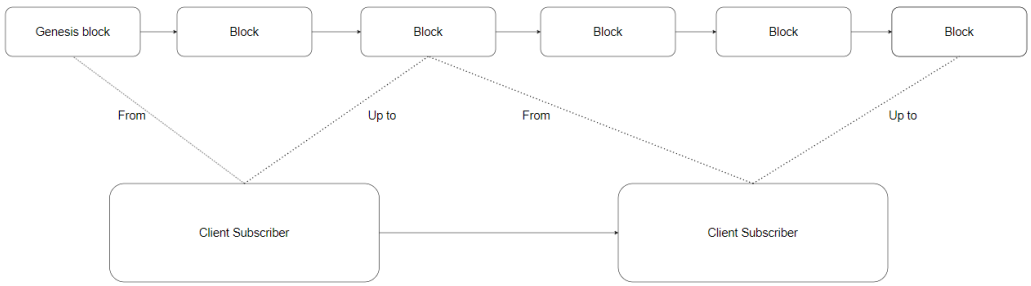

Figure 32: Fetching only the required amount of information

- State-delta - This event also occurs when a block is committed but in contrast has all state changes that occurred at a given address.

For our use case, the focus will be on State-delta, since it can tell us everything that changed inside the ledger. That being said, by default, State-delta will give us all information that changed from the start. This represents a problem, since any request will lead to the information being downloaded repeatedly. Fortunately, Hyperledger Sawtooth let us specify the "LastKnownBlockIds". This corresponds to the ID of the first block we want to listen to. A representation of this behavior can be seen on the Figure 32.

Listing 5.5: Listen for changes

```
private void listenForChanges(Boolean listening, Gson jsonParser, BlockchainController requestHandler) throws
    InvalidProtocolBufferException {
    // Listen for as long as possible
    while (listening) {
        // Fetch all events
        EventList eventList = EventList.parseFrom(eventStream.receive().getContent());
        eventList.getEventsList().forEach(event -> {
            // Filter only the events you need
            if (event.getEventType().equals("sawtooth/state-delta")) {
                event.getAttributesList().forEach(attribute -> {
                    // Get the address where changes where made
                    if(attribute.getKey().equals("address")){
                        try {
                            getDataFromAddress(requestHandler, jsonParser, attribute.getValue());
                        } catch (ExecutionException | InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                });
            }
        });
    }
}
```

	Client Signing			Transaction Processor			State Delta		
	Complete?	Stable API?	Maturity	Complete?	Stable API?	Maturity	Complete?	Stable API?	Maturity
Python	✓	✓	1	✓	✓	1	✓	✓	1
Go	✓	✓	1	✓	✓	1	✓	✓	1
JavaScript	✓	✓	1	✓	✓	2	✓	✓	2
Rust	✓		1	✓		1	✓		1
Java			3			3			3
C++			3			3			3
Swift			3	N/A	N/A	N/A	N/A	N/A	N/A

Figure 33: SDK Maturity [36]

After fetching the field “data” from the event Snippet 5.5, we grab the attribute “address“ in order to know where the change of state took place. With this information, a requester is done, getting the current state for that address.

5.8 CHALLENGES

As is usual for every project, there were many challenges both in planning and implementation. However, there is one that stands above the rest. And this had to do with the language chose for the implementation.

As is now apparent, the development was done in Java, with the help of Sawtooth SDK to ease the development requirement. At the time of conception, this seemed to be a fairly normal choice, since java was a known language and Sawtooth was pretty open about supporting a vast amount of them. To drive the point even further, we can see mentions of Java as one of the major languages all the way back to version 1.0.

- 1 - Recommended: Well supported and heavily used;
- 2 - Community support only (core maintainers do not update these SDKs);
- 3 - Experimental: Might have known issues and future API changes.

Unfortunately, as we can see in the image 33, it seems the support for the Java SDK was left behind over the years, leading to an unstable and bug prone SDK. Even worse, the documentation was servery outdated, or was just straight up missing information. Example of this is docs about Java SDK only having information regarding how to build a transaction from Client side, but zero information on how to create a node, create a Transaction Family, or listen to event.

```
Event {
  event_type = "sawtooth/state-delta",
  attributes = [Attribute { key = "address", value = "abc...def" }],
  event_data = <bytes>
}
```

Figure 34: State example [36]

To bypass the lack of documentation, we had to rely on old Hyperledger Sawtooth examples available on GitHub. These were rare and most often than not servery outdated.

Lack of documentation was also a problem when trying different consensus algorithms. These usually require complex rules, but all Hyperledger Sawtooth offers in the official documentation is small examples usually with a small number of nodes (1 and 2).

Keeping with the same topic, the most notable issue that stems from the lack of SDK support, had to do with the Sawtooth Events. While, the idea behind an event is fairly simple, “Subscribe to the type of event you want and access the data field“ unfortunately, this not the case. Usually, an event of type “state-delta“, has the structure presented in the Figure 34, with data corresponding to the change made to the current ledger state. The SDK offers a way to get the value by using “getStringUtf8”. Unfortunately for Java SDK, this does not return the expected result. Something that is not a problem for other languages, as tests were done for Python and JavaScript with no issue.

6

CONCLUSION AND FUTURE WORK

Although there are multiple examples of contact tracing applications, it is clear that there was a widespread lack of trust in them. This resulted from issues like the heavenly reliance on private companies like Google and Apple, the lack of transparency about how the data is being used, the inability to choose what was shared, and sometimes trying to impose its use without addressing the underlying issues. This lack of trust was the main reason for the lack of adoption by the general population, and the fast decline in users after the first issues became widespread. Resulting in a feedback loop where the fewer users actively using it, the less useful it is, followed by even fewer users using it.

By using blockchain technology, it is possible to solve many of the concerns related to the mistreatment of data, allowing every user to check what information is being used and how is it being handheld.

It also needs to be pointed out that Hyperledger is a significant performance improvement in the handling of parallel transactions when compared with *Bitcoin* and *Ethereum*. With the appearance of new alternatives like for example *Hyperledger Sawtooth*, it is now possible to develop large-scale applications without having to fear performance issues or suffering the shortcomings of alternative technologies like DAG-based solutions. By making use of the *Sawtooth SDK*, it is possible to ease the development cost, since many of the modules needed for such applications are already present.

Considering the initial aim of creating a Contact Tracing application that would offer increased privacy by relying on Hyperledger Sawtooth, the goal was achieved. For reference, initially the objectives were:

O1 Survey the state of the art regarding *Distributed Ledgers*;

O2 Survey the state of the art regarding Contact Tracing and why so many of these applications were a letdown;

O3 Analyses and design a platform for contact tracing that leverages *Distributed Ledgers* / Blockchain to answer potential concerns.

"Survey the state of the art regarding *Distributed Ledgers*". We started with the basic concepts of what is a distributed ledger, followed by an introduction to blockchain and the historical connection is has with Bitcoin. Subsequently, we looked at *Bitcoin* and how it compares to more modern implementations like *Ethereum*, for this, a survey was done on *Ethereum*, and why *Smart Contracts* are a game changer. An in the end, we study why *Bitcoin* and *Ethereum* are not the best when it comes to high amount of transactions. And why *Hyperledger* is the clear winner for this type of operation.

"Survey the state of the art regarding Contact Tracing and why so many of these applications were a letdown". We started by analysing what applications exist and how they differ from each other. For that, we picked distinct applications, with distinct implementations, some centralized, some decentralized, some in between. And tried to understand why the great majority were having sub par results.

"Analyses and design a platform for contact tracing that leverages *Distributed Ledgers* / Blockchain to answer potential concerns". By joining the knowledge gained during the two previous objectives, we could develop an application that leverages *Hyperledger Sawtooth* in order to give a response to the most vocal concerns., it bets on transparency to achieve this result.

Despite all the challenges, there is always more work that can be done. After looking into applications like the one used by New Zealand (NZ Covid Tracer), is interesting to see the way they initial used contact tracing. By relying on QR-codes to show the geographic location where a user was present.

- It must be possible to register a place / geographic location;
- It must be possible to generate a QR-code based on than ID;
- Information for that place needs to be stored in an of-chain database;
- The application needs to support reading QR-codes;
- The ledger data model should change from one address for each user to multiple addresses per user.

Being the main challenge, the interaction between saved locations and their information outside the ledger.

Another interesting feature would be providing a platform for data analysis. Were researchers or event users could gather some metrics to beater understand the pandemic. Examples of this would be:

- Total number of infections.
- Average amount of contacts each infected user is in contact.
- What were the peaks of infection.
- Joining the previous feature with this, and measure place by infection rate.

It is important to mention the 2 publications I had the pleasure of being part of. Both were done in partnership with Marco Oliveira and under the coordination of Catarina Isabel Ferreira Viveiros Tavares and Marisa da Silva Maximiano. The first one was entitled "Building trust with a contact tracing application: a blockchain approach"and the second one "Immunity Passport Ledger". Fortunately, both were submitted to the World Congress on Information and Communication Technologies (WICT 2020) and promptly accepted. Leading to being invited as a speaker for the IEEE@Home Blockchain Series 2020. Where I had the pleasure of doing a presentation about “Building trust with a contact tracing application” for ruffly 20 minutes.

Furthermore, all the code developed is available under GitHub projects that provide the code under an open-source license [92] [93]

In the end, the development of a contact tracing application allowed for a better understanding of the workflow involved in contact tracing and the challenges related to a distributed solution.

BIBLIOGRAPHY

[1] Eugene Y Chan e Najam U Saqib. «Privacy concerns can explain unwillingness to download and use contact tracing apps when COVID-19 concerns are high». en. *Comput. Human Behav.* 119 (jun. de 2021), p. 106718.

[2] CDC. *What You Need to Know about Variants*. https://www.cdc.gov/coronavirus/2019-ncov/variants/variant.html. Accessed: 2021-10-9. Set. de 2021.

[3] Tyler Shelby et al. «Lessons Learned From COVID-19 Contact Tracing During a Public Health Emergency: A Prospective Implementation Study». en. *Front Public Health* 9 (ago. de 2021), p. 721952.

[4] Jane Bambauer e Brian Ray. *Covid- 19 apps are terrible — they don ' t have to be*. https://s3.documentcloud.org/documents/20424830/bambauer-and-ray-final-2.pdf. Accessed: 2021-9-26. Nov. de 2020.

[5] Satoshi Nakamoto e Www bitcoin.org. «Bitcoin: A Peer-to-Peer Electronic Cash System» ().

[6] *Ethereum Whitepaper*. https://ethereum.org/en/whitepaper/. Accessed: 2021-10-9.

[7] About Hyperledger. *Purpose of this paper*. https://www.hyperledger.org/wp-content/uploads/2021/11/HL_Paper_HyperledgerOverview_102721.pdf. Accessed: 2021-11-28.

[8] Sven Braden. «CLI_Factsheet_Immutability_of_Blockchain_explained_ITMO_example.pdf» (abr. de 2018).

[9] Silva John. *879.pdf on Egnyte*. https://sansorg.egnyte.com/dl/uYi8qKXs3X. Accessed: 2021-11-28.

[10] Wikipedia contributors. *Hash function*. https://en.wikipedia.org/w/index.php?title=Hash_function&oldid=1054996574. Accessed: NA-NA-NA. Nov. de 2021.

[11] Ralph C. Merkle. «merkle.pdf» ().

[12] Robert Shostak, Marshall Pease e Leslie Lamport. «The byzantine generals problem». *ACM Trans. Program. Lang. Syst.* 4.3 (1982), pp. 382–401.

[13] Cynthia Dwork e Noni Naor. «Pricing via Processing, Or, Combatting Junk Mail, Advances in Cryptology» (1982).

[14] «A survey on the security of blockchain systems». *Future Gener. Comput. Syst.* 107 (jun. de 2020), pp. 841–853.

[15] Dominique Guegan. *Public Blockchain versus Private blockhain*. https://halshs.archives-ouvertes.fr/halshs-01524440/document. Accessed: 2021-11-28.

[16] Kirill Kuvshinov et al. *Disciplina: Blockchain for Education*. https://www.disciplina.io/yellowpaper.pdf. Accessed: 2021-9-26. Jul. de 2018.

[17] «Permissioned blockchain frameworks in the industry: A comparison». *ICT Express* 7.2 (jun. de 2021), pp. 229–233.

[18] *Bitcoin (BTC) Preço, Gráfico, Capitalização de Mercado*. pt. https://coinmarketcap.com/pt-br/currencies/bitcoin/. Accessed: 2022-1-23.

[19] W. Dai. «b-money» ().

[20] Vivek Vishnumurthy, Sangeeth Ch e Emin Sirer. «KARMA : A Secure Economic Framework for Peer-To-Peer Resource Sharing» (jun. de 2003).

[21] *Reusable Proofs of Work*. https://nakamotoinstitute.org/finney/rpow/index.html. Accessed: 2021-11-28.

[22] *Bit Gold*. https://nakamotoinstitute.org/bit-gold/. Accessed: 2021-11-28.

[23] Matthew Beedham. *All you need to know about Bitcoin network nodes*. https://thenextweb.com/news/bitcoin-blockchain-nodes-network. Accessed: 2021-11-24.

[24] *Mastering Bitcoin*. O'Reilly Media, Inc.

[25] *Ethereum brand assets*. en. https://ethereum.org/en/assets/. Accessed: 2022-1-23.

[26] *What is Ethereum: A beginners guide to ETH cryptocurrency*. https://cointelegraph.com/ethereum-for-beginners/what-is-ethereum-a-beginners-guide-to-eth-cryptocurrency. Accessed: 2021-9-26.

[27] *ethereumbook/07smart-contracts-solidity.asciidoc at develop · ethereumbook/ethereumbook · GitHub*. https://github.com/ethereumbook/ethereumbook/blob/develop/07smart-contracts-solidity.asciidoc.

[28] *Solidity — Solidity 0.8.10 documentation*. https://docs.soliditylang.org/en/v0.8.10/. Accessed: 2021-11-28.

[29] Linda Xie. *A beginner's guide to Ethereum tokens*. https://blog.coinbase.com/a-beginners-guide-to-ethereum-tokens-fbd5611fe30b. Accessed: 2021-11-24. Mai. de 2017.

[30] Hussey Matt. *What Are ERC-20 Tokens? | The Beginner's Guide - Decrypt*. https://decrypt.co/resources/erc20. Nov. de 2020.

[31] *ERC-20 Token Standard*. https://ethereum.org/en/developers/docs/standards/tokens/erc-20/. Accessed: 2021-11-25.

[32] Ameer Rosic. *What is An Ethereum Token: The Ultimate Beginner's Guide*. en. https://blockgeeks.com/guides/Ethereum-token/. Accessed: 2021-11-25. Jun. de 2017.

[33] *CryptoKitties_WhitePapurr_V2_en.pdf*. https://drive.google.com/file/d/1soo-eAaJHzhw_XhFGMJp3VNcQoM43byS/view. Accessed: 2021-11-28.

[34] CryptoKitties. *CryptoKitties*. https://www.cryptokitties.co/. Accessed: 2022-1-23.

[35] Hemang Bhanushali et al. «Digital Certificates Using Blockchain: An Overview». Abr. de 2019.

[36] *About Sawtooth Events — Sawtooth latest documentation*. https://sawtooth.hyperledger.org/docs/core/nightly/1-2/app_developers_guide/about_events.html. Accessed: 2021-9-27.

[37] *Hyperledger Fabric*. zh. https://www.hyperledger.org/use/fabric. Accessed: 2022-1-23. Ago. de 2017.

[38] Sajana P. e Sindhu M. *On Blockchain Applications: Hyperledger Fabric And Ethereum*. https://acadpubl.eu/jsi/2018-118-18/articles/18c/84.pdf. Accessed: 2021-11-25.

[39] *Introduction — hyperledger-fabricdocs main documentation*. https://hyperledger-fabric.readthedocs.io/en/latest/blockchain.html. Accessed: 2021-11-25.

[40] «COVID19-contact-tracer-508.pdf» (jul. de 2020).

[41] *Infection prevention and control: Contact tracing*. https://www.who.int/news-room/q-a-detail/contact-tracing. Accessed: 2021-9-26. Mai. de 2017.

[42] Dyani Lewis. *Why many countries failed at COVID contact-tracing — but some got it right*. https://www.nature.com/articles/d41586-020-03518-4. Accessed: 2021-10-5. Dez. de 2020.

[43] Matthew Abueg et al. «Modeling the combined effect of digital exposure notification and non-pharmaceutical interventions on the COVID-19 epidemic in Washington state». Set. de 2020.

[44] Shamir Mukhi et al. «An innovative mobile data collection technology for public health in a field setting» (set. de 2018).

[45] Vasco Dias INESC TEC e ISPUP ... «AIPD_STAYAWAY_v2.0_09_2020.pdf» (ago. de 2020).

[46] *stayaway-app: Official repository for the STAYAWAY COVID mobile application.*

[47] *documents: Decentralized Privacy-Preserving Proximity Tracing – Documents.* en.

[48] *Decentralized_Privacy-Preserving_Proximity_Tracing.pdf.* Mai. de 2020.

[49] Aaqib Bashir Dar et al. *Applicability of mobile contact tracing in fighting pandemic (COVID-19): Issues, challenges and solutions.* Nov. de 2020.

[50] MANSO MIGUEL. *60% já apagaram a StayAway Covid: são 1,8 milhões de portugueses | Saúde | PÚBLICO.* https://www.publico.pt/2021/01/15/tecnologia/noticia/60-ja-apagaram-stayaway-covid-sao-18-milhoes-portugueses-1946366.

[51] *60% já apagaram a StayAway Covid: são 1,8 milhões de portugueses | Saúde | PÚBLICO.* https://www.publico.pt/2021/01/15/tecnologia/noticia/60-ja-apagaram-stayaway-covid-sao-18-milhoes-portugueses-1946366. Accessed: 2021-10-5. Jan. de 2021.

[52] «Proposta de Lei n.º 62/XIV» (out. de 2020).

[53] Sam Shead. *Apple accused of breaching European privacy law by French start-up group.* https://www.cnbc.com/2021/03/09/apple-accused-of-breaching-eu-privacy-law-by-french-start-up-group.html. Accessed: 2021-10-9.

[54] Adam Satariano. «Google Is Fined $57 Million Under Europe's Data Privacy Law». *The New York Times* (jan. de 2019).

[55] *COVID Tracker App: Data Protection Information Notice (DPIN).* https://www2.hse.ie/services/covid-tracker-app/data-protection-information-notice.html. Accessed: 2021-11-24.

[56] Colm Gorey. *Watch: The inside story of developing Ireland's Covid Tracker app.* en. https://www.siliconrepublic.com/future-human/ireland-covid-tracker-app-nearform-future-human. Accessed: 2021-11-24. Nov. de 2020.

[57] *TousAntiCovid*. en. https://bonjour.tousanticovid.gouv.fr/en/. Accessed: 2022-1-23.

[58] Claude Castelluccia et al. *ROBERT: ROBust and privacy-presERving proximity tracing*. https://hal.inria.fr/hal-02611265/document. Accessed: 2021-11-24.

[59] Julie M. e Manon C. *TousAntiCovid : les nouveaux critères de détection des "cas contact"*. https://www.sortiraparis.com/actualites/coronavirus/articles/231388-tousanticovid-les-nouveaux-criteres-de-detection-des-cas-contact. Accessed: 2021-11-24. Nov. de 2020.

[60] *NZ COVID Tracer QR codes | Ministry of Health NZ*. https://www.health.govt.nz/our-work/diseases-and-conditions/covid-19-novel-coronavirus/covid-19-resources-and-tools/nz-covid-tracer-app/nz-covid-tracer-qr-codes.

[61] *NZ COVID Tracer QR codes*. https://www.health.govt.nz/our-work/diseases-and-conditions/covid-19-novel-coronavirus/covid-19-resources-and-tools/nz-covid-tracer-app/nz-covid-tracer-qr-codes. Accessed: 2021-11-24.

[62] *Getting started with Business Connect*. https://businessconnect.govt.nz/getting-started/. Accessed: 2021-11-24.

[63] *TraceTogether*. https://www.tracetogether.gov.sg/common/privacystatement/. Accessed: 2021-11-30.

[64] Jason Bay et al. *BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders*. https://bluetrace.io/static/bluetrace_whitepaper-938063656596c104632def383eb33b3c.pdf. Accessed: 2021-11-24.

[65] *Singapore's Updated TraceTogether Privacy Policy Could Erode Public Trust*. https://www.csis.org/blogs/new-perspectives-asia/singapores-updated-tracetogether-privacy-policy-could-erode-public-trust. Accessed: 2021-11-24.

[66] *Singapore police can access COVID-19 contact tracing data for criminal investigations | ZDNet*. https://www.zdnet.com/article/singapore-police-can-access-covid-19-contact-tracing-data-for-criminal-investigations/.

[67] *More than 4.2m people using TraceTogether, token distribution to resume soon: Lawrence Wong, Politics News & Top Stories - The Straits Times*. https://www.straitstimes.com/singapore/politics/parliament-more-than-42m-people-using-tracetogether-token-distribution-to-resume.

[68] *TraceTogether Adoption Surpasses 70% - CCs to Re-open For Token Collection Progressively*. https://www.smartnation.gov.sg/media-hub/press-releases/tracetogether-adoption-surpasses-70. Accessed: 2021-11-24.

[69] Hao Xu et al. «BeepTrace: Blockchain-enabled Privacy-preserving Contact Tracing for COVID-19 Pandemic and Beyond» (mai. de 2020). arXiv: 2005.10103 [cs.DC].

[70] Glasgow-University. *University news*. https://www.gla.ac.uk/news/headline_752925_en.html. Accessed: 2021-9-26. Set. de 2020.

[71] Tham Yuen-C. *More than 4.2m people using TraceTogether, token distribution to resume soon: Lawrence Wong, Politics News & Top Stories - The Straits Times*. https://www.straitstimes.com/singapore/politics/parliament-more-than-42m-people-using-tracetogether-token-distribution-to-resume. Accessed: 2021-10-5. Jan. de 2021.

[72] Vikash Kumar Das. *Role of Directed Acyclic Graphs in the Blockchain Landscape*. en. https://www.blockchain-council.org/blockchain/role-of-directed-acyclic-graphs-in-the-blockchain-landscape/. Accessed: 2021-9-27. Set. de 2020.

[73] Wenhui Yang et al. «LDV: A Lightweight DAG-Based Blockchain for Vehicular Social Networks». *IEEE Trans. Veh. Technol.* 69.6 (jun. de 2020), pp. 5749–5759.

[74] Vibha Nehra, Ajay K Sharma e Rajiv K Tripathi. «Blockchain Implementation for Internet of Things Applications». unknown, jan. de 2020, pp. 113–132.

[75] Google / Apple. «Contact Tracing Cryptography Specification » (abr. de 2020).

[76] *Why many countries failed at COVID contact-tracing — but some got it right*. https://www.nature.com/articles/d41586-020-03518-4.

[77] Luca Ferretti et al. «Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing». *Science* 368.6491 (mai. de 2020).

[78] Sato Mia. *Singapore's police now have access to contact tracing data | MIT Technology Review*. https://www.technologyreview.com/2021/01/05/1015734/singapore-contact-tracing-police-data-covid/. Jan. de 2021.

[79] Dixys L Hernández-Rojas et al. «Design and Practical Evaluation of a Family of Lightweight Protocols for Heterogeneous Sensing through BLE Beacons in IoT Telemetry Applications». en. *Sensors* 18.1 (dez. de 2017).

[80] *spec: AltBeacon Technical Specification*. Accessed: 2021-8-20.

[81] *SQLite Home Page*. https://www.sqlite.org/index.html. Accessed: 2021-11-22.

[82] *About SQLite*. https://www.sqlite.org/about.html. Accessed: 2021-11-22.

[83] Toshendra Kumar Sharma. *Hyperledger Sawtooth Or Hyperledger Fabric : Which Is better?* en. https://www.blockchain-council.org/hyperledger/hyperledger-sawtooth-or-hyperledger-fabric-which-is-better/. Accessed: 2021-11-24. Mai. de 2019.

[84] Dan Brown. *Review of Five popular Hyperledger DLTs- Fabric, Besu, Sawtooth, Iroha and Indy*. en. https://training.linuxfoundation.org/blog/review-of-five-popular-hyperledger-dlts-fabric-besu-sawtooth-iroha-and-indy/. Accessed: 2021-11-24. Fev. de 2021.

[85] Hyperledger. *Data Model Considerations in Hyperledger Sawtooth*. en. https://www.hyperledger.org/blog/2018/01/16/data-model-considerations-in-hyperledger-sawtooth. Accessed: 2021-11-24. Jan. de 2018.

[86] Vincenzo Gallicchio. «Enhancing the Security of Blockchains by Using Trusted Execution Environments» (2020).

[87] *Docker Engine overview*. https://docs.docker.com/engine/. Accessed: 2021-11-28. Nov. de 2021.

[88] Sergey Grybniak. *What You Need To Know About pBFT Consensus*. https://dzone.com/articles/what-you-need-to-know-about-pbft-consensus-plus-ap. Accessed: 2021-11-25. Ago. de 2019.

[89] *CBOR*. https://cbor.io/. Accessed: 2021-11-24.

[90] *Get started*. https://zeromq.org/get-started/. Accessed: 2021-11-28.

[91] *Language Guide*. https://developers.google.com/protocol-buffers/docs/overview. Accessed: 2021-11-28.

[92] Tomás Honório. *GitHub - SawtoothNode*. https://github.com/Thonorio/SawtoothNode. Accessed: 2021-11-30.

[93] Tomás Honório. *GitHub - ContactTracingSawtooth*. https://github.com/Thonorio/ContactTracingSawtooth. Accessed: 2021-11-30.

APPENDIX

A

APPENDIX A

Nome	Country	Number of Users	Penetration	Technology	Centralized / Decentralized	Voluntary [1]	Data used only within the app [2]	Data destruction [3]
StayAway Covid	Portugal	1.152.000 (January 2021)	11,20%	Bluetooth, Google/Apple	Decentralized*	Y	Y	Y
Covid Tracker	Ireland	1.300.000 (November 2020)	26,50%	Bluetooth, Google/Apple	Decentralized*	Y	Y	Y
TousAntiCovid	France	5.500.000 (October 2020)	7,50%	Bluetooth	Centralized	Y	Y	Y
NZ Covid Tracer	New Zealand	N/A	N/A	Bluetooth, *QR-Code*	Centralized	Y	Y	Y
TraceTogether	Singapore	3.800.000 (December 2020)	70% +	Bluetooth, BlueTrace	Centralized	N	N	Y
BeepTrace	N/A	N/A	N/A	Bluetooth, GPS	Decentralized	N/A	Y	Y

Table 1: Contact Tracing application compared

* Rely on a centralized server to store information of the infected users

DECLARATION

I declare, under commitment of honor, that the work presented in this project report, with the title *"Exposure Tracker Ledger"*, is original and was performed by Tomás Honório Oliveira (2190338) under the supervision of Professor Catarina I. Reis (catarina.reis@ipleiria.pt) and Professor Marisa Maximiano (marisa.maximiano@ipleiria.pt).

Leiria, November of 2021

Tomás Honório Oliveira