VRIJE
UNIVERSITEIT
BRUSSEL

Master thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in de Toegepaste Informatica

ANONYMOUS CONTACT TRACING FUSION

Yassine Bouagaz

Academic Year 2020-2021

Promotor: Prof. Dr. Beat Signer
Supervisor: Maxim Van de Wynckel
Science and Bio-Engineering Sciences

VRIJE
UNIVERSITEIT
BRUSSEL

Proefschrift ingediend met het oog op het behalen van de graad van Master of Science in de Toegepaste Informatica

ANONYMOUS CONTACT TRACING FUSION

Yassine Bouagaz

Acadamiejaar 2020-2021

Promotor: Prof. Dr. Beat Signer
Supervisor: Maxim Van de Wynckel
Wetenschappen en Bio-Ingenieurswetenschappen

Acknowledgements

Foremost, I would like to express my sincere gratitude to my promoter Prof.Dr. Beat Signer for giving me the opportunity to work on this thesis. I would like to thank my supervisor Maxim Van de Wynckel, for his time and effort into helping me throughout this thesis. He was always available to help and guide me into the right direction.

I would also like to thank all the professors and staff, that despite the measures taken for the COVID-19 managed to share their knowledge and keeping everything organised.

Last but not least, a big thank you to my family who gave me the chance to study, supported and motivated me during all these years. Thank you!

Abstract

The emergence of the COVID-19 pandemic has not only an impact on the medical world, but also on technology. The increasing amount of infections and the fact that not everyone is vaccinated yet triggered researchers to find a way to control the number of contaminations. Contact tracing was one of the main methods to keep these numbers in control, at the start of the pandemic there was no real coordination between all the contact tracing solutions. Coordination can extend the reach of getting access to more contact trace information from registered positive tested individuals, and as a result increase the probability to receive an alert. In this thesis we will investigate different contact tracing techniques that are in use and learn about methods to preserve data in a knowledge base. Contact tracing information can be considered sensitive information, so it is important to protect the privacy of all people using contact tracing solutions. In our proposed solution, we will design a prototype that will merge the contact trace information of positive tested users gathered out of different types of contact tracing techniques together in one database. The prototype will support proximity-based and location-based techniques. We will investigates ways to anonymise data to design a solution in order to preserve the anonymity. We will retain only the strict minimum of information needed to be able to contact trace. In the evaluation we will evaluate the effectiveness of the fusion of different type of contact tracing methods with the aid of a simulation.

Samenvatting

Het uitbreken van de COVID-19 pandemie heeft niet alleen een impact gehad op de medische wereld, maar ook op de technologische. De toename van de infecties en het feit dat de meerderheid nog niet gevaccineerd is, heeft ondezoekers ertoe aangezet een oplossing te zoeken om de aantal besmettingen onder controle te houden. Contact tracing is een effectieve manier om deze aantallen laag te houden, maar er is een gebrek aan coordinatie tussen de verschillende contact tracing oplossingen. Coordinatie zou de verschillende contact tracing applicaties toegang geven tot meer contact trace data en hierdoor de kansen vergroten op een alert. In dit proefschrift onderzoeken we een aantal manieren om aan contact tracing te doen. We leren methodes om informatie bij te houden in een kennisbank. Contact tracing data is heel gevoelige informatie, dus is het heel belangrijk om de privacy van contact tracing applicatie gebruikers te beschermen. We zullen voor de oplossing een prototype ontwerpen, deze zal contact tracing informatie van positief geteste gebruikers samenvoegen in een databank. De prototype zal enkel tokengebaseerde en locatiegebaseerde applicaties toelaten. Om de anonimiteit van de gebruikers te verzekeren zullen we enkel het strict minimum aan informatie bijhouden die nodig is om aan contact tracing te doen. In ons evaluatie zullen we aan de hand van een simulatie de doeltreffendheid van verschillende contact tracing methodes evalueren.

Contents

Chapter 1

Introduction

The Coronavirus pandemic of 2019 (COVID-19) has created many challenges in the research world. Given the circumstances that at the time of writing not everyone is vaccinated yet, it is important to keep the number of contaminations in the world as low as possible. Prevention of further contaminations is crucial in order to make this possible. One method to do so is by making use of contact tracing [1, 2]. This method implies that we trace back all people who have had contact with the infected person, by testing them and checking if they are infected. Contact tracing will be effective if the infected is isolated to prevent further spread [3].

In the past, researchers used this method to examine the spread among patients with genital chlamydia infection [4]. Eichner [5] demonstrates that in a population where 20 percent is immune, we can contain a smallpox outbreak within 6 months by the use of contact tracing and case isolation. Tracing people back manually is resource intensive and given the speed in which it is spread, it is more effective to choose for digital contact tracing. The fact that we have to rely on the memory of the infected when applying manual contact tracing is not reliable, especially when the place of infection is densely populated [6]. The Ebola epidemic in West Africa [7] was an evidence that applying paper-based contact tracing was a big challenge. This was due to numerous problems: "*incomplete identification of contacts, delays in communication, loss of contact lists, inadequate data collection and transcription errors*" [8]. To improve and counteract these weaknesses, Guinea introduced mobile health tools to support contact tracing and surveillance. The report conducted by Sacks [9] concluded that training, investment and oversight will reinforce indirectly the health system to prevent outbreaks.

CNN quoted in November 2020 *"Several countries have introduced tracking app technology to monitor citizens' movements and potential exposure to COVID-19, including Australia, Japan and Singapore, but there is no coordination between the systems and they have had varying degrees of success."* [10] This claims that there is no coordination between the different contact tracing tools. This is a problem that we encounter today with COVID-19. It means that every application will have their own database and in many cases a different way of working. That gives a considerable complexity tracing back people who use different mobile applications. During this thesis we will do research to find a way to collect data coming from infected persons using Bluetooth-based, QR-based and location-based contact tracing applications. By giving the possibility to push infected cases through an Application Programming Interface (API) in a database. This database will store the data that may be sensitive in an anonymous way so that it will be difficult to track back the person behind the data. By creating an anonymous database there will be a possibility to connect several applications. This will increase probability to have a match when checking if a user was in close contact of an infected person, even if the user uses a different mobile application.

1.1 Contribution

The lack of coordination between the contact tracing applications affected us to find a potential solution to this problem. At the moment of writing researchers did some progress by improving the interoperability between some of the decentralised BLE-applications used in Europe[1]. Our contribution consists of investigate the different types of contact tracing solutions to gain more insight to design a prototype that will not only centralise data of positive tested persons using token-based applications but also location-based applications. Learn about anonymisation to be aware of the dangers of publishing information and information knowledge bases, this will influence the way we design our prototype.

1.2 Thesis Structure

This thesis is structured in the following way:

- In Chapter 2 we will go through different methods of contact tracing tools more specific outbreak response tools, symptom tracking tools and proximity & location-based tracing tools. We will learn about information knowledge bases and anonymsation techniques.
- In Chapter 3 we will propose a potential solution towards the identified problem presented in the problem statement, which defines a lack of coordination between different contact tracing applications. We will set up the requirements and shape our database based on the gained knowledge during our research. This chapter will also contain the implementation of the proposed solution in the form of a prototype.
- In Chapter 4 we will evaluate our prototype by conducting a simulation where we will simulate each contact tracing method separately. Then we will merge the positive cases into the prototype and compare the results.
- Chapter 5 will be our final chapter, where we will discuss about future improvements and we will end with a conclusion of our thesis.

[1] https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/travel-during-coronavirus-pandemic/mobile-contact-tracing-apps-eu-member-states$_en$

Chapter 2

Background & Related Work

The goal of this chapter is to analyse and discuss contact tracing, learn more about information knowledge bases and anonymisation. First we will zoom into the different contact tracing techniques. Next, we discuss the possibilities of using an information knowledge base. Finally, we will analyse ways to anonymise data that may identify users of contact tracing applications.

2.1 Contact Tracing Tools

COVID-19 triggered many researchers to dive back into solutions regarding automating of contact tracing. In cases like this digitising contract tracing applications can facilitate the process [11]. According to the World Health Organization (WHO) we can classify the digital contact tracing applications into different categories [12]. These categories will be discussed in Section 2.1.1, Section 2.1.2 and Section 2.1.3.

2.1.1 Outbreak Response Tools

This category of tools manage the cases by keeping the cases of infection and the contact information in a digital database. A paper written by a Greece university explains a project they made. This project had as purpose to help public health authorities. They created an expendable online platform *Safe in COVID-19* that consists of three roles: (I) public authorities will use a web application that will help to have an overview of the situation, thereby making decisions easier.
(II) Then we have the healthcare professionals that will also use an web application for support of the connected patients. (III) The citizens and their families will use a mobile application, important for recording data that is related to COVID-19 [13].

Go.Data[1] is the result of several years of experience in outbreaks. This tool is created to relieve the workload of the outbreak response teams, by proposing a tool that can handle large amounts of data. According to Go.Data, the tool should ease the process of tracing back potential contacts, make it possible to visualise contagion growth and should provide flexibility for several types of outbreaks.

[1] `https://www.who.int/godata`

2.1.2 Symptom Tracking Tools

These tools gather information that users provide, in terms of signs and symptoms. These are key because it will give the user more clarity about a possible infection for a disease based on the input. To get more out of these type of tools it is interesting to integrate them into the process of contact tracing. This will accelerate the process between detection of the infection and alerting the persons who were in close contact. But users still need to pay attention when using symptoms tracking tools. They can not assure avoiding misdiagnosis, so it is not recommended to fully rely on the tool. That is why it is important that most of these tool integrate a follow-up after a possible contamination is detected [12].

Georgetown University in the United States worked on a monitoring and reporting system for COVID-19, this system is designed by a team consisting of people from different disciplines. Figure 2.2 shows how the system is working. In the first step the institution/agency grants access to the system. By granting access, the system assigns an *entity ID* and provides a *report authorization code*. After this step the individual can be monitored with the unique assigned ID which can not be linked to the individual, this to maintain the privacy of the individual. Then in the Symptom Tracker System, the individual will be able to check if the symptoms may be a risk factor for COVID-19. These were selected by consulting clinical experts. When this is done, a report will be generated by the agency/institution using the the authorization code This report will clarify if the person can be taken off isolation or if he needs further assessment based on the symptoms [15].

In China the Fujian University of Technology worked on a Quick Response (QR) code-based contact tracing framework, where each individual obtains a QR code that can be seen as an electronic certificate of the person's health status. The QR code can display two different colours (Figure 2.1). If the code colours green the person is safe, but when the colour is orange the person may represent a risk. The QR codes are read and analysed by scanners. These scanners are placed at entrances of crowded places where large amount of people are expected. The outcome gives the possibility to enter or refuse entry into a building [14].

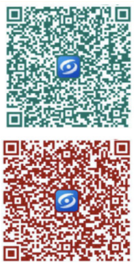


Figure 2.1: Color Codes QR-Framework [14]

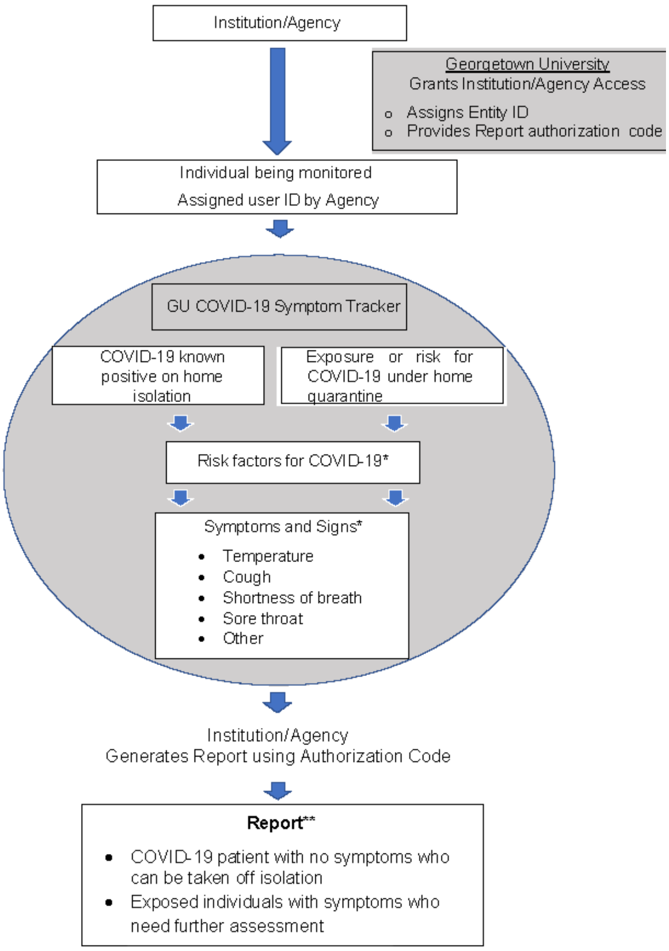

Institution/Agency
Georgetown University
Grants Institution/Agency Access
Assigns Entity ID
Provides Report authorization code
Individual being monitored
Assigned user ID by Agency
GU COVID-19 Symptom Tracker
COVID-19 known positive on home isolation
Exposure or risk for COVID-19 under home quarantine
Risk factors for COVID-19*
Symptoms and Signs*
Temperature
Cough
Shortness of breath
Sore throat
Other
Institution/Agency
Generates Report using Authorization Code
Report**
COVID-19 patient with no symptoms who can be taken off isolation
Exposed individuals with symptoms who need further assessment

Figure 2.2: Symptom Tracking System [15]

2.1.3 Proximity & Location-based Tracing Tools

In this section we will discuss app-based contact tracing techniques. Nowadays it is difficult to imagine someone without a smartphone. By using this advantage we could simplify the task of tracing back people, who might be in the area of an infected person. Especially if the spaces where it can be used are densely populated, think for example cities. Most common methods in the market are using Bluetooth Low Energy (BLE), we will discuss them because there are several protocols using this technology. Also QR Code can be used for contact tracing. It can both be used for symptom tracking as for location tracking. The last type we will discuss is the Location-based. All these different methods have the same purpose to memorize and report if there are positive cases.

Bluetooth Proximity Tracing

BLE is a way to automate contact tracing, by using the advantage of detecting devices that are in close proximity. When a device detects another device, it can share common information that can be memorised on each device. Afterwardz, this information will help to know if the two were in close contact in case if one of the two persons was infected. The key reasons why BLE is a good choice for contact tracing is because of the privacy, accuracy and feasibility that this technology provides [16].

Nowadays many contact tracing apps are in use, but TraceTogether[2] was the first app deployed by a government. This was an initiative of the Singaporean government that also open sourced their protocol Bluetrace[3]. But also the software giants Google and Apple joined forces by proposing an exposure notification system[4]. This led to concerns about the privacy within European academics to whom the Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT) project was created. This project received many critics, that led to the creation of a new team that worked on the Decentralized Privacy-Preserving Proximity Tracing(DP-3T) project [17].
We will go through some of the well-known protocols that use BLE to do contact tracing. The protocols that we will discuss can be grouped into two bigger groups namely the centralised and the decentralised applications.

Decentralised apps will store the common identifier that is generated between two devices and will do the processing of matches on the mobile device itself. When a person is infected, this person will upload identifiers that are stored on the device. It comes down to the fact that if other users download the identifiers at a later moment they will be able to see if they were in contact with an infected person. Protocols that make use of a decentralised architecture are for example Decentralized Privacy-Preserving Proximity Tracing(DP-3T) and Temporary Contact Numbers Protocol (TCN).

On the contrary, centralised apps are storing and processing the identifiers on a central server. This server is administrated by health authorities, and consists of a system that will warn the user if they were in close contact with an infected user. Figure 2.3[5] illustrates the difference between these two systems.

[2] `https://www.tracetogether.gov.sg`

[3] `https://bluetrace.io`

[4] `https://covid19.apple.com/contacttracing`

[5] `https://www.bbc.com/news/technology-52355028`

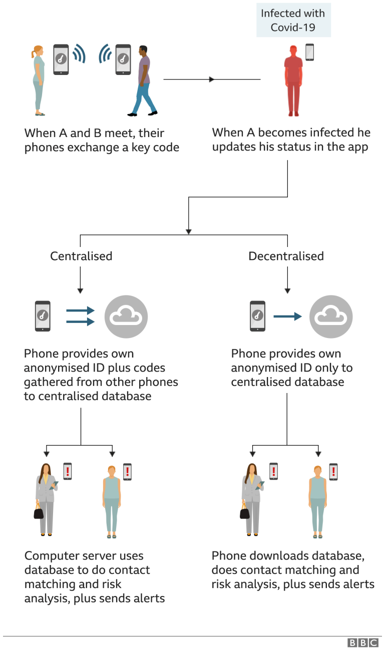

Infected with Covid-19
When A and B meet, their phones exchange a key code
When A becomes infected he updates his status in the app
Centralised
Decentralised
Phone provides own anonymised ID plus codes gathered from other phones to centralised database
Phone provides own anonymised ID only to centralised database
Computer server uses database to do contact matching and risk analysis, plus sends alerts
Phone downloads database, does contact matching and risk analysis, plus sends alerts
BBC

Figure 2.3: Centralised vs decentralised tracing apps

The Decentralized Privacy-Preserving Proximity Tracing (DP-3T) is an open protocol brought alive by two technical universities (ETH & EPFL) in Switzerland. This project has gathered a great number researchers to work on it from several universities in Europe. This protocol is known to be privacy-friendly because it only pushes the strict minimum of contact tracing information to the back end server.

A "White Paper" discusses three different implementations, we will explain the Low-cost design protocol. This protocol make use of Ephemeral IDs (EphIDs) that are generated at the beginning of each day with a secret day seed. Each day will have a new seed, created by hashing the old one. The number of generated identifiers on a day depends of the duration in minutes, that an identifier can be used. This duration is called epoch. By means of a pseudo random generator (PRG) containing a pseudo-random function with the daily seed and fixed public string, the codes are generated. The device will pick randomly a generated code and broadcast it during the configured epoch. After this period ends a new code will be broadcasted. The reason they are not using always the same identifier is for preventing that the location can be tracked by use of broadcast identifiers. Devices in the area will store following attributes: the received EphID, an exposure measurement that will help to determine the exposure duration of the device with a device from a COVID-19 positive tested user and lastly the day when the EphID was detected. Figure 2.4[6] illustrates the process when a patient is diagnosed.

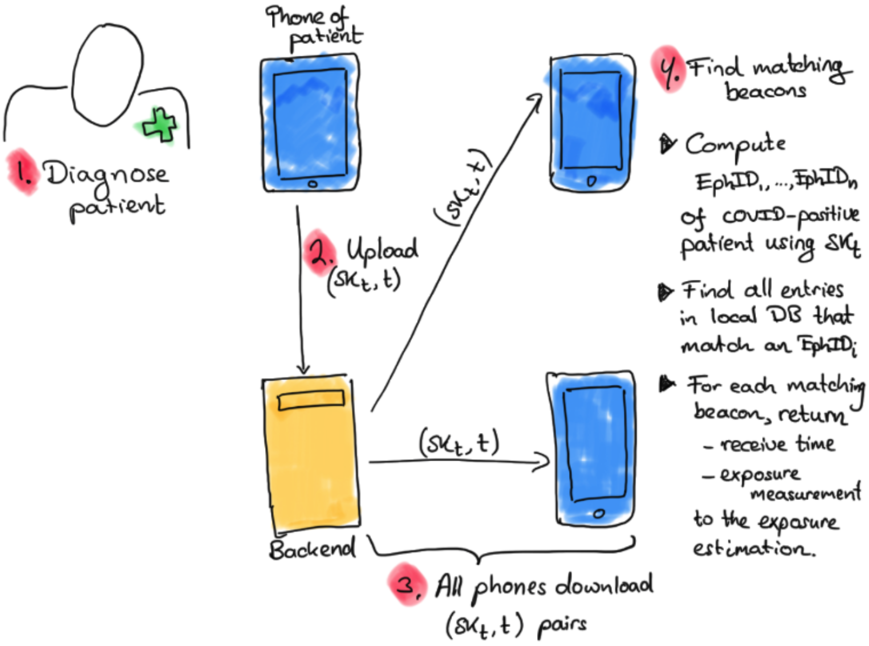

Phone of patient
1. Diagnose patient
2. Upload (SK_t, t)
(SK_t, t)
(SK_t, t)
Backend
3. All phones download (SK_t, t) pairs
4. Find matching beacons
▶ Compute $EphID_1, \ldots, EphID_n$ of COVID-positive patient using SK_t
▶ Find all entries in local DB that match an $EphID_i$
▶ For each matching beacon, return
- receive time
- exposure measurement
to the exposure estimation.

Figure 2.4: Process of DP-3T Proximity Tracing

[6] https://github.com/DP-3T/documents/blob/master/

The difference between the low cost implementation which we have just described, the unlinkable design protocol and the hybrid design protocol is that the last two provide better privacy properties. The unlinkable design hashes and store the identifier of positive tested users in a Cuckoo filter. Instead of using a list of seeds from users who tested positive. The Hybrid design will generate more often seeds and will use them within a time window to generate identifiers. Instead of using the same seed for a whole day like in the low-cost design[7]. In Belgium we use Coronalert, a Bluetooth-based application that is used in combination with manual contact research. This application is based on the DP-3T technology and the German Corona-Warn-App[8].

The Temporary Contact Numbers Protocol (TCN) is also a decentralised protocol. As the name of the protocol indicates, it will make use of temporary contact numbers that will be generated and stored. When two users with the same application come in close contact, they will exchange through Bluetooth each others' TCN number. Each device will store a TCN number and a corresponding timestamp. When someone is positive it will send the generated TCNs and timestamps to the server. People will be able to retrieve these and check if there is match, a match means that this person was in close contact with the infected person. According to the paper written by members of the IEEE, TCN was the most used protocol at that moment. Six applications were running using this protocol when the paper was published [18].

[7] `https://github.com/DP-3T/documents/blob/master/`

[8] https://www.coronawarn.app

Location Tracking/Based Tracing

Location-based tracing is another way of handling contact tracing. This method uses locations and timestamps to keep track of the users movements. In case of an infection these data will be shared centrally where other users will have the possibility to see if there is any match. The most known way to calculate the location is by use of global positioning system (GPS). Signals emitted by the GPS satellites will be received by the device. The time delay of each received signal will be interpreted as a measure to calculate distance to the satellite. These measures will help calculating the position by a technique called trilateration. The position will be returned in the form of longitude and altitude [18].
According to IndoorAtlas GPS not ideal to use in densely constructed areas, this technology does not work well there. Instead of using GPS, WiFi access points are used by transmitting signals to devices. The location of all these access points are gathered on a map, therefore it is possible to give the position of a device accurate to 50 metres. Another way to track location is by using BLE beacons, as shown on the left side a Figure 2.5 that illustrates how to track the right location of the receiver using trilateration technique. The right-hand side Figure 2.5 shows how mobile devices transmit and receive information using beacons to help calculate the distance by BLE [16].

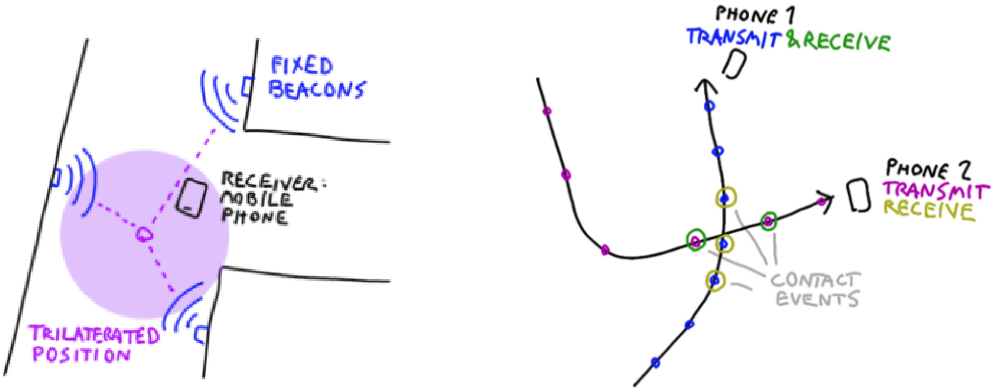


Figure 2.5: Proximity detection through iBeacons [16]

In China researchers developed a mini-program within the application WeChat[9]. The reason why it is implemented in WeChat is because it is a very popular mobile application in China. This mini-program will trace close contacts of all patients that use WeChat and gather global positioning system (GPS) locations. This will

[9] https://www.wechat.com

facilitate the tracking and will help to quarantine persons with a risk of infection as fast as possible [19].

QR Code-based Tracing

Quick Response Codes are used more and more nowadays, but it all began in Japan where it was used to track components in factories. Denso [20] was the company behind this technology back in 1994. There was a need to improve barcode scanners, so Masahiro Hara built a team to develop a two dimensional (2D) code. As a final result the QR code was invented. It consists of a position detection pattern at the three corners as shown in Figure 2.6 with the correct ratio 1:1:3:1:1 to prevent false recognition. The reason of the popularity is because of the capacity to contain more information than bar codes and the fact that this technology can be used in various ways on smartphones. It can be used to add new contacts by adding the new contact's QR, promote a new event so people only need to scan the code to have all necessary information about it or to give geographic location so people only need to scan the code to know the exact address. These were a few examples where the use of QR codes offers benefits in replacing manual work by just a scan [21].

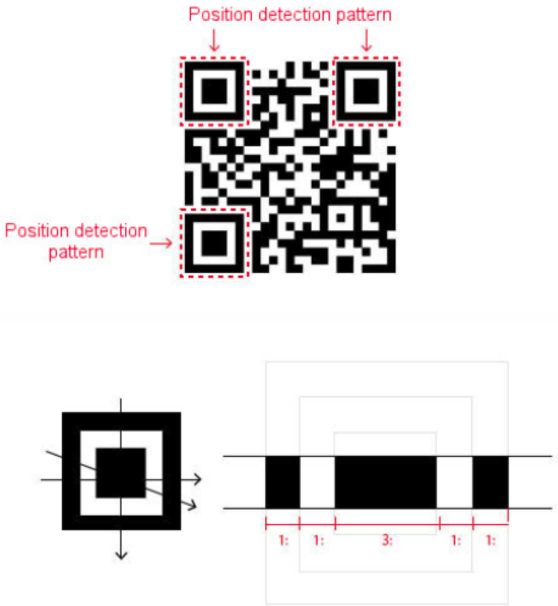


Figure 2.6: Position Detection Pattern & Ratio [20]

In this thesis we will investigate the way how this technology can help in the contact tracing process. Contact tracing will be possible by triggering an app after scanning a QR code with a mobile phone. Depending on the process the person will register either a location and a timestamp or fill in a form with the personal information. Figure 2.7 shows two QR codes used at the Vrije Universiteit Brussel (VUB) WISE lab to check in and check out. By checking in the code redirects the person to a webpage where the user can insert their contact information. This application was made using the weave.ly platform that makes it possible to create functional prototypes by drag and drop [22].

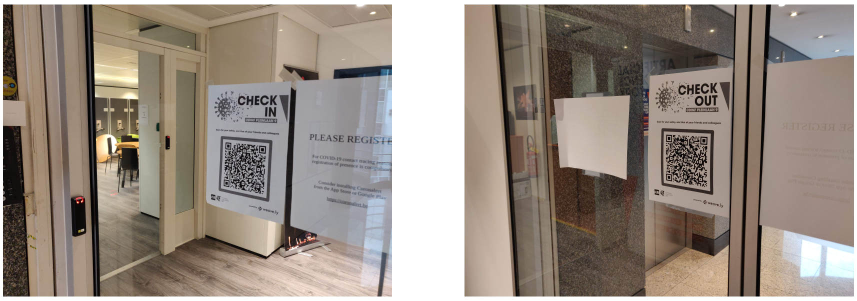


Figure 2.7: VUB check in - check out

The SafeEntry[10] app is an example of how QR code-based contact tracing was used. This application was mandatory in public places and businesses. People were required to scan when entering and when leaving a place. Besides location and time other personal information is stored centrally secured server [23].
QR codes are also used in a decentralised way, see as for example the Zwaai[11] app. This app can be used in two ways either when meeting another person using the app, or when entering a place. The app generates constantly new random numbers. When scanning someones else personal QR code, it will save each other's number and timestamp of when the scan took place. When entering a place the user will scan a code that will be unique to this place and when leaving he will do the same the same thing. The data is saved locally so privacy of the user can be preserved. In case of a positive diagnose, the information that is kept locally will be uploaded by the health authorities on the central server. So other apps will be able to check for matches in the server [24].

[10] https://www.safeentry.gov.sg
[11] https://zwaai.app

2.2 Information Knowledge Base

> *"Linked Open Data (LOD) is Linked Data which is released under an open license, which does not impede its reuse for free."* – *Tim Berners-Lee*[12]

To be able to see if there is a possibility to use an open data knowledge base, it is important to first understand what it all entails.

The concept open data is about making data available and accessible to anyone. In a way that there are no restrictions to re-use it for commercial or non-commercial purposes and to redistribute the data in other forms. Using open data can offer many value when it is well interpreted. Government open data for example is a huge source of information that can be used in different ways to improve many areas in our society. By making data public it increase the thrust of the population because there is more transparency. Moreover, but it can also bring innovation, improvements of products or services and when combined with other sources it expands the possibilities [25].

It becomes interesting when we can link this open data to external sets of data. It is also the way how the web of data evolves nowadays, by creating links between data from different sources. In order to interlink with other linked open data, data needs to meet the requirements of the highest rank in a ranking proposed by Tim Berners-Lee. Figure 2.8 that illustrates the *Five Star Linked Open Data* ranking. This ranking consists of a maximum of five stars. Each star is granted based on how the data is published. Berners-Lee's idea was to stimulate the quality of the shared data. According to the W3 and Ontotext that we consulted, these stars are universal and can be described as follow [27, 28]:

Interesting in this ranking is that each star is earned when it offers an additional benefit. One star is when your data is available in any format with an open licence, to be open data. The next star can be earned when it is machine-readable for example in an Excel binary File Format (XLS). Three stars are earned when the data is published in a non-proprietary file format for example in a comma-separated values (CSV) file. The next star when the data is structured following principles that Berner-Lee outlined. Where he put importance on using Uniform Resource Identifiers (URIs) to identify or name things. Secondly these URIs need to start with HTTP, so it can be retrieved by other users. Using open standards Resource Description Framework (RDF) for example and a query language such as

[12] https://en.wikipedia.org/wiki/Linked_data

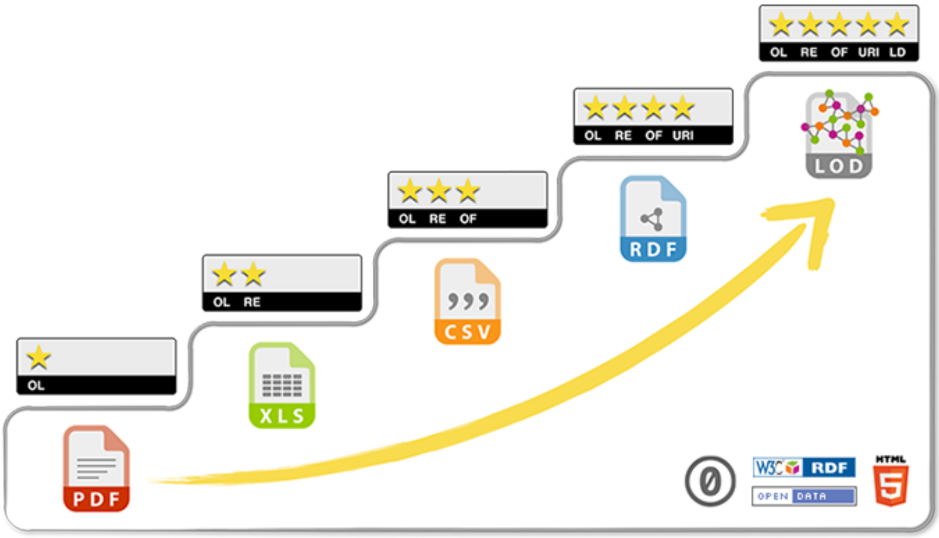


Figure 2.8: Five Star Linked Open Data [26]

Protocol and RDF Query Language (SPARQL) to be able to retrieve and manipulate data. The last star is earned when it already earned previous stars and that it offers the possibility to interlink with other open data. This will broadening of the web and opportunities to discover more interlinked data [29].

Like explained in the five star ranking, SPARQL is a query language used to manipulate open data. Data that is differently accessed than in relational databases. That is why it is stored in NoSQL databases, designed in a way to store large amount of data that is unstructured and rapidly evolving through the time in a efficient and tidy manner. Figure 2.9[13] depicts types of NoSQL databases, most common are [30]:

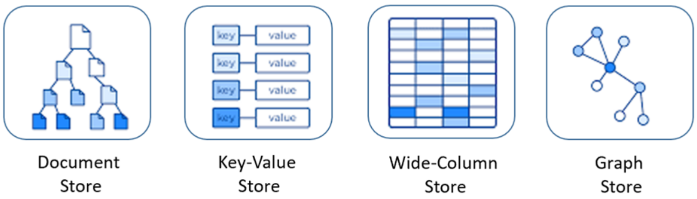


Figure 2.9: NoSQL databases types

[13] https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data

Document-oriented databases are a storage system where data is stored in the form of a JSON, XML, binary document in a collection-based manner. To speed up searches indexing is applied on field names. MongoDB[14] is a well known open source database system that support this style of database.
Key-Value databases are a way of managing data in a dictionary or hash table based way. A dictionary is made up of a collection of objects. An object is composed of fields containing data that describes the object. Redis[15] is an open source database that provides support to various types of data structures.
Column-based databases store their entries in columns instead of rows. Each column contains data of the same type, this improve performance for calculations. HBase[16] is a database supporting realtime access to large amounts of data.
Graph-based database is a type of database based on graph-theory, consisting of unique identified nodes connected by outgoing/incoming edges and a set of properties. Nodes suppose to represent entities and the edges the relation between those entities. Neo4j[17] is a platform that supports this type of database.

In this thesis we are looking for a system that is reliable and can ensure us data integrity of sensitive data. That is why it need to meet the ACID (Atomicity, Consistency, Isolation, Durability) properties. The following paper [31] compared a relational and NoSQL database,both have their advantages or disadvantages. But the results concluded that it is more convenient to opt for a relational database when ACID properties are priority. But in terms of performance it is obvious that NoSQL performs better with larger sets of data.

[14] `https://www.mongodb.com`
[15] `https://redis.io`
[16] `https://hbase.apache.org`
[17] `https://neo4j.com`

2.3 Anonymisation Techniques

Data in general can be very rich in information, especially when it can be linked, as we discussed in Section 2.2. When data is made public, it can serve for many purposes. In this thesis we mainly work with contact tracing data. Contact tracing data contains sensitive personal data that needs to be store, it can be be misused to reveal the person behind it. So, it is important to understand the concept anonymous data and to learn how to make data anonymous.

Data anonymisation consists of de-identifying data by means of anonymisation techniques to prevent that the data can be used to reveal someone's identity. The National Institute of Standards and Technology (NIST) provided a guide that gives guidelines on how to protect confidentiality of Personally Identifiable Information (PII). The guide posses recommendations that should be implemented if you are having organisations or doing businesses with U.S. Federal government agencies, but may be useful to use in general when dealing with sensitive data. Many organisations deal with large sets of data, using the manual helps distinguish and understand the important information they have on their disposal. They should be able to identify information that could be PII. This could be information that describes the person directly or information that can be linked or be linkable to that person. E.g. name, birth date, birth place, address information, contact information, biometric data can be used to identify the person directly, Figure 2.10[18] depicts an example of information which can be linked to a person. Linkable or information that can be linked are for example place of employment, education, religion, etc [32].

[18] `https://www.imperva.com/learn/data-security/personally-identifiable-information-pii/`

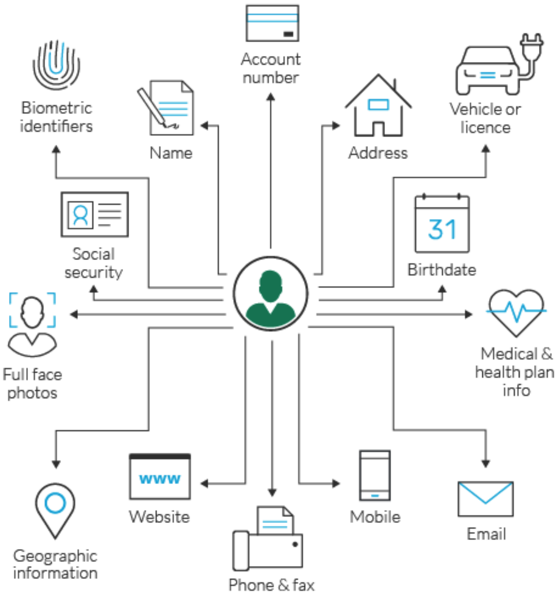
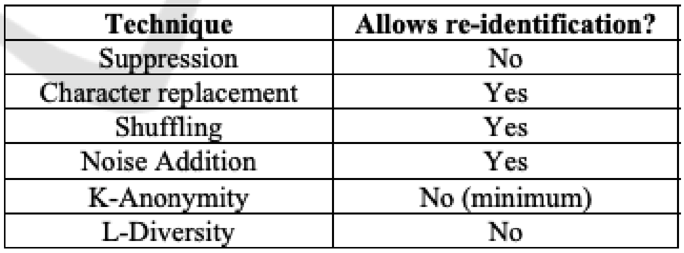


Figure 2.10: Personal Identifiable Information (PII)

Following paper [33] emphasis the importance of privacy of personal data issues. The importance to know how to erase all links to a particular individual. Before getting into the anonymisation techniques. It is important to know that there is difference between pseudonymised data and anonymised data. Pseudonymised data is data that can not deduce us to an individual without the use of external information. While with anonymised data it is impossible to find out the person behind the data even with additional information.

Techniques [34, 33]: *Suppression*, is a technique where we either remove identifiable values in a dataset or replace values by meaningless padding. By applying this technique it is impossible to find out what the removed or replaced values were before. *Character replacement*, is an approach where we cover a part of a text or attribute. Generally by using symbols, just enough to make the data anonymous. *Shuffling*, a technique where we keep the data original. But shuffle attributes that has no impact on the result. This will not guarantee anonymisation, therefore it is better to use it in combination with other techniques. *Noise Addition* [35] is a method used to increase confidentiality. We use this method on nominal data where we add or multiply the original data with a random or stochastic number. It is important to know that if it is not well applied there is major risk that data lose significance. *Generalisation*, a technique where the focus lays on making data less specific, by working with i.e. ranges instead of specific numbers or omitting house numbers in addresses.



Technique	Allows re-identification?
Suppression	No
Character replacement	Yes
Shuffling	Yes
Noise Addition	Yes
K-Anonymity	No (minimum)
L-Diversity	No

Figure 2.11: Risk of re-identification per technique [33]

Despite applying anonymisation on sensitive data, things can go wrong. In the paper by Elliot et al. [36] they describe three examples of cases where companies failed to maintain anonymity of persons after publishing data.
The first example is the one of AOL[19], an online service provider in the United States. They decided to release search histories of a big number of users in 2006. User names were replaced by numbers by using a process called pseudonymising. So, in the eyes of AOL the data was safe enough to be published. But, two problems were not taken into account. firstly, users especially when they are not famous may search for their own name. Secondly, by linking all sensitive searches made by a pseudonymised user persons could be identified.
The second example described how Netflix as part of a competition published a dataset containing data of 500 000 pseudonymised subscribers. By combining the published dataset with the of Internet Movie Database (IMDb)[20] researchers where able to find similarity, especially when they looked into the ratings of movies that were successful which is pretty unique.
Finally, the New York cabs back in 2013. They published information about the cabs daily journey. Information that could be delicate was hashed. The hash was not good enough, so by doing some reverse engineering it was possible to determine locations and destinations of celebrities. Based on the pictures taken by paparazzi, just before celebrities enters a cab. By looking up where the picture was taken or by searching the medallion number that is visible on the picture, it was possible to derive the person identity.
These examples shows us that publishing data should not be taken lightly. It is also difficult to take into account all the assets of someone with bad intentions.

[19] https://www.aol.com

[20] https://www.imdb.com

Chapter 3

Solution & Implementation

The aim of the solution is to increase the probabilities to be alerted if we may be in the area of a positively tested person. In this chapter we propose a solution for facilitating contact tracing. What we want to achieve is a system where we will be able to link different contact tracing methods. The focus in the solution will be on proximity and location-based tracing tools. The solution consists of an API that will make it possible to group all positively diagnosed users of different applications in a database. This solution is focused on applications that make use of BLE, QR codes and Geographic locations. Figure 3.1 illustrates the design of the system.

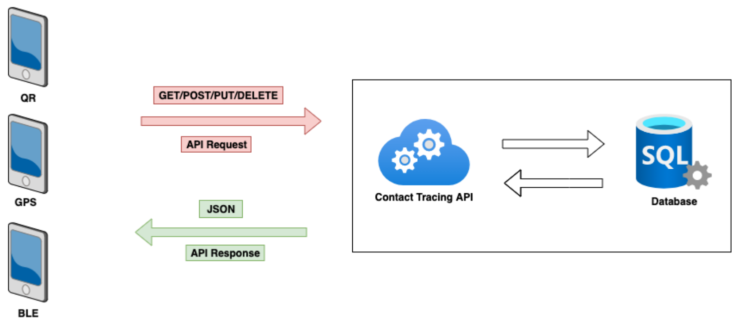


Figure 3.1: System architecture

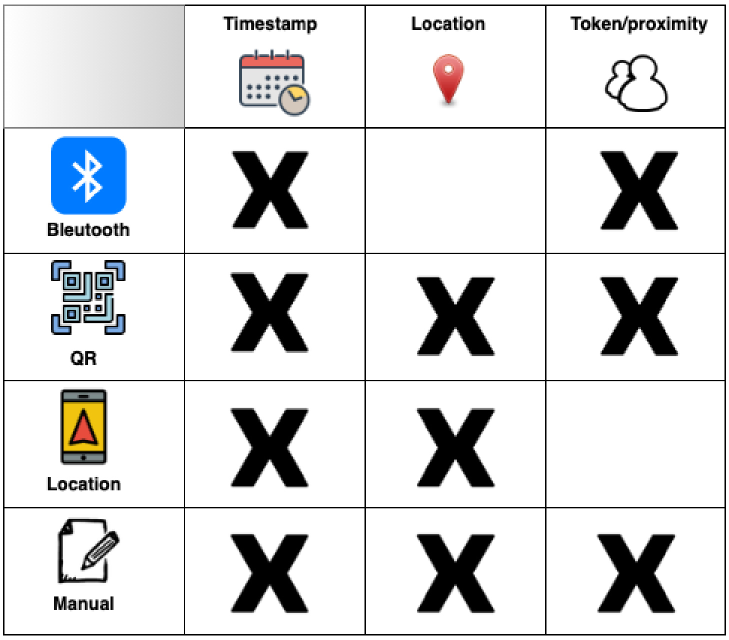
The implementation will be a centralised system where mobile applications can subscribe, send requests with contact tracing data of positive diagnosed users and send requests to check if there may be a match with the uploaded data of positive tested users. To preserve the anonymity from whom the data is stored, we will keep our database internal and not opt for an open database solution. In Section 3.1 we will mention that location data is needed for a location based tracing method, this type of the data is to sensitive to make public. Locations could then be used like discussed in Section 2.3 to reveal the person behind it and possible abuse of it.

Based on the architecture and requirements of the CT API prototype, we will explain the way we implemented the solution. We will discuss the technology used to build our prototype, the way how it works and the tools used to test it.

3.1 Requirements

In our related work we saw different types of contact tracing methods, each with their own way of working. The aim of our solution is to bring together the data of all positive tested persons in one place. Therefore, it is important to know which data needs to be uploaded to be able to retrieve potential matches with infected individuals. For the sake of the users of different applications, we will limit the collection of data needed to find matches to a strict minimum. BLE-based, QR code-based and Geographic location based are the three methods of contact tracing that will be supported by the API.
As mentioned in the introduction, our solution will be a centralised based solution. It is important to know that many solutions described in the related work are decentralised in terms architecture. This will make it difficult for the decentralised types to use this system proposed. The reason is because the data that will be uploaded will be kept internal. So, checking if a person was in the neighbourhood of an infected person will be done in the API. Decentralised applications do this check in the person's own device.

But important, in order to know on what to check we have to first focus on what data each method needs to provide us. Figure 3.2 summarises the data needed in order to get it working. The rows describes the methods and the columns the type of data that should be provided. In our related work we described three technologies used during this pandemic to do contact tracing. *Bluetooth technology* (Section 2.1.3) based solutions from which we will use the solution with an ephemeralid/token and a timestamp. The second row of the figure shows the *QR-based solution*, which will use timestamps, locations and tokens depending of how the application is working. As mentioned in related work, there were two ways of using this approach. Either when two persons meet each other they both scan their QR codes, this will be kept together with a timestamp. The other way is when entering a place by scanning when entering and when leaving, this will held the location, the time of entering and the time of leaving. The third row describes *Geolocation-based* applications that will held the time that a person was at a certain place. More precisely a starttime, endtime and a location. The fourth row is for information purpose to show the possibilities of data that can be held when doing manual contact tracing. We can track the time, location and people which were near a person.



	Timestamp	Location	Token/proximity
Bleutooth	X		X
QR	X	X	X
Location	X	X	
Manual	X	X	X

Figure 3.2: Input per method

3.1.1 Functional Requirements

In order to use the API, it is required that an application is registered. To have control of the applications allowed to use our API, it is important to let it have an audit before we grant it access. This to ensure that it can preserve the security of the stored data. But this part is considered out of scope for this thesis. We will mainly focus on the fusion part of the contact trace data. Our prototype will store data of different contact trace methods. It will be possible to retrieve the number of matches encountered in the database based on the data stored by infected users. We mean by number of matches, the number of times that the contact trace information in our database matches with the one sent by the application. Per method we will list up each action that our system need to provide:

BLE Applications

- It should be possible to both upload tokens and timestamps.
- It should be possible to check if a user was in the neighbourhood of an infected user and return the number of matches.

Geographic Location-based

- It should be possible to upload locations and timestamps.

- It should be possible to check if a user was in the neighbourhood of an infected user and return a number of matches.

QR Code-based

- If it is used when meeting another person then it should be possible to upload a token and a timestamp.
- If it is used on a static location then it should be possible to upload a location and a timestamp.
- It should be possible to check if a user was in the neighbourhood of an infected user.

3.1.2 Non-Functional Requirements

- It should hold better results than using only 1 method
- Contact Trace older than 14 days (time needed to be in quarantine after exposure) needs to be removed of the database.[1]

[1] https://www.cdc.gov/coronavirus/2019-ncov/if-you-are-sick/quarantine.html

3.2 Data Management

Now that we know which data we need and the actions that we want to perform, it is important to build up a database from these requirements. The purpose of the database will be to store all the reported cases of infected persons. The identity of the person has a major importance, we decided to work only with the strict minimum data to be able to do the contact tracing. This, to ensure privacy of the infected person and the person behind the application consulting the API. In the related work in Section 2.3 we discussed some techniques that help with preserving this privacy. However, these techniques may not be applicable if we limit our choice of data kept in our database.

To create the design of the database we based ourselves on following sources [37, 38], which help us to form our final Entity Relationship Diagram (ERD). An ERD consists of entities, attributes and relationships. Our entities will be *Application*, *Method*, *Proximty_Tracing* and *Location_Based_Tracing*.
Each entity will have his attributes. But before assigning the attributes, we will identify the relationships between the entity types. Figure 3.3 shows the relation between the entities with a cardinality.

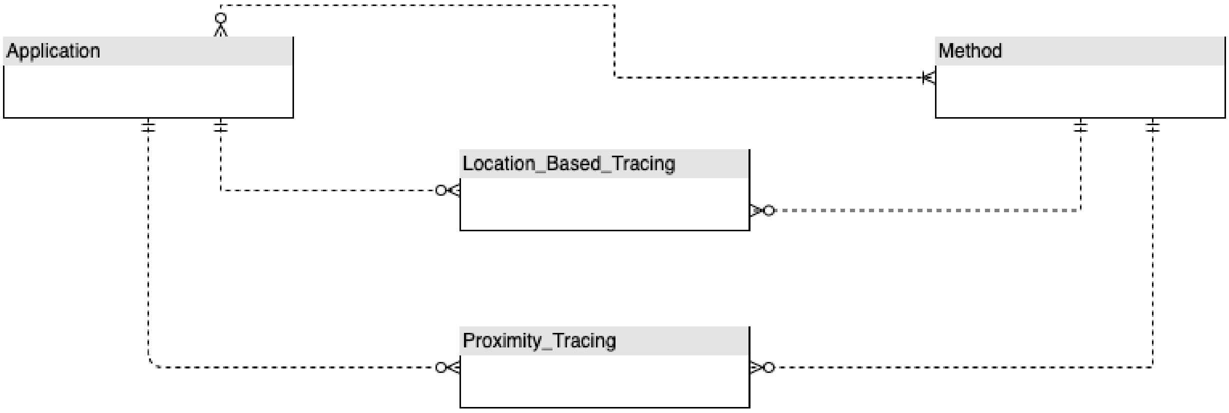


Figure 3.3: Relation between entities

An application can have one or many methods. Our solution will support BLE-based, QR-based and Geolocation-based solutions. An application can be working with QR, but it is also possible that it is combined with BLE. This is envisaged in the case that some applications will use a combination of several methods. But it is not possible to have an application using no method at all. It is possible that the registered application has zero or many location traces and this with using one method. It is possible that an application is registered in the system but will have zero location-based tracing data. But is also possible that it will contain many location-based tracing data. Our system can have zero or many proximity traces and this with using one method. The same applies here as for location based tracing.

The previous step allowed us to create relationships between the entities with corresponding cardinalities. We will give this entities more information by means of attributes.
Figure 3.4 displays the result of the relationship between the entities with their attributes.

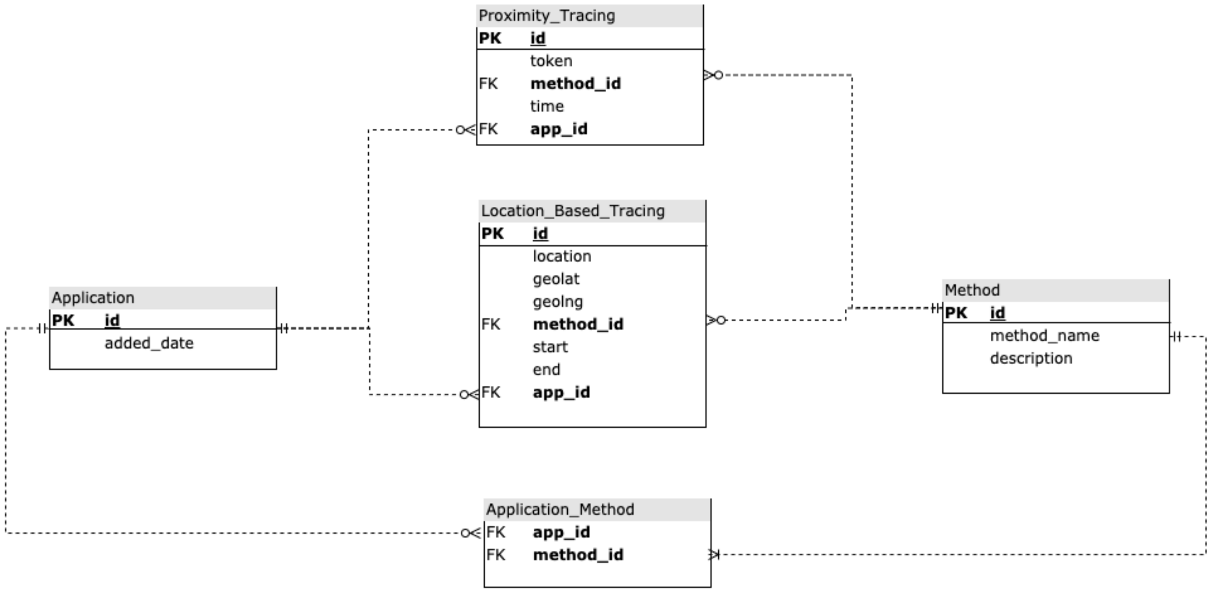


Figure 3.4: ER diagram

`Application` will consist of an *id* that will be unique for the application, this will also be the primary key of the table. The second attribute will be *added_date*, that is the date when the application was added to the system. It is a conscious choice to only have those attribute, we only need the strict minimum of data.

`Proximity_Tracing` will consist of *id* this to preserve uniqueness of the record. The *token* uploaded by the infected person, the *time* when it was collected. In order to know which application uploaded this token, we will also keep a foreign key of the application *app_id.*

`Location_Based_Tracing` will have an unique *id* that will serve as primary key. *Location, geolat and geolng* are needed to be able to see if we may be in the area of an infected person. Location can be used to be more specific, in cases that an application contains several rooms or floors. In addition to geolat and geolng, we can also use location name for example if each floor in a building has another QR code. The *method_id* is a foreign key, to know which method was used to create this record. A *start and end* time in order to know of the person that was infected, was at the same time on the same place. The *app_id* as foreign key to know which application has uploaded this record.

In order to know which technology was used we have `Method`. It will have a primary key *id*, a *method_name* (QR,BLE,LOC) and a description of the method. `Application_Method` is needed to link the entities *Application* and *Method* because of their many to many relationship. Because we can have applications that use more than one method.

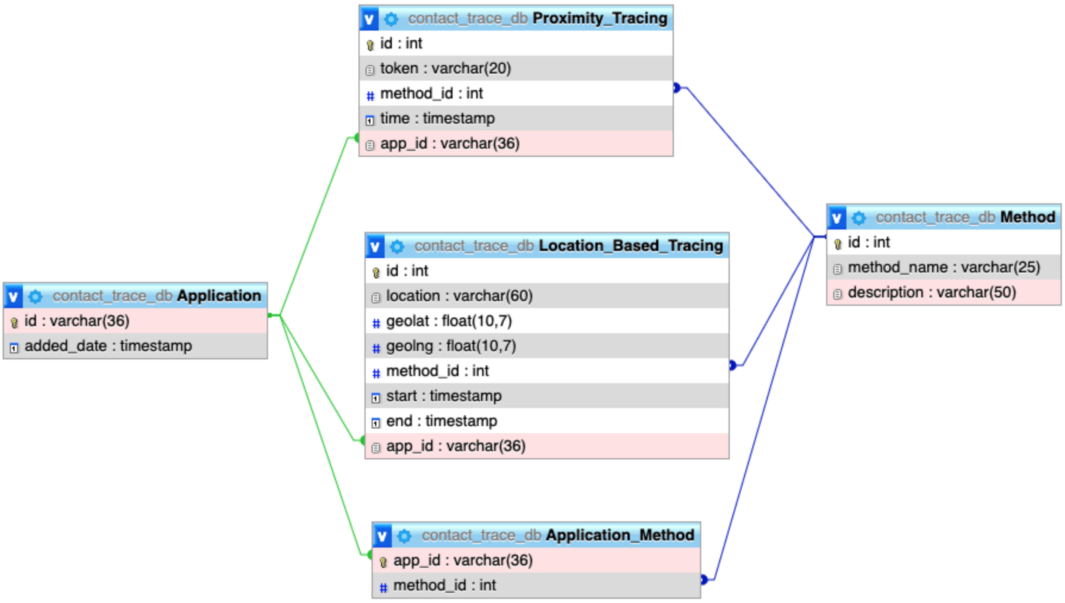


Figure 3.5: Database Schema

Figure 3.5 depicts the final result of our database that we will use in the implementation with the corresponding types for each attribute. Contact trace information of the infected persons will be stored in our database. For the prototype we opt for a MySQL[2]. A relational database was a logical choice because we were looking for a system that respects the ACID properties. Our Database (Figure 3.5) has also a certain relation between the tables that needs to be respected, this confirmed our choice.
In order to meet the non-functional requirement to keep only contact trace information maximum 14 days. We set up an event scheduler that deletes on daily basis data older than 14 days.

[2] https://www.mysql.com

3.3 Sequence Diagram

The key functionality of the API will obviously be uploading infected cases and checking if we were in close contact with an infected person. In Section 3.2 we provided two tables (Proximity_Tracing & Location_Based_Tracing) that will held contact trace information. The reason of this, was the difference of information that must be stored due to the difference of methods. This results into two ways of uploading contact trace information.

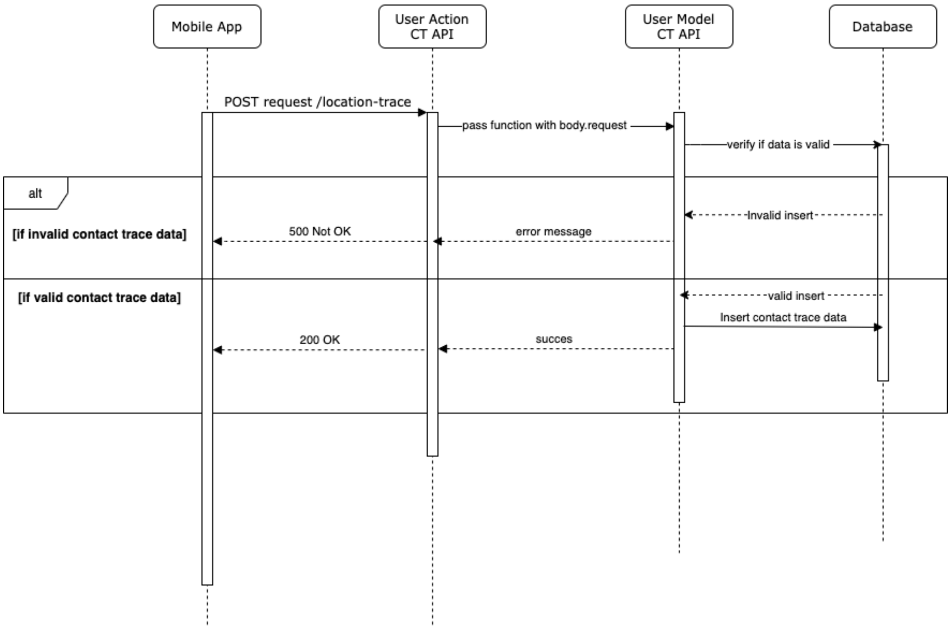

Mobile App
User Action CT API
User Model CT API
Database
POST request /location-trace
pass function with body.request
verify if data is valid
alt
[if invalid contact trace data]
Invalid insert
500 Not OK
error message
[if valid contact trace data]
valid insert
Insert contact trace data
200 OK
succes

Figure 3.6: POST contact trace data of an infected person

Figure 3.6 depicts the process of uploading contact tracing information by using a POST request. If it is location based information that need to be uploaded, the body of the request will contain latitude, longitude, eventually location that will be transformed into latitude & longitude, method used, a start, an end to have specific information about when a person was at a certain place and the application id. If all attributes are valid, the record will be inserted. Else there will be an error instead of an insert.

The process for proximity tracing is similar only the body of the request differs. Instead of giving a location, longitude, (location), start and end time. The application will give a token, time, method and the application id. Figure 3.7 shows how the API should handle the request, to check if we were near a reported infected person. The application will send a request body to our API. Similar as in the upload, we also have two different requests to handle this task here. For the proximity trace the body will contain a token and timestamp. For the location based it will contain the longitude, altitude, optionally the location, the start and end time. When these parameters are transmitted they will be used in a query to check whether there may be a match or not. In case that the query return a value above zero, this means the person was in contact with an infected person. Otherwise, there was no contact so the person is safe.

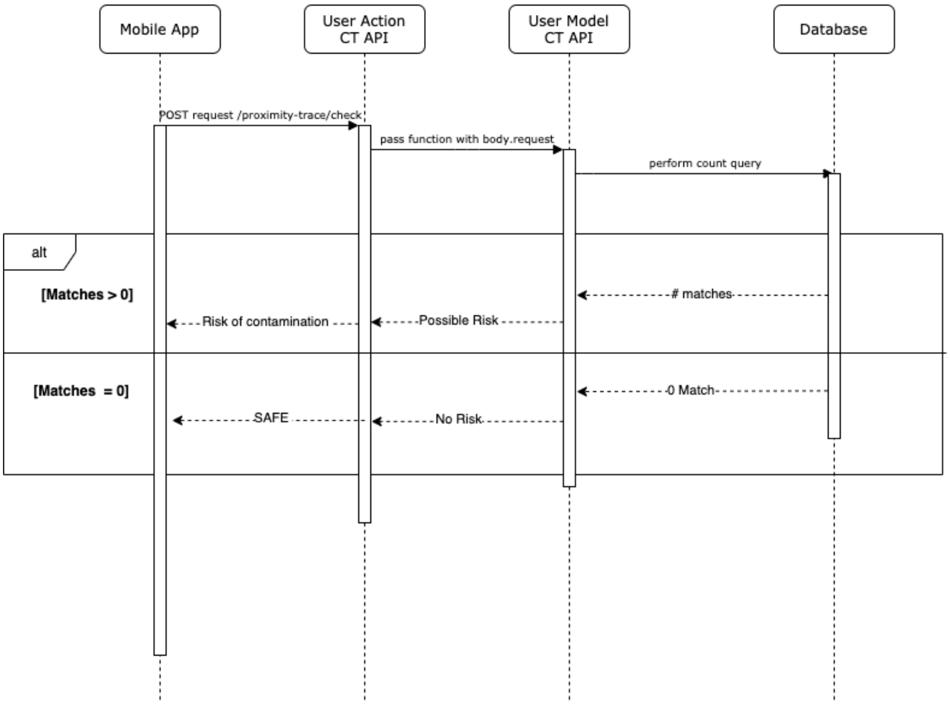


Figure 3.7: Check if person was in area of an infected person

3.4 Technologies

To build the prototype we used Node.js[3], Node.js is a runtime-environment that makes use of the programming language Javascript for building web applications or back-end API services. We used the Express[4] framework to create the REST API. By installing it using the Node.js package manager (npm), this package manager makes it possible to install libraries that are published in an online repository. This repository contains a huge number of free tools. From which we also installed node-geocoder[5] to use OpenCage Geocoding API[6] that helped us to convert coordinates to and from places. We also installed body-parser[7], this helped us to parse incoming request bodies. To enable the connection with the mysql database we installed mysql[8] package. To create unique IDs for each registered application in our database we used the package uuid[9].

We installed phpMyAdmin[10], this gives us the possibility to manage our database by using a graphical user interface running on the localhost. During development we used Postman[11], to test whether the requests sent to the API are being treated. After development we switched to Swagger[12], this interface gives us an overview of the possible requests. It is also a proper way to have a clear documentation of our API.

[3] `https://nodejs.org/en/`

[4] `https://expressjs.com`

[5] `https://www.npmjs.com/package/node-geocoder`

[6] `https://opencagedata.com`

[7] `https://www.npmjs.com/package/body-parser`

[8] `https://www.npmjs.com/package/mysql`

[9] `https://www.npmjs.com/package/uuid`

[10] `https://www.phpmyadmin.net`

[11] `https://www.postman.com`

[12] `https://swagger.io`

3.5 REST API Design

For the solution we will opt for an API. A suitable style for the implementation would be a Representational State Transfer (REST). When applying this architectural style we have to take into consideration the principles [39] that governs this design.

- It is important that each resource has an unique name and is accessible via an Uniform Resource Identifier (URI), avoid the use of CRUD names in the Uniform Resource Locator (URL). Use for documents singular nouns, for collections is it preferable to use plural nouns. It is also recommended to hide the technology used and prevent using lowercase, instead use hyphens.

- It is recommended to use content negotiation, to specify the content-type (e.g. JSON, XML, etc) accepted by the API using the HTTP Accept header instruction.

- The use of appropriate methods to apply operations to avoid tunneling requests. Use *POST* to create, *Get* to retrieve, *PUT* to update, *DELETE* to delete resources, *OPTIONS* is used to retrieve available interactions, *HEAD* is identical as *GET* but to obtain metadata.

- The communication between API and client is *stateless*, the server does not keep the state of the client. This makes the API less complex, because the client is the one providing the information with each request.

3.6 Server

When describing the technologies used, we mentioned that we installed npm package Express.js. This package is needed to build up the server that will listen to incoming requests originating from the different applications. The server will be running on port 5000, and refer the URL requests to the right part of our application using the routes file. The root URL of our application is *http://localhost:5000*, going to it gives blank screen with the following message *Contact Tracing API*. This means that the server is running.

3.7 Routes

As mentioned before requests are redirected to the right file, to use the right function according to the request. It is therefore necessary to import the action files containing the functions that each of these HTTP requests will need.

- POST */proximity-traces*
- GET */proximity-traces/check*
- POST */location-traces*
- GET */location-traces/check*

For each action that we expect from an application we provide a route. To push proximity trace information of an infected person, the application needs to use a POST method containing the right URL */proximity-traces* and a JSON body with the correct values. The function *prox.create* will process together with the body.

To retrieve the number of potential infected contacts we will need the GET */proximity-traces/check* request with the appropriate parameters to be checked in the database.

For location based tools we will expect a POST */location-traces* request with a JSON body entailing the values to be stored in the database.

GET */location-traces/check* including parameters including the values needed to check if there are potential matches.

All these requests will be redirected to the right function to process the request and returning the right response.

3.8 Communication

In order to exchange information between the server and the applications we will use JavaScript Object Notation (JSON) format. This format is lightweight and easy to map with objects. We will use it to send contact trace information into our API, where it will be parsed and stored according to the method to which it belongs.

3.9 Storing Process

After a person has tested positive he will be able to send through the application his contact tracing information to the API. Depending of the method, the API will expect different information of the application. The application needs to be registered in the system to be able to send information. That is why we will only allow information of application that are recognised by means of an appId to use our API.

3.9.1 Proximity Based

A proximity based approach will expect the application to send contact tracing information in JSON format (Listing 3.1) with a body-request to the API by using POST method to following url: `/proximity-traces`.

```
{
    "token": "Token51617887678",
    "methodId": 1,
    "time": "2021-03-24 16:33:16",
    "appId": "9954833b-50d6-4a2f-aa35-1cd71c026c87"
}
```

Listing 3.1: POST Proximity Based Contact Trace Information

The token, methodId, time & appId will be stored in the Proximity_Tracing table. MethodId will be kept to know which method was used to do the tracing. The time will be kept to know when the token was created, the appId to know from which application the information originally came from. Also to keep only information of applications that are registered.

3.9.2 Location Based

When using a location based approach the API will expect the following information of the application (Listing 3.2). This information is sent in a body-request through a POST method using url: `/location-traces`.

```
{
    "location": "Pleinlaan 9, 1050 Ixelles - Elsene, Belgium",
    "geolat":"",
    "geolng":"",
    "methodId": 3,
    "start": "2021-03-25 16:30:12",
    "end": "2021-03-25 16:35:12",
    "appId": "45054cdf-9654-TEST-9560-40d77d9a1cb2"
}
```

Listing 3.2: POST Location Based Contact Trace Information

The information will be stored in the Location_Based_Tracing table. We opted to offer the possibility to give location or the coordinates, but we rely more on the coordinates because they are more accurate. We considered to use Geocoder to translate addresses into coordinates and vice versa, but the Geocoder API doesn't always translate accurately.

3.10 Matching Process

To see if there is a potential threat that a person was in contact with a positif tested person, the API provides to check this. The application of the user will sent contact trace information, this information will be checked in the API. Our API supports three contact tracing methods (QR, BLE, LOC), the prototype that we build provide functionalities for two categories. Both will be supported, but they will expect different input.

3.10.1 Proximity Based

The proximity based approach will expect the application to send a token and a timestamp in a query parameter to the API by using GET method to following url: /proximity-traces/check. The model of the API will process (Appendix Listing A.1) the request and will return the number of potential matches.

3.10.2 Location Based

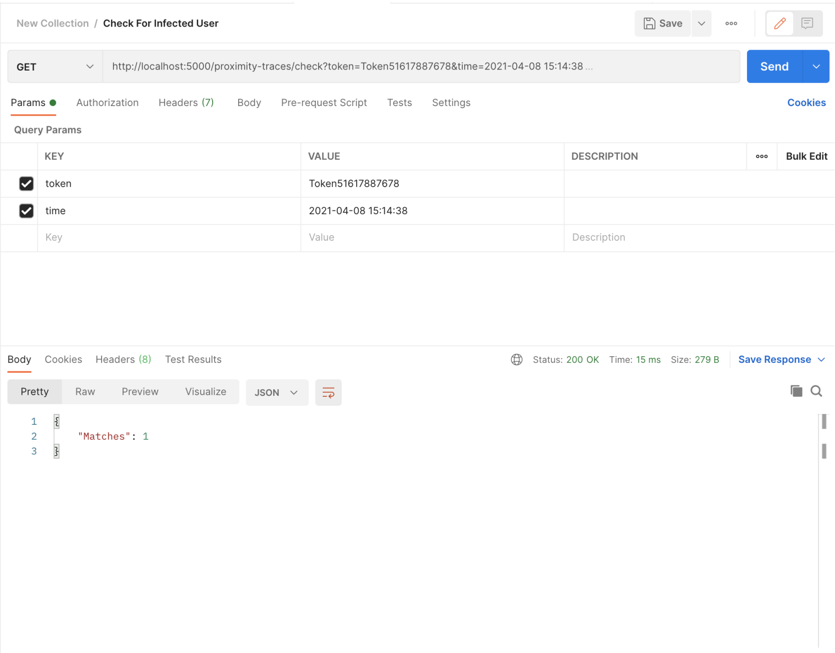
The location based approach will expect the application to send location or longitude/altitude, start & end time in a query parameter to the API by using GET method to following url: /location-traces/check. The model of the API will retrieve the number of matches based on the given parameters using the appropriate query (Appendix Listing A.2 or Listing A.3)

3.11 Security

To ensure the access to the API, applications need to be registered in the system. The applications will be identified by a universally unique identifier (uuid). They will be able to retrieve and upload data if the they dispose of a correct API key. This to ensure that only applications that are in the system can have interaction with the API. To secure the channel with the server we will enable HTTPS, to ensure that the communication is secured.

3.12 Testing

Important during the development was finding a method to test the requests. We used Postman, the API platform offers the possibility to test your API locally in a graphical user interface. The platform helped me to test the request needed to populate the database and the possibility to retrieve possible matches. Figure 3.8 shows a snippet of the tool, it provides the necessary fields to make the requests work. The example on the figure illustrates how we managed to send a Get request to our API, to retrieve the number of token-based matches. We had to choose the type of request, in our case it was a GET. The URL were the request should be sent and the two keys (token & time) along with their two values. We used this tool to test if we also could retrieve the matches of location-based contact tracing methods. To test the storing process for the location-based and token-based method contact, we only needed to change the request type to POST, the URL depending of the method and use the body field to insert the correct data to be transmitted.



New Collection / Check For Infected User

GET http://localhost:5000/proximity-traces/check?token=Token51617887678&time=2021-04-08 15:14:38 ...

Query Params

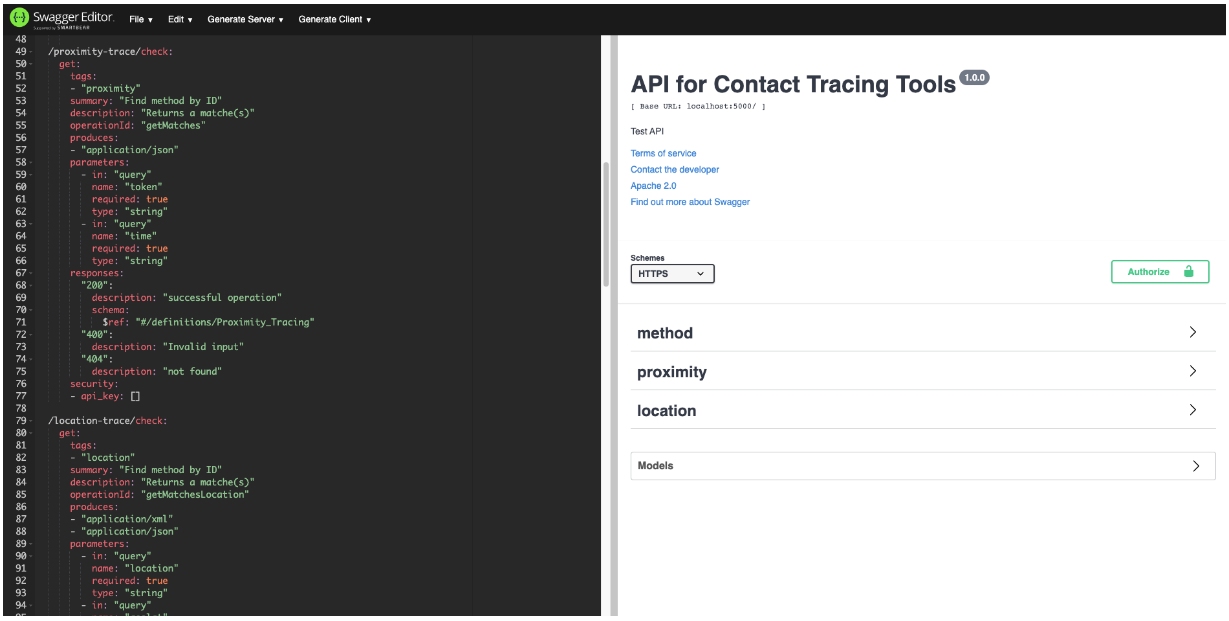
	KEY	VALUE	DESCRIPTION
☑	token	Token51617887678	
☑	time	2021-04-08 15:14:38	
	Key	Value	Description

Status: 200 OK Time: 15 ms Size: 279 B

```
{
    "Matches": 1
}
```

Figure 3.8: Postman

To document the final result of the API, we used Swagger. Figure 3.9 illustrates a snippet of the documentation tool. The tool helps to list all the possible requests of the API. To draw up all these requests we had to use YAML [13], a human-readable data-serialization language.



```
  /proximity-trace/check:
    get:
      tags:
      - "proximity"
      summary: "Find method by ID"
      description: "Returns a matche(s)"
      operationId: "getMatches"
      produces:
      - "application/json"
      parameters:
        - in: "query"
          name: "token"
          required: true
          type: "string"
        - in: "query"
          name: "time"
          required: true
          type: "string"
      responses:
        "200":
          description: "successful operation"
          schema:
            $ref: "#/definitions/Proximity_Tracing"
        "400":
          description: "Invalid input"
        "404":
          description: "not found"
      security:
      - api_key: []

  /location-trace/check:
    get:
      tags:
      - "location"
      summary: "Find method by ID"
      description: "Returns a matche(s)"
      operationId: "getMatchesLocation"
      produces:
      - "application/xml"
      - "application/json"
      parameters:
        - in: "query"
          name: "location"
          required: true
          type: "string"
        - in: "query"
```

API for Contact Tracing Tools

[Base URL: localhost:5000/]

Test API

Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

Schemes: HTTPS

Figure 3.9: Swagger Editor

[13] https://yaml.org/

Chapter 4

Evaluation

In order to evaluate the implementation discussed in Chapter 3, we decided to conduct a simulation. It is important to know that we will focus on the combination of the contact tracing techniques. Thus, we aim to demonstrate that by using our solution, we can increase the accuracy of the matches of being near a contaminated person. For the simulation we will use the map of the WISE/AI labs on Pleinlaan 9(Figure 4.1).

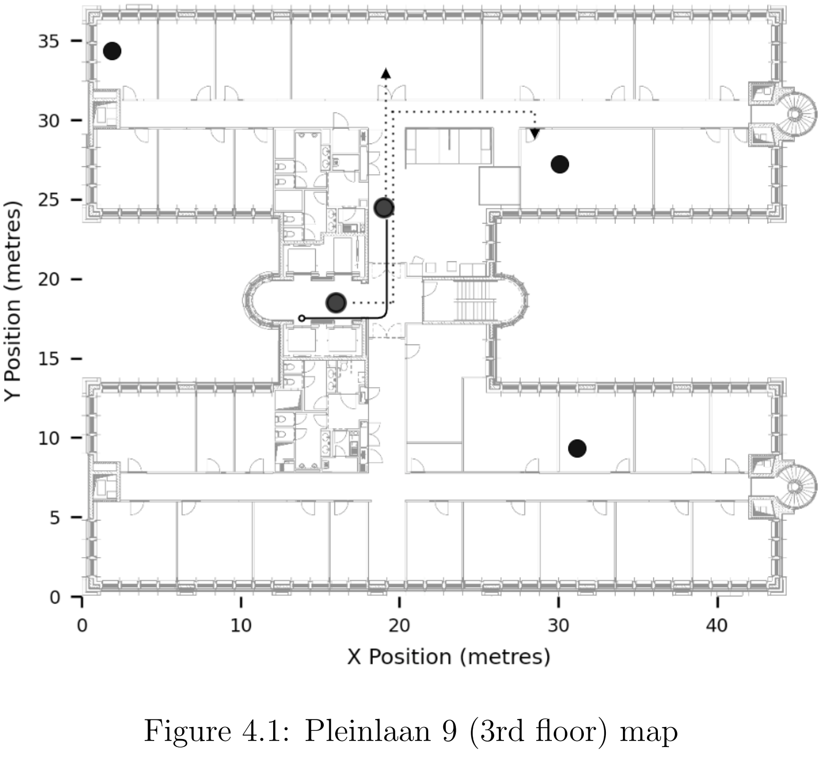


Figure 4.1: Pleinlaan 9 (3rd floor) map

4.1 Simulated Dataset

We will be working with a simulated dataset, which means that we will generate fictitious data. We opted for this approach because this epidemic forbid us to meet in large numbers. The fact that we needed infected users to make the evaluation was also a reason. This data will be used as our control data at the end of the simulation. The dataset was generated using Python, containing generated data of 100 persons including: person, x, y, timestamp, infected, traject. The dataset is a simulation of the trajectory from the 100 persons. The number is deliberately chosen to have a representative number at the end of the experiment. Each x and y represents a position for a certain person, depending on the trajectory that the person is making in the WISE/AI lab on Pleinlaan 9 (3rd floor). We have foreseen three routes randomly selected per person. We set the number of infection to 6%. Our control data will be our reference when we will compare with the data, which we will use to do contact tracing using the API.
We will use the original dataset to generate for each person their contact trace method (QR, BLE & LOC). 80% of the total persons use QR method, 15% of the total persons use BLE method, 35% of the total person use LOC method and there will be a little group using none of the methods at all. Table 4.1 is a snippet of the generated CSV file after randomising the contact trace method(s) for each person.

person	QR	BLE	LOC	x	y	timestamp	infected	traject
Person 0	True	False	False	16.0	18.5	1617887616	False	0
Person 0	True	False	False	17.0	18.5	1617887617	False	0
Person 0	True	False	False	18.0	18.5	1617887618	False	0
Person 0	True	False	False	19.0	18.5	1617887619	False	0
Person 0	True	False	False	19.0	18.5	1617887620	False	0
Person 0	True	False	False	19.0	19.473333333333333	1617887621	False	0
Person 0	True	False	False	19.0	20.446666666666673	1617887622	False	0
Person 0	True	False	False	19.0	21.42	1617887623	False	0
Person 0	True	False	False	19.0	22.39333333333333	1617887624	False	0
Person 0	True	False	False	19.0	23.36666666666667	1617887625	False	0
Person 0	True	False	False	19.0	24.340000000000003	1617887626	False	0
Person 0	True	False	False	19.0	25.31333333333333	1617887627	False	0
Person 0	True	False	False	19.0	26.286666666666665	1617887628	False	0
Person 0	True	False	False	19.0	27.26	1617887629	False	0
Person 0	True	False	False	19.0	28.23333333333333	1617887630	False	0
Person 0	True	False	False	19.0	29.20666666666667	1617887631	False	0
Person 0	True	False	False	19.0	30.180000000000003	1617887632	False	0
Person 0	True	False	False	19.0	31.15333333333333	1617887633	False	0
Person 0	True	False	False	19.0	32.126666666666665	1617887634	False	0

Table 4.1: Simulated CSV in Python

Table 4.2: Number of persons per method

	QR	LOC	BLE	None
Randomized	80%	35%	15%	x
Number of persons	81	31	13	12
Number of infected persons per type	10	3	2	0

Table 4.1 resumes the total number of persons involved per contact tracing method. It is possible that a person uses more than one contact tracing method. 12 persons in the simulation uses none of these methods at all and under these 12 persons we have 0 infected persons. We also give per method the number of infected persons, to indicate the number of potential contaminators.

After this is done we evaluate each method separately by counting the number of matches for each method.Before proceeding to the counting part, we created in Python functions that will calculate for each method the number of matches. Criteria to be a match for the QR method was that the person was at the same time on the same place as an infected persons and both using the QR method (appendix listing B.1). For the BLE method, both persons had to use BLE and supposed to be at the same time within a range of 2 metres from each other (appendix listing B.2). To be able to find matches for the persons using LOC method both person need to be at the same time and the within a range of 4-6 metres from each other (appendix listing B.3).
The result of each method will be
kept separately, so we can compare it with the results given back by our fusion API.

The following step is inserting the CSV generated to calculate matches for each method individually in our fusion API. We used Python scripts to insert each contact tracing method (Appendix Listing B.4, Listing B.5, Listing B.6) through our API in the database.

4.2 Comparison

Table 4.3: Comparison of the results

	QR	LOC	BLE
Number of matches individually	71	22	13
Number of matches with API	71	28	13

After having counting the number of matches and inserting the data in the right table through our API using a script written in Python (Listing B4), we started to check whether a person had a match in our API. For the persons who uses QR nothing changed because every not infected QR person came in contact with an infected QR person. For LOC persons there was a change in the number of matches when using our API, because the API merges the location of all infected persons using either a QR or a LOC contact tracing method. The persons using LOC method could not be linked before when using a separate application to see whether there was a match. Now, because we also keep the infected QR persons in our API we could also reach the persons using the LOC method. For the BLE method there was no difference between using it separately or with the API, because we opted to work with the same application. Suppose that we worked with several users using a different application for BLE with the same way of working, we will had fewer matches when counting the matches separately.

Table 4.4: Persons using different LOC CT applications

Person	LOC	APP	Time	Infected
Person 1	(1,1)	1	22-05-2021 13:00:00	True
Person 2	(1,1)	1	22-05-2021 13:00:00	False
Person 3	(1,1)	2	22-05-2021 13:00:00	False
Person 4	(1,1)	2	22-05-2021 13:00:00	False

Working with 2 different applications to do location-based contact tracing without the help of the designed prototype has one major drawback. We will demonstrate this problem with the help of Table 4.2. Suppose that 4 persons were in the same place (1,1) at same time(22-05-2021 13:00:00), *Person 1* & *Person 2* use the same application, but *Person 3* & *Person 4* a different one. At the moment *Person 1* test positive only *Person 2* will be receiving an alert, because *Person 3* & *Person 4* use a different application they will not be alerted. The prototype solves this problem by accepting different applications in the system. The condition for location-based tracing tools to be able to use the API is that they need to supply the API with the necessary inputs (Time, Location). This will make it possible to reach positive tested people using different applications.

Chapter 5

Conclusion and Future Work

We proposed a prototype to solve the problem of the coordination between the different contact tracing solutions. We tackled this by investigating the different solutions in the Background & Related Work. Our solution support proximity-based and location-based applications or systems. We didn't opt to use an open database because the location-based data could be misused. We discussed some examples in Section 2.3 of published data that revealed the identity individuals. That is also the reason why we opted for the internal database. We have deliberately chosen to transfer contact trace data of positive tested user to the API by using the uuid of the application. The prototype will therefore not include information that could lead directly to a person.

The evaluation demonstrates us that we could benefit of the advantage of merging different methods together in one system gave us a bigger probability to have a match. Different applications using the same method could also benefit of the advantage offered by the API, like demonstrated in the evaluation with Table 4.4.

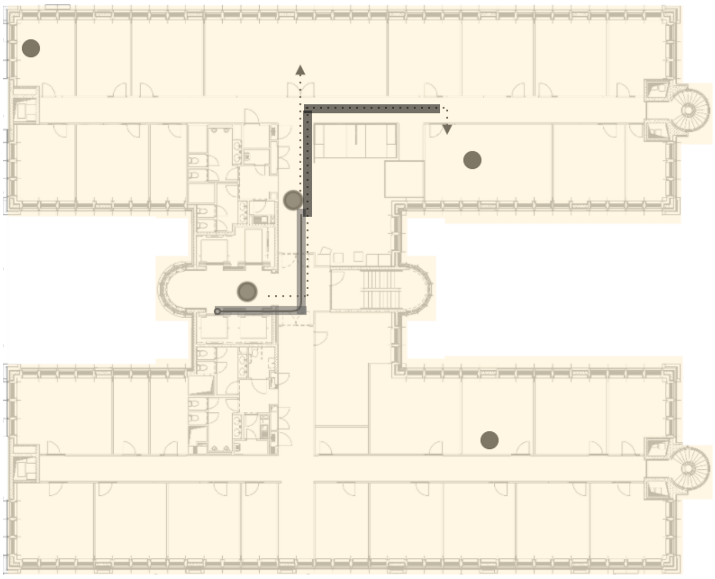


Figure 5.1: Geometric forms

This prototype supports only point geometry, so only QR location-based users who were at the same time and within the range of users using other location-based applications could be matched. We can improve this by using polygon geometry (Yellow area Figure 5.1) instead of point geometry for QR-based contact tracing methods.
During evaluation we did not took into account buildings with floors, not all location-based applications support contact tracing in these cases. This could be resolved by providing an extra field in our database and expect location-based tracing tools to adapt to tracking within buildings. This adaptation could involve an extra technology necessary to be able to work within buildings of public places, by including for example iBeacons [16] this will make it possible to have more accurate positioning.

We strongly believe that this prototype could evolve further and may help when contact tracing would be necessary again (hopefully not).

Appendices

Appendix A

Contact Tracing API

A.1 Proximity Query

```
`SELECT COUNT(*) as Matches
                FROM Proximity_Tracing
                WHERE token = ?
                AND  time = ?`
```

Listing A.1: Query to retrieve BLE matches

A.2 Location Query

```
(`SELECT Count(*) as matches
        FROM Location_Based_Tracing lbt
        WHERE (lbt.location = ? OR (lbt.geolat = ? AND
           lbt.geolng = ?))
        AND lbt.method_id = 3
        AND (? <= lbt.end AND ? >= lbt.start)`
```

Listing A.2: Query to retrieve QR matches

```
(`SELECT Count(*) as matches
          FROM Location_Based_Tracing lbt
          WHERE (6371 * acos( cos( radians(lbt.geolat) )
              * cos( radians( ? ) )
          * cos( radians( ? ) - radians(lbt.geolng) ) +
             sin( radians(lbt.geolat) )
          * sin( radians( ? ) ) ) * 1000 ) <= 4
          AND (? <= lbt.end AND ? >= lbt.start)`
```

Listing A.3: Query to retrieve LOC matches

We relied on the website Movable Type Scripts[1] to calculate the distance between two coordinates in Listing A.3.

A.3 Node-Geocoder

```
  geo.geocode('geolat, geolong', function(err, res) {
   console.log(res);
});
```

Listing A.4: Translate coordinates to location

```
  geo.geocode(location, function(err, res) {
     newLoc.geolat = res[0].latitude
     newLoc.geolng = res[0].longitude
});
```

Listing A.5: Translate location to coordinates

[1] https://www.movable-type.co.uk/scripts/latlong.html

Appendix B

Evaluation

B.1 Contact Matching

```python
def matchQR():
res = pd.DataFrame(columns=["person 1", "x", "y", "timestamp", "infected", "person 2"])
  df = pd.read_csv("final.csv")
  for person_idx in range(0, 100):
   person_trajectory = df[df['person'] == 'Person {}'.format(person_idx)]
   for point,idx in person_trajectory.iterrows():
    match = df[(df['timestamp'] == idx['timestamp']) & (df['person'] != 'Person {}'.format(person_idx)) &
      (df['infected'] == True) & (df['QR'] == True) & (idx['QR'] == True)]
    if not match.empty:
     res = res.append({
          'person 1': match.values[0][0],
          'x': match.values[0][4],
          'y': match.values[0][5],
          'timestamp': match.values[0][6],
          'infected':match.values[0][7],
          'person 2': 'Person {}'.format(person_idx)
          }, ignore_index=True)
  res.to_csv('matchQR.csv', index=False)
```

Listing B.1: Match QR

```
 def matchBLE():
  res = pd.DataFrame(columns=["person 1", "x", "y", "timestamp", "infected", "person 2", "token"])
  df2 = pd.read_csv("final.csv")
  df2['x'] = pd.to_numeric(df2['x'])
  df2['y'] = pd.to_numeric(df2['y'])
  for person_idx in range(0, 100):
   person_trajectory = df2[df2['person'] == 'Person {}'.format(person_idx)]
   for point,idx in person_trajectory.iterrows():
    match = df2[(df2['timestamp'] == idx['timestamp']) & (df2['person'] != 'Person {}'.format(person_idx)) &
      (df2['infected'] == True) & (df2['BLE'] == True) & (idx['BLE'] == True)   ]
    if not match.empty:
     x_dis = abs(match.values[0][4] - int(idx['x']))
     y_dis = abs(match.values[0][5] - int(idx['y']))
     dis = math.sqrt(math.pow(x_dis, 2) + math.pow(y_dis, 2))
     if dis <= 2:
      res = res.append({
                'person 1': match.values[0][0],
                'x': match.values[0][4],
                'y': match.values[0][5],
                'timestamp': match.values[0][6],
                'infected':match.values[0][7],
                'person 2': 'Person {}'.format(person_idx),
                'token': 'Token{}{}'.format(person_idx, idx['timestamp'])
                }, ignore_index=True)
    res.to_csv('matchBLE.csv', index=False)
```

Listing B.2: Match BLE

```
def matchLOC():
  res = pd.DataFrame(columns=["person 1", "x", "y", "timestamp", "infected", "person 2", "person 2 x", "person 2 y", "distance"])
  df1 = pd.read_csv("final.csv")
  for person_idx in range(0, 100):
   person_trajectory = df1[df1['person'] == 'Person {}'.format(person_idx)]
   for point,idx in person_trajectory.iterrows():
    match = df1[(df1['timestamp'] == idx['timestamp']) & (df1['person'] != 'Person {}'.format(person_idx)) &
      (df1['infected'] == True) & (df1['LOC'] == True) & (idx['LOC'] == True)]
    if not match.empty:
     x_dis = abs(match.values[0][4] - int(idx['x']))
     y_dis = abs(match.values[0][5] - int(idx['y']))
     dis = math.sqrt(math.pow(x_dis, 2) + math.pow(y_dis, 2))
     r = randint(4,6)
     if dis < r:
       res = res.append({
          'person 1': match.values[0][0],
          'x': match.values[0][4],
          'y': match.values[0][5],
          'timestamp': match.values[0][6],
          'infected': match.values[0][7],
          'person 2': 'Person {}'.format(person_idx),
          'person 2 x': idx['x'],
          'person 2 y': idx['y'],
          'distance': dis
          }, ignore_index=True)
  res.to_csv('matchLOC.csv', index=False)
```

Listing B.3: Match LOC

B.2 POST to API

```
def insertBLE():
  url='http://localhost:5000/proximity-trace'
  headers = {"content-type":"application/json"}
  df = pd.read_csv('API contact trace data/BLE.csv')
  for point,i in df.iterrows():
      requests.post(url,
          json={
              'token': i['token'],
              'method_id': 1 ,
              'time': datetime.datetime.fromtimestamp(i['timestamp']).isoformat(),
              'app_id': "9954833b-50d6-4a2f-aa35-1cd71c026c87"
          })
```

Listing B.4: POST request for BLE to API

```
def insertQR():
  url='http://localhost:5000/location-trace'
  headers = {"content-type":"application/json"}
  dfx = pd.read_csv("API contact trace data/QR.csv")
  enter = dfx[dfx['timestamp'] == dfx.groupby('person')['timestamp'].transform('min')].drop_duplicates(subset=['person'], keep='last')
  out = dfx[dfx['timestamp'] == dfx.groupby('person')['timestamp'].transform('max')].drop_duplicates(subset=['person'], keep='last')
  for i in range(0, len(enter.index)):
   for j in range(0, len(out.columns)):
      print(enter.values[i][3], out.values[i][3])
      requests.post(url,
       json={
          'location': "Pleinlaan 9, 1050 Ixelles - Elsene, Belgium",
          'geolat': "",
          'geolng': "",
          'method_id': 3,
          'start': datetime.datetime.fromtimestamp(enter.values[i][3]).isoformat(),
          'end': datetime.datetime.fromtimestamp(out.values[i][3]).isoformat(),
          'app_id': "45054cdf-9654-4ab3-9560-40d77d9a1cb2"
      })
```

Listing B.5: POST request for QR to API

```
def insertLOC():
  url='http://localhost:5000/location-trace'
  headers = {"content-type":"application/json"}
  df = pd.read_csv('API contact trace data/LOC.csv')
  for point,i in df.iterrows():
      requests.post(url,
       json={
          'location': " ",
          'geolat': convert(i['x'], i['y'])[0],
          'geolng': convert(i['x'], i['y'])[1],
          'method_id': 2,
          'start': datetime.datetime.fromtimestamp(i['timestamp']).isoformat(),
          'end': datetime.datetime.fromtimestamp(i['timestamp']).isoformat(),
          'app_id':   "a4148a61-cd50-4698-a460-a29343dbd92f"
      })
```

Listing B.6: POST request for LOC to API

Bibliography

[1] Ken TD Eames and Matt J Keeling. Contact Tracing and Disease Control. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270(1533):2565–2571, December 2003.

[2] Roy M Anderson, B Anderson, and Robert M May. *Infectious Diseases of Humans: Dynamics and Control.* Oxford university press, February 1992.

[3] Don Klinkenberg, Christophe Fraser, and Hans Heesterbeek. The Effectiveness of Contact Tracing in Emerging Epidemics. *PLOS ONE*, 1(1):e12, December 2006.

[4] I Thelin, AM Wennström, and PA Mårdh. Contact-Tracing in Patients with Genital Chlamydial Infection. *Sexually Transmitted Infections*, 56(4):259–262, August 1980.

[5] Martin Eichner. Case Isolation and Contact Tracing can Prevent the Spread of Smallpox. *American Journal of Epidemiology*, 158(2):118–128, July 2003.

[6] Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner, Michael Parker, David Bonsall, and Christophe Fraser. Quantifying SARS-CoV-2 Transmission Suggests Epidemic Control with Digital Contact Tracing. *Science*, 368:eabb6936, May 2020.

[7] Lawrence O. Gostin, Daniel Lucey, and Alexandra Phelan. The Ebola Epidemic: A Global Health Emergency. *JAMA*, 312(11):1095–1096, September 2014.

[8] Lisa O Danquah, Nadia Hasham, Matthew MacFarlane, Fatu E Conteh, Fatoma Momoh, Andrew A Tedesco, Amara Jambai, David A Ross, and Helen A Weiss. Use of a Mobile Application for Ebola Contact Tracing and Monitoring in Northern Sierra Leone: a Proof-of-Concept Study. *BMC Infectious Diseases*, 19(1):810, September 2019.

[9] Jilian A Sacks, Elizabeth Zehe, Cindil Redick, Alhoussaine Bah, Kai Cowger, Mamady Camara, Aboubacar Diallo, Abdel Nasser Iro Gigo, Ranu S Dhillon, and Anne Liu. Introduction of Mobile Health Tools to Support Ebola Surveillance and Contact Tracing in Guinea. *Global Health: Science and Practice*, 3(4):646–659, December 2015.

[10] China's Xi Jinping is Pushing for a Global Covid QR code. He may Struggle to Convince the World. `https://edition.cnn.com/2020/11/23/asia/china-xi-qr-code-coronavirus-intl-hnk/index.html`, 2020. Accessed: 2020-11-23.

[11] Elliot Mbunge. Integrating Emerging Technologies into COVID-19 Contact Tracing: Opportunities, Challenges and Pitfalls. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews*, 14(6):1631–1636, December 2020.

[12] World Health Organization et al. Digital Tools for COVID-19 Contact Tracing: annex: Contact Tracing in the Context of COVID-19, 2 June 2020. Technical report, World Health Organization, June 2020.

[13] Dimitrios G Katehakis, Georgios Kavlentakis, Nikos Stathiakis, Fokion Logothetidis, Angelina Kouroubali, Haridimos Kondylakis, Yannis Petrakis, Vassilis Tzikoulis, and Stavros Kostomanolakis. An outbreak Response tool to Effectively Support Surveillance of Suspect, Probable and Confirmed Incidence Cases while Staying Safe in COVID-19. In *2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 432–437. IEEE, 2020.

[14] Ichiro Nakamoto, Sheng Wang, Yan Guo, and Weiqing Zhuang. A QR Code–based Contact Tracing Framework for Sustainable Containment of COVID-19: Evaluation of an Approach to Assist the Return to Normal Activity. *JMIR mHealth and uHealth*, 8(9):e22321, September 2020.

[15] Seble G Kassaye, Amanda Blair Spence, Edwin Lau, David M Bridgeland, John Cederholm, Spiros Dimolitsas, and JC Smart. Rapid Deployment of a Free, Privacy-Assured COVID-19 Symptom Tracker for Public Safety During Reopening: System Development and Feasibility Study. *JMIR Public Health Surveillance*, 6(3):e19399, 2020.

[16] Why use Bluetooth for Contact Tracing? `https://medium.com/indooratlas/why-use-bluetooth-for-contact-tracing-1585feb024dc`, 2020. Accessed: 2021-03-02.

[17] Europe's Plan for Contact Tracing Apps against COVID-19. https://www.law.kuleuven.be/citip/blog/europes-plan-for-contact-tracing-apps-against-covid-19/. Accessed: 2021-01-23.

[18] Mohammad Jabed Morshed Chowdhury, Md Sadek Ferdous, Kamanashis Biswas, Niaz Chowdhury, and Vallipuram Muthukkumarasamy. COVID-19 Contact Tracing: Challenges and Future Directions. *IEEE Access*, 2020.

[19] Shaoxiong Wang, Shuizi Ding, and Li Xiong. A New System for Surveillance and Digital Contact Tracing for COVID-19: Spatiotemporal Reporting Over Network and GPS. *JMIR mHealth and uHealth*, 8(6):e19457, June 2020.

[20] QR Code Development Story. https://www.denso-wave.com/en/technology/vol1.html. Accessed: 2021-04-08.

[21] A Sankara Narayanan. QR Codes and Security Solutions. *International Journal of Computer Science and Telecommunications*, 3(7):69–72, 2012.

[22] VUB Digitalises COVID-Registration using weave.ly. https://weave.ly/blog/vub-digitalises-covid-registration-using-weave-ly/. Accessed: 2021-04-20.

[23] Quick Response (QR) Codes as an Approach to Contact Tracing for COVID-19. https://esnetwork.ca/briefings/qr-codes-as-an-approach-to-contact-tracing-for-covid-19/. Accessed: 2021-04-12.

[24] Andrew S Hoffman, Bart Jacobs, Bernard van Gastel, Hanna Schraffenberger, Tamar Sharon, and Berber Pas. Towards a seamful ethics of Covid-19 contact tracing apps? *Ethics and Information Technology*, pages 1–11, 2020.

[25] Open Data Handbook. https://opendatahandbook.org/guide/en/what-is-open-data/. Accessed: 2021-03-16.

[26] Tim Berners-Lee, 5-star Open Data Plan. https://evolutionpoint.net/the-role-of-linked-open-data-in-seo/. Accessed: 2021-03-20.

[27] What is Five-Star Linked Open Data? https://www.ontotext.com/knowledgehub/fundamentals/five-star-linked-open-data/. Accessed: 2021-03-20.

[28] 5 Star Linked Open Data. https://dvcs.w3.org/hg/gld/raw-file/default/glossary/index.html#x5-star-linked-open-data. Accessed: 2021-03-20.

[29] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data: The Story so Far. In *Semantic services, interoperability and web applications: emerging concepts*, volume 5, pages 1–22.

[30] Rebika Rai and Prashant Chettri. Chapter six - nosql hands on. In Pethuru Raj and Ganesh Chandra Deka, editors, *A Deep Dive into NoSQL Databases: The Use Cases and Applications*, volume 109 of *Advances in Computers*, pages 157–277. Elsevier, 2018.

[31] Kosovare Sahatqija, Jaumin Ajdari, Xhemal Zenuni, Bujar Raufi, and Florije Ismaili. Comparison between Relational and NOSQL Databases. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0216–0221, Opatija, Croatia, May 2018.

[32] Erika McCallister. *Guide to Protecting the Confidentiality of Personally Identifiable Information.* Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, April 2010.

[33] Joana Ferreira Marques and Jorge Bernardino. Analysis of Data Anonymization Techniques. In *KEOD*, pages 235–241, Online, November 2020.

[34] Suntherasvaran Murthy, Asmidar Abu Bakar, Fiza Abdul Rahim, and Ramona Ramli. A Comparative Study of Data Anonymization Techniques. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 306–309, Washington, USA, May 2019.

[35] Kato Mivule. Utilizing Noise Addition for Data Privacy, an Overview. *arXiv preprint arXiv:1309.3958*, 2013.

[36] Mark Elliot, Kieron O'hara, Charles Raab, Christine M O'Keefe, Elaine Mackey, Chris Dibben, Heather Gowans, Kingsley Purdam, and Karen McCullagh. Functional Anonymisation: Personal Data and the Data Environment. *Computer Law & Security Review*, 34(2):204–221, 2018.

[37] Il-Yeol Song and K. Froehlich. Entity-Relationship Modeling. *IEEE Potentials*, 13(5):29–34, January 1995.

[38] What is Entity Relationship Diagram (ERD)? `https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/`. Accessed: 2021-03-20.

[39] Carlos Rodríguez, Marcos Baez, Florian Daniel, Fabio Casati, Juan Carlos Trabucco, Luigi Canali, and Gianraffaele Percannella. REST APIs: a Large-Scale Analysis of Compliance with Principles and Best Practices. In *International conference on web engineering*, pages 21–39. Springer, May 2016.