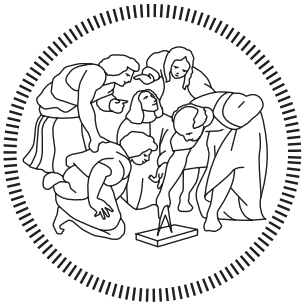
AY 2021-2022



POLITECNICO
MILANO 1863

SMBUD Project

Covid Certificates-Oriented MongoDB Database

Pablo Giaccaglia - 10626428
Santi Pier Pistone - 10867402
Salvatore Cassata - 10560790
Stefano Vighini - 10622788
Zhitao He - 10763530

Master of Science in
Computer Science & Engineering

February 2, 2022

Contents

1 Introduction

This project's purpose is to keep track of **COVID-19 pandemic** data about people, authorized bodies, vaccines, tests and, most of all, Covid certificates of vaccination or testing by designing and implementing a document-based **MongoDB** database. The primary objective is to support a fast tool that checks the validity of the certificate. The data stored allows to extract actionable insights concerning various statistical purposes, involving information such as health services, vaccination & testing hubs and vaccine lots, even though the database is not optimized for these tasks, since the already mentioned main goal regards certificates validity check.

2 Specification & Hypothesis

In order to fulfill the purpose described in the introduction, we created a database with just 3 type of MongoDB documents, managed within their homonym collections:

- **people**, that include all available and necessary demographic information and about people's certification of vaccines and tests.
- **healthcare services**, with main information and also its hubs (where the vaccine or the test has been taken)
- **vaccine lots**. This entity containins specific information about the vaccine lot.

We chose to put the core information in the main document '**Person**', and additional information in other documents. These can be retrieved with an aggregation, but the idea is that these shouldn't be the main accessed data for every-day use.

3 Data Model

3.1 ER Diagram

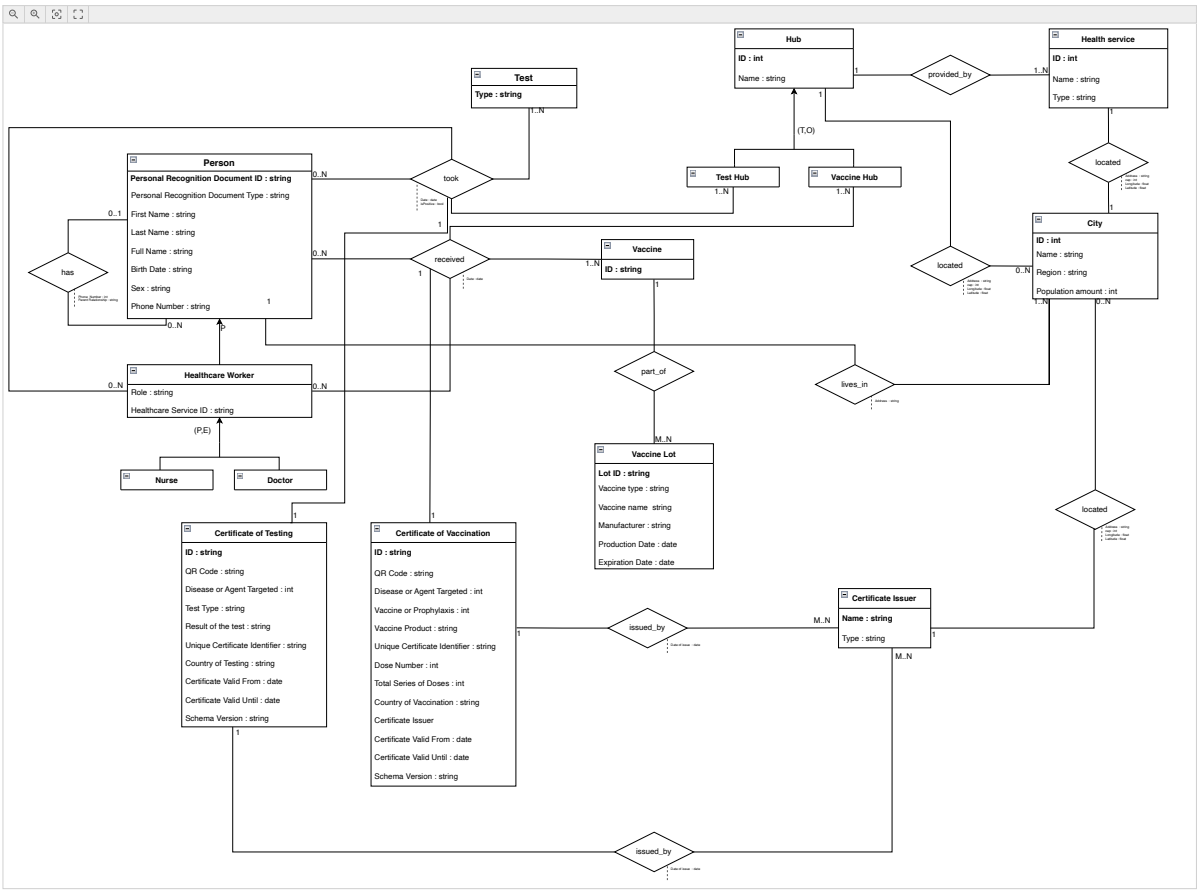


Figure 1: ER Diagram

3.2 Document Diagram

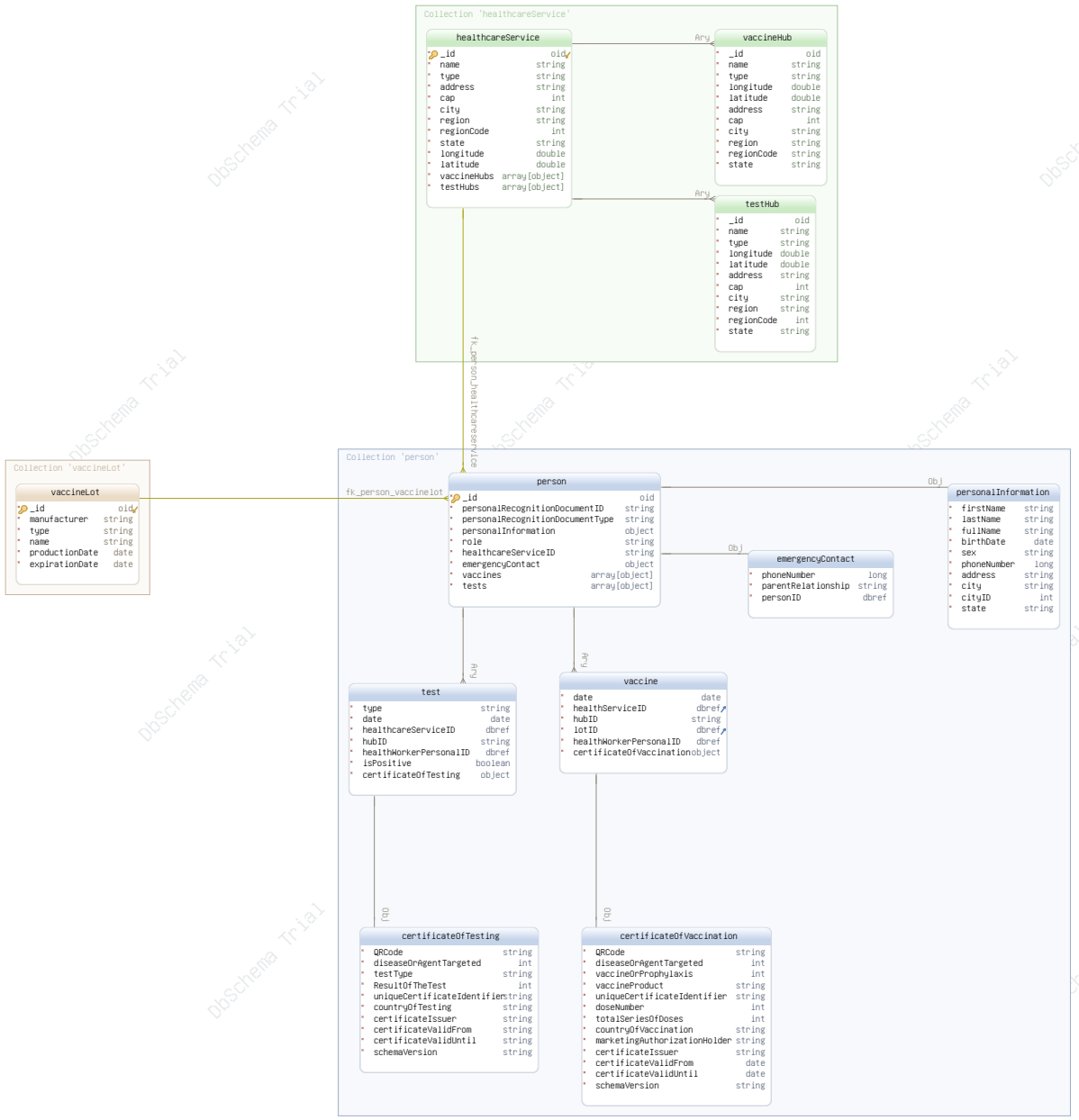


Figure 2: Document Diagram

3.3 From ER to document

The translation of the ER schema in a document approach was not straight forward. We chose to keep some entities, we pushed others inside the 'main' entity. We chose the main entity to be 'person', as our main goal is to verify people's certificates and this entity should be the 'key' of our searches.
Then, there are 2 additional root documents, **HealthcareService** and **VaccineLot**, as they are not related to the person itself. This decision made the database lighter and a bit more readable. Further details will be provided.

Our key reasoning was based on:

- simplify the entities that are not strictly important ('cities') and push their content in related entities
- choose one main entity per group and 'walk through' the relationships to create sub-documents.

To achieve these goals ad-hoc alterations were needed.

One example of modification is that we decided to merge in some ways the vaccine and certificate, but keeping the difference (that will be explained in the next subsection '**Design choices**').
Another decision we made is that when one sub-entity is shared between 2 main entities (for example, the hub, **VaccineHub** or **TestHub**, shared between **HealthcareService** document and **Test/Vaccine** document), it will be stored as a sub-document in the more entailed one, and as a reference in the other one.
We also grouped some information in sub-documents that didn't exist in the ER, like '**personalInformation**', just for readability purposes.

3.4 Design choices

- **information about vaccine lots**. We chose to separate this information from the vaccine in the person just because this data is not so important for our purposes. So, as it may be very redundant, and not so queried, we've put it in a different entity.
- **people that are nurses or doctors**. As stated in the ER diagram, a person can be either a nurse, a doctor or none of them. To distinguish, we kept the same schema for all people, including the role and the healthcareServiceID. If the person is a 'normal' one, these fields are set to null.
- **city information.** Not to overload the database with a lot of entities, we chose to push all information about cities in the respective documents and sub-documents.
- **vaccines, tests and certificates**. We chose to put at the first level of '**vaccines**' and '**tests**' array (in the '**person**' document) the information related and useful for our database. To stick to reality, we also put inside of it a sub-document '**certificate**' that emulates the Green Pass (only in part). Therefore this data is also pretty difficult to query, as it is 'encoded' and structured by a third-party entity (the certificate issuer).

4 Database Creation

The database creation process has relied on various **CSV** datasets freely available online and on **Python**[1] programming language, in order to extract the information suitable to build an appropriate dataset fulfilling our needs. Since information comes from different providers, such as public companies (e.g **DatiOpen.it, dati.salute.gov.it, ec.europa.eu/health**), the need to have a data processing pipeline was high. Thus through several Python functions and libraries we sequentially applied different modification to data, including:

- Null fields rows removal
- Duplicate rows removal
- Standardization of fields to have only first letter uppercase
- Document delimiter conversion to match CSV standard (e.g from "**;**" to "**,**" delimiter)
- Generation of random documents IDs, with respect to the real document number format (e.g : 2 letters and 7 digits for Italian passport number)
- Geographical information (e.g geographic coordinates, address) of places with **GMaps Geocoding API**[2] by providing partial information (e.g : name and the region of a pharmacy), since most of the datasets found online often have incomplete information.
- Generation of random birth dates, phone numbers and assignment of random emergency contacts (including phone number, parental relationship and person ID), residence address, Covid tests, vaccine doses and their respective hubs of administration to people.

The MongoDB philosophy according to which a collection does not require its documents to have the same schema was partially adopted, since instead of having missing fields the document stores null values. This choice was made to ease the bulk population of the database through the Python PyMongo Driver, anyway the database can be adapted to a schema-less or a document validation enforcement structure.
In the following sections, for each collection of the database we provide a brief description of particular problems we had to face during the data retrieval process.

4.1 Collection 'person'

This collection contains the core documents of the database, storing all the information related to a person concerning Covid certificates. The 'person' document is composed by several fields, including embedded documents and references to other documents.
Non Object or Array fields are:

- **_id**: objectId of mongoDB. It corresponds to the encoding of 'personalRecognitionDocumentID'
- **personalRecognitionDocumentID**: the documentID used to identify the person in real life.
- **personalRecognitionDocumentType**: the type of the above document, which is, in the current database storing italian data, one of the following:
  - Italian passport
  - Italian electronic identity card
  - Italian driving license card
- **role**: 'nurse', 'doctor' or null.
- **healthcareServiceID**: the documentID, if the person is a nurse or doctor, null if not.

The array fields are:

- **vaccines**: list of received vaccines.
- **test**: list of taken tests.

Finally among the document's field there are several objects and subobjects, which are analyzed in the following subsections.

4.1.1 SubDocument 'personalInformation'

This document contains demographic details of a person, such as the first name and the phone number. The original data comes from the **2019 Facebook breach**[3]. We selected a single text file coming from the Italian section of the dump, firstly reduced the amount of information (original file size is around **800 MB**) to 5000 entries, then we parsed it, dealing with a non standard data record format, removed unused information (such as relationship status) and added random realistic information (such as birth date).
In particular the document's fields are:

- **firstName**
- **lastName**
- **fullName**
- **birthDate**
- **sex**
- **phoneNumber**
- **address**
- **city**
- **state**

4.1.2 SubDocument 'emergencyContact'

This document contains minimum information about an emergency contact person. The document is composed by the following fields:

- **phoneNumber**
- **parentalRelationship**: degree of kinship with the main person. if the subdocument is not null, the value of this fields in the current database is one of the following:
  - spouse
  - sibling
  - friend
  - father
  - mother
  - son
  - daughter
  - nephew
  - friend
- **personID**: DBRef to the person. It has to be underline that this is not really an ID, but an object containing $ref, $id, and $db fields according to which is the correct way to handle **MongoDB references**[4].

4.1.3 SubDocument 'vaccine'

Element of the '**vaccines**' array.
This document contains all the information regarding a vaccine dose, organized into 2 "groups". The first group is composed by several fields storing information about administration place, date and healthcare personnel. In particular these fields are:

- **date** : Date of the vaccination. In the database this value ranges from **2021-01-05** to **2021-05-30**
- **healthServiceID**: randomly generated 24 digits hexadecimal code identifying a health service organization operating in a certain area. In the database these ids are related to italian '**ASL**' Offices. It has to be underlined that this is not really an ID, but an object containing $ref, $id, and $db fields according to which is the correct way to handle **MongoDB references**[4].
- **hubID**: randomly generated 24 digits hexadecimal code identifying a vaccination hub.
- **lotID**: randomly generated 24 digits hexadecimal code identifying a vaccine lot. It has to be underlined that this is not really an ID, but an object containing $ref, $id, and $db fields according to which is the correct way to handle **MongoDB references**[4].
- **healthWorkerPersonID**: randomly generated 24 digits hexadecimal code representing the fixed length hex encoding of the personalRecognitionDocumentID belonging to the healthcare worker who administered the dose. It has to be underlined that this is not really an ID, but an object containing $ref, $id, and $db fields according to which is the correct way to handle **MongoDB references**[4].

The second group composed by a '**certificateOfVaccination**' Object, which is analyzed in the following subsection.

4.1.4 SubSubDocument 'certificateOfVaccination'

This document contains information which reflects a real green certificate. In fact its fields (excepting the **QRCode** field) are compliant to the official **European eHealth network COVID certificate JSON Schema Specification**[5], even though not all the specified fields are here included. In particular these fields are :

- **QRCode**: This string value represent the encoded JSON '**certificateOfVaccination**' document, excluding this field. The process for producing this remotely mimics what the **European eHealth Trust Framework for Certificates**[6] expects. The process we applied is the following:
  - JSON Dump of the Python dictionary representing the certificate.
  - UTF-8 encoding of the JSON string.
  - Base 45 encoding of the bytes generated by the previous encoding.
  - Compression with zlib of the previously generated bytes.
  - QR Code Image generation with qrcode Python library.
  - Base 64 encoding of image bytes.
  - Bytes conversion to string.

  Through appropriate functions the process can be performed backwards to obtain both the QR Code and the original JSON string.
- **diseaseOrAgentTargeted**: This value set has a single entry **840539006** , which is the code for COVID19 from SNOMED CT.
- **vaccineOrProphylaxis**: **SNOMED CT** code indicating the vaccine or prophylaxis used. The mapping is the following:
  - **SARS-CoV2 antigen vaccine**: 1119305005
  - **SARS-CoV2 mRNA vaccine**: 1119349007
- **vaccineProduct**: Code complying the Union Register of medicinal products code system representing Medicinal product used for the specific dose of vaccination, The mapping is the following:
  - **Pfizer Vaccine**: EU/1/20/1528

  - **Moderna Vaccine**: EU/1/20/1507
  - **AstraZeneca Vaccine**: EU/1/21/1529
  - **Janssen Vaccine**: EU/1/20/1525

- **uniqueCertificateIdentifier**: Unique certificate identifier (**UVCI**), whose structure mimics the one specified in this document only in terms of sequence of digits and characters.
- **doseNumber**:Sequence number (positive integer) of the dose given during a vaccination event. 1 for the first dose, 2 for the second dose etc.
- **totalSeriesOfDoses**: Total number of doses (positive integer) in a complete vaccination series according to the used vaccination protocol. In the database this value is set to 1 for "Janssen" vaccine certificate, 2 in all the other cases.
- **countryOfVaccination**: Country expressed as a 2-letter ISO3166 code. In the current database this value is 'IT', since the data regards Italy.
- **marketingAuthorizationHolder**: Marketing authorisation holder code from **EMA SPOR Organisations Management System**. The mapping is the following:
  - **AstraZeneca AB**: ORG-100001699
  - **Biontech Manufacturing GmbH**: ORG-100030215
  - **Janssen-Cilag International**: ORG-100001417
  - **Moderna Biotech Spain**: ORG-100031184
- **certificateIssuer**: Name of the organisation that issued the certificate. In the current database this value is always *Italian Ministry of Health* since the data regards Italy.
- **certificateValidFrom**: The first date on which the certificate is considered to be valid, provided in the format **YYYY-mm-ddTHH:MM:ss**. Following what specified here, in the current database this date if after 15 days from the first dose and after 3 days in case of "Janssen" Vaccine.
- **certificateValidUnti**: The last date on which the certificate is considered to be valid, assigned by the certificate issuer, provided in the format **YYYY-mm-ddTHH:MM:ss**. Following what specified here, in the current database this date is after 28 days from the vaccination in case of first dose, 270 days in case of second dose or single dose.
- **schemaVersion**: Value matching the identifier of the schema version used for producing the EUDCC. In the current database this value is set to **1.0.0**.

4.1.5 SubDocument 'test'

Element of the '**tests**' array.
This document contains all the information regarding a taken test, organized into 2 "groups". The first group is composed by several fields storing information about administration place, date and healthcare personnel. In particular these fields are:

- **type**: Type of the test {Molecular test/Antigen test/Antibody test}
- **date**: Date of the testing. In the current database this value ranges from **2020-03-10** to **2021-12-08**
- **healthcareServiceID**: randomly generated 24 digits hexadecimal code identifying a health service organization operating in a certain area. In the current database these ids are related to italian **ASL** Offices. It has to be underlined that this is not really an ID, but an object containing $ref, $id, and $db fields according to which is the correct way to handle **MongoDB references**[4].
- **hubID**: randomly generated 24 digits hexadecimal code identifying a testing hub.
- **healthWorkerPersonalID**: randomly generated 24 digits hexadecimal code representing the fixed length hex encoding of the personalRecognitionDocumentID belonging to the healthcare worker who administered the dose. It has to be underlined that this is not really an ID, but an object containing $ref, $id, and $db fields according to which is the correct way to handle **MongoDB references**[4].
- **isPositive**: Boolean value field, which indicates the outcome of the test.

The second group composed by a '**certificateOfVTesting**' Object, which is analyzed in the following subsection.

4.1.6 SubSubDocument 'certificateOfTesting'

This document contains information which reflects a real green certificate. In fact its fields (excepting the **QRCode** field) are compliant to the official European eHealth network COVID certificate JSON Schema Specification, even though not all the specified fields are here included. In particular these fields are :

- **QRCode**: This string value represent the encoded JSON 'certificateOfVaccination' document, excluding this field. The process for producing this remotely mimics what the European eHealth Trust Framework for Certificates expects. The process we applied is the following:
  - JSON Dump of the Python dictionary representing the certificate.
  - UTF-8 encoding of the JSON string.
  - Base 45 encoding of the bytes generated by the previous encoding.
  - Compression with zlib of the previously generated bytes.
  - QR Code Image generation with qrcode Python library.
  - Base 64 encoding of image bytes.
  - Bytes conversion to string.
- **diseaseOrAgentTargeted**: **diseaseOrAgentTargeted**: This value set has a single entry **840539006** , which is the code for COVID19 from **SNOMED CT**.
- **testType**: The type of the test used's LOINC code, based on the material targeted by the test. According to this report, the mapping is the following :
  - **Molecular**: 94309-2
  - **Antigen**: 94558-4
  - **Antibody**: 94762-2
- **resultOfTheTest**: coded value based on **SNOMED CT**. The mapping is the following:
  - **Detected**: 260373001
  - **Not Detected**: 260415000
- **uniqueCertificateIdentifier**: Unique certificate identifier (**UVCI**), whose structure mimics the one specified in this document only in terms of sequence of digits and characters.
- **countryOfTesting**: Country expressed as a 2-letter **ISO3166** code. In the current database this value is *IT*, since the data regards Italy.
- **certificateIssuer**: Name of the organisation that issued the certificate. In the current database this value is always *Italian Ministry of Health* since the data regards Italy.
- **certificateValidFrom**: The first date on which the certificate is considered to be valid, provided in the format **YYYY-mm-ddTHH:MM:ss**. Following what specified here, in the current database this date is after 3 days for the Antibody test, 2 days the Molecular test and 0 days for the Antigen test, which are the result wait days.
- **certificateValidUnti**: The last date on which the certificate is considered to be valid, assigned by the certificate issuer, provided in the format **YYYY-mm-ddTHH:MM:ss**. Following what specified here, in the current database this date is after 3 days from the result date for the Molecular test and the Antibody test, 2 days for the Antigen test.
- **schemaVersion**: Value matching the identifier of the schema version used for producing the **EUDCC**. In the current database this value is set to **1.0.0**.

4.2 Collection 'healthcareService'

This document contains all the information regarding authorized bodies, providing vaccination and testing hubs in a certain area, such as a region or a portion of it. In the current database these represent Italian *ASL* Offices. The document's structure is organized into 2 "groups". The first group is composed by several fields storing information about the service itself such as the type and its headquarters location. In particular these fields are:

- **_id**: ObjectId of mongoDB.
- **name**
- **type**
- **address**
- **cap**
- **city**
- **region**
- **regionCode**
- **state**
- **coordinates**: object containing the GPS coordinates of the healthcare service. In detail this is a GeoJSON '**Point**' object type[7] .

The second group is composed by two arrays:

- **vaccineHubs**
- **testHubs**

These two type of documents are analyzed in the following section.

4.2.1 SubSubDocuments 'vaccineHub' & 'testHub'

This document contains information regarding a vaccination hub, such as location information, name and type.
In particular the document's fields are:

- **_id**: ObjectId of MongoDB.
- **name**
- **type**
- **coordinates**: object containing the GPS coordinates of the healthcare service. In detail this is a GeoJSON '**Point**' object type.
- **address**
- **cap**
- **city**
- **region**
- **regionCode**
- **state**

4.3 Collection 'vaccineLot'

This document contains information regarding Vaccines lot, such as the manufacturer and the expiration date.
In particular the document's fields are:

- **_id**: ObjectId of MongoDB.
- **manufacturer**: for ex. 'Astrazeneca'.
- **type**: for ex. 'mRNA'
- **productionDate**: Provided in the format YYYY-mm-ddTHH:MM:ss.
- **expirationDate**: Provided in the format YYYY-mm-ddTHH:MM:ss.

5 Queries & Commands

5.1 Queries

- **Check if a person has a valid certificate (sample or vaccine)**

```
db.person.find({
    $or: [{
        'personalRecognitionDocumentID': "UO40378TZ",
        'vaccines': { $elemMatch: {
        'certificateOfVaccination.certificateValidFrom':
        { $lte: new ISODate() },
        'certificateOfVaccination.certificateValidUntil':
        { $gte: new ISODate() }
    }}}, {
        'personalRecognitionDocumentID': "UO40378TZ",
        'tests': { $elemMatch: {
        'certificateOfTesting.certificateValidFrom': {$lte: new ISODate()},
        'certificateOfTesting.certificateValidUntil':
        {$gte: new ISODate() }
}}}]}).count() > 0
```

```
> db.person.find({
    $or: [ {
      'personalRecognitionDocumentID': "UO40378TZ",
      'vaccines': { $elemMatch: {
        'certificateOfVaccination.certificateValidFrom': { $lte: new ISODate() },
        'certificateOfVaccination.certificateValidUntil': { $gte: new ISODate() }
      }}
    }, {
      'personalRecognitionDocumentID': "UO40378TZ",
      'tests': { $elemMatch: {
          'certificateOfTesting.certificateValidFrom': { $lte: new ISODate() },
          'certificateOfTesting.certificateValidUntil': { $gte: new ISODate() }
      }}}] }).count() > 0
< true
```

Figure 3: Result of Query 1

- **Get where, when a person did the vaccine, and who was the nurse/doctor who did it.**

```
db.person.aggregate(
    {$unwind: "$vaccines"},
    {$addFields: {"hubID": {"$toObjectId": "$vaccines.hubID"}}},
    { $lookup: {
        from: 'healthcareService',
        localField:'hubID',
        foreignField: 'vaccineHubs._id',
        as: 'hub'} },
    { $lookup: {
        from: 'person',
        localField:'vaccines.healthWorkerPersonalID.$id',
        foreignField:'_id',
        as: 'worker'} },
    { $project: {
        "date":"$vaccines.date",
        "place": {
        "city":"$hub.city",
        "address":"$hub.address",
        "region":"$hub.region",
        "state":"$hub.state"
        },
        "worker":"$worker.personalInformation"
}})
```

```
{ _id: ObjectId("4c4b323630353032364c4b"),
  date: 2021-01-21T00:00:00.000Z,
  place:
   { city: [ 'Napoli' ],
     address: [ 'Centro Direzionale Is. F/9' ],
     region: [ 'Campania' ],
     state: [ 'Italy' ] },
  worker:
   [ { firstName: 'Cono Paolo',
       lastName: 'Santoro',
       fullName: 'Cono Paolo Santoro',
       birthDate: 1939-10-11T00:00:00.000Z,
       sex: 'male',
       phoneNumber: 393701457102,
       address: 'Via Benedetto Fortini 139',
       city: 'Resana',
       state: 'ITA' } ] }
```

Figure 4: Result of Query 2

- **Retrieve the emergency contact of a person.**

```
db.person.find(
    {'personalRecognitionDocumentID': 'LK2605026F'},
    {'emergencyContact.phoneNumber':1})
```

```
> db.person.find
  ({'personalRecognitionDocumentID': 'LK2605026F'},{'emergencyContact.phoneNumber':1})
< { _id: ObjectId("4c4b32363035303236464c4b"),
    emergencyContact: { phoneNumber: 393510544400 } }
```

Figure 5: Result of Query 3

- **Count the number of samples a person has taken.**

```
db.person.aggregate([
    {"$match":{'personalRecognitionDocumentID':"LK2605026F"}},
    {$project:{count:{$size:"$tests"}}}])
```

```
> db.person.aggregate
  ([{"$match":{'personalRecognitionDocumentID':"LK2605026F"}},
    {$project:{count:{$size:"$tests"}}}])
< { _id: ObjectId("4c4b32363035303236464c4b"), count: 1 }
```

Figure 6: Result of Query 4

- **Retrieve the amount of people Vaccinated per date**

```
db.person.aggregate([
    {$unwind:"$vaccines"},
    {"$group":{_id:"$vaccines.date",count:{"$sum":1}}}])
```

```
> db.person.aggregate
  ([{$unwind:"$vaccines"},
    {"$group":{_id:"$vaccines.date",count:{"$sum":1}}}])
< { _id: 2021-01-16T00:00:00.000Z, count: 32 }
  { _id: 2021-05-27T00:00:00.000Z, count: 18 }
  { _id: 2021-05-11T00:00:00.000Z, count: 22 }
  { _id: 2021-03-14T00:00:00.000Z, count: 52 }
  { _id: 2021-02-02T00:00:00.000Z, count: 64 }
  { _id: 2021-03-07T00:00:00.000Z, count: 64 }
  { _id: 2021-04-20T00:00:00.000Z, count: 45 }
```

Figure 7: Result of Query 5

- **Retrieve the top 5 dates per amount of tested people**

```
db.person.aggregate([
    {$unwind:"$tests"},
    {"$group":{_id:"$tests.date",count:{"$sum":1}}},
    {$sort:{"count":-1}},{$limit:5}])
```

```
> db.person.aggregate
  ([{$unwind:"$vaccines"},
  {"$group":{_id:"$vaccines.date",count:{"$sum":1}}},
  {$sort:{"count":-1}},{$limit:5},
  {$project:{"date":"$_id"}}])
< { _id: 2021-04-17T00:00:00.000Z,
    date: 2021-04-17T00:00:00.000Z }
  { _id: 2021-03-05T00:00:00.000Z,
    date: 2021-03-05T00:00:00.000Z }
  { _id: 2021-03-24T00:00:00.000Z,
    date: 2021-03-24T00:00:00.000Z }
  { _id: 2021-04-05T00:00:00.000Z,
    date: 2021-04-05T00:00:00.000Z }
  { _id: 2021-04-30T00:00:00.000Z,
    date: 2021-04-30T00:00:00.000Z }
```

Figure 8: Result of Query 6

- **Compute the average number of persons tested per Health Service and return the top 5 Health Service per average persons tested.**

```
db.person.aggregate([
    {$unwind:"$tests"},
    {$lookup: {
        from: 'healthcareService',
        localField: 'tests.healthcareServiceID.$id',
        foreignField: '_id',
        as: 'healthcareServices'
    }},
    {$group: {
        _id:"$healthcareServices.name",
        count:{$sum:1}}},
    {$sort:{"count":-1}},
    {$limit:5}])
```

```
> db.person.aggregate([{$unwind:"$tests"},{$lookup:{
        from: 'healthcareService',
        localField: 'tests.healthcareServiceID.$id',
        foreignField: '_id',
        as: 'healthcareServices'
      }},{$group:{_id:"$healthcareServices.name",count:{$sum:1}}},{$sort:{"count":-1}},{$limit:5}])
< { _id: [ 'Asur Marche' ], count: 238 }
  { _id: [ 'A.S.L. Napoli 1 Centro' ], count: 229 }
  { _id: [ 'Asp Ragusa' ], count: 206 }
  { _id: [ 'A.S.L. Caserta' ], count: 125 }
  { _id: [ 'Asl Biella' ], count: 113 }
```

Figure 9: Result of Query 7

5.2 Commands

The following subsection presents some useful commands, with the aim of showing database's utility.

- **A person has taken the vaccine:** add it.

```
db.person.update(
    { personalRecognitionDocumentID:"LK2605026F" },
    { $push: { vaccines: {
        date: new ISODate("2021-12-11"),
        healthServiceID:{$ref:"healthcareService",
        $id:"␣53554d32353953554d323539",
        $db:"SMBUD"},
        hubID:"414948313335414948313335",
        lotID:{
        $ref:"vaccineLot",
        $id:"492876f0ca87930680eb7747",
        $db:"SMBUD"},
        healthWorkerPersonalID:{
        $ref:"person",
        $id:ObjectId("4c4b32360353032364c4b"), $db:"SMBUD"},
        certificateOfVaccination: {
            QRCode:"iVBORw0KGgoAAAANSUhEUgAABJIAAASSAQAAA...",
            diseaseOrAgentTargeted:840539006,
            vaccineOrProphylaxis:1119349007,
            vaccineProduct:"EU/1/20/1507",
            uniqueCertificateIdentifier:"01IT5UTPPVIJ713AJCY2...",
            doseNumber:2,
            totalSeriesOfDoses:2,
            countryOfVaccination:"IT",
            marketingAuthorizationHolder:"ORG-100031184",
            certificateIssuer:"Italian␣Ministry␣of␣Health",
            certificateValidFrom: new ISODate("2021-12-20"),
            certificateValidUntil:new ISODate("2022-09-20"),
            schemaVersion:"1.0.0"
        }
    }}}
)
```

- **Delete the first covid test a person has taken.**

```
db.person.update(
    { personalRecognitionDocumentID:"LK2605026F" },
    { $unset: {'tests.0':1} } )
db.person.update(
    { personalRecognitionDocumentID:"LK2605026F" },
    { $pull: { tests:null } } )
```

- **Update the emergency phone number of a person.**

```
db.person.update(
    { personalRecognitionDocumentID:"LK2605026F" },
    { $set: { 'emergencyContact.phoneNumber':"393510544400" } }
)
```

A Delivery content

Apart from this report, the delivery folder also contains the following files:

- **Python scripts** to create CSVs and database. The Python project is mainly composed by 4 files
  - **csvmanipulation.py**: it contains various utility functions to generate random data&manipulate CSV files.
  - **main.py**: it contains a class to handle database connection and collections/documents creation through various functions.

  - **qrcodeutils.py**: it contains various utility functions to encode text, generate, read and recognize qr codes, decode strings related to certificates.
  - **utils.py**: it contains various utility function to generate pseudo random data, such as phone numbers and document ids.

- **Folders** containing the various versions of CSVs files, to keep track of the subsequent modifications (these are located inside the Python project folder, inside the **datasets** folder)
- **Database dump** file.
- **Text file** with queries&commands.

References

[1] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[2] Google, "Google maps geocoding api documentation," *https://developers.google.com/maps/documentation/geocoding/*

[3] TheGuardian, "Facebook data leak," *https://www.theguardian.com/technology/2021/apr/03/500-million-facebook-users-website-hackers*, 2021.

[4] MongoDB, "Database references," *https://docs.mongodb.com/manual/reference/database-references/*.

[5] E. Union, "Technical specifications for eu digital covid certificates," *https://ec.europa.eu/health/sites/default/files/ehealth/docs/covid-certificate$_{j}son_{s}pecification_{e}n.pdf$*.

[6] E. Union, "verifiable vaccination certificates - basic interoperability elements," *https://ec.europa.eu/health/sites/default/files/ehealth/docs/vaccination-proof$_{i}nteroperability - guidelines_{e}n.pdf$*.

[7] MongoDB, "Geojson objects," *https://docs.mongodb.com/manual/reference/geojson/*, 2021.