Academic Year 2020/2021

POLITECNICO
MILANO 1863

Computer Science and Engineering

System and Methods for Big and Unstructured Data

Project Report - MongoDB

Matteo Falzi - 10638723
Flavio La Manna - 10620549
Filippo Manzardo - 10864201
Paolo Marzolo - 10668259
Matteo Regge - 10619213

Prof. Marco Brambilla

December 12, 2021

1 Introduction

The measures taken by the Government for the management of the pandemic have generated a multitude of documents, the creation of an application for the management of these documents by the authorities and the citizen has become far more than a need. The authorities have created a Covid pass for people vaccinated and/ or tested. This pass removes the movement limits imposed by the restrictions due to the pandemic, allows access to bars, restaurants, cinemas, schools as it ensures that the citizen who is in possession of it is in a "safe" state of health. In this way the government can track the movements of the people and ensures that there is less possibility of spreading the virus among the population. Obviously the certificates are released by authorized bodies such as hospitals and private clinics. The certificates have a limited validity so there is an expiration date that if exceeded, they need to be updated.

1.1 Delivery specification

The goal of the project is: designing, storing and using a NoSQL document-based storage for collecting the certificates of vaccination or testing and the authorized bodies that can release them. The certificates contain the personal information of the people, their Covid vaccionations and tests. All these documents will be used to support an app for COVID-19. All the work done can be found at this link: GitHub Repo

1.2 Assumptions

Validation: We decided to model the validity of the certificates with this field. It's composed by two sub-fields: Version and Expiration Date. Version indicates the law that was in force when the certificate was released and every rule has a different duration validity (used to calculate the expiration date). In this way we can keep up with the evolution of the rules because if a new rule is enacted, we can update this sub-field for the documents that need to be updated and consequently recalculate the expiration date.

Places: This collection contains the authorized bodies (like hospitals, pharmacies, clinics) that we considered. They all have a name, geographic coordinates and geographic information.The dataset containing places was retrieved from GeoNames and it has been pre-processed for better use.

Doctor/Nurse: For each test and vaccine it's always indicated the doctor who did the vaccine (with his personal information) and at least one nurse that assisted (with their personal information). There can be at most five nurses for one test or vaccine.

2 Relational Model

To approach the problem, we developed an ER Model based of 7 entities: Person with Doctor and Nurse, Test, Vaccines,Place and the main certificate. Following the image of the ER diagram and the descriptions of the classes and relations.

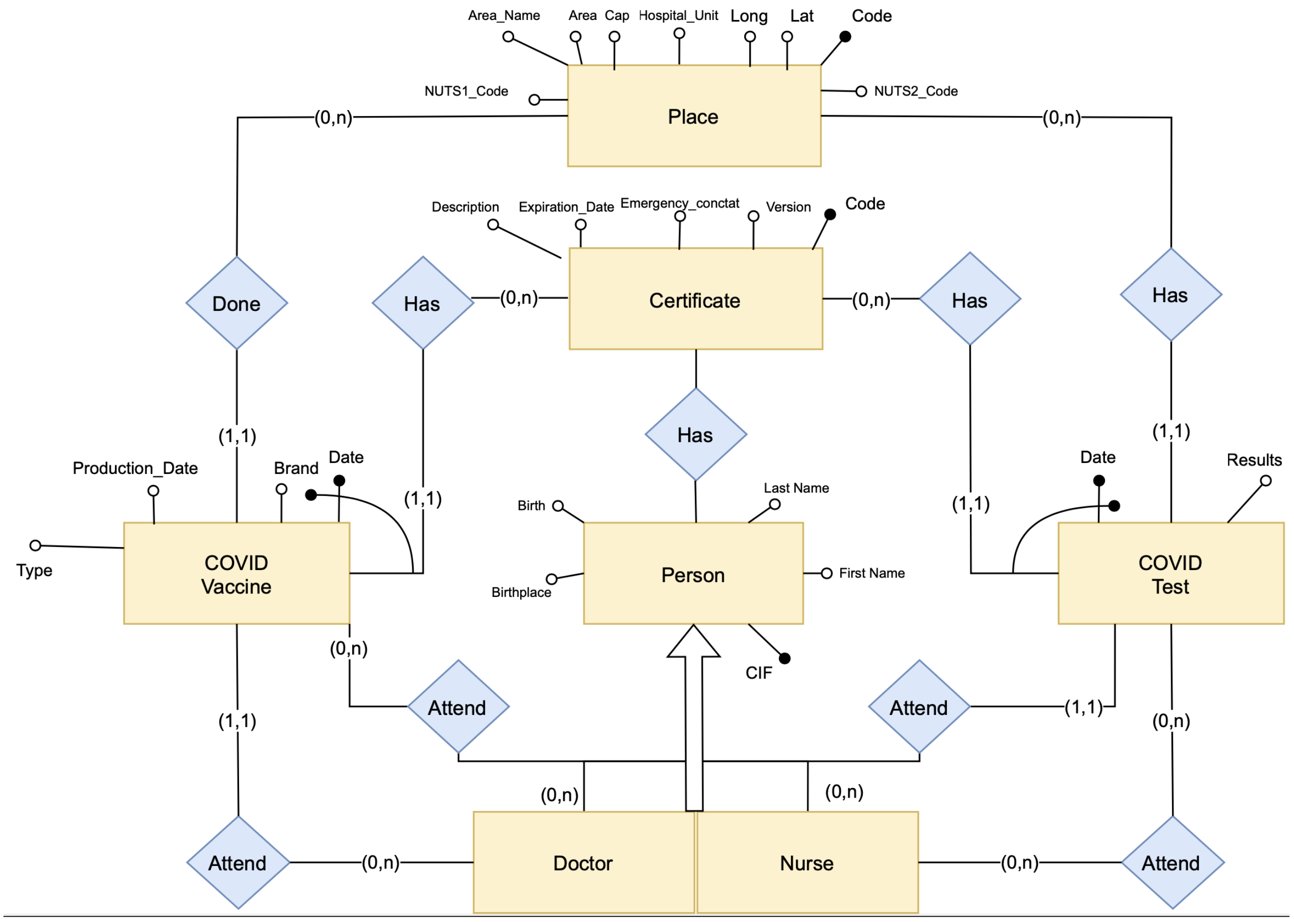


Figure 1: Entity-Relation Model

2.1 Logical Model

```
####### Entities #######

Person(CIF,First_Name,Last_Name,Birth,Birthplace)

Doctor(CIF,First_Name,Last_Name,Birth,Birthplace)

Nurse(CIF,First_Name,Last_Name,Birth,Birthplace)

Place(Code,Hospital_Unit,Lat,Long,CAP,Area,Area_name,NUTS1_Code,NUTS2_Code)

Tests(Code,Date,Results)

Vaccines(Code,Date,Brand,Type,Production_Date)

Certificate(Code,Version,Expiration_date,Emergency_Contact,Description)
```

3 MongoDB Implementation

To generate entities and relations, we coded a python script called "generate_dataset.py" that generates:

- 10000 Certificate
- 300 Places

Once that the script has been executed, 2 output json files can be found in scripts/output.

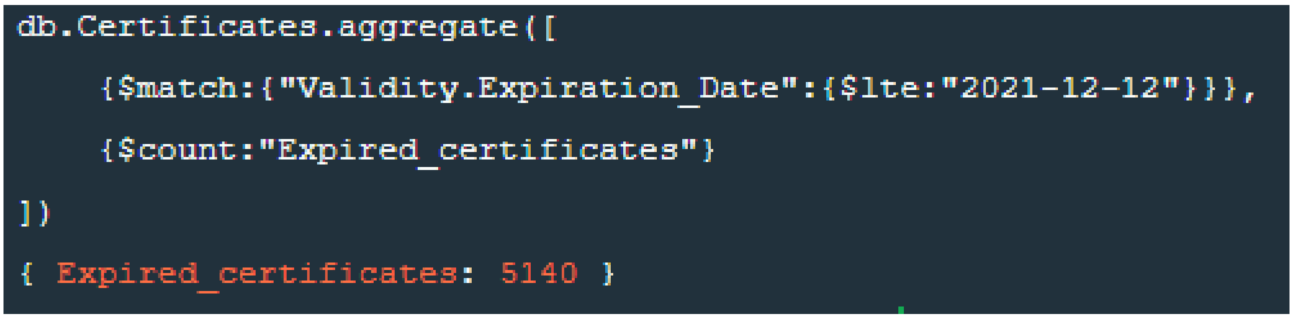
3.1 Queries

Now we list some queries we wrote to inspect our database for useful information about COVID.

Query n. 1

Count how many people have an expired certificate

```
db.Certificates.aggregate([
    { $match:{"Validity.Expiration_Date":{$lte:"2021-12-12" }}},
    { $count:"Expired_certificates" }
])
```

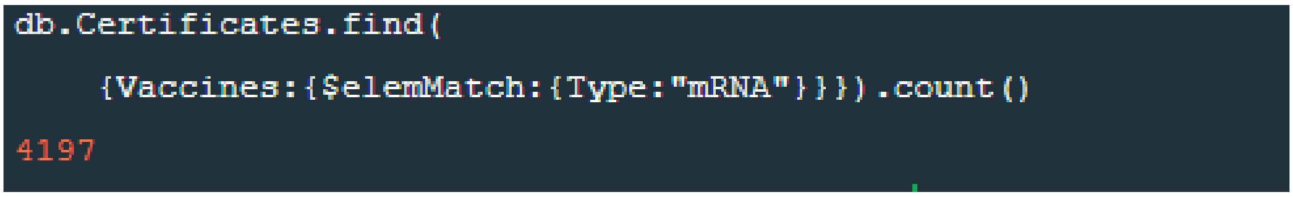


```
db.Certificates.aggregate([
    {$match:{"Validity.Expiration_Date":{$lte:"2021-12-12"}}},
    {$count:"Expired_certificates"}
])
{ Expired_certificates: 5140 }
```

Query n. 2

Count how many people did at least one mRna vaccine

```
db.Certificates.find(
    {Vaccines:{$elemMatch:{Type:"mRNA"}}}).count()
```

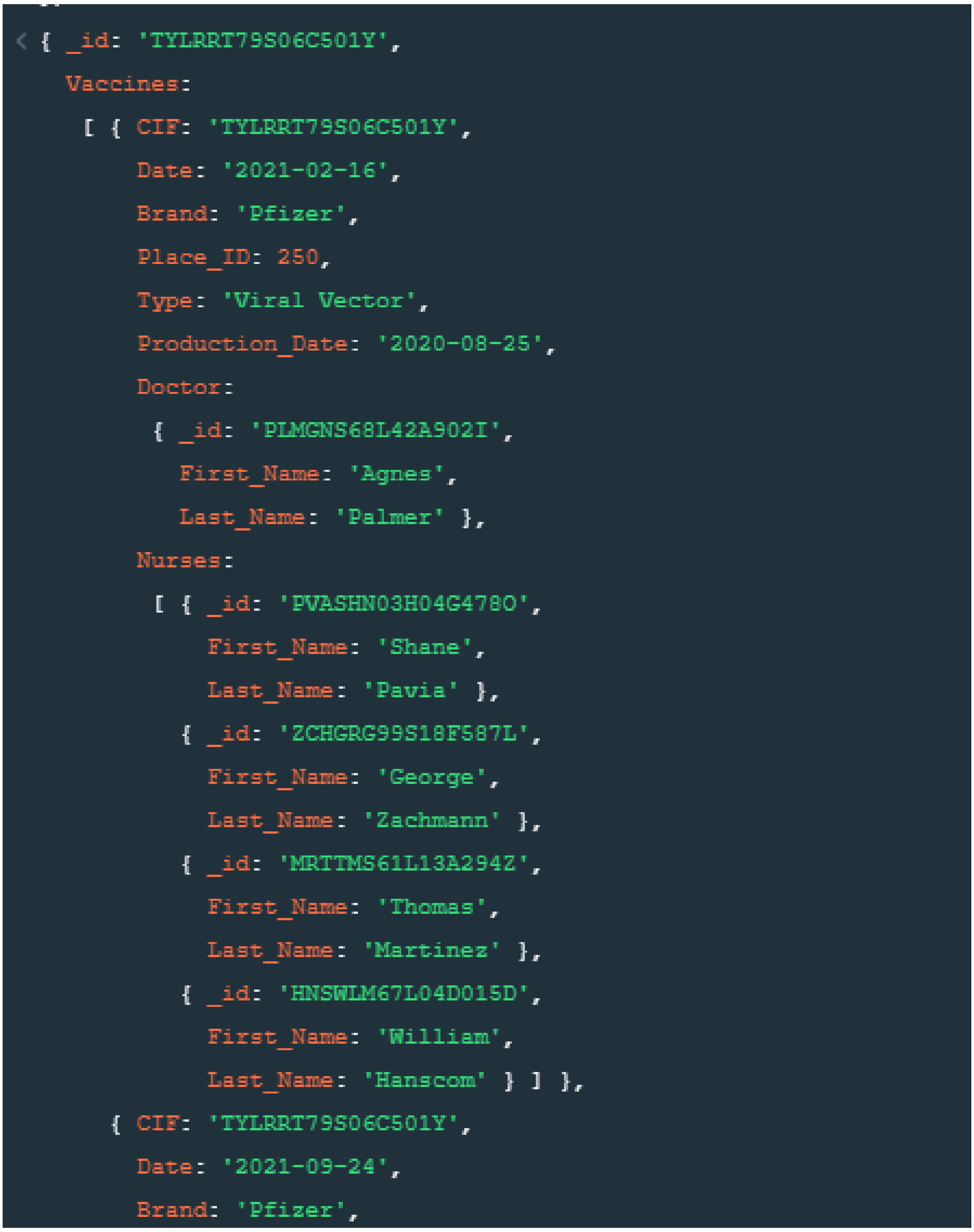


```
db.Certificates.find(
    {Vaccines:{$elemMatch:{Type:"mRNA"}}}).count()
4197
```

Query n. 3

Find the people who had two doses of Pfizer

```
db.Certificates.aggregate([
    {$project:{
                CIF:1,
                Vaccines: {$filter:{input:"$Vaccines", as:"v",
                cond:{$eq:["$$v.Brand", "Pfizer"]}}}}},
    {$match:{"Vaccines" :{$size:2}}}
])
```

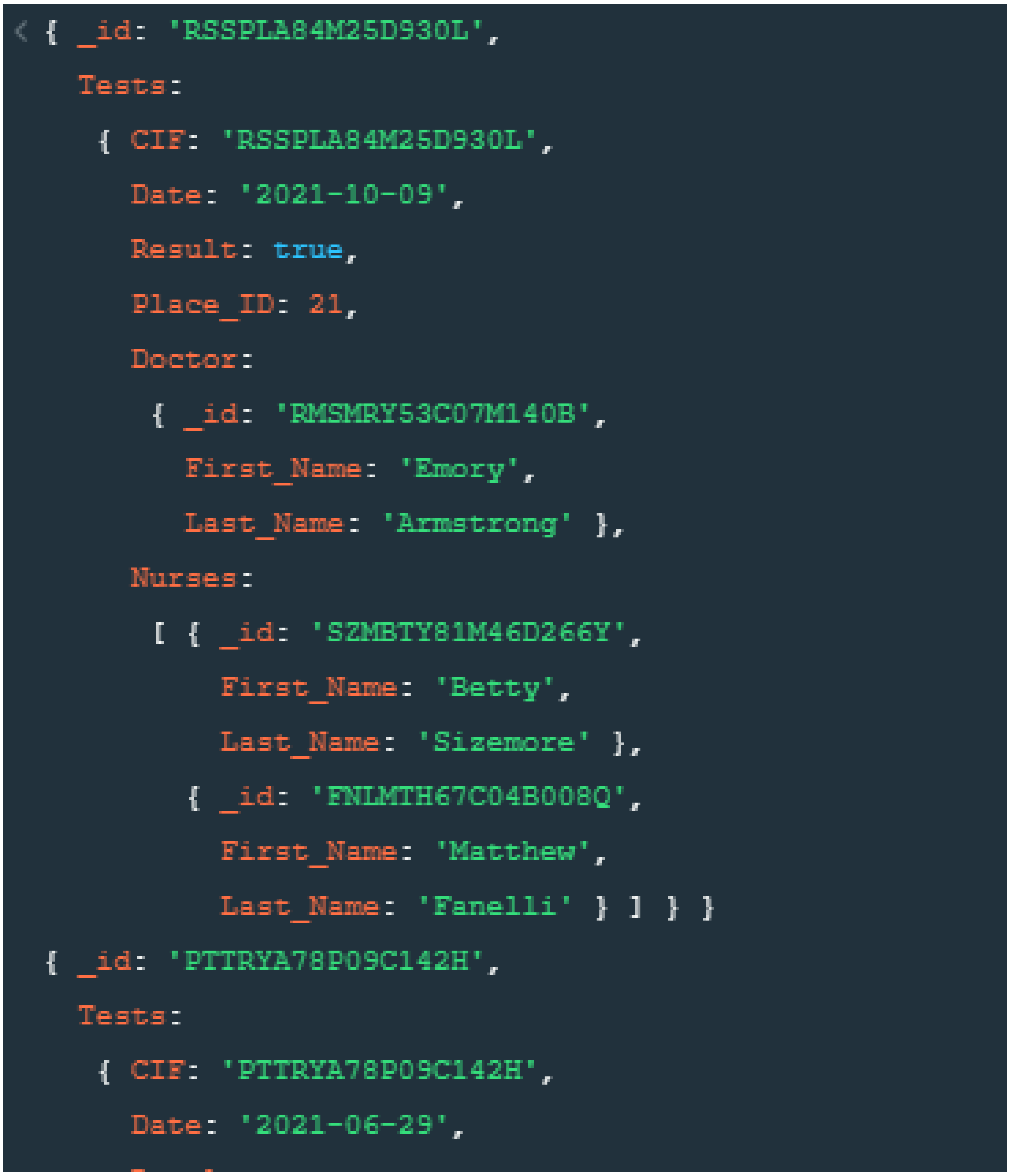


```
< { _id: 'TYLRRT79S06C501Y',
    Vaccines:
     [ { CIF: 'TYLRRT79S06C501Y',
         Date: '2021-02-16',
         Brand: 'Pfizer',
         Place_ID: 250,
         Type: 'Viral Vector',
         Production_Date: '2020-08-25',
         Doctor:
          { _id: 'PLMGNS68L42A902I',
            First_Name: 'Agnes',
            Last_Name: 'Palmer' },
         Nurses:
          [ { _id: 'PVASHN03H04G478O',
              First_Name: 'Shane',
              Last_Name: 'Pavia' },
            { _id: 'ZCHGRG99S18F587L',
              First_Name: 'George',
              Last_Name: 'Zachmann' },
            { _id: 'MRTTMS61L13A294Z',
              First_Name: 'Thomas',
              Last_Name: 'Martinez' },
            { _id: 'HNSWLM67L04D015D',
              First_Name: 'William',
              Last_Name: 'Hanscom' } ] },
       { CIF: 'TYLRRT79S06C501Y',
         Date: '2021-09-24',
         Brand: 'Pfizer',
```

Query n. 4

Find the people who had a positive result in the last test they did

```
db.Certificates.aggregate([
     {$project:{
                _id:1,
                Tests:{$reduce:{
                                input:"$Tests",
                                initialValue:{Date:"0000-00-00"},
                                in:{$cond:[{$gte:["$$this.date",
                                "$$value.date"]},
                                "$$this", "$$value"]}}}}},
     {$match:{"Tests.Result":true}}
])
```

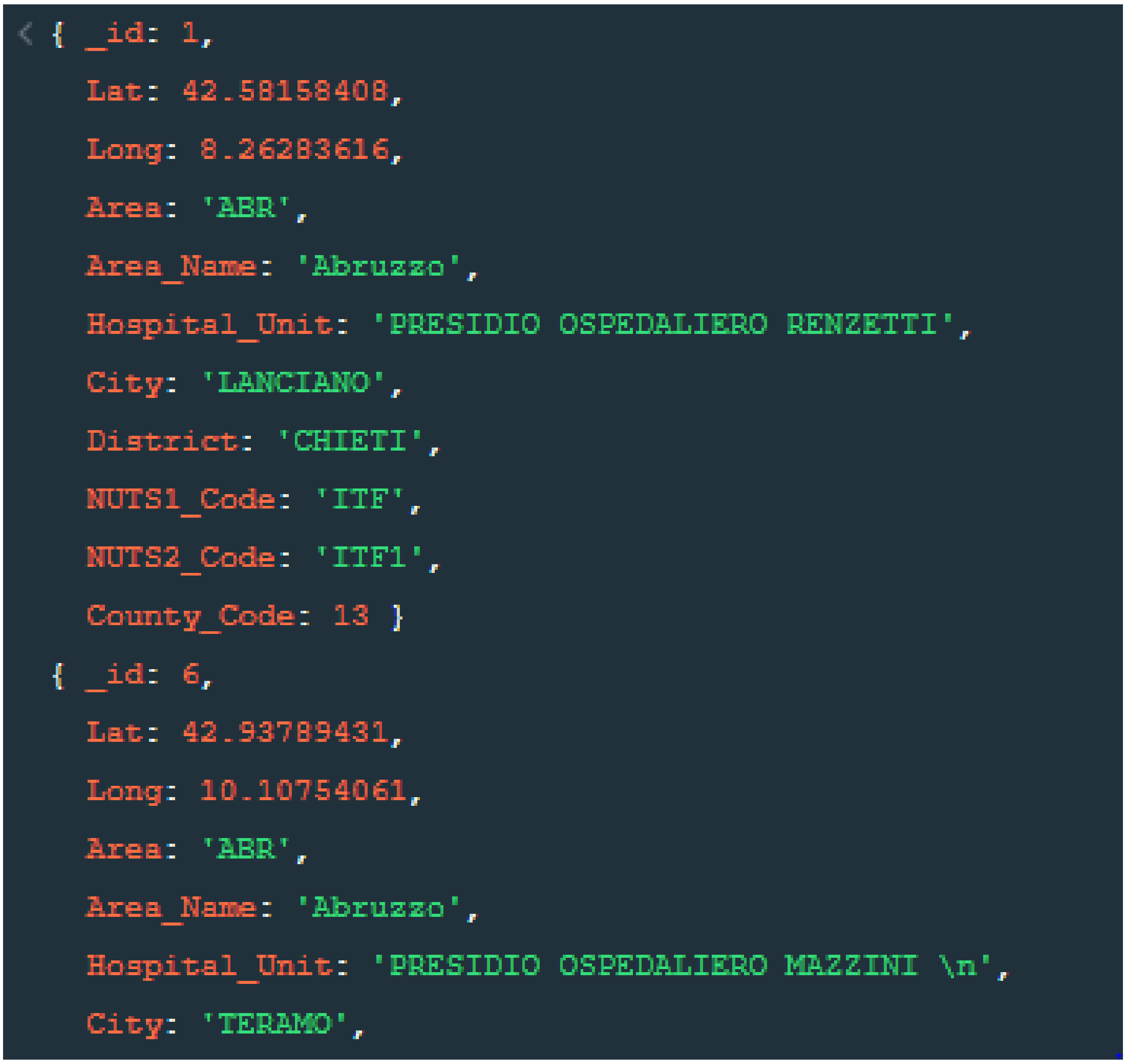


```
< { _id: 'RSSPLA84M25D930L',
    Tests:
     { CIF: 'RSSPLA84M25D930L',
       Date: '2021-10-09',
       Result: true,
       Place_ID: 21,
       Doctor:
        { _id: 'RMSMRY53C07M140B',
          First_Name: 'Emory',
          Last_Name: 'Armstrong' },
       Nurses:
        [ { _id: 'SZMBTY81M46D266Y',
            First_Name: 'Betty',
            Last_Name: 'Sizemore' },
          { _id: 'FNLMTH67C04B008Q',
            First_Name: 'Matthew',
            Last_Name: 'Fanelli' } ] } }
  { _id: 'PTTRYA78P09C142H',
    Tests:
     { CIF: 'PTTRYA78P09C142H',
       Date: '2021-06-29',
```

Query n. 5

Find all the hospitals

```
db.Places.find({
      $text:{"$search":"Presidio Ospedaliero"}
}).limit(50)
```

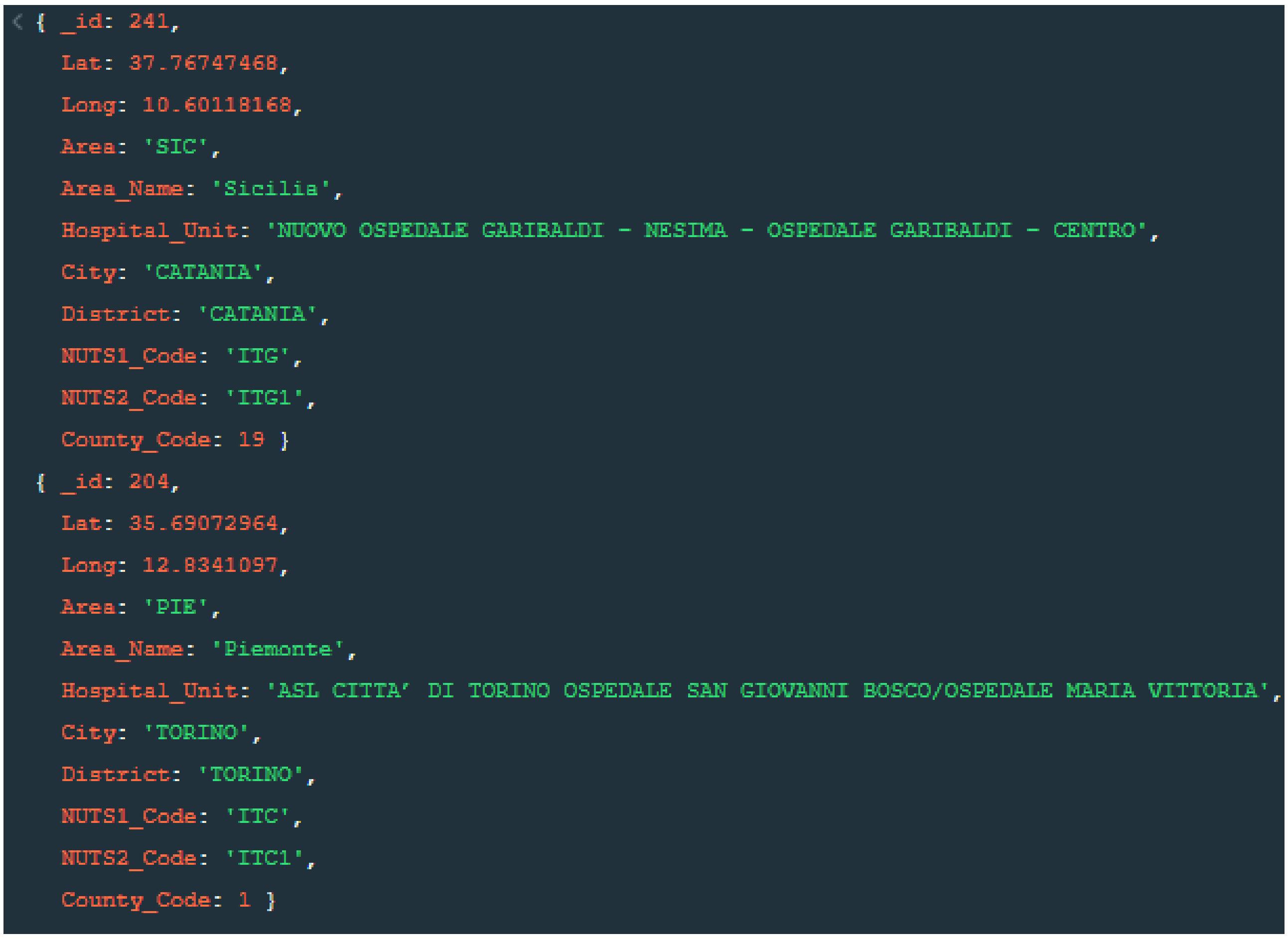


```
< { _id: 1,
    Lat: 42.58158408,
    Long: 8.26283616,
    Area: 'ABR',
    Area_Name: 'Abruzzo',
    Hospital_Unit: 'PRESIDIO OSPEDALIERO RENZETTI',
    City: 'LANCIANO',
    District: 'CHIETI',
    NUTS1_Code: 'ITF',
    NUTS2_Code: 'ITF1',
    County_Code: 13 }
  { _id: 6,
    Lat: 42.93789431,
    Long: 10.10754061,
    Area: 'ABR',
    Area_Name: 'Abruzzo',
    Hospital_Unit: 'PRESIDIO OSPEDALIERO MAZZINI \n',
    City: 'TERAMO',
```

Query n. 6

Find all the hospitals and order them with the text search score

```
db.Places.aggregate([
        {$match:{$text:{$search:"ospedale"}}}, {$sort:{score:{$meta:"textScore"}}}
])
```



```
< { _id: 241,
    Lat: 37.76747468,
    Long: 10.60118168,
    Area: 'SIC',
    Area_Name: 'Sicilia',
    Hospital_Unit: 'NUOVO OSPEDALE GARIBALDI - NESIMA - OSPEDALE GARIBALDI - CENTRO',
    City: 'CATANIA',
    District: 'CATANIA',
    NUTS1_Code: 'ITG',
    NUTS2_Code: 'ITG1',
    County_Code: 19 }
  { _id: 204,
    Lat: 35.69072964,
    Long: 12.8341097,
    Area: 'PIE',
    Area_Name: 'Piemonte',
    Hospital_Unit: 'ASL CITTA' DI TORINO OSPEDALE SAN GIOVANNI BOSCO/OSPEDALE MARIA VITTORIA',
    City: 'TORINO',
    District: 'TORINO',
    NUTS1_Code: 'ITC',
    NUTS2_Code: 'ITC1',
    County_Code: 1 }
```

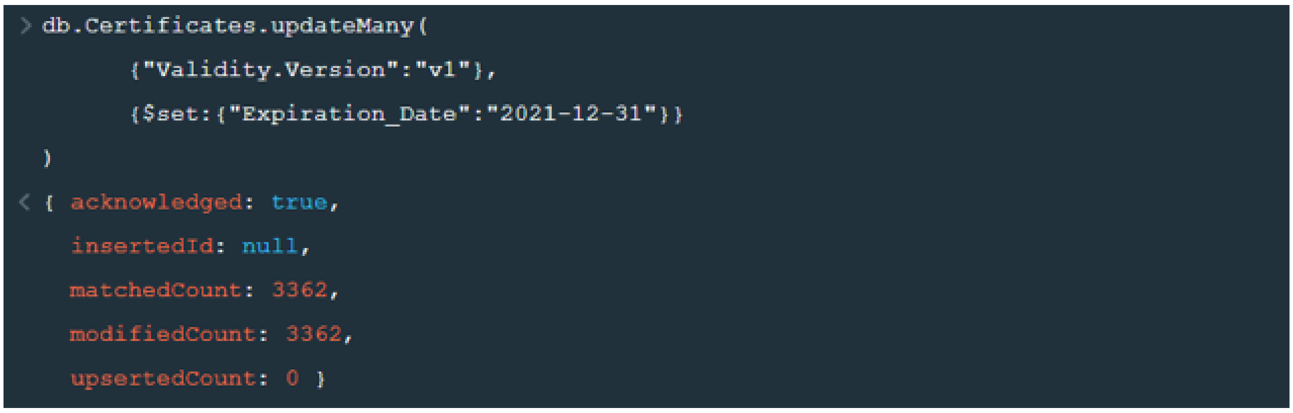
3.2 Commands

Below some commands that manipulates the database.

3.2.1 Command n. 1

For all certificates with version "v1", set the expiration date to: 31/12/2021

```
db.Certificates.updateMany(
      {"Validity.Version":"v1"},
      {$set:{"Expiration_Date":"2021-12-31"}}
)
```

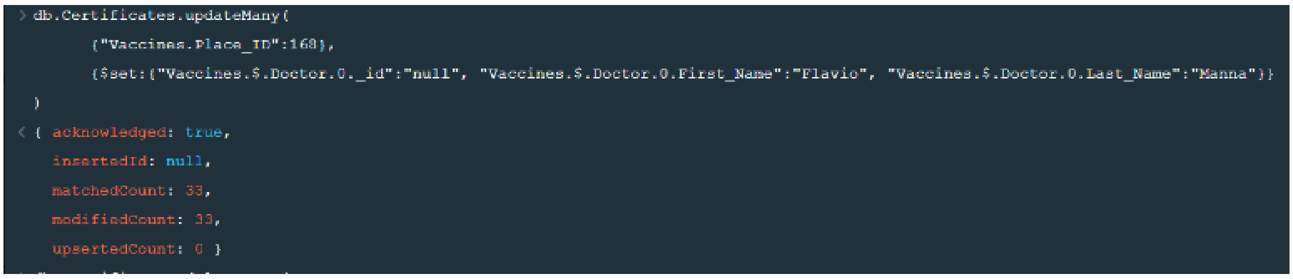


```
> db.Certificates.updateMany(
        {"Validity.Version":"v1"},
        {$set:{"Expiration_Date":"2021-12-31"}}
  )
< { acknowledged: true,
    insertedId: null,
    matchedCount: 3362,
    modifiedCount: 3362,
    upsertedCount: 0 }
```

3.2.2 Command n. 2

For all the people who did a vaccine in the place 168, set the doctor id to "null", doctor first name as "Flavio" and doctor last name as "Manna"

```
db.Certificates.updateMany(
      {"Vaccines.Place_ID":168},
      {$set:{"Vaccines.$.Doctor.0._id":"null",
      "Vaccines.$.Doctor.0.First_Name":"Flavio",
      "Vaccines.$.Doctor.0.Last_Name":"Manna"}}
)
```

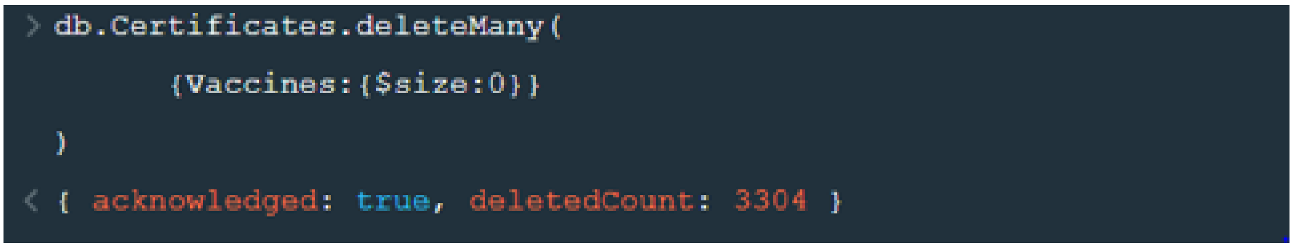


```
> db.Certificates.updateMany(
        {"Vaccines.Place_ID":168},
        {$set:{"Vaccines.$.Doctor.0._id":"null", "Vaccines.$.Doctor.0.First_Name":"Flavio", "Vaccines.$.Doctor.0.Last_Name":"Manna"}}
  )
< { acknowledged: true,
    insertedId: null,
    matchedCount: 33,
    modifiedCount: 33,
    upsertedCount: 0 }
```

3.2.3 Command n. 3

Delete all certificates without vaccine

```
db.Certificates.deleteMany(
        {Vaccines:{$size:0}}
)
```



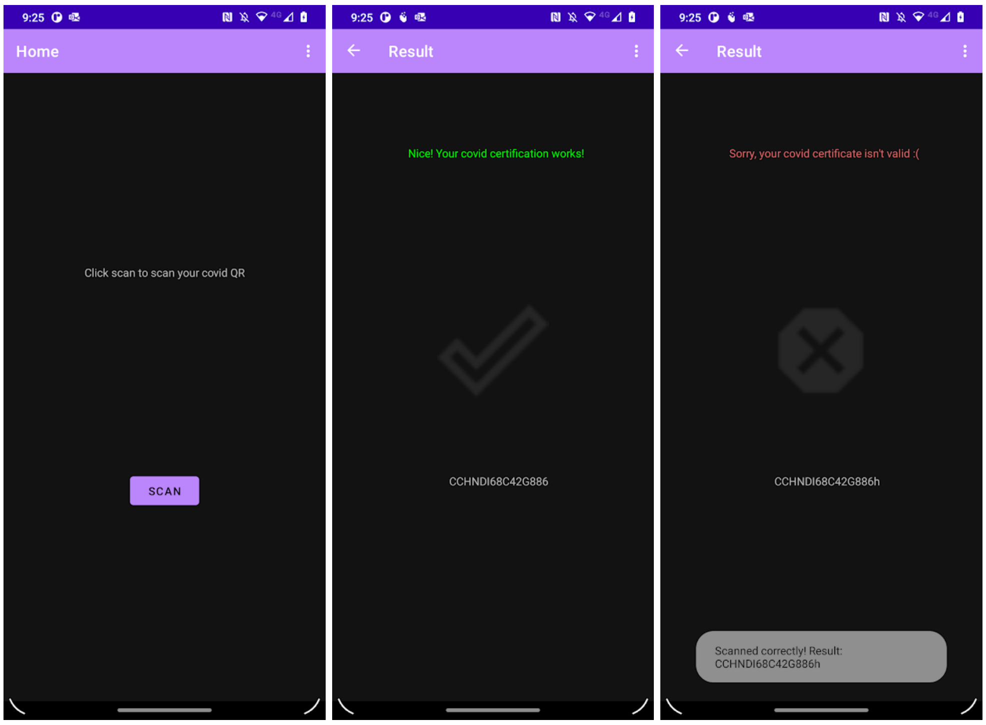
```
> db.Certificates.deleteMany(
        {Vaccines:{$size:0}}
  )
< { acknowledged: true, deletedCount: 3304 }
```

4 User Interface Implementation

For our User Interface, this time we decided to make something that while being less flashy could actually be useful, at the very least as a mock-version of a larger piece in a puzzle. At the same time, just like last time, we wanted to learn new skills, so this time we went for a native mobile app.
As we quickly learned, the MongoDB interaction for mobile apps isn't as easy as we thought; in fact, we spent most of our time actually reaching the database, instead of coding additional features or UX improvements. Due to this, the final result is very... feature-oriented, as opposed to design focused.

4.1 Screens and UX

The app has two simple screens: on the first one, the user taps "scan" to scan the QR code; then, the user is moved to a scanning page, where it can access a viewfinder and scan a QR code. Once the QR code is scanned, a result page is loaded, with either a checkmark or a cross, and some text explaining the result.



Click scan to scan your covid QR

Nice! Your covid certification works!

CCHNDI68C42G886

Sorry, your covid certificate isn't valid :(

CCHNDI68C42G886h

Scanned correctly! Result:
CCHNDI68C42G886h

4.2 Connecting to the database and querying

Behind the scenes, we tested various ways to connect to the database: first, we used a complete library called KMongo; unfortunately, the android JVM lacks some fundamental features that impede the correct functioning of any driver: we leave this link to learn more (it helped us a ton!). Then we moved to MongoDB's own solution: their driver developed in-house. Unfortunately, that didn't go well either: the driver relies on an additional layer of abstraction called MongoDB Realm, which, although it adds some very advanced functionality such as syncing multiple clients with conflict resolution, has a complicated documentation and didn't play well with our "simple and straight" mentality. Confused and afraid, we tried a different solution to check if our database was really reachable at all: a simple JS script proved how easy it was to use the nodeJS driver for MongoDB. Due to this, we removed all our complicated query-response logic from the app, and turned it into a simple GET request for a middleware we set up on our laptop. This worked remarkably well: we now understand why the normal way of using MongoDB is implementing a similar kind of middleware.

4.3 The app itself

Overall, the app was simplified with our move to the NodeJS middleware. It now features a single activity, with two fragments which represented the home and the result screen. The ViewModel is shared across the whole app, since no data hiding is required. For our single GET request we used volley, and used data binding to fill in our xml fields. Kotline revealed itself as a very complete language; some design choices require some getting used to, but we felt its general "vibe" is more modern than Java.

5 Conclusion and possible improvements

In this report, we outlined the design and implementative choices behind the second part of the SAMBUD Fall 2021 project at Polimi. The project highlights the potential of MongoDB, a documents based NoSQL DB, as a modern way of storing and managing data compared to normal databases. From the practical point of view it has been interesting to see how the complexity of the ER model has become such a lean structure, highlighting the advantages of working on documents. MongoDB allows for convenient data management, the queries were easy to understand and mirrored the underlying structure. This project has broadened our understanding and vision of NoSQl DBc, using a storage structure that is inspired by the one of the real world allowing us to move away from the classic world of sql.