

PROJECT WORK - MONGODB INFORMATION SYSTEM PROF. MARCO
BRAMBILLA

Systems and Methods for Big and Unstructured Data



POLITECNICO
MILANO 1863

Curti Gabriele [10624502]
Cutrupi Lorenzo [10629494]
Samuele Mariani [10622653]
Alessandro Molteni [10623928]
Matteo Monti [10622780]

December 11, 2021

Contents

1	Introduction	2
1.1	Problem specifications	2
1.2	Hypothesis	3
2	ER Diagram	4
3	Dataset description and creation	5
4	Queries and Commands	9
4.1	Queries	9
4.1.1	Given a person_id, check if the corresponding person has the green pass	9
4.1.2	Return the number of people that have the super green pass	10
4.1.3	Number of people that got vaccinated in a specific day	10
4.1.4	Name and surname of people vaccinated by a certain doctor	11
4.1.5	Name and surname of people who got vaccinated in a specific vaccination center with a certain vaccine	12
4.1.6	Name and surname of people who got vaccinated in center "c4ceUtvT"	13
4.1.7	Number of people who got the third dose	13
4.2	Commands	14
4.2.1	Update the tests field of a specific person with a new test	14
4.2.2	Elimination of a certificate from the DataBase	15
4.2.3	Modify the phone number of a person emergency contact	15
5	UI Application	16
5.1	Connecting the application to the DataBase	16
5.2	Perform queries	17
6	Team composition	19
7	Sources	19

1 Introduction

The project is about designing, storing and querying a database using technologies shown during the lessons. In this case we were asked to build a MongoDB database to support the storing, retrieving and checking of Covid-19 certificates and information about facilities authorized to issue vaccinations and Covid-19 tests.

1.1 Problem specifications

Information that needs to be stored in each certificate:

- Personal data for each person
 - Name and surname
 - Birthdate
 - Other relevant personal information
- Vaccination data
 - Brand
 - Lot information
 - Somministration date
 - Somministration place
 - Doctors and nurses that where present during the vaccination
- Covid tests data
 - Test type
 - Where the test was taken
 - Test date
 - Doctor and nurses present during the procedure
- Emergency contact details
 - Name and surname
 - Phone number

The DataBase has to include information about the authorized centers delivering the vaccines and the tests:

- Name of the center
- Address
- Type of entity
- Department issuing the test/vaccine

1.2 Hypothesis

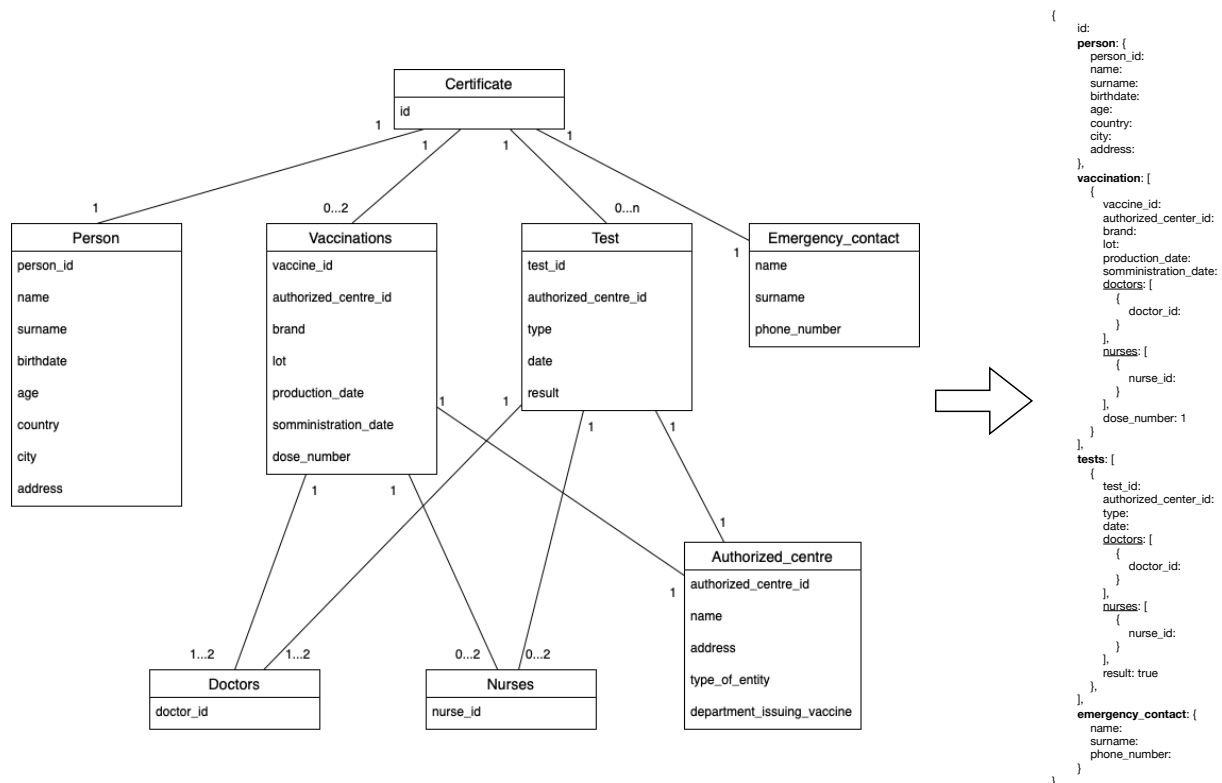
We made some hypothesis to simplify the construction of the database without restricting its capabilities:

Hypothesis	Description
H.1	The chosen number of keys included in the DataBase is small (only 100 entries) but can easily be expanded in the future.
H.2	The certificate validity-check is external from the database. To simplify the writing operations only the date in which a vaccine/test was taken is stored in the certificate, relying on the query to check the validity.
H.3	The certificate is considered expired after 9 month have passed after the last vaccination dose, or (in case of absent or expired vaccination) 72 hours after the last negative test.
H.4	In the case of the super certificate check, we only check the state of the last vaccine.
H.5	Only people with 2 vaccine doses at most have been considered while populating the dataset, but the database can easily be extended to any number of doses.

2 ER Diagram

The following diagram represent the internal structure of the database documents. Each document models a different certificate.

Each Certificates consists of four major parts: the **Person** field, the **Vaccination** field, the **Test** field and the **Emergency Contact** field. Two additional classes are present: **Doctors** and **Nurses**, which contains the personal IDs of the doctors and nurses that where present during the vaccination or testing procedure.



A similar document structure is used to model the different **Authorized centers**.

3 Dataset description and creation

All the raw data for the DataBase population process concerning people personal information was generated using the website [fakenamegenerator.com](https://www.fakenamegenerator.com) which allows the generation of massive amount of data in .csv format. Using the website option **Order in Bulk** it was possible to generate a .csv file containing 200 different entries (only 100 entries were uploaded as stated in hypothesis **H.1**), each characterized by 8 fields: Name, Surname, City, StreetAddress, Country, Birthdate, Age and TelephoneNumber.

In order to complete the dataset with the remaining certificates fields and the authorized centers data, the .csv file was then converted to a .json using the following python script:

The initial part of the code defines some lists of vaccine and tests information that will be randomly selected and included in the certificates. Useful Authorized centers information is also manually defined (only three out of ten centers are included for illustrative purposes):

```

1  from collections import OrderedDict
2  import json
3  import random, string
4  import datetime
5  import csv
6
7
8  personCounter = 0
9  vaccineBrand = ['Pfizer-BioNTech', 'Moderna', 'Johnson & Johnson Janssen', 'Sputnik V']
10 testType = ['Antigen tests', 'Molecular/PCR tests', 'Antibody']
11 authorized_centers_ids = ['c4ceUtvT', 'HPrj94aa', 'omKWu98X', 'ZvTJzMvG', 'oEEMSkZ3', 'PRfcJsn6', 'ieSviq2u',
12 'KtvGsZmf', 'VAk5kZ65', 'HfdsCoq2']
13 authorized_centers = [
14     {
15         'id' : 'c4ceUtvT',
16         'name' : 'Policlinico di Modena',
17         'address' : 'Via del Pozzo, 71, 41125 Modena MO',
18         'type' : 'Policlinico',
19         'department' : 'COVID-19 Unit'
20     },
21     {
22         'id' : 'HPrj94aa',
23         'name' : 'Centro MICI -Ospedale Sandro Pertini',
24         'address' : 'Via dei Monti Tiburtini, 385, 00157 Roma RM',
25         'type' : 'Ospedale',
26         'department' : 'Reparto di pronto soccorso'
27     },
28     {
29         'id' : 'omKWu98X',
30         'name' : 'Humanitas Research Hospital',
31         'address' : 'Via Alessandro Manzoni, 56, 20089 Rozzano MI',
32         'type' : 'Research Hospital',
33         'department' : 'COVID-19 Unit'
34     },
35     [...]
36 ]

```

The following function is used to parse the .csv file containing the people raw data mentioned above, and return a list of dictionaries containing the rows of the .csv file:

```

1 def csvParser():
2     with open('certificationRawData.csv', 'r') as file:
3         csv_file = csv.DictReader(file)
4         peopleDataRaw = {}
5         peopleDataRaw['entries'] = []
6         for row in csv_file:
7             peopleDataRaw['entries'].append(dict(row))
8     return peopleDataRaw
9
10 peopleDataRaw = csvParser()

```

In order to generate the random fields of the certificate such as: production_date, somministration_date, vaccines and tests ids, certificate ids, centers ids and vaccine lot, the following functions were created (only two are included for illustrative purposes):

```

1 #Random date generator for the vaccine production date
2 def randomDate():
3     start_date = datetime.datetime(2021, 1, 1)
4     end_date = datetime.datetime(2021, 11, 1)
5
6     time_between_dates = end_date - start_date
7     days_between_dates = time_between_dates.days
8     random_number_of_days = random.randrange(days_between_dates)
9     random_date = start_date + datetime.timedelta(days=random_number_of_days)
10
11     return random_date
12
13
14 #Random alphanumeric 16-digits code generator for vaccines Ids and for tests Ids
15 def randomId():
16     randomId = ''.join(random.choice(string.ascii_lowercase + string.digits) for _ in range(16))
17     return randomId

```

The person, vaccinations, tests, Emergency_contact, doctors and nurses document fields are all generated as dictionaries with randomized fields (below only the functions to generate the person field and the vaccinations field are included for illustrative purposes. The certificate and all the other remaining elements are generated in a similar manner):

```

1 # Generates a random person
2 def generatePerson(personCounter):
3     person = OrderedDict()
4     person['person_id'] = personCounter
5     person['name'] = peopleDataRaw['entries'][personCounter]['GivenName']
6     person['surname'] = peopleDataRaw['entries'][personCounter]['Surname']
7     person['birthdate'] = peopleDataRaw['entries'][personCounter]['Birthday']
8     person['age'] = peopleDataRaw['entries'][personCounter]['Age']
9     person['country'] = peopleDataRaw['entries'][personCounter]['CountryFull']
10    person['city'] = peopleDataRaw['entries'][personCounter]['City']
11    person['address'] = peopleDataRaw['entries'][personCounter]['StreetAddress']
12    return person
13

```

```

14
15 # Each person can have between 0 and 2 vaccinations
16 def generateVaccination(vaccineIndex):
17     vaccination = OrderedDict()
18     vaccination['vaccine_id'] = randomId()
19     vaccination['authorized_center_id'] = random.choice(authorized_centers_ids)
20     vaccination['brand'] = random.choice(vaccineBrand)
21     vaccination['lot'] = randomLot()
22     productionDate = randomDate()
23     vaccination['production_date'] = productionDate.strftime("%Y-%m-%d")
24     vaccination['somministration_date'] = randomSomministrationDate(productionDate).strftime("%Y-%m-%d")
25     vaccination['doctors'] = []
26     for index in range(random.randint(1, 2)):
27         vaccination['doctors'].append(generateDoctor())
28     vaccination['nurses'] = []
29     for index in range(random.randint(1, 2)):
30         vaccination['nurses'].append(generateNurse())
31     vaccination['dose_number'] = vaccineIndex
32     return vaccination

```

A file named `certificatesData.json` is created using the `json.dump()` function:

```

1 with open('certificatesData.json', 'w') as outfile:
2     for index in range(100):
3         certificateDocument = OrderedDict()
4         certificateDocument['certificate'] = generateCertificate()
5         personCounter += 1
6         json.dump(certificateDocument, outfile, ensure_ascii=False, indent=4)
7
8     for index in range(10):
9         centerDocument = OrderedDict()
10        centerDocument['authorized_center'] = generateCenter(index)
11        json.dump(centerDocument, outfile, ensure_ascii=False, indent=4)

```

An additional code section was created to make the data type fields of the `.json` file, compatible with the data format used by **MongoDB Compass**:

```

1 with open('certificatesData.json', 'r') as outfile:
2     line = outfile.readline()
3     finalOut = ""
4     while (line):
5         first = line.split(':')
6         if (first[0] == "somministration_date\" or
7             first[0] == "production_date\" or
8             first[0] == "date\" or
9             first[0] == "birthdate\" ):
10            print(first[1])
11            second = "{\"$date\":\"" + first[1].split(',')[0] + "\",\n"
12            print(second)
13            finalOut = finalOut + first[0] + ":" + second
14        else: finalOut = finalOut + line
15        line = outfile.readline()
16
17
18

```



```

19  #open text file
20  text_file = open("certificatesData.json", "w")
21
22  #write string to file
23  text_file.write(finalOut)
24
25  #close file
26  text_file.close()

```

Finally the generated `certificatesData.json` file is uploaded in the *CovidCertificationData* DataBase inside the *Certificate* collection using the MongoDB Compass function *ADD DATA*.

The screenshot shows the MongoDB Compass interface for the `CovidCertificationData.Certificates` collection. The top bar indicates 109 documents with a total size of 91.5KB and an average size of 839B. Below the bar, the 'Documents' tab is active, showing a filter bar with the filter `{ field: 'value' }` and buttons for 'FIND', 'RESET', and 'OPTIONS'. The main area displays a list of documents, with the first one expanded to show its structure:

```

{
  "_id": ObjectId("61a9f0d693194739d385d1da"),
  "id": "46s7zbsm8bpygg65sr74snxx",
  "person": {
    "person_id": 0,
    "name": "Vitale",
    "surname": "Cremonesi",
    "birthdate": 1997-07-08T22:00:00.000+00:00,
    "age": "24",
    "country": "Italy",
    "city": "Casoni",
    "address": "Lungodora Napoli 92"
  },
  "vaccination": Array,
  "tests": Array,
  "emergency_contact": {
    "name": "Viola",
    "surname": "Rossi",
    "phone_number": "0346 4403095"
  }
}

```

Below this, two more documents are partially visible, showing their `_id`, `id`, and `person` fields.

The complete version of the above code, the final `.json` data file and the `.csv` file containing the people random generated data, are available in the project delivery folder.

4 Queries and Commands

In order to retrieve useful data from the dataset, from both user perspective and big data analysis perspective, some simple queries were designed with the intent of simulating some of the basic operations that such DataBase could be used for. Additionally three commands were designed with the intent of demonstrating how the database could be modified and updated.

All the stated queries and commands have been created using MongoDB Compass included shell.

4.1 Queries

4.1.1 Given a person_id, check if the corresponding person has the green pass

```
db.Certificates.find(
  {"$and" : [
    {"person.person_id" : 2},
    {"$or":[{"$and":[{"tests.date": {"$gt": new
      Date(new Date().getTime() - 1000*3600*72)}},{"tests.result":
      "false"}]},
    {"vaccination.somministration_date": {"$gt": new Date(new
      Date().getTime() - 1000*3600*24*275)}}]}
  ]},{'person.name': 1,'person.surname': 1}
)
```

In order to slow down the epidemic spread, facilities such as museums, cinemas and restaurants must check if their customers have the green pass. Given a person_id, this query fulfills this task by checking whether the corresponding person has been given a vaccine dose no later than 275 days ago (approximately 9 months) or if the subject underwent a swab with negative result within the last 72 hours.

```
> db.Certificates.find(
  {"$and" : [
    {"person.person_id" : 2},
    {"$or":[{"$and":[{"tests.date": {"$gt": new Date(new Date().getTime() - 1000*3600*72)}},{"tests.result": "false"}]},
    {"vaccination.somministration_date": {"$gt": new Date(new Date().getTime() - 1000*3600*24*275)}}]}
  ]},{'person.name': 1,'person.surname': 1}
)
< { _id: ObjectId("61a9f0d693194739d385d1dc"),
  person: { name: 'Aida', surname: 'Pagnotto' } }
```

4.1.2 Return the number of people that have the super green pass

```
db.Certificates.countDocuments(  
  {"vaccination.somministration_date": {"$gt": new  
    Date(ISODate().getTime() - 1000*3600*24*275)}}  
)
```

In order to check the effectiveness of the last legal provisions, the government may need to know how many people are getting vaccinated and, consequently, how many people have been given the super green pass, which is valid if a person has received a vaccine dose no later than 275 days ago (approximately 9 months).

```
> db.Certificates.countDocuments(  
  {"vaccination.somministration_date": {"$gt": new Date(ISODate().getTime() - 1000*3600*24*275)}}  
)  
< 50
```

4.1.3 Number of people that got vaccinated in a specific day

```
db.Certificates.countDocuments({  
  "vaccination.somministration_date": ISODate("2021-08-04")  
)
```

Simple query but very useful to check the evolution of the vaccinal campaign.

```
> db.Certificates.countDocuments({  
  "vaccination.somministration_date": ISODate("2021-08-04")  
})  
< 1
```

4.1.4 Name and surname of people vaccinated by a certain doctor

```
db.Certificates.find(  
  {'vaccination.doctors.doctor_id' : 1013},  
  {'person.name': 1, 'person.surname': 1}  
)
```

This query retrieves the people who got vaccinated by the medic with id 1013.

```
> db.Certificates.find(  
  {'vaccination.doctors.doctor_id' : 1013},  
  {'person.name': 1, 'person.surname': 1}  
)  
< { _id: ObjectId("61a9f0d693194739d385d1fb"),  
    person: { name: 'Lisandro', surname: 'Fallaci' } }  
{ _id: ObjectId("61a9f0d693194739d385d20e"),  
  person: { name: 'Deborah', surname: 'Greece' } }  
{ _id: ObjectId("61a9f0d693194739d385d216"),  
  person: { name: 'Verdiana', surname: 'Dellucci' } }  
{ _id: ObjectId("61a9f0d693194739d385d21e"),  
  person: { name: 'Gilberto', surname: 'Lori' } }  
{ _id: ObjectId("61a9f0d693194739d385d22b"),  
  person: { name: 'Prospero', surname: 'Gallo' } }  
{ _id: ObjectId("61a9f0d693194739d385d22e"),  
  person: { name: 'Marino', surname: 'Cremonesi' } }  
{ _id: ObjectId("61a9f0d693194739d385d239"),  
  person: { name: 'Eduardo', surname: 'Ricci' } }  
{ _id: ObjectId("61a9f0d693194739d385d23a"),  
  person: { name: 'Tecla', surname: 'Conti' } }  
{ _id: ObjectId("61a9f0d693194739d385d23c"),  
  person: { name: 'Alba', surname: 'Lorenzo' } }
```

4.1.5 Name and surname of people who got vaccinated in a specific vaccination center with a certain vaccine

```
db.Certificates.find(
  {"$and": [
    {"vaccination.authorized_center_id" : "c4ceUtvT"},
    {"vaccination.brand": 'Johnson & Johnson Janssen'}]},
  {"person.name": 1, "person.surname":1}
)
```

Data analysis has shown that the 'Johnson & Johnson Janssen' vaccine provided to the vaccination center with center_id c4ceUtvT was not effective due to bad conservation. People who got vaccinated in these particular conditions need to be given an extra dose to achieve immunity.

```
> db.Certificates.find(
  {"$and": [
    {"vaccination.authorized_center_id" : "c4ceUtvT"},
    {"vaccination.brand": 'Johnson & Johnson Janssen'}
  ]},
  {"person.name": 1, "person.surname":1}
)
< { _id: ObjectId("61a9f0d693194739d385d1e7"),
  person: { name: 'Iacopo', surname: 'Pinto' } }
{ _id: ObjectId("61a9f0d693194739d385d1f2"),
  person: { name: 'Nestore', surname: 'Romano' } }
{ _id: ObjectId("61a9f0d693194739d385d1f6"),
  person: { name: 'Raul', surname: 'Gallo' } }
{ _id: ObjectId("61a9f0d693194739d385d201"),
  person: { name: 'Cristina', surname: 'Schiavone' } }
{ _id: ObjectId("61a9f0d693194739d385d229"),
  person: { name: 'Maia', surname: 'Moretti' } }
```

4.1.6 Name and surname of people who got vaccinated in center "c4ceUtvT"

```
db.Certificates.find(
  {"vaccination.authorized_center_id" : "c4ceUtvT"},
  {"person.name": 1, "person.surname":1}
)
```

In order to rank the most efficient vaccination centers, the government needs to know all the people who got vaccinated in a specific vaccination center.

```
> db.Certificates.find(
  {"vaccination.authorized_center_id" : "c4ceUtvT"},
  {"person.name": 1, "person.surname":1}
)
< { _id: ObjectId("61a9f0d693194739d385d1dd"),
  person: { name: 'Verdiana', surname: 'Lorenzo' } }
{ _id: ObjectId("61a9f0d693194739d385d1e7"),
  person: { name: 'Iacopo', surname: 'Pinto' } }
{ _id: ObjectId("61a9f0d693194739d385d1f2"),
  person: { name: 'Nestore', surname: 'Romano' } }
{ _id: ObjectId("61a9f0d693194739d385d1f6"),
  person: { name: 'Raul', surname: 'Gallo' } }
{ _id: ObjectId("61a9f0d693194739d385d1fc"),
  person: { name: 'Luigi', surname: 'Piccio' } }
{ _id: ObjectId("61a9f0d693194739d385d201"),
  person: { name: 'Cristina', surname: 'Schiavone' } }
{ _id: ObjectId("61a9f0d693194739d385d21a"),
  person: { name: 'Mattia', surname: 'Napolitani' } }
{ _id: ObjectId("61a9f0d693194739d385d229"),
  person: { name: 'Maia', surname: 'Moretti' } }
{ _id: ObjectId("61a9f0d693194739d385d22c"),
  person: { name: 'Lamberto', surname: 'Costa' } }
```

4.1.7 Number of people who got the third dose

```
db.Certificates.countDocuments({'vaccination': {$size : 3}})
```

Useful query in case the government needs to know how many people are continuing the vaccination cycle. This query simply returns zero due to the design choices taken during the DataBase population phase, but it has been included just as an example in perspective of future developments.

4.2 Commands

4.2.1 Update the tests field of a specific person with a new test

```

db.Certificates.updateOne(
  { "person.person_id": 4},
  { $push : { tests : {
    test_id : "9jkghty87fs45cv",
    authorized_center_id : "ieSviq2u",
    type : "Molecular/PCR tests",
    date : ISODate("2021-10-21"),
    doctors : [
      {
        doctor_id : 1014
      }
    ],
    nurses : [
      {
        nurse_id : 1964
      }
    ],
    result : "false"
  }}
})

```

When entered in the console this command updates the test field of the person with id = 4 that initially had only two tests with a third test that has the specified properties.

```

_id: ObjectId("61a9f0d693194739d385d1de")
id: "7rha0ajbdnp02bh8pkteomm"
person: Object
  person_id: 4
  name: "Cinzia"
  surname: "Baresi"
  birthdate: 1986-10-23T23:00:00.000+00:00
  age: "35"
  country: "Italy"
  city: "Collobiano"
  address: "Piazza San Carlo 122"
  vaccination: Array
  tests: Array
    0: Object
    1: Object
  emergency_contact: Object

```

Figure 1: Person with id = 4 **before** the execution of the command

```

_id: ObjectId("61a9f0d693194739d385d1de")
id: "7rha0ajbdnp02bh8pkteomm"
person: Object
  person_id: 4
  name: "Cinzia"
  surname: "Baresi"
  birthdate: 1986-10-23T23:00:00.000+00:00
  age: "35"
  country: "Italy"
  city: "Collobiano"
  address: "Piazza San Carlo 122"
  vaccination: Array
  tests: Array
    0: Object
    1: Object
    2: Object
      test_id: "9jkghty87fs45cv"
      authorized_center_id: "ieSviq2u"
      type: "Molecular/PCR tests"
      date: 2021-10-21T00:00:00.000+00:00
  doctors: Array
    0: Object
      doctor_id: 1014
  nurses: Array
    0: Object
      nurse_id: 1964
      result: "false"
  emergency_contact: Object

```

Figure 2: Person with id = 4 **after** the execution of the command

4.2.2 Elimination of a certificate from the DataBase

```
db.Certificates.deleteOne({ "person.name" : "Vitale", "person.surname" : "Cremonesi" })
```

This command will simply eliminate the certificate belonging to "Vitale Cremonesi" from the DataBase.

4.2.3 Modify the phone number of a person emergency contact

```
db.Certificates.update(
  { "id" : "46s7zbsm8bpygg65sr74snxx" },
  { $set: {
    "emergency_contact.phone_number" : "0528 1949992"
  }
})
```

This command will modify the `phone_number` field of the `emergency_contact` of the certificate that has `id` equal to the one specified.

```
_id: ObjectId("61a9f0d693194739d385d1da")
id: "46s7zbsm8bpygg65sr74snxx"
> person: Object
~ vaccination: Array
> tests: Array
~ emergency_contact: Object
  name: "Viola"
  surname: "Rossi"
  phone_number: "0346 4403095"
```

Figure 3: Phone number **before** the execution of the command

```
_id: ObjectId("61a9f0d693194739d385d1da")
id: "46s7zbsm8bpygg65sr74snxx"
> person: Object
> vaccination: Array
> tests: Array
~ emergency_contact: Object
  name: "Viola"
  surname: "Rossi"
  phone_number: "0528 1949992"
```

Figure 4: Phone number **after** the execution of the command

5 UI Application

The UI application is a graphic implementation of the queries described above. It is made in Java with the use of a couple of external libraries (JavaFX for the UI, MongoDB for the queries to the database). It is composed of a single page, which handles all the queries: you can select the query or the action you want to perform, and for each of them a brief explanation of what will be asked to the database will be displayed along with an optional text fields (in case the user has to insert some information), and the results of the query.

5.1 Connecting the application to the DataBase

Connecting to MongoDB DataBase is quite simple and requires few lines of code:

```
//string to connect to the correct database
MongoClientURI uri = new
    MongoClientURI("mongodb+srv://root:password123456*qwerty@cluster0.k5ok7.mongodb.net/
test?authSource=admin&replicaSet=atlas-z7a47q-shard-0&readPreference=primary&appName=MongoDB%20
Compass&ssl=true");

//creates a client associated with the string above
MongoClient mongoClient = new MongoClient(uri);

//connects to the correct database in the client
MongoDatabase database = mongoClient.getDatabase("CovidCertificationData");

//selects the correct collection
MongoCollection<Document> collection = database.getCollection("Certificates");
```

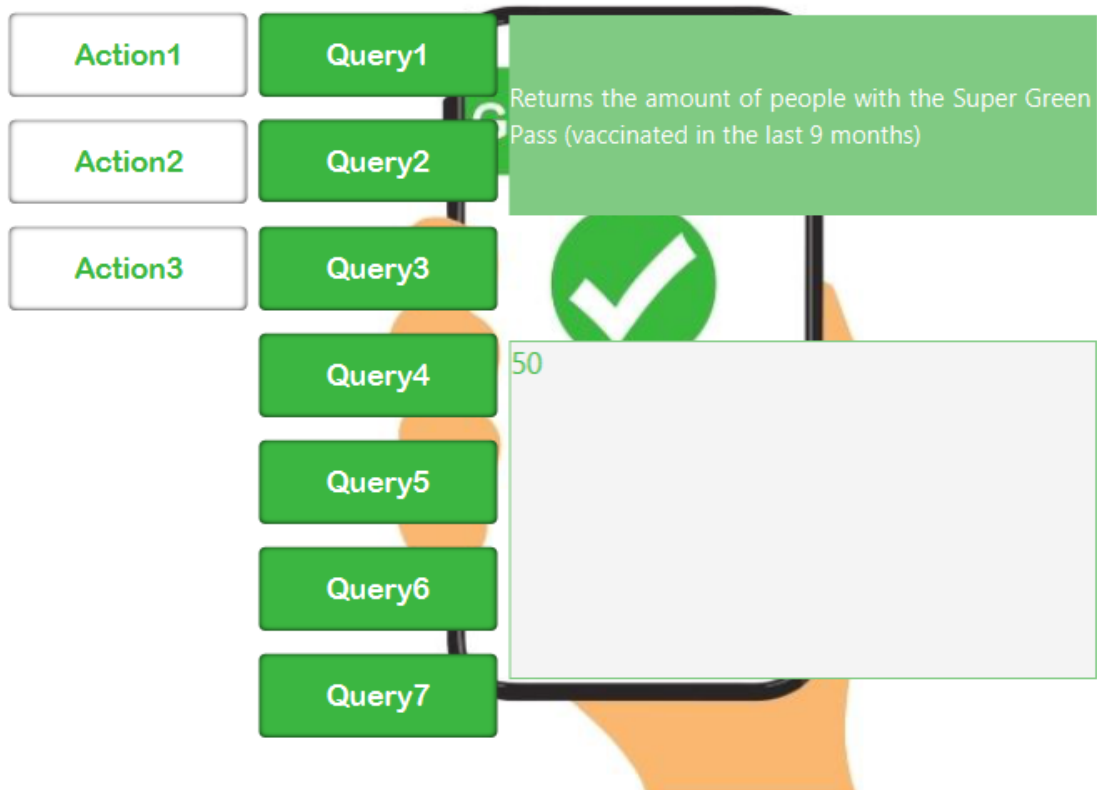
5.2 Perform queries

Queries from Java are similar to the ones from MongoDB, with small differences in the syntax.

Example of a query without inputs:


```
LocalDate nowDate = LocalDate.now().minusMonths(9);  
Bson query = gte("vaccination.somministration_date", nowDate);  
count = collection.countDocuments(query);
```

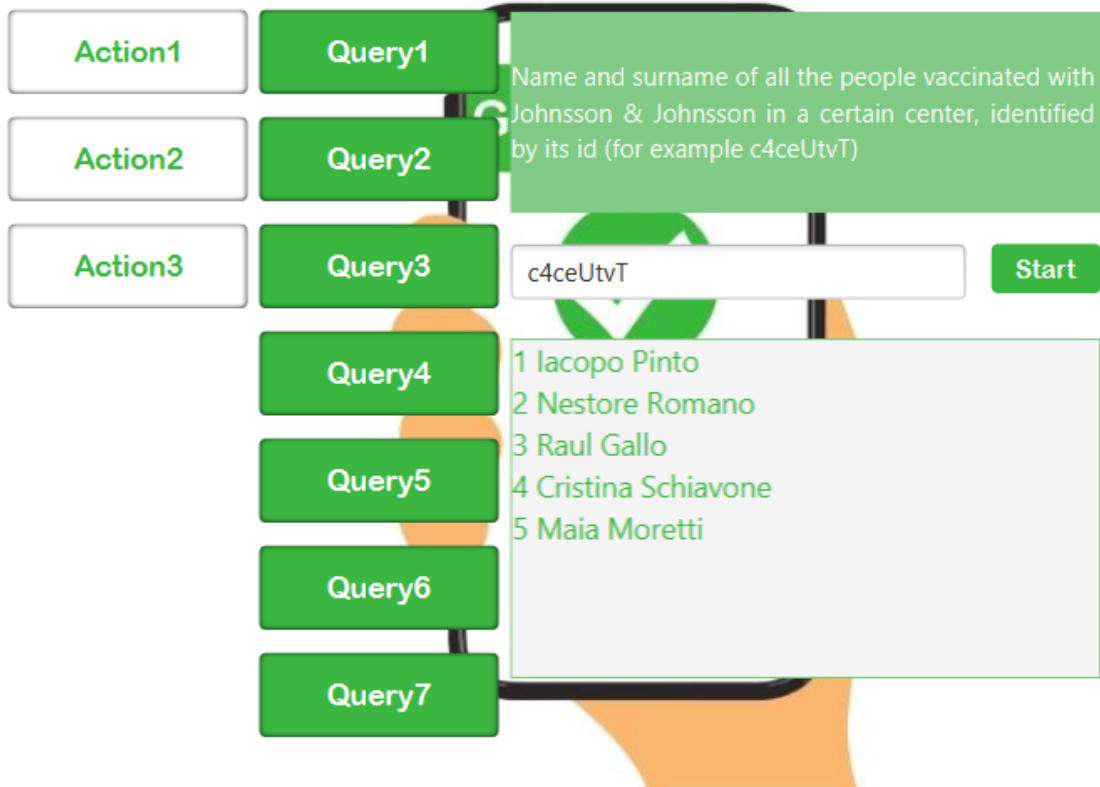
Green Pass Simulator



Example of a query with inputs:

```
Bson req1 = eq("vaccination.authorized_center_id", id);
Bson req2 = eq("vaccination.brand", "Johnson & Johnson Janssen");
MongoCursor<Document> cursor = collection.find(and(req1, req2)).iterator();
try {
    while (cursor.hasNext()) {
        Document a = cursor.next();
        Document person = a.get("person", Document.class);
        String name = person.getString("name");
        String surname = person.getString("surname");
        resultList.add(name + " " + surname);
    }
} finally {
    cursor.close();
}
return resultList;
```

 Green Pass Simulator



The interface consists of a vertical column of buttons on the left, labeled Action1, Action2, Action3, Query1, Query2, Query3, Query4, Query5, Query6, and Query7. Action1, Action2, and Action3 are white with green text. Query1 through Query7 are green with white text. To the right of the Query buttons is a large green rectangular area. Query1's text is displayed inside this area: "Name and surname of all the people vaccinated with Johnson & Johnson in a certain center, identified by its id (for example c4ceUtvT)". Below this text is a white input field containing "c4ceUtvT" and a green "Start" button. Below the input field is a white rectangular area containing a list of names: "1 Iacopo Pinto", "2 Nestore Romano", "3 Raul Gallo", "4 Cristina Schiavone", and "5 Maia Moretti".

6 Team composition

The project was realized by:

- Curti Gabriele, 10624502
- Cutrupi Lorenzo, 10629494
- Samuele Mariani, 10622653
- Alessandro Molteni, 10623928
- Matteo Monti, 10622780

7 Sources

- Slides from the lessons and exercise session.
- fakenamegenerator.com to generate raw data.
- mongodb.com/manual documentation manual for MongoDB.
- stackoverflow.com for some help in constructing the queries and the python and Java codes.