

Group Project Part 3 - ElasticSearch  
of Systems and Methods for Big and Unstructured Data Course  
(SMBUD)

held by  
Brambilla Marco  
Tocchetti Andrea

**Group 14**

Banfi Federico  
10581441

Carotenuto Alessandro  
10803080

Donati Riccardo  
10669618

Mornatta Davide  
10657647

Zancani Lea  
10608972

Academic year 2021/2022



**POLITECNICO**  
MILANO 1863

# Contents

<b>1</b>	<b>Problem Specification</b>	<b>3</b>
<b>2</b>	<b>Hypotheses</b>	<b>3</b>
<b>3</b>	<b>Queries and Commands</b>	<b>5</b>
3.1	Queries . . . . .	5
3.1.1	Find all the people with the Reinforced GreenPass . . .	5
3.1.2	Find the number of vaccinations ordered per month . .	7
3.1.3	Find the age average of unvaccinated people . . . . .	7
3.1.4	Find all currently infected people . . . . .	8
3.1.5	Find the number of tests per authorized body (JOIN)	10
3.2	Commands . . . . .	11
3.2.1	Insert a new Authorized Body (CREATE) . . . . .	11
3.2.2	Insert a test in a Certification (UPDATE) . . . . .	12
3.2.3	Delete all the Authorized Bodies that test and are pharmacies (DELETE) . . . . .	13
<b>4</b>	<b>References &amp; Sources</b>	<b>14</b>

# 1 Problem Specification

The technological solutions based on the use of Big Data are not limited exclusively to the tracking and profiling of vast amounts of data to obtain information useful for practical purposes - as seen in previous deliveries - but are also fundamental in the field of research and of statistical studies. This time, in fact, our project is focused on the use of an Information Retrieval system specifically designed to the collection and the in-depth study of data to obtain more precise and meaningful analyzes.

In particular, the ElasticSearch search engine will be used to monitor the spread of the SARS-CoV-2 virus in recent times from a global point of view and to track the progress of the vaccination campaign from a perspective no longer linked to individuals but to a larger scale. Thanks to the IR-based approach on which it is based, ElasticSearch is the most suitable system for this kind of task as, compared to other SQL and NoSQL solutions, it gives the possibility to compute more accurate searches by exploiting the main features available: the full-text search and the ranking strategy used to evaluate the relevance of the results obtained, which guarantee more wide-ranging query outcomes in order of "importance".

## 2 Hypotheses

During the design phase, some considerations were made regarding how to structure and implement the database to obtain a solution that was effective and performing but at the same time consistent with the real current scenarios. First of all, two types of documents have been created: Certification and AuthorizedBody; the first represents a sort of "medical chart" containing the following fundamental information for each individual:

1. list of tests she/he has undergone,
2. list of vaccine doses that have been administered,
3. personal details (such as name, surname, date of birth, etc.),
4. one or more emergency contacts to call in case of need.

Within the documents stored in the database for each of these four fields, subfields have been provided (as shown in the "jsonSchema.json" file into the "documents" folder) that allow you to manage the various data with MongoDB in order to execute specific queries and commands.

The second document contains details related to Authorized Bodies, i.e. institutional places where it is possible to get vaccinated and/or where one

can undergo a test, the fields are based on specific assumptions about where these places are located and the healthcare personnel working there.

At last, in order to record a greater number of elements to be processed within the dataset, while ensuring the meaningfulness and consistency of the information stored, some assumptions and limitations have been established in the creation of the data managed by the database, therefore:

- each Authorized Body is associated with a document identified by a unique ID, which is directly referred to in the test/vaccine lists of the Certification document;
- each test/vaccine was associated with only one member of the healthcare personnel who represents a sort of "responsible" for that given administration;
- for the sake of simplicity in data management, it was decided that for each Authorized Body there should be a list consisting of at most 5 members of the healthcare personnel, "responsible" for the various tests/vaccines to which they are associated;
- each person can come from any Italian location, but only Authorized Bodies belonging to the city of Milan with the relative coordinates have been examined for the purpose of performing more in-depth analysis on the data;
- vaccines were administered throughout the last year while tests during the last month.

## 3 Queries and Commands

The correct functioning of the information system involves the implementation of some essential commands and queries for the database in order to properly support the app and to ensure the right execution of searches among the data available for statistical or practical purposes.

First of all you need to load the .csv files in two separate collections (authorizedBodies and certifications), you can find them following the path "db/ab.csv" and "db/certification.csv"

When you are importing the data make sure to change the datatypes in:

- All the dates/datetime from String to Date
- In certifications test/vaccine.id\_authorized\_body from String to ObjectId
- In authorizedBodies \_id from String to ObjectId

### 3.1 Queries

#### 3.1.1 Find all the people with the Reinforced GreenPass

This query allows us to find all people in possession of a reinforced green-pass after the latest regulations, that is, all those who have been vaccinated in the last 9 months.

##### QUERY

```
db.certifications.find({
  "$and": [{"vaccination":{"$exists":true}},
  {"vaccination.datetime":{"$gte":new Date(
    ISODate().getTime() - 1000 * 3600 * 24 * 270)}}},
  {"vaccination.datetime":{"$lte":new Date(
    ISODate().getTime())}}]
},{
  "person.name":1,
  "person.surname":1,
  "vaccination":1
})
```

The output is given by:

- The ObjectId of the certification document
- The name and the surname of the person
- The datetime of the vaccination

#### OUTPUT

```
{ _id: ObjectId("61af82fb2468d4592b37b6ee"),  
  person: { name: 'Luca', surname: 'Colombo' },  
  vaccination:  
    [ { datetime: 2021-09-07T17:31:42.000Z },  
      { datetime: 2021-10-05T17:31:42.000Z } ] }  
{ _id: ObjectId("61af82fb2468d4592b37b70e"),  
  person: { name: 'Lorenzo Federico Giuseppe',  
            surname: 'Tomasi' },  
  vaccination: [ { datetime: 2021-11-14T10:18:51.000Z } ] }  
{ _id: ObjectId("61af82fb2468d4592b37b6e9"),  
  person: { name: 'Vincenzo', surname: 'Maggiani' },  
  vaccination:  
    [ { datetime: 2021-10-23T15:51:37.000Z },  
      { datetime: 2021-11-20T16:51:37.000Z } ] }  
  . . .
```

### 3.1.2 Find the number of vaccinations ordered per month

This query allows us to retrieve some statistical data on the number of the vaccinations per month.

#### QUERY

```
db.certifications.aggregate(  
  { $unwind : "$vaccination" },  
  { $group: {  
    _id: {month: { $month: "$vaccination.datetime" },  
    year: { $year: "$vaccination.datetime" } },  
    count: { $sum: 1 }  
  }  
}, {  
  "$sort": {count: -1}  
}  
)
```

The output is given by:

- The count of the vaccinations
- The months and year

#### OUTPUT

```
{ _id: { month: 11, year: 2021 }, count: 41 }  
{ _id: { month: 2, year: 2021 }, count: 36 }  
{ _id: { month: 10, year: 2021 }, count: 36 }  
{ _id: { month: 8, year: 2021 }, count: 36 }  
{ _id: { month: 4, year: 2021 }, count: 35 }  
{ _id: { month: 3, year: 2021 }, count: 32 }  
{ _id: { month: 12, year: 2021 }, count: 29 }  
{ _id: { month: 5, year: 2021 }, count: 27 }  
{ _id: { month: 7, year: 2021 }, count: 27 }  
{ _id: { month: 9, year: 2021 }, count: 26 }  
{ _id: { month: 1, year: 2021 }, count: 22 }  
{ _id: { month: 6, year: 2021 }, count: 14 }
```

### 3.1.3 Find the age average of unvaccinated people

This query also allows us to get a statistical information about the age of the unvaccinated people

## QUERY

```
db.certifications.aggregate([
  { $match : {
    "person.birthdate" : { $exists : true},
    "vaccine":{"$exists":false}
  } },
  { $project : {"ageInMillis" : {$subtract :
    [new Date(), "$person.birthdate"] } } },
  { $project : {"age" : {$divide :
    ["$ageInMillis", 31558464000] } }},
  // take the floor of the previous number:
  { $project : {"age" : {$subtract : ["$age",
    {$mod : ["$age",1]]}}}},
  { $group : { _id : true, avgAge : { $avg : "$age" } } },
  { $project: { "avgAge": { $round: ["$avgAge", 2] } } }
])
```

The output is given by:

- The age average

## OUTPUT

```
{ _id: true, avgAge: 58.14 }
```

### 3.1.4 Find all currently infected people

This query allows us to find people currently infected, i.e. those whose last test is positive.



## QUERY

```
db.certifications.aggregate([
  {$addFields : {test : {$reduce : {
    input : "$test",
    initialValue : {datetime :
      new ISODate('2000-01-01T00:00:00')},
    in : {$cond: [{gte : ["$$this.datetime",
      "$$value.datetime"]}, "$$this", "$$value"]}}}
  }},
  { $match: {"test.result": "Positive"}},
  {$project: {
    "person.name": "$person.name",
    "person.surname": "$person.surname",
    "person.codice_fiscale": "$person.codice_fiscale",
  }}
])
```

The output is given by:

- The ObjectId of the certification document
- The name and surname of the person
- The CF of the person

## OUTPUT

```
{ _id: ObjectId("61af82fb2468d4592b37b71e"),
  person:
    { name: 'Adele',
      surname: 'Mantovani',
      codice_fiscale: 'MNTDLA62D60C059I' } }
{ _id: ObjectId("61af82fb2468d4592b37b71b"),
  person:
    { name: 'Ferruccio',
      surname: 'Meloni',
      codice_fiscale: 'MLNFRC61D17B246A' } }
{ _id: ObjectId("61af82fb2468d4592b37b7ab"),
  person:
    { name: 'Riccardo',
      surname: 'Brumat',
      codice_fiscale: 'BRMRCR65T22F356R' } }
```

### 3.1.5 Find the number of tests per authorized body (JOIN)

This query allows us to find the number of tests that each authorized body did, in this query we use the JOIN of the two document in order to retrieve the name and the type of the authorized body starting from the id in the certification document.

MongoDB is not optimized to perform this type of operation like relational databases, so the query is optimized to perform the join on as few elements as possible grouping the tests in pipeline before the join.

#### QUERY

```
db.certifications.aggregate([
  { $unwind : "$test" },
  {$group:{
    _id:"$test.id_authorized_body",
    count:{$sum:1}
  }},
  {
    "$lookup": {
      "from": "authorizedBodies",
      "localField": "_id",
      "foreignField": "_id",
      "as": "authorizedBodyInfo"
    }
  },
  {
    "$project":{_id:0,count:1,
      "authorizedBodyInfo.name":"$authorizedBodyInfo.name",
      "authorizedBodyInfo.type":"$authorizedBodyInfo.type"
    }
  }
])
```

The output is given by:

- The count of the tests done
- The name of the authorized body
- The type of the authorized body

## OUTPUT

```
{ count: 17,
  authorizedBodyInfo: [ { name: [ 'Policlinico' ],
                        type: [ 'Hospital' ] } ] }
{ count: 21,
  authorizedBodyInfo: [ { name: [ 'Auxologico San Luca' ],
                        type: [ 'Hospital' ] } ] }
{ count: 25,
  authorizedBodyInfo: [ { name: [ 'San Raffaele' ],
                        type: [ 'Hospital' ] } ] }
{ count: 19,
  authorizedBodyInfo: [ { name: [ 'San Paolo' ],
                        type: [ 'Hospital' ] } ] }
{ count: 26,
  authorizedBodyInfo: [ { name: [ 'San Giuseppe' ],
                        type: [ 'Hospital' ] } ] }
{ count: 15,
  authorizedBodyInfo: [ { name: [ 'Ospedale Fiera' ],
                        type: [ 'Hospital' ] } ] }
. . .
```

## 3.2 Commands

### 3.2.1 Insert a new Authorized Body (CREATE)

In this command we insert in the authorizedBodies collection a new document.

## COMMAND

```
db.authorizedBodies.insertOne({
  "address":{
    "address":"Via Roma",
    "cap":"24030",
    "geopoint":"45.0000 9.0000",
    "house":"13b",
    "municipality":"Bergamo",
  },
  "department":"covid test",
  "name":"Farmacia Comunale",
  "type":"pharmacy",
  "healthcarePersonnel":[
    {
      "cellphone":"+393476574382",
      "codice_fiscale":"DNTHGB87P15G764J",
      "name":"Carlo",
      "role":"pharmacist",
      "surname":"Merlutti",
    }
  ]
})
```

### 3.2.2 Insert a test in a Certification (UPDATE)

In this command we add to a given certification (via ObjectId) a new test as a subdocument in the ArrayList field test.

#### COMMAND

```
db.certifications.updateOne({
  "_id":ObjectId('61af82fb2468d4592b37b70e')
},{
  "$push":{
    "test":{
      "$each":[{
        "datetime":ISODate('2021-12-08T16:20:00'),
        "healthcarePersonnel":{
          "codice_fiscale":"PLCLRT48C02C224Y",
          "name":"Alberto Christian",
          "surname":"Paolucci"
        },
        "id_authorized_body":
          ObjectId('61acd3e25ca7e964a122def1'),
        "result":"Positive",
        "type":"Molecular"
      }]
    }
  }
})
```

### 3.2.3 Delete all the Authorized Bodies that test and are pharmacies (DELETE)

In this command we simulate a change of regulations where the pharmacies can no longer test people and therefore we delete this kind of authorized body from the collection.

#### COMMAND

```
db.authorizedBodies.deleteMany({
  "$and":[
    {"type":"pharmacy"},
    {"department":"covid test"}]
})
```

## 4 References & Sources

- [1] Course Slides
- [2] <https://pysimplegui.readthedocs.io/en/latest/call%20reference/>
- [3] <https://docs.mongodb.com/manual/>
- [4] <https://pymongo.readthedocs.io/en/stable/>
- [5] <http://iniball.altervista.org/Software/ProgER>
- [6] <https://pandas.pydata.org/docs/>