POLITECNICO
MILANO 1863

A.Y. 2021-2022

Computer science and engineering

Systems and Methods for Big and Unstructured Data
Delivery 2

Report

Beretta Davide
Bollini Matteo
Caleffi Nicolò
De Nicolò Andrea
Madhkour Amor

December 12th 2021

Table of contents

1.Introduction

This document describes the relevant for the realization of a document-based database supporting a certification app that keeps the records of all vaccinations and tests for COVID-19 for all registered people. What follows is the result of combined efforts of the students in group number **26**:

- Beretta Davide (10656835)
- Bollini Matteo (10615991)
- Caleffi Nicolò(10588181)
- De Nicolò Andrea(10837851)
- Madhkour Amor(10794343)

More specifically, in **chapter two** it is provided a brief explanation of the problem and the specification of the project work along with our assumptions.

In **chapter three** it is shown the conceptual model of the database, which is then followed by a description of the relevant aspects of the domain that was modeled.

In **chapter four** it is described the structure of the documental database, in particular we specify which are the collections composing the DB and the fields of the documents they contain.

Finally, the queries and commands are reported in **chapter five**. For each statement its goal and a brief explanation are given, flanked by an example of the results produced upon its execution.

2. Specification

The development of vaccines against COVID-19 divided the population into two categories of people: the first, the vaccinated ones, are less likely to get COVID or to spread the disease, while the second, unvaccinated, are exposed to a higher risk of getting COVID and moreover of developing dangerous symptoms. Governments around the world started applying different levels of restrictions based on this distinction, keeping softer restrictions to the ones who are vaccinated and stronger ones to the other slice of population. Unvaccinated people who want the same freedom of vaccinated ones can prove they are not a risk for the rest of the population with a test that proves their negativity to COVID-19.
An electronic application can easily keep track of who is vaccinated or not and all the history of the tests a person made. In particular a document-based database is a good data storage for reproducing physical paper documents such as certificates that contain all the information about the person and their vaccines/tests, without a particular need to put them in a relationship.

The developed database allows storing personal information about the single individuals such as name, surname, demographical data and a certification of all vaccines and tests done, along with a reference to an emergency contact. It stores all the information about vaccinations and tests such as the place and the date they were issued and the medical staff attending, and information about the vaccines (brand, type, lot, production date...). Moreover, dedicated documents allow storing data about the authorized bodies allowed to emit the certifications.
In particular we store, for each person:

- SSN (Social Security Number), name, birth date and health-relevant attributes (height, weight);
- Their address;
- A list of all vaccinations taken;
- A list of all tests done;
- An emergency contact (i.e., a person to be called in case of emergency), with the contact name, phone number, email and address.

For each vaccination:

- Brand, type, lot and production date of the vaccine used;
- The issuing date and place;
- A list of all the medical staff (doctors, nurses) attending the vaccination.

For each test:

- The result;
- The issuing date and place;
- A list of all the medical staff (doctors, nurses) attending the vaccination.

For each authorized body:

- Service name and address, used as identifier;
- GPS position;
- The type of entity (hospital, pharmacy, ...)
- The department issuing the vaccine (i.e., the department the organization belongs to, which provides vaccines to be used)

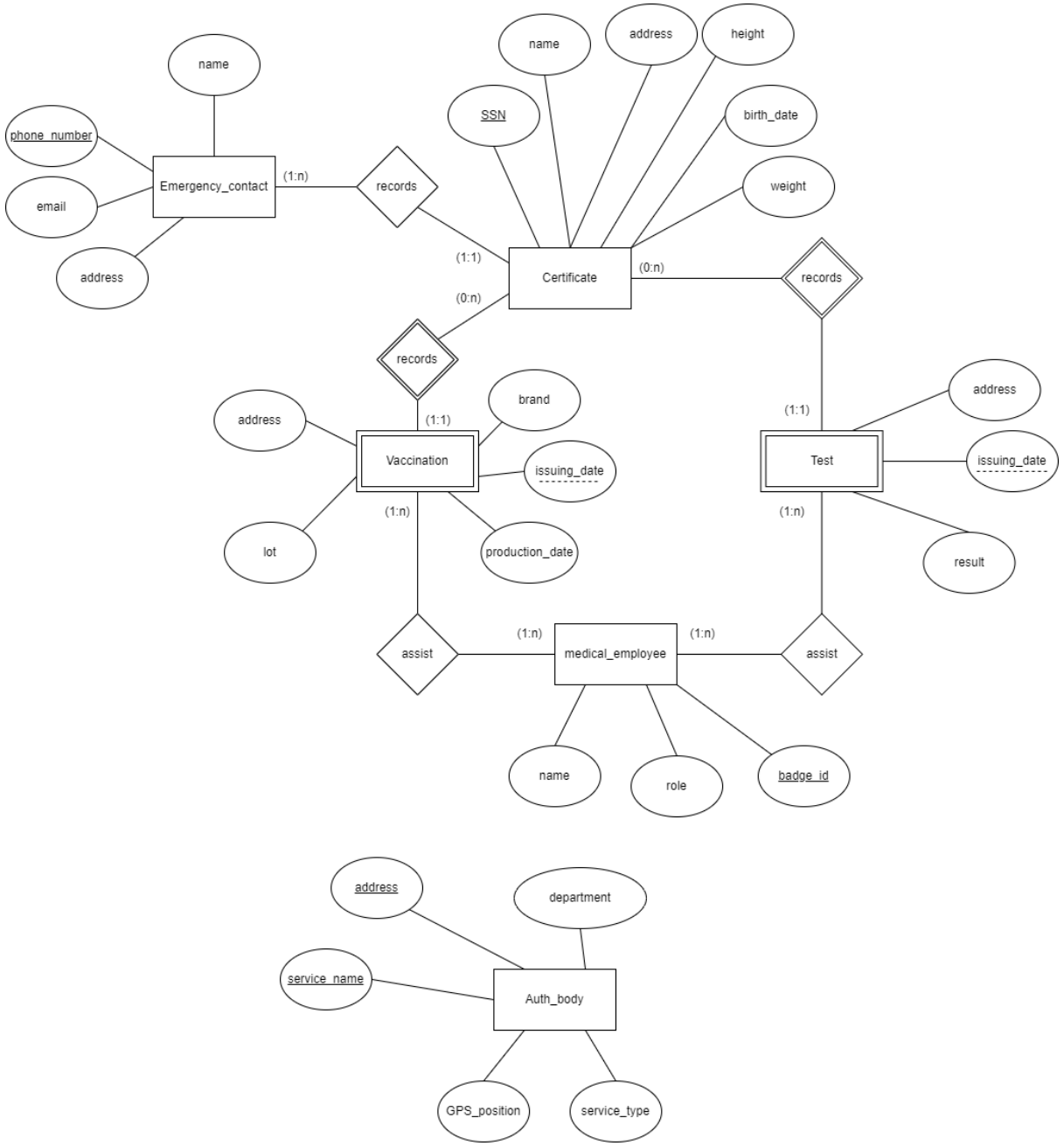
2.A Assumptions

- A single type of test is considered
- Since the expiration date of vaccination and tests may change (in particular the expiration date of vaccinations may change after the vaccination has been performed), we assume the application knows the validity time of tests and vaccinations (based on the dose number and the brand). The application may be updated or notified when the validity time changes. This way we store only the issuing date and do not need to modify many documents (a bit slow operation) when a validity date changes.
- Each person can be uniquely identified by his/her ssn or another equivalent code (for instance, the italian “codice fiscale”)
- Each medical employee can be uniquely identified by its badge id. If his role changes, then his badge changes too.
- Any number of nurses or doctors can attend a vaccination or test
- Each authorized body is uniquely identified by its name and address
- Even if an authorized body can perform vaccinations or tests in a place different from its head office, they usually operate only near their head office (for instance, a hospital may operate in a near hub, but not too far away). This assumptions is used by geospatial queries
- Each vaccine lot is uniquely identified by a code
- The database may include certificates that have only vaccinations or only tests
- It’s possible to determine if a person can take a new vaccination and which brand of vaccine to use at a given date based on the number of vaccinations she has already taken, the brand of them, the date of the last vaccination and the date of her last positive test if any.

3.Conceptual Model

In the following section we will discuss the Entity-Relationships conceptual diagram. It was built to describe a simplified scenario to keep track of Covid-19 vaccinations and tests.

In the picture below there is a schema of the ER conceptual model we based our database on.



The entity person is not portrayed because of the 1 to 1 relationship existing between the person and its certificate. Instead the attributes related to a person are directly considered attributes of the entity certificate. Which, as it can be seen from the diagram, contains information about the emergency contact, the tests taken as well as the vaccine doses taken by the person associated with that very certificate.

Tests and vaccines are described as weak entities since they have no primary key. Instead they are uniquely identified by considering the SSN of the person to whom they were given and the date in which they were administered. Moreover, for each vaccine is kept track of the medical staff that was present during the administration of the vaccine dose or test. The staff is composed of at least one employee which can be either a doctor or a nurse (specified in the role attribute).

The authorized body is represented as a separated entity with no relationship whatsoever to any other entity.

obs: in all entities, in which they figure, the name and address are a single attribute but they actually contain multiple informations. When it comes to names, they include both the first name (or multiple first names if necessary) and the surname (if any) of the person, emergency contact or employee. Whereas the addresses are single strings each of which in the format <city>, <street> <civic number>.

4. Database structure

The database is divided in two collections: certificates (which stores people certificates) and organizations (which stores data regarding authorized bodies).
We prefer to let the _id field to be automatically generated, and we add an unique index to the fields that are the primary key in the ER for the Certificate and Auth_body entities. This way, you don't need to wonder about what the _id field contains: the primary key field(s) has a significant name, the unique constraint and the speed up in search given by the index.

4.A certificates collection

The "certificates" collection stores the Certificate documents, which are the main entities we want to store. From the E/R diagram, the document is obtained as a JSON object with a field for each attribute of the Certificate entity. Then, embedded documents are added for the other entities that are in relation with it. The procedure is recursively applied till all entities have been included.
In the following:

- The fields "name" include both the first name (or multiple first names if necessary) and the surname (if any) of the person.
- The fields "birth_date", "issuing_date", and "production_date" are stored using the MongoDB Date type
- Primitive fields are shown using the syntax (the type of the field can usually be intuitively derived from the field name) <field_name>:
- Arrays of embedded documents are shown with the syntax:[{<description of an embedded document>} ,{...},...]
- If a certificate doesn't include any vaccination (or test), the "vaccinations" (or "tests") field is an empty array (i.e., it's value is [])
- The field "_id" is omitted

```
Certificate:{
      ssn:
      name: (contains both name and surname)
      birth_date:
      height:
      weight:
      address:

      "vaccinations":[
       {
                  brand:
                  vaccine_type:
                  lot:
                  production_date:
```

```
                issuing_date:
                address:
                employees:[
                        {
                                bedge_id:
                                role:
                                name:
                        },
                        {...},
                        ...
                ]
        },
        {...},
        …
    ]
    “tests”:[
        {
                positive:
                issuing_date:
                address:
                employees:[
                        Medical_employee{
                                        bedge_id:
                                        role:
                                        name:
                        },
                        {...},
                        ...
                ]
        },
        {...},
        …
    ]
    emergency_contact:{
        name:
        phone_number:
        email:
        address:
    }
}
```

obs: A regular unique index is applied to the field “ssn”

4.B organizations collection

This is the other stand-alone document we implement to keep track of the entities that release the certificates.

```
Auth_body:{
    service_name:
    address:
```

```
      service_type:
      department:
      gps_position:
}
```

The gps_position string is stored as a GeoJSON Object, that has the format:

```
"gps_position": {
        "type": "Point",
        "coordinates": [
            <longitude>,
            <latitude>
        ]
    }
```

obs: A regular and unique index is applied to the fields “service_name” and “address.
obs: A geospatial index is applied to the field “gps_position”.

4.C Database creation

You can generate the database from the dump provided in the delivery folder.
In order to do so, you need to use the mongorestore utility provided with the MongoDB Database tools. Run the command:

mongorestore --uri mongodb+srv://<username>:<password>@<cluster> <dump_folder_path>

In an Atlas cluster, the first part can be found under databases->command line tools->Binary Import and Export Tools. You just need to add the <dump_folder_path>.

In case of any problems generating the database, follow this procedure to manually create it and import the data from the provided json files:

- Create an empty database
- Create a collection called “certificates” and populate it importing the data from the **certificates.json** file
- Create a collection called “organizations” and populate it importing the data from the **authorized_bodies.json** file
- Create the indexes executing the commands:
    - db.certificates.createIndex({"ssn": 1}, { unique: true })
    - db.organizations.createIndex({"service_name": 1,"address": 1}, { unique: true })
    - db.organizations.createIndex({ gps_position: "2dsphere" })

obs: The data has been randomly generated through some Python scripts. Some data may be a bit inconsistent. We used regions as departments.

5 Queries and commands

5.A Queries

In this section some possible queries are shown. These queries are written to be used from Mongosh after the database is selected with the command **use <datbase_name>** (in our case use projectSMBUD). Values can be assigned to variables with the command **var <variable_name>=<value>**. You can assign arrays or other json objects to variables in order to provide complex parameters.

5.A.1 Last vaccination and test

This query is used to retrieve the last vaccination (date and vaccine brand) and test (date and result) of a person. The application can use these data to decide whether the last vaccination or test has expired or not.

Parameters:

- ssn: the person ssn (string)

```
db.certificates.aggregate(
	{ $match:{"ssn":ssn} },
	{ $project:{
		"vaccination":{$last:"$vaccinations"},
		"test":{$last:"$tests"}
		}

	},
	{ $project:{
		"vaccination":{
			"issuing_date":"$vaccination.issuing_date",
			"brand":"$vaccination.brand"
		},
		"test":{
			"issuing_date":"$test.issuing_date",
			"positive":"$test.positive"
		}}
	}
)
```

Possible result:

```
{ _id: ObjectId("61b62b7d2751a39537d7ca5a"),
  vaccination: { issuing_date: 2021-05-26T00:00:00.000Z, brand: 'Moderna' },
```

```
  test: { issuing_date: 2021-06-02T00:00:00.000Z, positive: false } }
```

5.A.2 Data for the next dose

This query retrieves some data regarding previous vaccination and tests that can be useful to decide if a new dose of vaccine should be issued to a person, and which brand to use.
Given a person ssn, it retrieves:

- the dates and the brands of previous vaccinations
- the date of the last positive test

We have assumed that this data is enough for the doctor to define if the person should take a new vaccination, and which brand to be used.
In the query result, the field "last_positive_test" contains the date of the last positive test, and is null if there are no positive tests; the field vaccinations is an empty array if the person has not any vaccination.

Parameters:

- ssn: the person ssn (string)

```
db.certificates.aggregate([
      { $match:{"ssn":ssn} },
      { $project:{
            "vaccinations":1,
            "tests":1,
            }
      },
      { $unwind:{
            path:"$tests",
            preserveNullAndEmptyArrays: true
            }
      },
      { $match:{ //if there are no tests, I must match the single record I
have
            $or:[
                  {"test.positive":{"$exists":false} },
                  {"test.positive":true}]
            }
      },
      { $group:{
            _id:"$vaccinations",
            "last_positive_test":{$max:"$tests.issuing_date"}
            }
      },
      { $project:{ //since _id is not meaningful, change it to vaccinations
            "_id":0,
            "vaccinations":"$_id",
            "last_positive_test":1
```

```
            }
      },
      { $project:{
            "vaccinations.issuing_date":1,
            "vaccinations.brand":1,
            "last_positive_test":1 //null if there are no tests
            }
      }
])
```

Possible result:

```
{ last_positive_test: 2021-06-02T00:00:00.000Z,
  vaccinations:
   [ { issuing_date: 2021-05-09T00:00:00.000Z, brand: 'AstraZeneca' },
     { issuing_date: 2021-05-26T00:00:00.000Z, brand: 'Moderna' } ] }
```

5.A.3 Vaccines used in a given period

This statistical query returns, for each brand, the number of vaccines used in a given period. Brands that have not been used aren't considered. The result is ordered from the brand with the highest number of used doses to the ones with the lowest.

Parameters:

- min_date: the first date to be considered (included)
- max_date: the last date to be considered (included)

```
db.certificates.aggregate([
      { $match:{"vaccinations.0":{$exists:1}}},
      { $unwind:"$vaccinations"},
      { $match:{
            "vaccinations.issuing_date":{
            $gte:min_date,
                  $lte:max_date}
        }
      },
      { $group:
        {
          _id:"$vaccinations.brand",
          count:{$sum:1}
        }
    },
      { $sort:{"count":-1}}
])
```

Possible result:

```
{ _id: 'Pfizer', count: 51 }
{ _id: 'Moderna', count: 45 }
{ _id: 'AstraZeneca', count: 44 }
```

5.A.4 Number of vaccinated for city

This statistical query returns, for each city, the number of people that have done at least a certain number of doses. Results are sorted on the number of vaccinated with at least the given number of doses. A limit to the number of results has to be specified. Cities without at least a person with the given number of doses are ignored.

Parameters:

- required_doses: the number of doses to be counted
- max_results: the maximum number of results

```
db.certificates.aggregate([
      { $project : { city : {
            $arrayElemAt:[
                  {$split: ["$address", ", "]},
                  0]
            },
            vaccinations:"$vaccinations"

      } },
      { $match:{
            $expr: { $gte: [{$size:"$vaccinations"}, required_doses]}
            }
      },
      { $group : { _id: "$city",count:{$sum:1} } },
      { $sort : {   "count" : -1 } },
      { $limit: max_results}
])
```

Possible result:

```
{ _id: 'Verona', count: 13 }
{ _id: 'Bergamo', count: 11 }
{ _id: 'Pesaro', count: 10 }
```

5.A.5 Organizations of a type within a given radius

This query is based on the assumption that authorized bodies operate near their own head office (their address field). For instance, a hospital may operate in the near hubs. This means that it can be useful to know the location of the authorized bodies near a given position, because they are the ones that are likely to operate there. This query retrieves the entities of a given type near a position (returns the name, the distance and the gps position).

Parameters:

- longitude: the longitude of the position to be considered
- latitude: the latitude of the position to be considered
- radius: the maximum distance at which search for authorized bodies (in meters)
- required_type: the type of authorized body to search for

```
db.organizations.aggregate([
   {
     $geoNear: {
        near: { type: "Point", coordinates: [longitude, latitude ] },
        distanceField: "dist.calculated",
        maxDistance: radius,
            query: { service_type: required_type },
        includeLocs: "dist.location",
        spherical: true
     }
   },{$project:{"service_name":1,
"dist.calculated":1,"gps_position.coordinates":1}}
])
```

Possible result:

```
{ _id: ObjectId("61b62d402751a39537d7cb34"),
  service_name: 'A.O. SAN PAOLO - MILANO ',
  gps_position: { coordinates: [ 11.154773741824545, 43.46253362118465 ] },
  dist: { calculated: 0 } }
{ _id: ObjectId("61b62d402751a39537d7cb4b"),
  service_name: 'A.O.U. Cagliari',
  gps_position: { coordinates: [ 11.153393982959589, 43.46433888206183 ] },
  dist: { calculated: 229.8097726869612 } }
{ _id: ObjectId("61b62d402751a39537d7cb42"),
  service_name: 'IOV',
  gps_position: { coordinates: [ 11.153000529084403, 43.46471704802944 ] },
  dist: { calculated: 282.13930711114693 } }
{ _id: ObjectId("61b62d402751a39537d7cb40"),
  service_name: 'AZIENDA OSPED.S.GIOVANNI BATTISTA DI TOR',
  gps_position: { coordinates: [ 11.151085479993881, 43.46093431080507 ] },
  dist: { calculated: 347.137675655333 } }
{ _id: ObjectId("61b62d402751a39537d7cb3f"),
```

```
  service_name: 'Azienda Osp. Univ. G. Martino',
  gps_position: { coordinates: [ 11.150147463267846, 43.462891231628554 ] },
  dist: { calculated: 375.90644564959183 } }
{ _id: ObjectId("61b62d402751a39537d7cb39"),
  service_name: 'UNIV.STUDI NAPOLI-FEDERICO II-FAC.MEDIC.',
  gps_position: { coordinates: [ 11.150667276387624, 43.464149782462236 ] },
  dist: { calculated: 377.42756720140056 } }
```

5.A.6 Number of organizations within a given radius

Similar to the previous one, this query searches for organizations near a position, and returns for each organization the number of them that have been found. Types that are not near the position aren't considered.

Parameters:

- longitude: the longitude of the position to be considered
- latitude: the latitude of the position to be considered
- radius: the maximum distance at which search for authorized bodies (in meters)

```
db.organizations.aggregate([
   {
     $geoNear: {
        near: { type: "Point", coordinates: [longitude, latitude ] },
        distanceField: "dist.calculated",
        maxDistance: radius,
        spherical: true
     }
   },
   {$group : { _id: "$service_type",count:{$sum:1} } }
])
```

Possible result:

```
{ _id: 'pharmacy', count: 4 }
{ _id: 'hospital', count: 6 }
```

5.A.7 Number of people who got covid after the last dose

This statistical query returns the number of people who got covid after taking their last dose of vaccine. Useful to evaluate the effect of the vaccine in the last months (assuming that the interval between a dose and the next one is not too long).

Parameters: no parameter required.

```
db.certificates.aggregate([
	{$match:
		{$and:
			[{"vaccinations.1":{$exists:1}},
			{"tests.0":{$exists:1}}] //at least one test done
		}
	},
	{$project:{
		"ssn":1,
		"tests":1,
		"last_dose":{$last:"$vaccinations"}}
	},
	{$unwind:"$tests"},
	{$match:{"tests.positive":true}},
	{$match:{"$expr": { "$gt": ["$tests.issuing_date",
"$last_dose.issuing_date"]}} },
	{$group:{ _id: "$ssn"}}, //removes duplicates (a person who got
multiple positive tests after the last dose counts once only)
	{$count:"People who got covid after last dose"}
])
```

Possible result:

```
{ 'People who got covid after last dose': 9 }
```

5.B Commands

The following are some commands that can be used to alter the database.
In the following:

- date parameters are of type Date (e.g., a valid assignment for the variable issuing_date could be var issuing_date = ISODate('2021-12-18T00:00:00.000+00:00'))
- addresses are in the format “City, street number” where city is the city name. If a city cannot be uniquely identified by its name, other data such as the province or the region should be included before the comma.

5.B.1 Add a vaccination

This command adds a vaccination to a given certificate (identified by the person ssn). It assumes that the certificate is already in the database and that the new vaccination has been performed after all the previous ones.

Parameters:

- ssn: the identifier of the person/certificate to update
- issuing_date: the date when the vaccine has been issued
- production_date: the date when the vaccine has been produced
- address: the address where the vaccination has been performed
- lot: the identified of the vaccine lot
- brand: the vaccine brand
- vaccine_type: the type of the vaccine
- employees: the medical employees that supervisioned the vaccination. This parameter is an array in the format used for the medical employees as described in section 4.A.

```
db.certificates.updateOne(
	{"ssn":ssn},
	{$push: {
		vaccinations: {
			issuing_date:issuing_date,
			address:address,
			lot:lot,
			brand:brand,
			vaccine_type:vaccine_type,
			production_date:production_date,
			employees:employees
		}
	}}
)
```

Possible result:

```
{ acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0 }
```

5.B.2 Add a test

This command adds a vaccination to a given certificate (identified by the person ssn). It assumes that the certificate is already in the database and that the new test has been performed after all the previous ones.

Parameters:

- ssn: the identifier of the person/certificate to update
- issuing_date: the date when the test has been performed
- address: the address where the vaccination has been performed

- result: the result of the test (true if the person results infected, false otherwise)
- employees: the medical employees that supervisioned the vaccination. This parameter is an array in the format used for the medical employees as described in section 4.A.

```
db.certificates.updateOne(
{"ssn":ssn},{
 $push: {
   tests: {
              issuing_date: issuing_date,
               positive:result,
               address:address,
               employees: employees
             }
       }
 }
)
```

Possible result:

```
{ acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0 }
```

5.B.3 Add a new authorized body

This command adds a new authorized body.

Parameters:

- service_name: the name of the authorized body
- address: the address of the authorized body
- service_type: the type authorized body (e.g. “hospital”)
- department: the department of the authorized body
- longitude: the longitude of the authorized body
- latitude: the latitude of the authorized body

```
db.organizations.insertOne({
            service_name:service_name,
            address:address,
            service_type:service_type,
            department:department,
```

```
            gps_position:{
                  "type":"Point",
                  "coordinates":[longitude,latitude]
            }
      }
)
```

Possible result:

```
{ acknowledged: true,
  insertedId: ObjectId("61b652c2381184fc7431ce62") }
```