Pandemic Information System Model

SYSTEMS AND METHODS FOR BIG AND UNSTRUCTURED DATA

PROF. MARCO BRAMBILLA

SECOND DELIVERY
MONGODB PROJECT

December 2021

Avci Oguzhan - 10557284
Gentile Nicole - 10594355
Rigamonti Davide - 10629791
Singh Raul - 10623232
Tagliaferri Mattia - 10572418

POLITECNICO
MILANO 1863

Contents

1 Introduction **2**
- 1.1 Problem Specification . . . 2
- 1.2 Hypoteses . . . 2

2 Database **4**
- 2.1 Document Diagram . . . 4
- 2.2 Dataset description . . . 4
- 2.3 Views . . . 5
  - 2.3.1 Certificates with datetimes . . . 5
  - 2.3.2 Join bewteen certificates and authorized bodies . . . 6
- 2.4 Queries . . . 6
  - 2.4.1 Check validity by SSN . . . 6
  - 2.4.2 Defective Lot . . . 6
  - 2.4.3 Find Certificate by Id . . . 7
  - 2.4.4 Percentage of fully vaccinated people by age groups . . . 7
  - 2.4.5 Percentage of boosters by age groups . . . 8
  - 2.4.6 Percentage of vaccinated by age groups . . . 9
  - 2.4.7 Number of people with only the first dose . . . 9
  - 2.4.8 Percentage of vaccinated people . . . 10
  - 2.4.9 Number of tests per organization . . . 10
  - 2.4.10 Number of vaccines per organization . . . 10
  - 2.4.11 Number of vaccines per brand . . . 10
  - 2.4.12 Number of vaccines per type . . . 11
- 2.5 Commands . . . 11
  - 2.5.1 Add new vaccine . . . 11
  - 2.5.2 Delete old tests . . . 12
  - 2.5.3 Update emergency contact . . . 12
  - 2.5.4 Update government rules . . . 13

3 Application **14**
- 3.1 Description . . . 14
- 3.2 User Guide . . . 14
- 3.3 Screenshots . . . 15

4 References and sources **18**

1 Introduction

1.1 Problem Specification

The idea of the project is that of building a dataset containing information about *"green"* certifications and people/organizations acquiring and releasing them.
A certificate is released whenever a person takes a vaccination shot, takes a test to check if they are infected with Covid-19 of if they recover from Covid-19.
All certificates contain information about the person involved, the organization issuing the certificate and information about the vaccine or the test.
The purpose of this project is to build a system that allows to check the validity of these certificates according to the latest government rules.

1.2 Hypoteses

- Vaccines and tests

  We assumed that vaccines and tests don't expire and can be used as soon as they are produced without accounting for delivery time.

  Similarly to what happens in reality, we assumed that there is one certification for each test/vaccine, so that a person can have different certifications associated but not all of them may be valid at a given time.

  We also assumed that the minimum time elapsed between two doses for the same person is at least one day in order to have a better distribution of vaccinations inside the generated dataset; furthermore, all doses must be of the same vaccine brand and a single person can have up to 3 vaccine doses (this can easily be extended).

  Given the previous assumptions we don't consider positive test results to have any relevance and we do not store them in this database since its purpose is to contain certifications.

  Vaccines inside the generated dataset are produced from 01/01/2021 to 26/11/2021 with a number ranging from 0 to 2 lots per day, adding up to a total of 330 lots numbered from *1* to *330*.

  Tests inside the generated dataset are performed from 01/01/2021 to 26/11/20 21 and their lots are numbered from *1200* to *1600*.

  The validity check for certifications is performed by the application (more on this can be found in the Application section of this document).

- Other entities

  The SSN field for people is considered to be a unique identifier for the single individual.

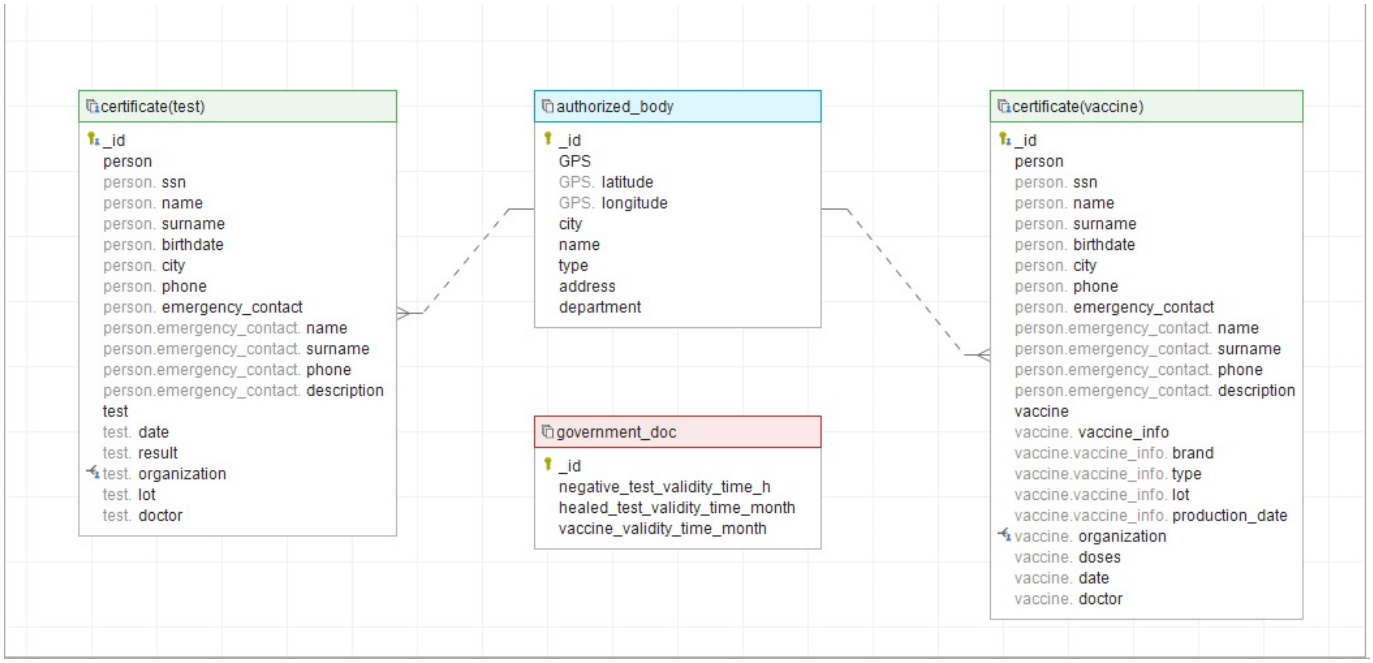
Names for regular people, doctors, cities and authorized bodies have no relevance and were chosen randomly, therefore information such as phone numbers for people and coordinates for authorized bodies aren't designed to be consistent among the dataset.

All authorized bodies can release all kinds of certificates.

People inside the generated dataset were born between years 1941 and 2011, therefore their age ranges roughly from *10 y.* to *80 y.* considering 2021 to be current year.

2 Database

2.1 Document Diagram



The diagram represents the first phase of the database design.

Since MongoDB is a documental database, we decided to use a document diagram to better describe the structure of our project.
The elements in the diagram are:

- `Authorized Body`: it contains information about any organization that can release vaccines or tests.
- `Government Document`: it represents the government rules about the duration of the various types of certificates, the validity period is expressed in hours for the negative tests and in months for vaccines and tests done after healing from Covid-19.
- `Certificate`: there are two types of certificates: one for tests and one for vaccines. Test certificates include information about the result of the test, while vaccine certifications include brand, type, production date of the vaccine and the number of the dose. Both contain information about the owner, including its emergency contact, the lot of test/vaccine, the organization that released it, the date it was made and the doctor that was attending the process.

2.2 Dataset description

We represented a population of about 200 people taking a total of 330 vaccinations (190 first doses, 120 second doses and 20 third doses) and 300 tests inside a total of 270

authorized bodies (130 hospitals, 80 pharmacies and 60 specialized centers).
Certifications are characterized by information about the related person (SSN, name, surname, birthdate, city, phone number and an emergency contact including the name, surname, phone number and description for a person to contact in case of emergency) and the vaccination (date, organization, doctor and information about the vaccine such as brand, type, lot and production date) or the test (lot, result, date, organization and doctor).
Authorized bodies are characterized by: GPS coordinates (latitude and longitude), city, name, type (pharmacy, hospital or specialized centers), address and department (only for hospitals).
There is also a document containing informations about the latest government rules on the time validity of tests and vaccines certifications; we assumed test validity is 24 hours (whereas in case a negative test implies that the person has healed, its duration is extended to 6 months), while vaccinations validity is 9 months.
The complete script used for the creation of the dataset, together with a database dump will be provided alongside this document.

2.3 Views

Views are used to make queries easier to be executed.

2.3.1 Certificates with datetimes

This view is used to convert all dates contained in certificates to datetime format. Moreover, we also add a field representing the age of the person.

```
db.createView(
"certificate_datetimes",
"certificate",
[
{"$addFields":{
        "vaccine.date": {
        "$dateFromString": {
          "dateString": "$vaccine.date"
        }
      },
       "test.date": {
        "$dateFromString": {
          "dateString": "$test.date"
        }
      },
       "person.birthdate": {
        "$dateFromString": {
          "dateString": "$person.birthdate"
        }
      }
}},
{"$addFields":{
  "person.age": {
        "$subtract": [{ $year: new Date() },
        { $year: "$person.birthdate" }]
      }
}}
])
```

2.3.2 Join bewteen certificates and authorized bodies

This view is used to retrieve the information about the authorized bodies issuing vaccinations or tests.

```
db.createView(
"certificate_authorizedBodies",
"certificate",
[
{ "$addFields":{
  "vaccine.organization": {
    "$toObjectId": "$vaccine.organization"}
    ,
    "test.organization": {
    "$toObjectId": "$test.organization"}
  }
},
{"$lookup": {
    "from": "authorized_bodies",
    "localField": "vaccine.organization",
    "foreignField": "_id",
    "as": "vaccine.organization"
  }
},
{"$lookup": {
    "from": "authorized_bodies",
    "localField": "test.organization",
    "foreignField": "_id",
    "as": "test.organization"
  }
}
])
```

2.4 Queries

2.4.1 Check validity by SSN

```
db.certificate.find( {  "person.ssn": ssn,
  "vaccine.doses": { "$gt": n_dose }}
)
```

Checks if a specific person has a newer vaccine dose than the specified one.
The query is used to invalidate old vaccination certificates.
It takes the following parameters:

- `ssn` *(string)*: social security number of the person;
- `n_dose` *(int)*: number of the dose.

2.4.2 Defective Lot

```
db.certificate.find(  {"vaccine.vaccine_info.lot": lot},
{"person.ssn":1,
"person.name":1,
"person.surname":1,
"person.phone":1})
```

Returns SSN, name, surname and phone number of people vaccinated with the specified defective vaccine lot.
It takes the following parameters:

- `lot` *(int)*: number of the defective lot.

2.4.3 Find Certificate by Id

```
db.certificate.find(
{"_id": ObjectId(id)
 })
```

Returns the certificate with a given id. The retrieved document is later used to check its validity according to the date, the government rules and, in case of vaccine, the presence of newer vaccinations.
It takes the following parameters:

- `id` *(string)*: id of the certificate.

2.4.4 Percentage of fully vaccinated people by age groups

```
// total number of people
var personNumber = db.certificate.distinct("person.ssn").length;

// divide people into groups and calcultate the percentage of second doses
// for each group
db.certificate_datetimes.aggregate(
      {$match: {"vaccine.doses": 2}},
      {$group: {
        _id: {
            $concat: [
              {$cond: [ {$and: [
                            {$lt: ["$person.age", 20]} ]}, "Under 20", ""]},
              {$cond: [ {$and: [ {$gte: ["$person.age", 20]},
                            {$lt: ["$person.age", 40]} ]}, "20 - 40", ""]},
              {$cond: [ {$and: [ {$gte: ["$person.age", 40]},
                            {$lt: ["$person.age", 50]} ]}, "40 - 50", ""]},
              {$cond: [ {$and: [ {$gte: ["$person.age", 50]},
                            {$lt: ["$person.age", 60]} ]}, "50 - 60", ""]},
              {$cond: [ {$and: [ {$gte: ["$person.age", 60]},
                            {$lt: ["$person.age", 70]} ]}, "60 - 70", ""]},
              {$cond: [ {$gte: ["$person.age", 70]}, "Over 70", ""]},
            ]
        },
        count: {$sum: 1},
      }},
      {$project: {
        count: 1,
        percentage: {$multiply:[{$divide: ["$count", personNumber]}, 100]}
      }},
      {$sort: {_id: 1}}
 )
```

Returns the percentage of fully vaccinated people (i.e. the ones having at least the second dose of vaccine) divided by age groups.

2.4.5 Percentage of boosters by age groups

```
// find the percentage of boosters (3rd dose) for each age group
var personNumber = db.certificate.distinct("person.ssn").length;
db.certificate_datetimes.aggregate(
      {$match: {"vaccine.doses": 3}},
      {$group: {
        _id: {
          $concat: [
              {$cond: [ {$and: [ {$gte: ["$person.age", 10]},
                                 {$lt: ["$person.age", 20]} ]}, "Under 20", ""]},
              {$cond: [ {$and: [ {$gte: ["$person.age", 20]},
                                 {$lt: ["$person.age", 40]} ]}, "20 - 40", ""]},
              {$cond: [ {$and: [ {$gte: ["$person.age", 40]},
                                 {$lt: ["$person.age", 50]} ]}, "40 - 50", ""]},
              {$cond: [ {$and: [ {$gte: ["$person.age", 50]},
                                 {$lt: ["$person.age", 60]} ]}, "50 - 60", ""]},
              {$cond: [ {$and: [ {$gte: ["$person.age", 60]},
                                 {$lt: ["$person.age", 70]} ]}, "60 - 70", ""]},
              {$cond: [ {$gte: ["$person.age", 70]}, "Over 70", ""]},
            ]
        },
        count: {$sum: 1},
      }},
      {$project: {
        count: 1,
        percentage: {$multiply:[{$divide: ["$count", personNumber]}, 100]}
      }},
      {$sort: {_id: 1}}
 )
```

Returns the percentage of people having the 3rd dose for each age group.

2.4.6 Percentage of vaccinated by age groups

```
var personNumber = db.certificate.distinct("person.ssn").length;

db.certificate_datetimes.aggregate(
      {$match: {"vaccine.doses": 1}},
      {$group: {
        _id: {
          $concat: [
              {$cond: [ {$and: [ {$gte: ["$person.age", 10]},
                                 {$lt: ["$person.age", 20]} ]}, "Under 20", ""]},
              {$cond: [ {$and: [ {$gte: ["$person.age", 20]},
                                 {$lt: ["$person.age", 40]} ]}, "20 - 40", ""]},
              {$cond: [ {$and: [ {$gte: ["$person.age", 40]},
                                 {$lt: ["$person.age", 50]} ]}, "40 - 50", ""]},
              {$cond: [ {$and: [ {$gte: ["$person.age", 50]},
                                 {$lt: ["$person.age", 60]} ]}, "50 - 60", ""]},
              {$cond: [ {$and: [ {$gte: ["$person.age", 60]},
                                 {$lt: ["$person.age", 70]} ]}, "60 - 70", ""]},
              {$cond: [ {$gte: ["$person.age", 70]}, "Over 70", ""]},
            ]
        },
        count: {$sum: 1},
      }},
      {$project: {
        count: 1,
        percentage: {$multiply:[{$divide: ["$count", personNumber]}, 100]}
      }},
      {$sort: {_id: 1}}
 )
```

Returns the percentage of people who got the 1st dose for each age group.

2.4.7 Number of people with only the first dose

```
// find the ssn of people having the first dose
var first_dose = db.certificate.find({"vaccine.doses": 1}).count()

// count number of people having the second dose
var second_dose = db.certificate.find( { "vaccine.doses":2 } ).count()

first_dose - second_dose
```

Returns how many people did the first dose but not the second.

2.4.8 Percentage of vaccinated people

```
// find total number of vaccinated people
// (i.e. the ones having at least the first dose)
var total_vaccinated = db.certificate.find({"vaccine.doses": 1}).count()

// find total number of people
var total_people = db.runCommand(
   {
      distinct: "certificate",
      key: "person.ssn",
   }
).values.length

// calculate the percentage
total_vaccinated/total_people * 100
```

Returns the percentage of vaccinated people over the total number of people.

2.4.9 Number of tests per organization

```
db.certificate_authorizedBodies.aggregate( [
{ $sortByCount: "$test.organization.name" } ] )
```

Returns the number of tests made by each organization.

2.4.10 Number of vaccines per organization

```
db.certificate_authorizedBodies.aggregate( [
{ $sortByCount: "$vaccines.organization.name" } ] )
```

Returns the number of vaccines made by each organization.

2.4.11 Number of vaccines per brand

```
db.certificate.aggregate( [ { $sortByCount: "$vaccine.vaccine_info.brand" } ] )
```

Returns the number of vaccines made for each brand:

- *'Moderna'*,
- *'Johnson & Johnson'*
- *'Pfizer-BioNTech'*
- *'AstraZeneca'*

2.4.12 Number of vaccines per type

```
db.certificate.aggregate( [ { $sortByCount: "$vaccine.vaccine_info.type" } ] )
```

Returns the number of vaccines made for each type ('Genetic' or 'Viral vector').

2.5 Commands

2.5.1 Add new vaccine

```
db.certificate.insertOne(
{
    "person": {
    "ssn": ssn,
    "name": name,
    "surname": surname,
    "birthdate": birthdate,
    "city": city,
    "phone": phone,
    "emergency_contact": {
        "name": e_name,
        "surname": e_surname,
        "phone": e_phone,
        "description": description
        }
    },
    "vaccine": {
        "vaccine_info": {
            "brand": brand,
            "type": type,
            "lot": lot,
            "production_date": prod_date
        },
    "organization": id_org,
    "doses": doses,
    "date": vaccine_date,
    "doctor": doctor
    }
});
```

Inserts a new vaccine in the database. All dates are expressed as strings in "YYYY-MM-DD" format.
It takes the following parameters:

- `ssn` *(string)*: social security number of the person;
- `name` *(string)*: name of the person;
- `surname` *(string)*: surname of the person;
- `birthdate` *(string)*: birthdate of the person;
- `city` *(city)*: city where the person lives;
- `phone` *(string)*: phone number of the person;
- `e_name` *(string)*: name of the emergency contact;

- `e_surname` *(string)*: surname of the emergency contact;
- `e_phone` *(string)*: phone of the emergency contact;
- `description` *(string)*: description of the emergency contact;
- `brand` *(string)*:brand of the vaccine;
- `type` *(string)*: type of the vaccine;
- `lot` *(int)*: lot of the vaccine;
- `prod_date` *(string)*: production date of the vaccine;
- `id_org` *(string)*: id of the organization releasing the vaccine;
- `doses` *(int)*: number of the vaccine dose;
- `vaccine_date` *(string)*: date of the vaccine;
- `doctor` *(string)*: name of the doctor attending the vaccination.

2.5.2 Delete old tests

```
db.certificate_datetimes.find({
  "test.date": {"$lte": new Date(new Date().setMonth(new Date().getMonth()-3))}
}).forEach(function(doc) {
    db.certificate.deleteMany({ "_id": doc._id });
  });
```

Delete all tests older than 3 months.

2.5.3 Update emergency contact

```
db.certificate.updateMany(
{"person.ssn": ssn},
{"$set" : {
"person.emergency_contact.name": name,
"person.emergency_contact.surname": surname,
"person.emergency_contact.phone": phone,
"person.emergency_contact.description": description
}
})
```

Updates the emergency contact of a certain person.
It takes the following parameters:

- `ssn` *(string)*: social security number of the person;
- `name` *(string)*: name of the emergency contact;
- `surname` *(string)*: surname of the emergency contact;

- `phone` *(string)*: phone of the emergency contact;
- `description` *(string)*: description of the emergency contact;

2.5.4 Update government rules

```
db.government_rules.updateOne(
          {},
          {"$set" : {
                    "negative_test_validity_time_h": negative_test_time,
                    "healed_test_validity_time_month": healed_test_time,
                    "vaccine_validity_time_month": vaccine_time
          }})
```

Updates the government rules about the validity of tests and vaccines.
It takes the following parameters:

- `test_time` *(int)*: hours for which a negative test is valid;
- `vaccine_months` *(int)*: months for which a vaccine is valid;
- `healed_test_time` *(int)*: months for which a test obtained after healing from Covid-19 is valid.

3 Application

3.1 Description

The app is split into 3 tabs:

- The first tab contains a panel where you can request the generation of a QR code for a certificate given the SSN of a person.
  You can choose to generate a certificate related to the vaccine doses' certificates, the most recent negative test or the most recent healed test.
  After generating it, a *.png* file containing the QR code is saved on your working directory where the JAR has been launched.

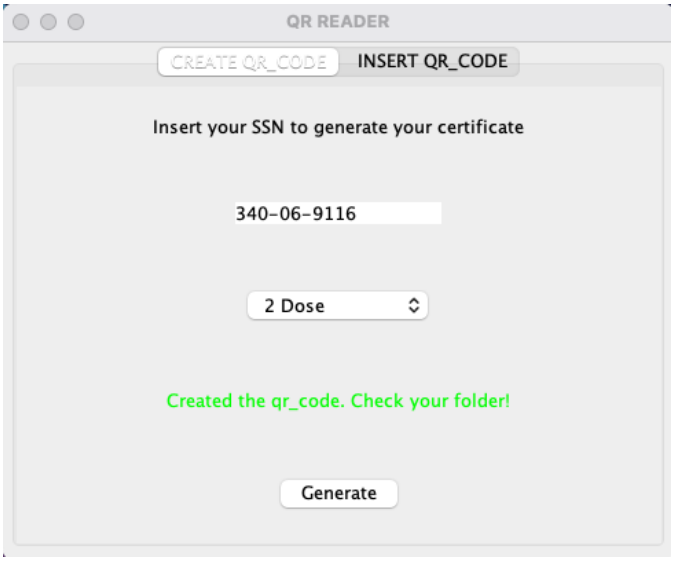
- The second tab contains a panel where you can choose an image containing a QR code and checks if the associated certificate is valid or not.
  A certificate is judged as valid if the corresponding certificate is found inside the database (retrieved using its id encoded in *base64*) and matches the validity conditions given for its certificate type.
  If the certificate is a vaccine, we check there are no newer vaccines made after it, otherwise it is not valid; then we check that the date matches the validity conditions given by the government.
  If the certificate is a test, we simply check its date according to government rules.

- (Please note that this works only if you run the application on Windows) The third tab contains a panel that integrates a webcam component capable of scanning the QR code and checks its validity.

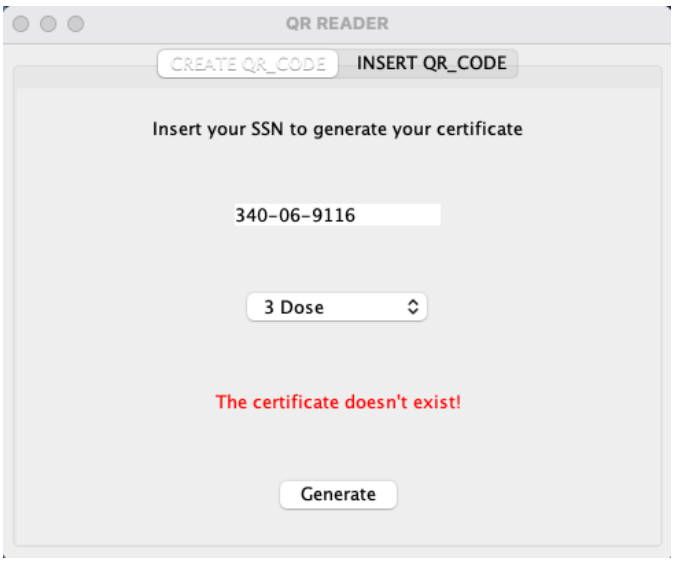
3.2 User Guide

The app is written in Java, therefore it requires Java 11 or newer versions in order to run properly.
There are two JAR files, one with the webcam version (intended to be run on Windows), and the other one without the webcam.
In order to connect to the MongoDB service a connection string must be provided in the format specified by the Java MongoDB Driver inside a file named `ConnectionString.txt` placed in the working folder of the JARs.
In addition, the database must be named *'project2'* and the collections' names must be: *"authorized_body"*, *"certificate"* and *"government_doc"*.
A default connection string is already provided along with the application but it is not guaranteed to function since it has been used for testing purposes.
The generated dataset can be imported either using MongoDB Compass and manually creating a database and importing all the collections from the files contained in `deliverables/creation/database_collections` or running a *mongorestore* using MongoDB Database Tools from the dump located at `deliverables/creation/dump`.

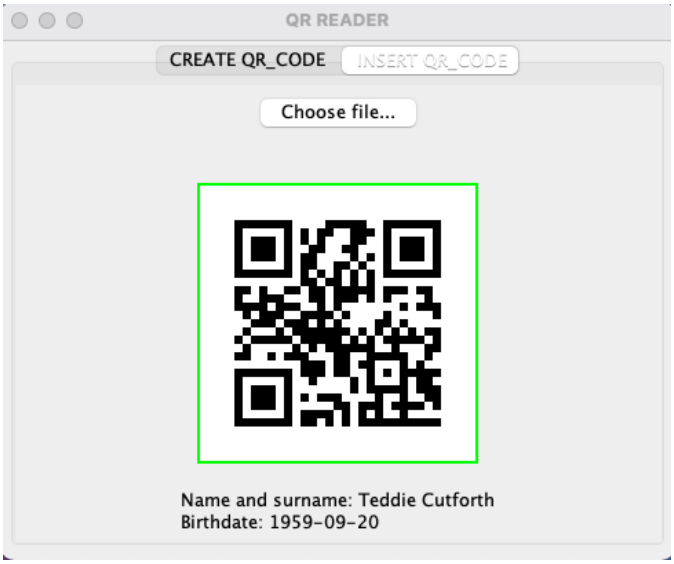
3.3 Screenshots



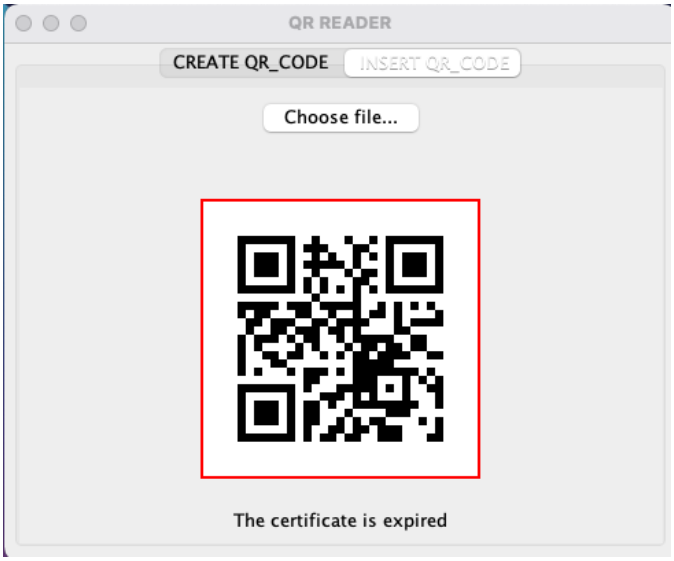
Generation of a QR code.



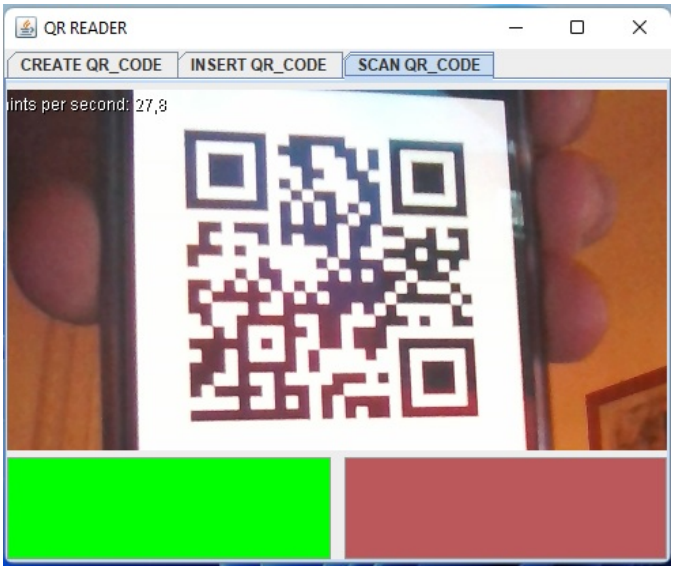
Trying to generate a QR code for a non-existing certificate.



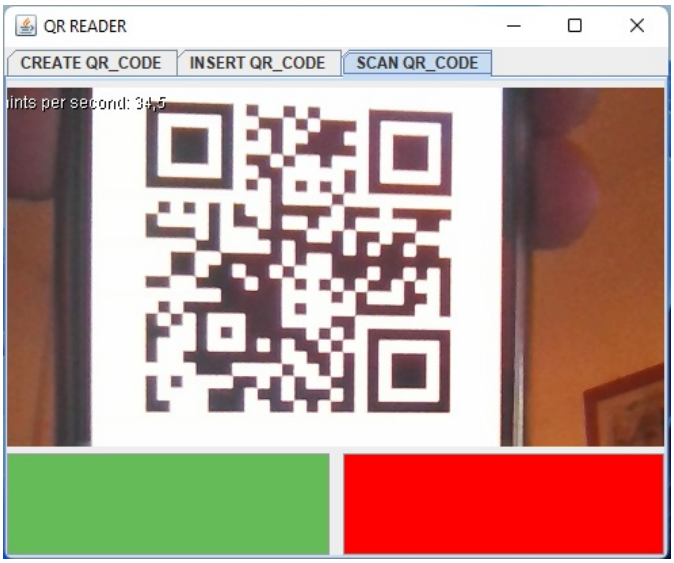
Scan of a valid QR code.



Scan of an invalid Qr code.



Scan of a valid QR code via webcam.



Scan of an invalid Qr code via webcam.

4 References and sources

In order to develop these project, the following tools were used:

- MongoDB and the MongoDB Compass to build and navigate the database and write queries;
- LaTeX to write the report;
- Github as a versioning and collaboration mean;
- Java, JavaFX, Scene builder, MongoDB Driver and the ZXing library to build the application;
- https://www.Mockaroo.com to generate realistic data for people, authorized bodies and certifications;
- link to the resource used for the pharmacy population;
- link to the resource used for the hospital population.