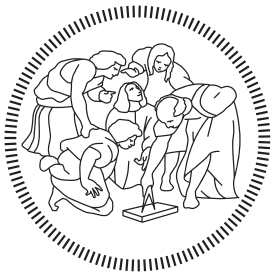
AY 2021/2022



Politecnico di Milano

SMBUD 2021 - Project work delivery 2

PoliPass

Group 20

Margherita	Musumeci	(10600069)
Matteo	Nunziante	(10670132)
Piero	Rendina	(10629696)
Andrea	Sanchini	(10675541)
Enrico	Zuccolotto	(10666354)

Professor
Marco Brambilla

January 15, 2022

Contents

1 Introduction

Since early 2020, the COVID-19 virus has represented a worldwide threat.
So far, the farmaceutical companies have been developing vaccines to tackle viral spreading. Since they represent the only way to avoid other struggling periods at home, everyone must be encouraged to get vaccinated and well informed about the risks of catching the virus. Some countries have adopted a certificate, namely the "green pass", to accomplish the goal of reaching a high percentage of vaccinated people.
In Italy, the green pass is intrinsically bound with the current regulations that allow only the green pass possessors to join restaurants, public transports, even working places. Without delving too much into the details related to this instrument, our goal is to provide an efficient way to check the certificate validity.
The project makes usage of a document-based data structure enabled by MongoDB application to store all information about people and vaccinations/tests issuers.
The database is coupled with a Graphical User Interface designed to retrieve and show the details of each certification.

2 Assumptions

The project relies on the hypotheses described below:

1. A **green pass** can be obtained through a negative test or vaccination.
2. The presence of the field **green pass** inside the document infers the actual validity of the green pass itself.
3. In the case of **green pass** absence for a vaccinated person, it means that the subject is positive.
4. The insertion of a positive test entails the **green pass** deletion (if present before). If a negative one is uploaded afterwards, it triggers the creation of a new green pass, valid for two days for a non-vaccinated person or equal[1] to the previous one otherwise.
5. To comply with current regulations: a person with a single COVID-19 Janssen Vaccine dose or double dose of Astrazeneca Vaccine will potentially have one more among Pfizer and Moderna.
6. All the different issuers can perform either vaccinations or tests.
7. Each issuer has at least one doctor and one nurse.
8. The maximum number of vaccine doses so far is **three**. They are temporally spaced: the second dose comes at least thirty days after the first, the third one, instead, is delivered after at least a hundred and fifty days after the second[2].
9. All the vaccinations and the tests are administered in the presence of both a doctor and a nurse.

[1] By saying "equal" we mean with the same expiration date and related to the same vaccination.

[2] For the single dose COVID-19 Janssen Vaccine, the following dose is considered as the third one.

3 Conceptual model of the database

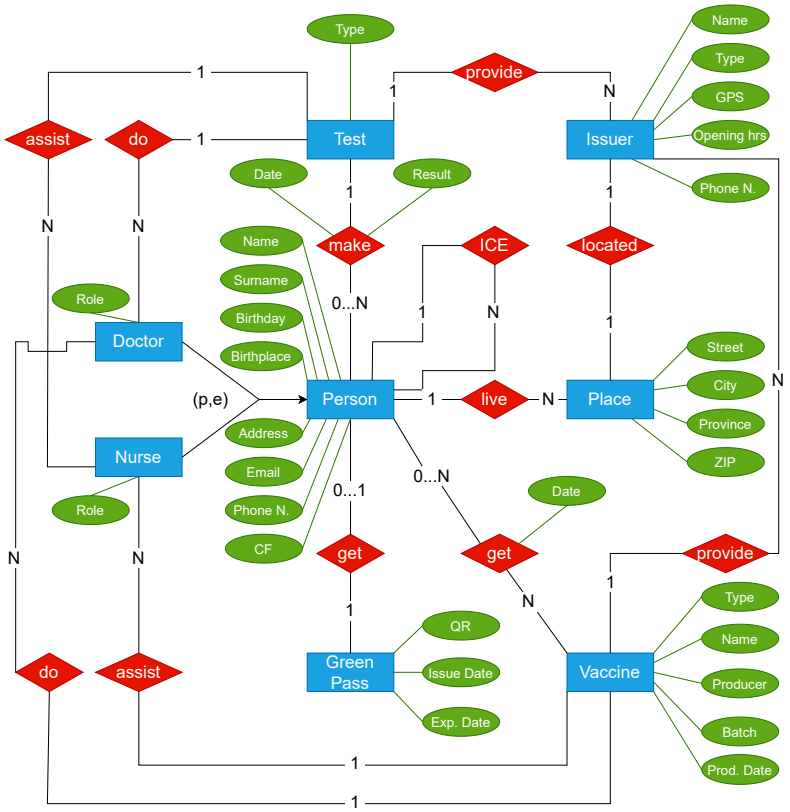


Figure 1: *Entity Relationship model*

This entity relationship diagram has been translated into JSON-like documents, which are the main components of a MongoDB database. The logical design and the detailed structure of such documents are discussed in section 4.

3.1 Primary Keys

The ID attribute is not specified among the ER attributes to gain clearness.

1. Person : ID
2. Place : ID
3. Issuer : ID
4. Green Pass : ID
5. Vaccine : [Producer , Name]
6. Test : Type
7. make(test): [personID, testType, Date, Time]
8. ICE[3] : [personID, personID]
9. get(vaccine) : [personID, vaccineType , vaccineName , Date]

[3]ICE stands for In Case Of Emergency: it describes the emergency contact possession.

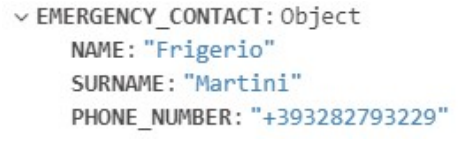
4 Dataset description

The dataset is organized into two collections.
The first one, named **covid_certificates**, contains a document for each person. The person's record includes emergency contact, email, address and personal details. The people who are either doctors or nurses have a field named ***role*** to describe their status. For the others, the occupation is deemed unnecessary. Furthermore, it stores two lists for vaccinations (if got) and tests (if taken), both ordered from the most recent one to the least recent.
There is a predominant application of embedding strategy (Figure 4) since each field is self-contained except for the ***issuer*** inside vaccination and test. The issuer field points to a document belonging to the issuers' collection through its id. Embedding is necessary to speed up the queries according to the logic of "store together what is queried together"[4]. Moreover, it is supported by the fact that we are saving historical data (test results or vaccinations are unlikely to be changed).
The second collection, named **issuers**, gathers information concerning the issuers. More specifically, each document belonging to this collection retains data such as the type, the name, the address and worthwhile details for an issuer. It is worth mentioning that, as in the first collection, the embedding technique prevails, even though we have two ids arrays containing the references to nurses and doctors. The database is randomly generated and tuned with some basic consistency checks that ensure the assumptions in section 2.

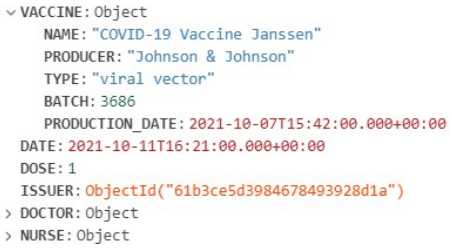
Figure 2: *Example of document stored in the* ***covid_certificates*** *collection.*

Figure 3: *Example of document stored in the* ***issuers*** *collection.*

[4]https://www.mongodb.com/community/forums/t/best-way-to-store-products-and-its-attributes-information-in-mongodb/130338



(a) *An embedded emergency contact document.*



(b) *An embedded vaccination document.*

(c) *An embedded doctor document.*

(d) *An embedded green pass document.*

Figure 4: *Embedded documents.*

4.1 Remarks

- The **password** in the person's document is encoded through a 256-SHA algorithm.
- The **QR code** field inside the embedded *green pass* document stores a binary-encoded version of the image containing the actual QR. Since the picture is not large, we have decided to put it in the document instead of relying on GridFS.

5 Queries

5.1 Find the number of vaccine doses for each person.

```
db.covid_certificates.aggregate({
$project: {FISCAL_CODE: 1, NumberOfVaccinations: {$cond: {if: {$isArray: "$VACCINATIONS"}, then: {$size: "$VACCINATIONS"}, else: 0}}}
})
```

5.2 Find the number of tests for each person.

```
db.covid_certificates.aggregate({
$project: {FISCAL_CODE: 1, NumberOfTests: {$cond: {if: {$isArray: "$TESTS"}, then: {$size: "$TESTS"}, else: 0}}}
})
```

5.3 Find all the positive people.

```
db.covid_certificates.aggregate([
{$project: {FISCAL_CODE: 1 ,lastTest: {$arrayElemAt: ["$TESTS", 0]}}},
{$match: {"lastTest.RESULT": "positive"}}
])
```

5.4 Find the five people that have been tested positive more times than everyone else.

```
db.covid_certificates.aggregate([
{$match: {TESTS: {$exists: true}}},
{$project: {FISCAL_CODE: 1, NumberOfTests: {$size: {$filter: {input: "$TESTS", as: "t",cond: {$eq: ["$$t.RESULT", "positive"]}}}}},
{$sort: {"NumberOfTests": -1}},
{$limit: 5}
])
```

5.5 Find the rate of tested positive people for each type of vaccine.

```
db.covid_certificates.aggregate([
{$match: {VACCINATIONS: {$exists: true}}},
{$project: {vaccine: {$arrayElemAt: ["$VACCINATIONS", 0]} , test: {$arrayElemAt: ["$TESTS" , 0]}}},
{$project: {vaccine: "$vaccine" , resultTest: {$cond: {if: {$eq: ["$test.RESULT" , "positive"]}, then: 1, else: 0}}}},
{$group: {_id: {name: "$vaccine.VACCINE.NAME"} , totalVaccine: {$sum: 1} , totalPositive: {$sum: "$resultTest"}}},
{$project: {name: "$name" , rate: {$divide: ["$totalPositive" , "$totalVaccine"]}}}
])
```

5.6 Find the five doctors that did the biggest number of vaccinations.

```
db.covid_certificates.aggregate([
{$project: {vaccines: "$VACCINATIONS"}},
{$unwind: "$vaccines"},
{$project: {doc: "$vaccines.DOCTOR"}},
{$group: {_id: {doc: "$doc"}, count: {$sum: 1}}},
{$sort: {count: -1}},
{$limit: 5}
])
```

5.7 Find the five nurses that administered most tests.

```
db.covid_certificates.aggregate([
{$project: {tests: "$TESTS"}},
{$unwind: "$tests"},
{$project: {nurse: "$tests.NURSE"}},
{$group: {_id: {nurse: "$nurse"}, count: {$sum: 1}}},
{$sort: {count: -1}},
{$limit: 5}
])
```

5.8 Find the percentage of at least fifty years old vaccinated people.

```
db.covid_certificates.aggregate([
{$project: {age: {$divide:[{$subtract: ["$$NOW","$BIRTHDATE"]}, 1000 * 24 * 60 * 60 * 365]},VACCINATIONS: "$VACCINATIONS"}},
{$match: {age: {$gte: 50}}},
{$project: {age: "$age" , VacPersonOverAge: { $cond: [ {$not: ['$VACCINATIONS']},0, 1 ]}}},
{$group: {_id: null ,totNumOfVaccinatedPerson: {$sum: 1}, numOfVaccinatedPersonOverAge: {$sum: "$VacPersonOverAge"}}},
{$project: {numOfVaccinatedPersonOverAge: "$numOfVaccinatedPersonOverAge" , totalNumberOfVaccinatedPerson: "$totNumOfVaccinatedPerson" ,
rate: {$divide: ["$numOfVaccinatedPersonOverAge" , "$totNumOfVaccinatedPerson"]}}}
])
```

5.9 Find tests type ordered by popularity.

```
db.covid_certificates.aggregate([
{$project: {tests: "$TESTS"}},
{$unwind: "$tests"},
{$project: {type: "$tests.TYPE"}},
{$group: {_id: {type: "$type"}, count: {$sum: 1}}},
{$sort: {count: -1}}
])
```

5.10 Find vaccines type ordered by popularity.

```
db.covid_certificates.aggregate([
{$project: {vaccine: "$VACCINATIONS"}},
{$unwind: "$vaccine"},
{$project: {type: "$vaccine.VACCINE.NAME"}},
{$group: {_id: {type: "$type"}, count: {$sum: 1}}},
{$sort: {count: -1}}
])
```

5.11 Find the emergency contacts of all the positive people.

```
db.covid_certificates.aggregate([
{$project: {FISCAL_CODE: 1,EMERGENCY_CONTACT: 1,result: {$arrayElemAt: ["$TESTS", 0]}}},
{$match: {"result.RESULT": "positive"}},
{$project: {FISCAL_CODE: 1,EMERGENCY_CONTACT: 1}}
])
```

5.12 Find all the people vaccinated with a given lot of vaccine.

```
db.covid_certificates.find({
"VACCINATIONS.VACCINE.BATCH": 13521 ,
"VACCINATIONS.VACCINE.NAME": "COVID-19 Vaccine Moderna"
})
```

5.13 Find the person with the largest number of tests without vaccinations.

```
db.covid_certificates.aggregate([
{$match: {$and: [{VACCINATIONS: {$exists: false}} , {TESTS: {$exists: true}}]}},
{$addFields: {numberOfTests: {$size: "$TESTS"}}},
{$sort: {numberOfTests: -1}} ,
{$limit: 1}
])
```

5.14 Find all the issuers in Milan province.

```
db.issuers.find({"LOCATION_DETAILS.PROVINCE":"MI"})
```

5.15 Find the number of not vaccinated people.

```
db.covid_certificates.aggregate([
{$match: {VACCINATIONS: {$exists: false}}},
{$count: "NumberOfNonVaccinatedPersons"}
])
```

5.16 Find the five youngest people with an active green pass.

```
db.covid_certificates.aggregate([
{$match:{GREEN_PASS: {$exists: true}}},
{$sort: {BIRTHDATE: -1}},
{$limit: 5}
])
```

5.17 Find all the people whose green pass will expire within one month.

```
db.covid_certificates.aggregate([
{$match:{GREEN_PASS: {$exists: true}}},
{$match: {"GREEN_PASS.EXPIRATION_DATE":{$lte: new Date(new Date().getTime()+(30*60*60*1000)) }}}
])
```

5.18 Find the mean age of all positive people.

```
db.covid_certificates.aggregate([
{$match: {$and: [{BIRTHDATE: {$exists: true}} , {TESTS: {$exists: true}}]}},
{$addFields: {result: {$arrayElemAt: ["$TESTS", 0]}}},
{$match: {"result.RESULT": "positive"}},
{$addFields: {age: {"$divide":[{$subtract: [new Date() ,"$BIRTHDATE"]}, 1000 * 24 * 60 * 60 * 365]}}},
{$group: {_id: null , Average_age: {$avg: "$age"}}},
{$project: {Average_age: "$Average_age" , _id: 0}}
])
```

5.19 Find the five issuers that the biggest number of vaccines.

```
db.covid_certificates.aggregate([
{$project: {vaccines: "$VACCINATIONS"}},
{$unwind: "$vaccines"},
{$project: {issuer: "$vaccines.ISSUER"}},
{$group: {_id: {issuer: "$issuer"}, count: {$sum: 1}}},
{$sort: {count: -1}},
{$limit: 5},
{$lookup: {from: "issuers",localField: "_id.issuer",foreignField: "_id",as: "issuer"}}
])
```

6 Commands

6.1 Insert one person's document.

```
db.covid_certificates.insertOne({
NAME: "Eudosia",
SURNAME: "Di Paola",
BIRTHDATE: new Date("1954-02-07T00:00:00.000Z"),
FISCAL_CODE: "DPLDSE54B47D072V",
BIRTH_PLACE: "Cortiglione",
PHONE_NUMBER: "+393052718550",
EMAIL: "eudosia.di paola@polipass.it",
ADDRESS: "Via Cappellini 20133 Milano MI",
EMERGENCY_CONTACT: {
NAME: "Ellera",
SURNAME: "Longo",
PHONE_NUMBER: "3409155160"
},
PASSWORD: "5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8",
ROLE: "nurse"
})
```

6.2 Drop the collection relative to the COVID-19 certificates.

```
db.drop_collection("covid_certificates")
```

6.3 Change the result of a test.

```
db.covid_certificates.updateOne (
{_id: ObjectId("61aa537c384ba2b586a9c044")},
{$set: {'TESTS.0.RESULT': "negative"}}
)
```

6.4 Insert one new vaccine's dose.

```
db.covid_certificates.findOneAndUpdate(
{_id: ObjectId("61af1962ec1830832950888b")},
{$push: {VACCINATIONS: {
$each: [{
VACCINE: {
NAME: "Vaxzevria",
PRODUCER: "AstraZeneca",
TYPE: "viral vector",
BATCH: 5920,
PRODUCTION_DATE: new Date("2021-07-28T00:00:00.000Z")
 },
DATE: new Date("2021-08-02T19:58:00.000Z")
,
DOSE: 1,
ISSUER: ObjectId("61aa537c384ba2b586a9c04d"),
DOCTOR:
{
NAME: "Maia",
SURNAME: "Prati",
FISCAL_CODE: "PRTMAI53B49C564W"
},
NURSE:
{
NAME: "Eudosia",
SURNAME: "Di Paola",
FISCAL_CODE: "DPLDSE54B47D072V"
}
}],
$sort: {DATE: -1}
}
}})
```

6.5 Insert a positive test and delete the active green pass.

```
db.covid_certificates.findOneAndUpdate(
{"_id": ObjectId("61b3101d176c7e6faaf3884f")},
{$push: {TESTS: {
$each: [{
ISSUER: ObjectId("61b3102b176c7e6faaf388c2"),
DATE: new Date ("2021-11-27T07:31:00.000Z"),
TYPE: "antibody",
RESULT: "positive",
DOCTOR: {
NAME: "Bianchina",
SURNAME: "Abbiategrasso",
FISCAL_CODE: "BBTBCH51R53G770P"
},
NURSE: {
NAME: "Onestina",
SURNAME: "Costa",
FISCAL_CODE: "CSTNTN71M67H505N"
}
}],
$sort: {DATE: -1}}},
$unset: {GREEN_PASS: ""}
})
```

7 Application

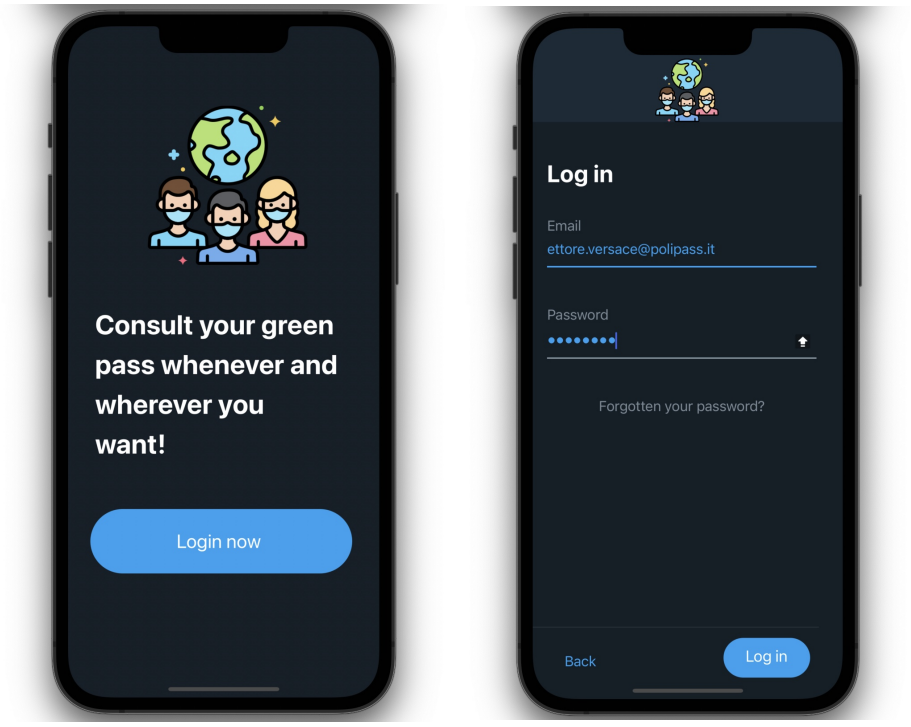
Each user, through the application, can visualize his Green Pass and the relative QR Code.
The user must be previously registered in the system, no registration method has been implemented. The issuer that records the data about tests, vaccines, etc. is assumed to also enter the data relating to the individual in the database. The application allows visualizing both vaccine Green Pass and tests Green Pass only if they are currently valid. In the case of an expired or revoked Green Pass an error message is displayed that notifies the event.

7.1 Frameworks

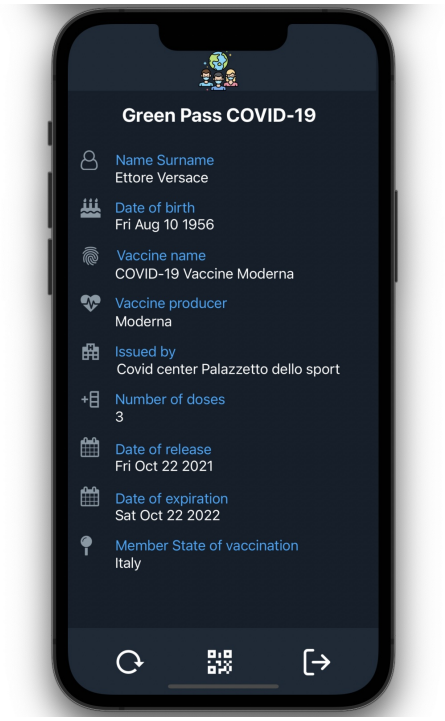
The application is developed with

1. Client : Expo - React Native
2. Server : Express - Node.js

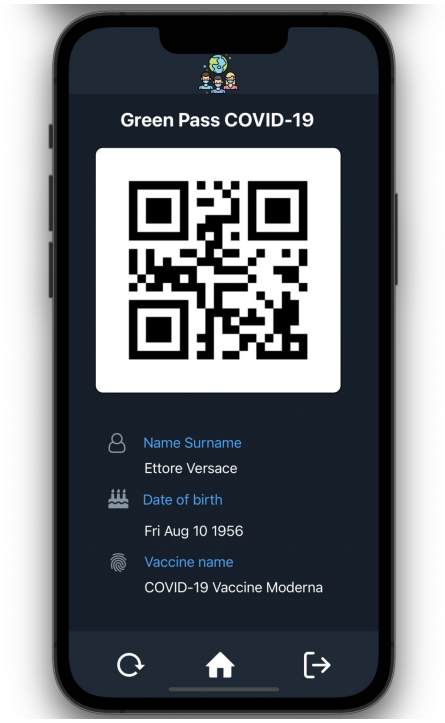
7.2 Screenshots



(a) *Front page* (b) *Login page*



(a) *Home page with Green Pass data*



(b) *QR Code related to Green Pass*

8 Resources

To access the application **email** and **password** are required.
Below there is the list of emails and respective password to sign in.

Email	Password	Features
lino.cannavaro@polipass.it	password	single dose vaccine, green pass lasting nine months
pacina.musumeci@polipass.it	password	double dose vaccinated without green pass due to the last positive swab.
ordalia.montezemolo@polipass.it	password	doctor without vaccinations and green pass
nunzio.rendina@polipass.it	password	green pass from negative test the one provided with vaccination is expired.
felicia.mazzucchelli@polipass.it	password	green pass lasting one year vaccination cycle completed

By default the *password* is set to **password** for everyone.
You can also create further accesses by yourself. You just need to look in the database, find name and surname of a person in a **covid certificate** and then sign in with the pair

name.surname@polipass.it	**password**

All the contents covered in this report are available at the link below:
https://github.com/MargheritaMusumeci/PoliPass