Software Development Process Assignment: inSalute, a Java JPA-Based Backend

Davide Mammarella

University of Milan Bicocca
d.mammarella@campus.unimib.it

Abstract. The following report discusses a telemedicine software backend solution for medical reports management that improves the life of the patient and the doctor by connecting them digitally. The mission is to provide a concrete aid to the management of one's health through a medical reports exchange platform with doctors for a simple, safe and effective experience.

Keywords: Spring Boot, Spring Data JPA, Spring Web, CRUD, Search, H2, PostgreSQL, Docker

Repository Link: `https://gitlab.com/DavideMammarella/2020_assignment3_insalute`

Contents

1 Introduction

The application present a **backend** structure that includes data persistence via **Spring Data JPA** and has also been made extensible to a possible **frontend** by adding endpoints via **Spring Web**. In order to verify the correct functioning of **CRUD** (Create, Read, Update, Delete) and **search** operations, **integration tests** for each entity are available. The same tests can be done directly by querying the application via **HTTP methods**. The application can be installed and run by following the instructions in the *README* file within the repository.

In the current state of the application we can:

- *Create, read, update, delete* reports, blood tests and covid tests (and their associated entities),
- *Create, read, update, delete* doctors (and their associated entities) and also *search* for doctors associated with a patient,
- *Create, read, update, delete* patients (and their associated entities),
- *Create, read, update, delete* health facilities and medical laboratories, including those internal to facilities (and their associated entities).

2 Responsibility of elements in the source code

2.1 Main

The **source code** can be found in *src/main/java* and the base package is *it/unimib/insalute*, which contains within it the following packages described with their responsibilities:

- **Models**: package that contains the entities[1] of the data access layer, each entity is mapped to the respective table in the database layer.
- **Repositories**: package that contains the repositories of the data access layer, each repository is mapped to the respective entity.
  - Each repository extends **JpaRepository** making available CRUD operations
- **Services**: package that contains the services of the service layer responsible for managing the application logic, each entity has its own service.
- **Controllers**: package that contains the controllers of the controller layer responsible for managing the requests coming from the view, each entity has its own controller.

Documentation detailing the code is made available at the following address:
`https://davidemammarella.gitlab.io/insalute_backend_doc/`

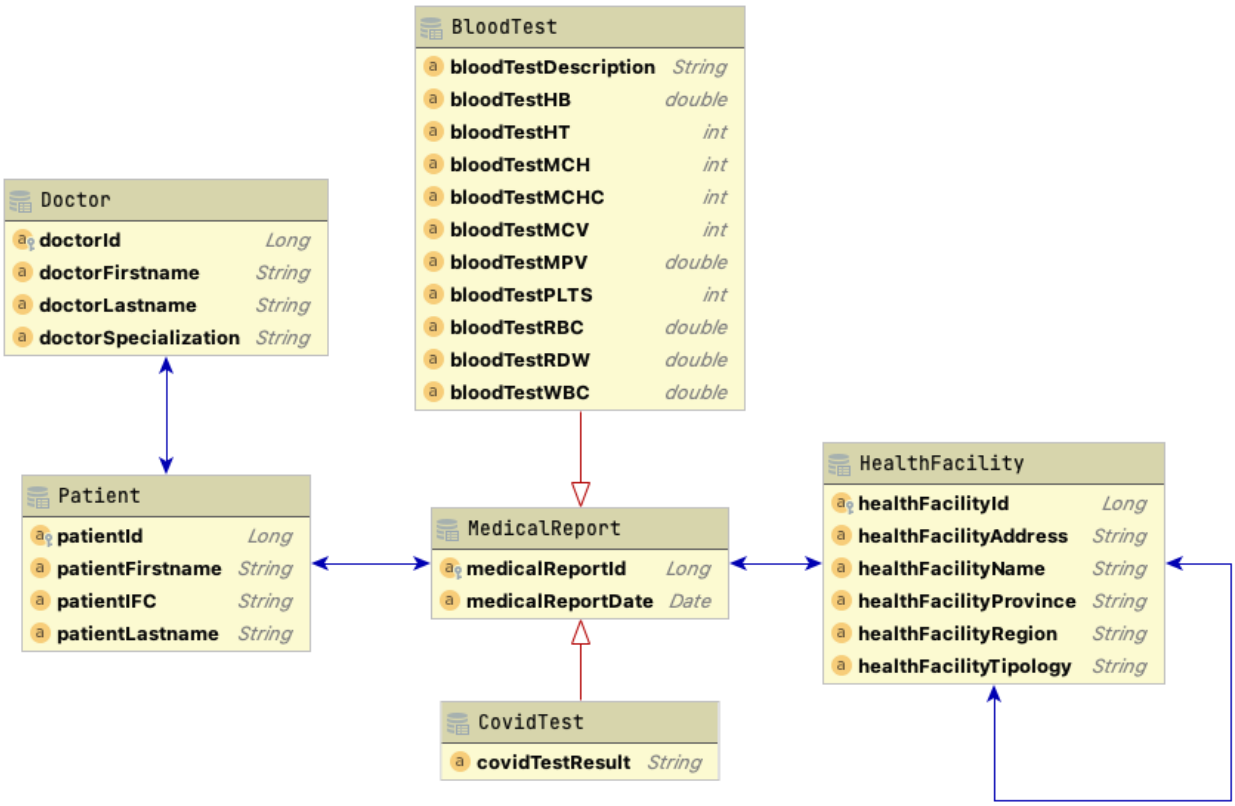

Figure 1: Entities of the data access layer.

2.2 Test

The *tests* can be found in *src/test/java* and are all **integration tests** as they allow to verify the correct mapping and functioning of the backend. In order to be able to execute them and therefore to test the data access layer, which requires a database, an in-memory database is implemented through H2. **All entities**, **their CRUD relationships** and a **search operation** are tested, including **generalization (IS-A)** and **self-loop**. To perform a countercheck of these tests, it is possible to query the application directly using the respective endpoints defined in the control layer.

3 Database

In order to obtain a clean structure and facilitate an agile methodology in which code and database development teams can work independently, two databases have been created and used according to the environment, separating the production database from the development database even though they are structurally identical.

Production Environment

In production is used a database that the application needs to function properly. Based on **PostgreSQL** as DBMS, this is the database on which the database development team works on and can be created and populated by following the instructions in the *README* file inside the `main/resources/db` folder.

Development Environment

In development is used a database that allows the development team to test the application in a safe way, without the possibility of modifying the production database. Based on **H2** as RDBMS, this database is automatically created at runtime from the entities present in the data access layer.

Parallel Development/Production Database Strategy

This strategy is based on the use of **Spring Boot Profiles** in order to define different properties depending on the environment in which the application is running, more in detail:

- **Production Database**: created using SQL and validated by **Hibernate** configuration `ddl-auto` set to `validate`. This way on every start of the application Hibernate verifies that the created schema is compliant with the entities present in the data access layer.
- **Development Database**: created at runtime using Hibernate configuration `ddl-auto` set to `create-drop`. This way on every start of the application Hibernate creates a schema based on the entities present in the data access layer and drops it on shutdown.

The described properties, together with the properties of the corresponding dialects and connection drivers, are defined in the `.properties` files located in the `resources` folders of the relative paths, or `main/resources` and `test/resources`.

3.1 Entity Relationship Diagram

The ERD[2] includes 6 entities and 5 relationships of which an IS-A generalization and a self-loop.

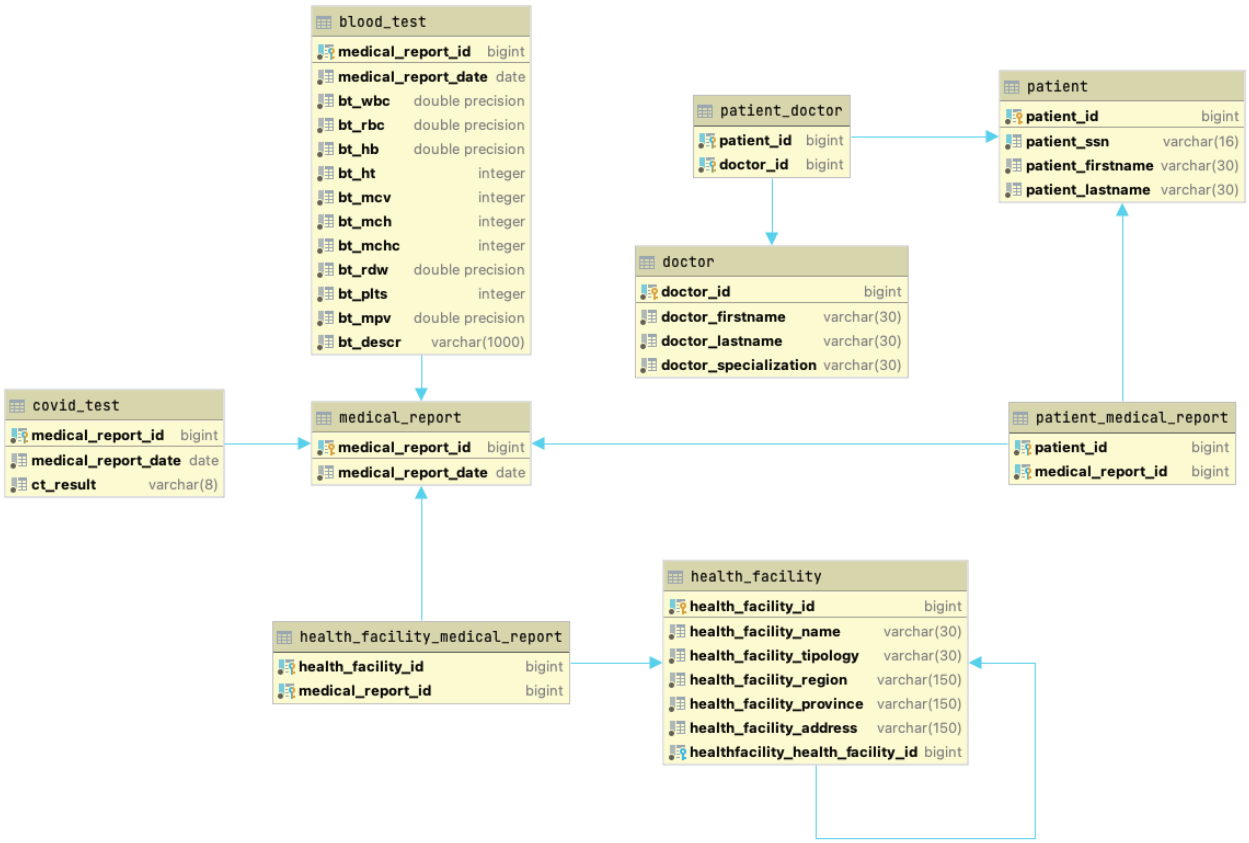

Figure 2: Entity Relationship Diagram.

In the following the entities and their relationships are described:

1. **Doctor**
   - In a **many-to-many** relationship with *Patient* defined by the *patient_doctor* table.
2. **Patient**
   - In a **many-to-many** relationship with *Doctor* defined by the *patient_doctor* table.
   - In a **many-to-many** relationship with *Medical Report* defined by the *patient_medical_report* table.
3. **Health Facility**
   - In a **Self-Loop** relationship, because an *health facility* can have several *Medical Laboratories* within it.
   - In a **many-to-many** relationship with *Medical Report* defined by the *health_facility_medical_report* table.
4. **Medical Report** (**IS-A**, parent entity)
   - In a **many-to-many** relationship with *Health Facility* defined by the *health_facility_medical_report* table.
5. **Covid Test** (**IS-A**, child entity of *Medical Report*)
6. **Blood Test** (**IS-A**, child entity of *Medical Report*)