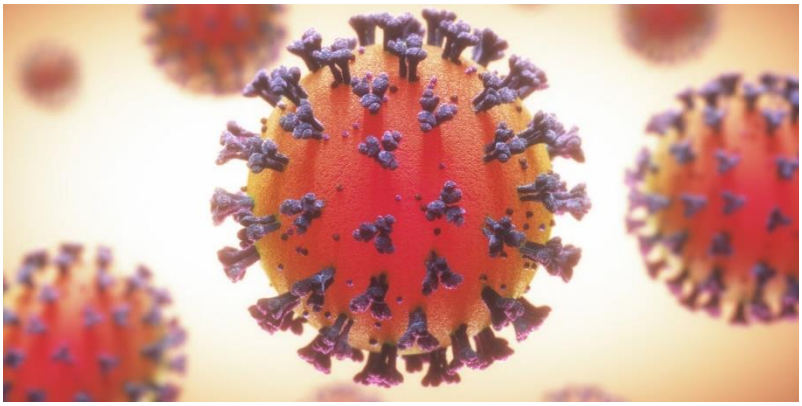
2020-2021

COVID TRACKER

DATABASE PROJECT



BY: BASILIO AGUIRRE, ENRIQUE GONZÁLEZ, CARMEN POZO, BEATRIZ LEÓN

Index:

1. System´s intent
2. Menu
3. Packages and classes
4. Interfaces
5. E-R diagram
6. Uml diagram

1. System's intent

Nowadays, we live in a pandemic, and many companies have changed in the way they work. For instance, today is really common that a great part of the staff of the business working at home, but what is essential for these companies is that many workers are in quarantine because of the virus and they lose many days at their job, therefore, the benefits shorten greatly.

Our project, CovidTracker, is a database focused on a concrete company. In this case, the intent of our system is to store the workers of the company that have been diagnosed with COVID in the database.

Then, with the patients in the database, the system will calculate the days the patient has been at home because of sick leave and after that, the benefits that the company is losing because this patient has COVID. If the person has been more than 10 days at home, there will be a message since this worker needs to be replaced.

In our project, we have defined different roles, such as the doctor or the CEO of the company, that can apply different functionalities of the database depending on their job titles. For instance, the Human Resources responsible can see if a patient needs a replacement, modify the data of a person in concrete etc.

2. Menu

We have created a main menu that connects to the data base has the option of register and log in.

```
public static void menuPrinicpal() throws Exception {
    dbman.connect();
    paman.connect();

    while (true) {
        System.out.println("\nWELCOME! ");
        System.out.println("\nChoose an option : ");
        System.out.println("1.Register ");
        System.out.println("2.Log in");
        System.out.println("0.EXIT. ");
```

Depending on the role of the user of the data base we have created different submenus, one for each role. Each one has different options that the user can choose.

1. Administrator's menu

```
private static void MenuAdm() throws Exception {
    while (true) {
        System.out.println("\n1.View a patient. ");
        System.out.println("2.View a patient from a XML file");
        System.out.println("3.Introduce a new a patient. ");
        System.out.println("4.Save a patient in a XML file ");
        System.out.println("0.EXIT. ");
        System.out.println("\nChoose an option : ");
```

2. CEO´s menu

```
private static void MenuCEO() throws Exception {
    while (true) {
        System.out.println("\n1.View a patient. ");
        System.out.println("2.Add a patient from a XML file");
        System.out.println("3.Delete a patient. ");
        System.out.println("4.Add a new doctor");
        System.out.println("5.Save a doctor in a XML file");
        System.out.println("6.View a doctor");
        System.out.println("7.Add a doctor from a XML file");
        System.out.println("8.Create an Html file for a doctor");
        System.out.println("0.EXIT. ");
        System.out.println("\nChoose an option : ");
```

3. HHRR's menu

```
private static void MenuHHRR() throws Exception {
    while (true) {
        System.out.println("\n1.Look replacement. ");
        System.out.println("2.Modify patient. ");
        System.out.println("3.View all the patient's information.");
        System.out.println("4.View all the doctor's information. ");
        System.out.println("0.EXIT. ");
        System.out.println("\nChoose an option : ");
```

4. Doctor's menu

```
private static void MenuDoc() throws Exception {
    while (true) {
        System.out.println("\n1. Introduce new patient. ");
        System.out.println("2.Save a patient in a XML file ");
        System.out.println("3.Create an Html file for a patient");
        System.out.println("4. Add a patient's covid test");
        System.out.println("5. List patients");
        System.out.println("0.EXIT. ");
        System.out.println("\nChoose an option : ");
```

5. Informatic's menu

```
private static void MenuInformatic() throws Exception {
    while (true) {
        System.out.println("\n1.Delete a user from a role. ");
        System.out.println("2.Modify the role of a user. ");
        System.out.println("0.EXIT. ");
        System.out.println("\nChoose an option : ");
```

3. Packages and classes

We have created many packages:

- Ifaces: (interfaces of our database)
  - DBManager
  - JaxbManager
  - UserManager
- Jaxb:
  - DTDChecker
  - Jaxb
- Jdbc:
  - JDBCManager
- Jpa:
  - JPAUseManager
- Pojos: (Main classes, represent the entities of our E-R diagram)
  - Covid_Test
  - Doctor
  - Patient
  - Quarantine
  - Symptoms
- Pojos.users:
  - Role
  - User
- Utils:
  - CustomErrorHandler
  - SQLDateAdapter
- Ui: (Main class with the menu)
  - InputOutput
  - Menu (main)

4. Interfaces

We have created 3 interfaces:

- **DBManager : all the SQL method's.**

```
   public void connect();
This method is used to connect to the database.

   public void disconnect();
It disconnects from the database.

   public void addPerson(Patient p);
It receives a patient and adds it to the database.

   public Patient searchPatientByName(String name);
It receives the name of the patient you want to search and returns all
information of the patient.

   public void ModifyPatient(Patient p);
It receives a patient, it allows to choose the feature of the patient the
user wants to modify and it modifies it.

   public Doctor searchDoctorbyId(int id);
It receives a doctor's id and returns the doctor that has that id in the
data base.

   public Doctor searchDoctorbyName(String name);
It receives a doctor's name and returns the doctor that has that name in
the data base.

   public void delete_patient(String name);
It receives a patient's and deletes the patient with that name.

   public void symptoms_patient(Patient p, Symptoms s);¡
It receives a patient and a symptom and adds the ids to the
patient_symptoms's table.

   public Patient test_patient(Patient pat);
It receives a patient, adds a new test to the patient and returns the
patient.

   public Date last_test(Patient patnotest);
It receives a patient and returns the data of the test to generate days
off work.

   public void viewDoctors();
It prints the names of all the doctors.

   public void quarantine_patient(Patient p, Quarantine s);
It receives a patient and a quarantine and adds the ids to the
patient_quarantine's table.

   public void dropTables();
It drops all the tables of the data base.

   public void viewPatient(int id);
It receives a patient's id and prints the name of the patient.
```

```
   public Integer searchDoctorId(String name);
```
It receives a doctor's name and returns the id of the doctor that has that name in the data base

```
public List<Integer> searchSymptomsId(Integer id);
```
It receives a patient's id and returns a list of all the symptom's ids from the patient.

```
   public String searchSymptomstype(Integer id);
```
It receives a symptom's id and returns the symptom's type.

```
   public List<Integer> searchQuarantineId(Integer id);
```
It receives a patient's id and returns a list of all the quarantine's ids from the patient.

```
   public String searchQuarantinereason(Integer id);
```
It receives a quarantine's id and returns the quarantine's reason.

```
   public void viewPatientsName();
```
It prints the names of all the doctors.

```
   public List<Doctor> viewAllDoctors();
```
It returns a list with all the doctors.

```
   public List<Patient> viewAllPatients();
```
It returns a list with all the patients.

- **JaxbManager : all XML method's**

```
public void java2XmlPAT(String filename) throws Exception;
```
It receives the name of the file where the patient is going to be stored and transforms the patient from java to Xml and saves it in the new file.

```
   public void java2XmlDOC(String filename) throws Exception;
```
It receives the name of the file where the doctor is going to be stored and transforms the doctor from java to Xml and saves it in the new file.

```
   public void simpleTransform(String sourcePath, String xsltPath,String
resultDir);
```
It creates the html file. It receives a sourcePath that is an absolute path to source xml, a file.xsltPath that is an absolute path to xslt file and a resultDir that os the directory where you want to put resulting files.

```
   public void xml2JavaPAT(String filename) throws JAXBException;
```
It receives the name of the file where the patient is stored and transforms the patient from Xml to java and adds it to the data base.

```
   public void xml2JavaDOC(String filename) throws JAXBException;
```
It receives the name of the file where the doctor is stored and transforms the doctor from Xml to java and adds it to the data base.

- **UserManager: All JPA method's**

```
   public void connect();
```
This method is used to connect to the database.

```
   public void disconnect();
```
It disconnects from the database.

```
   public void newUser(User u);
```
It receives a user and creates a new user.

```
   public void newRole(Role r);
```
It receives a role and creates a new role. The roles of the data base are administrator, CEO, HHRR, doctor and informatic.

```
   public Role getRole(int id);
```
It receives a role's id and returns the role.

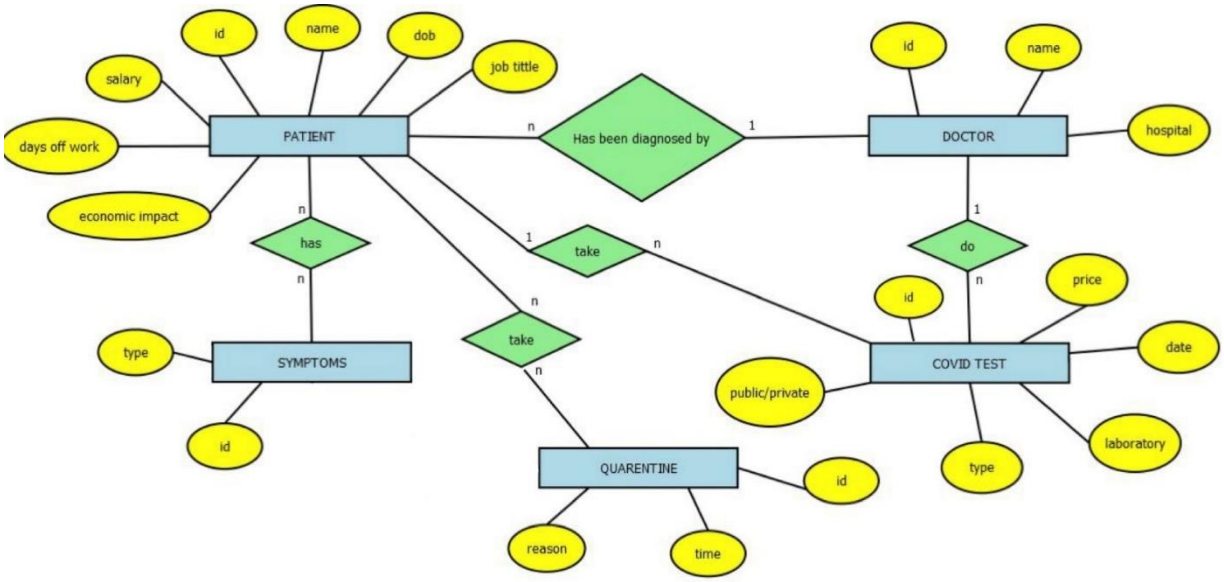
```
   public List<Role> getRoles();
```
It returns a list with all the roles.

```
   public User checkPassword(String email, String password);
```
It receives the email and the password of a user and check if the password is correct.

```
   public void deleteRole(User user);
```
It receives a user, and it deletes it.

```
   public void changeRole(User user);
```
It receives a user and modify the user's role.

5. E-R diagram



First of all, we provide a detailed information of our **patient** (the employee that has to take quarentine). This information includes his name, ID, his date of birth, the days that has to be off wok, his job tittle, salary and the economic impact that that sick leave will cause in the enterprise.

In addition, we also indicate the **doctor** who has treated the patient. Providing information about the name of the doctor, his or her ID and the hospital where he or she works.

Another entity is the one called symptoms which has two attributes: ID and type, including fever, dry cough, tiredness, pains...

This data base also includes an entity called **quarantine.** We will have access to the information of the quarantine: its ID, the reason of taking quarantine and the duration of the quarantine.

When the patient takes a **covid test**, we provide the information of that test referencing its price, ID, the type of test, the laboratory where the test has been taken, if its public or private and the date.

In terms of relations, we´ve provided two Many-To-Many relationships. The one between patient and quarantine in which ‘n’ patients can take ‘n’ quarantines and the relationship between patient and symptoms since ‘n’ patients can have ‘n’ symptoms.

Patient is also related with doctor in a One-To-Many relationship showing that a doctor can treat more than one patient. The same occurs with patient and covid test, as one patient can take more than one covid test. The last relationship is between doctor and covid test. This is also a One-To-Many relationship in which the doctor can do more than just one covid test.

6. UML Class Diagram

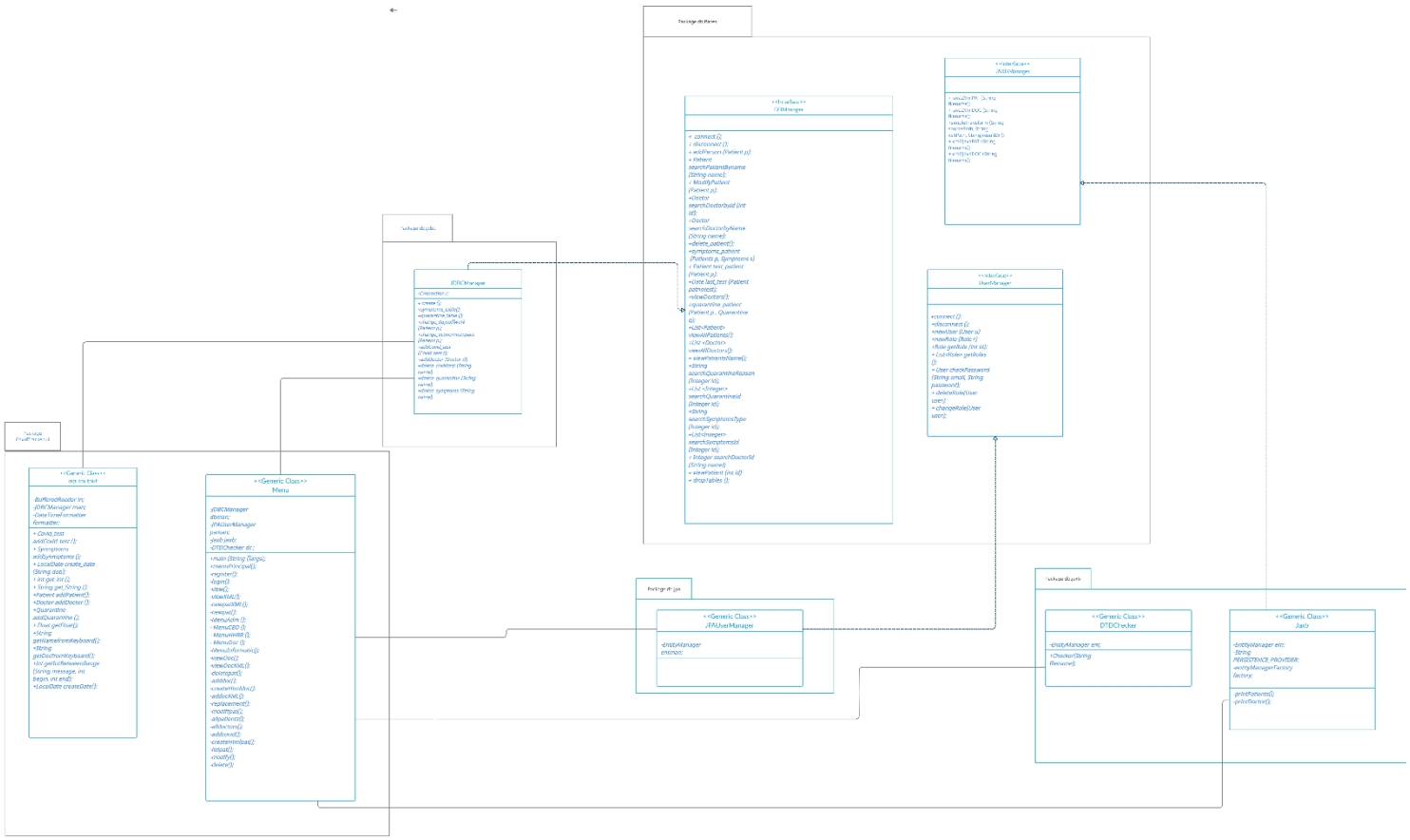
We have designed an UML diagram of our project composed of 7 packages.

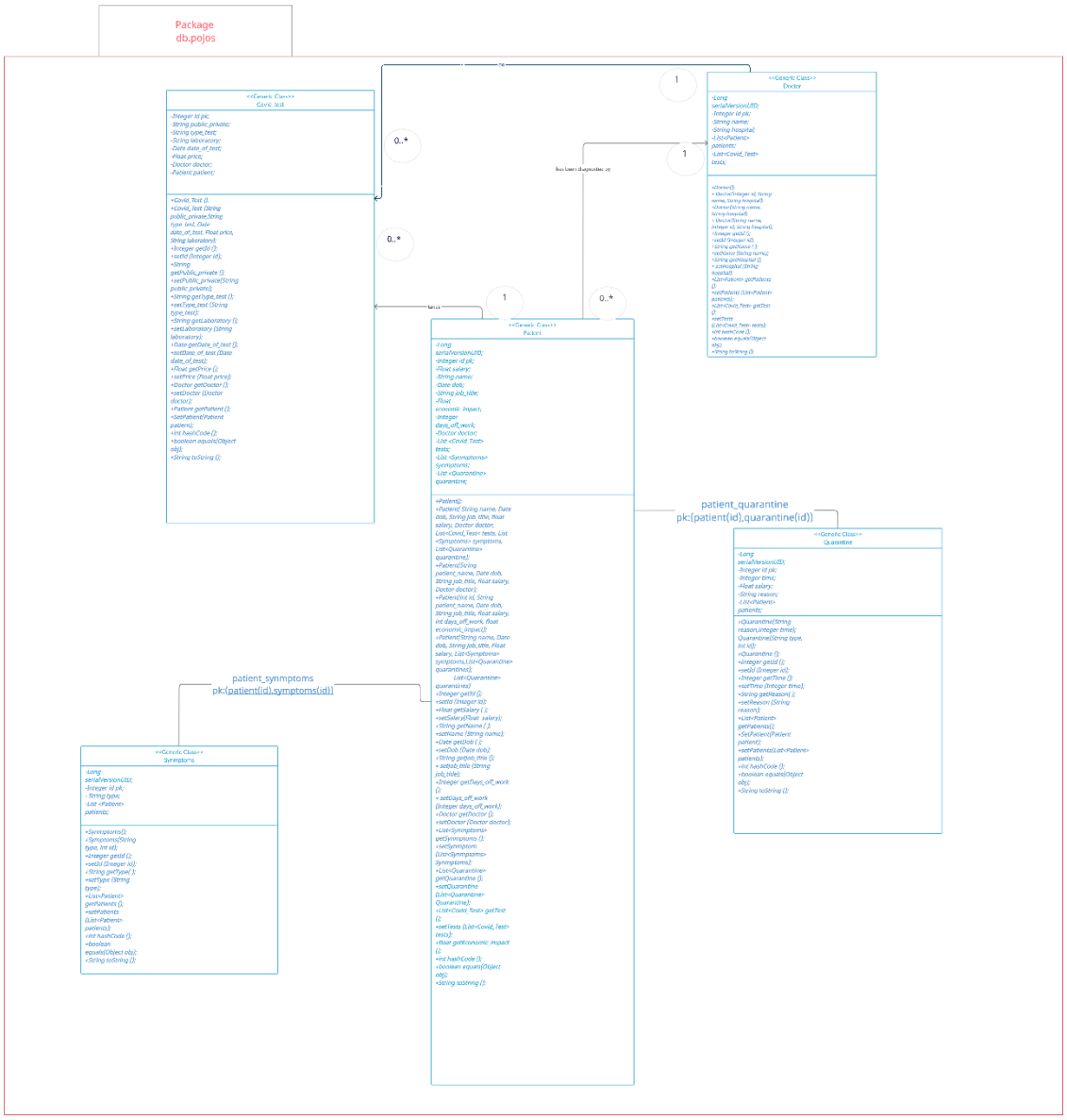
Each package is represented as a rectangle with its name at the top left corner. Each class is represented with another rectangle which is divided in three different parts: at the top, the name of the class ant its type depending on if it's an interface or a generic class; in the middle we can see attributes (and objects of other class in order to see the associations clearly); finally, at the bottom, we added the different methods of each class.

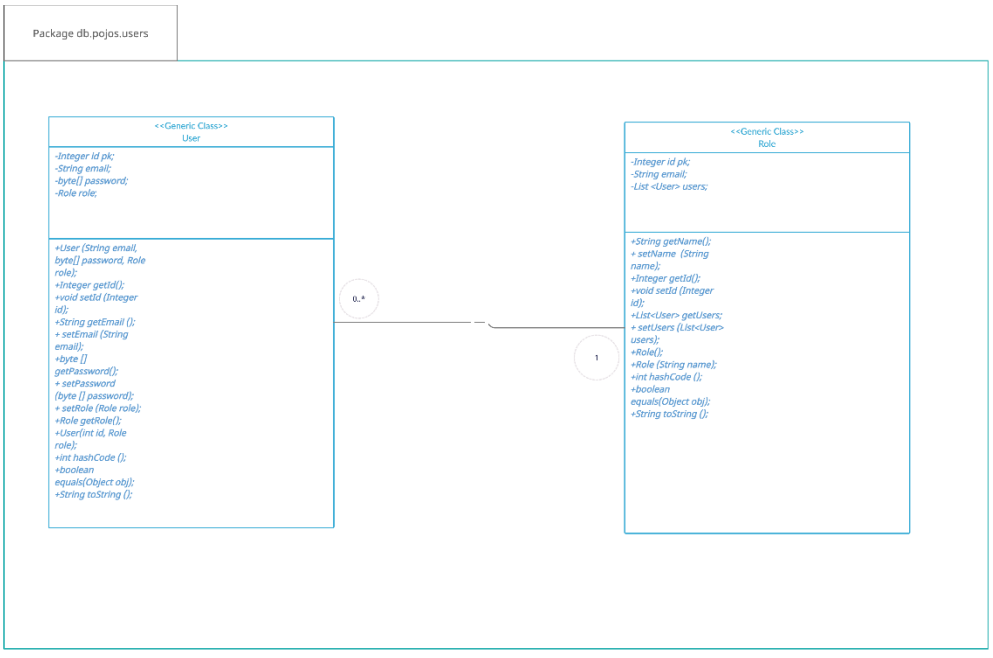
In order to represent relationships between classes, we drought lines that connect classes. If we are talking about a relation of association, we designed solid lines as for example in JDBCManager with inputoutput. In case the association has multiplicity, like in both packages of pojos, we indicated it above both ends of the line. This represents how the other part sees it. However, if the relation of association is Many-to-Many, we represented the 'association table' within the solid line indicating the name and the pk of this association table.

We also included realization. Realization is the relationship between an interface and the class that implements it. It is exposed as a dotted line with an empty narrow pointing to the interface. An example of this relation is JDBCManger with DBManager or JPAUserManager with UserManager.

*To see the Uml more clear please click here: GitHub project




Package
db.pojos
<<Generic Class>>
Covid_Test
<<Generic Class>>
Doctor
<<Generic Class>>
Patient
<<Generic Class>>
Quarantine
<<Generic Class>>
Symptoms
patient_quarantine
pk:(patient(id),quarantine(id))
patient_symptoms
pk:(patient(id),symptoms(id))


Package db.pojos.users
<<Generic Class>>
User
<<Generic Class>>
Role