NAOUR BOUTAINA

JEE : ORM, JPA, HIBERNATE, SPRING DATA

4-ème année IIR G21

Emsi les orangers

Compte-rendu

Introduction

Énoncé

Conception

Code source

Captures écran de l’exécution

Conclusion

INTRODUCTION

Onglet 1

Dans cette partie on verra pourquoi il faut utiliser des Framework ORM . En effet le but d'Hibernate est de libérer le développeur de 95 % des tâches de programmation liées à la persistance des données communes. Il assure la portabilité de votre application si vous changer de SGBD. Ainsi propose au développeur des méthodes d’accès aux bases de données plus efficace ce qui devrait rassurer les développeurs.

ÉNONCE

Objectif : Créer une application qui permet de gérer des patients en utilisant Spring Boot,Spring Data,jpa ,hibernate ,H2 ,MySql.

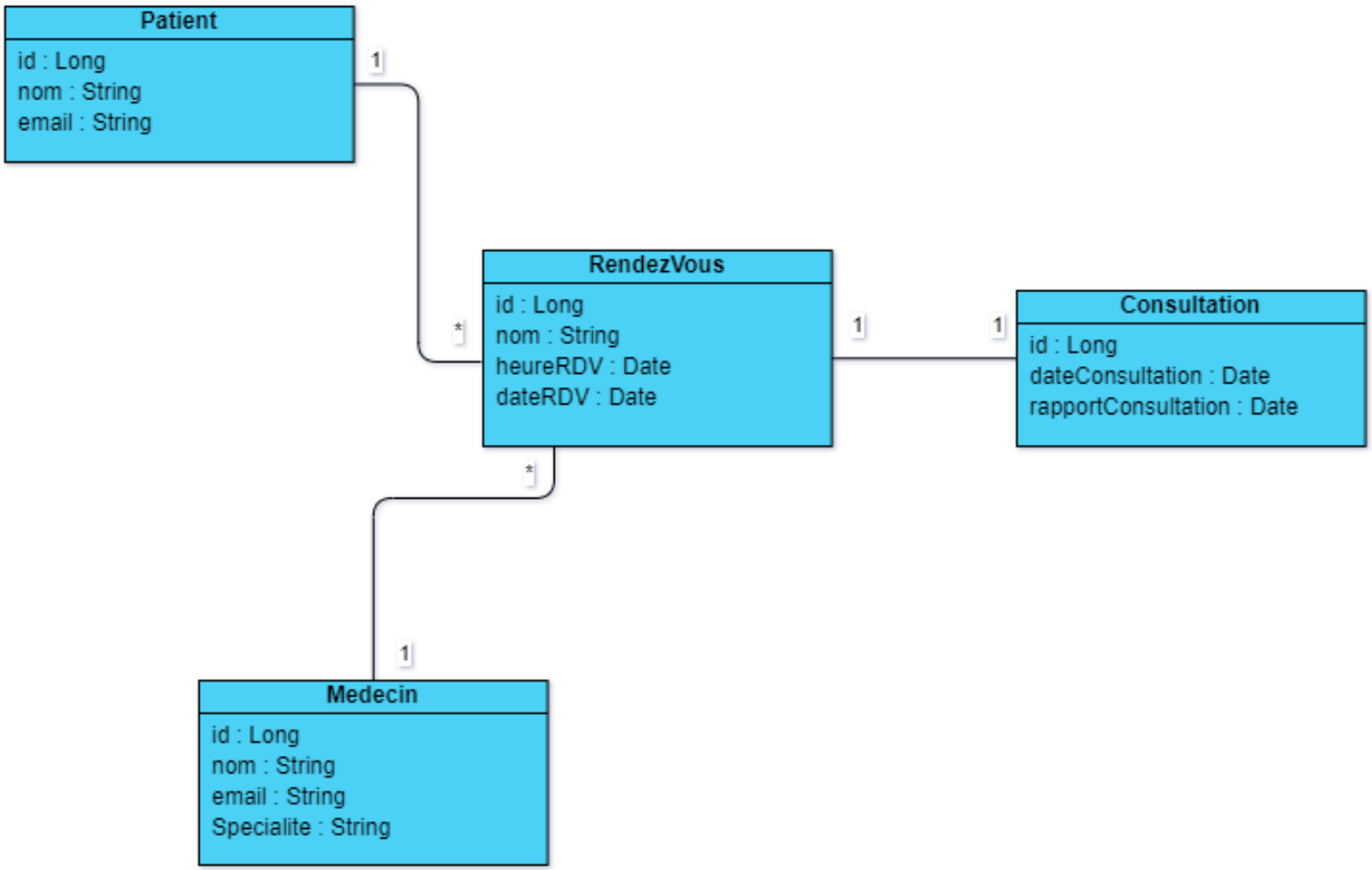
1. On souhaite gérer les rendez-vous des consultations des patients effectuées par des médecins.

2. Chaque Rendez-vous concerne un patient et un médecin.

3. Pour chaque rendez-vous on associe une seule consultation issue de rendez-vous.

4. Un patient peut prendre plusieurs rendez vous

CONCEPTION


Patient
id : Long
nom : String
email : String
1
*
RendezVous
id : Long
nom : String
heureRDV : Date
dateRDV : Date
1
1
Consultation
id : Long
dateConsultation : Date
rapportConsultation : Date
*
1
Medecin
id : Long
nom : String
email : String
Specialite : String

CODE SOURCE

Lien du code source en GitHub :

PARTIE 1 : (Cas d'une seule entité JPA)

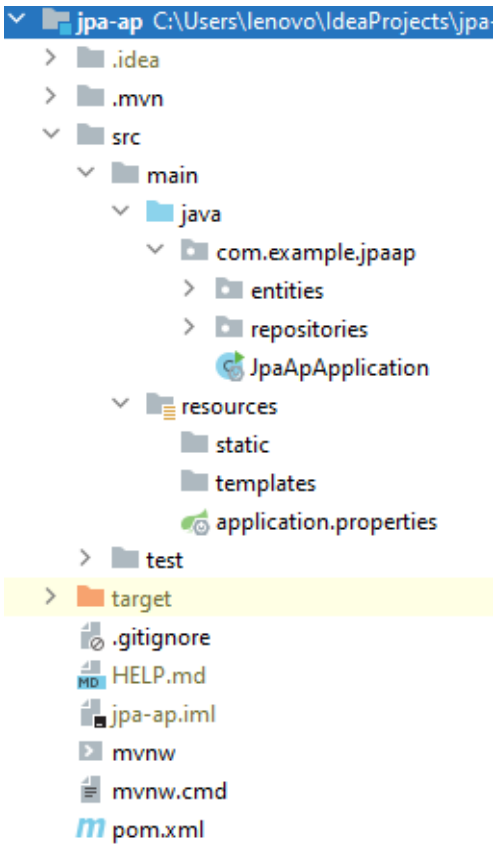
https://github.com/bout-aina/jee_tp/tree/main/jpa-ap

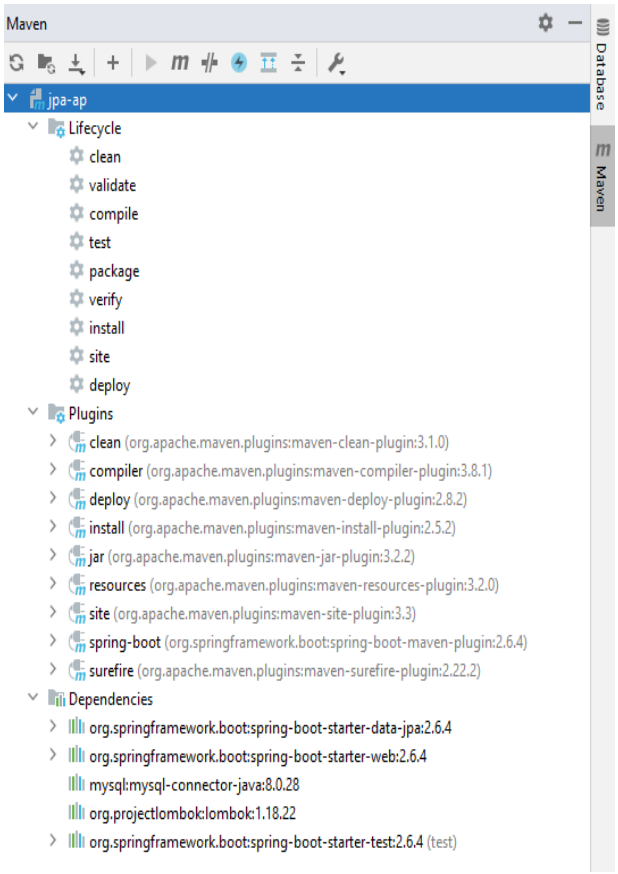
PARTIE 2 : (Cas de plusieurs entités JPA avec relations)

https://github.com/bout-aina/jee_tp/tree/main/hospital

SCREEN SHOTS

A. Création d'un projet Spring Boot : Structure du projet





B. Application.properties:

Onglet 5

```
#spring.datasource.url=jdbc:mysql://localhost:3306/DBP?createDatabaseIfNotExist=true
spring.datasource.url=jdbc:mysql://localhost:3306/PAT
spring.datasource.username=root
spring.datasource.password=
#spring.h2.console.enabled=true
server.port=8082
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect =org.hibernate.dialect.MariaDBDialect
spring.jpa.show-sql=true
```

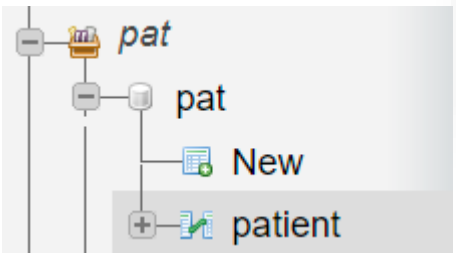
SCREEN SHOTS

C. Démarrer l'application Spring:

```
@Autowired
private PatientRepository patientRepository;

public static void main(String[] args) {

    SpringApplication.run(JpaApApplication.class, args);

}

@Override
public void run(String... args) throws Exception {

    for (int i = 0; i <100 ; i++)
        patientRepository.save(
                new Patient( id: null, nom: "boutaina", new Date(), Math.random()>0.5?true:false, (int) (Math.random()*100)));
```

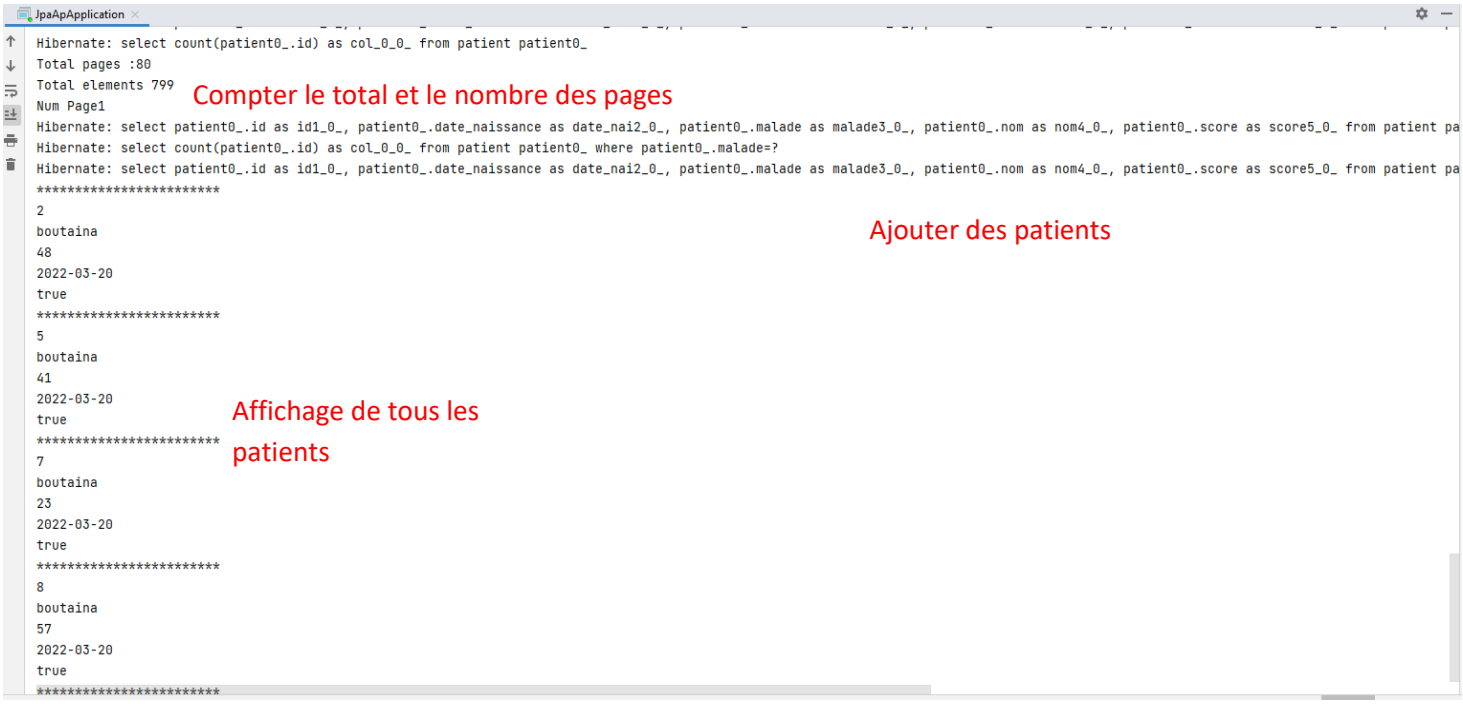
```
"C:\Program Files\Java\jdk1.8.0_301\bin\java.exe" ...

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.6.4)

2022-03-28 20:21:37.569  INFO 11884 --- [           main] com.example.jpaap.JpaApApplication       : Starting JpaApApplication using Java 1.8.0_301 on DESKTOP-5MBLE4U with PID 11884 (C:\
2022-03-28 20:21:37.569  INFO 11884 --- [           main] com.example.jpaap.JpaApApplication       : No active profile set, falling back to 1 default profile: "default"
2022-03-28 20:21:39.132  INFO 11884 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2022-03-28 20:21:39.288  INFO 11884 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 105 ms. Found 1 JPA repository interface
2022-03-28 20:21:41.038  INFO 11884 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8082 (http)
```

Structure de la table Produits générée



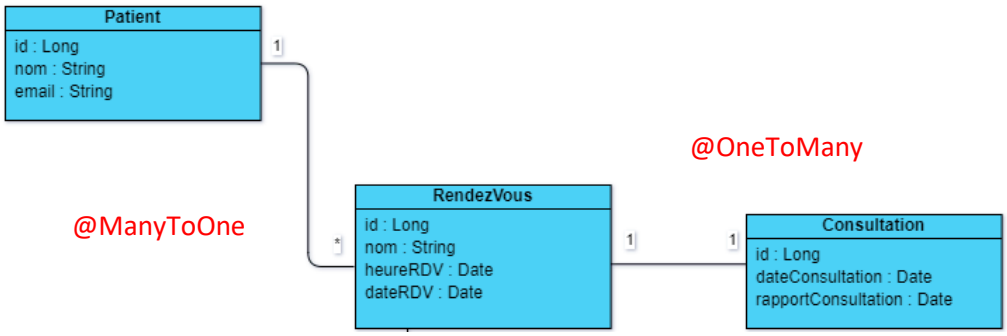
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	bigint(20)			No	None		AUTO_INCREMENT
2	date_naissance	date			Yes	NULL		
3	malade	bit(1)			No	None		
4	nom	varchar(50)	utf8mb4_general_ci		Yes	NULL		
5	score	int(11)			No	None		

⇨Tester les méthodes



⇨ CAS D'UNE ASSOCIATION DE TYPE ONE TO MANY

Diagramme de classes : Modèle Objet



Patient.java

```
13  @AllArgsConstructor
14
15  public class Patient {
16      @Id
17      @GeneratedValue(strategy = GenerationType.IDENTITY)
18
19      private Long id;
20      private String nom;
21      @Temporal(TemporalType.DATE)
22      private Date dateNaissance;
23      private boolean malade;
24      @OneToMany(mappedBy = "patient",fetch = FetchType.LAZY)
25      private Collection<RendezVous> rendezVous;
26
27  }
```

SCREEN SHOTS

Medecin.java

```
@AllArgsConstructor
public class Medecin {
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)

    private Long id;
    private String nom;
    private String email;
    private String specialite;
    @OneToMany(mappedBy = "medecin", fetch = FetchType. LAZY)
@JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
            private Collection<RendezVous> rendezVous;
}
```

RendezVous.java

```
import java.util.Date;
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class RendezVous {
    @Id

    private String id;
    private Date date;
    @Enumerated(EnumType.STRING)
    private StatusRDV status;
    @ManyToOne
    @JsonProperty(access =  JsonProperty.Access.WRITE_ONLY)
    private Patient patient;
```

⇨ Implémentation de l'interface IHospitalService :

HospitalServiceImpl.java

```java
@Service
@Transactional
public class HospitalServiceImpl implements IHospitalService {
    private PatientRepository patientRepository;
    private MedecinRepository medecinRepository;
    private RendezVousRepository rendezVousRepository;
    private ConsultationRepository consultationRepository;

    public HospitalServiceImpl(PatientRepository patientRepository,
                               MedecinRepository medecinRepository,
                               RendezVousRepository rendezVousRepository,
                               ConsultationRepository consultationRepository) {
        this.patientRepository = patientRepository;
        this.medecinRepository = medecinRepository;
        this.rendezVousRepository = rendezVousRepository;
        this.consultationRepository = consultationRepository;
    }

    @Override
    public Patient savePatient(Patient patient) { return patientRepository.save(patient); }

    @Override
    public Medecin saveMedecin(Medecin medecin) { return medecinRepository.save(medecin); }

    @Override
    public RendezVous saveRDV(RendezVous rendezVous) {
        rendezVous.setId(UUID.randomUUID().toString());


        return rendezVousRepository.save(rendezVous);
    }

    @Override
    public Consultation saveConsultation(Consultation consultation) {
```

IHospitalService.java

```java
package com.example.hospital.service;

import com.example.hospital.entities.Consultation;
import com.example.hospital.entities.Medecin;
import com.example.hospital.entities.Patient;
import com.example.hospital.entities.RendezVous;

public interface IHospitalService   {
    Patient savePatient(Patient patient);
    Medecin saveMedecin(Medecin medecin);
    RendezVous saveRDV (RendezVous rendezVous);
    Consultation saveConsultation (Consultation consultation);
}
```

⇨ Creation d’un conntrolleur de patients pour lister tous les patients sous format json

```java
package com.example.hospital.web;

import com.example.hospital.entities.Patient;
import com.example.hospital.repositories.PatientRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;
@RestController
public class PatientRestController {
    @Autowired
    private PatientRepository patientRepository;
    @GetMapping("/patients")
    public List<Patient> patientlist() { return patientRepository.findAll(); }
}
```

localhost:8086/patients


```
[
  - {
        id: 1,
        nom: "Mohamed",
        dateNaissance: "2022-03-29",
        malade: false,
      - rendezVous: [
          - {
                id: "44f8bb61-6b84-404c-bc63-14cf82e93c05",
                date: "2022-03-29T21:53:26.596+00:00",
                status: "PENDING",
              - medecin: {
                    id: 1,
                    nom: "boutaina",
                    email: "boutaina@gmail.com",
                    specialite: "Cardio",
                },
              - consultation: {
                    id: 1,
                    dateConsultation: "2022-03-29T21:53:26.628+00:00",
                    rapport: "Rapport de la consultion ....",
                },
            }
        ],
    },
  - {
        id: 2,
        nom: "Hassan",
        dateNaissance: "2022-03-29",
        malade: false,
        rendezVous: [ ],
    },
```


[0].rendezVous[0].medecin

⇨ Execution de l'application

HospitalApplication.java

```
CommandLineRunner start(IHospitalService hospitalService,
                        PatientRepository patientRepository,
                        RendezVousRepository rendezVousRepository,
                        MedecinRepository medecinRepository) {
    return args -> {
        Stream.of("Mohamed", "Hassan", "Najat").forEach(name -> {
            Patient patient = new Patient();
            patient.setNom(name);
            patient.setDateNaissance(new Date());
            patient.setMalade(false);
            hospitalService.savePatient(patient);
        });
        Stream.of("boutaina", "aymane", "souad").forEach(name -> {
            Medecin medecin = new Medecin();
            medecin.setNom(name);
            medecin.setEmail(name+"@gmail.com");
            medecin.setSpecialite(Math.random()>0.5?"Cardio":"Dentiste");
            hospitalService.saveMedecin(medecin);
        });
        Patient patient=patientRepository.findById(1L).orElse( other: null);
        Patient patient1=patientRepository.findByNom( name: "Mohamed");

        Medecin medecin=medecinRepository.findByNom("boutaina");
        RendezVous rendezVous=new RendezVous ();
        rendezVous.setDate(new Date());
        rendezVous.setStatus(StatusRDV.PENDING);
        rendezVous.setMedecin(medecin);
        rendezVous.setPatient (patient);
        RendezVous saveDRDV =hospitalService.saveRDV(rendezVous);
        System.out.println(saveDRDV.getId());
        RendezVous rendezVous1=rendezVousRepository.findAll().get(0);
        Consultation consultation=new Consultation();
        consultation.setDateConsultation(new Date());
```

Git | Run | TODO | Problems | Profiler | Build | Dependencies | Spring | Terminal

All files are up-to-date (2 minutes ago)

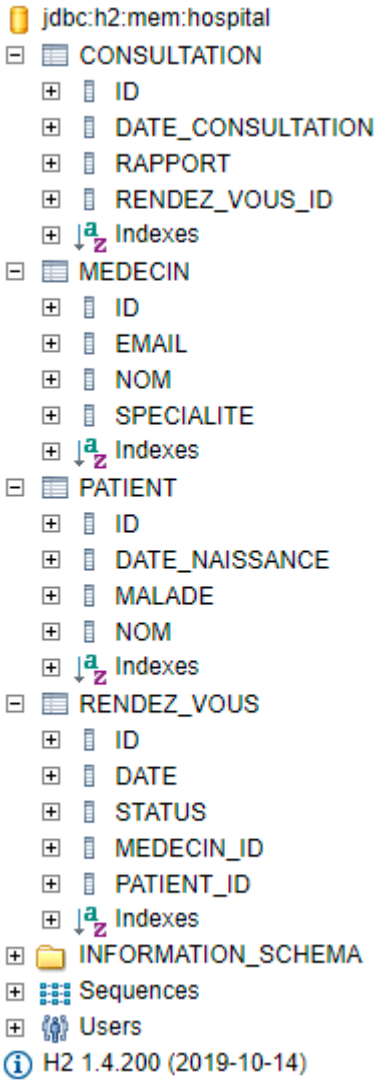
application.properties

```
spring.datasource.url=jdbc:h2:mem:hospital
spring.h2.console.enabled=true
server.port=8086
```

Run: HospitalApplication

Console | Actuator

```
 :: Spring Boot ::                (v2.6.4)

2022-03-29 21:58:57.900  INFO 944 --- [           main] c.example.hospital.HospitalA
2022-03-29 21:58:57.903  INFO 944 --- [           main] c.example.hospital.HospitalA
2022-03-29 21:58:59.534  INFO 944 --- [           main] .s.d.r.c.RepositoryConfigura
2022-03-29 21:58:59.619  INFO 944 --- [           main] .s.d.r.c.RepositoryConfigura
2022-03-29 21:59:00.318  INFO 944 --- [           main] o.s.b.w.embedded.tomcat.Tomc
2022-03-29 21:59:00.328  INFO 944 --- [           main] o.apache.catalina.core.Stand
2022-03-29 21:59:00.328  INFO 944 --- [           main] org.apache.catalina.core.Sta
2022-03-29 21:59:00.448  INFO 944 --- [           main] o.a.c.c.C.[Tomcat].[localhos
2022-03-29 21:59:00.448  INFO 944 --- [           main] w.s.c.ServletWebServerApplic
2022-03-29 21:59:00.488  INFO 944 --- [           main] com.zaxxer.hikari.HikariData
2022-03-29 21:59:00.633  INFO 944 --- [           main] com.zaxxer.hikari.HikariData
2022-03-29 21:59:00.643  INFO 944 --- [           main] o.s.b.a.h2.H2ConsoleAutoConf
2022-03-29 21:59:00.803  INFO 944 --- [           main] o.hibernate.jpa.internal.uti
2022-03-29 21:59:00.863  INFO 944 --- [           main] org.hibernate.Version
2022-03-29 21:59:01.088  INFO 944 --- [           main] o.hibernate.annotations.comm
2022-03-29 21:59:01.198  INFO 944 --- [           main] org.hibernate.dialect.Dialec
2022-03-29 21:59:01.933  INFO 944 --- [           main] o.h.e.t.j.p.i.JtaPlatformIni
2022-03-29 21:59:01.948  INFO 944 --- [           main] j.LocalContainerEntityManage
2022-03-29 21:59:02.433  WARN 944 --- [           main] JpaBaseConfiguration$JpaWebC
2022-03-29 21:59:02.838  INFO 944 --- [           main] o.s.b.w.embedded.tomcat.Tomc
2022-03-29 21:59:02.853  INFO 944 --- [           main] c.example.hospital.HospitalA
65254485-3557-476a-b9cb-03550994be41
```



Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM CONSULTATION
```

SELECT * FROM CONSULTATION;

ID	DATE_CONSULTATION	RAPPORT	RENDEZ_VOUS_ID
1	2022-03-29 21:59:03.093	Rapport de la consultion	65254485-3557-476a-b9cb-03550994be41

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM MEDECIN
```

SELECT * FROM MEDECIN;

ID	EMAIL	NOM	SPECIALITE
1	boutaina@gmail.com	boutaina	Dentiste
2	aymane@gmail.com	aymane	Dentiste
3	souad@gmail.com	souad	Dentiste

(3 rows, 8 ms)

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM PATIENT
```

SELECT * FROM PATIENT;

ID	DATE_NAISSANCE	MALADE	NOM
1	2022-03-29	FALSE	Mohamed
2	2022-03-29	FALSE	Hassan
3	2022-03-29	FALSE	Najat

(3 rows, 0 ms)

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM RENDEZ_VOUS
```

SELECT * FROM RENDEZ_VOUS;

ID	DATE	STATUS	MEDECIN_ID	PATIENT_ID
65254485-3557-476a-b9cb-03550994be41	2022-03-29 21:59:03.063	PENDING	1	1

(1 row, 0 ms)

CONCLUSION

Alors dans cette partie on a pu de gérer les associations suivantes

- @OneToMany et @ManyToOne ainsi que @OneToOne

L' utilisation de JPA permet à l' application d' être indépendante du Framework ORM utilisé.