Nom : EL garti Fatima

Atelier 2 :ORM,JPA,Hibernate,Spring Data

1. Introduction :

Dans cet atelier nous avons créé une application pour la gestion d'un hôpital avec spring boot , spring data , jpa , hibernate et h2 database .

2. Enoncé :

Partie 1 :

1. Installer Intellij Ultimate
2. Créer projet Spring Boot avec les dépendances :
   - Spring data JPA
   - Web
   - Lombok
   - H2 database
3. Créer l'entité JPA Patient
4. Créer l'interface patientRepository basée sur springData
5. Configurer la data source dans application.properties
6. Tester quelques opérations de la couche DAO pour :
   - Ajouter des patients
   - Afficher les patients
   - Consulter un patient
   - Mettre à jour un patient
   - Supprimer un patient
   - Chercher des patients avec différents critères
   - Tester la pagination
   - Migrer l'application de H2 vers MySQL

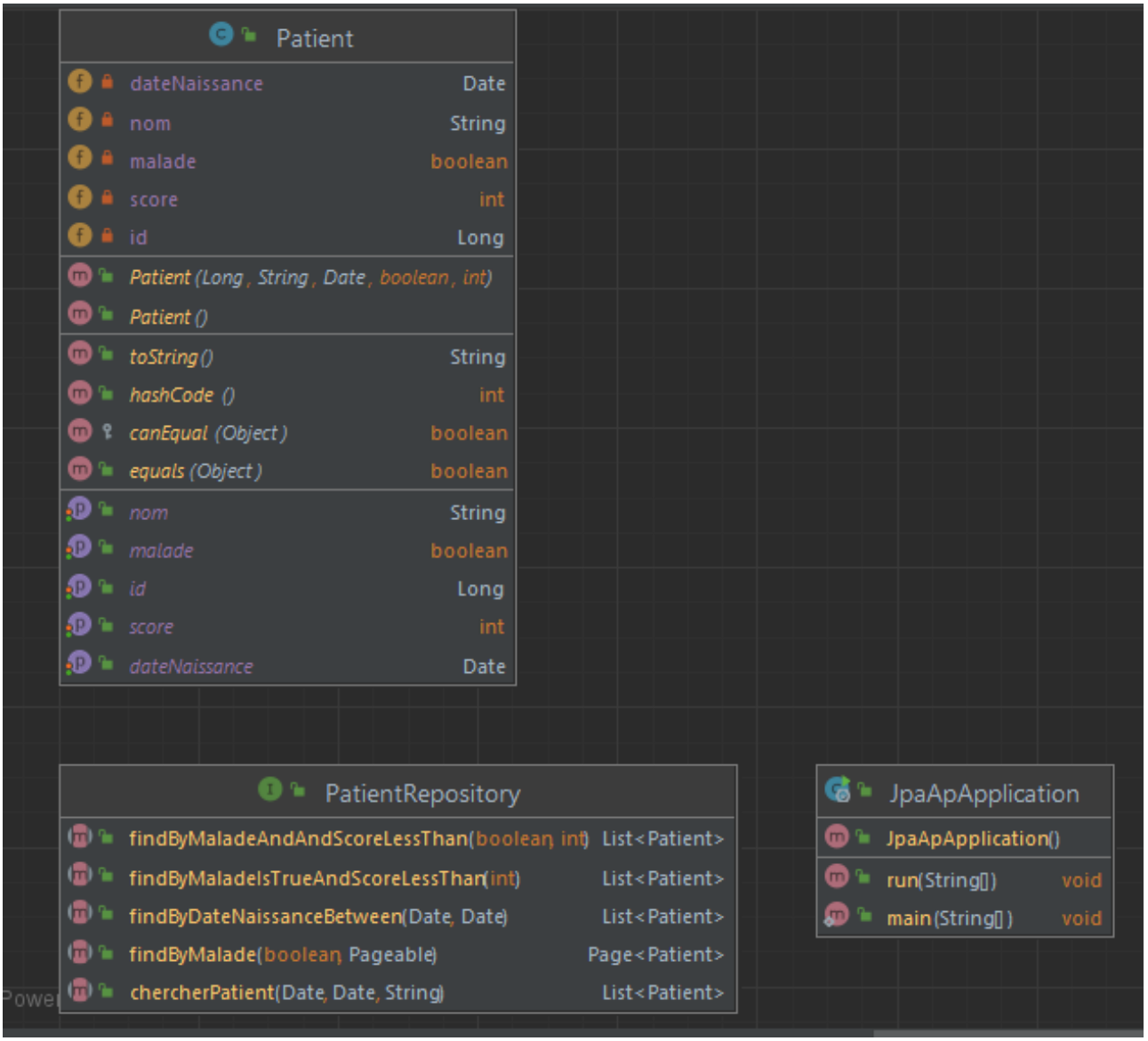
Partie 2 :

1. Créer les entités :
   a. Patient
   b. Medecin
   c. Consultation

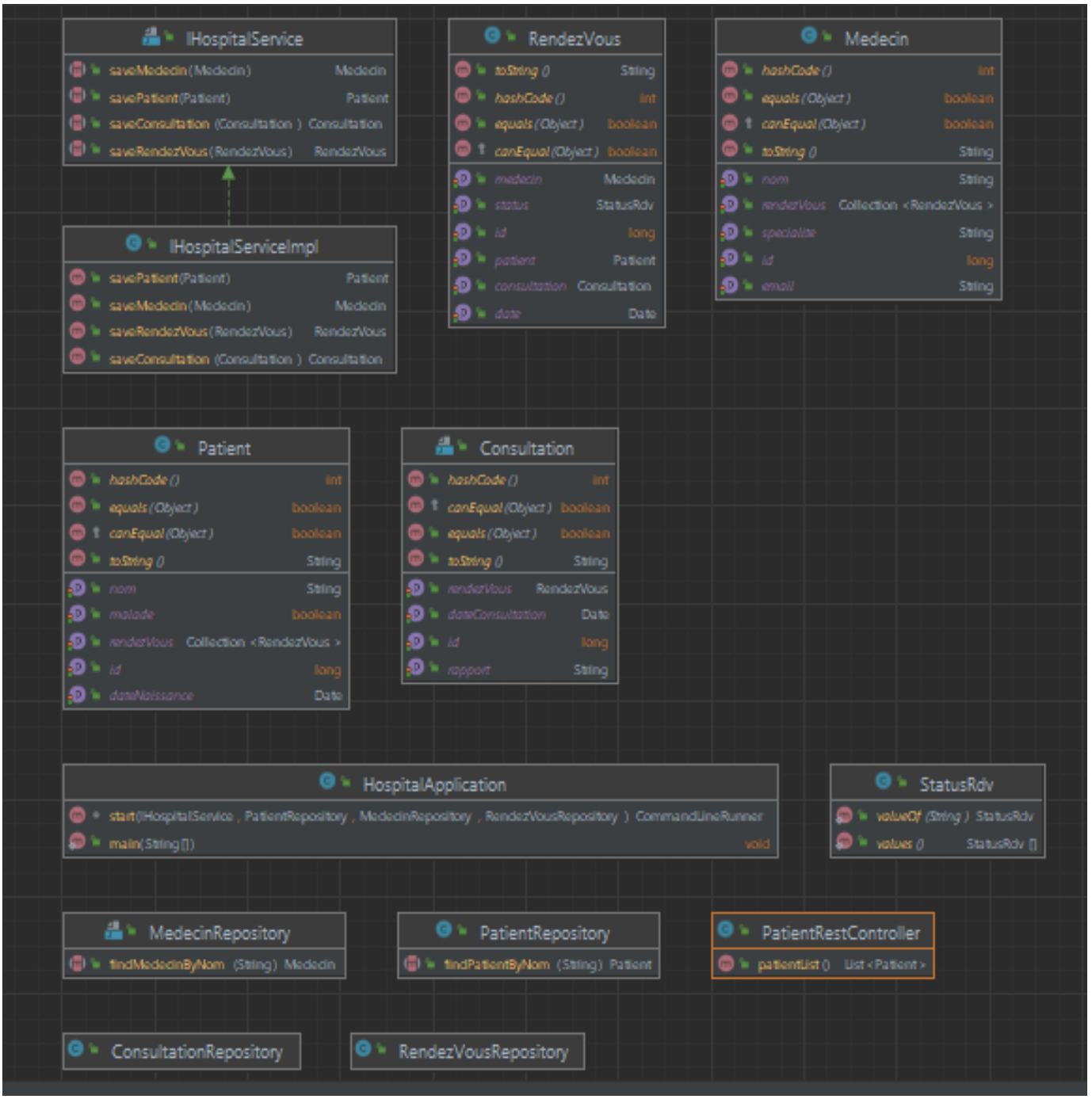
    d. RenderVous
2. Etablir les OneToOne , OneToMany , ManyToOne , ManyToMany relations entre les entités
3. Créer les interfaces :
    a. PatientRepository
    b. MedecinRepository
    c. ConsultationRepository
    d. RenderVousRepository
4. Insérer des patients , medecins , un rendezvous et une consultation dans la base de données à l'aide des interfaces et les afficher .
5. Créer l'interface IHospitalService et son implémentation : la classe IHospitalServiceImpl
6. Créer la classe patientRestController dans le package web et afficher les données des patients sur http://localhost:8086/patients

3. Conception et architecture :

4. Partie 1 :


Patient
dateNaissance Date
nom String
malade boolean
score int
id Long
Patient (Long, String, Date, boolean, int)
Patient ()
toString () String
hashCode () int
canEqual (Object) boolean
equals (Object) boolean
nom String
malade boolean
id Long
score int
dateNaissance Date
PatientRepository
findByMaladeAndAndScoreLessThan(boolean, int) List<Patient>
findByMaladeIsTrueAndScoreLessThan(int) List<Patient>
findByDateNaissanceBetween(Date, Date) List<Patient>
findByMalade(boolean, Pageable) Page<Patient>
chercherPatient(Date, Date, String) List<Patient>
JpaApApplication
JpaApApplication()
run(String[]) void
main(String[]) void

a. partie 2 :



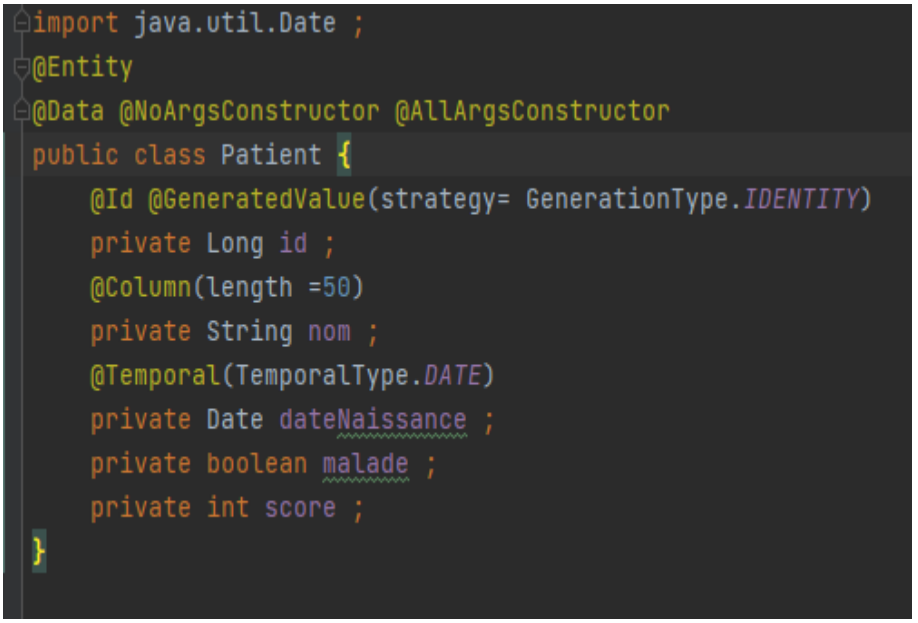
5. Code Source :

- **Github Repository :**
  - **Partie 1 :** https://github.com/felgarti/Atelier2-jpa-SpringBoot-Part1.git
  - **Partie 2 :** https://github.com/felgarti/atelier2-jpa-springboot-part2.git

6. Captures d’écran :

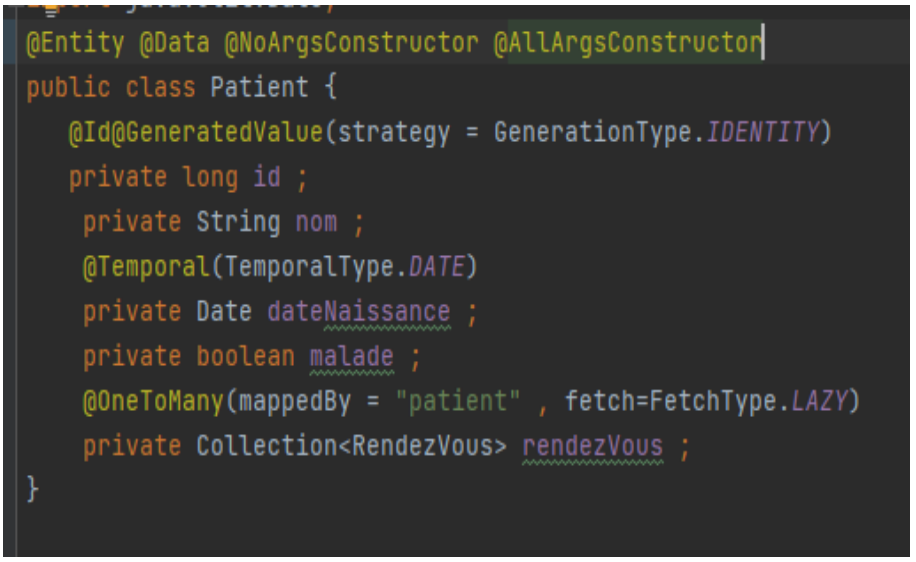
a. Patient:

i. Partie 1 :



```
import java.util.Date ;
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Patient {
    @Id @GeneratedValue(strategy= GenerationType.IDENTITY)
    private Long id ;
    @Column(length =50)
    private String nom ;
    @Temporal(TemporalType.DATE)
    private Date dateNaissance ;
    private boolean malade ;
    private int score ;
}
```

ii. Partie 2 :



```
@Entity @Data @NoArgsConstructor @AllArgsConstructor
public class Patient {
   @Id@GeneratedValue(strategy = GenerationType.IDENTITY)
   private long id ;
    private String nom ;
    @Temporal(TemporalType.DATE)
    private Date dateNaissance ;
    private boolean malade ;
    @OneToMany(mappedBy = "patient" , fetch=FetchType.LAZY)
    private Collection<RendezVous> rendezVous ;
}
```

b. Medecin :

```java
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor

public class Medecin {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id ;
    private String nom ;
    private  String specialite ;
    private  String email ;
    @OneToMany(mappedBy = "medecin" , fetch=FetchType.LAZY)
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private Collection<RendezVous> rendezVous ;

}
```

c. RendezVous :

```java
@Entity
@Data@AllArgsConstructor@NoArgsConstructor
public class RendezVous {
    @Id@GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private Date date;
    @Enumerated(EnumType.STRING)
    private StatusRdv status;
@ManyToOne
@JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private Patient patient ;
@ManyToOne
    private  Medecin medecin ;
@OneToOne(mappedBy = "rendezVous" , fetch= FetchType.LAZY)
private Consultation consultation ;
}
```

d. Consultation :

```java
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Consultation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private  long id ;
    private Date dateConsultation ;
    private  String rapport  ;
    @OneToOne
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private RendezVous rendezVous ;
}
```

e. PatientRepository :

i. Partie 1 :

```
public interface PatientRepository extends JpaRepository<Patient,Long> {
   // public List<Patient> findByMalade(boolean m) ;
public Page<Patient> findByMalade(boolean m , Pageable pageable) ;
List<Patient> findByMaladeAndAndScoreLessThan(boolean m , int s) ;
   List<Patient> findByMaladeIsTrueAndScoreLessThan( int s) ;
   List<Patient> findByDateNaissanceBetween(Date d1 , Date d2) ;
@Query("select p from Patient  p where p.dateNaissance between  :x and :y or p.nom like :z")
   List<Patient> chercherPatient(@Param("x") Date d1 , @Param("y") Date d2 ,@Param("z") String nom ) ;

}
```

ii. Partie 2 :

```
public interface PatientRepository extends JpaRepository<Patient,Long> {
  Patient findPatientByNom(String nom );
}
```

f. MedecinRepository :

```
public interface MedecinRepository extends JpaRepository<Medecin,Long> {
  Medecin findMedecinByNom(String nom) ;
}
```

g. RendezVousRepository :

```
public interface RendezVousRepository extends JpaRepository<RendezVous,Long> {
}
```

h. ConsultationRepository :

```
public interface ConsultationRepository extends JpaRepository<Consultation,Long> {
}
```

i. Application Partie 1 :

- **Pagination : code**

```
Page<Patient> patients = patientRepository.findAll(PageRequest.of( page: 0, size: 5)) ;
System.out.println("Total pages :  " + patients.getTotalPages());
System.out.println("Total elements  :  "+patients.getTotalElements());
System.out.println("Num current page :  "+patients.getNumber());
```

- **Pagination : affichage**

```
Total pages :  16
Total elements  :  80
Num current page :  0
```

- **Page des patients : code**

```java
Page<Patient> patients = patientRepository.findAll(PageRequest.of( page: 0, size: 5)) ;
patients.forEach(p->{
    System.out.println(p.getId());
    System.out.println(p.getNom());
    if(p.isMalade())
        System.out.println("malade");
    else  System.out.println("not malade");
    System.out.println(p.getDateNaissance());
    System.out.println(p.getScore());
    System.out.println("----------------------------------");
```

- **Page des patients : affichage**

```
1
fatima
not malade
2022-03-28
63
----------------------------------
2
hamza
malade
2022-03-28
64
----------------------------------
3
manal
not malade
2022-03-28
89
```

- **Find By malade : code**

```
System.out.println("--------------BY MALADE --------------------");
Page<Patient> byMalade= patientRepository.findByMalade( m: true,  PageRequest.of( page: 0, size: 4));
byMalade.forEach(p->{
    System.out.println(p.getId());
    System.out.println(p.getNom());
    System.out.println(p.getDateNaissance());
    System.out.println(p.getScore());
    if(p.isMalade())
        System.out.println("malade");
    else  System.out.println("not malade");
    System.out.println("----------------------------------");
```

- **Find By malade :affichage**

```
--------------BY MALADE --------------------
2
hamza
2022-03-28
64
malade
----------------------------------
4
tester
2022-03-28
61
malade
----------------------------------
```

- **findByNom : code**

```
List<Patient> patientCherche = patientRepository.chercherPatient( d1: null , d2: null , nom: "hamza") ;
patientCherche.forEach(p->
        {
            System.out.println( "    score : " +p.getScore() +" nom :  "+ p.getNom() );
        });
```

- **findByNom : affichage :**

```
score : 50 nom :  hamza
score : 37 nom :  hamza
score : 12 nom :  hamza
```

j. Application Part 2 :

```
CommandLineRunner start(IHospitalService hospitalService , PatientRepository patientRepository , MedecinRepo
                        RendezVousRepository rendezVousRepository ){
    return args ->{
        Stream.of("fatima" , " hamza" , " manal").forEach(name->
        {
            Patient p = new Patient() ;
            p.setNom(name);
            p.setDateNaissance(new Date());
            p.setMalade(false);
            hospitalService.savePatient(p) ;

        });
```

```
Stream.of("khadija" , " yousra " , " anas ").forEach(name->
{
    Medecin p = new Medecin() ;
    p.setNom(name);
   p.setEmail(name+"@gmail.com");
   p.setSpecialite(Math.random()>0.5?"cardio":"generaliste");
    hospitalService.saveMedecin(p) ;

});
```

```
Patient patient = patientRepository.findPatientByNom("fatima") ;
Medecin medecin = medecinRepository.findMedecinByNom("yousra") ;
RendezVous rdv = new RendezVous() ;
rdv.setMedecin(medecin);
rdv.setPatient(patient);
rdv.setDate(new Date());
rdv.setStatus(StatusRdv.PENDING);
hospitalService.saveRendezVous(rdv) ;
Consultation clt = new Consultation() ;
RendezVous rdv1 = rendezVousRepository.findById(1L).orElse( other: null) ;
clt.setRendezVous(rdv1);
clt.setDateConsultation(new Date());

clt.setRapport("rapport du rdv ");
hospitalService.saveConsultation(clt) ;
```

- **h2-console :**

SELECT * FROM CONSULTATION;

ID	DATE_CONSULTATION	RAPPORT	RENDEZ_VOUS_ID
1	2022-03-28 21:19:06.629	rapport du rdv	1

(1 row, 33 ms)

SELECT * FROM MEDECIN;

ID	EMAIL	NOM	SPECIALITE
1	khadija@gmail.com	khadija	generaliste
2	yousra @gmail.com	yousra	cardio
3	anas @gmail.com	anas	generaliste

(3 rows, 1 ms)

SELECT * FROM RENDEZ_VOUS;

ID	DATE	STATUS	MEDECIN_ID	PATIENT_ID
1	2022-03-28 21:19:06.611	PENDING	*null*	1

(1 row, 3 ms)

SELECT * FROM PATIENT;

ID	DATE_NAISSANCE	MALADE	NOM
1	2022-03-28	FALSE	fatima
2	2022-03-28	FALSE	hamza
3	2022-03-28	FALSE	manal

(3 rows, 3 ms)

k. Hospital service :

```
public interface IHospitalService {
    Patient savePatient(Patient p);
    Medecin saveMedecin(Medecin m) ;
    RendezVous saveRendezVous(RendezVous rdv) ;
    Consultation saveConsultation(Consultation c) ;

}
```

```java
@Service
@Transactional
public class IHospitalServiceImpl implements IHospitalService {

    private PatientRepository patientRepository ;
    private MedecinRepository medecinRepository ;
    private RendezVousRepository rendezVousRepository ;
    private ConsultationRepository consultationRepository ;

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

    }
```

```java
@Override
public Patient savePatient(Patient p) { return  patientRepository.save(p) ; }

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

@Override
public RendezVous saveRendezVous(RendezVous rdv) { return rendezVousRepository.save(rdv); }

@Override
public Consultation saveConsultation(Consultation c) { return consultationRepository.save(c); }
```

I. PatientRestController :

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

http://localhost:8086/patients :


```
[{"id":1,"nom":"fatima","dateNaissance":"2022-03-28","malade"
{"id":1,"dateConsultation":"2022-03-28T19:19:06.629+00:00","r
{"id":3,"nom":" manal","dateNaissance":"2022-03-28","malade":
```

7. Conclusion :

Les outils Spring utilisés permettent de gérer les tables de base de données grâces aux entités et interfaces plus facilement que sans l'aide d'un framework. Dans cet atelier , nous avons testé l'ajout , la suppression , l'affichage et la recherche grâce aux méthodes déclarées dans les interfaces repositories.