Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

ENSET

ا لمدرسة العليا لأساتذة التعليم التقني المحمدية

جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Compte rendu du

JPA Hibernate Spring Data

Filière :

« Génie du Logiciel et des Systèmes Informatiques Distribués »

GLSID

Gestion des patients

Réalisé par :

Najwa ZRAIDI

Professeur :

M. Mohamed YOUSSFI

Année Universitaire : 2022-2023

Introduction

La gestion des patients fait référence à l'ensemble des processus impliqués dans la prise en charge des patients dans un établissement de santé. Cela peut inclure la gestion de l'admission des patients, la collecte de leurs informations de base et de leur historique médical, la planification et la coordination des traitements, la gestion des dossiers médicaux, la gestion des rendez-vous et des horaires, ainsi que la coordination des soins de santé entre différents professionnels de la santé.

1. Créer l'entité JPA Patient ayant les attributs :
   - id de type Long
   - nom de type String
   - dateNaissanec de type Date
   - malade de type boolean
   - score de type int

```
package ma.enset.entities;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

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;

}
```

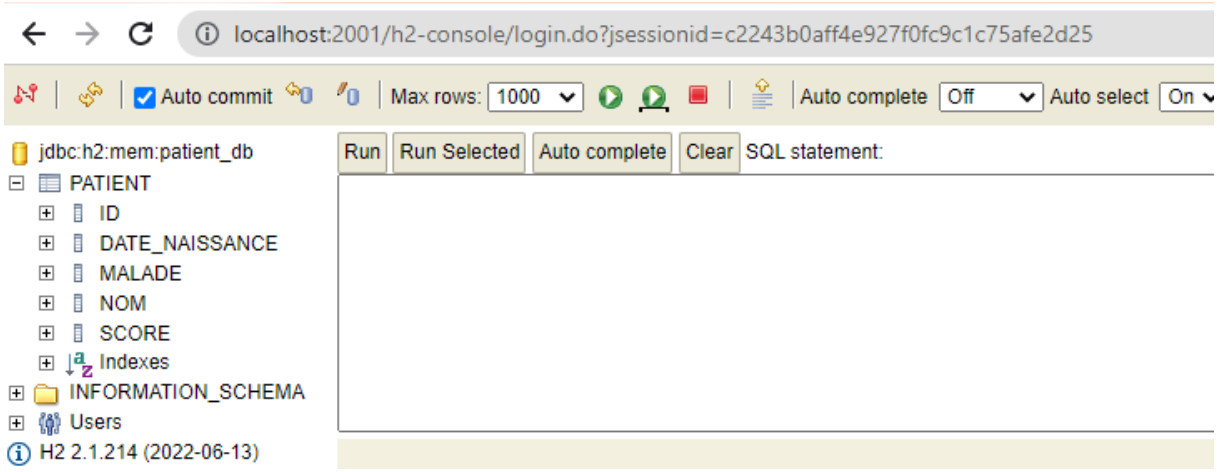
2. Configurer l'unité de persistance dans le ficher application.properties

⇨ Ajout de dépendance :

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
```

```
spring.datasource.url=jdbc:h2:mem:patient_db
spring.h2.console.enabled=true
server.port=2001
```

⇨ La base de données : patient_db et pour accéder à cette bdd taper Localhost :N° de port (2001) /h2_console



3. Créer l'interface JPA Repository basée sur Spring data

```
package ma.enset.repositories;

import ma.enset.entities.Patient;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.util.Date;
import java.util.List;

//Mapping objet relationel
public interface PatientRepository extends JpaRepository<Patient,Long> {
public List<Patient> findByMalade(boolean m);
Page<Patient> findByMalade(boolean m, Pageable pageable);
List<Patient> findByMaladeAndScoreLessThan(boolean m,int score);
List<Patient> findByMaladeIsTrueAndScoreLessThan(int score);
List<Patient> findByDateNaissanceBetween(Date d1, Date d2);
List<Patient> findByDateNaissanceBetweenAndMaladeIsTrue(Date d1, Date d2);
List<Patient> findByDateNaissanceBetweenAndMaladeIsTrueOrNomLike(Date d1,
Date d2,String mc);
//2eme methode pour l'affichage
@Query("select p from Patient p where p.dateNaissance between :x and :y or
p.nom like :z")
List<Patient> chercherPatients(@Param("x") Date d1,@Param("y") Date
d2,@Param("x") String mc);
@Query("select p from Patient p where p.nom like :N and p.score<:S ")
List<Patient> chercherPatients(@Param("N") String nom,@Param("S") int
scoreMin);
}
```

⇨ Cette interface contient les méthodes basées sur spring et d'autre avec les requêtes

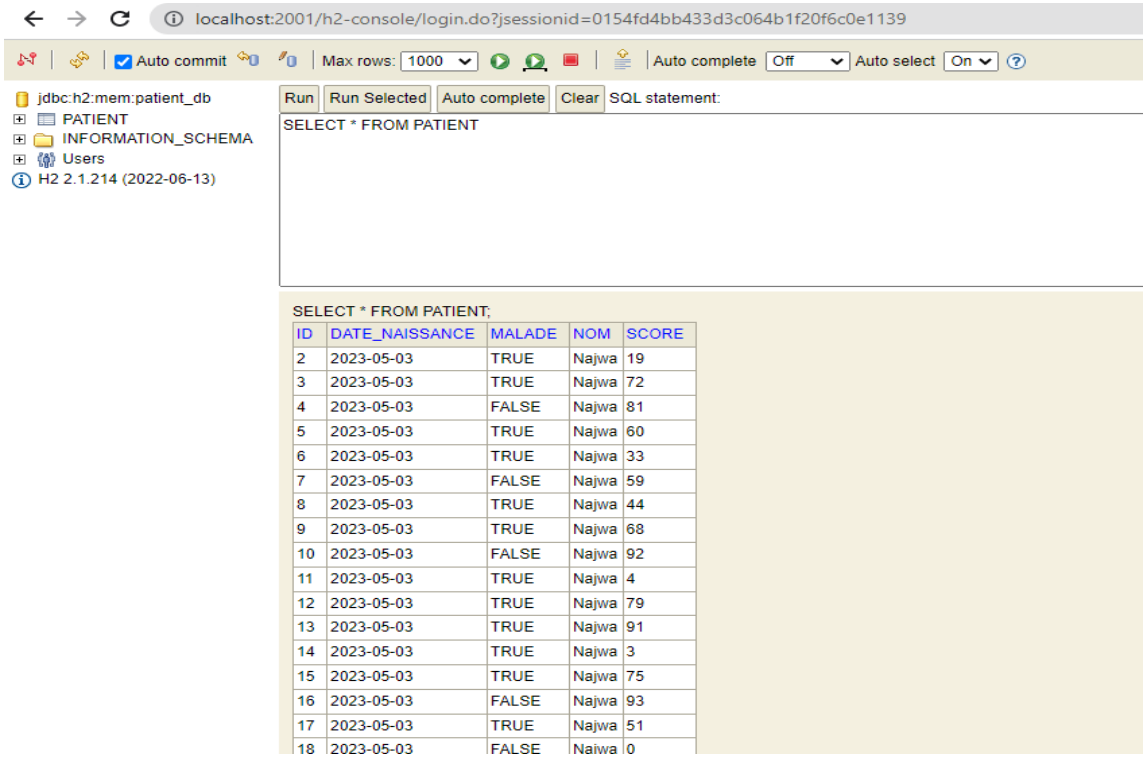
4. Tester quelques opérations de gestion de patients :

- Ajouter des patients

- Consulter tous les patients
- Consulter un patient
- Chercher des patients
- Mettre à jour un patient
- supprimer un patient

```
5. package ma.enset;

   import ma.enset.entities.Patient;
   import ma.enset.repositories.PatientRepository;
   import org.springframework.beans.factory.annotation.Autowired;
   import org.springframework.boot.CommandLineRunner;
   import org.springframework.boot.SpringApplication;
   import org.springframework.boot.autoconfigure.SpringBootApplication;
   import org.springframework.data.domain.Page;
   import org.springframework.data.domain.PageRequest;

   import java.util.Date;
   import java.util.List;
   import java.util.Optional;

   @SpringBootApplication
   public class Devoir3Application implements CommandLineRunner {
       //injection des dependances
       @Autowired
       private PatientRepository patientRepository;
       public static void main(String[] args) {
           SpringApplication.run(Devoir3Application.class, args);
       }

       @Override
       public void run(String... args) throws Exception {
           for (int i=0;i<100;i++){
               patientRepository.save(new Patient(null,"Najwa",new
   Date(),Math.random()>0.5?true:false,(int)(Math.random()*100)));
           }
          /* patientRepository.save(new Patient(null,"Awjan",new
   Date(),false,10));
           patientRepository.save(new Patient(null,"Najwa",new
   Date(),true,45));
           patientRepository.save(new Patient(null,"Hania",new
   Date(),false,78));
          */ //liste des patients
           //pagination pagerequest
           Page<Patient>
   patients=patientRepository.findAll(PageRequest.of(0,5));
           System.out.println("Total des pages :
   "+patients.getTotalPages());
           System.out.println("Total des elemets :
   "+patients.getTotalElements());
           System.out.println("N° de page  : "+patients.getNumber());
           List<Patient> patientList=patients.getContent();
           //List<Patient>
   byMalade=patientRepository.findByMalade(true);
           Page<Patient>
   byMalade=patientRepository.findByMalade(true,PageRequest.of(0,4));
           List<Patient>
   patientList1=patientRepository.chercherPatients("%N%",780);
           System.out.println("Liste des patients");
```

```
            byMalade.forEach(p->{
                System.out.println(p.getId());
                System.out.println(p.getNom());
                System.out.println(p.isMalade());
            });
            System.out.println("*************************");
            Patient
    patient=patientRepository.findById(1L).orElseThrow(()->new
    RuntimeException("Patient n'existe pas"));
            // or Patient
    patient=patientRepository.findById(1L).orElse(null);
            if(patient!=null){
                System.out.println(patient.getId());
                System.out.println(patient.getNom());
                System.out.println(patient.isMalade());
            }
            //update
            patient.setScore(870);
            patientRepository.save(patient);
            //suppression
            patientRepository.deleteById(1L);
            System.out.println("Liste des patients avec filtrage :");
            byMalade.forEach(p->{
                System.out.println(p.getId());
                System.out.println(p.getNom());
                System.out.println(p.isMalade());
                System.out.println(p.getScore());
            });
        }
    }
```



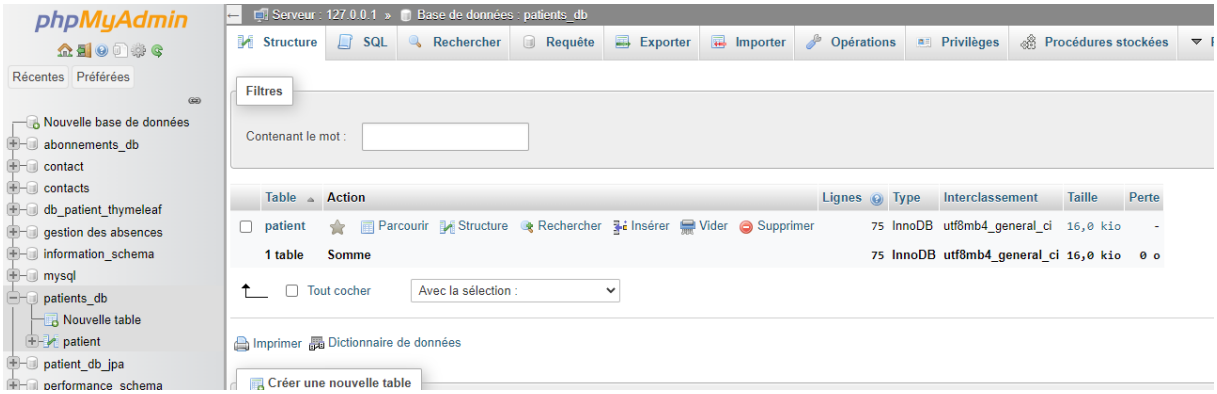
ID	DATE_NAISSANCE	MALADE	NOM	SCORE
2	2023-05-03	TRUE	Najwa	19
3	2023-05-03	TRUE	Najwa	72
4	2023-05-03	FALSE	Najwa	81
5	2023-05-03	TRUE	Najwa	60
6	2023-05-03	TRUE	Najwa	33
7	2023-05-03	FALSE	Najwa	59
8	2023-05-03	TRUE	Najwa	44
9	2023-05-03	TRUE	Najwa	68
10	2023-05-03	FALSE	Najwa	92
11	2023-05-03	TRUE	Najwa	4
12	2023-05-03	TRUE	Najwa	79
13	2023-05-03	TRUE	Najwa	91
14	2023-05-03	TRUE	Najwa	3
15	2023-05-03	TRUE	Najwa	75
16	2023-05-03	FALSE	Najwa	93
17	2023-05-03	TRUE	Najwa	51
18	2023-05-03	FALSE	Najwa	0

6. Migrer de H2 Database vers MySQL

⇨ Ajout de dépendance :

```xml
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.32</version>
</dependency>
```

```
# commentaire => spring.datasource.url=jdbc:h2:mem:patient_db
#spring.h2.console.enabled=true
server.port=2001
spring.datasource.url=jdbc:mysql://localhost:3306/Patient_DB_JPA?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
#show les requetes sql executes
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
```



8. Reprendre les exemples du Patient, Médecin, rendez-vous, consultation

- Patient :

```
package ma.enset.entities;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.Collection;
import java.util.Date;
@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> rendezVousCollation;
}
```

- Médecin

```
package ma.enset.entities;

import com.fasterxml.jackson.annotation.JsonProperty;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.Collection;
@Entity
@Data
@NoArgsConstructor
@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;
}
```

- MedecinRepository

```
package ma.enset.repositories;

import ma.enset.entities.Medecin;
import ma.enset.entities.Patient;
import org.springframework.data.jpa.repository.JpaRepository;

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


}
```

- MedecinController

```
-   package ma.enset.web;

    import ma.enset.entities.Medecin;
    import ma.enset.repositories.MedecinRepository;
    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 MedecinRestController {
    @Autowired
        private MedecinRepository medecinRepository;
    @GetMapping("/medecins")
    public List<Medecin> medecins() {
        return medecinRepository.findAll();

    }
}
```

- Consultation

```
package ma.enset.entities;

import com.fasterxml.jackson.annotation.JsonProperty;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.Date;
@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;

}
```

- ConsultationRepository

```
package ma.enset.repositories;

import ma.enset.entities.Consultation;
import ma.enset.entities.Patient;
import org.springframework.data.jpa.repository.JpaRepository;

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

- ConsultationController

```java
package ma.enset.web;

import ma.enset.entities.Consultation;
import ma.enset.repositories.ConsultationRepository;
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 ConsultationRestController {
    @Autowired
        private ConsultationRepository consultationRepository;
    @GetMapping("/Consultations")
    public List<Consultation> Consultations() {
        return consultationRepository.findAll();

    }



}
```

- RendezVous

```java
package ma.enset.entities;

import com.fasterxml.jackson.annotation.JsonProperty;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

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
    //ajout mais ne pas consulter dans la lecteur
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private Patient patient;
    @ManyToOne
    private  Medecin medecin;
    @OneToOne(mappedBy = "rendezVous")
    private Consultation consultation;
}
```

- Class StatusRDV

```java
package ma.enset.entities;

public enum StatusRDV {
```

```
        EN_COURS,
        ANNULER,
        VALIDER
    }
```

- RendezVousRepository

```
-   package ma.enset.repositories;

    import ma.enset.entities.Patient;
    import ma.enset.entities.RendezVous;
    import org.springframework.data.jpa.repository.JpaRepository;

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

- RendezVousController

```
-   package ma.enset.web;

    import ma.enset.entities.RendezVous;
    import ma.enset.repositories.RendezVousRepository;
    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 RendezVousRestController {
        @Autowired
            private RendezVousRepository rendezVousRepository;
        @GetMapping("/rendez-Vous")
        public List<RendezVous> RendezVous() {
            return rendezVousRepository.findAll();

        }

    }
```

- Interface HospitalService dans les services qui englobe les méthodes d'ajout, modification et suppression

```
package ma.enset.service;

import ma.enset.entities.Consultation;
import ma.enset.entities.Medecin;
import ma.enset.entities.Patient;
import ma.enset.entities.RendezVous;

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

```
    Consultation saveConsultation(Consultation consultation);
}
```

- Class HospitalServiceImpl implementation interface IHospitalService

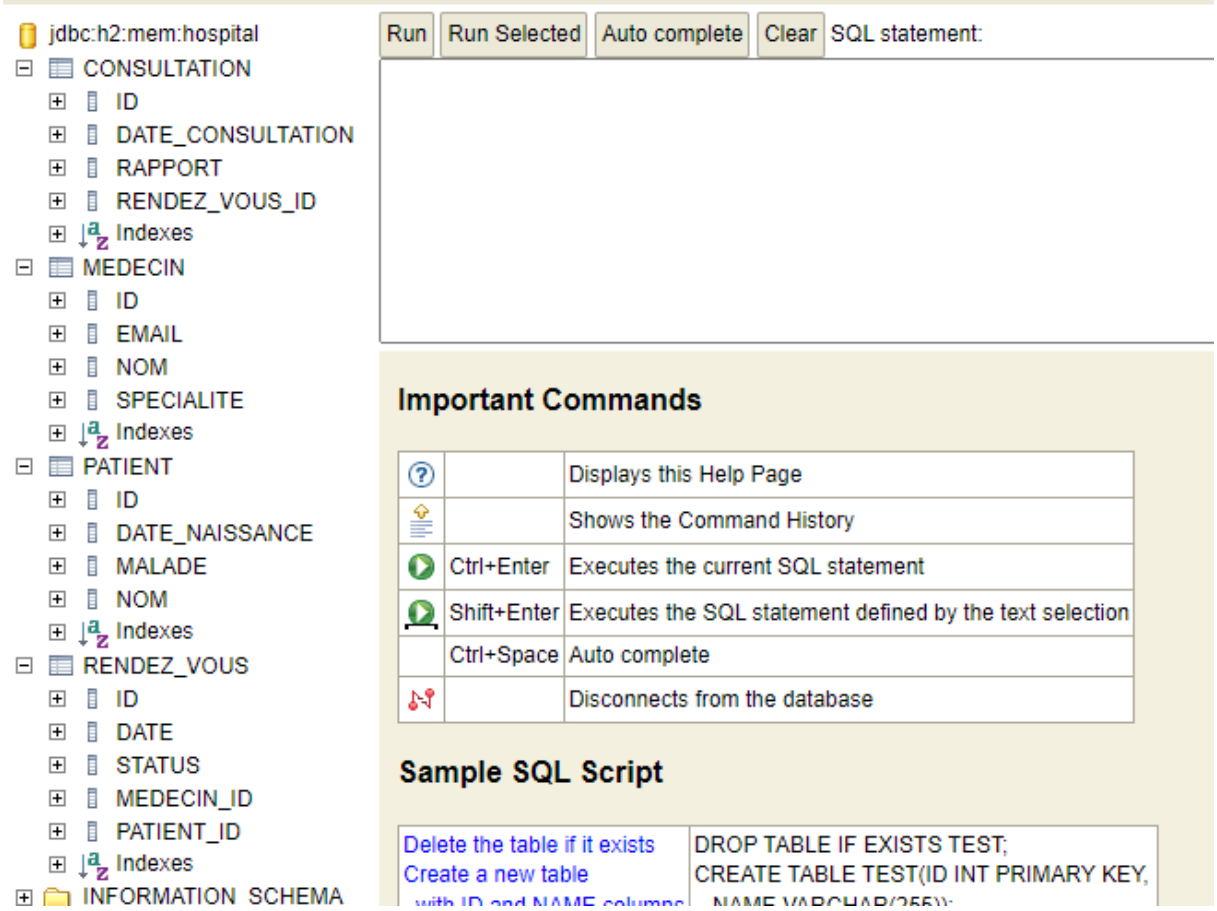
```
package ma.enset.service;

import jakarta.transaction.Transactional;
import ma.enset.entities.Consultation;
import ma.enset.entities.Medecin;
import ma.enset.entities.Patient;
import ma.enset.entities.RendezVous;
import ma.enset.repositories.ConsultationRepository;
import ma.enset.repositories.MedecinRepository;
import ma.enset.repositories.PatientRepository;
import ma.enset.repositories.RendezVousRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.UUID;

@Service
@Transactional
public class HospitalServiceImpl implements IHospitalService {
    //Injection des depandances
    //@Autowired
    private PatientRepository patientRepository;
    //@Autowired
    private MedecinRepository medecinRepository;
    //@Autowired
    private RendezVousRepository rendezVousRepository;
    //@Autowired
    private ConsultationRepository consultationRepository;
    //Injection des depandances by constructor
    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) {
        //Generation de id string
        rendezVous.setId(UUID.randomUUID().toString());
        return rendezVousRepository.save(rendezVous);
    }
```

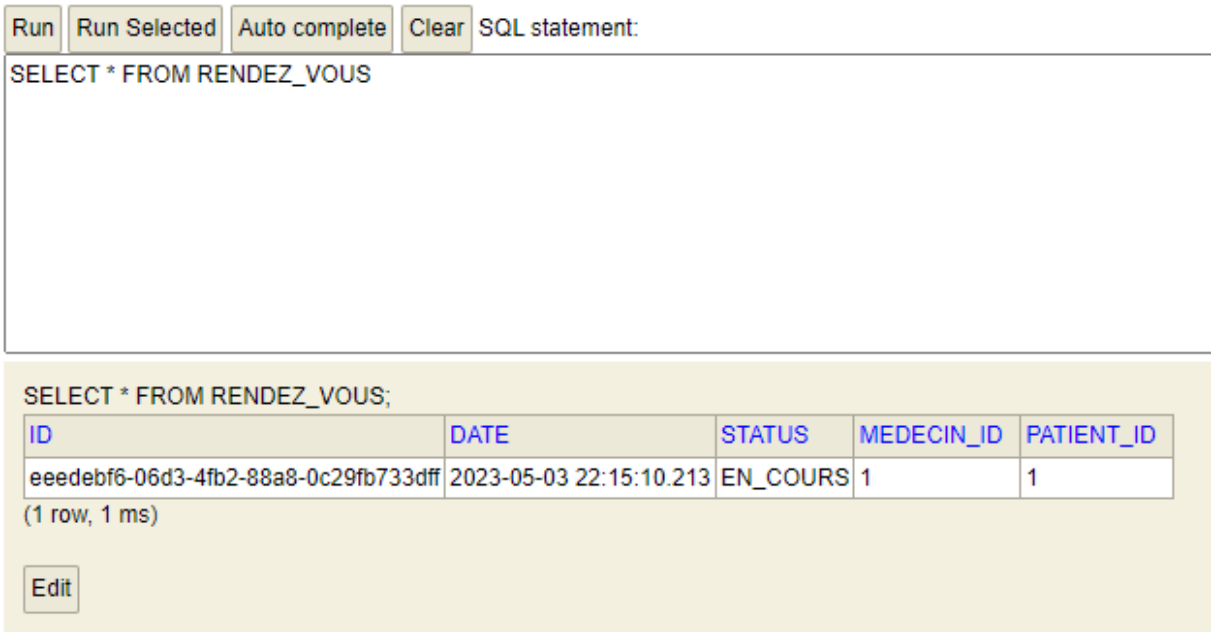
```
        @Override
        public Consultation saveConsultation(Consultation consultation) {
            return consultationRepository.save(consultation);
        }
    }
```

- Test d'application :

```
package ma.enset;

import ma.enset.entities.*;
import ma.enset.repositories.ConsultationRepository;
import ma.enset.repositories.MedecinRepository;
import ma.enset.repositories.PatientRepository;
import ma.enset.repositories.RendezVousRepository;
import ma.enset.service.IHospitalService;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

import java.util.Date;
import java.util.stream.Stream;

@SpringBootApplication
public class HospitalApplication {

    public static void main(String[] args) {
        SpringApplication.run(HospitalApplication.class, args);
    }
    @Bean
    //test
    CommandLineRunner start(IHospitalService
hospitalService,PatientRepository patientRepository, MedecinRepository
medecinRepository, RendezVousRepository rendezVousRepository){
    //CommandLineRunner start(PatientRepository patientRepository,
MedecinRepository medecinRepository, RendezVousRepository
rendezVousRepository, ConsultationRepository consultationRepository){
        return args -> {
            Stream.of("Khalid","Islam","Awjan")
                    .forEach(
                            name->{
                                Patient patient=new Patient();
                                patient.setNom(name);
                                patient.setDateNaissance(new Date());
                                patient.setMalade(false);
                                //patientRepository.save(patient);
                                hospitalService.savePatient(patient);
                            }
                    );
            Stream.of("Aymane","Yassmin","Yasin")
                    .forEach(
                            name->{
                                Medecin medecin=new Medecin();
                                medecin.setNom(name);
                                medecin.setEmail(name+"@gmail.com");

medecin.setSpecialite(Math.random()>0.5?"Cardio":"Dentiste");
```

```
                                //medecinRepository.save(medecin);
                                hospitalService.saveMedecin(medecin);
                            }
                    );
            Patient patient=patientRepository.findById(1L).orElse(null);
            Patient patient1=patientRepository.findByNom("Awjan");

            Medecin medecin=medecinRepository.findByNom("Aymane");

            RendezVous rendezVous=new RendezVous();
            rendezVous.setDate(new Date());
            rendezVous.setStatus(StatusRDV.EN_COURS);
            rendezVous.setMedecin(medecin);
            rendezVous.setPatient(patient);
            //rendezVousRepository.save(rendezVous);
            hospitalService.saveRDV(rendezVous);


           // RendezVous
rendezVous1=rendezVousRepository.findById(1L).orElse(null);
            RendezVous rendezVous1=rendezVousRepository.findAll().get(0);
            Consultation consultation=new Consultation();
            consultation.setDateConsultation(new Date());
            consultation.setRendezVous(rendezVous1);
            consultation.setRapport("Rapport de a consultation .....");
            //consultationRepository.save(consultation);
            hospitalService.saveConsultation(consultation);
        };
    }
}
```


jdbc:h2:mem:hospital
CONSULTATION
ID
DATE_CONSULTATION
RAPPORT
RENDEZ_VOUS_ID
Indexes
MEDECIN
ID
EMAIL
NOM
SPECIALITE
Indexes
PATIENT
ID
DATE_NAISSANCE
MALADE
NOM
Indexes
RENDEZ_VOUS
ID
DATE
STATUS
MEDECIN_ID
PATIENT_ID
Indexes
INFORMATION_SCHEMA
Run
Run Selected
Auto complete
Clear
SQL statement:
Important Commands
Displays this Help Page
Shows the Command History
Ctrl+Enter
Executes the current SQL statement
Shift+Enter
Executes the SQL statement defined by the text selection
Ctrl+Space
Auto complete
Disconnects from the database
Sample SQL Script
Delete the table if it exists
Create a new table
DROP TABLE IF EXISTS TEST;
CREATE TABLE TEST(ID INT PRIMARY KEY,


Run
Run Selected
Auto complete
Clear
SQL statement:
SELECT * FROM RENDEZ_VOUS
SELECT * FROM RENDEZ_VOUS;
ID
DATE
STATUS
MEDECIN_ID
PATIENT_ID
eeedebf6-06d3-4fb2-88a8-0c29fb733dff
2023-05-03 22:15:10.213
EN_COURS
1
1
(1 row, 1 ms)
Edit

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

```
SELECT * FROM PATIENT
```

SELECT * FROM PATIENT;

ID	DATE_NAISSANCE	MALADE	NOM
1	2023-05-03	FALSE	Khalid
2	2023-05-03	FALSE	Islam
3	2023-05-03	FALSE	Awjan

(3 rows, 0 ms)

Edit

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

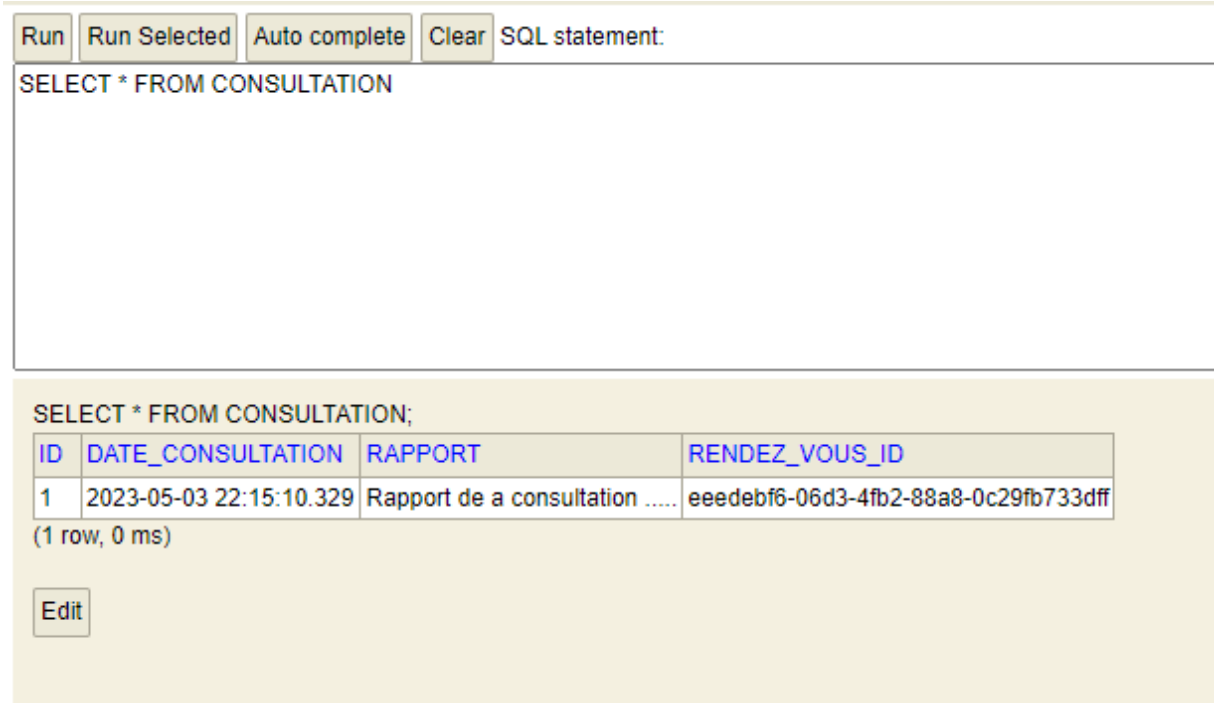
```
SELECT * FROM MEDECIN
```

SELECT * FROM MEDECIN;

ID	EMAIL	NOM	SPECIALITE
1	Aymane@gmail.com	Aymane	Dentiste
2	Yassmin@gmail.com	Yassmin	Cardio
3	Yasin@gmail.com	Yasin	Cardio

(3 rows, 0 ms)

Edit



Conclusion :

En conclusion, JPA Hibernate Spring Data est une technologie puissante et fiable pour la gestion des patients qui peut aider les établissements de santé à améliorer leur efficacité opérationnelle tout en offrant une meilleure qualité de soins aux patients.