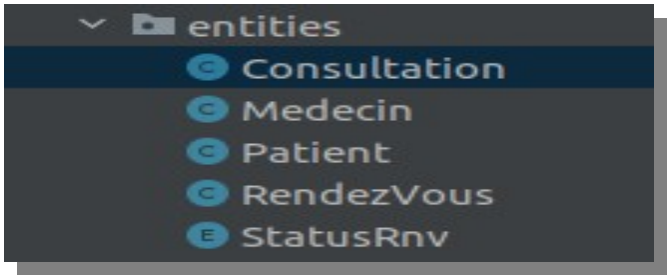
Activité pratique : La gestion des consultations

Créé par : Mohamed Ait Lahcen

1. Les entités de l’application :

Cette application contient quatre entités :



- **Consultation :**

```java
import java.util.Date;
11 usages
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Consultation {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Date dateConsultation;
    private String rapport;
    1 usage
    @OneToOne
    @JsonProperty(access =JsonProperty.Access.WRITE_ONLY)
    private RendezVous rendezVous;

}
```

- **Medecin :**

```
import ...
13 usages
@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> rendezVousCollection;
}
```

- **Patient :**

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

}
```

- **Rendez-vous :**

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

- **Le type Enumeration StatusRnv :**

```
package ma.enset.hospital.entities;

public enum StatusRnv {
    1 usage
    pending,
    canceled,
    done
}
```

2. Les interfaces repository :

Cette application contient une interface repository pour chaque entité :

- **ConsultationRepository :**

```
package ma.enset.hospital.repositories;

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

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

- **MedecinRepository :**

```
5 usages
public interface MedecinRepository extends JpaRepository<Medecin,Long> {
    1 usage
    Medecin findByNom(String name);
}
```

- **PatientRepository :**

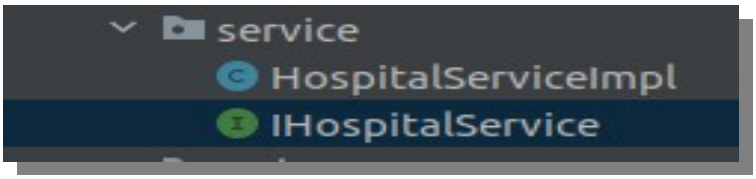
```
7 usages
public interface PatientRepository extends JpaRepository<Patient,Long> {
1 usage
Patient findByNom(String name);
}
```

- **RendezVousRepository :**

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

3. La couche service (Metier) :

Ce package contient une interface et son implémentation en fait il représente la couche metier de l'application :



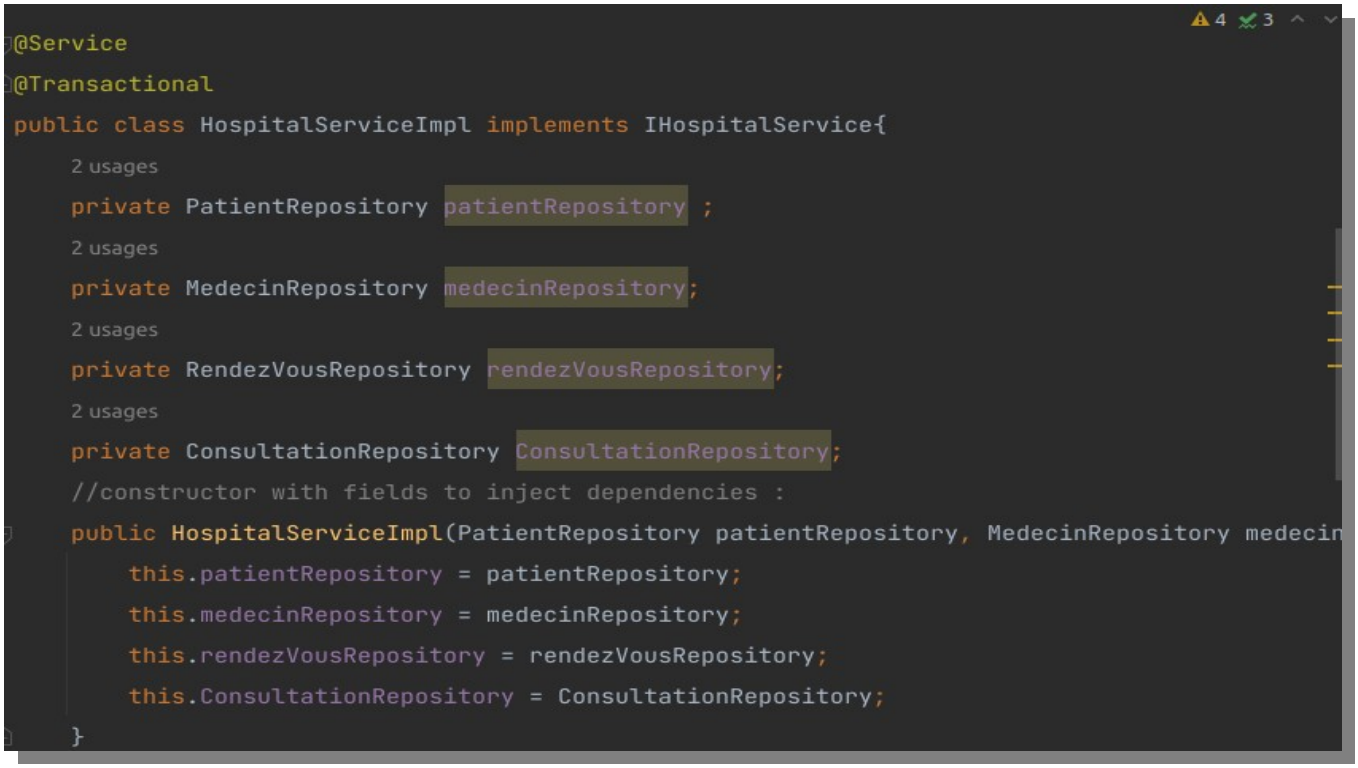
- **IHospitalService :**

cette interface qui contient les fonctions qui permettent d'ajouter les instances des entités ou bien les enregistrements des entités dans la base de données :

```
3 usages  1 implementation
public interface IHospitalService {
    1 usage  1 implementation
    public Patient savePatient(Patient patient);
    1 usage  1 implementation
    public Medecin saveMedecin(Medecin medecin);
    1 usage  1 implementation
    public RendezVous saveRendezVous(RendezVous rendezVous);
    1 usage  1 implementation
    public Consultation saveConsultation(Consultation consultation);
}
```

- **HospitalServiceImpl :**

  cette classe contient la déclaration des fonctions de l'interface IhospitalService :



```
@Service
@Transactional
public class HospitalServiceImpl implements IHospitalService{
    2 usages
    private PatientRepository patientRepository ;
    2 usages
    private MedecinRepository medecinRepository;
    2 usages
    private RendezVousRepository rendezVousRepository;
    2 usages
    private ConsultationRepository ConsultationRepository;
    //constructor with fields to inject dependencies :
    public HospitalServiceImpl(PatientRepository patientRepository, MedecinRepository medecin
        this.patientRepository = patientRepository;
        this.medecinRepository = medecinRepository;
        this.rendezVousRepository = rendezVousRepository;
        this.ConsultationRepository = ConsultationRepository;
    }
```

```
    1 usage
    @Override
    public Patient savePatient(Patient patient) { return patientRepository.save(patient); }
    1 usage
    @Override
    public Medecin saveMedecin(Medecin medecin) { return medecinRepository.save(medecin); }
    1 usage
    @Override
    public RendezVous saveRendezVous(RendezVous rendezVous) {
        rendezVous.setId(UUID.randomUUID().toString());
        return rendezVousRepository.save(rendezVous);
    }
    1 usage
    @Override
    public Consultation saveConsultation(Consultation consultation) {
        return ConsultationRepository.save(consultation);
    }
}
```

4. La connexion avec la base de données :

Le type de la base de données utilisé dans cette application est h2 voici le fichier configuration.properties qui contient le type de la base de données utilisé :

```
spring.datasource.url=jdbc:h2:mem:hospital
#spring.datasource.url=jdbc:mysql://localhost:3306/hospitalDB?createDatabaseIfNotExist=true
#spring.datasource.username=root
#spring.datasource.password=
#le parametre upadte signifiie que a chaque fois on ajoute des donnï¿½es au tableau on n'ecrase
#spring.jpa.hibernate.ddl-auto=update
#spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MariaDBDialect
spring.h2.console.enabled=true
server.port=8082
#spring.jpa.show-sql=true
```

- **HostpitalApplication :**

  Cette classe permet d’interagir avec les table de la base de données voici une partie de code de cette classe dans laquelle j’ai ajouté des enregistrements dans la base de données :

```java
@Bean
CommandLineRunner start(IHospitalService HospitalService,PatientRepository patientRepository,MedecinRepository medecinReposit
                        RendezVousRepository rendezVousRepository,ConsultationRepository consultationRepository){
    return args-> {
        Stream.of( ...ts: "nom1","nom2","nom3").forEach(name->{
            Patient patient = new Patient();
            patient.setNom(name);
            patient.setDate_naissance(new Date());
            patient.setMalade((Math.random()*10 )%2 ==0? true : false);
            HospitalService.savePatient(patient);
        });
        //la création des medecins :
        Stream.of( ...ts: "medecin1","medecin2","medecin3").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);
        });
```

```java
        });
        Patient patient1=patientRepository.findById(1L).orElse( t: null);
        Patient patient = patientRepository.findByNom( name: "nom1");
        Medecin medecin = medecinRepository.findByNom( name: "medecin1");
        RendezVous rendezVous = new RendezVous();
        rendezVous.setDate(new Date());
        rendezVous.setStatus(StatusRnv.pending);
        rendezVous.setMedecin(medecin);
        rendezVous.setPatient(patient);
        HospitalService.saveRendezVous(rendezVous);
        RendezVous rendezVous1=rendezVousRepository.findAll().get(0);
        Consultation consultation =new Consultation();
        consultation.setDateConsultation(new Date());
        consultation.setRendezVous(rendezVous1);
        consultation.setRapport("rapport de la consultation.....");
        HospitalService.saveConsultation(consultation);


    };
```

◆ Les tables de la bases de données :

1. La table CONSULTATION :

```
SELECT * FROM CONSULTATION;
```

ID	DATE_CONSULTATION	RAPPORT	RENDEZ_VOUS_ID
1	2022-05-06 17:30:08.665	rapport de la consultation.....	2ea3522f-a4ca-41d0-9687-fa0579cbc9f1

(1 row, 14 ms)

2. La table MEDECIN :

```
SELECT * FROM MEDECIN;
```

ID	EMAIL	NOM	SPECIALITE
1	medecin1@gmail.com	medecin1	cardio
2	medecin2@gmail.com	medecin2	dentiste
3	medecin3@gmail.com	medecin3	cardio

(3 rows, 4 ms)

3. La table PATIENT :

```
SELECT * FROM PATIENT;
```

ID	DATE_NAISSANCE	MALADE	NOM
1	2022-05-06	FALSE	nom1
2	2022-05-06	FALSE	nom2
3	2022-05-06	FALSE	nom3

(3 rows, 5 ms)

4. La table RENDEZ_VOUS :

```
SELECT * FROM RENDEZ_VOUS;
```

ID	DATE	STATUS	MEDECIN_ID	PATIENT_ID
2ea3522f-a4ca-41d0-9687-fa0579cbc9f1	2022-05-06 17:30:08.566	pending	1	1

(1 row, 18 ms)

5. Le contrôleur de l’application :

Dans ce contrôleur j’ai déclaré une fonction qui permet de retourner la liste des patients qui sont enregistrés dans la base de données sous format JSON et j’ai lui affecté la route /patients voici le code et le résultat obtenu :

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

```
[
  {
    "id": 1,
    "nom": "nom1",
    "date_naissance": "2022-05-06",
    "malade": false,
    "rendezVousCollection": [
      {
        "id": "2ea3522f-a4ca-41d0-9687-fa0579cbc9f1",
        "date": "2022-05-06T17:30:08.566+00:00",
        "status": "pending",
        "medecin": {
          "id": 1,
          "nom": "medecin1",
          "email": "medecin1@gmail.com",
          "specialite": "cardio"
        },
        "consultation": {
          "id": 1,
          "dateConsultation": "2022-05-06T17:30:08.665+00:00",
          "rapport": "rapport de la consultation....."
        }
      }
    ]
  },
  {
    "id": 2,
    "nom": "nom2",
    "date_naissance": "2022-05-06",
    "malade": false,
    "rendezVousCollection": []
  },
  {
    "id": 3,
    "nom": "nom3",
    "date_naissance": "2022-05-06",
    "malade": false,
    "rendezVousCollection": []
  }
]
```