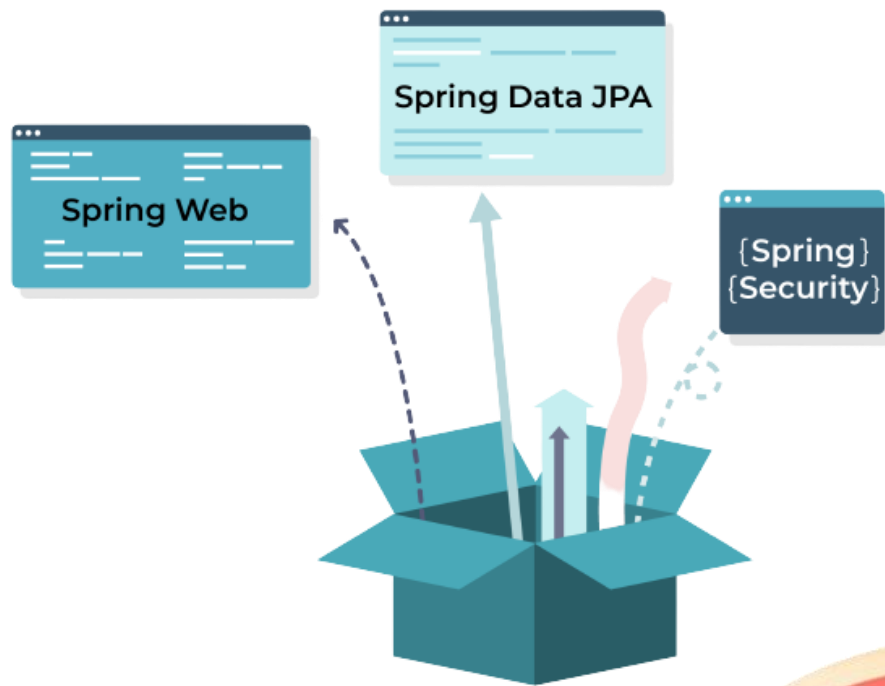


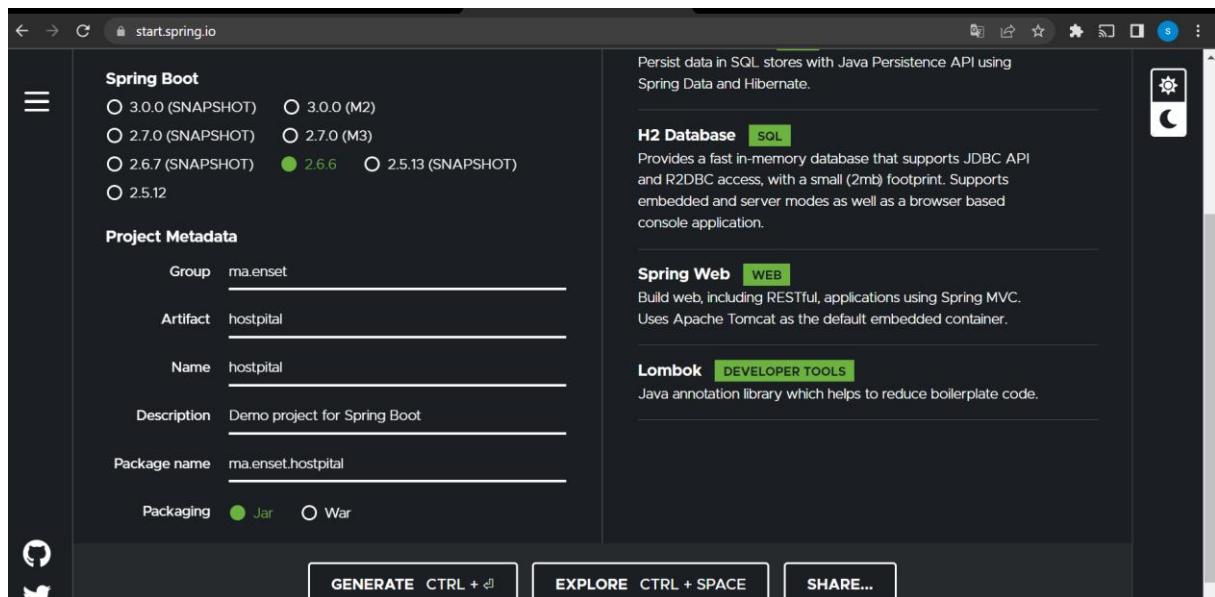
Compte rendu de :

SEANCE 04



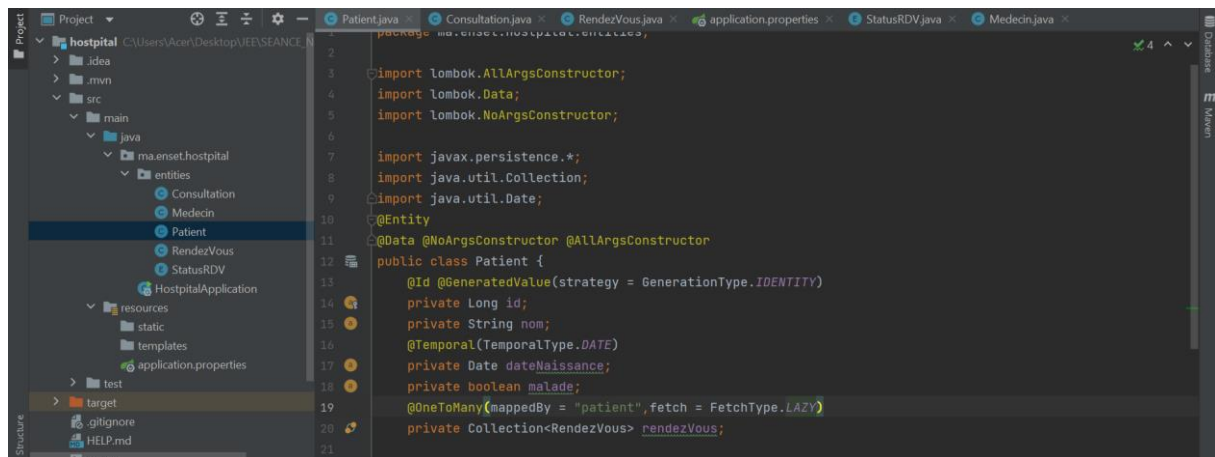
Réalisé Par : AYOUIJIL Soukayna
Département : Mathématiques et Informatique
Filière : II-BDCC
Professeur : M. Mohamed YOUSSEFI

Etape 1 : Création d'un projet spring à partir :

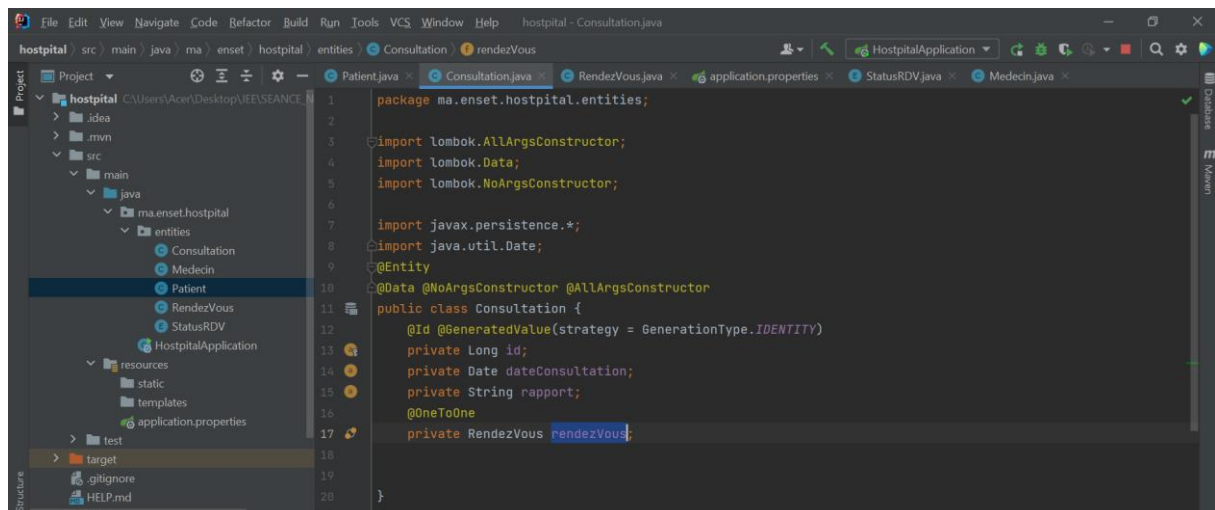


Etape 2 : Gestion des entités

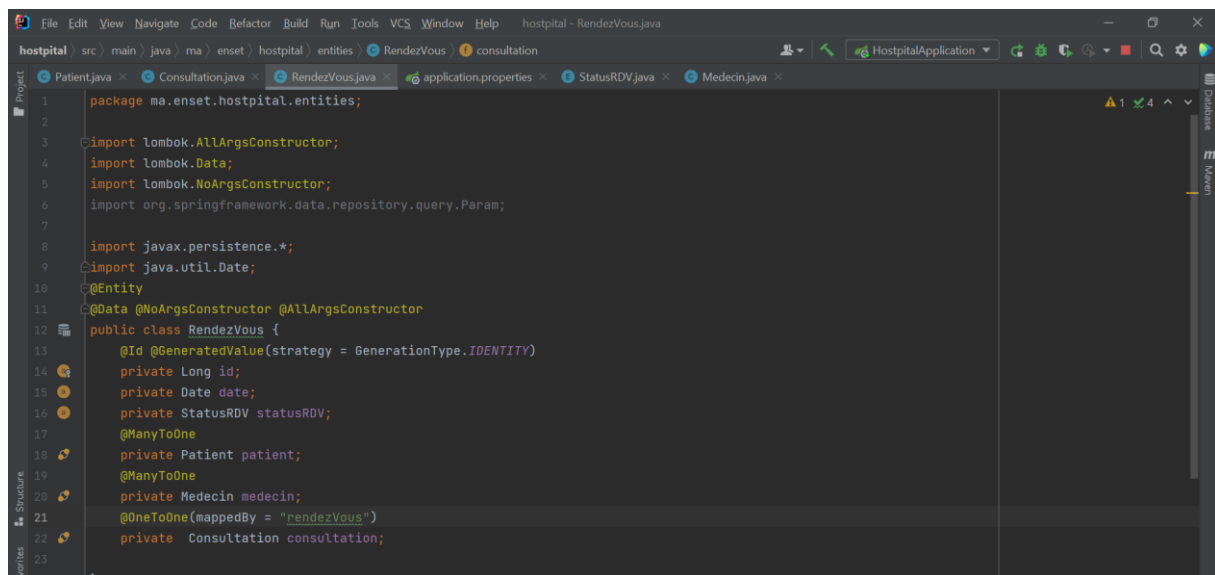
- Patient



- Consultation

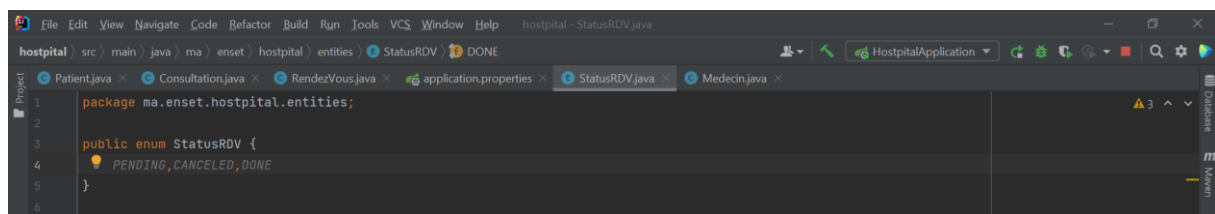


- Rendez-Vous



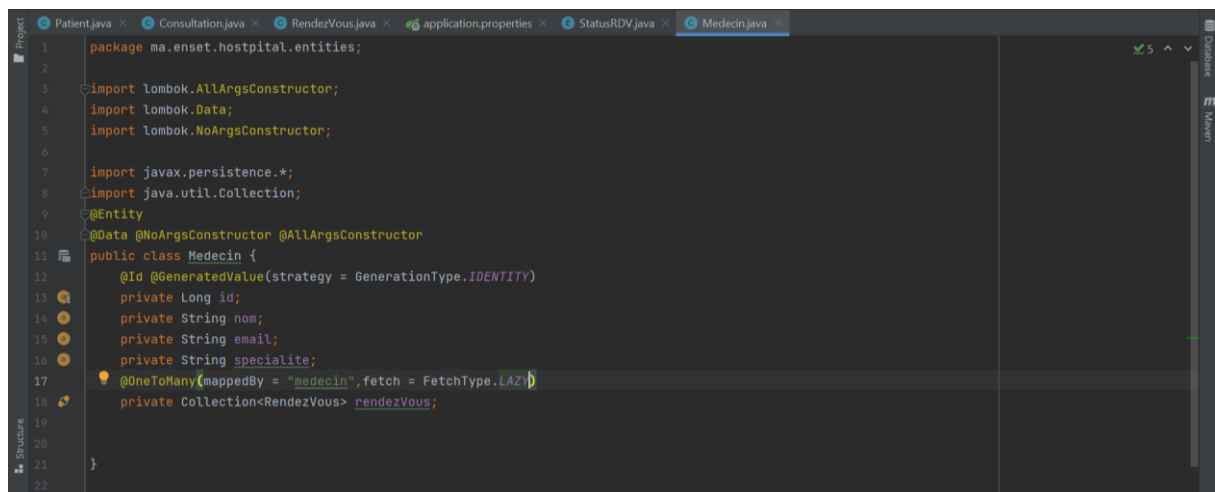
```
1 package ma.enset.hostpital.entities;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6 import org.springframework.data.repository.query.Param;
7
8 import javax.persistence.*;
9 import java.util.Date;
10
11 @Entity
12 @Data @NoArgsConstructor @AllArgsConstructor
13 public class RendezVous {
14     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Long id;
16     private Date date;
17     private StatusRDV statusRDV;
18     @ManyToOne
19     private Patient patient;
20     @ManyToOne
21     private Medecin medecin;
22     @OneToOne(mappedBy = "rendezVous")
23     private Consultation consultation;
```

- Enum : StatusRDV

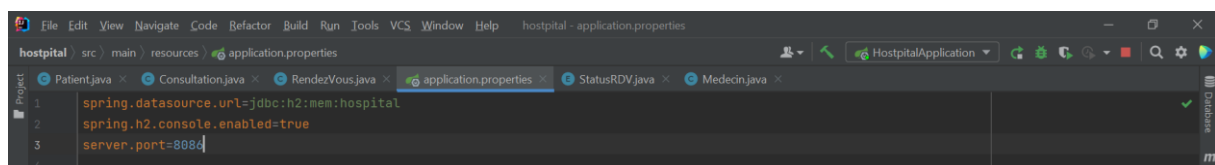


```
1 package ma.enset.hostpital.entities;
2
3 public enum StatusRDV {
4     PENDING, CANCELED, DONE
5 }
6
```

- Medecin



```
1 package ma.enset.hostpital.entities;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6
7 import javax.persistence.*;
8 import java.util.Collection;
9
10 @Entity
11 @Data @NoArgsConstructor @AllArgsConstructor
12 public class Medecin {
13     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Long id;
15     private String nom;
16     private String email;
17     private String specialite;
18     @OneToMany(mappedBy = "medecin", fetch = FetchType.LAZY)
19     private Collection<RendezVous> rendezVous;
20 }
21
22
```



```
1 spring.datasource.url=jdbc:h2:mem:hospital
2 spring.h2.console.enabled=true
3 server.port=8080
4
```

Etape 3 : Démarrage de l'application spring

Important Commands

Icon	Description
?	Displays this Help Page
📜	Shows the Command History
⏎	Executes the current SQL statement
⇧⏎	Executes the SQL statement defined by the text selection
⌘␣	Auto complete
🔌	Disconnects from the database

Sample SQL Script

Action	SQL Statement
Delete the table if it exists	DROP TABLE IF EXISTS TEST;
Create a new table with ID and NAME columns	CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
Add a new row	INSERT INTO TEST VALUES(1, 'Hello');
Add another row	INSERT INTO TEST VALUES(2, 'World');
Query the table	SELECT * FROM TEST ORDER BY ID;
Change data in a row	UPDATE TEST SET NAME='H' WHERE ID=1;
Remove a row	DELETE FROM TEST WHERE ID=2;
Help	HELP ...

Etape 4 : Ajout des patients

```
Stream.of("Mohammed", "Hssan", "Najat").forEach(
    name->{
        Patient patient = new Patient();
        patient.setNom(name);
        patient.setDateNaissance(new Date());
        patient.setMalade(false);
        patientRepository.save(patient);
    }
);
```

Etape 5 : Résultats

SELECT * FROM PATIENT;

ID	DATE_NAISSANCE	MALADE	NOM
1	2022-04-17	FALSE	Mohammed
2	2022-04-17	FALSE	Hssan
3	2022-04-17	FALSE	Najat

(3 rows, 10 ms)

Etape 6 : Ajout des medecins

```
Stream.of("Hayat", "Aya", "Soukayna").forEach(
    name->{
        Medecin medecin = new Medecin();
        medecin.setNom(name);
        medecin.setSpecialite(Math.random() > 0.5 ? "Cardio" : "Dentiste");
        medecin.setEmail(name + "@gmail.com");
        medecinRepository.save(medecin);
    }
);
```

Etape 7 : Résultats

Run	Run Selected	Auto complete	Clear	SQL statement:
SELECT * FROM MEDECIN				
SELECT * FROM MEDECIN;				
ID	EMAIL	NOM	SPECIALITE	
1	Hayat@gmail.com	Hayat	Dentiste	
2	Aya@gmail.com	Aya	Cardio	
3	Soukayna@gmail.com	Soukayna	Cardio	
(3 rows, 7 ms)				
Edit				

Etape 8 : Ajout des rendez-vous

```
50
51 Patient patient = patientRepository.findById(1L).orElse( other: null);
52 Patient patient1 = patientRepository.findById(1L).orElse( other: null);
53 Medecin medecin = medecinRepository.findById(2L).orElse( other: null);
54 RendezVous rendezVous = new RendezVous();
55 rendezVous.setDate(new Date());
56 rendezVous.setStatusRDV(StatusRDV.PENDING);
57 rendezVous.setMedecin(medecin);
58 rendezVous.setPatient(patient);
59 rendezVousRepository.save(rendezVous);
60
61 }
```

Etape 9 : Résultats

H2 Console

localhost:8086/h2-console/login.do?sessionId=87ed1050602e8338941a91d7aed8dd58

Auto commit

Max rows: 1000

Auto complete

Auto select

jdbc:h2:mem:hospital

CONSULTATION

MEDECIN

PATIENT

RENDEZ_VOUS

INFORMATION_SCHEMA

Sequences

Users

H2 1.4.200 (2019-10-14)

Run

Run Selected

Auto complete

Clear

SQL statement:

SELECT * FROM RENDEZ_VOUS

SELECT * FROM RENDEZ_VOUS;

ID	DATE	STATUSRDV	MEDECIN_ID	PATIENT_ID
1	2022-04-17 03:04:53.298	PENDING	2	1

(1 row, 3 ms)

Etape 10 : Ajout des consultations

```
Consultation consultation = new Consultation();
RendezVous rendezVous1=rendezVousRepository.findById(1L).orElse( other: null);
consultation.setDateConsultation(new Date());
consultation.setRendezVous(rendezVous1);
consultation.setRapport("Rapport de consultation ....");

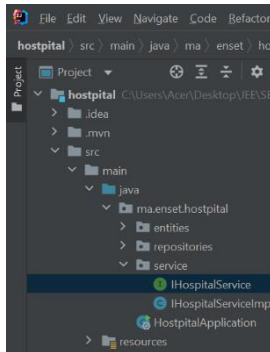
consultationRepository.save(consultation);
```

Etape 10 : Résultats

Run	Run Selected	Auto complete	Clear	SQL statement:
SELECT * FROM CONSULTATION				
SELECT * FROM CONSULTATION;				
ID	DATE_CONSULTATION	RAPPORT	RENDEZ_VOUS_ID	
1	2022-04-17 02:35:25.351	Rapport de consultation	1	
(1 row, 10 ms)				

Etape 11 : Création de la couche Service

- L'interface IHospitalService



```
1 package ma.enset.hospital.service;
2
3 import ...
4
5
6
7
8
9
10 public interface IHospitalService {
11     Patient savePatient(Patient patient);
12     Medecin saveMedecin(Medecin medecin);
13     RendezVous saveRendezVous(RendezVous rendezVous);
14     Consultation saveConsultation(Consultation consultation);
15     Patient findByNomPatient(String name);
16     Optional<Patient> findByIdPatient(Long id);
17     Medecin findByNomMedecin(String name);
18     Optional<RendezVous> findByIdRDV(Long id);
19 }
```

- Implémentation de l'interface IHospitalService

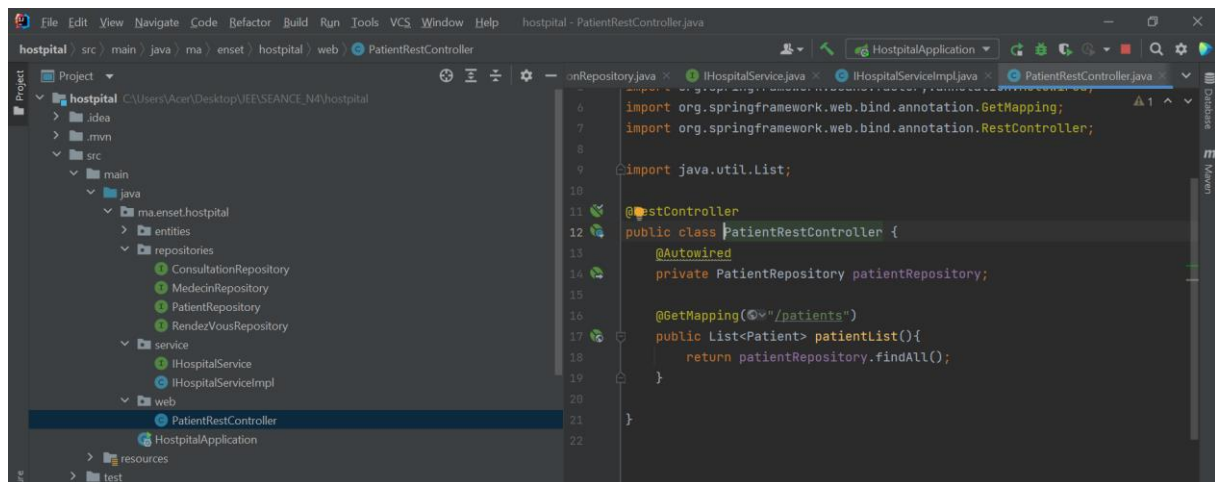
The screenshot shows the implementation of the `IHospitalService` interface in the `IHospitalServiceImpl` class. The class is annotated with `@Service` and `@Transactional`. It implements the `savePatient` method by delegating the call to the `patientRepository`.

```
14
15 @Service
16 @Transactional
17 public class IHospitalServiceImpl implements IHospitalService {
18     private PatientRepository patientRepository;
19     private MedecinRepository medecinRepository;
20     private RendezVousRepository rendezVousRepository;
21     private ConsultationRepository consultationRepository;
22
23     public IHospitalServiceImpl(PatientRepository patientRepository, MedecinRepository medecinRepository, RendezVousRepository rendezVousRepository, ConsultationRepository consultationRepository) {
24         this.patientRepository = patientRepository;
25         this.medecinRepository = medecinRepository;
26         this.rendezVousRepository = rendezVousRepository;
27         this.consultationRepository = consultationRepository;
28     }
29
30     @Override
31     public Patient savePatient(Patient patient) {
32         return patientRepository.save(patient);
33     }
34 }
```

The screenshot shows the implementation of the `IHospitalService` interface in the `IHospitalServiceImpl` class, focusing on the `find` methods. The class implements the `findByNomPatient`, `findByIdPatient`, `findByNomMedecin`, and `findByIdRDV` methods by delegating the calls to the respective repositories.

```
49
50
51 @Override
52 public Patient findByNomPatient(String name) {
53     return patientRepository.findByNom(name);
54 }
55
56 @Override
57 public Optional<Patient> findByIdPatient(Long id) {
58     return patientRepository.findById(id);
59 }
60
61 @Override
62 public Medecin findByNomMedecin(String name) {
63     return medecinRepository.findByNom(name);
64 }
65
66 @Override
67 public Optional<RendezVous> findByIdRDV(Long id) {
68     return rendezVousRepository.findById(id);
69 }
70 }
```

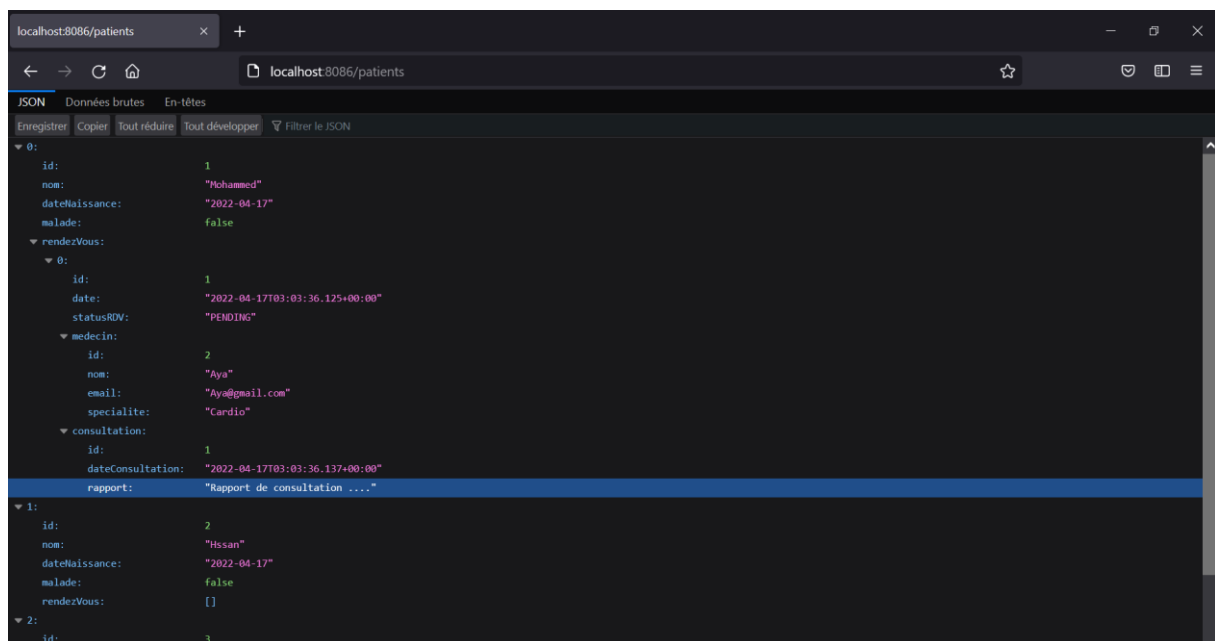
Etape 11 : Création de la couche Web - PatientRestController



The screenshot shows an IDE with the 'hospital' project structure on the left. The 'web' package contains 'PatientRestController.java'. The code in the file is as follows:

```
1 import org.springframework.web.bind.annotation.GetMapping;
2 import org.springframework.web.bind.annotation.RestController;
3
4 import java.util.List;
5
6 @RestController
7 public class PatientRestController {
8
9     @Autowired
10     private PatientRepository patientRepository;
11
12     @GetMapping("/patients")
13     public List<Patient> patientList(){
14         return patientRepository.findAll();
15     }
16 }
```

Etape 12 : Resultats



The screenshot shows a web browser displaying the JSON response for the GET /patients endpoint. The response is a list of three patient objects. The first patient has a consultation report.

```
{
  "id": 1,
  "nom": "Mohammed",
  "dateNaissance": "2022-04-17",
  "malade": false,
  "rendezVous": {
    "id": 1,
    "date": "2022-04-17T03:03:36.125+00:00",
    "statusRDV": "PENDING",
    "medecin": {
      "id": 2,
      "nom": "Aya",
      "email": "Aya@gmail.com",
      "specialite": "Cardio"
    },
    "consultation": {
      "id": 1,
      "dateConsultation": "2022-04-17T03:03:36.137+00:00",
      "rapport": "Rapport de consultation ...."
    }
  }
},
{
  "id": 2,
  "nom": "Hssan",
  "dateNaissance": "2022-04-17",
  "malade": false,
  "rendezVous": []
},
{
  "id": 3
}
```