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 de l'activité pratique 4

Gérer les associations et l'héritage entre les entités Partie I

Filière :
« Génie du Logiciel et des Systèmes Informatiques Distribués »
GLSID2

Module : Architecture Distribuée et Middlewares

Élaboré par :

ELMAJNI Khaoula

Encadré par :

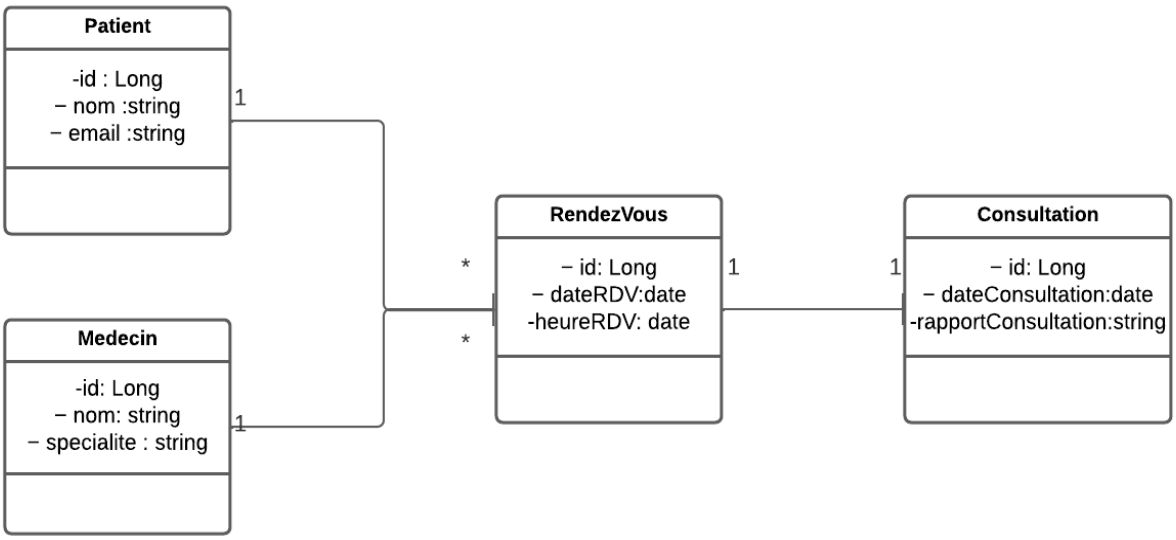
M. YOUSSFI Mohammed

Année Universitaire : 2021-2022

Introduction

Dans ce travail on va initier avec le Mapping Objet Relationnel (ORM), étant donné comme faire une relation et correspondance entre les objets d'une application et les données stockées dans une table dans une BDDR, et la gestion des associations.

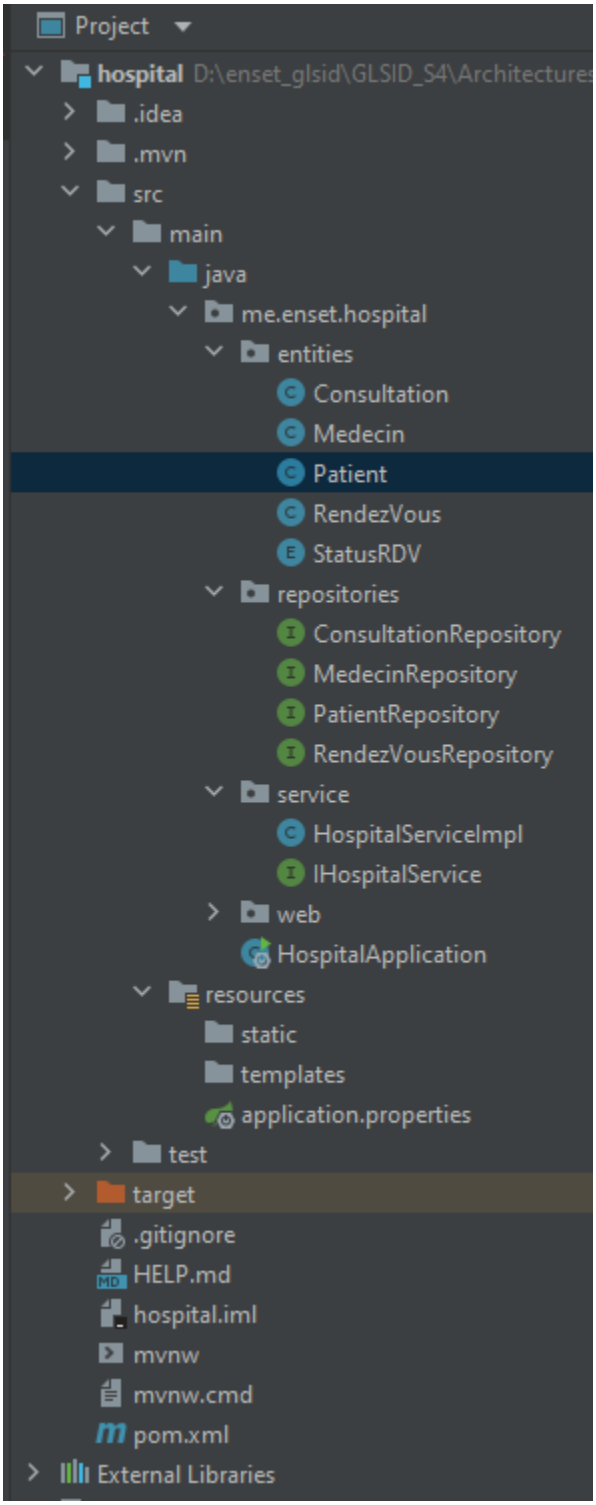
Le modèle conceptuel de données de l'application



Cette application rassemble les différentes associations possibles dans une application à travers ses entités :

- @OneToMany
- @ManyToOne
- @ManyToMany
- @OneToOne

Structure technique du projet :



Exigences fonctionnelles :

L'application permet de :

- Gérer des les patients :
    1. Ajouter un patient
    2. Consulter tous les patients
    3. Consulter les patients dont le nom contient un mot clé.
- Gérer les médecins :
    1. Ajouter un médecin
    2. Consulter un medecin
- Gérer les rendez-vous :
    1. ajouter un rendez-vous qui concerne un patient et un médecin
    2. Consulter les rendez-vous
- Gérer les consultations :
    1. Ajouter une consultation qui concerne un rendez-vous

Les entités JPA :

Entité Consultation:

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

Entité Medecin :

```
@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;
}
```

Entité Patient:

```
@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;
}
```

Entité Rendez-vous :

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class RendezVous {
    @Id //@GeneratedValue(strategy =
GenerationType.IDENTITY)
    private String id;
    private Date dateCreation;
    //pour le stockage de type string
    @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;
}
```

Les interfaces JPA

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

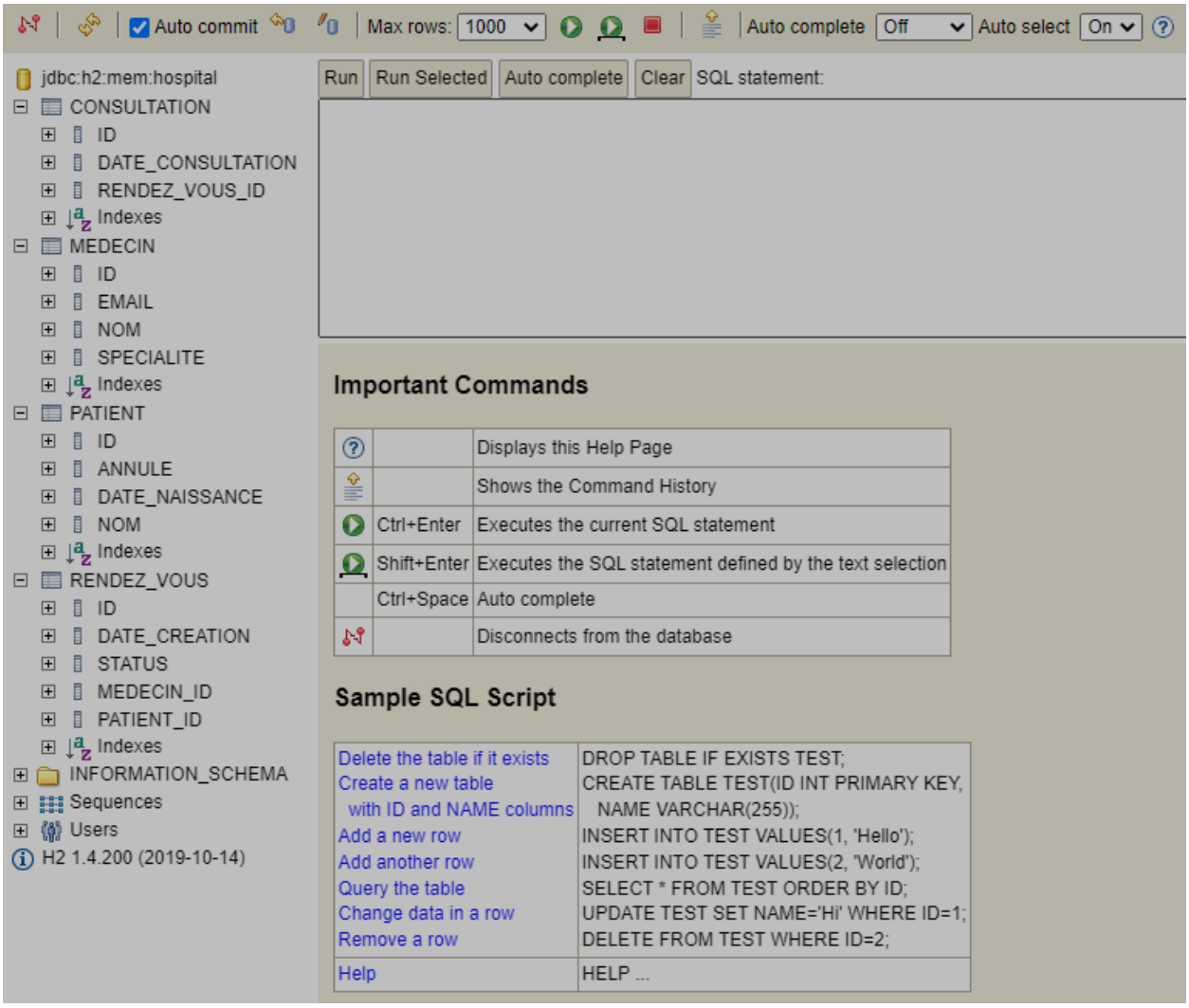
```
public interface MedecinRepository extends
JpaRepository<Medecin, Long> {
    Medecin findByNom(String nom);
}
```

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

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

Résultat

La base de données h2 générée(les tables avec les différentes associations) :



L'affichage des patients de la base de données :

Le controleur des patients (/patients)

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


localhost:8088/patients
Boîte de réception (...
Free Web Hosting...
Stack Overflow - W...
Free to Use Clip Art...
KhaoulaElmajni
[
{
"id": 1,
"nom": "moahemmed",
"dateNaissance": "2022-03-12",
"malade": false,
"rendezVous": [
{
"id": "acdbdc5e-812e-4905-933b-022c5135ea0d",
"dateCreation": "2022-03-12T10:29:22.517+00:00",
"status": "CANCELED",
"medecin": {
"id": 1,
"nom": "khadija",
"email": "khadija@gmail.com",
"specialite": "chirurgie"
},
"consultation": {
"id": 1,
"dateConsultation": "2022-03-12T10:29:22.560+00:00",
"rapportConsultation": "rapport..."
}
}
]
},
{
"id": 2,
"nom": "khaoula",
"dateNaissance": "2022-03-12",
"malade": false,
"rendezVous": []
},
{
"id": 3,
"nom": "ahmed",
"dateNaissance": "2022-03-12",
"malade": false,
"rendezVous": []
}
]
1 // 20220312112936

Conclusion

Ce travail nous a permis de bien connaitre comment le Mapping Objet Relationnel se fait au niveau du Java grace à l'API JPA, et on a initié avec Hibernate, et bien évidement la manière de configuration d'un projet Spring pour faciliter la tache au développeur.