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

Spring MVC JPA Hibernate Spring Data Thymeleaf

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

```
2. package ma.enset.entities;

   import jakarta.persistence.*;
   import jakarta.validation.constraints.DecimalMin;
   import jakarta.validation.constraints.NotEmpty;
   import jakarta.validation.constraints.Size;
   import lombok.AllArgsConstructor;
   import lombok.Data;
   import lombok.NoArgsConstructor;
   import org.springframework.format.annotation.DateTimeFormat;

   import java.util.Date;
   // Data => setters et getters  NoArgsConstructor => constructeur sans
   parametre
   @Entity //Notion de JPA
   @Data @AllArgsConstructor @NoArgsConstructor
   public class Patient {
       @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
       private Long id;
       //annotation de validation
       @NotEmpty
       @Size(min=4,max=40)
       private String nom;
       @Temporal(TemporalType.DATE) //garder la forme de date
       @DateTimeFormat(pattern = "yyyy-MM-dd")
       private Date dateNaissance;
       private boolean malade;
       @DecimalMin("100")
       private int score;
   }
```

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

```java
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;

public interface PatientRespository extends JpaRepository<Patient,Long> {
 Page<Patient> findByNomContains(String Recherche, Pageable pageable);
}
```

3. 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

```java
package ma.enset;

import ma.enset.entities.Patient;
import ma.enset.repositories.PatientRespository;
import org.hibernate.tool.schema.spi.CommandAcceptanceException;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

import java.util.Date;

@SpringBootApplication
public class Devoir4Application {

    public static void main(String[] args) {
        SpringApplication.run(Devoir4Application.class, args);
    }
    // l'exection de programme dans CommandLineRunner
        //@Bean=> execution de fonction CommandLineRunner
        CommandLineRunner commandLineRunner(PatientRespository
patientRespository){
        return  args -> {
            //Insertition des patients
            patientRespository.save(new Patient(null,"Najwa",new
Date(),false,212));
            patientRespository.save(new Patient(null,"Islam",new
Date(),true,122));
            patientRespository.save(new Patient(null,"Khalid",new
Date(),true,854));
            patientRespository.save(new Patient(null,"Hania",new
Date(),false,300));

            //Affiche la liste des patients
            patientRespository.findAll().forEach(p->{
```

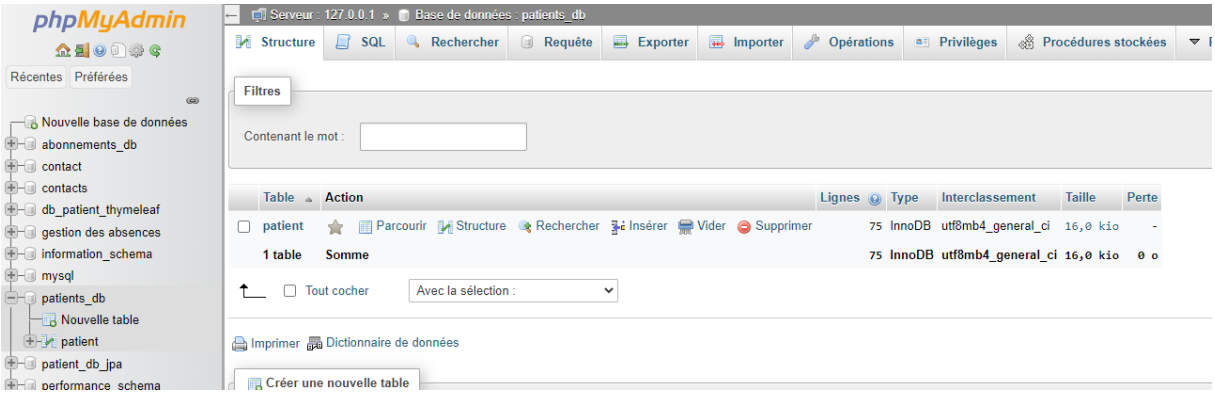
```
                    System.out.println(p.getNom());
                });
        };
        }
        @Bean
        PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder();
        }

}
```

5. Migrer de MySQL

⇨ Ajout de dépendance :

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

```
server.port=1806
spring.datasource.url=jdbc:mysql://localhost:3306/Patients_DB?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
```



7. PatientController

```
package ma.enset.web;

import jakarta.validation.Valid;
import lombok.AllArgsConstructor;
import ma.enset.entities.Patient;
```

```java
import ma.enset.repositories.PatientRespository;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

import java.util.List;

//Pour une application nous avons besoin d'un Controller
@Controller
//Injection des dependances via Constructor
@AllArgsConstructor
public class PatientController {
    private PatientRespository patientRespository;

//acces à la methode patient
    @GetMapping(path="/user/index")
    // la methode patient return un vue
    //Module de spring MVC
    public String patient(Model model,
                          //pagination
                           @RequestParam(name="page",defaultValue =
"0") int page,
                          //size de page
                          @RequestParam(name="size",defaultValue =
"5") int size,
                          @RequestParam(name="Recherche",defaultValue
= "") String Recherche

    ){
        Page<Patient>
pagePatients=patientRespository.findByNomContains(Recherche,PageReque
st.of(page, size));
        // stocker dans le modul
        model.addAttribute("listPatients",pagePatients.getContent());
        model.addAttribute("pages",new
int[pagePatients.getTotalPages()]);
        model.addAttribute("currentPage",page);
        model.addAttribute("Recherche",Recherche);
        return "patients";
   }
@GetMapping("/admin/delete")
@PreAuthorize("hasRole('ROLE_ADMIN')")
public  String delete(long id,String Recherche,int page ){
        patientRespository.deleteById(id);
        return
"redirect:/user/index?page="+page+"&Recherche="+Recherche;
}
@GetMapping("/")
    public  String home(){
        return "redirect:/user/index";
    }

    @GetMapping("/user/patients")
    @ResponseBody
```

```
    public List<Patient> patientList(){
        return  patientRespository.findAll();
    }

    @GetMapping("/admin/formPatients")
    @PreAuthorize("hasRole('ROLE_ADMIN')")
public String formPatient(Model model){
        model.addAttribute("patient",new Patient());
        return "formPatients";
    }
    @PostMapping(path="/admin/save")
    @PreAuthorize("hasRole('ROLE_ADMIN')")
    public  String save(Model model, @Valid Patient patient,
BindingResult bindingResult,
                        @RequestParam(defaultValue = "0") int page,
                        @RequestParam(defaultValue = "")String
Recherche){
        if(bindingResult.hasErrors()) return "formPatients";
        patientRespository.save(patient);
        return
"redirect:/user/index?page="+page+"&Recherche="+Recherche;
    }
    @GetMapping("/admin/editPatients")
    @PreAuthorize("hasRole('ROLE_ADMIN')")
    public String editPatients(Model model,Long id,String
Recherche,int page){
        Patient patient=patientRespository.findById(id).orElse(null);
        if(patient==null) throw new RuntimeException("Patient
introuvable");
        model.addAttribute("patient",patient);
        model.addAttribute("page",page);
        model.addAttribute("Recherche",Recherche);

        return "editPatients";
    }
}
```

- Les ressources de Template avec la lisions de thymleaf cote serveur :

  - Template.html

```
- <!DOCTYPE html>
  <html lang="en" xmlns:th="http://www.thymeleaf.org"
        xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
  <head>
      <meta charset="UTF-8">
      <title>Gestion des patients</title>
      <link rel="stylesheet"
  href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
      <link rel="stylesheet" href="/webjars/bootstrap-
  icons/1.10.3/font/bootstrap-icons.css">
      <script
  src="/webjars/bootstrap/5.2.3/js/bootstrap.bundle.min.js"></script
  >
  </head>
  <body>

  <nav class="navbar navbar-expand-sm bg-dark navbar-dark">
      <div class="container-fluid">
```

```
            <ul class="navbar-nav">
                <li class="nav-item"><a class="navbar-brand"
    href="#"> <b>   G</b>estion<br>des
    <b>P</b>atients</a></li>
                <li class="nav-item">
                    <a class="nav-link active"
    th:href="@{/user/index}">Home</a>
                </li>
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" href="#"
    role="button" data-bs-toggle="dropdown">Patients</a>
                    <ul class="dropdown-menu">
                        <li><a class="dropdown-item"
    th:if="${#authorization.expression('hasRole(''ADMIN'')')}"
    th:href="@{/admin/formPatients}" ><b>N</b>ouveau</a></li>
                        <li><a class="dropdown-item"
    th:href="@{/user/index}"><b>C</b>hercher</a></li>
                    </ul>
                </li>
            </ul>
            <ul class="navbar-nav">
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" href="#"
    role="button" data-bs-toggle="dropdown"
                       th:text="${#authentication.name}"></a>
                    <ul class="dropdown-menu">
                        <li>
                            <form method="post"
    th:action="@{/logout}">
                                <button class="dropdown-item"
    type="submit"><b>L</b>ogout</button>
                            </form>
                        </li>
                        <li><a class="dropdown-item"
    href="#"><b>P</b>rofile</a></li>
                    </ul>
                </li>
            </ul>
        </div>
    </nav>

    <div layout:fragment="content">

    </div>

    </body>
    </html>
```

- Patients.html

```
<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="template"
>
    <meta charset="UTF-8">
    <title>Gestion des patients</title>
```

```
    <link rel="stylesheet"
href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
    <head>
</head>
<body>
<div layout:fragment="content">
<div class="container mt-4" >
    <div class="card">
        <div class="card-header">
            <h3>Liste des patients</h3>
        </div>
        <div class="card-body">
            <form method="get" th:action="@{/user/index}">
                <label>Recherche : </label>
                <input type="text" name="Recherche"
th:value="${Recherche}">
                <button type="submit" class="btn btn-outline-primary"><svg
xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-search" viewBox="0 0 16 16">
                    <path d="M11.742 10.344a6.5 6.5 0 1 0-1.397 1.398h-
.001c.03.04.062.078.098.115l3.85 3.85a1 1 0 0 0 1.415-1.414l-3.85-
3.85a1.007 1.007 0 0 0-.115-.1zM12 6.5a5.5 5.5 0 1 1-11 0 5.5 5.5 0 0 1 11
0z"/>
                </svg></button>
            </form>
            <table class="table">
                <thead>
                <tr>
                    <th>ID</th>
                    <th>NOM</th>
                    <th>DATE</th>
                    <th>MALADE</th>
                    <th>SCORE</th>
                </tr>
                </thead>
                <tbody>
                <tr th:each="p:${listPatients}">
                    <td th:text="${p.id}"></td>
                    <td th:text="${p.getNom()}"></td>
                    <td th:text="${p.dateNaissance}"></td>
                    <td th:text="${p.malade}"></td>
                    <td th:text="${p.getScore()}"></td>
                    <td
th:if="${#authorization.expression('hasRole(''ADMIN'')')}">
                        <a   class="btn btn-outline-success"
                             th:href="@{
                        /admin/editPatients(id=${p.id},
                        Recherche=${Recherche},
                        page=${currentPage})}"
                        ><svg xmlns="http://www.w3.org/2000/svg" width="16"
height="16" fill="currentColor" class="bi bi-pencil-square" viewBox="0 0 16
16">
                            <path d="M15.502 1.94a.5.5 0 0 1 0 .706L14.459
3.69l-2-2L13.502.646a.5.5 0 0 1 .707 0l1.293 1.293zm-1.75 2.456-2-2L4.939
9.21a.5.5 0 0 0-.121.196l-.805 2.414a.25.25 0 0 0 .316.316l2.414-.805a.5.5
0 0 0 .196-.12l6.813-6.814z"/>
                            <path fill-rule="evenodd" d="M1 13.5A1.5 1.5 0
0 0 2.5 15h11a1.5 1.5 0 0 0 1.5-1.5v-6a.5.5 0 0 0-1 0v6a.5.5 0 0 1-.5.5h-
11a.5.5 0 0 1-.5-.5v-11a.5.5 0 0 1 .5-.5H9a.5.5 0 0 0 0-1H2.5A1.5 1.5 0 0 0
1 2.5v11z"/>
                        </svg></a>
```

```
                    </td>
                    <td
th:if="${#authorization.expression('hasRole(''ADMIN'')')}">
                        <a  onclick="return confirm('Etes vous sure ?')"
class="btn btn-outline-danger"
                            th:href="@{/admin/delete(
                        id=${p.getId()},
                        Recherche=${Recherche},
                        page=${currentPage})}"
                        ><svg xmlns="http://www.w3.org/2000/svg" width="16"
height="16" fill="currentColor" class="bi bi-trash" viewBox="0 0 16 16">
                            <path d="M5.5 5.5A.5.5 0 0 1 6 6v6a.5.5 0 0 1-1
0V6a.5.5 0 0 1 .5-.5zm2.5 0a.5.5 0 0 1 .5.5v6a.5.5 0 0 1-1 0V6a.5.5 0 0 1
.5-.5zm3 .5a.5.5 0 0 0-1 0v6a.5.5 0 0 0 1 0V6z"/>
                            <path fill-rule="evenodd" d="M14.5 3a1 1 0 0 1-
1 1H13v9a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2V4h-.5a1 1 0 0 1-1-1V2a1 1 0 0 1 1-
1H6a1 1 0 0 1 1-1h2a1 1 0 0 1 1 1h3.5a1 1 0 0 1 1 1v1zM4.118 4 4 4.059V13a1
1 0 0 0 1 1h6a1 1 0 0 0 1-1V4.059L11.882 4H4.118zM2.5 3V2h11v1h-11z"/>
                        </svg></a>
                    </td>
                </tr>
                </tbody>
            </table>
            <ul class="nav nav-pills">
                <li th:each="page,status : ${pages}">
                    <a th:class="${status.index==currentPage? 'btn btn-dark
ms-1':'btn btn-outline-dark ms-1'}"
                       th:text="${status.index }"

th:href="@{/user/index(page=${status.index},Recherche=${Recherche})}"
                    ></a>
                </li>
            </ul>
        </div>
    </div>
</div>
</div>
</body>
</html>
```

- formPatient.html

```
- <!DOCTYPE html>
   <html lang="en"
         xmlns:th="http://www.thymeleaf.org"
         xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
         layout:decorate="template"
   >
   <meta charset="UTF-8">
   <title>Formulaire de patient</title>
   <link rel="stylesheet"
   href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
   <head>
   </head>
   <body>
   <div layout:fragment="content">
     <div class="col-md-6 offset-3">
   <form method="post" th:action="@{save}">
     <div>
       <label for="nom">Nom :</label>
```

```
        <input id="nom" type="text" class="form-control"  name="nom"
    th:value="${patient.nom}">
        <span  class="text-danger" th:errors="${patient.nom}"></span>
      </div>
      <div>
        <label>Date Naissance :</label>
        <input type="date" class="form-control"  name="dateNaissance"
    th:value="${patient.dateNaissance}">
        <span class="text-danger"
    th:errors="${patient.dateNaissance}"></span>
      </div>
      <div>
        <label>Malade :</label>
        <input type="checkbox" class="form-control form-check-input"
    name="malade" th:checked="${patient.malade}">
        <span class="text-danger"
    th:errors="${patient.malade}"></span>
      </div>
      <div>
        <label>Score :</label>
        <input type="text" class="form-control"  name="score"
    th:value="${patient.score}">
        <span class="text-danger" th:errors="${patient.score}"></span>
      </div>
      <button type="submit" class="btn btn-outline-primary"
    >Enregistrer</button>
    </form>
    </div>
    </div>
    </body>
    </html>
```

- editPatient.html

```
<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="template"
>
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<div layout:fragment="content">
  <div class="col-md-6 offset-3">
    <form method="post"
th:action="@{save(page=${page},Recherche=${Recherche})}">
      <div>
        <label for="id">Patient N° :</label>
        <input id="id" type="text" class="form-control"  name="id"
th:value="${patient.id}" disabled>
        <span  class="text-danger" th:errors="${patient.id}"></span>
      </div>
      <div>
        <label for="nom">Nom :</label>
        <input id="nom" type="text" class="form-control"  name="nom"
```

```
th:value="${patient.nom}">
        <span  class="text-danger" th:errors="${patient.nom}"></span>
      </div>
      <div>
        <label>Date Naissance :</label>
        <input type="date" class="form-control"  name="dateNaissance"
th:value="${patient.dateNaissance}">
        <span class="text-danger"
th:errors="${patient.dateNaissance}"></span>
      </div>
      <div>
        <label>Malade :</label>
        <input type="checkbox" class="form-control form-check-input"
name="malade" th:checked="${patient.malade}">
        <span class="text-danger" th:errors="${patient.malade}"></span>
      </div>
      <div>
        <label>Score :</label>
        <input type="text" class="form-control"  name="score"
th:value="${patient.score}">
        <span class="text-danger" th:errors="${patient.score}"></span>
      </div>
      <button type="submit" class="btn btn-outline-primary"
>Enregistrer</button>
    </form>
  </div>
</div>
</body>
</html>
```

9. La partie spring security:
   - configSecurity

```
- package ma.enset.security;


   import org.springframework.beans.factory.annotation.Autowired;
   import org.springframework.context.annotation.Bean;
   import org.springframework.context.annotation.Configuration;
   import
   org.springframework.security.config.annotation.method.configuratio
   n.EnableMethodSecurity;
   import
   org.springframework.security.config.annotation.web.builders.HttpSe
   curity;
   import
   org.springframework.security.config.annotation.web.configuration.E
   nableWebSecurity;
   import org.springframework.security.core.userdetails.User;
   import org.springframework.security.core.userdetails.UserDetails;
   import
   org.springframework.security.core.userdetails.UserDetailsService;
   import
   org.springframework.security.core.userdetails.UsernameNotFoundExce
   ption;
   import
   org.springframework.security.crypto.password.PasswordEncoder;
   import
   org.springframework.security.provisioning.InMemoryUserDetailsManag
```

```
er;
import
org.springframework.security.provisioning.UserDetailsManager;
import org.springframework.security.web.SecurityFilterChain;

//classe de configuration
@Configuration
//activation de security web
@EnableWebSecurity
@EnableMethodSecurity(prePostEnabled = true)
public class SecurityConfig {
    @Autowired
        private PasswordEncoder passwordEncoder;

    @Bean
    public InMemoryUserDetailsManager
inMemoryUserDetailsManager(){
        //noop=> no password encoder
        return new InMemoryUserDetailsManager(

User.withUsername("user1").password(passwordEncoder.encode("1234")
).roles("USER").build(),

User.withUsername("user2").password(passwordEncoder.encode("1234")
).roles("USER").build(),

User.withUsername("admin").password(passwordEncoder.encode("1234")
).roles("ADMIN","USER").build()
                );
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity
httpSecurity) throws Exception {
       //httpSecurity.formLogin();
        httpSecurity.formLogin().loginPage("/login").permitAll();
        //httpSecurity.rememberMe();
        //
httpSecurity.authorizeHttpRequests().requestMatchers("webjars/**",
"h2-console/*").permitAll();
       //Toutes les requetes necessite une authentification

//httpSecurity.authorizeHttpRequests().requestMatchers("/user/**")
.hasRole("USER");

//httpSecurity.authorizeHttpRequests().requestMatchers("/admin/**"
).hasRole("ADMIN");

httpSecurity.authorizeHttpRequests().anyRequest().authenticated();

httpSecurity.exceptionHandling().accessDeniedPage("/notAuthorized"
);
        return httpSecurity.build();
    }

}
```

- SecurityController

```java
package ma.enset.web;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
@Controller
public class SecurityController {
    @GetMapping("/notAuthorized")
    public String notAuthorized(){
        return "notAuthorized";
    }

    @GetMapping("/login")
    public String login(){
        return "login";
    }
}
```

- Login.html

```html
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Authentification</title>
  <style>
  html {
    scroll-behavior: smooth;
  }

  * {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }

  body {
    font-family: -apple-system, BlinkMacSystemFont, sans-serif;
    background: #121212; /* fallback for old browsers */
    overflow-x: hidden;

    height: 100%;

    /* code to make all text unselectable */
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    -o-user-select: none;
    user-select: none;
  }

  /* Disables selector ring */
  body:not(.user-is-tabbing) button:focus,
  body:not(.user-is-tabbing) input:focus,
  body:not(.user-is-tabbing) select:focus,
  body:not(.user-is-tabbing) textarea:focus {
    outline: none;
  }

  /* ########################################################## */
```

```css
h1 {
  color: white;

  font-size: 35px;
  font-weight: 800;
}

.flex-container {
  width: 100vw;

  margin-top: 60px;

  display: flex;
  justify-content: center;
  align-items: center;
}

.content-container {
  width: 500px;
  height: 350px;
}

.form-container {
  display: flex;
  justify-content: center;
  align-items: center;

  width: 500px;
  height: 350px;

  margin-top: 5px;
  padding-top: 20px;

  border-radius: 12px;

  display: flex;
  justify-content: center;
  flex-direction: column;

  background: #1f1f1f;
  box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.199);
}

.subtitle {
  font-size: 11px;
  color:whitesmoke;
}

input {
  border: none;
  border-bottom: solid rgb(143, 143, 143) 1px;
  border-radius: 8px;
  padding: 0 15px;
  margin-bottom: 30px;
  background: none;
  color: rgba(238, 227, 227, 0.55);
  height: 35px;
  width: 350px;
}
```

```
  .submit-btn {
    cursor: pointer;
    border: none;
    border-radius: 8px;
    box-shadow: 2px 2px 7px #38d39f70;
    background: #38d39f;
    color: rgba(255, 255, 255, 0.8);
    transition: all 1s;
  }

  .submit-btn:hover {
    color: rgb(255, 255, 255);

    box-shadow: none;
  }

</style>
</head>
<body>
<div class="flex-container">
  <div class="content-container">
    <div class="form-container">
      <form method="post" th:action="@{/login}">
        <br>
        <br>
        <span class="subtitle">Nom utilisateur :</span>
        <br>
        <input type="text" name="username" value="">
        <br>
        <span class="subtitle">Mot de passe :</span>
        <br>
        <input type="password" name="password" value="">
        <br><br>
        <!--<input type="checkbox" class="form-control form-check-input"
name="remember-me" >Remember me
        <br><br>-->
        <input type="submit" value="Se connecter" class="submit-btn">
      </form>
    </div>
  </div>
</div>
</body>
</html>
```

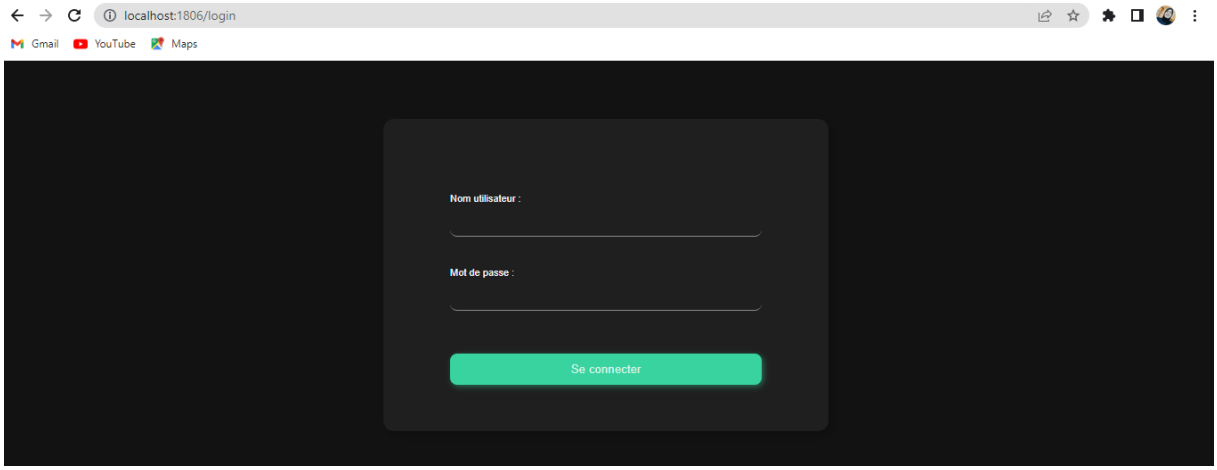
- Page d'autorisation qui s'appelle NotAuthorized

```
<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="template"
>
    <meta charset="UTF-8">
    <title>notAuthorized</title>
    <link rel="stylesheet"
href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
    <head>
</head>
```

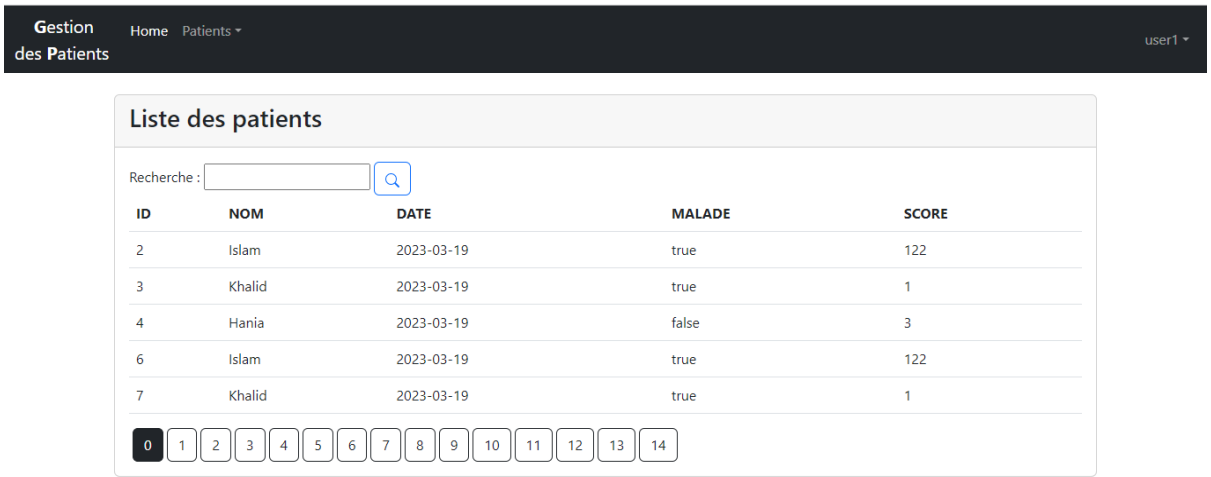
```
<body>
<div layout:fragment="content">
    <div class="alert alert-danger mt-3" >
        <h2><b>L'accès à ce page n'est pas autorisé.contacter votre
administrateur</b></h2>
    </div>
</div>
</body>
</html>
```

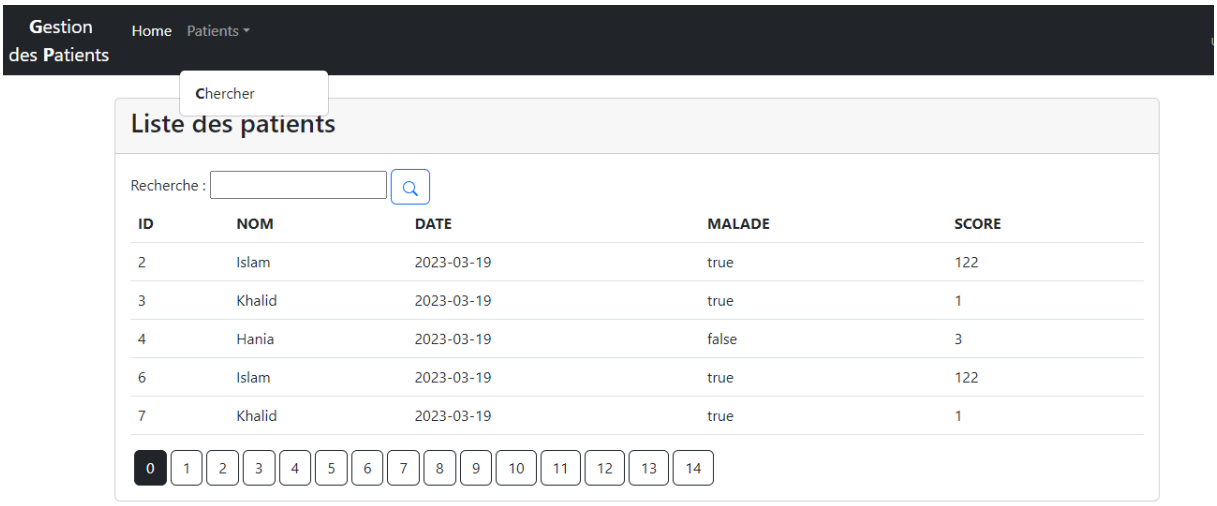
10. Démonstration :

⇨ Accès à l'application localhost : N°port (1806)

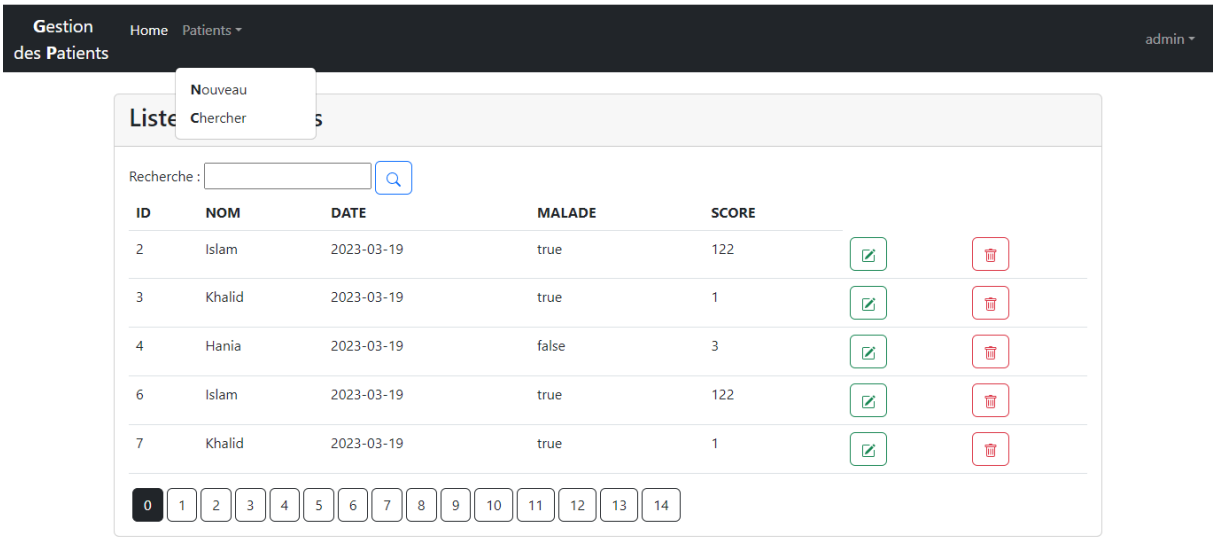


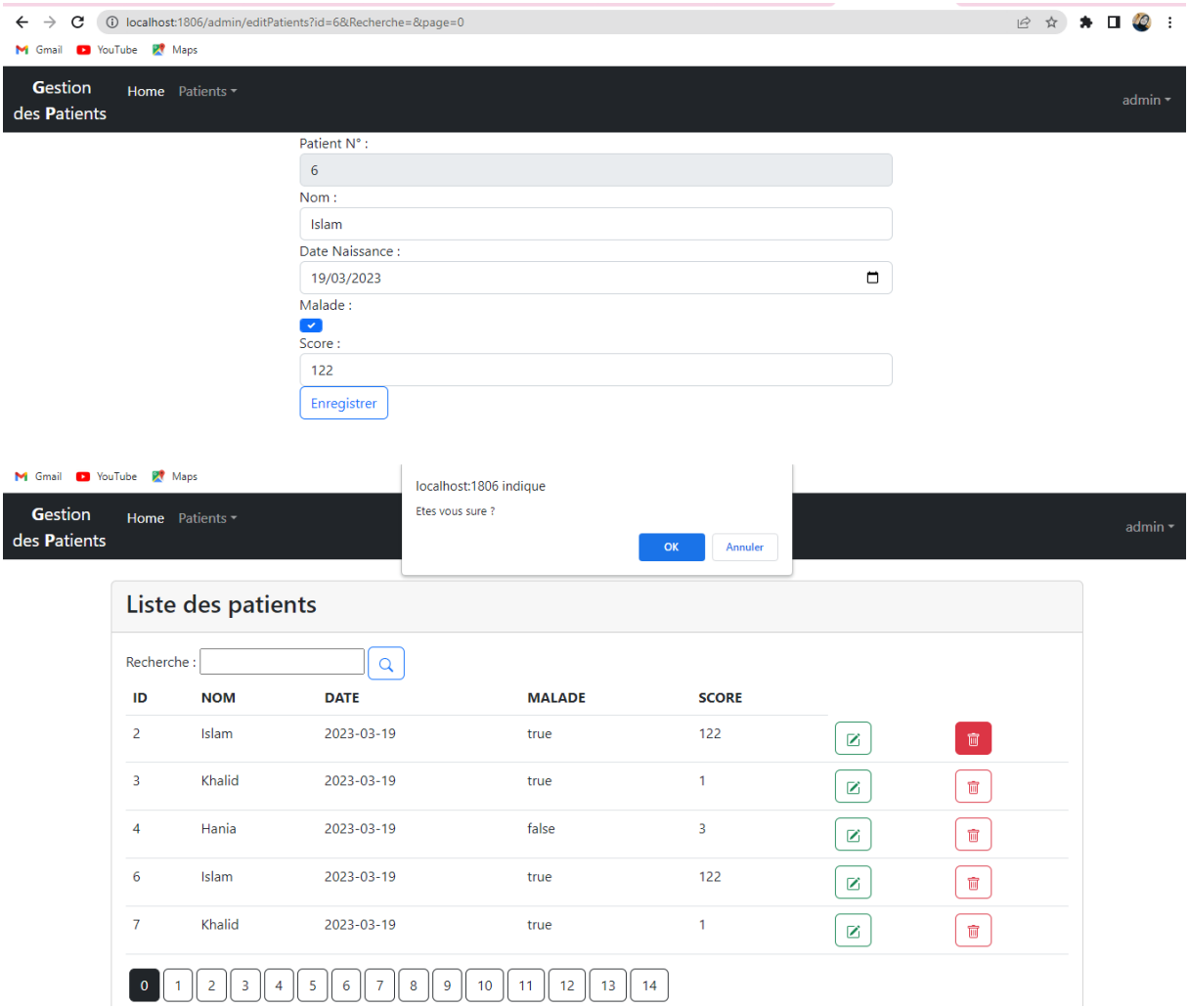
- Entant qu'utilisateur il peut consulter et chercher les patients :





- Entant qu'admin il a les droits des utilisateurs aussi il a la possibilité d'ajouter, Modifier et supprimer les patients





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.