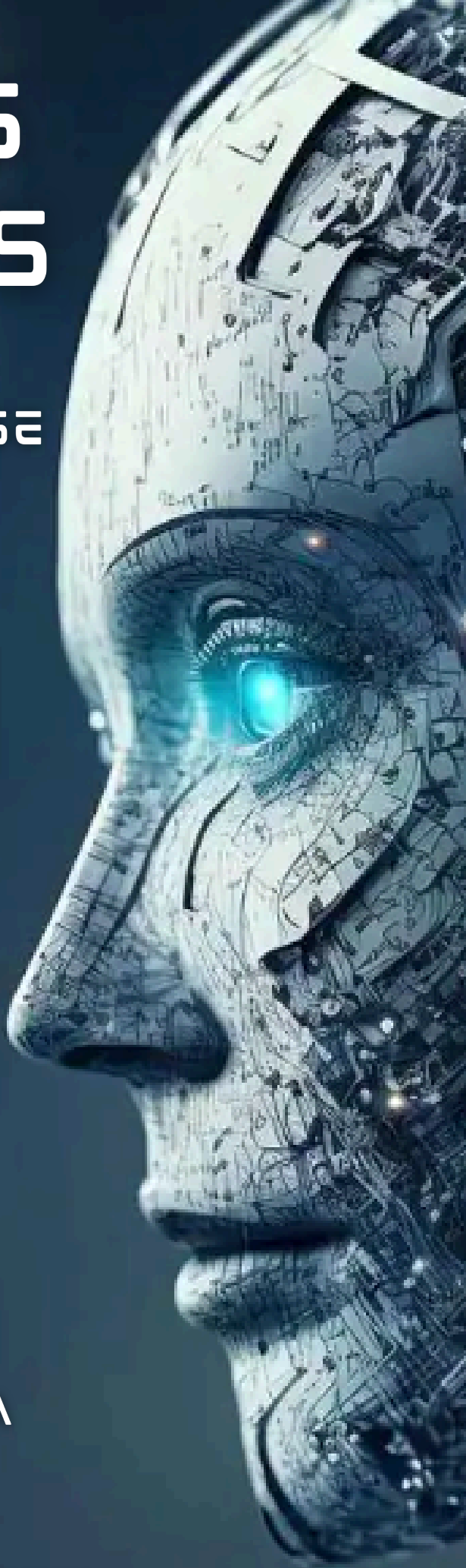
SYSTÈMES DISTRIBUÉS

MASTER INTELLIGENCE ARTIFICIELLE ET ANALYSE DES DONNÉES



RÉALISÉE PAR :

ANOADA NOHAYLA

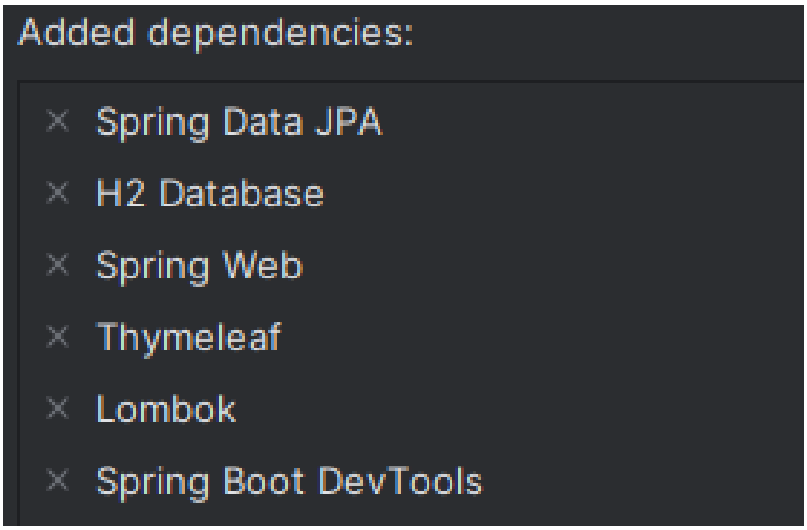
ENCADRÉ PAR :

M. MOHAMED YOUSSFI

ACTIVITÉ PRATIQUE N°3
SPRING MVC SPRING DATA JPA THYMELEAF

PARTIE 1 :

1. CRÉER UNE APPLICATION WEB JEE BASÉE SUR SPRING MVC, THYLEMEAF ET SPRING DATA JPA



2. CLASSE ENTITÉ (PATIENT)

```java
@Entity
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class Patient
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private Date dateNaissance;
    private boolean malade;
    private int score;
}
```

3. INTERFACE REPOSITORY (PATIENTREPOSITORY)

```java
public interface PatientRepository extends JpaRepository<Patient,Long>
{
    1 usage
    Page<Patient> findByNomContains(String keyword, Pageable pageable);
    no usages
    @Query("select p from Patient p where p.nom like:x")
    Page<Patient> chercher(@Param("x") String keyword, Pageable pageable);
}
```

4. MAIN APPLICATION CLASS

```java
@SpringBootApplication
public class AnoadaNohaylaTp31Application implements CommandLineRunner
{
    @Autowired
    private PatientRepository patientRepository;

    public static void main(String[] args)
    {
        SpringApplication.run(AnoadaNohaylaTp31Application.class, args);
    }

    @Override
    public void run(String... args) throws Exception
    {
        patientRepository.save(new Patient( id: null, nom: "Mohamed",new Date(), malade: false, score: 34));
        patientRepository.save(new Patient( id: null, nom: "Hanane",new Date(), malade: false, score: 4321));
        patientRepository.save(new Patient( id: null, nom: "Imane",new Date(), malade: true, score: 34));
    }
}
```

5. CLASSE CONTRÔLEUR

```java
@Controller
@AllArgsConstructor
public class PatientRestController
{
    @Autowired
    private PatientRepository patientRepository;

    @GetMapping("/index")
    public String index(Model model,
                        @RequestParam(name = "page",defaultValue = "0") int p,
                        @RequestParam(name = "size",defaultValue = "4")  int s,
                        @RequestParam(name = "keyword",defaultValue = "")  String kw)
    {
        Page<Patient> pagePatients=patientRepository.findByNomContains(kw,PageRequest.of(p,s));
        model.addAttribute( attributeName: "ListPatients",pagePatients.getContent());
        model.addAttribute( attributeName: "pages",new int[pagePatients.getTotalPages()]);
        model.addAttribute( attributeName: "currentPage",p);
        model.addAttribute( attributeName: "keyword",kw);
```

```
        model.addAttribute( attributeName: "currentPage",p);
        model.addAttribute( attributeName: "keyword",kw);
        return "patients";
    }
    @GetMapping("/delete")
    public String index(@RequestParam(name = "id" ) Long id,
                        @RequestParam(name = "page",defaultValue = "0") int page,
                        @RequestParam(name = "keyword",defaultValue = "")  String keyword)
    {
        patientRepository.deleteById(id);
        return "redirect:/index?page="+page+"&keyword="+keyword;
    }

    @GetMapping("/")
    public String index()
    {
        return "redirect:/index";
    }
}
```

6. MODÈLE HTML/THYMELEAF

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Patients</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.3.3/css/bootstrap.min.css">
    <script src="/webjars/jquery/3.7.1/jquery.min.js"></script>
    <link rel="stylesheet" href="/webjars/bootstrap-icons/1.11.3/font/bootstrap-icons.css">
</head>
<body>

    <div class="p-3">
        <div class="card">
            <div class="card-header">Liste patients</div>
            <div class="card-body">
                <form method="get" th:action="@{index}">
                    <label>Keywword:</label>
                    <input class="form-control" type="text" name="keyword" th:value="${keyword}">
                    <button type="submit" class="btn btn-info">
                        <i class="bi bi-search"></i>
                    </button>
                </form>
                <table class="table">
                    <thead>
                        <tr>
                            <th>ID</th> <th>Nom</th> <th>Date</th> <th>Malade</th> <th>Score</th> <th>Action</th>
                        </tr>
                        <tr th:each="p:${ListPatients}">
                            <td th:text="${p.id}"></td>
                            <td th:text="${p.nom}"></td>
                            <td th:text="${p.dateNaissance}"></td>
                            <td th:text="${p.malade}"></td>
                            <td th:text="${p.score}"></td>
                            <td>
                                <a onclick="javascript:return confirm('Etes vous sure?')" th:href="@{delete(id=${p.id},
                                keyword=${keyword}, page=${currentPage})}" class="btn btn-danger">
                                    <i class="bi bi-trash"></i>
                                </a>
                            </td>
                        </tr>
```

```
                </thead>
            </table>
            <ul class="nav nav-pills">
                <li th:each="value,item:${pages}">
                    <a th:href="@{/index(page=${item.index})}"
                       th:class="${(currentPage==item.index)?'btn btn-info ms-1':'btn btn-outline-info ms-1'}"
                       th:text="${item.index}"></a>
                </li>
            </ul>
        </div>
    </div>
</div>
</body>
```

Liste patients

Keywword:

ID	Nom	Date	Malade	Score	Action
1	Mohamed	2024-04-19 01:44:36.0	false	34	
2	Hanane	2024-04-19 01:44:38.0	false	4321	
3	Imane	2024-04-19 01:44:38.0	true	34	
4	Mohamed	2024-04-19 01:48:45.0	false	34	

0 1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24 25 26
27 28 29 30 31 32 33 34

7. FICHIER DE PROPRIÉTÉS

```
spring.application.name=Anoada_Nohayla_Tp3_1
server.port=8084
spring.datasource.url=jdbc:mysql://localhost:3306/patients_db?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
```

PARTIE 2 :

1. ENTITY CLASS (PATIENT)

LES ANNOTATIONS UTILISÉES AIDENT À DÉFINIR LA STRUCTURE DE LA TABLE DANS LA BASE DE DONNÉES ET À VALIDER LES DONNÉES.

```java
@Entity
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class Patient
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotEmpty
    @Size(min=4,max=48)
    private String nom;
    @Temporal(TemporalType.DATE)
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date dateNaissance;
    private boolean malade;
    @DecimalMin("100")
    private int score;
}
```

2. CONTRÔLEUR (PATIENTRESTCONTROLLER)

CE CONTRÔLEUR GÈRE LES REQUÊTES HTTP LIÉES AUX PATIENTS. IL CONTIENT DES MÉTHODES POUR AFFICHER LA LISTE DES PATIENTS, AJOUTER DE NOUVEAUX PATIENTS, MODIFIER DES PATIENTS EXISTANTS, SUPPRIMER DES PATIENTS. LES ANNOTATIONS COMME @GETMAPPING ET @POSTMAPPING SONT UTILISÉES POUR MAPPER LES REQUÊTES HTTP AUX MÉTHODES CORRESPONDANTES.

```java
@Controller
@AllArgsConstructor
public class PatientRestController
{
    @Autowired
    private PatientRepository patientRepository;

    @GetMapping("/index")
    public String index(Model model,
                        @RequestParam(name = "page",defaultValue = "0") int p,
                        @RequestParam(name = "size",defaultValue = "4")  int s,
                        @RequestParam(name = "keyword",defaultValue = "")  String kw)
    {
        Page<Patient> pagePatients=patientRepository.findByNomContains(kw,PageRequest.of(p,s));
        model.addAttribute( attributeName: "ListPatients",pagePatients.getContent());
        model.addAttribute( attributeName: "pages",new int[pagePatients.getTotalPages()]);
        model.addAttribute( attributeName: "currentPage",p);
        model.addAttribute( attributeName: "keyword",kw);
        return "patients";
    }
    @GetMapping("/delete")
    public String index(@RequestParam(name = "id" ) Long id,
                        @RequestParam(name = "page",defaultValue = "0") int page,
                        @RequestParam(name = "keyword",defaultValue = "")  String keyword)
    {
        patientRepository.deleteById(id);
        return "redirect:/index?page="+page+"&keyword="+keyword;
    }

    @GetMapping("/")
    public String index() { return "redirect:/index"; }

    @GetMapping("/patients")
    @ResponseBody
    public List<Patient> listPatients() { return patientRepository.findAll(); }

    @GetMapping("/formPatients")
    public String formPatients(Model model)
    {
        model.addAttribute( attributeName: "patient",new Patient());
        return "formPatients";
    }

    @PostMapping(path="/save")
    public String save(Model model, @Valid Patient patient,
                       BindingResult bindingResult,
                       @RequestParam(defaultValue = "0") int page,
                       @RequestParam(defaultValue = "")  String keyword)
    {
        if(bindingResult.hasErrors()) return "formPatients";
        patientRepository.save(patient);
        return "redirect:/index?page="+page+"&keyword="+keyword;
    }

    @GetMapping("/editPatient")
    public String editPatient(Model model,Long id ,int page,String keyword)
    {
        Patient patient=patientRepository.findById(id).orElse( other: null);
        if(patient==null) throw new RuntimeException("patient introuvable");
        model.addAttribute( attributeName: "patient",patient);
        model.addAttribute( attributeName: "page",page);
        model.addAttribute( attributeName: "keyword",keyword);
        return "editPatient";
    }
}
```

3. Modèles HTML

- template1.html

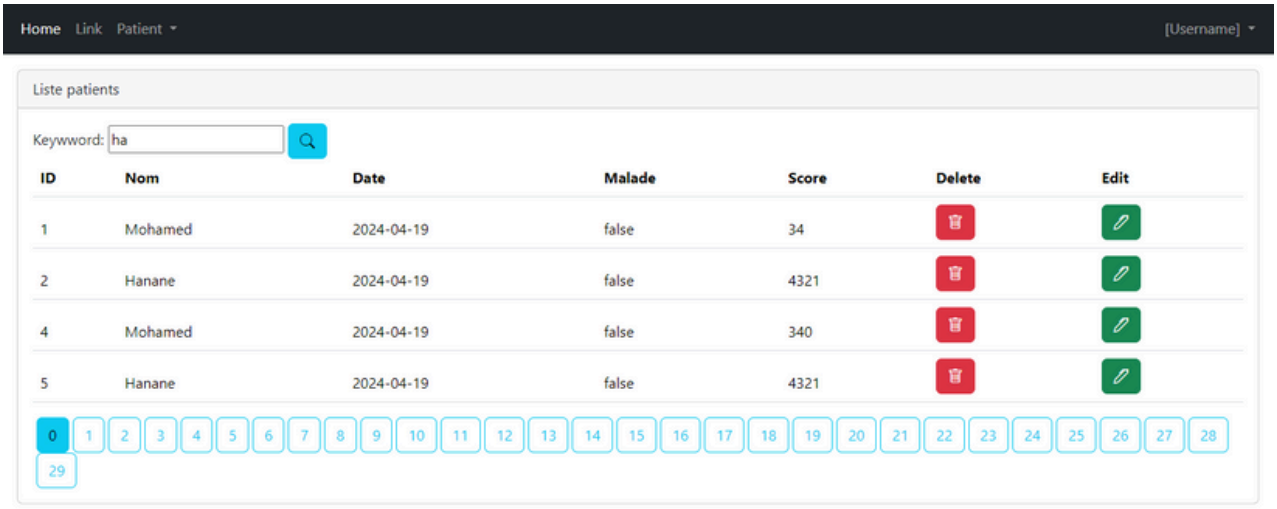
Ce modèle sert de fondement commun pour les autres pages, en incluant une barre de navigation et une mise en page standardisée.

```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>Patients</title>
    <link rel="stylesheet" type="text/css" href="/webjars/bootstrap/5.3.3/css/bootstrap.min.css">
    <script src="/webjars/bootstrap/5.3.3/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="nav-link active" th:href="@{index}">Home</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">Link</a>
                </li>
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">
                        Patient
                    </a>
                    <ul class="dropdown-menu">
                        <li><a class="dropdown-item" th:href="@{/formPatients}">Nouveau</a></li>
                        <li><a class="dropdown-item" th:href="@{index}">Chercher</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">
                        [Username]
                    </a>
                    <ul class="dropdown-menu">
                        <li><a class="dropdown-item" th:href="@{/formPatients}">Logout</a></li>
                    </ul>
                </li>
            </ul>
        </div>
    </nav>
    <section layout:fragment="content1">
    </section>
</body>
</html>
```

- PATIENTS.HTML

CETTE PAGE AFFICHE LA LISTE DES PATIENTS AVEC DES FONCTIONNALITÉS DE PAGINATION ET DE RECHERCHE.

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="template1">
<head>
    <meta charset="UTF-8">
    <title>Patients</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.3.3/css/bootstrap.min.css">
    <script src="/webjars/jquery/3.7.1/jquery.min.js"></script>
    <link rel="stylesheet" href="/webjars/bootstrap-icons/1.11.3/font/bootstrap-icons.css">
</head>
<body>
    <div layout:fragment="content1">
        <div class="p-3">
            <div class="card">
                <div class="card-header">Liste patients</div>
                <div class="card-body">
                    <form method="get" th:action="@{index}">
                        <label>Keywword:</label>
                        <input type="text" name="keyword" th:value="${keyword}">
                        <button type="submit" class="btn btn-info">
                            <i class="bi bi-search"></i>
                        </button>
                    </form>
                    <table class="table">
                        <thead>
                            <tr>
                                <th>ID</th> <th>Nom</th> <th>Date</th> <th>Malade</th>
                                <th>Score</th> <th>Delete</th><th>Edit</th>
                            </tr>
                            <tr th:each="p:${ListPatients}">
                                <td th:text="${p.id}"></td>
                                <td th:text="${p.nom}"></td>
                                <td th:text="${p.dateNaissance}"></td>
                                <td th:text="${p.malade}"></td>
                                <td th:text="${p.score}"></td>
                                <td>
                                    <a onclick="javascript:return confirm('Etes vous sure?')"
                                       th:href="@{delete(id=${p.id},keyword=${keyword},
                                       page=${currentPage})}" class="btn btn-danger">
                                        <i class="bi bi-trash"></i>
                                    </a>
                                </td>
                                <td>
                                    <a th:href="@{editPatient(id=${p.id},
                                    keyword=${keyword}, page=${currentPage})}" class="btn btn-success">
                                        <i class="bi bi-pen"></i>
                                    </a>
                                </td>
                            </tr>
                        </thead>
                    </table>
                    <ul class="nav nav-pills">
                        <li th:each="value,item:${pages}">
                            <a th:href="@{/index(page=${item.index})}"
                               th:class="${(currentPage==item.index)?'btn btn-info ms-1':'btn btn-outline-info ms-1'}"
                               th:text="${item.index}"></a>
                        </li>
                    </ul>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```



- FORMPATIENTS.HTML

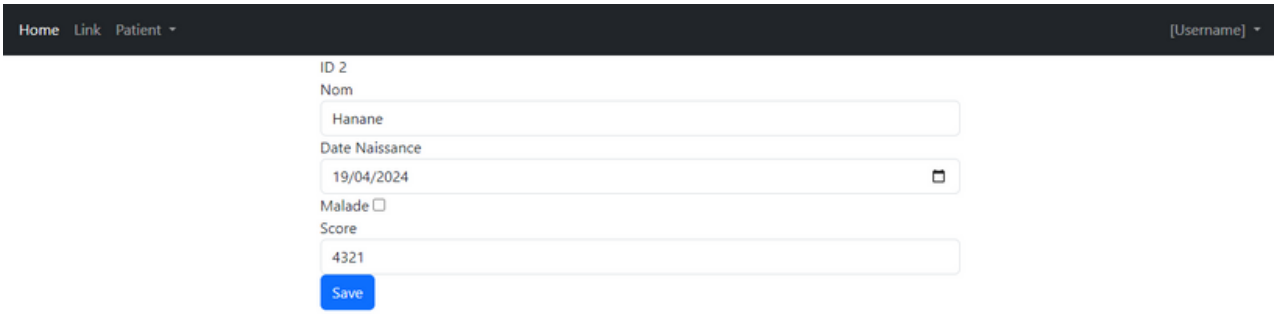
  CE FORMULAIRE PERMET D'AJOUTER DE NOUVEAUX PATIENTS OU DE MODIFIER DES PATIENTS EXISTANTS.

```html
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="template1">
<head>
    <meta charset="UTF-8">
    <title>Patients</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.3.3/css/bootstrap.min.css">
    <script src="/webjars/jquery/3.7.1/jquery.min.js"></script>
    <link rel="stylesheet" href="/webjars/bootstrap-icons/1.11.3/font/bootstrap-icons.css">
</head>
<body>
    <div layout:fragment="content1">
        <div class="col-md-6 offset-3">
            <form method="post" th:action="@{save}">
                <div>
                    <label for="nom">Nom</label>
                    <input id="nom" class="form-control" type="text" name="nom" th:value="${patient.nom}">
                    <span class="text-danger" th:errors="${patient.nom}"></span>
                </div>
                <div>
                    <label>Date Naissance</label>
                    <input class="form-control" type="date" name="dateNaissance" th:value="${patient.dateNaissance}">
                    <span class="text-danger" th:errors="${patient.dateNaissance}"></span>
                </div>
                <div>
                    <label>Malade</label>
                    <input type="checkbox" name="malade" th:checked="${patient.malade}">
                    <span class="text-danger" th:errors="${patient.malade}"></span>
                </div>
                <div>
                    <label>Score</label>
                    <input class="form-control" type="text" name="score" th:value="${patient.score}">
                    <span class="text-danger" th:errors="${patient.score}"></span>
                </div>
                <button type="submit" class="btn btn-primary">Save</button>
            </form>
        </div>
    </div>
</body>
</html>
```

- EDITPATIENT.HTML

  CE FORMULAIRE EST UTILISÉ POUR MODIFIER LES DÉTAILS D'UN PATIENT EXISTANT.

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="template1">
<head>
    <meta charset="UTF-8">
    <title>Patients</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.3.3/css/bootstrap.min.css">
    <script src="/webjars/jquery/3.7.1/jquery.min.js"></script>
    <link rel="stylesheet" href="/webjars/bootstrap-icons/1.11.3/font/bootstrap-icons.css">
</head>
<body>
    <div layout:fragment="content1">
        <div class="col-md-6 offset-3">
            <form method="post" th:action="@{save(page=${page},keyword=${keyword})}">
                <div>
                    <label for="id">ID</label>
                    <label th:text="${patient.id}">ID</label>
                    <input id="id" class="form-control" type="hidden" name="id" th:value="${patient.id}">
                </div>
                <div>
                    <label for="nom">Nom</label>
                    <input id="nom" class="form-control" type="text" name="nom" th:value="${patient.nom}">
                    <span class="text-danger" th:errors="${patient.nom}"></span>
                </div>
                <div>
                    <label>Date Naissance</label>
                    <input class="form-control" type="date" name="dateNaissance" th:value="${patient.dateNaissance}">
                    <span class="text-danger" th:errors="${patient.dateNaissance}"></span>
                </div>
                <div>
                    <label>Malade</label>
                    <input type="checkbox" name="malade" th:checked="${patient.malade}">
                    <span class="text-danger" th:errors="${patient.malade}"></span>
                </div>
                <div>
                    <label>Score</label>
                    <input class="form-control" type="text" name="score" th:value="${patient.score}">
                    <span class="text-danger" th:errors="${patient.score}"></span>
                </div>
                <button type="submit" class="btn btn-primary">Save</button>
            </form>
        </div>
    </div>
</body>
</html>
```



PARTIE 3 :

INMEMOMY AUTHENTICATION

1. CONFIGURATION DE LA SÉCURITÉ AVEC SPRING SECURITY (SECURITYCONFIG)

- CETTE CLASSE CONFIGURE LA SÉCURITÉ DE L'APPLICATION EN UTILISANT SPRING SECURITY.
- ELLE GÈRE L'AUTHENTIFICATION DES UTILISATEURS ET L'AUTORISATION D'ACCÈS AUX DIFFÉRENTES PARTIES DE L'APPLICATION EN FONCTION DES RÔLES.
- LA CONFIGURATION INCLUT LA PAGE DE LOGIN, LES AUTORISATIONS POUR LES URLS, ET LA GESTION DES SESSIONS.

```java
@Configuration
@EnableWebSecurity
@EnableMethodSecurity(prePostEnabled = true)
public class SecurityConfig
{
    @Autowired
    private PasswordEncoder passwordEncoder;
    nohayla *
    @Bean
    public InMemoryUserDetailsManager inMemoryUserDetailsManager()
    {
        return new InMemoryUserDetailsManager(
                User.withUsername("nohayla").password(passwordEncoder.encode( rawPassword: "1234"))
                        .roles("USER").build(),
                User.withUsername("user2").password(passwordEncoder.encode( rawPassword: "1234"))
                        .roles("USER").build(),
                User.withUsername("admin").password(passwordEncoder.encode( rawPassword: "1234"))
                        .roles("USER","ADMIN").build()
        );
    }
```

```
    nohayla
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception
    {
        httpSecurity.formLogin().loginPage("/login").permitAll();
        httpSecurity.authorizeRequests().requestMatchers("/webjars/**").permitAll();
        httpSecurity.rememberMe();
//          httpSecurity.authorizeRequests().requestMatchers("/user/**").hasRole("USER");
//          httpSecurity.authorizeRequests().requestMatchers("/admin/**").hasRole("ADMIN");
        httpSecurity.authorizeRequests().anyRequest().authenticated();
        httpSecurity.exceptionHandling().accessDeniedPage( accessDeniedUrl: "/notAuthorized");
        return httpSecurity.build();
    }
}
```

2. CONTRÔLEURS (PATIENTRESTCONTROLLER ET SECURITYCONTROLLER)

- PATIENTRESTCONTROLLER

CE CONTRÔLEUR GÈRE LES REQUÊTES LIÉES AUX PATIENTS COMME L'AFFICHAGE, L'AJOUT, LA MODIFICATION ET LA SUPPRESSION.

```
@Controller
@AllArgsConstructor
public class PatientRestController
{
    @Autowired
    private PatientRepository patientRepository;
    nohayla
    @GetMapping("/user/index")
    public String index(Model model,
                        @RequestParam(name = "page",defaultValue = "0") int p,
                        @RequestParam(name = "size",defaultValue = "4")  int s,
                        @RequestParam(name = "keyword",defaultValue = "")  String kw)
    {
        Page<Patient> pagePatients=patientRepository.findByNomContains(kw,PageRequest.of(p,s));
        model.addAttribute( attributeName: "ListPatients",pagePatients.getContent());
        model.addAttribute( attributeName: "pages",new int[pagePatients.getTotalPages()]);
        model.addAttribute( attributeName: "currentPage",p);
        model.addAttribute( attributeName: "keyword",kw);
        return "patients";
    }
    nohayla
    @GetMapping("/admin/delete")
    @PreAuthorize("hasRole('ROLE_ADMIN')")
    public String index(@RequestParam(name = "id" ) Long id,
                        @RequestParam(name = "page",defaultValue = "0") int page,
                        @RequestParam(name = "keyword",defaultValue = "")  String keyword)
    {
        patientRepository.deleteById(id);
        return "redirect:/user/index?page="+page+"&keyword="+keyword;
    }
    nohayla
    @GetMapping("/")
    public String index() { return "redirect:/user/index"; }
```

```java
    @GetMapping("/user/patients")
    @ResponseBody
    public List<Patient> listPatients() { return patientRepository.findAll(); }
    nohayla
    @GetMapping("/admin/formPatients")
    @PreAuthorize("hasRole('ROLE_ADMIN')")
    public String formPatients(Model model)
    {
        model.addAttribute( attributeName: "patient",new Patient());
        return "formPatients";
    }
    nohayla
    @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 keyword)
    {
        if(bindingResult.hasErrors()) return "formPatients";
        patientRepository.save(patient);
        return "redirect:/user/index?page="+page+"&keyword="+keyword;
    }
    nohayla
    @GetMapping("/admin/editPatient")
    @PreAuthorize("hasRole('ROLE_ADMIN')")
    public String editPatient(Model model,Long id ,int page,String keyword)
    {
        Patient patient=patientRepository.findById(id).orElse( other: null);
        if(patient==null) throw new RuntimeException("patient introuvable");
        model.addAttribute( attributeName: "patient",patient);
        model.addAttribute( attributeName: "page",page);
        model.addAttribute( attributeName: "keyword",keyword);
        return "editPatient";
    }
}
```

- SECURITYCONTROLLER

CE CONTRÔLEUR GÈRE LES PAGES DE LOGIN ET D'ACCÈS NON AUTORISÉ.

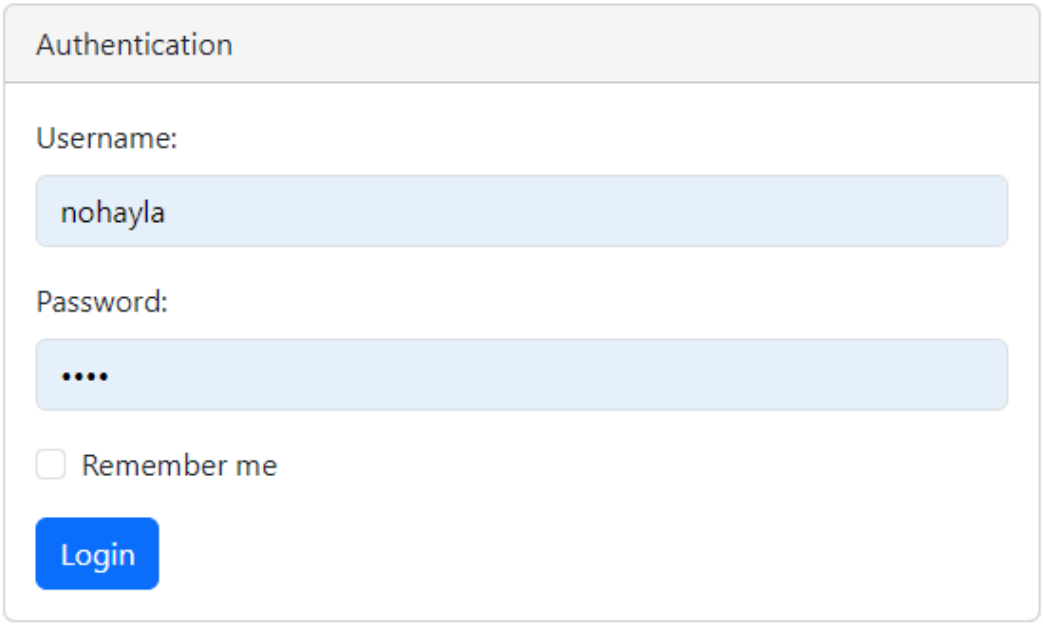
```java
@Controller
public class SecurityController
{
    nohayla
    @GetMapping("/notAuthorized")
    public String notAuthorized()
    {
        return "notAuthorized";
    }

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

3. VUES THYMELEAF

- LOGIN.HTML

UNE VUE POUR L'AUTHENTIFICATION DES UTILISATEURS.

```html
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.3.3/css/bootstrap.min.css">
    <script src="/webjars/jquery/3.7.1/jquery.min.js"></script>
    <link rel="stylesheet" href="/webjars/bootstrap-icons/1.11.3/font/bootstrap-icons.css">
</head>
<body>
    <div class="row mt-3">
        <div class="row t-3">
            <div class="col-ad-ó offset-3">
                <div class="card">
                    <div class="card-header">Authentication</div>
                    <div class="card-body">
                        <form method="post" th:action="@{/login}">
                            <div class="mb-3 at-3"...>
                            <div class="mb-3 at-3"...>
                            <div class="form-check mb-3">
                                <div class="form-check-label">
                                    <input type="checkbox" class="form-check-input" name="remember-me">
                                    Remember me
                                </div>
                            </div>
                            <button type="submit" class="btn btn-primary">Login</button>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```



- template1.html

Un template de base utilisé pour d'autres pages avec une barre de navigation commune.

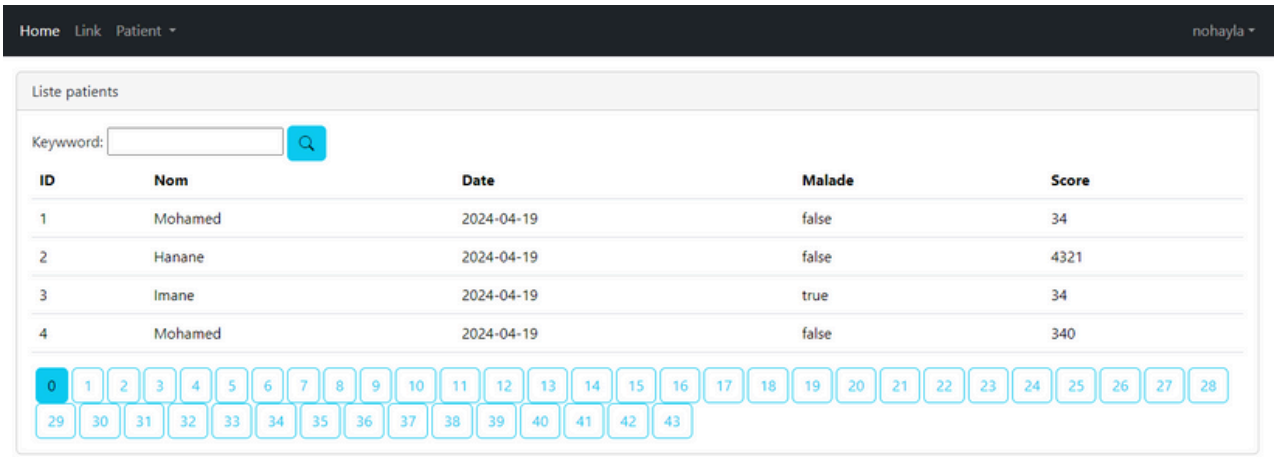
```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
    <meta charset="utf-8">
    <title>Patients</title>
    <link rel="stylesheet" type="text/css" href="/webjars/bootstrap/5.3.3/css/bootstrap.min.css">
    <script src="/webjars/bootstrap/5.3.3/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="nav-link active" th:href="@{/user/index}">Home</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">Link</a>
                </li>
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">
                        Patient
                    </a>
                    <ul class="dropdown-menu">
                        <li><a class="dropdown-item" th:if="${#authorization.expression('hasRole(''ADMIN'')')}"
                               th:href="@{/admin/formPatients}">Nouveau</a></li>
                        <li><a class="dropdown-item" th:href="@{/user/index}">Chercher</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">Logout</button>
                            </form>
                        </li>
                    </ul>
                </li>
            </ul>
        </div>
    </nav>
    <section layout:fragment="content1">
    </section>
</body>
</html>
```

- PATIENTS.HTML

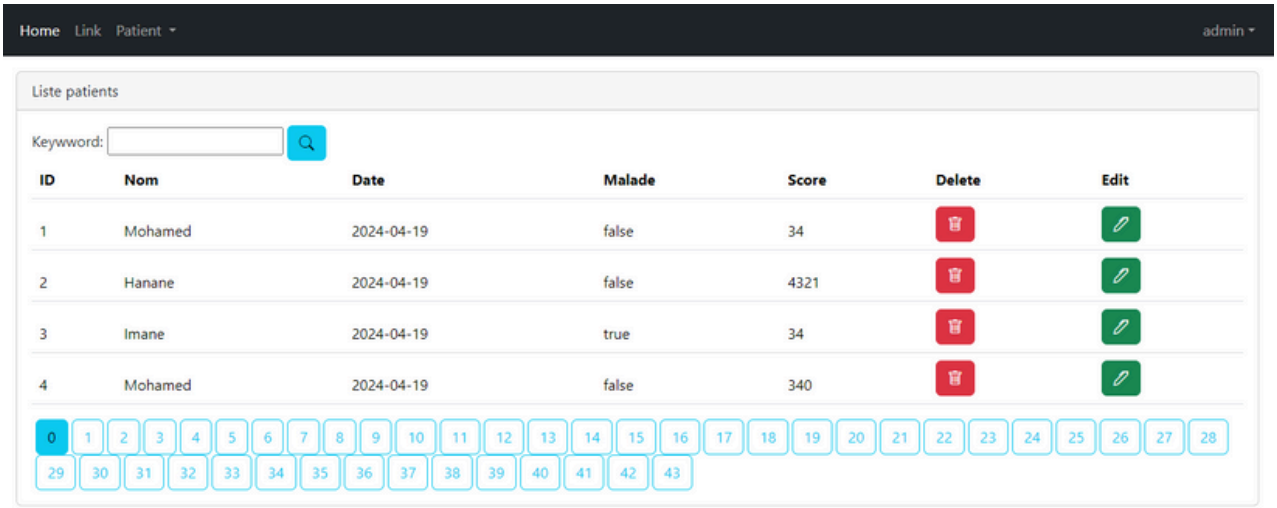
UNE VUE POUR AFFICHER LA LISTE DES PATIENTS AVEC DES FONCTIONNALITÉS DE RECHERCHE, D'ÉDITION ET DE SUPPRESSION.

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="template1">
<head>
    <meta charset="UTF-8">
    <title>Patients</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.3.3/css/bootstrap.min.css">
    <script src="/webjars/jquery/3.7.1/jquery.min.js"></script>
    <link rel="stylesheet" href="/webjars/bootstrap-icons/1.11.3/font/bootstrap-icons.css">
</head>
<body>
    <div layout:fragment="content1">
        <div class="p-3">
            <div class="card">
                <div class="card-header">Liste patients</div>
                <div class="card-body">
                    <form method="get" th:action="@{/user/index}">
                        <label>Keywword:</label>
                        <input type="text" name="keyword" th:value="${keyword}">
                        <button type="submit" class="btn btn-info">
                            <i class="bi bi-search"></i>
                        </button>
                    </form>
                    <table class="table">
                        <thead>
                            <tr>
                                <th>ID</th> <th>Nom</th> <th>Date</th> <th>Malade</th><th>Score</th>
                                <th th:if="${#authorization.expression('hasRole(''ADMIN'')')}">Delete</th>
                                <th th:if="${#authorization.expression('hasRole(''ADMIN'')')}">Edit</th>
                            </tr>
                            <tr th:each="p:${ListPatients}">
                                <td th:text="${p.id}"></td>
                                <td th:text="${p.nom}"></td>
                                <td th:text="${p.dateNaissance}"></td>
                                <td th:text="${p.malade}"></td>
                                <td th:text="${p.score}"></td>
                                <td th:if="${#authorization.expression('hasRole(''ADMIN'')')}">
                                    <a onclick="javascript:return confirm('Etes vous sure?')"
                                       th:href="@{/admin/delete(id=${p.id},keyword=${keyword},
                                       page=${currentPage})}" class="btn btn-danger">
                                        <i class="bi bi-trash"></i>
                                    </a>
                                </td>
                                <td th:if="${#authorization.expression('hasRole(''ADMIN'')')}">
                                    <a th:href="@{/admin/editPatient(id=${p.id},
                                    keyword=${keyword}, page=${currentPage})}" class="btn btn-success">
                                        <i class="bi bi-pen"></i>
                                    </a>
                                </td>
                            </tr>
                        </thead>
                    </table>
                    <ul class="nav nav-pills">
                        <li th:each="value,item:${pages}">
                            <a th:href="@{/user/index(page=${item.index})}"
                               th:class="${(currentPage==item.index)?'btn btn-info ms-1':'btn btn-outline-info ms-1'}"
                               th:text="${item.index}"></a>
                        </li>
                    </ul>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

SI ROLE EST UN UTILISATEUR



SI ROLE EST UN ADMIN



- FORMPATIENTS.HTML

UNE VUE POUR AJOUTER LES INFORMATIONS D'UN PATIENT AVEC DES FONCTIONNALITÉS DE VALIDATION.

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="template1">
<head>
    <meta charset="UTF-8">
    <title>Patients</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.3.3/css/bootstrap.min.css">
    <script src="/webjars/jquery/3.7.1/jquery.min.js"></script>
    <link rel="stylesheet" href="/webjars/bootstrap-icons/1.11.3/font/bootstrap-icons.css">
</head>
<body>
    <div layout:fragment="content1">
        <div class="col-md-6 offset-3">
            <form method="post" th:action="@{/admin//admin/save}">
                <div>
                    <label for="nom">Nom</label>
```

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

- EDITPATIENTS.HTML

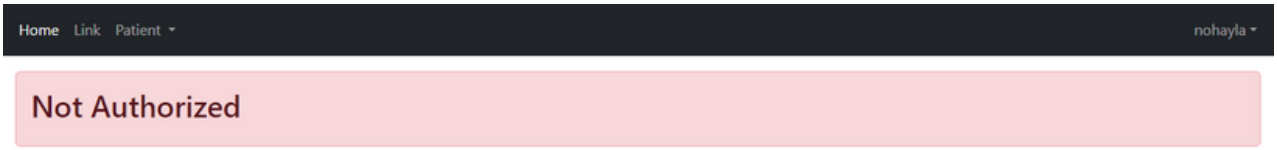
UNE VUE POUR MODIFIER LES INFORMATIONS D'UN PATIENT AVEC DES FONCTIONNALITÉS DE VALIDATION.

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="template1">
<head>
    <meta charset="UTF-8">
    <title>Patients</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.3.3/css/bootstrap.min.css">
    <script src="/webjars/jquery/3.7.1/jquery.min.js"></script>
    <link rel="stylesheet" href="/webjars/bootstrap-icons/1.11.3/font/bootstrap-icons.css">
</head>
<body>
    <div layout:fragment="content1">
        <div class="col-md-6 offset-3">
            <form method="post" th:action="@{/admin/save(page=${page},keyword=${keyword})}">
                <div>
                    <label for="id">ID</label>
                    <label th:text="${patient.id}">ID</label>
                    <input id="id" class="form-control" type="hidden" name="id" th:value="${patient.id}">
                </div>
                <div>
                    <label for="nom">Nom</label>
                    <input id="nom" class="form-control" type="text" name="nom" th:value="${patient.nom}">
                    <span class="text-danger" th:errors="${patient.nom}"></span>
```

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

- NOTAUTHORIZED.HTML

  UNE VUE AFFICHÉE LORSQU'UN UTILISATEUR TENTE D'ACCÉDER À UNE PAGE NON AUTORISÉE.



4. CLASSE PRINCIPALE DE L'APPLICATION SPRING BOOT (ANOADANOHAYLATP31APPLICATION)

- CETTE CLASSE EST LE POINT D'ENTRÉE DE L'APPLICATION SPRING BOOT.

- Elle initialise l'application, configure la base de données et définit un bean pour le cryptage des mots de passe.

```java
@SpringBootApplication
public class AnoadaNohaylaTp31Application implements CommandLineRunner
{
    @Autowired
    private PatientRepository patientRepository;
    nohayla
    public static void main(String[] args) { SpringApplication.run(AnoadaNohaylaTp31Application.class, args); }
    nohayla
    @Override
    public void run(String... args) throws Exception
    {
        /*
        patientRepository.save(new Patient(null,"Mohamed",new Date(),false,34));
        patientRepository.save(new Patient(null,"Hanane",new Date(),false,4321));
        patientRepository.save(new Patient(null,"Imane",new Date(),true,34));
        */
    }
    nohayla
    @Bean
    PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }
}
```

JDBC Authentication

1. application.properties

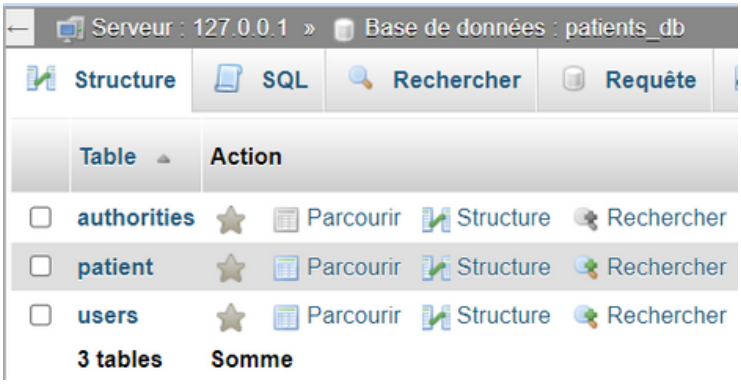
Ces paramètres sont essentiels pour garantir le bon fonctionnement de l'application Spring Boot avec la base de données MySQL spécifiée.

```properties
spring.application.name=Anoada_Nohayla_Tp3_1
server.port=8084
spring.datasource.url=jdbc:mysql://localhost:3306/patients_db?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=none
spring.jpa.defer-datasource-initialization=true
spring.sql.init.mode=always
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
```

2. SQL INITIALIZATION

DES SCRIPTS SQL SONT FOURNIS POUR CRÉER LES TABLES NÉCESSAIRES (UTILISATEURS, AUTORITÉS) ET INSÉRER LES DONNÉES INITIALES DES UTILISATEURS.

```sql
create table if not exists users(username varchar(50) not null primary key,
    password varchar(500) not null,enabled boolean not null);
create table if not exists authorities (username varchar(50) not null,authority varchar(50) not null,
    constraint fk_authorities_users foreign key(username) references users(username));
create unique index if not exists ix_auth_username on authorities (username,authority);
```

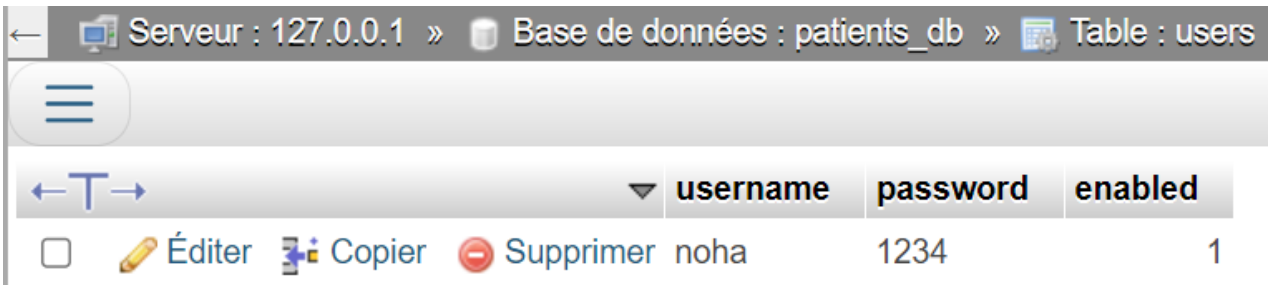


3. INSTRUCTIONS SQL

- UN FICHIER SQL

UTILISÉ POUR EXÉCUTER DES SCRIPTS SQL DIRECTEMENT SUR LA BASE DE DONNÉES

```sql
INSERT INTO `users` (`username`, `password`, `enabled`)
    VALUES ('noha', '1234', '1');
```



- CommandLineRunner

Ce code utilise un CommandLineRunner pour créer des utilisateurs au démarrage de l'application Spring Boot. Chaque utilisateur est vérifié pour voir s'il existe déjà, et s'il n'existe pas, il est créé avec un rôle spécifique. C'est une approche pratique pour initialiser des données utilisateur au lancement de l'application.

```java
@Bean
CommandLineRunner commandLineRunner(JdbcUserDetailsManager jdbcUserDetailsManager, PasswordEncoder passwordEncoder)
{
    return args -> {
        if (!jdbcUserDetailsManager.userExists( username: "user11")) {
            jdbcUserDetailsManager.createUser(User.withUsername("user11")
                    .password(passwordEncoder.encode( rawPassword: "1234"))
                    .roles("USER").build());
        }

        if (!jdbcUserDetailsManager.userExists( username: "user22")) {
            jdbcUserDetailsManager.createUser(User.withUsername("user22")
                    .password(passwordEncoder.encode( rawPassword: "1234"))
                    .roles("USER").build());
        }

        if (!jdbcUserDetailsManager.userExists( username: "admin2")) {
            jdbcUserDetailsManager.createUser(User.withUsername("admin2")
                    .password(passwordEncoder.encode( rawPassword: "1234"))
                    .roles("USER", "ADMIN").build());
        }
    };
}
```

Serveur : 127.0.0.1 » Base de données : patients_db » Table : users

	username	password	enabled
Éditer Copier Supprimer	admin2	$2a$10$67cHYId0eeIHnRdeIZxvouCjDZat8oIOS5Ww3zpCQ3U...	1
Éditer Copier Supprimer	noha	1234	1
Éditer Copier Supprimer	user11	$2a$10$nvSWvSh18MD0k9PpHbUEgefssuxg/pRjuYxyFGJyAt3...	1
Éditer Copier Supprimer	user22	$2a$10$nMYEhHrR3YenxB18pKrwguZKDR/7ujf.r24E9sXu/r5...	1

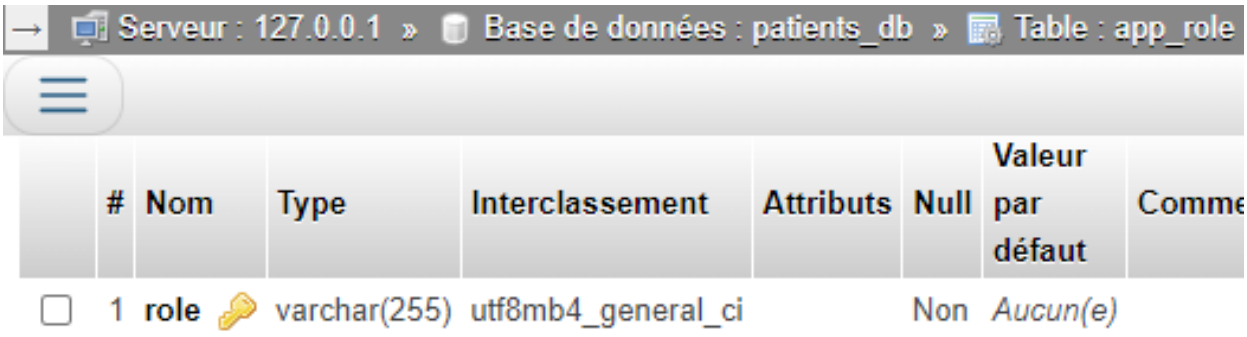
UserDetails Service

1. Entités de sécurité

- AppRole

Cette entité représente un rôle dans le système. Elle est annotée avec @Entity et contient un attribut role annoté avec @Id qui représente le nom du rôle.

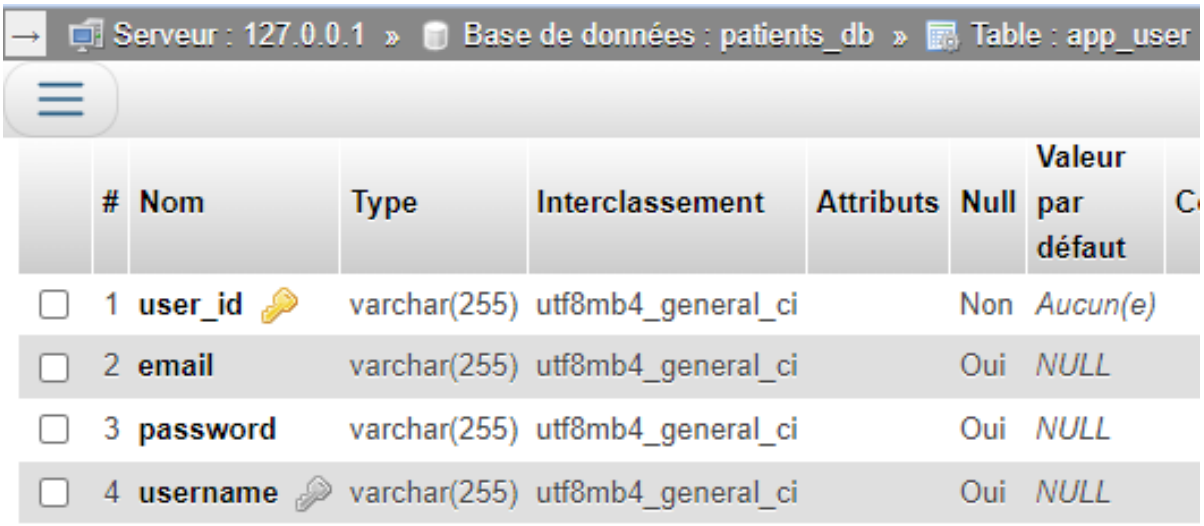
```java
@Entity @Data
@NoArgsConstructor @AllArgsConstructor @Builder
public class AppRole
{
    @Id
    private String role;
}
```



- AppUser

Cette entité représente un utilisateur dans le système. Elle est annotée avec @Entity et contient des attributs tels que userId, username, password, email, et une liste de rôles (roles). L'attribut username est annoté avec @Column(unique = true) pour garantir l'unicité des noms d'utilisateur.

```java
@Entity
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class AppUser
{
    @Id
    private String userId;
    @Column(unique = true)
    private String username;
    private String password;
    private String email;
    @ManyToMany(fetch = FetchType.EAGER)
    private List<AppRole> roles;
}
```



	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	C
☐	1	user_id	varchar(255)	utf8mb4_general_ci		Non	*Aucun(e)*	
☐	2	email	varchar(255)	utf8mb4_general_ci		Oui	*NULL*	
☐	3	password	varchar(255)	utf8mb4_general_ci		Oui	*NULL*	
☐	4	username	varchar(255)	utf8mb4_general_ci		Oui	*NULL*	

2. Répertoires de données

- AppRoleRepository

Interface qui étend JpaRepository pour la gestion des entités AppRole

```
public interface AppRoleRepository extends JpaRepository<AppRole,String>
{
}
```

- AppUserRepository

Interface qui étend JpaRepository pour la gestion des entités AppUser. Elle comprend également une méthode findByUsername pour rechercher un utilisateur par nom d'utilisateur.

```
public interface AppUserRepository extends JpaRepository<AppUser,String>
{
    4 usages   nohayla
    AppUser findByUsername(String username);
}
```

3. Services de sécurité

- AccountService

Interface définissant les opérations liées aux utilisateurs (AppUser) et aux rôles (AppRole), telles que l'ajout d'un nouvel utilisateur, d'un nouveau rôle, l'attribution d'un rôle à un utilisateur...

```java
public interface AccountService
{
    3 usages  1 implementation  nohayla
    AppUser addNewUser(String username,String password,String email, String confirmPassword);
    2 usages  1 implementation  nohayla
    AppRole addNewRole(String role);
    4 usages  1 implementation  nohayla
    void addRoleToUser(String username, String role);
    no usages  1 implementation  nohayla
    void removeRoleFromUser(String username, String role);
    1 usage  1 implementation  nohayla
    AppUser loadUserByUsername(String username);
}
```

- AccountServiceImpl

Implémentation de AccountService. Elle contient les méthodes concrètes pour ajouter un nouvel utilisateur, un nouveau rôle, attribuer un rôle à un utilisateur... Les opérations sont effectuées en utilisant les répertoires (AppUserRepository et AppRoleRepository) et le PasswordEncoder.

```java
@Service
@Transactional
@AllArgsConstructor
public class AccountServiceImpl implements AccountService
{
    private AppUserRepository appUserRepository;
    private AppRoleRepository appRoleRepository;
    private PasswordEncoder passwordEncoder;

    3 usages  nohayla
    @Override
    public AppUser addNewUser(String username, String password, String email, String confirmPassword)
    {
        AppUser appUser=appUserRepository.findByUsername(username);
        if(appUser!=null) throw new RuntimeException("this user already exist");
        if(!password.equals(confirmPassword)) throw new RuntimeException("Passwords not match");
        AppUser user=AppUser.builder()
                .userId(UUID.randomUUID().toString())
                .username (username)
                .password(passwordEncoder.encode(password))
                .email(email)
                .build();
        return appUserRepository.save(appUser);
    }
```

```java
    @Override
    public AppRole addNewRole(String role)
    {
        AppRole appRole = appRoleRepository.findById(role).orElse( other: null);
        if(appRole!=null) throw new RuntimeException("This role already exist");
        appRole = AppRole.builder()
            .role(role)
            .build();
        return appRoleRepository.save(appRole);
    }

    4 usages  nohayla
    @Override
    public void addRoleToUser(String username, String role)
    {
        AppUser appUser = appUserRepository.findByUsername(username);
        AppRole appRole = appRoleRepository.findById(role).get();
        appUser.getRoles().add(appRole);
        //appUserRepository.save(appUser);
    }

    no usages  nohayla
    @Override
    public void removeRoleFromUser(String username, String role)
    {
        AppUser appUser = appUserRepository.findByUsername(username);
        AppRole appRole = appRoleRepository.findById(role).get();
        appUser.getRoles().add(appRole);
    no usages  nohayla
    @Override
    public void removeRoleFromUser(String username, String role)
    {
        AppUser appUser = appUserRepository.findByUsername(username);
        AppRole appRole = appRoleRepository.findById(role).get();
        appUser.getRoles().add(appRole);
    }

    1 usage  nohayla
    @Override
    public AppUser loadUserByUsername(String username)
    {
        return appUserRepository.findByUsername(username);
    }
}
```

4. Services d'authentification et d'autorisation

- UserDetailServiceImpl

Implémentation de UserDetailsService pour charger les détails de l'utilisateur à partir du service AccountService. Elle utilise le PasswordEncoder pour vérifier les mots de passe et crée des UserDetails à partir des entités AppUser pour l'authentification.

```java
@Service
@AllArgsConstructor
public class UserDetailServiceImpl implements UserDetailsService
{
    private AccountService accountService;
    no usages  nohayla *
    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException
    {
        AppUser appUser = accountService.loadUserByUsername(username);
        if(appUser==null) throw new UsernameNotFoundException(String
                .format("User %s not found", username));
        String[] roles =appUser.getRoles().stream().map(u->u.getRole()).toArray(String[]::new);
        UserDetails userDetails= User
                .withUsername (appUser.getUsername())
                .password(appUser.getPassword())
                .roles(roles).build();
        return userDetails;
    }
}
```

5. Configuration de sécurité

- SecurityConfig

Configuration de la sécurité de l'application. Elle est annotée avec @Configuration, @EnableWebSecurity, et @EnableMethodSecurity(prePostEnabled = true) pour activer la sécurité au niveau des méthodes. La classe configure les règles de sécurité, la gestion des utilisateurs en mémoire ou via JDBC, les filtres de sécurité,...

```java
@Configuration
@EnableWebSecurity
@EnableMethodSecurity(prePostEnabled = true)
@AllArgsConstructor
public class SecurityConfig
{
    private PasswordEncoder passwordEncoder;
    private UserDetailServiceImpl userDetailServiceImpl;
    //@Bean
    no usages  nohayla
    public JdbcUserDetailsManager jdbcUserDetailsManager(DataSource dataSource)
    {
        return new JdbcUserDetailsManager(dataSource);
    }
    //@Bean
    no usages  nohayla
    public InMemoryUserDetailsManager inMemoryUserDetailsManager()
    {
        return new InMemoryUserDetailsManager(
                User.withUsername("nohayla").password(passwordEncoder.encode( rawPassword: "1234"))
                        .roles("USER").build(),
                User.withUsername("user2").password(passwordEncoder.encode( rawPassword: "1234"))
                        .roles("USER").build(),
                User.withUsername("admin").password(passwordEncoder.encode( rawPassword: "1234"))
                        .roles("USER","ADMIN").build()
        );
    }
    nohayla
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception
    {
        httpSecurity.formLogin().loginPage("/login").defaultSuccessUrl("/").permitAll();
        httpSecurity.authorizeRequests().requestMatchers("/webjars/**").permitAll();
        httpSecurity.rememberMe();
//          httpSecurity.authorizeRequests().requestMatchers("/user/**").hasRole("USER");
//          httpSecurity.authorizeRequests().requestMatchers("/admin/**").hasRole("ADMIN");
        httpSecurity.authorizeRequests().anyRequest().authenticated();
        httpSecurity.exceptionHandling().accessDeniedPage( accessDeniedUrl: "/notAuthorized");
        httpSecurity.userDetailsService(userDetailServiceImpl);
        return httpSecurity.build();
    }
}
```

6. Classe principale de l'application Spring Boot (AnoadaNohaylaTp31Application)

Gérer la sécurité des utilisateurs, les rôles, et la gestion des données au démarrage de l'application.

```java
@SpringBootApplication
public class AnoadaNohaylaTp31Application implements CommandLineRunner
{
    @Autowired
    private PatientRepository patientRepository;
    nohayla
    public static void main(String[] args) { SpringApplication.run(AnoadaNohaylaTp31Application.class, args); }
    nohayla
    @Override
    public void run(String... args) throws Exception
    {
        /*
        patientRepository.save(new Patient(null,"Mohamed",new Date(),false,34));
        patientRepository.save(new Patient(null,"Hanane",new Date(),false,4321));
        patientRepository.save(new Patient(null,"Imane",new Date(),true,34));
        */
    }
    //@Bean
    no usages  nohayla
    CommandLineRunner commandLineRunner(JdbcUserDetailsManager jdbcUserDetailsManager, PasswordEncoder passwordEncoder)
    {
        return args -> {
            if (!jdbcUserDetailsManager.userExists( username: "user11")) {
                jdbcUserDetailsManager.createUser(User.withUsername("user11")
                        .password(passwordEncoder.encode( rawPassword: "1234"))
                        .roles("USER").build());
            }

            if (!jdbcUserDetailsManager.userExists( username: "user22")) {
                jdbcUserDetailsManager.createUser(User.withUsername("user22")
                        .password(passwordEncoder.encode( rawPassword: "1234"))
                        .roles("USER").build());
            }

            if (!jdbcUserDetailsManager.userExists( username: "admin2")) {
                jdbcUserDetailsManager.createUser(User.withUsername("admin2")
                        .password(passwordEncoder.encode( rawPassword: "1234"))
                        .roles("USER", "ADMIN").build());
            }
        };
    }
    //@Bean
    no usages  nohayla *
    CommandLineRunner commandLineRunnerUserDetails(AccountService accountService)
    {
       return args->{
         accountService.addNewRole("USER");
         accountService.addNewRole("ADMIN");

         accountService.addNewUser( username: "user10",  password: "1234",  email: "user1@gmail.com",  confirmPassword: "1234");
         accountService.addNewUser( username: "user20",  password: "1234",  email: "user2@gmail.com",  confirmPassword: "1234");
         accountService.addNewUser( username: "admin10",  password: "1234",  email: "admin@gmail.com",  confirmPassword: "1234");

         accountService.addRoleToUser( username: "user10",  role: "USER");
         accountService.addRoleToUser( username: "user20",  role: "USER");
         accountService.addRoleToUser( username: "admin10",  role: "USER");
         accountService.addRoleToUser( username: "admin10",  role: "ADMIN");
      };
    }
    nohayla
    @Bean
    PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }
}
```

7. application.properties

Ces paramètres sont essentiels pour garantir le bon fonctionnement de l'application Spring Boot avec la base de données MySQL spécifiée.

```
spring.application.name=Anoada_Nohayla_Tp3_1
server.port=8084
spring.datasource.url=jdbc:mysql://localhost:3306/patients_db?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.defer-datasource-initialization=true
spring.sql.init.mode=always
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
```