

Département d'Informatique

Filiere : IAAD

A ,U :2023-2024



Université Moulay Ismail faculté des Sciences Meknès

Département d'Informatique

TPN°3 : Spring MVC Spring Data JPA
Thymeleaf
Module : Systèmes distribués

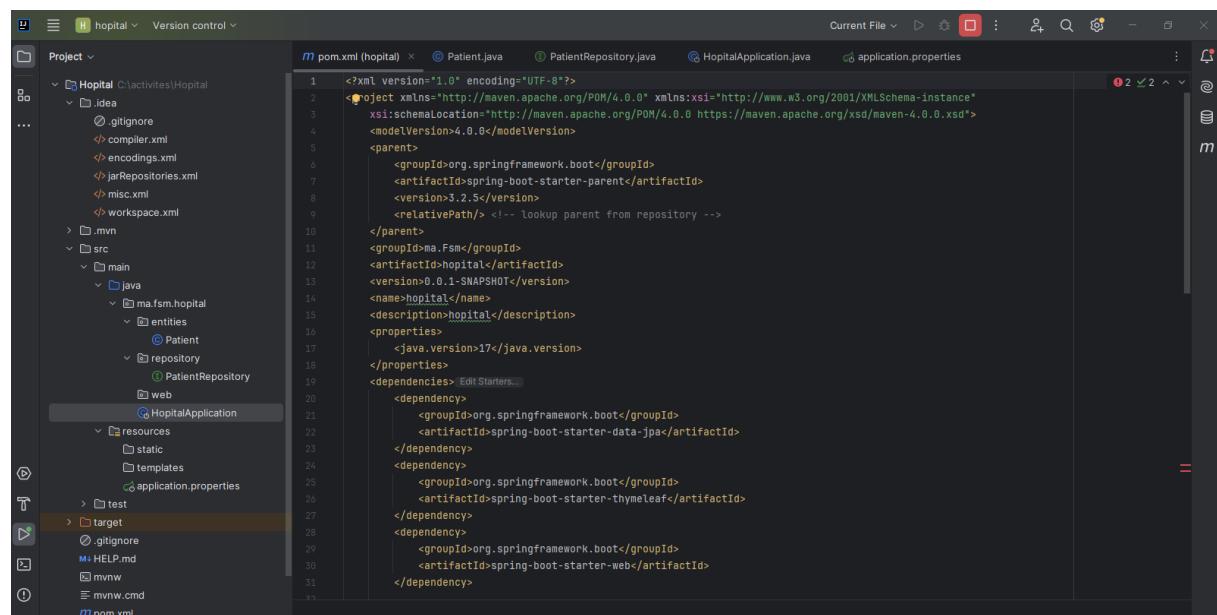
Réalisé par :

- *AIT BOUCHALA fatime ezzahrae*

⇒ Partie 1 :

Créer une application Web JEE basée sur Spring MVC, Thymeleaf et Spring Data JPA qui permet de gérer les patients. L'application doit permettre les fonctionnalités suivantes :

- Afficher les patients
- Faire la pagination
- Chercher les patients
- Supprimer un patient
- Faire des améliorations supplémentaires



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "Project". It includes a .idea folder, .gitignore, compiler.xml, encodings.xml, jarRepositories.xml, misc.xml, workspace.xml, .mvn, src, main, java, ma.fsm.hopital, entities, Patient, repository, PatientRepository, HopitalApplication, resources, static, templates, and application.properties.
- Pom.xml Content:** The pom.xml file is open in the editor. It defines a Maven project with version 1.0, encoding UTF-8. It uses the Spring Boot Starter Parent artifact with version 3.2.5. It has a parent dependency on org.springframework.boot with version 4.0.0. It specifies a group ID of ma.Fsm and an artifact ID of hopital, with a version of 0.0.1-SNAPSHOT. The name is set to hopital and the description to hopital. It includes properties for Java version 17 and a dependency on org.springframework.boot:spring-boot-starter-data-jpa. It also includes dependencies for org.springframework.boot:spring-boot-starter-thymeleaf and org.springframework.boot:spring-boot-starter-web.

```

1 package ma.fsm.hopital.entities;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.GeneratedValue;
5 import jakarta.persistence.GenerationType;
6 import jakarta.persistence.Id;
7 import lombok.NoArgsConstructor;
8 import lombok.AllArgsConstructor;
9 import lombok.Builder;
10 import lombok.Data;
11 import lombok.NoArgsConstructor;
12
13 import java.util.Date;
14
15 @Entity // Jpa
16 @Data
17 @AllArgsConstructor
18 @NoArgsConstructor
19 @Builder
20 @Id
21 @GeneratedValue(strategy = GenerationType.IDENTITY)
22 private Long id;
23 private String nom;
24 private Date dateNaissance;
25 private boolean malade;
26 private int score;
}

```

```

1 package ma.fsm.hopital.repository;
2
3 import ma.fsm.hopital.entities.Patient;
4 import org.springframework.data.domain.Page;
5 import org.springframework.data.domain.Pageable;
6 import org.springframework.data.jpa.repository.JpaRepository;
7 import org.springframework.data.jpa.repository.Query;
8 import org.springframework.data.repository.query.Param;
9
10 public interface PatientRepository extends JpaRepository<Patient, Long> {
11
12     no usages
13     Page<Patient> findByNomContains(String keyword, Pageable pageable);
14
15     no usages
16     @Query("select p from Patient p where p.nom like :x")
17     Page<Patient> chercher(@Param("x") String keyword, Pageable pageable);
}

```

Pour se connecter au base de données h2.

```

1 spring.application.name=hospital
2
3 server.port=8084
4
5 spring.datasource.url=jdbc:h2:mem:patients-db
6
7
8
9
|spring.h2.console.enabled=true

```

On exécute.

```

21     SpringApplication.run(HopitalApplication.class, args);
22 }
23
24 @Override
25 public void run(String... args) throws Exception {
26
27     /*Patient patient = new Patient();
28     patient.setId(null);
29     patient.setNom("Mohamed");
30     patient.setDateNaissance(new Date());
31     patient.setMalade(false);
32     patient.setScore(23);
33
34     Patient patient2 = new Patient(null, "Yassine", new Date(), false, 123);
35
36     Patient patient3 = Patient.builder()
37         .nom("Imane")
38         .dateNaissance(new Date())
39         .score(30)
40         .malade(true)
41         .build();*/
42
43     patientRepository.save(new Patient(null, "Mohamed", new Date(), false, 34));
44     patientRepository.save(new Patient(null, "Hanane", new Date(), false, 4521));
45     patientRepository.save(new Patient(null, "Imane", new Date(), true, 34));
46 }
47

```

```

:: Spring Boot :: (v3.2.5)

2024-04-21T23:21:51.542+01:00 INFO 6292 --- [hospital] [ restartedMain] ma.fsm.hopital.HopitalApplication : Starting HopitalApplication using Java 17.0.1 with PID 6292 (C:\act...
2024-04-21T23:21:51.712+01:00 INFO 6292 --- [hospital] [ restartedMain] ma.fsm.hopital.HopitalApplication : No active profile set, falling back to 1 default profile: "default"
2024-04-21T23:21:52.445+01:00 INFO 6292 --- [hospital] [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-propert...
2024-04-21T23:21:52.446+01:00 INFO 6292 --- [hospital] [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level...
2024-04-21T23:21:58.931+01:00 INFO 6292 --- [hospital] [ restartedMain] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-04-21T23:21:59.425+01:00 INFO 6292 --- [hospital] [ restartedMain] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 400 ms. Found 1 JPA repo...
2024-04-21T23:22:07.262+01:00 INFO 6292 --- [hospital] [ restartedMain] o.s.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8084 (http)
2024-04-21T23:22:07.650+01:00 INFO 6292 --- [hospital] [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-04-21T23:22:07.651+01:00 INFO 6292 --- [hospital] [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.20]
2024-04-21T23:22:08.163+01:00 INFO 6292 --- [hospital] [ restartedMain] o.a.c.c.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext
2024-04-21T23:22:08.164+01:00 INFO 6292 --- [hospital] [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 15715 ms
2024-04-21T23:22:08.473+01:00 INFO 6292 --- [hospital] [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-04-21T23:22:09.931+01:00 INFO 6292 --- [hospital] [ restartedMain] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:patients-db
2024-04-21T23:22:09.934+01:00 INFO 6292 --- [hospital] [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-04-21T23:22:09.964+01:00 INFO 6292 --- [hospital] [ restartedMain] o.s.b.a.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:...
2024-04-21T23:22:11.279+01:00 INFO 6292 --- [hospital] [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2024-04-21T23:22:12.076+01:00 INFO 6292 --- [hospital] [ restartedMain] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.4.4.Final
2024-04-21T23:22:12.425+01:00 INFO 6292 --- [hospital] [ restartedMain] o.h.t.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2024-04-21T23:22:13.910+01:00 INFO 6292 --- [hospital] [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeLeaver setup: ignoring JPA class transformer
2024-04-21T23:22:18.769+01:00 INFO 6292 --- [hospital] [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000469: No JTA platform available (set 'hibernate.transaction.jta.platform' to ...
2024-04-21T23:22:18.897+01:00 INFO 6292 --- [hospital] [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-04-21T23:22:19.946+01:00 INFO 6292 --- [hospital] [ restartedMain] o.s.d.j.n.query.QueryEnhancerFactory : Hibernate is in classpath; If applicable, HQL parser will be used.
2024-04-21T23:22:21.419+01:00 WARN 6292 --- [hospital] [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database c...
2024-04-21T23:22:22.558+01:00 WARN 6292 --- [hospital] [ restartedMain] ion$DefaultTemplateResolverConfiguration : Cannot find template location: classpath:/templates/ (please add som...
2024-04-21T23:22:22.972+01:00 INFO 6292 --- [hospital] [ restartedMain] o.s.b.a.OptionalLiveReloadServer : LiveReload server is running on port 55729
2024-04-21T23:22:23.070+01:00 INFO 6292 --- [hospital] [ restartedMain] o.s.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8084 (http) with context path ''
2024-04-21T23:22:23.091+01:00 INFO 6292 --- [hospital] [ restartedMain] ma.fsm.hopital.HopitalApplication : Started HopitalApplication in 34.718 seconds (process running for 40...

```

Important Commands

	Displays this Help Page
	Shows the Command History
	Ctrl+Enter Executes the current SQL statement
	Shift+Enter Executes the SQL statement defined by the text selection
	Ctrl+Space Auto complete
	Disconnects from the database

Sample SQL Script

```

Delete the table if it exists      DROP TABLE IF EXISTS TEST;
Create a new table      CREATE TABLE TEST(ID INT PRIMARY KEY,
with ID and NAME columns      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='Hi' WHERE ID=1;
  
```

- La partie web :

```

package ma.fsm.hopital.web;
import org.springframework.ui.Model;
import lombok.AllArgsConstructor;
import ma.fsm.hopital.entities.Patient;
import ma.fsm.hopital.repository.PatientRepository;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import java.util.List;

@Controller
@AllArgsConstructor
public class PatientController {
    private PatientRepository patientRepository;

    @GetMapping("/*index")
    public String index(Model model){
        List<Patient> patientList=patientRepository.findAll();
        model.addAttribute(attributeName: "ListPatients", patientList);
        return "patients";
    }
}
  
```

Si on exécute ce code :

```

import ...;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Autowired;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

@SpringBootApplication
public class HopitalApplication implements CommandLineRunner {
    @Autowired
    private PatientRepository patientRepository;

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

    @Override
    public void run(String... args) throws Exception {
        Patient patient = new Patient();
        patient.setId(null);
        patient.setNom("Mohamed");
        patient.setDateNaissance(new Date());
        patient.setMalade(false);
        patient.setScore(34);
        Patient patient2 = new Patient(null, "Yassine", new Date(), false, 123);
        Patient patients3 = Patient.builder()
                .nom("Imane")
                .dateNaissance(new Date())
                .score(36)
                .malade(true)
                .build();
        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));
    }
}

```

Le Fichier Html :

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Liste patients</title>
    <link rel="stylesheet" href="webjars/<th>"/>
</head>
<body>
    <h2>Liste patients</h2>
    <table>
        <thead>
            <tr>
                <th>ID</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="${p.nom}"></td>
                <td th:text="${p.dateNaissance}"></td>
                <td th:text="${p.malade}"></td>
                <td th:text="${p.score}"></td>
            </tr>
        </tbody>
    </table>
</body>
</html>

```

La page html affiche la table suivants :



Pour une meilleure apparence de notre liste des patients on va utiliser Bootstrap, donc on va ajouter les dépendances de Bootstrap dans le fichier pom.xml :

The screenshot shows a Java Spring Boot project structure in the left sidebar. The project name is 'hopital'. It includes packages for 'main' (containing 'Patient', 'repository', and 'PatientController') and 'web' (containing 'Patients.html' and 'application.properties'). The right side displays the content of the 'pom.xml' file, which defines dependencies for Spring Boot, Thymeleaf, and WebJars.

```

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>bootstrap</artifactId>
    <version>5.2.3</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>

```

On va utiliser Bootstrap dans notre fichier HTML :

The screenshot shows a Thymeleaf template ('Patients.html') with embedded CSS and Thymeleaf logic. It lists patients from a database and applies Bootstrap styles to the table rows.

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://thymeleaf.org">
    <head>
        <meta charset="UTF-8">
        <title>Title</title>
        <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css" th:href="@{/webjars/bootstrap/5.2.3/css/bootstrap.min.css}">
    </head>
    <body>
        <h2> Liste patients</h2>
        <table class="table">
            <thead>
                <tr>
                    <th>ID</th> <th>Date</th> <th>Malade</th> <th>Score</th>
                </tr>
            <tbody>
                <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>
                </tr>
            </tbody>
        </table>
    </body>
</html>

```

Après exécution on aura comme suivant :

The screenshot shows the resulting HTML output in a browser. The page title is 'localhost:8084/index'. The table header is 'Liste patients'. The data is as follows:

ID	Date	Malade	Score
1	Mohamed	2024-04-22 15:57:18.325	false 34
2	Hanane	2024-04-22 15:57:18.42	false 4321
3	Imane	2024-04-22 15:57:18.421	true 34

Nous avons effectué quelques changements dans le fichier html pour une meilleure apparence de notre liste.

The screenshot shows a Java development environment with a browser preview window and a code editor window.

Browser Preview:

- Title bar: Activité Pratique N°...
- Address bar: localhost:8084/index
- Content: A table titled "Liste Parents" with columns ID, Date, Malade, and Score. It contains three rows of data.

ID	Date	Malade	Score
1	2024-04-22 16:14:27.429	false	34
2	2024-04-22 16:14:27.527	false	4321
3	2024-04-22 16:14:27.528	true	34

Code Editor:

```

1 <!DOCTYPE html>
2 <html lang="en" xmlns:th="http://thymeleaf.org">
3     <head>
4         <meta charset="UTF-8">
5         <title>Title</title>
6         <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css"/>
7     </head>
8     <body>
9         <div class="p-3">
10            <div class="card">
11                <div class="card-header">Liste Parents</div>
12                <div class="card-footer">
13                    <table class="table">
14                        <thead>
15                            <tr>
16                                <th>ID</th> <th>Date</th> <th>Malade</th> <th>Score</th>
17                            </tr>
18                        <tr th:each="p:${listPatients}">
19                            <td th:text="${p.id}"></td>
20                            <td th:text="${p.nom}"></td>
21                            <td th:text="${p.dateNaissance}"></td>
22                            <td th:text="${p.malade}"></td>
23                            <td th:text="${p.score}"></td>
24                        </tr>
25                    </thead>
26                    <tbody>
27                        </tbody>
28                    </table>
29                </div>
30            </div>
31        </body>
32    </html>

```

Ainsi nous avons utilisé la dépendance devtools pour que le code exécute automatiquement après tout changement

- Pour aller de base de données h2 au MySQL on doit télécharger les dépendances MySQL, et on change la base de données dans le fichier application.properties :

The screenshot shows a Java development environment with a project structure and a code editor window.

Project Structure:

- Project: hospital
- src
 - main
 - java
 - ma.fsm.hopital
 - Patient
 - repository
 - PatientRepository
 - web
 - PatientController
 - HopitalApplication
 - resources
 - static
 - templates
 - Patients.html
 - test
 - target
 - .gitignore
 - HELP.md
 - mvnw
 - mvnw.cmd

Code Editor:

```

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<dependency>

```

```

application.properties
server.port=8084
spring.datasource.url=jdbc:h2:mem:patients-db
spring.datasource.enabled=true
spring.datasource.url=jdbc:mysql://localhost:3306/hopital?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
spring.jpa.show-sql=true

application.properties
package ma.fsm.hopital.web;
import org.springframework.ui.Model;
import lombok.AllArgsConstructor;
import ma.fsm.hopital.entities.Patient;
import ma.fsm.hopital.repository.PatientRepository;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import java.util.List;

@Controller
@AllArgsConstructor
public class PatientController {
    private PatientRepository patientRepository;

    @GetMapping(@PathVariable("index"))
    public String index(Model model){
        List<Patient> patientList=patientRepository.findAll();
        model.addAttribute("ListPatients",patientList);
        return "patients";
    }
}

PatientController.java
HopitalApplication.java
Patients.html
PatientRepository.java
Patient.java
pom.xml (hopital)
application.properties

```

Console output:

```

2024-04-22T20:37:06.119+01:00  WARN 1084 --- [hospital] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view=false if you don't want this.
2024-04-22T20:37:07.327+01:00  INFO 1084 --- [hospital] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 55729
2024-04-22T20:37:07.459+01:00  INFO 1084 --- [hospital] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8084 (http) with context path ''
2024-04-22T20:37:07.484+01:00  INFO 1084 --- [hospital] ma.fsm.hopital.HopitalApplication : Started HopitalApplication in 11.589 seconds (process running for 11.589)
Hibernate: insert into patient (date_naissance,malade,nom,score) values (?, ?, ?, ?)
Hibernate: insert into patient (date_naissance,malade,nom,score) values (?, ?, ?, ?)
Hibernate: insert into patient (date_naissance,malade,nom,score) values (?, ?, ?, ?)

```

Et on a le suivant dans le phpMyAdmin.

Nouvelle base de données: hopital

Table: patient

Affichage des lignes 0 - 2 (total de 3, traitement en 0,0024 seconde(s))

Opérations sur les résultats de la requête

	id	date_naissance	malade	nom	score
<input type="checkbox"/>	1	2024-04-22 20:37:07.000000	0	Mohamed	34
<input type="checkbox"/>	2	2024-04-22 20:37:08.000000	0	Hanane	4321
<input type="checkbox"/>	3	2024-04-22 20:37:08.000000	1	Imane	34

Alors quand on redémarre l'application, elle nous rajouter les données.

The screenshot shows the phpMyAdmin interface with the 'patient' table selected. The table has columns: id, date_naissance, malade, nom, and score. The data consists of 12 rows, each with a unique ID, birth date, status (malade), name, and score. The names listed are Mohamed, Hanane, and Imane.

	id	date_naissance	malade	nom	score
1	1	2024-04-22 20:37:07.000000	0	Mohamed	34
2	2	2024-04-22 20:37:08.000000	0	Hanane	4321
3	3	2024-04-22 20:37:08.000000	1	Imane	34
4	4	2024-04-22 20:41:55.000000	0	Mohamed	34
5	5	2024-04-22 20:41:55.000000	0	Hanane	4321
6	6	2024-04-22 20:41:55.000000	1	Imane	34
7	7	2024-04-22 20:42:08.000000	0	Mohamed	34
8	8	2024-04-22 20:42:09.000000	0	Hanane	4321
9	9	2024-04-22 20:42:09.000000	1	Imane	34
10	10	2024-04-22 21:04:43.000000	0	Mohamed	34
11	11	2024-04-22 21:04:43.000000	0	Hanane	4321
12	12	2024-04-22 21:04:43.000000	1	Imane	34

Et le vue se voit comme suivant :

The screenshot shows a web browser displaying a table titled 'Liste Parents'. The table has columns: ID, Date, Malade, and Score. The data is identical to the one shown in the phpMyAdmin table, with 12 rows for Mohamed, Hanane, and Imane.

ID	Date	Malade	Score
1	Mohamed	2024-04-22 20:37:07.0	false 34
2	Hanane	2024-04-22 20:37:08.0	false 4321
3	Imane	2024-04-22 20:37:08.0	true 34
4	Mohamed	2024-04-22 20:41:55.0	false 34
5	Hanane	2024-04-22 20:41:55.0	false 4321
6	Imane	2024-04-22 20:41:55.0	true 34
7	Mohamed	2024-04-22 20:42:08.0	false 34
8	Hanane	2024-04-22 20:42:08.0	false 4321
9	Imane	2024-04-22 20:42:08.0	true 34
10	Mohamed	2024-04-22 21:04:43.0	false 34
11	Hanane	2024-04-22 21:04:43.0	false 4321
12	Imane	2024-04-22 21:04:43.0	true 34

la pagination :

On spécifie dans l'url la page qu'on veut afficher et combien d'éléments .

The screenshot shows an IDE (IntelliJ IDEA) displaying the 'Patients.html' template file. The file contains HTML and JSP code. It includes a table with columns: ID, Date, Malade, and Score. The table is populated using a `<th:each>` loop. Below the table is a navigation bar with a `` element containing links for different pages.

```

<tbody>
    <tr>
        <th>ID</th> <th>Date</th> <th>Malade</th> <th>Score</th>
    </tr>
    <th:each="p:${ListPatients}">
        <td>${p.id}</td>
        <td>${p.nom}</td>
        <td>${p.dateNaissance}</td>
        <td>${p.malade}</td>
        <td>${p.score}</td>
    </th:each>
</tbody>

```

The screenshot shows a Java Spring Boot project structure in the left sidebar. The `PatientController.java` file is open in the main editor. The code defines a controller for patient management, using `@Controller` and `@ModelAttribute` annotations. It handles a GET request for the index page, which retrieves patients from the repository and adds them to a model. The `Patients.html` template is selected in the sidebar.

```

1 package ma.fsm.hopital.web;
2
3 import org.springframework.data.domain.PageRequest;
4 import org.springframework.ui.Model;
5 import lombok.AllArgsConstructor;
6 import ma.fsm.hopital.entities.Patient;
7 import ma.fsm.hopital.repository.PatientRepository;
8 import org.springframework.data.domain.Page;
9 import org.springframework.stereotype.Controller;
10 import org.springframework.web.bind.annotation.GetMapping;
11 import org.springframework.web.bind.annotation.RequestParam;
12 import java.util.List;
13
14 @Controller
15 @AllArgsConstructor
16 public class PatientController {
17     private PatientRepository patientRepository;
18
19     @GetMapping("/index")
20     public String index(Model model,
21                         @RequestParam(name="page", defaultValue = "0") int p,
22                         @RequestParam(name="size", defaultValue = "4") int s){
23         Page<Patient> pagePatients=patientRepository.findAll(PageRequest.of(p,s));
24         model.addAttribute("ListPatients",pagePatients.getContent());
25         model.addAttribute("pages",new int[pagePatients.getTotalPages()]);
26         return "patients";
27     }
28
29 }
30
31

```

On utilise les boutons pour se déplacer d'une page vers l'autre

Si en appuyer sur la page 3 elle affiche le suivant :

The screenshot shows a web browser window with the URL `localhost:8084/index?page=3`. The page displays a table titled "Liste Patients" containing four rows of patient data. Below the table is a navigation bar with numbered buttons from 0 to 9, where button 3 is highlighted.

ID	Date	Malade	Score	
13	Mohamed	2024-04-22 21:12:30.0	false	34
14	Hanane	2024-04-22 21:12:30.0	false	4321
15	Imane	2024-04-22 21:12:30.0	true	34
16	Mohamed	2024-04-22 21:14:32.0	false	34

0 1 2 3 4 5 6 7 8 9

Si en appuyer sur la page 7 elle affiche le suivant :

The screenshot shows a web browser window with the URL `localhost:8084/index?page=7` in the address bar. The page displays a table titled "Liste Patients" containing four rows of data. The columns are labeled "ID", "Date", "Malade", "Score", and an unnamed column. The data is as follows:

ID	Date	Malade	Score	
29	Hanane	2024-04-22 21:17:04.0	false	4321
30	Imane	2024-04-22 21:17:04.0	true	34
31	Mohamed	2024-04-22 21:32:40.0	false	34
32	Hanane	2024-04-22 21:32:40.0	false	4321

Below the table is a navigation bar with buttons labeled 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

Pour Colorer la page courante, nous avons ajouter les lignes de code suivants au fichier html et la classe PatientController :

Screenshot of a Java code editor showing PatientController.java:

```

6 import ma.fsm.hopital.entities.Patient;
7 import ma.fsm.hopital.repository.PatientRepository;
8 import org.springframework.data.domain.Page;
9 import org.springframework.stereotype.Controller;
10 import org.springframework.web.bind.annotation.GetMapping;
11 import org.springframework.web.bind.annotation.RequestParam;
12 import java.util.List;
13
14 @Controller
15     @AllArgsConstructor
16     public class PatientController {
17         private PatientRepository patientRepository;
18
19         @GetMapping("/index")
20         public String index(Model model,
21                             @RequestParam(name="page",defaultValue = "0") int p,
22                             @RequestParam(name="size",defaultValue = "4")int s){
23             Page<Patient> pagePatients=patientRepository.findAll(PageRequest.of(p,s));
24             model.addAttribute( attributeName: "ListPatients",pagePatients.getContent());
25             model.addAttribute( attributeName: "pages",new int[pagePatients.getTotalPages()]);
26             model.addAttribute( attributeName: "currentPage",p);
27             return "patients";
28         }
29     }
30
31
32

```

Screenshot of a browser window showing the Patients.html page at localhost:8084/index?page=9. The page displays a table of patients with columns: ID, Date, Malade, and Score. Below the table is a navigation bar with numbered buttons from 0 to 11.

ID	Date	Malade	Score
37	Mohamed	2024-04-22 22:02:07.0	false 34
38	Hanane	2024-04-22 22:02:07.0	false 4321
39	Imane	2024-04-22 22:02:07.0	true 34
40	Mohamed	2024-04-22 22:10:50.0	false 34

Navigation buttons: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

Screenshot of the Patients.html template file:

```

<thead>
<tr>
    <th>ID</th> <th>Date</th> <th>Malade</th> <th>Score</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>
</tr>
</thead>
</table>


- a href="#">
                    <th:text>${status.index}</th:text>
                </th:a>
            <th:endif>
        </a>
    </li>

```

Pour créer un formulaire dans notre vue :

```

4   <head>
5     <meta charset="UTF-8">
6     <title>Title</title>
7     <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
8   </head>
9   <body>
10  <div class="p-3">
11    <div class="card">
12      <div class="card-header">Liste Patients</div>
13      <div class="card-body">
14        <form method="get" th:action="@{index}">
15          <label>keyword:</label>
16          <input type="text" name="keyword">
17          <button type="submit" class="btn btn-info">chercher</button>
18        </form>
19      </div>
20      <div class="card-footer">
21        <table class="table">
22          <thead>
23            <tr>
24              <th>ID</th> <th>Date</th> <th>Malade</th> <th>Score</th>
25            </tr>
26          <tr th:each="p:${ListPatients}">
27            <td th:text="${p.id}"></td>
28            <td th:text="${p.nom}"></td>
29            <td th:text="${p.dateNaissance}"></td>
30            <td th:text="${p.malade}"></td>
31            <td th:text="${p.score}"></td>
32          </tr>
33        </thead>
34      </table>
35    </div>
36  </div>

```

ID	Date	Malade	Score
1	Mohamed	2024-04-22 20:37:07.0	false 34
2	Hanane	2024-04-22 20:37:08.0	false 4321
3	Imane	2024-04-22 20:37:08.0	true 34
4	Mohamed	2024-04-22 20:41:55.0	false 34

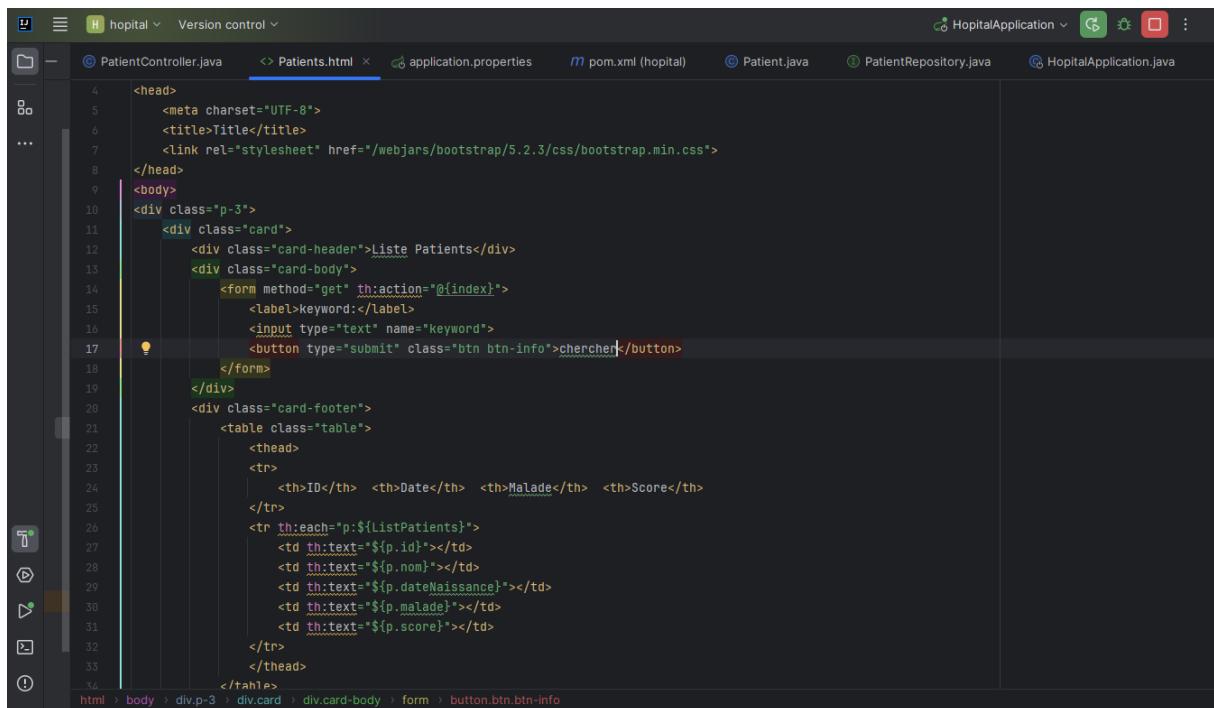
On fait un changement au niveau de PatientController :

```

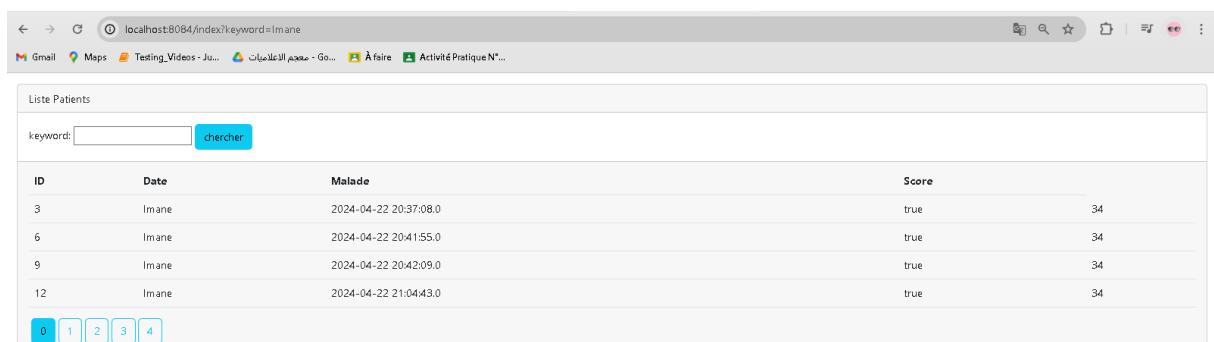
1 package ma.fsm.hopital.web;
2
3 import org.springframework.data.domain.PageRequest;
4 import org.springframework.ui.Model;
5 import lombok.AllArgsConstructor;
6 import ma.fsm.hopital.entities.Patient;
7 import ma.fsm.hopital.repository.PatientRepository;
8 import org.springframework.data.domain.Page;
9 import org.springframework.stereotype.Controller;
10 import org.springframework.web.bind.annotation.GetMapping;
11 import org.springframework.web.bind.annotation.RequestParam;
12 import java.util.List;
13
14 @Controller
15 @AllArgsConstructor
16 public class PatientController {
17     private PatientRepository patientRepository;
18
19     @GetMapping("^{}/index")
20     public String index(Model model,
21                         @RequestParam(name="page",defaultValue = "0") int p,
22                         @RequestParam(name="size",defaultValue = "4")int s,
23                         @RequestParam(name="keyword",defaultValue = "") String kw)
24     {
25         Page<Patient> pagePatients=patientRepository.findByNomContains(kw,PageRequest.of(p,s));
26         model.addAttribute( attributeName: "ListPatients",pagePatients.getContent());
27         model.addAttribute( attributeName: "pages",new int[pagePatients.getTotalPages()]);
28         model.addAttribute( attributeName: "currentPage",p);
29         return "patients";
30     }
31 }
32

```

le fichier html :



```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.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>keyword:</label>
                <input type="text" name="keyword">
                <button type="submit" class="btn btn-info">chercher</button>
            </form>
        </div>
        <div class="card-footer">
            <table class="table">
                <thead>
                    <tr>
                        <th>ID</th> <th>Date</th> <th>Malade</th> <th>Score</th>
                    </tr>
                </thead>
                <tbody th:each="p:${ListPatients}">
                    <tr>
                        <td th:text="${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>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
</div>
```



localhost:8084/index/keyword=Imane

ID	Date	Malade	Score
3	Imane	2024-04-22 20:37:08.0	true 34
6	Imane	2024-04-22 20:41:55.0	true 34
9	Imane	2024-04-22 20:42:09.0	true 34
12	Imane	2024-04-22 21:04:43.0	true 34

0 1 2 3 4

Pour que la valeur recherchée doit être restée dans la barre de recherche :

```
© PatientController.java × <> Patients.html application.properties pom.xml (hopital) © Patient.java © PatientRepository.java
1 package ma.fsm.hopital.web;
2
3 import org.springframework.data.domain.PageRequest;
4 import org.springframework.ui.Model;
5 import lombok.AllArgsConstructor;
6 import ma.fsm.hopital.entities.Patient;
7 import ma.fsm.hopital.repository.PatientRepository;
8 import org.springframework.data.domain.Page;
9 import org.springframework.stereotype.Controller;
10 import org.springframework.web.bind.annotation.GetMapping;
11 import org.springframework.web.bind.annotation.RequestParam;
12 import java.util.List;
13
14 @Controller
15 @AllArgsConstructor
16 public class PatientController {
17     private PatientRepository patientRepository;
18
19     @GetMapping(PathVariable"/index")
20     public String index(Model model,
21                         @RequestParam(name="page",defaultValue = "0") int p,
22                         @RequestParam(name="size",defaultValue = "4")int s,
23                         @RequestParam(name="keyword",defaultValue = "") String kw)
24     {
25         Page<Patient> pagePatients=patientRepository.findByNomContains(kw,PageRequest.of(p,s));
26         model.addAttribute( attributeName: "ListPatients",pagePatients.getContent());
27         model.addAttribute( attributeName: "pages",new int[pagePatients.getTotalPages()]);
28         model.addAttribute( attributeName: "currentPage",p);
29         model.addAttribute( attributeName: "keyword",kw );
30     }
31     return "patients";
32 }
```

```
© PatientController.java × <> Patients.html application.properties pom.xml (hopital) © Patient.java © PatientRepository.java
6     <title>Title</title>
7     <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
8 </head>
9 <body>
10 <div class="p-3">
11     <div class="card">
12         <div class="card-header">Liste Patients</div>
13         <div class="card-body">
14             <form method="get" th:action="@{index}">
15                 <label>Keyword:</label>
16                 <input type="text" name="keyword" th:value="${keyword}" />
17                 <button type="submit" class="btn btn-info">chercher</button>
18             </form>
19         </div>
20         <div class="card-footer">
21             <table class="table">
22                 <thead>
23                     <tr>
24                         <th>ID</th> <th>Date</th> <th>Malade</th> <th>Score</th>
25                     </tr>
26                 <tr th:each="p:${ListPatients}">
27                     <td th:text="${p.id}"></td>
28                     <td th:text="${p.nom}"></td>
29                     <td th:text="${p.dateNaissance}"></td>
30                     <td th:text="${p.malade}"></td>
31                     <td th:text="${p.score}"></td>
32                 </tr>
33             </thead>
34             <tbody>
35                 <li th:each="value.item:${names}">
36             </tbody>
37         </table>
38     </div>
39 
```

A screenshot of a web browser window titled "localhost:8084/index?keyword=Hanane". The page displays a table titled "Liste Patients" with columns: ID, Date, Malade, and Score. There are five rows of data. Below the table is a navigation bar with buttons numbered 0 through 5.

ID	Date	Malade	Score
2	Hanane	2024-04-22 20:37:08.0	false 4321
5	Hanane	2024-04-22 20:41:55.0	false 4321
8	Hanane	2024-04-22 20:42:09.0	false 4321
11	Hanane	2024-04-22 21:04:43.0	false 4321

0 1 2 3 4 5

Pour supprimer un patient on va faire le suivant :

A screenshot of a web browser window titled "localhost:8084/index?keyword=Hanane". The page displays a table titled "Liste Patients" with columns: ID, Date, Malade, and Score. There are five rows of data. Each row contains a red "Delete" button in the last column. Below the table is a navigation bar with buttons numbered 0 through 17.

ID	Date	Malade	Score	
2	Hanane	2024-04-22 20:37:08.0	false 4321	<button>Delete</button>
3	Imane	2024-04-22 20:37:08.0	true 34	<button>Delete</button>
4	Mohamed	2024-04-22 20:41:55.0	false 34	<button>Delete</button>
5	Hanane	2024-04-22 20:41:55.0	false 4321	<button>Delete</button>

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

il faut comme même afficher un message de confirmation lorsqu'on veut supprimer un patient :

```

21   <table class="table">
22     <thead>
23       <tr>
24         <th>ID</th> <th>Date</th> <th>Malade</th> <th>Score</th>
25       </tr>
26     <tr th:each="p:${ListPatients}">
27       <td th:text="${p.id}"></td>
28       <td th:text="${p.nom}"></td>
29       <td th:text="${p.dateNaissance}"></td>
30       <td th:text="${p.malade}"></td>
31       <td th:text="${p.score}"></td>
32       <td>
33         <a href="#" onclick="return confirm('Etes vous sur?')><th:href="@{delete(id=${p.id})}" class="btn btn-danger">Delete</a>
34       </td>
35     </tr>
36   </thead>
37 </table>
38 <ul class="nav nav-pills">
39   <li th:each="value,item:${pages}">
40     <a th:href="@{index(page=${item.index},keyword=${keyword})}"
41       th:class="${(currentPage==item.index)?'btn btn-info ms-1':'btn btn-outline-info ms-1'}"
42       th:text="${item.index}"></a>
43
44   </li>
45 </ul>
46 </div>
47 </div>
48 </body>
49 </html>

```

html > body > div.p-3 > div.card > div.card-footer > table.table > thead > tr > td > a.btn.btn-danger

The screenshot shows a web application running on localhost:8084. A modal dialog box is displayed in the center, asking "Etes vous sure?" (Are you sure?). Below the modal, there is a table titled "Liste Patients" with the following data:

ID	Date	Malade	Score	
2	Hanane	2024-04-22 20:37:08.0	false	<button>Delete</button>
3	Imane	2024-04-22 20:37:08.0	true	<button>Delete</button>
4	Mohamed	2024-04-22 20:41:55.0	false	<button>Delete</button>
5	Hanane	2024-04-22 20:41:55.0	false	<button>Delete</button>

Below the table is a navigation bar with buttons labeled from 0 to 18.

Pour ne perde pas le mot clé et rester dans le même page après delete on va faire le suivants :

```


| ID | Date    | Malade                | Score |                                                                               |
|----|---------|-----------------------|-------|-------------------------------------------------------------------------------|
| 2  | Hanane  | 2024-04-22 20:37:08.0 | false | <a href="#" onclick="javascript:return confirm('Etes vous sure?')">Delete</a> |
| 3  | Imane   | 2024-04-22 20:37:08.0 | true  | <a href="#" onclick="javascript:return confirm('Etes vous sure?')">Delete</a> |
| 4  | Mohamed | 2024-04-22 20:41:55.0 | false | <a href="#" onclick="javascript:return confirm('Etes vous sure?')">Delete</a> |
| 5  | Hanane  | 2024-04-22 20:41:55.0 | false | <a href="#" onclick="javascript:return confirm('Etes vous sure?')">Delete</a> |



- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18

```

```

PatientController.java × Patients.html application.properties pom.xml (hopital) Patient.java PatientRepository.java
14  @Controller
15      @AllArgsConstructor
16  public class PatientController {
17      private PatientRepository patientRepository;
18
19      @GetMapping("/index")
20  public String index(Model model,
21                      @RequestParam(name="page",defaultValue = "0") int p,
22                      @RequestParam(name="size",defaultValue = "4")int s,
23                      @RequestParam(name="keyword",defaultValue = "") String kw)
24  {
25      Page<Patient> pagePatients=patientRepository.findByNomContains(kw,PageRequest.of(p,s));
26      model.addAttribute( attributeName: "ListPatients",pagePatients.getContent());
27      model.addAttribute( attributeName: "pages",new int[pagePatients.getTotalPages()]);
28      model.addAttribute( attributeName: "currentPage",p);
29      model.addAttribute( attributeName: "keyword",kw);
30      return "patients";
31  }
32  @GetMapping("/delete")
33  public String delete(Long id,String keyword,int page){
34      patientRepository.deleteById(id);
35      return "redirect:/index?page="+page+"&keyword="+keyword;
36
37  }
38
39
40  }
41
42

```

Je doit supprimer l'élément mohamed.

ID	Date	Malade	Score		
26	Hanane	2024-04-22 21:16:59.0	false	4321	<button>Delete</button>
27	Imane	2024-04-22 21:16:59.0	true	34	<button>Delete</button>
28	Mohamed	2024-04-22 21:17:04.0	false	34	<button>Delete</button>
29	Hanane	2024-04-22 21:17:04.0	false	4321	<button>Delete</button>

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

Après la suppression : On a reste dans la page 4 alors le problème est résolu.

The screenshot shows a web browser window with the URL `localhost:8084/index?page=4&keyword=`. The page title is "Liste Patients". There is a search bar with the placeholder "keyword:" and a "chercher" button. Below the search bar is a table with columns: ID, Date, Malade, Score, and a "Delete" button. The table contains four rows of patient data. At the bottom of the page is a navigation bar with numbered buttons from 0 to 29, where button 4 is highlighted in blue.

ID	Date	Malade	Score	
26	Hanane	2024-04-22 21:16:59.0	false	4321 <button>Delete</button>
27	Imane	2024-04-22 21:16:59.0	true	34 <button>Delete</button>
29	Hanane	2024-04-22 21:17:04.0	false	4321 <button>Delete</button>
30	Imane	2024-04-22 21:17:04.0	true	34 <button>Delete</button>

Ajoutons Quelques améliorations supplémentaires : En utilisons bootstrap_icons de search et delete.

The screenshot shows an IDE interface with the project structure on the left and the `pom.xml` file content on the right. The project structure includes a `Patient` module with `repository`, `web`, `resources`, and `test` sub-modules. The `pom.xml` file lists several dependencies, including `bootstrap-icons` and `mysql-connector-j`.

```
<dependency>
    <groupId>org.webjars.npm</groupId>
    <artifactId>bootstrap-icons</artifactId>
    <version>1.10.3</version>
</dependency>
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

```
1 <!DOCTYPE html>
2 <html lang="en" xmlns:th="http://thymeleaf.org">
3
4 <head>
5     <meta charset="UTF-8">
6     <title>Title</title>
7     <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
8     <link rel="stylesheet" href="/webjars/bootstrap-icons/1.10.3/font/bootstrap-icons.css">
9
10 </head>
11 <body>
12     <div class="p-3">
13         <div class="card">
14             <div class="card-header">Liste Patients</div>
15             <div class="card-body">
16                 <form method="get" th:action="@{index}">
17                     <label>keyword:</label>
18                     <input type="text" name="keyword" th:value="${keyword}" />
19                     <button type="submit" class="btn btn-info">
20                         <i class="bi bi-search"></i>
21                     </button>
22                 </form>
23             </div>
24             <div class="card-footer">
25                 <table class="table">
26                     <thead>
27                     <tr>
```

Fichier Patients.html :

```
28 </tr>
29 <tr th:each="p:${ListPatients}">
30     <td th:text="${p.id}"></td>
31     <td th:text="${p.nom}"></td>
32     <td th:text="${p.dateNaissance}"></td>
33     <td th:text="${p.malade}"></td>
34     <td th:text="${p.score}"></td>
35     <td>
36         <a onclick="javascript:return confirm('Etes vous sur?')"
37             th:href="@{/delete(id=${p.id},keyword=${keyword},page=${currentPage})}"
38             class="btn btn-danger">
39             <i class="bi bi-trash"></i>
40         </a>
41     </td>
42 </tr>
43 </thead>
44 </table>
45 <ul class="nav nav-pills">
46     <li th:each="value,item:${pages}">
47         <a th:href="@{/index(page=${item.index},keyword=${keyword})}"
48             th:class="${(currentPage==item.index)?'btn btn-info ms-1':'btn btn-outline-info ms-1'}"
49             th:text="${item.index}"></a>
50     </li>
51     </ul>
52 </div>
53 </div>
54 </body>
55 </html>
```

html > body > div.p-3 > div.card > div.card-footer

Liste Patients

keyword: 🔍

ID	Date	Malade	Score	
39	Imane	2024-04-22 22:02:07.0	true	刪除
40	Mohamed	2024-04-22 22:10:50.0	false	刪除
41	Hanane	2024-04-22 22:10:51.0	false	刪除
42	Imane	2024-04-22 22:10:51.0	true	刪除

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

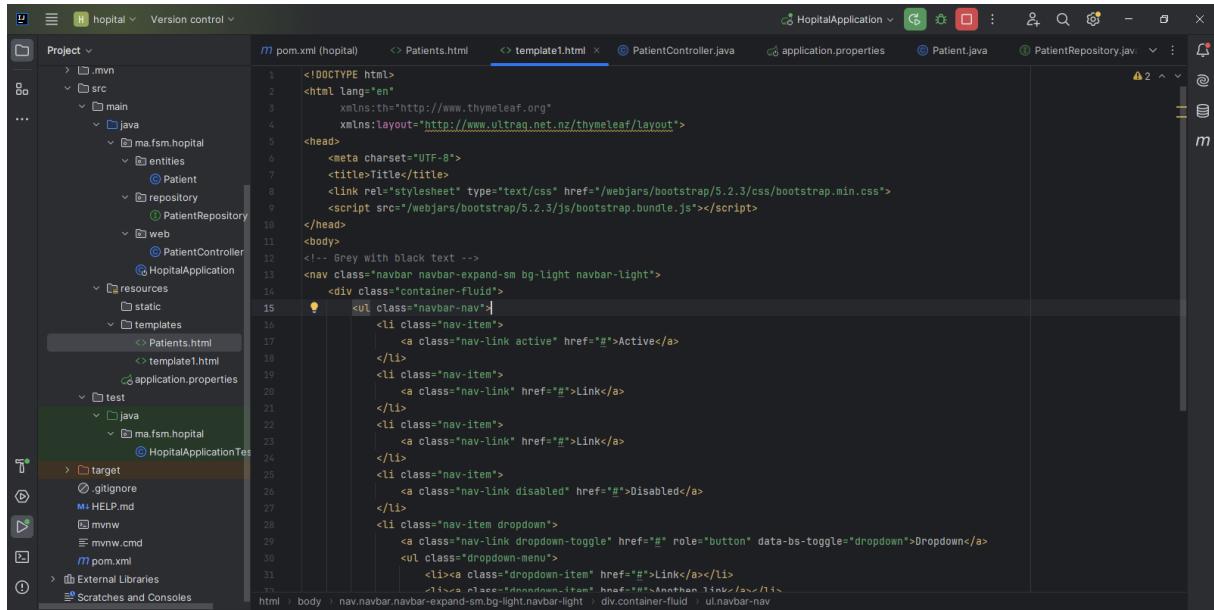
⇒ Partie 2 :

- Créer une page template1 Télécharger les dépendances.

```

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
    <groupId>nz.net.ultraq.thymeleaf</groupId>
    <artifactId>thymeleaf-layout-dialect</artifactId>
</dependency>
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>bootstrap</artifactId>
    <version>5.2.3</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>

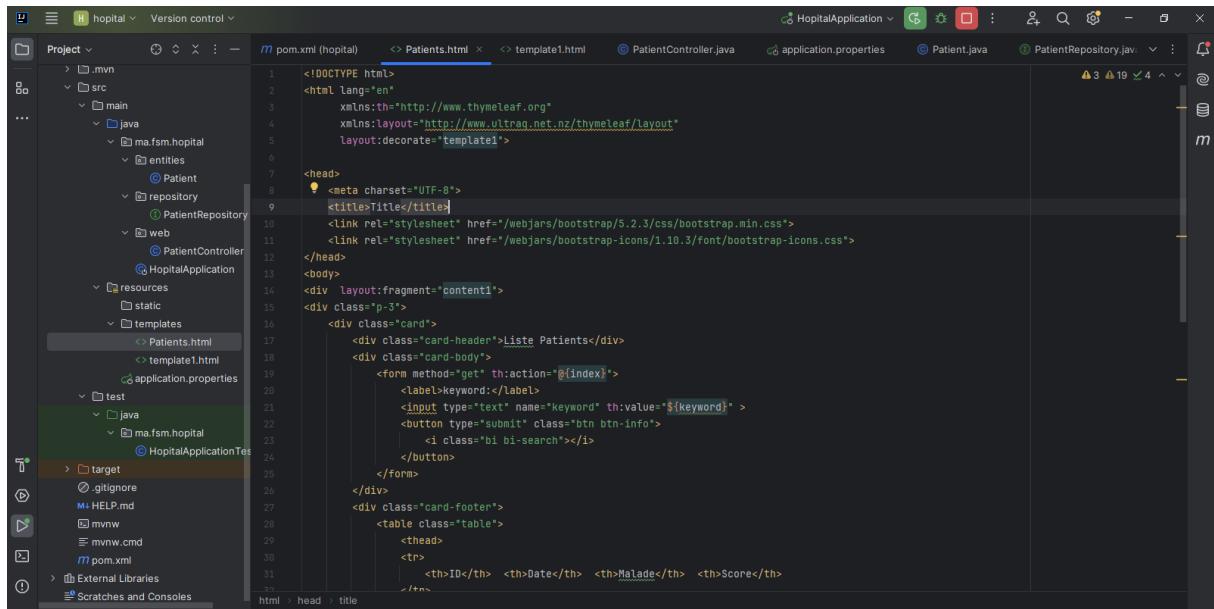
```



```
<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultrag.net.nz/thymeleaf/layout">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
    <script src="/webjars/bootstrap/5.2.3/js/bootstrap.bundle.js"></script>
</head>
<body>
    <!-- Grey with black text -->
    <nav class="navbar navbar-expand-sm bg-light navbar-light">
        <div class="container-fluid">
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link active" href="#">Active</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">Link</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">Link</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link disabled" href="#">Disabled</a>
                </li>
                <li class="nav-item dropdown">
                    <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">Dropdown</a>
                    <ul class="dropdown-menu">
                        <li><a class="dropdown-item" href="#">Link</a></li>
                    </ul>
                </li>
            </ul>
        </div>
    </nav>

```

Et utiliser la page template1 dans le fichier html Patients.



```
<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultrag.net.nz/thymeleaf/layout"
      layout:decorate="template1">
<head>
    <meta charset="UTF-8">
    <title>Title</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">
</head>
<body>
    <div layout:fragment="content">
        <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>Keyword:</label>
                        <input type="text" name="Keyword" th:value="${keyword}">
                        <button type="submit" class="btn btn-info">
                            <i class="bi bi-search"></i>
                        </button>
                    </form>
                </div>
                <div class="card-footer">
                    <table class="table">
                        <thead>
                            <tr>
                                <th>ID</th> <th>Date</th> <th>Malade</th> <th>Score</th>
                            </tr>
                        </thead>

```

On exécute notre application.

localhost:8084/index

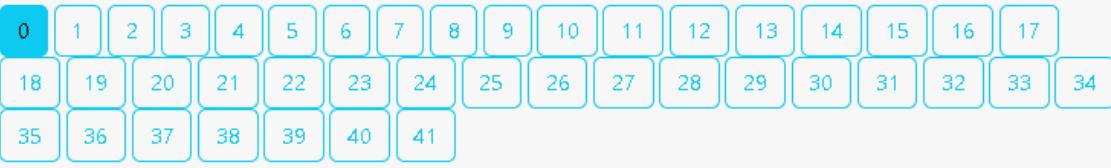
Gmail Maps Testing_Videos - Ju... Google - ملخص الأخبار Go... À faire Activité Pratique N°...

Active Link Link Disabled Dropdown ▾

Liste Patients

keyword: 

ID	Date	Malade	Score
9	Imane	2024-04-22 20:42:09.0	true 34 
11	Hanane	2024-04-22 21:04:43.0	false 4321 
12	Imane	2024-04-22 21:04:43.0	true 34 
13	Mohamed	2024-04-22 21:12:30.0	false 34 



➤ Colorer la barre de navigation.

```

patients.html      <> template1.html x  © PatientController.java   ⚡ application.properties   © Patient.java   ⚡ PatientRepos
1  <!DOCTYPE html>
2  <html lang="en"
3      xmlns:th="http://www.thymeleaf.org"
4      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
5  <head>
6      <meta charset="UTF-8">
7      <title>Title</title>
8      <link rel="stylesheet" type="text/css" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
9      <script src="/webjars/bootstrap/5.2.3/js/bootstrap.bundle.js"></script>
10 </head>
11 <body>
12     <!-- Grey with black text -->
13     <nav class="navbar navbar-expand-sm bg-dark navbar-dark">
14         <div class="container-fluid">
15             <ul class="navbar-nav">
16                 <li class="nav-item">
17                     <a class="nav-link active" href="#">Active</a>
18                 </li>
19                 <li class="nav-item">
20                     <a class="nav-link" href="#">Link</a>
21                 </li>
22                 <li class="nav-item">
23                     <a class="nav-link" href="#">Link</a>
24                 </li>
25                 <li class="nav-item">
26                     <a class="nav-link disabled" href="#">Disabled</a>
27                 </li>
28                 <li class="nav-item dropdown">
29                     <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">Dropdown</a>
30                     <ul class="dropdown-menu">
31                         <li><a class="dropdown-item" href="#">Link</a></li>
32                         <li><a class="dropdown-item" href="#">Another Link</a></li>
33                     </ul>
34                 </li>
35             </ul>
36         </div>
37     </nav>
38 
```

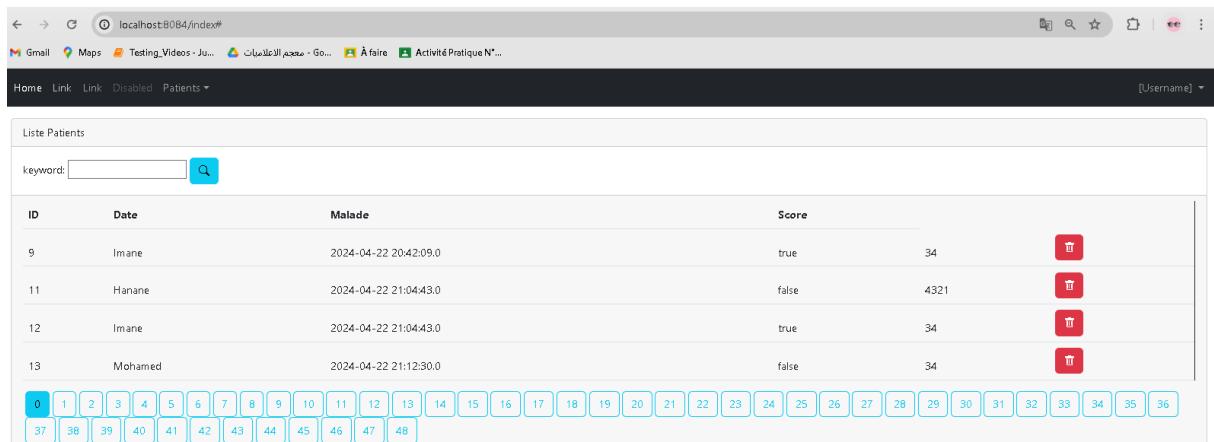
The screenshot shows a web browser window with the URL `localhost:8084/index#`. The page displays a table titled "Liste Patients". The table has columns: ID, Date, Malade, and Score. There are four rows of data:

ID	Date	Malade	Score
9	2024-04-22 20:42:09.0	Imane	true 34
11	2024-04-22 21:04:43.0	Hanane	false 4321
12	2024-04-22 21:04:43.0	Imane	true 34
13	2024-04-22 21:12:30.0	Mohamed	false 34

Below the table is a navigation bar with links: Active, Link, Link, Disabled, and Dropdown. At the bottom, there is a search bar labeled "keyword:" and a set of numbered buttons from 0 to 42.

Ajoutons au header les éléments suivants ajouter un nouveau patient, chercher un patient, et se déconnecter .

```
m pom.xml (hopital) <> Patients.html <> template1.html x PatientController.java application.properties Patient.java PatientRepo
20
21         </li>
22         <li class="nav-item">
23             <a class="nav-link" href="#">Link</a>
24         </li>
25         <li class="nav-item">
26             <a class="nav-link disabled" href="#">Disabled</a>
27         </li>
28         <li class="nav-item dropdown">
29             <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">Patients</a>
30             <ul class="dropdown-menu">
31                 <li><a class="dropdown-item" th:href="@{/formPatients}">Nouveau</a></li>
32                 <li><a class="dropdown-item" th:href="@{/index}">Chercher Patient</a></li>
33             </ul>
34         </li>
35     </ul>
36     <ul class="navbar-nav">
37         <li class="nav-item dropdown">
38             <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">
39                 [Username]
40             </a>
41             <ul class="dropdown-menu">
42                 <li><a class="dropdown-item" th:href="@{/formPatients}">Logout</a></li>
43             </ul>
44         </li>
45     </ul>
46   </div>
47 </nav>
48 <section layout:fragment="content1">
49
50 </section>
51
```



La méthode « formPatients » :On va créer un fichier HTML que l'on appelle formPatients.html, a fin d'afficher le formulaire d'ajout des patients.

```

15     @AllArgsConstructor
16     public class PatientController {
17         private PatientRepository patientRepository;
18
19         @GetMapping(@RequestMapping("/index"))
20         public String index(Model model,
21             @RequestParam(name="page",defaultValue = "0") int p,
22             @RequestParam(name="size",defaultValue = "4")int s,
23             @RequestParam(name="keyword",defaultValue = "") String kw){
24
25             Page<Patient> pagePatients=patientRepository.findByNomContains(kw,PageRequest.of(p,s));
26             model.addAttribute("listPatients",pagePatients.getContent());
27             model.addAttribute("pages",new int[pagePatients.getTotalPages()]);
28             model.addAttribute("currentPage",p);
29             model.addAttribute("keyword",kw);
30             return "patients";
31         }
32         @GetMapping(@RequestMapping("/delete"))
33         public String delete(Long id, String keyword,int page){
34             patientRepository.deleteById(id);
35             return "redirect:/index?page=" + page + "&keyword=" + keyword;
36         }
37         @GetMapping(@RequestMapping("/formPatients"))
38         public String formPatients(Model model){
39             model.addAttribute("patient",new Patient());
40             return "formPatients";
41         }
42     }
43 }
```

```

13     <div layout:fragment="content">
14         <div class="col-md-6 offset-3">
15
16             <form method="post" th:action="@{save}">
17                 <div>
18                     <label for="nom">Nom</label>
19                     <input id="nom" class="form-control" type="text" name="nom" th:value="${patient.nom}">
20                     <span th:errors="${patient.nom}"></span>
21                 </div>
22                 <div>
23                     <label>Date Naissance</label>
24                     <input class="form-control" type="date" name="dateNaissance" th:value="${patient.dateNaissance}">
25                     <span th:errors="${patient.dateNaissance}"></span>
26                 </div>
27                 <div>
28                     <label>Malade</label>
29                     <input class="checkbox" type="checkbox" name="malade" th:checked="${patient.malade}">
30                     <span th:errors="${patient.malade}"></span>
31                 </div>
32                 <div>
33                     <label>Score</label>
34                     <input class="form-control" type="text" name="score" th:value="${patient.score}">
35                     <span th:errors="${patient.score}"></span>
36                 </div>
37                 <button type="submit" class="btn btn-primary">Save</button>
38             </form>
39         </div>
40     </div>
41 
```

The screenshot shows a web browser window with the URL `localhost:8084/formPatients`. The page displays a form with the following fields:

- Nom:** An input field with the placeholder text "Nom".
- Date Naissance:** A date input field with the placeholder text "jj/mm/aaaa".
- Malade:** A checkbox input field.
- Score:** A text input field with the placeholder text "Score".
- Save:** A blue "Save" button at the bottom of the form.

➤ méthode save() :

```
29     model.addAttribute( attributeName: "keyword", kw);
30     return "patients";
31 }
32
33 @GetMapping(@RequestMapping("/delete"))
34 public String delete(Long id, String keyword, int page) {
35     patientRepository.deleteById(id);
36     return "redirect:/index?page=" + page + "&keyword=" + keyword;
37 }
38
39 }
40
41 @GetMapping("/formPatients")
42 public String formPatients(Model model) {
43     model.addAttribute( attributeName: "patient", new Patient());
44     return "formPatients";
45 }
46
47 @PostMapping (path = @RequestMapping("/save"))
48 public String formPatients(Model model, Patient patient) {
49     patientRepository.save(patient);
50     return "formPatients";
51 }
52
53 }
54
55
56
57 }
```

Ajout de annotation `@DateTimeFormat` au classe Patient :

```
2
3
4 import jakarta.persistence.Entity;
5 import jakarta.persistence.GeneratedValue;
6 import jakarta.persistence.GenerationType;
7 import jakarta.persistence.Id;
8 import lombok.AllArgsConstructor;
9 import lombok.Builder;
10 import lombok.Data;
11 import lombok.NoArgsConstructor;
12 import org.springframework.format.annotation.DateTimeFormat;
13
14 import java.util.Date;
15     13 usages
16     @Entity
17     @Data @NoArgsConstructor @AllArgsConstructor
18     public class Patient {
19         @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
20         private Long id;
21         private String nom;
22         @DateTimeFormat(pattern = "yyyy-MM-dd")
23         private Date dateNaissance;
24         private boolean malade;
25         private int score;
26
27
28 }
```

remplir le formulaire par le patient fatima ezzahrae

Nom
fatima ezzahrae
Date Naissance
jj/mm/aaaa
Malade
Score
44
Save

Le patient est bien enregistrer dans la listes des patients.

ID	Date	Malade	Score
247	2001-12-16 00:00:00.0	false	44

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 35 36
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59

- Faire la validation des formulaires Ajouter les dépendances de spring boot validation.

```
<artifactId>hopital</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>hopital</name>
<description>hopital</description>
<properties>
    <java.version>17</java.version>
</properties>
<dependencies> Edit Starters...
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <!-- https://mvnrepository.com/artifact/nz.net.ultraq.thymeleaf/thymeleaf-layout-dialect -->
    <dependency>
        <groupId>nz.net.ultraq.thymeleaf</groupId>
        <artifactId>thymeleaf-layout-dialect</artifactId>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-validation -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.webjars/bootstrap -->
    <dependency>
```

- Ajouter les annotation de validation.

```
33
34
35     @GetMapping(@RequestMapping("/delete"))
36     public String delete(Long id, String keyword, int page) {
37         patientRepository.deleteById(id);
38         return "redirect:/index?page=" + page + "&keyword=" + keyword;
39     }
40
41 }
42
43     @GetMapping(@RequestMapping("/formPatients"))
44     public String formPatients(Model model) {
45         model.addAttribute("patient", new Patient());
46         return "formPatients";
47     }
48
49     @PostMapping (path = @RequestMapping("/save"))
50     public String formPatientsSave(Model model, @Valid Patient patient, BindingResult bindingResult) {
51         if (bindingResult.hasErrors()) return "formPatients";
52         patientRepository.save(patient);
53         return "formPatients";
54     }
55
56 }
57
58
59
60 }
```

Ajouter annotation annotation @valid au PatientController pour ne pas remplir la base de données chaque fois.

```
m pom.xml (hopital)  ⚡ Patient.java  <> Patients.html  <> template1.html  ⚡ PatientController.java x  <> formPatients.html
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

    @GetMapping("/delete")
    public String delete(Long id, String keyword, int page) {
        patientRepository.deleteById(id);
        return "redirect:/index?page=" + page + "&keyword=" + keyword;
    }

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

    @PostMapping (path = "/save")
    public String formPatients(Model model, @Valid Patient patient, BindingResult bindingResult) {
        if (bindingResult.hasErrors()) return "formPatients";
        patientRepository.save(patient);
        return "formPatients";
    }
}
```

D'où la validation est bien effectuée.

localhost:8084/save

Gmail Maps Testing_Videos - Ju... Google االعالنات Go... À faire Activité Pratique N°...

Home Link Link Disabled Patients ▾ [Username] ▾

Nom

la taille doit être comprise entre 4 et 44
ne doit pas être vide

Date Naissance jj/mm/aaaa

Malade

Score 0

doit être supérieur ou égal à 100

Save

En ajoute le bouton Edit et ainsi la méthode EditPatient.

A screenshot of a code editor showing a JSP template named `Patients.html`. The code includes a table for displaying patient data and a navigation bar with page links.

```
<tr>
    <th>ID</th> <th>Date</th> <th>Malade</th> <th>Score</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=${currentPage})}"
            class="btn btn-danger">
            <i class="bi bi-trash"></i>
        </a>
    </td>
    <td>
        <a th:href="@{editPatients(id=${p.id})}" class="btn btn-success">
            Edit
        </a>
    </td>
</tr>
</thead>
</table>
<ul class="nav nav-pills">
    <li th:each="value,item:${pages}">
        <a th:href="@{/index(page=${item.index},keyword=${keyword})}">
            th:class="${(currentPage==item.index)?'btn btn-info ms-1':'btn btn-outline-info ms-1'}"
            th:text="${item.index}"></a>
    </li>
</ul>
```

A screenshot of a code editor showing a Java controller named `PatientController.java`. It contains methods for listing patients, saving new patients, and editing existing patients.

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

@PostMapping(path = @"/save")
public String save(Model model, @Valid Patient patient, BindingResult bindingResult) {
    if (bindingResult.hasErrors()) return "formPatients";
    patientRepository.save(patient);
    return "redirect:/index";
}

@PutMapping(path = @"/editPatients")
public String editPatients(Model model, Long id) {
    Patient patient=patientRepository.findById(id).orElse(null);
    if (patient==null) throw new RuntimeException("Patient introuvable");
    model.addAttribute("patient",patient);
    return "editPatients";
}
```

A screenshot of a code editor showing a JSP template named `editPatients.html`. It displays a form for editing patient information.

```
<link rel="stylesheet" href="/webjars/bootstrap-icons/1.10.3/font/bootstrap-icons.css">
</head>
<body>
<div layout:fragment="content">
    <div class="col-md-6 offset-3">

        <form method="post" th:action="@{save}">
            <div>
                <label>ID</label>
                <input class="form-control" type="text" 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>
        </form>
    </div>
</div>
```

On va prendre `id=9` on va essayer d'édition les informations du patient .

Liste Patients																																								
keyword: <input type="text"/> <input type="button" value="Search"/>																																								
ID	Date	Malade	Score																																					
9	Imane	2024-04-22 20:42:09.0	true	34	<input type="button" value="Delete"/> <input type="button" value="Edit"/>																																			
11	Hanane	2024-04-22 21:04:43.0	false	4321	<input type="button" value="Delete"/> <input type="button" value="Edit"/>																																			
12	Imane	2024-04-22 21:04:43.0	true	34	<input type="button" value="Delete"/> <input type="button" value="Edit"/>																																			
13	Mohamed	2024-04-22 21:12:30.0	false	34	<input type="button" value="Delete"/> <input type="button" value="Edit"/>																																			
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	35	36				
37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77

Editer les informations.

localhost:8084/editPatients?id=9

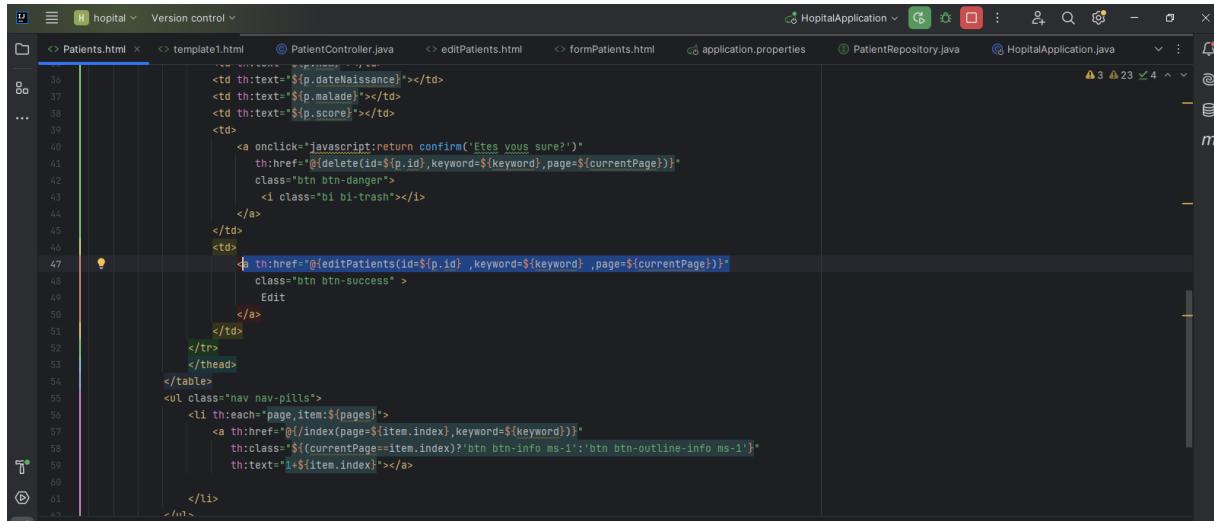
Home Link Link Disabled Patients [Username]

ID	9
Nom	zahrae
Date Naissance	dd/mm/yyyy
Malade	<input checked="" type="checkbox"/>
Score	134
<input type="button" value="Save"/>	

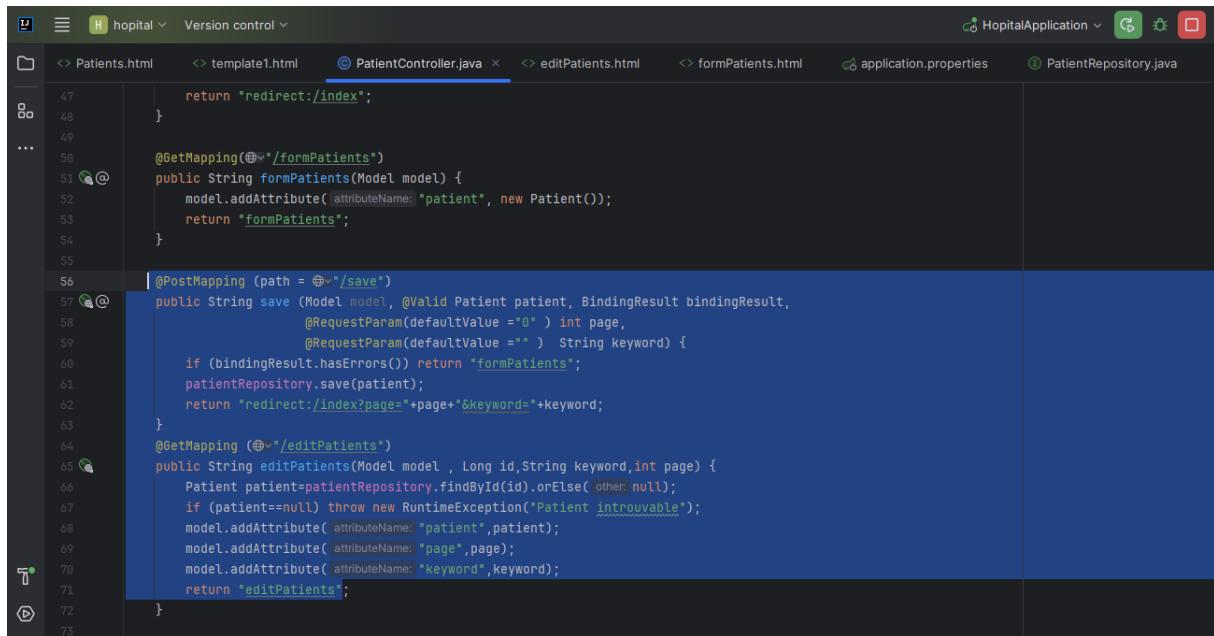
editPatient passe bien.

Liste Patients																																									
keyword: <input type="text"/> <input type="button" value="Search"/>																																									
ID	Date	Malade	Score																																						
9	zahrae	2021-02-12 00:00:00.0	true	134	<input type="button" value="Delete"/> <input type="button" value="Edit"/>																																				
11	Hanane	2024-04-22 21:04:43.0	false	4321	<input type="button" value="Delete"/> <input type="button" value="Edit"/>																																				
12	Imane	2024-04-22 21:04:43.0	true	34	<input type="button" value="Delete"/> <input type="button" value="Edit"/>																																				
13	Mohamed	2024-04-22 21:12:30.0	false	34	<input type="button" value="Delete"/> <input type="button" value="Edit"/>																																				
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	35	36					
37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78

Pour rester dans la même page lorsque on met une modification des données de patient.



```
<td th:text="${p.dateNaissance}"></td>
<td th:text="${p.nom}></td>
<td th:text="${p.score}"></td>
<td>
    <a onclick="return confirm('Etes vous sur?')>
        th:href="@{delete?id=${p.id},keyword=${keyword},page=${currentPage}}>
        class="btn btn-danger">
            <i class="bi bi-trash"></i>
    </a>
</td>
<td>
    <a th:href="@{editPatients(id=${p.id} ,keyword=${keyword} ,page=${currentPage})}">
        class="btn btn-success" >
            Edit
    </a>
</td>
</tr>
</thead>
</table>
<ul class="nav nav-pills">
    <li th:each="page, item:${pages}">
        <a th:href="@{/index?page=${item.index}, keyword=${keyword}}>
            th:class="${(currentPage==item.index)? 'btn btn-info ms-1': 'btn btn-outline-info ms-1'}>
            th:text="1+${item.index}"></a>
    </li>
</ul>
```

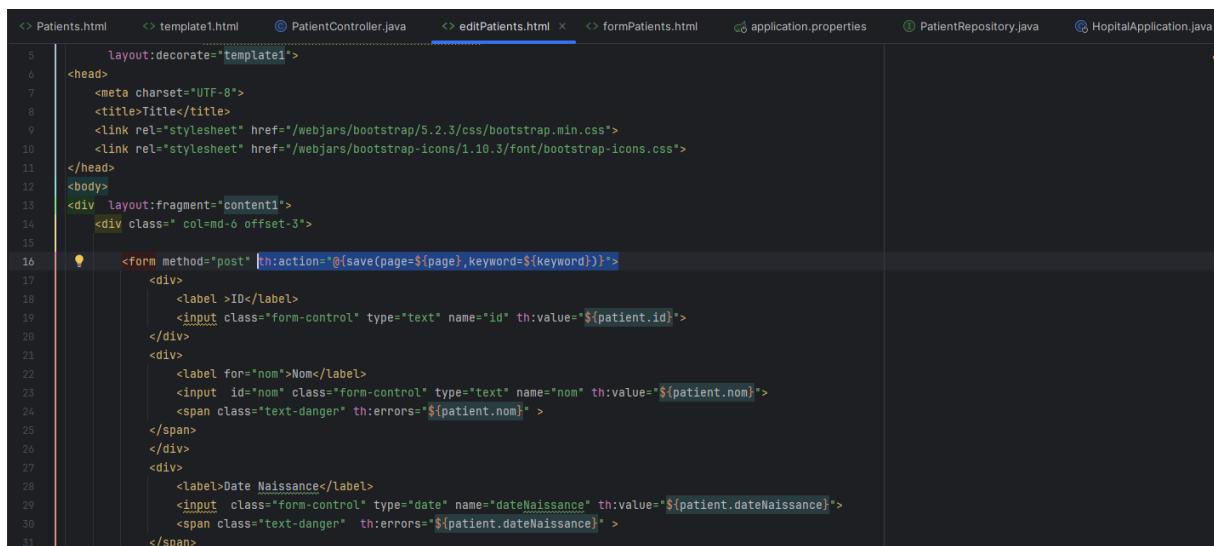


```
return "redirect:/index";}

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



```
<head>
    <meta charset="UTF-8">
    <title>Title</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">
</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>ID</label>
                <input class="form-control" type="text" 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>
        </form>
    </div>
</div>
```

localhost:8084/index?page=7&keyword=Mohamed

Liste Patients																																			
ID	Date	Malade	Score																																
103	Mohamed	2024-04-24 21:38:44.0	false	34	 																														
106	Mohamed	2024-04-24 21:42:26.0	false	34	 																														
109	Mohamed	2024-04-24 21:44:24.0	false	34	 																														
112	Mohamed	2024-04-24 21:49:01.0	false	34	 																														
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	35	36

localhost:8084/editPatients?id=103&keyword=Mohamed&page=7

ID	<input type="text" value="103"/>
Nom	<input type="text" value="Mohammed"/>
Date Naissance	<input type="text" value="jj/mm/aaaa"/>
Malade	<input type="checkbox"/>
Score	<input type="text" value="434"/>
	<input type="button" value="Save"/>

D'où on a rester dans la même page .

localhost:8084/index?page=7&keyword=Mohamed

Liste Patients																																		
ID	Date	Malade	Score																															
109	Mohamed	2024-04-24 21:44:24.0	false	34	 																													
112	Mohamed	2024-04-24 21:49:01.0	false	34	 																													
115	Mohamed	2024-04-24 21:49:40.0	false	34	 																													
118	Mohamed	2024-04-24 21:54:02.0	false	34	 																													
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	35

On ajouter un lien pour le Home :

```

4      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
5      <head>
6          <meta charset="UTF-8">
7          <title>Title</title>
8          <link rel="stylesheet" type="text/css" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
9          <script src="/webjars/bootstrap/5.2.3/js/bootstrap.bundle.js"></script>
10     </head>
11     <!-- Grey with black text -->
12     <nav class="navbar navbar-expand-sm bg-dark navbar-dark">
13         <div class="container-fluid">
14             <ul class="navbar-nav">
15                 <li class="nav-item">
16                     <a class="nav-link active" th:href="@{index}">Home</a>
17                 </li>
18                 <li class="nav-item">
19                     <a class="nav-link" href="#">Link</a>
20                 </li>
21                 <li class="nav-item">
22                     <a class="nav-link" href="#">Link</a>
23                 </li>
24                 <li class="nav-item">
25                     <a class="nav-link disabled" href="#">Disabled</a>
26                 </li>
27             <li class="nav-item dropdown">
28                 <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">Patients</a>
29                 <ul class="dropdown-menu">
30

```

On ne peut pas modifier id.

```

5      layout:decorate="template1">
6      <head>
7          <meta charset="UTF-8">
8          <title>Title</title>
9          <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
10         <link rel="stylesheet" href="/webjars/bootstrap-icons/1.10.3/font/bootstrap-icons.css">
11     </head>
12     <body>
13         <div layout:fragment="content1">
14             <div class="col-md-6 offset-3">
15
16                 <form method="post" th:action="@{save(page=${page},keyword=${keyword})}">
17                     <div>
18                         <label>ID:</label>
19                         <label th:text="${patient.id}"></label>
20                         <input class="form-control" type="hidden" name="id" th:value="${patient.id}">
21                     </div>
22                     <div>
23                         <label for="nom">Nom</label>
24                         <input id="nom" class="form-control" type="text" name="nom" th:value="${patient.nom}">
25                         <span class="text-danger" th:errors="${patient.nom}">
26                         </span>
27                     </div>
28                     <div>
29                         <label>Date Naissance</label>
30                         <input class="form-control" type="date" name="dateNaissance" th:value="${patient.dateNaissance}">
31                         <span class="text-danger" th:errors="${patient.dateNaissance}">

```

localhost:8084/editPatients?id=109&keyword=Mohamed&page=7

ID: 109
Nom: Mohamed
Date Naissance: jj/mm/aaaa
Malade:
Score: 34
Save

⇒ Partie 3 :

Sécurité avec Spring security :

On ajoute les dépendances de spring security au fichier pom.xml.

On exécute notre application qui génère un mode de passe pour accès au application.

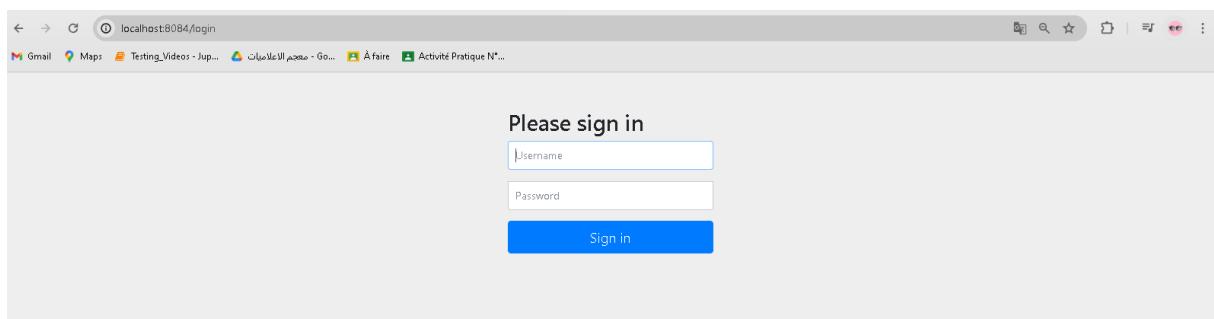
```

<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>

```

Using generated security password: 66831ca4-38c3-47ea-bd80-8b54a0c09e06

This generated password is for development use only. Your security configuration must be updated before running your application in production.



On sign in with username user and password genered by application.

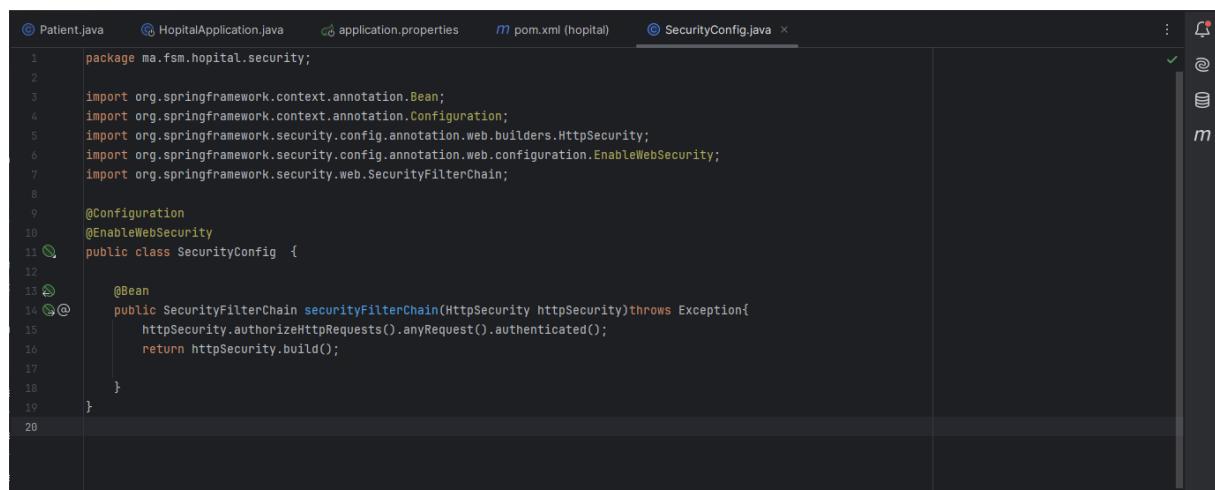


Après en accès a notre application avec succès.

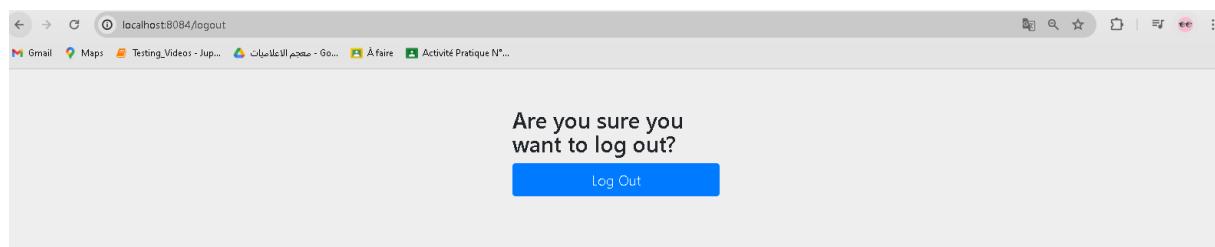
ID	Date	Malade	Score	
3	hanane		true	Delete Edit
5	Hanane	2024-04-26 14:42:49.0	false	Delete Edit
8	Hanane	2024-04-26 14:43:59.0	false	Delete Edit

➤ Personnaliser la configuration de security.

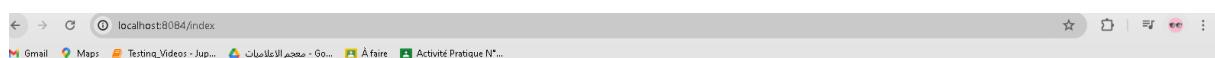
Donc pour se faire on va créer une classe SecurityConfig, dont laquelle on travaille avec deux annotations `@Configuration` et `@EnableWebSecurity`. On utilise également l'annotation `@Bean` pour que notre méthode s'exécute au démarrage. La ligne de configuration sert à nécessiter l'authentification pour toute requête de notre application :



```
1 package ma.fsm.hopital.security;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5 import org.springframework.security.config.annotation.web.builders.HttpSecurity;
6 import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
7 import org.springframework.security.web.SecurityFilterChain;
8
9 @Configuration
10 @EnableWebSecurity
11 public class SecurityConfig {
12
13     @Bean
14     public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception{
15         httpSecurity.authorizeHttpRequests().anyRequest().authenticated();
16         return httpSecurity.build();
17     }
18 }
19
20 }
```



L'accès est refusé.



Pour afficher le formulaire d'authentification .

```

3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5 import org.springframework.config.annotation.web.builders.HttpSecurity;
6 import org.springframework.security.core.userdetails.User;
7 import org.springframework.security.provisioning.InMemoryUserDetailsManager;
8 import org.springframework.security.web.SecurityFilterChain;
9
10 import static org.springframework.security.core.userdetails.User.withUsername;
11
12
13 @Configuration
14 @EnableWebSecurity
15 public class SecurityConfig {
16
17
18
19
20 @Bean
21 @Override
22 public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception{
23     httpSecurity.formLogin();
24     httpSecurity.authorizeHttpRequests().anyRequest().authenticated();
25     return httpSecurity.build();
26 }
27
28 }
```

Please sign in

You have been signed out

user1

Sign in

DevTools is now available in French. Always match Chrome's language [Switch DevTools to French](#) [Don't show again](#)

Elements Console Sources Network Performance Memory Application > Styles Computed Layout Event Listeners >

```

<!DOCTYPE html>
<html lang="en">
  <head> ...
  </head>
  <body>
    <div> ...
      <form class="form-signin" method="post" action="/login">
        ...
      </form>
    </div>
  </body>
</html>
```

html body div.container

un objet de type **InMemoryAuthentication**, qui permet de préciser au mémoire les utilisateurs qui ont le droit d'accéder à l'application.

On utilise **{noop}** pour dire à spring security ne pas utiliser un password encoder, il va juste comparer le mot de passe tel qu'il est.

```

import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
import org.springframework.security.web.SecurityFilterChain;

import static org.springframework.security.core.userdetails.User.withUsername;

@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public InMemoryUserDetailsManager inMemoryUserDetailsManager(){
        return new InMemoryUserDetailsManager(
                User.withUsername("user1").password("{noop}1234").roles("User").build(),
                User.withUsername("user2").password("{noop}1234").roles("User").build(),
                User.withUsername("admin").password("{noop}1234").roles("User", "ADMIN").build()
        );
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception{
        httpSecurity.formLogin();
        httpSecurity.authorizeHttpRequests().anyRequest().authenticated();
        return httpSecurity.build();
    }
}

```

Please sign in

You have been signed out

user2
....

Sign in

Name	Status	Type	Initiator	Size	Time	Waterfall
loginLogout	200	document	Other	2.0 kB	5 ms	
bootstrap.min.css	200	stylesheet	localhost:8084/login?logout	(disk cache)	5 ms	
signin.css	200	stylesheet	localhost:8084/login?logout	(disk cache)	5 ms	

Liste Patients

keyword:

ID	Date	Malade	Score																						
1	Mohamed	2024-04-26 14:34:26.0	false	134																					
3	hanane		true	134																					
4	Mohamed	2024-04-26 14:42:49.0	false	134																					
5	Hanane	2024-04-26 14:42:49.0	false	421																					
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

Name	Status	Type	Initiator	Size	Time	Waterfall
login	302	document	Other	489 B	179 ms	
localhost	302	document	login	444 B	6 ms	
index	200	document	localhost:8084	11.0 kB	77 ms	
bootstrap.min.css	200	stylesheet	index:6	195 kB	32 ms	
bootstrap.bundle.js	200	script	index:7	208 kB	34 ms	
bootstrap-icons.css	200	stylesheet	index:11	96.0 kB	44 ms	
bootstrap.min.css	200	stylesheet	index:10	195 kB	20 ms	
bootstrap-icons.woff2?724e3eb84...	200	font	bootstrap-icons.css	122 kB	35 ms	

- Utilisant encoder BCrypt :

```
34     @Override
35     public void run(String... args) throws Exception {
36
37         /*Patient patient = new Patient();
38         patient.setId(null);
39         patient.setNom("Mohamed");
40         patient.setDateNaissance(new Date());
41         patient.setMalade(false);
42         patient.setScore(23);
43
44         Patient patient2 = new Patient(null, "Yassine", new Date(), false, 123);
45         Patient patient3 = Patient.builder()
46             .nom("Imane")
47             .dateNaissance(new Date())
48             .score(56)
49             .malade(true)
50             .build();*/
51
52         patientRepository.save(new Patient(id: null, nom: "Mohamed", new Date(), malade: false, score: 134));
53         patientRepository.save(new Patient(id: null, nom: "Hanane", new Date(), malade: false, score: 421));
54         patientRepository.save(new Patient(id: null, nom: "Imane", new Date(), malade: true, score: 134));
55     }
56
57     @Bean
58     PasswordEncoder passwordEncoder(){
59         return new BCryptPasswordEncoder();
60     }
61 }
```

```
0  import org.springframework.security.core.userdetails.User;
1  import org.springframework.security.crypto.password.PasswordEncoder;
2  import org.springframework.security.provisioning.InMemoryUserDetailsManager;
3  import org.springframework.security.web.SecurityFilterChain;
4
5  import static org.springframework.security.core.userdetails.User.withUsername;
6
7  @Configuration
8  @EnableWebSecurity
9  public class SecurityConfig {
10
11     @Autowired
12     private PasswordEncoder passwordEncoder;
13
14     @Bean
15     public InMemoryUserDetailsManager inMemoryUserDetailsManager{
16         return new InMemoryUserDetailsManager(
17             User.withUsername("user1").password(passwordEncoder.encode("1234")).roles("User").build(),
18             User.withUsername("user2").password(passwordEncoder.encode("1234")).roles("User").build(),
19             User.withUsername("admin").password(passwordEncoder.encode("1234")).roles("User","ADMIN").build()
20         );
21     }
22
23     @Bean
24     public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception{
25         httpSecurity.formLogin();
26         httpSecurity.authorizeHttpRequests().anyRequest().authenticated();
27         return httpSecurity.build();
28     }
29
30 }
31
32
33
34
35
36
37 }
```

Le nom d'utilisateur authentifié User1 s'affiche au lieu de username.

The screenshot shows a web application running at localhost:8084/index?continue. The URL bar also includes links to Gmail, Maps, Testing_Videos - Jupyter Notebook, and Activité Pratique N°...

The page title is "Liste Patients". It features a search bar with a placeholder "keyword:" and a search icon. Below the search bar is a table with the following data:

ID	Date	Malade	Score	Action	Action
1	Mohamed	2024-04-26 14:34:26.0	false	View	Edit
3	hanane		true	View	Edit
4	Mohamed	2024-04-26 14:42:49.0	false	View	Edit
5	Hanane	2024-04-26 14:42:49.0	false	View	Edit

At the bottom of the table, there is a navigation bar with numbered buttons from 1 to 32, indicating multiple pages of data.

Pour se déconnecter, on a deux méthodes pour faire .



ID	Date	Malade	Score		
1	Mohamed	2024-04-26 14:34:26.0	false	134	
3	hanane		true	134	
4	Mohamed	2024-04-26 14:42:49.0	false	134	
5	Hanane	2024-04-26 14:42:49.0	false	421	

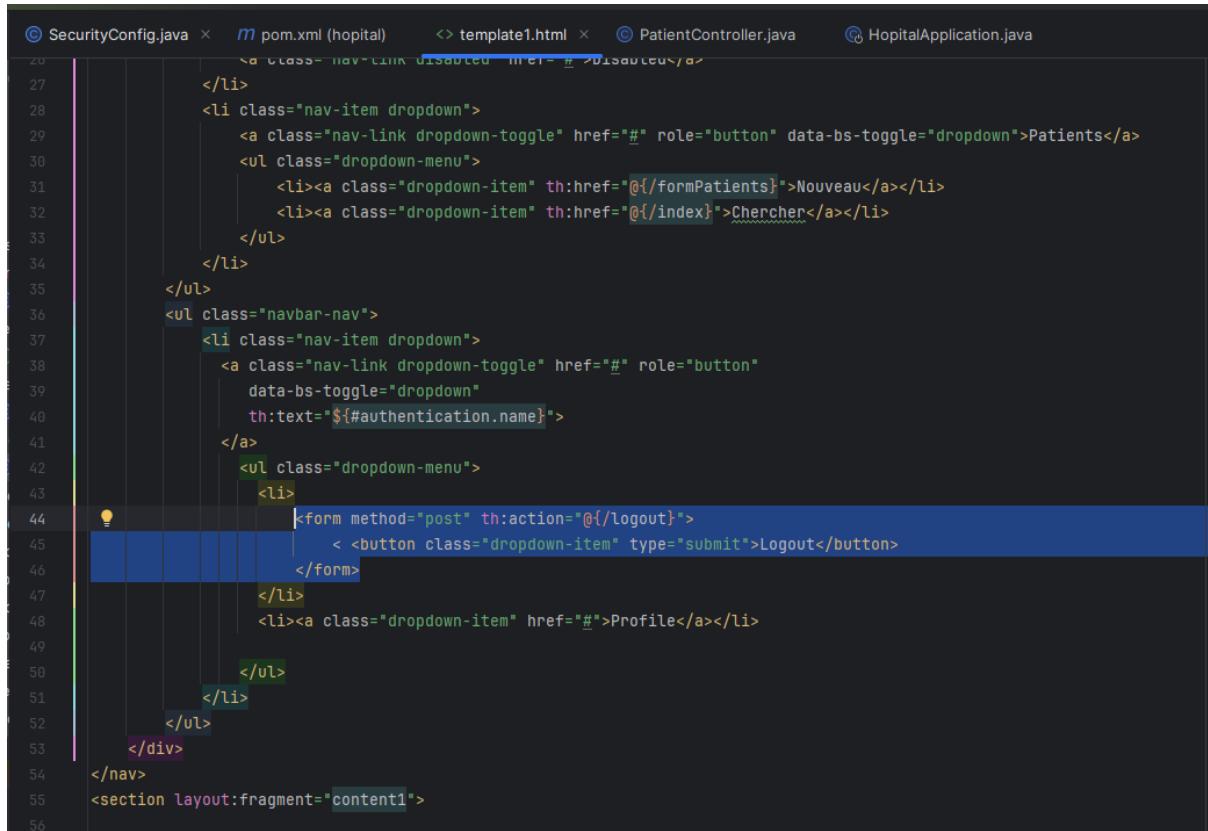
Méthode 1 :

```

20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

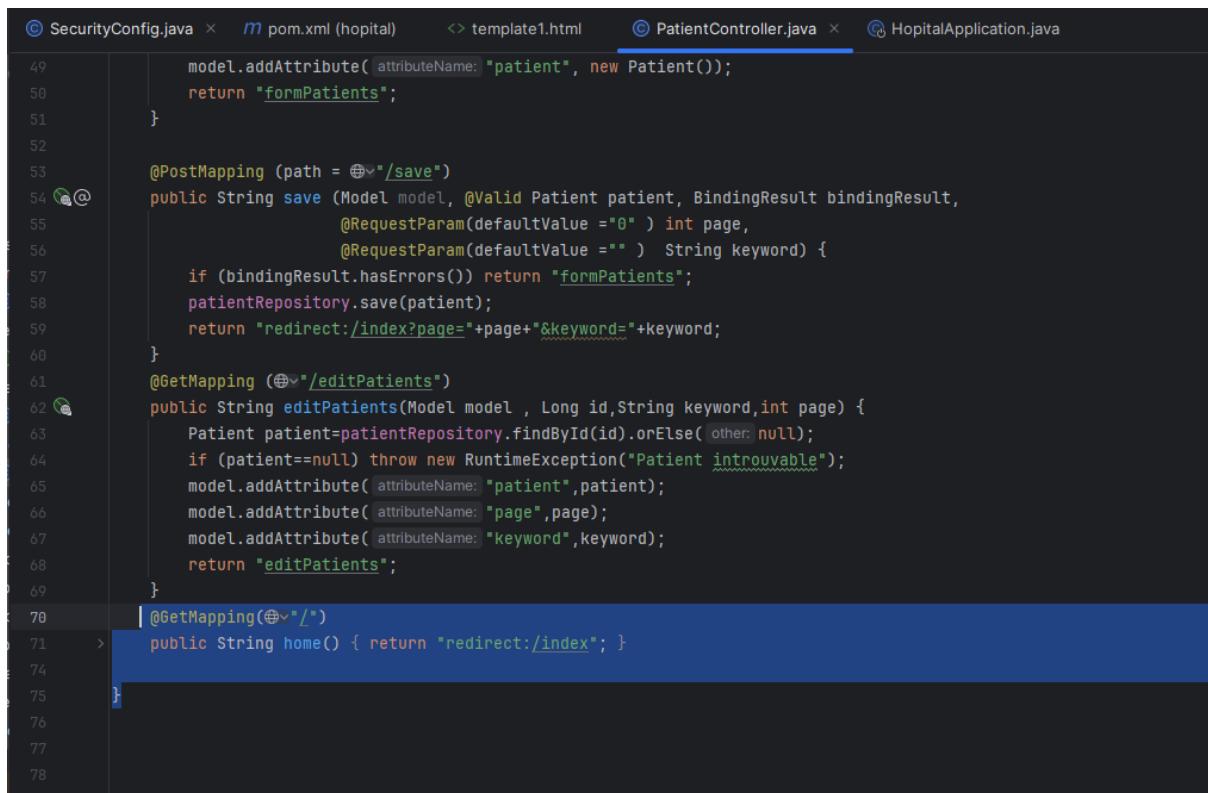
```

Méthode 2 : c'est lorsqu'on ne veut pas passer par un autre bouton logout.



The screenshot shows a code editor with several tabs: SecurityConfig.java, pom.xml (hôpital), template1.html, PatientController.java, and HopitalApplication.java. The template1.html tab is active, displaying the following HTML code:

```
<ul 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: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"
           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>
            <li><a class="dropdown-item" href="#"> Profile |</a></li>
        </ul>
    </li>
</ul>
</div>
<section layout:fragment="content1">
```



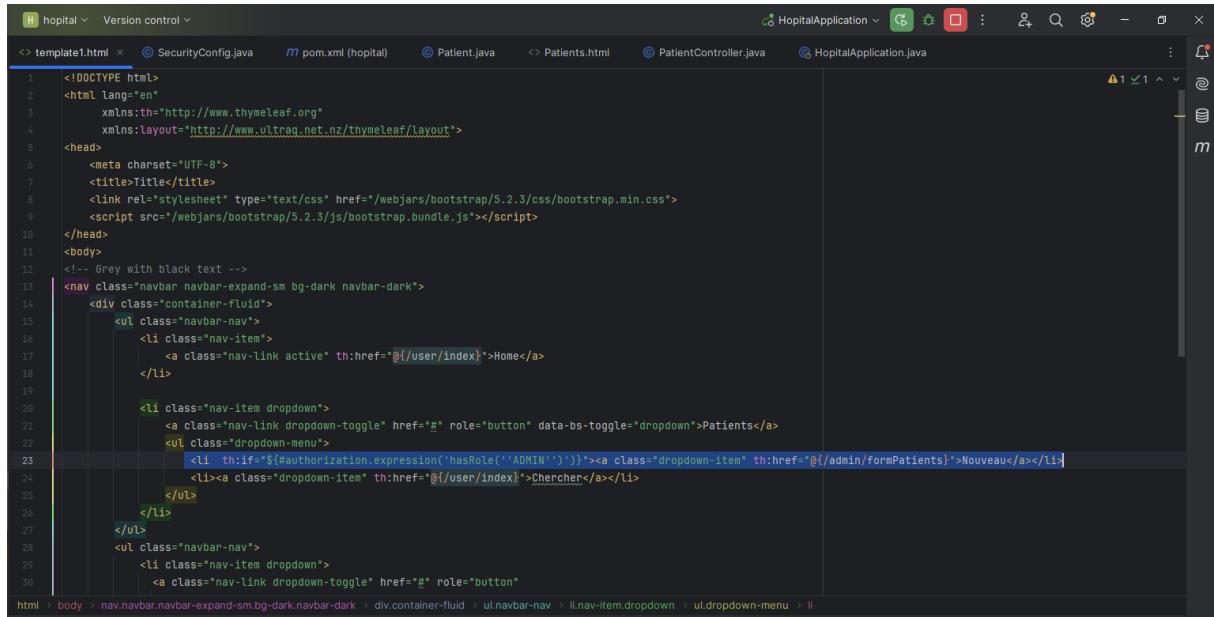
The screenshot shows a code editor with several tabs: SecurityConfig.java, pom.xml (hôpital), template1.html, PatientController.java (active), and HopitalApplication.java. The PatientController.java tab is active, displaying the following Java code:

```
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("/editPatients")
public String editPatients(Model model , Long id,String keyword,int page) {
    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 "editPatients";
}
@GetMapping("/")
public String home() { return "redirect:/index"; }
```

➤ Pour gérer Les droits accès.

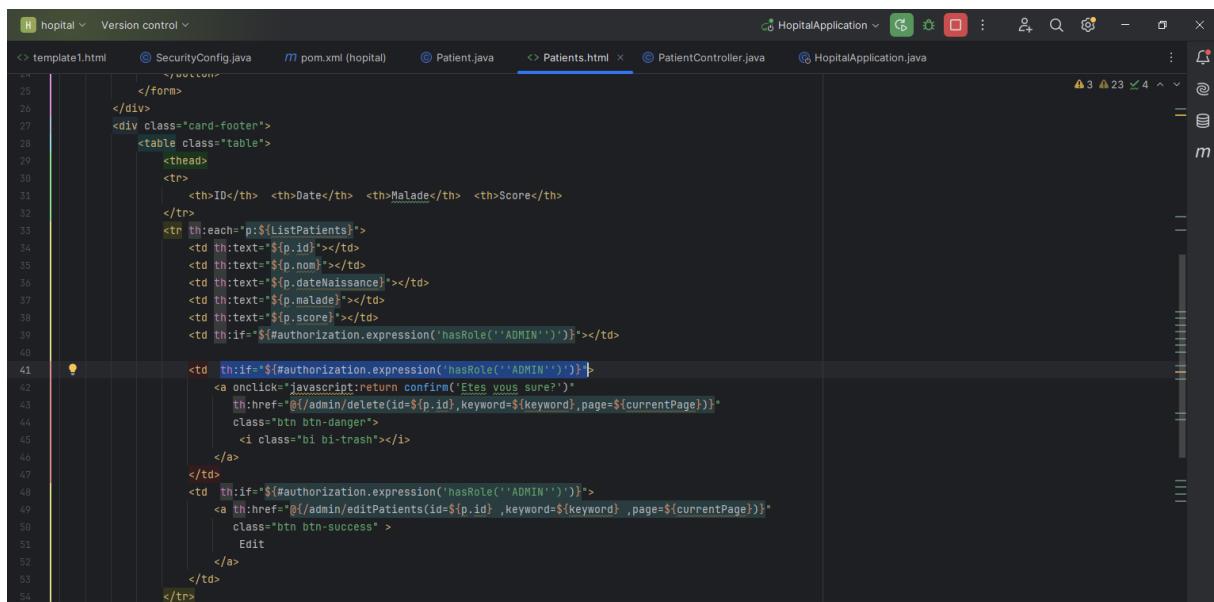
Voilà les boutons sont retirés lorsqu'on se connecte en tant que user.



A screenshot of a code editor showing the file `template1.html`. The code is a Thymeleaf template for a navigation bar. It includes imports for Thymeleaf and Bootstrap, defines a `nav` bar with a dropdown menu for 'Patients', and a separate `ul.navbar-nav` section.

```
<!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>title</title>
    <link rel="stylesheet" type="text/css" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
    <script src="/webjars/bootstrap/5.2.3/js/bootstrap.bundle.js"></script>
</head>
<body>
    <!-- Grey with black text -->
    <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 dropdown">
                    <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">Patients</a>
                    <ul class="dropdown-menu">
                        <li th:if="#{#authorization.expression('hasRole('ADMIN')')}"><a class="dropdown-item" 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" th:href="@{/user/index}">Logout</a>
                </li>
            </ul>
        </div>
    </nav>
</body>
```



A screenshot of a code editor showing the file `Patients.html`. This is a Thymeleaf template for displaying a list of patients. It features a table with columns for ID, Date, Malade, and Score. Each row contains buttons for 'Delete' and 'Edit'.

```
<div class="card-body">
    <table class="table">
        <thead>
            <tr>
                <th>ID</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="${p.nom}"></td>
                <td th:text="${p.dateNaissance}"></td>
                <td th:text="${p.malade}"></td>
                <td th:text="${p.score}"></td>
                <td th:if="#{#authorization.expression('hasRole('ADMIN')')}"></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=${currentPage}}"
                        class="btn btn-danger">
                        <i class="bi bi-trash"></i>
                    </a>
                </td>
                <td th:if="#{#authorization.expression('hasRole('ADMIN')')}">
                    <a th:href="@{/admin/editPatients(id=${p.id} ,keyword=${keyword} ,page=${currentPage})}"
                        class="btn btn-success">
                        Edit
                    </a>
                </td>
            </tr>
        </tbody>
    </table>
</div>
```

ID	Date	Malade	Score	
1	Mohamed	2024-04-26 14:34:26.0	false	134
3	hanane		true	134
4	Mohamed	2024-04-26 14:42:49.0	false	134
5	Hanane	2024-04-26 14:42:49.0	false	421

Lorsque user1 veut supprimer un patient application génère un message d'erreur comme quoi vous n'avez pas le droit de faire cette fonctionnalité.

En ajoute la configuration suivants.

```

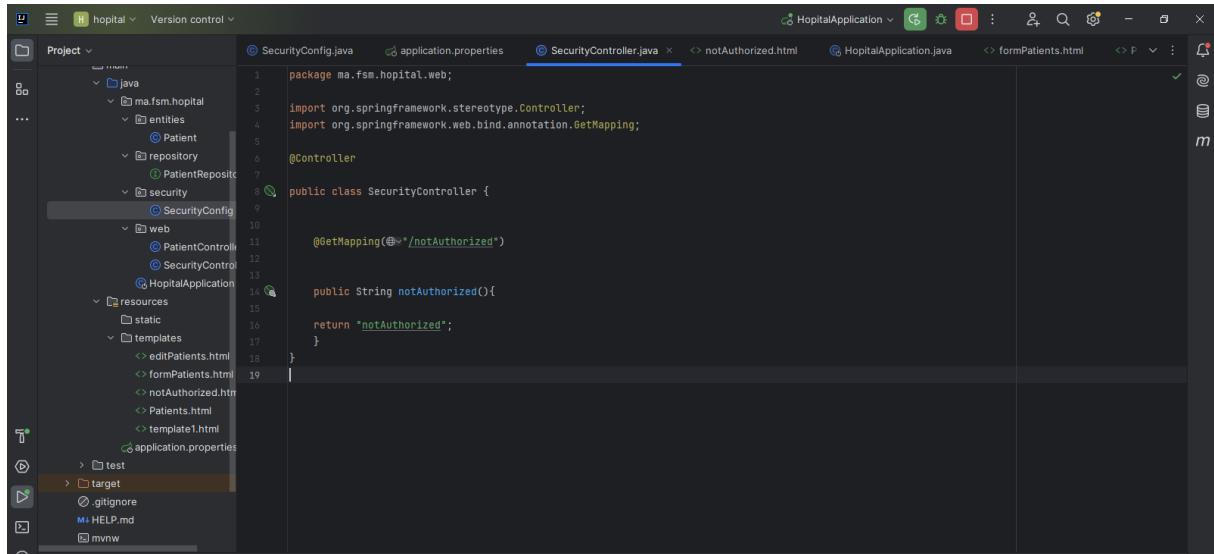
@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Autowired
    private PasswordEncoder passwordEncoder;

    @Bean
    public InMemoryUserDetailsManager inMemoryUserDetailsManager{
        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("USER", "ADMIN").build()
        );
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception{
        httpSecurity.formLogin();
        httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers(@"/deletePatient/**").hasRole("ADMIN"));
        httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers(@"/admin/**").hasRole("ADMIN"));
        httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers(@"/user/**").hasRole("USER"));
        httpSecurity.authorizeHttpRequests().anyRequest().authenticated();
        httpSecurity.exceptionHandling().accessDeniedPage( accessDeniedUrl: "/notAuthorized");
        return httpSecurity.build();
    }
}

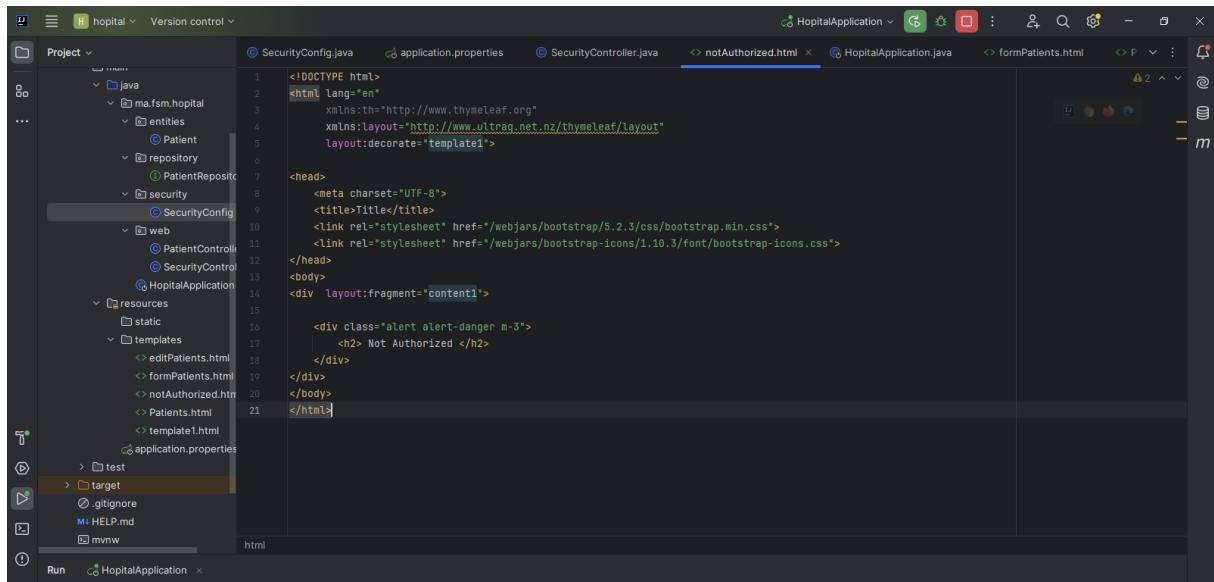
```

La Classe SecurityController.



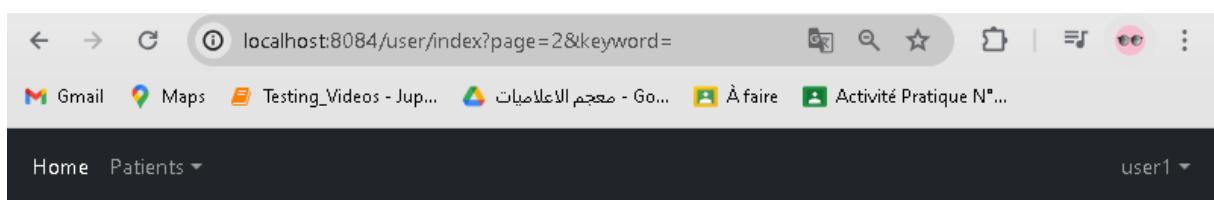
```
1 package ma.fsm.hopital.web;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.GetMapping;
5
6 @Controller
7 public class SecurityController {
8
9     @GetMapping(@"/notAuthorized")
10
11     public String notAuthorized(){
12
13         return "notAuthorized";
14     }
15 }
```

Le fichier notAuthorized.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>Title</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">
</head>
<body>
<div layout:fragment="content">
    <div class="alert alert-danger m-3">
        <h2> Not Authorized </h2>
    </div>
</div>
</body>
</html>
```

En exécute l'application le message suivant s' affiche.



Not Authorized

- Lorsque j'authentifier comme user j'ai pas le droit de supprimer ni d'éditer le patient mais je le droit de chercher patient et de faire pagination .

localhost:8084/user/index?page=4&keyword=

Gmail Maps Testing_Videos - Jup... متحف الاعلاميات - Go... À faire Activité Pratique N°...

Home Patients user2

Liste Patients

keyword:

ID	Date	Malade	Score
19	Mohamed	2024-04-26 14:55:45.0	false 134
20	Hanane	2024-04-26 14:55:45.0	false 421
21	Imane	2024-04-26 14:55:45.0	true 134
22	Mohamed	2024-04-26 14:59:40.0	false 134

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	35	
36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	
53	54	55	56	57	58	59	60	61	62	63	64	65	66				

localhost:8084/user/index?page=4&keyword=Hanane

Gmail Maps Testing_Videos - Jup... متحف الاعلاميات - Go... À faire Activité Pratique N°...

Home Patients user2

Liste Patients

keyword: Hanane

ID	Date	Malade	Score
50	Hanane	2024-04-26 15:25:06.0	false 421
53	Hanane	2024-04-26 15:28:37.0	false 421
56	Hanane	2024-04-26 15:31:19.0	false 421
59	Hanane	2024-04-26 15:35:05.0	false 421

➤ Lorsque j'authentifier comme admin j'ai le droit de supprimer et éditer et chercher faire pagination .

localhost:8084/user/index

Gmail Maps Testing_Videos - Jup... Google - مجمع الاعلاميات - Go... À faire Activité Pratique N°...

Home Patients admin

Liste Patients

keyword:

ID	Date	Malade	Score		
1	Mohamed	2024-04-26 14:34:26.0	false	134	
3	hanane		true	134	
4	Mohamed	2024-04-26 14:42:49.0	false	134	
5	Hanane	2024-04-26 14:42:49.0	false	421	

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 35
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
53 54 55 56 57 58 59 60 61 62 63 64 65 66

localhost:8084/user/index?page=5&keyword=Imane

Gmail Maps Testing_Videos - Jup... Google - مجمع الاعلاميات - Go... À faire Activité Pratique N°...

Home Patients admin

localhost:8084 indique

Etes vous sur?

OK Annuler

Liste Patients

keyword: Imane

ID	Date	Malade	Score		
69	Imane	2024-04-26 15:42:42.0	true	134	
72	Imane	2024-04-26 15:42:50.0	true	134	
75	Imane	2024-04-26 15:42:54.0	true	134	
78	Imane	2024-04-26 15:44:21.0	true	134	

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
19 20 21 22

localhost:8084/admin/editPatients?id=1&keyword=&page=1

ID: 1

Nom: Mohamed

Date Naissance: jj/mm/aaaa

Malade:

Score: 134

Save

localhost:8084/user/index?page=5&keyword=Imane

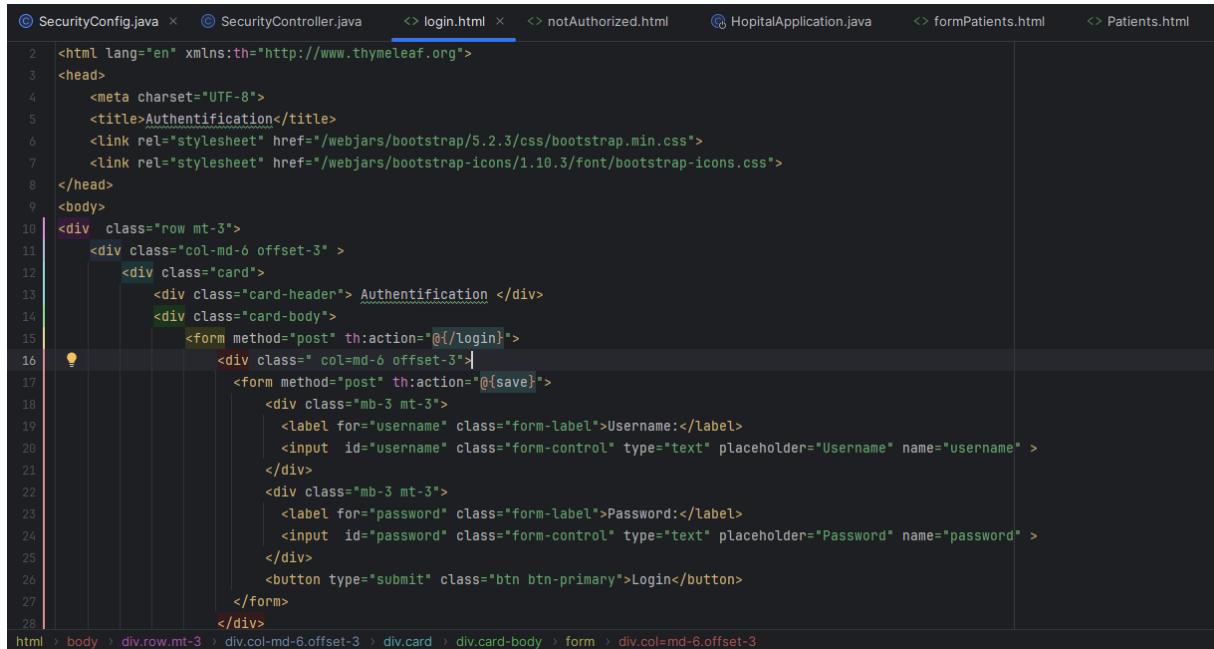
Liste Patients

keyword: Imane **Q**

ID	Date	Malade	Score		
69	Imane	2024-04-26 15:42:42.0	true	134	
72	Imane	2024-04-26 15:42:50.0	true	134	
75	Imane	2024-04-26 15:42:54.0	true	134	
78	Imane	2024-04-26 15:44:21.0	true	134	
1	2	3	4	5	
19	20	21	22	7	8
9	10	11	12	13	14
15	16	17	18		

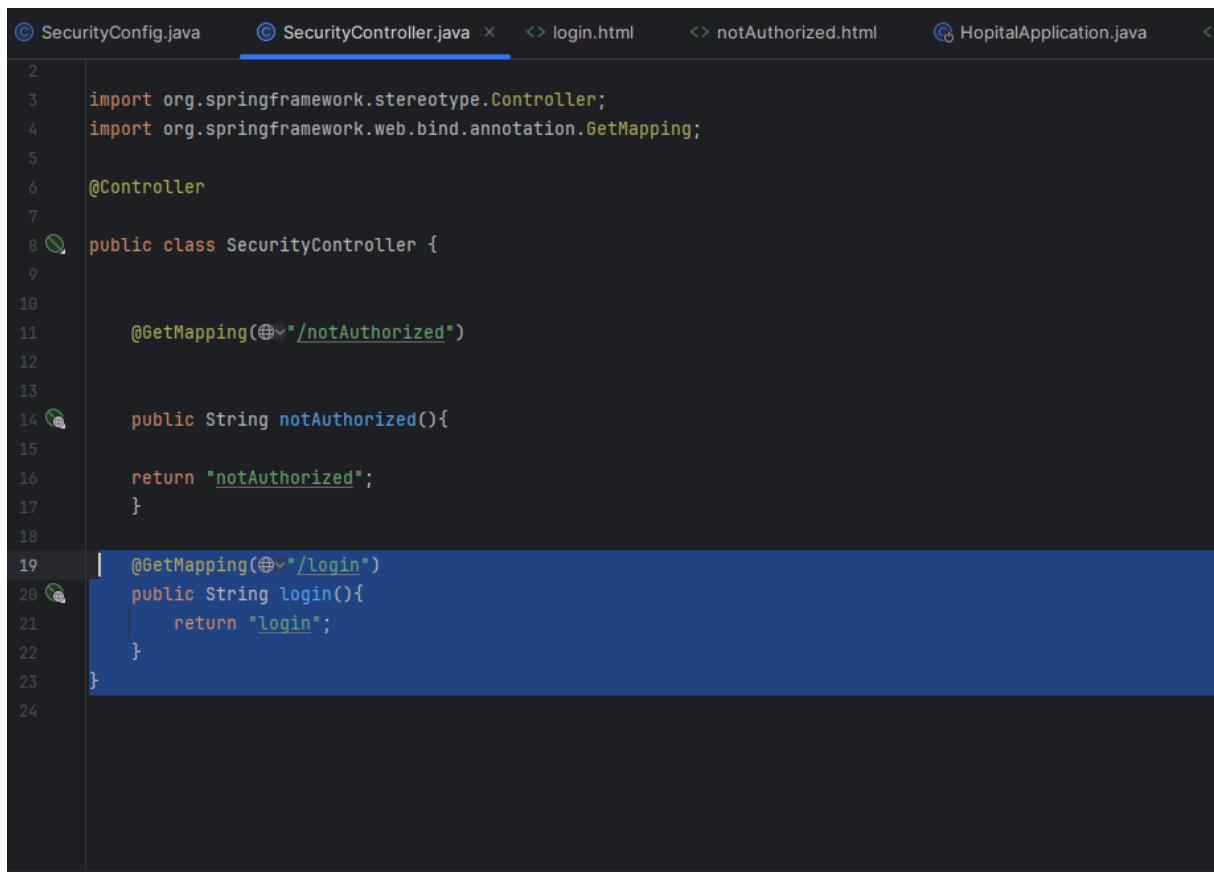
➤ Personnaliser le formulaire d'authentification.

Création du Ficher login.html :



```
2 <html lang="en" xmlns:th="http://www.thymeleaf.org">
3 <head>
4     <meta charset="UTF-8">
5     <title>Authentification</title>
6     <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
7     <link rel="stylesheet" href="/webjars/bootstrap-icons/1.10.3/font/bootstrap-icons.css">
8 </head>
9 <body>
10    <div class="row mt-3">
11        <div class="col-md-6 offset-3" >
12            <div class="card">
13                <div class="card-header"> Authentification </div>
14                <div class="card-body">
15                    <form method="post" th:action="@{/login}">
16                        <div class=" col-md-6 offset-3">
17                            <form method="post" th:action= "@{save}">
18                                <div class="mb-3 mt-3">
19                                    <label for="username" class="form-label">Username:</label>
20                                    <input id="username" class="form-control" type="text" placeholder="Username" name="username" >
21                                </div>
22                                <div class="mb-3 mt-3">
23                                    <label for="password" class="form-label">Password:</label>
24                                    <input id="password" class="form-control" type="text" placeholder="Password" name="password" >
25                                </div>
26                                <button type="submit" class="btn btn-primary">Login</button>
27                            </form>
28                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
```

La classe SecurityController :



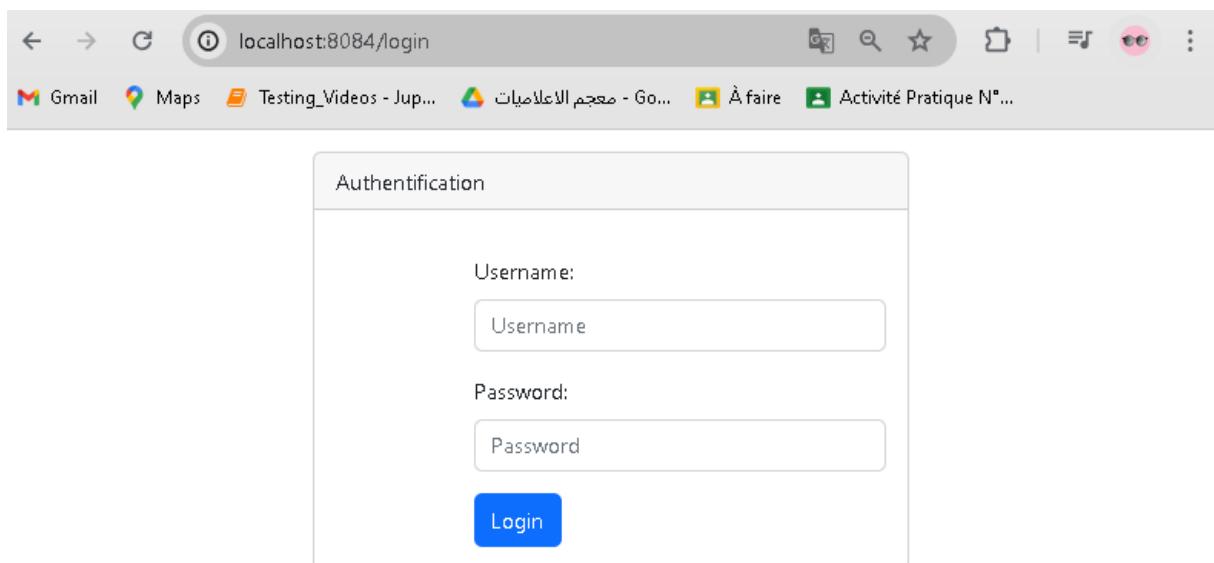
```
2
3     import org.springframework.stereotype.Controller;
4     import org.springframework.web.bind.annotation.GetMapping;
5
6     @Controller
7
8     public class SecurityController {
9
10         @GetMapping("@{/notAuthorized}")
11
12
13         public String notAuthorized(){
14
15             return "notAuthorized";
16         }
17
18         @GetMapping("@{/login}")
19         public String login(){
20             return "login";
21         }
22
23     }
24
```

```

① SecurityConfig.java × ② SecurityController.java      <> login.html      <> notAuthorized.html      ③ HopitalApplication.java      <> formPatients.html
1/ ④ public class SecurityConfig {
18     @Autowired
19     private PasswordEncoder passwordEncoder;
20
21     ⑤ @Bean
22     public InMemoryUserDetailsManager inMemoryUserDetailsManager(){
23         return new InMemoryUserDetailsManager(
24             User.withUsername("user1").password(passwordEncoder.encode("1234")).roles("USER").build(),
25             User.withUsername("user2").password(passwordEncoder.encode("1234")).roles("USER").build(),
26             User.withUsername("admin").password(passwordEncoder.encode("1234")).roles("USER", "ADMIN").build()
27         );
28     }
29
30     ⑥ @Bean
31     public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception{
32         httpSecurity.formLogin().loginPage("/login").permitAll();
33         httpSecurity.authorizeHttpRequests().requestMatchers(⑦ "/webjars/**").permitAll();
34         httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers(⑧ "/deletePatient/**").hasRole("ADMIN"));
35         httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers(⑨ "/admin/**").hasRole("ADMIN"));
36         httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers(⑩ "/user/**").hasRole("USER"));
37         httpSecurity.authorizeHttpRequests().anyRequest().authenticated();
38         httpSecurity.exceptionHandling().accessDeniedPage(⑪ "/notAuthorized");
39         return httpSecurity.build();
40     }
41 }
42
43
44

```

Notre nouvelle formulaire est le suivant :



Si on veut ajouter une case à cocher « Remember me » :

Ajoutons ces lignes au Ficher login.html :

```

15     <form method="post" th:action="@{/login}">
16         <div class="col-md-6 offset-3">
17             <form method="post" th:action="@{save}">
18                 <div class="mb-3 mt-3">
19                     <label for="username" class="form-label">Username:</label>
20                     <input id="username" class="form-control" type="text" placeholder="Username" name="username" >
21                 </div>
22                 <div class="mb-3 mt-3">
23                     <label for="password" class="form-label">Password:</label>
24                     <input id="password" class="form-control" type="text" placeholder="Password" name="password" >
25                 </div>
26                 <div class="form-check mb-3">
27                     <label class="form-check-label">
28                         <input class="form-check-input" type="checkbox" name="remember-me"> Remember me
29                     </label>
30                 </div>
31                 <button type="submit" class="btn btn-primary">Login</button>
32             </form>
33         </div>
34     </div>
35 </div>
36 </body>
37 </html>

```

html > body > div.row.mt-3 > div.col-md-6.offset-3 > div.card > div.card-body > form > div.col-md-6.offset-3 > form > div.form-check.mb-3

```

17     public class SecurityConfig {
18         @Autowired
19         private PasswordEncoder passwordEncoder;
20
21         @Bean
22         public InMemoryUserDetailsManager inMemoryUserDetailsManager() {
23             return new InMemoryUserDetailsManager(
24                 User.withUsername("user1").password(passwordEncoder.encode("1234")).roles("USER").build(),
25                 User.withUsername("user2").password(passwordEncoder.encode("1234")).roles("USER").build(),
26                 User.withUsername("admin").password(passwordEncoder.encode("1234")).roles("USER", "ADMIN").build()
27             );
28         }
29
30         @Bean
31         public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception{
32             httpSecurity.formLogin().LoginPage("/login").permitAll();
33             httpSecurity.rememberMe();
34             httpSecurity.authorizeHttpRequests().requestMatchers(/**).permitAll();
35             httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers("/deletePatient**").hasRole("ADMIN"));
36             httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers("/admin/**").hasRole("ADMIN"));
37             httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers("/user**").hasRole("USER"));
38             httpSecurity.authorizeHttpRequests().anyRequest().authenticated();
39             httpSecurity.exceptionHandling().accessDeniedPage("/notAuthorized");
40             return httpSecurity.build();
41         }
42     }
43 }
44

```

Voilà les résultats :

localhost:8084/login

Authentification

Username:

admin

Password:

1234

Remember me

Login

- On va stocker les données dans la base de données au lieu d'une mémoire .

pour protéger les ressources soit par annotation `@EnableMethodSecurity` :

```
1 package ma.fsm.hopital.security;
2
3 > import ...
4
5
6 @Configuration
7 @EnableWebSecurity
8 @EnableMethodSecurity(prePostEnabled = true)
9 public class SecurityConfig {
10     @Autowired
11     private PasswordEncoder passwordEncoder;
12
13     @Bean
14     public InMemoryUserDetailsManager inMemoryUserDetailsManager(){
15         return new InMemoryUserDetailsManager(
16             User.withUsername("user1").password(passwordEncoder.encode("1234")).roles("USER").build(),
17             User.withUsername("user2").password(passwordEncoder.encode("1234")).roles("USER").build(),
18             User.withUsername("admin").password(passwordEncoder.encode("1234")).roles("USER", "ADMIN").build()
19         );
20     }
21
22     @Bean
23     public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception{
24         httpSecurity.formLogin().loginPage("/login").permitAll();
25         httpSecurity.rememberMe();
26         httpSecurity.authorizeHttpRequests().requestMatchers("/webjars/**", "/h2-console/**").permitAll();
27         httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers("/deletePatient/**").hasRole("ADMIN"));
28         httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers("/admin/**").hasRole("ADMIN"));
29         httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers("/user/**").hasRole("USER"));
30         //httpSecurity.authorizeHttpRequests().anyRequest().authenticated();
31         //httpSecurity.exceptionHandling().accessDeniedPage("/notAuthorized");
32         return httpSecurity.build();
33     }
34
35     @...
36 }
37
38
39
40
41
42
43 }
```

Lombok requires enabled annotation processing

Soit on utilise l'annotation `@PreAuthorize`:

```
31     model.addAttribute("keyword", keyword);
32     return "patients";
33 }
34
35 @GetMapping("/admin/delete")
36 @PreAuthorize("hasRole('ROLE_ADMIN')")
37 public String delete(Long id, String keyword, int page) {
38     patientRepository.deleteById(id);
39     return "redirect:/user/index?page=" + page + "&keyword=" + keyword;
40 }
41
42
43 @GetMapping("/admin/formPatients")
44 @PreAuthorize("hasRole('ROLE_ADMIN')")
45 public String formPatients(Model model) {
46     model.addAttribute("patient", new Patient());
47     return "formPatients";
48 }
49
50
51 @PostMapping (path = @"/admin/save")
52 @PreAuthorize("hasRole('ROLE_ADMIN')")
53 public String save (Model model, @Valid Patient patient, BindingResult bindingResult,
54 @RequestParam(defaultValue = "") int page,
55 @RequestParam(defaultValue = "") String keyword) {
56     if (bindingResult.hasErrors()) return "formPatients";
57     patientRepository.save(patient);
58     return "redirect:/user/index?page=" + page + "&keyword=" + keyword;
59 }
60
61
62 @GetMapping (@"/admin/editPatients")
63 @PreAuthorize("hasRole('ROLE_ADMIN')")
64 public String editPatients(Model model , Long id, String keyword,int page) {
65     Patient patient=patientRepository.findById(id).orElse( other: null);
66     if (patient==null) throw new RuntimeException("Patient introuvable");
67     model.addAttribute("patient",patient);
68     model.addAttribute("page",page);
69 }
```

❖ La stratégie JDBC :

Stocker les utilisateurs dans la base de données relationnel.

The screenshot shows the phpMyAdmin interface at localhost/phpmyadmin/. The left sidebar lists databases: Nouvelle base de données, datab, ecomm, ecom_store, fsm-hopital, hopital, Nouvelle table, patient, information_schema, khanstore, ma_bd, mysql, performance_schema, phpmyadmin, products-db, test. The main area has two tabs: "Paramètres généraux" and "Paramètres d'affichage". In "Paramètres généraux", the connection collation is set to "utf8mb4_unicode_ci". In "Paramètres d'affichage", the language is set to "Français - French" and the theme is "pmahomme".

Soit en créer les tables suivants.

The screenshot shows the phpMyAdmin interface at localhost/phpmyadmin/index.php?route=/table/sql&db=hopital&table=patient. The left sidebar shows the same database list as the previous screenshot. The main area has a SQL editor with the following code:

```
Exécuter une ou des requêtes SQL sur la table « hopital.patient » : ↴
1: create table users(username varchar_ignorecase(50) not null primary key,password varchar_ignorecase(500) not null,enabled boolean not null);
2: create table authorities (username varchar_ignorecase(50) not null,authority varchar_ignorecase(50) not null,constraint fkAuthorities_users foreign key(username) references users(username));
3: create unique index ix_auth_username on authorities (username,authority);
```

Below the code are buttons for SELECT*, SELECT, INSERT, UPDATE, DELETE, Effacer, Format, and Récupérer la requête auto-sauvegardée. At the bottom are buttons for Délimiteur, Afficher à nouveau la requête après exécution, Conserver la boîte de requêtes, ROLLBACK à la fin, and Activer la vérification des clés étrangères, followed by an "Exécuter" button.

Ou soit par IntelliJ :

```

25     public class SecurityConfig {
26         @Autowired
27         private PasswordEncoder passwordEncoder;
28
29
30         @Bean
31         public JdbcUserDetailsManager jdbcUserDetailsManager(DataSource dataSource){
32             return new JdbcUserDetailsManager(dataSource);
33         }
34
35         @Bean
36         public InMemoryUserDetailsManager inMemoryUserDetailsManager(){
37             return new InMemoryUserDetailsManager(
38                 User.withUsername("user1").password(passwordEncoder.encode("1234")).roles("USER").build(),
39                 User.withUsername("user2").password(passwordEncoder.encode("1234")).roles("USER").build(),
40                 User.withUsername("admin").password(passwordEncoder.encode("1234")).roles("USER", "ADMIN").build()
41             );
42         }
43     }
44     @
45     public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception{
46         httpSecurity.formLogin().loginPage("/login").defaultSuccessUrl("/").permitAll();
47         httpSecurity.rememberMe();
48         httpSecurity.authorizeHttpRequests().requestMatchers(/**).permitAll();
49         httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers(/**).hasRole("ADMIN"));
50         httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers(/**).hasRole("ADMIN"));
51         httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers(/**).hasRole("USER"));
52         //httpSecurity.authorizeHttpRequests().anyRequest().authenticated();
53         //httpSecurity.exceptionHandling().accessDeniedPage("/notAuthorized");
54     }

```

Lombok requires enabled annotation processing
Enable annotation processing

Création de fichier schema.sql :

```

1 create table if not exists users(username varchar_ignorecase(50) not null primary key,password varchar_ignorecase(50) not null,enabled boolean);
2 create table authorities (username varchar_ignorecase(50) not null,authority varchar_ignorecase(50) not null,constraint fkAuthorities_users foreign key(username) references users(username));
3 create unique index ix_auth_username on authorities (username,authority);

```

Demandez à JPA de ne pas régénérer la base de données .

```

spring.application.name=hospital
server.port=8084
# spring.datasource.url=jdbc:h2:mem:patients-db
# spring.h2.console.enabled=true
spring.datasource.url=jdbc:mysql://localhost:3306/fsm-hospital?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
spring.jpa.show-sql=true

```

Création de fichier data.sql : Pour alimenter la base des données avec des données.

```
insert into users( username,password(enabled) values('user1', '1234', true);
```

Donc les tableaux sont crée avec succès.

Table	Action	Lignes	Type	Interclassement	Taille	Perte
authorities	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	16,0 kio	-
patient	Parcourir Structure Rechercher Insérer Vider Supprimer	415	InnoDB	utf8mb4_general_ci	48,0 kio	-
users	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	16,0 kio	-

Création des utilisateurs suivants :

```
patientRepository.save(new Patient( id null, nom: "Mohamed", new Date(), malade: false, score: 134));
patientRepository.save(new Patient( id null, nom: "Hanane", new Date(), malade: false, score: 421));
patientRepository.save(new Patient( id null, nom: "Imane", new Date(), malade: true, score: 134));
patientRepository.save(new Patient( id null, nom: "Layla", new Date(), malade: true, score: 164));
```

```
@Bean
CommandLineRunner commandLineRunner(JdbcUserDetailsManager jdbcUserDetailsManager){
```

```
    return args -> {
```

```
        jdbcUserDetailsManager.createUser(
```

```
            User.withUsername("user11").password(passwordEncoder().encode( rawPassword: "1234")).roles("USER").build()
```

```
        );
```

```
        jdbcUserDetailsManager.createUser(
```

```
            User.withUsername("user22").password(passwordEncoder().encode( rawPassword: "1234")).roles("USER").build()
```

```
        );
```

```
        jdbcUserDetailsManager.createUser(
```

```
            User.withUsername("admin2").password(passwordEncoder().encode( rawPassword: "1234")).roles("USER", "ADMIN").build()
```

```
        );
```

```

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

```

La création des tables Sous phpMyAdmin.

Table users :

The screenshot shows the phpMyAdmin interface for the 'users' table in the 'fsm-hospital' database. The table has columns: username, password, and enabled. There are three records:

	username	password	enabled
1	admin2	\$2a\$10\$m1EQTQD52z06TB2sqWUKu.Jj.iPY62JA4f04Krk3...	1
2	user11	\$2a\$10\$TgAw4oFJX1l6M0Bjb.fe24QRXkBFOFDF0hBsaNz...	1
3	user22	\$2a\$10\$8zaAOZZ0FCss9ckR4TH1e1a84bvypSAfE4QPehR1...	1

Table authorities :

The screenshot shows the phpMyAdmin interface for the 'authorities' table in the 'fsm-hospital' database. The table has columns: username and authority. There are four records:

	username	authority
1	admin2	ROLE_ADMIN
2	admin2	ROLE_USER
3	user11	ROLE_USER
4	user22	ROLE_USER

Pour gérer les utilisateurs et leurs rôles.

The screenshot shows a Java code editor with several files open in tabs: HopitalApplication.java, SecurityController.java, SecurityConfig.java, shema.sql, and application.properties. The HopitalApplication.java file contains code for creating users 'user11', 'user22', and 'admin2'. It uses Lombok annotations (@Data, @Builder, @AllArgsConstructor) and the Spring Security JdbcUserDetailsManager. A tooltip from the IDE indicates that Lombok requires annotation processing to be enabled. The code uses PasswordEncoder to encode passwords.

```
47     }
48     @Bean
49     CommandLineRunner commandLineRunner(JdbcUserDetailsManager jdbcUserDetailsManager){
50
51         return args -> {
52
53             UserDetails u1= jdbcUserDetailsManager.loadUserByUsername("user11");
54
55             if(u1==null)
56
57                 jdbcUserDetailsManager.createUser(
58
59                     User.withUsername("user11").password(passwordEncoder().encode("1234")).roles("USER").build()
60                 );
61             UserDetails u2= jdbcUserDetailsManager.loadUserByUsername("user22");
62
63             if(u2==null)
64                 jdbcUserDetailsManager.createUser(
65
66                     User.withUsername("user22").password(passwordEncoder().encode("1234")).roles("USER").build()
67                 );
68             UserDetails u3= jdbcUserDetailsManager.loadUserByUsername("admin2");
69
70             if(u3==null)
71                 jdbcUserDetailsManager.createUser(
72
73                     User.withUsername("admin2").password(passwordEncoder().encode("1234")).roles("USER","ADMIN").build()
74                 );
75
76         };
77     }
78     @Bean
79     PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

```

En exécute notre application et on se connecter avec username admin2 :

The screenshot shows a web browser window at localhost:8084/user/index. The URL bar shows the address. The page title is "Patients". The content area displays a table titled "Liste Patients" with columns: ID, Date, Malade, and Score. There are four rows of data. Each row has a "Delete" button (red with a white trash icon) and an "Edit" button (green with a white edit icon). Below the table is a large grid of numbered buttons (1-106), likely for navigating through many records. The top right corner of the browser window shows the user is logged in as "admin2".

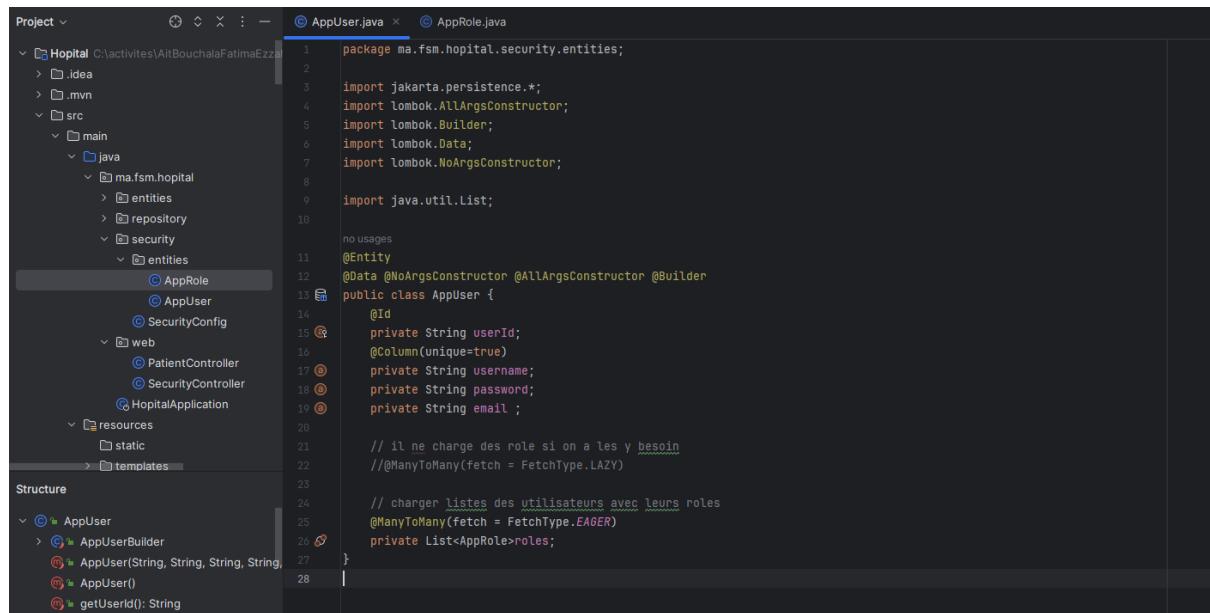
ID	Date	Malade	Score			
1	Mohamed	2024-04-26 14:34:26.0	false	134		
3	hanane		true	134		
5	Hanane	2024-04-26 14:42:49.0	false	421		
6	Imane	2024-04-26 14:42:49.0	true	134		

Avec username user1 :

❖ La stratige UserDetails Service :

Création de package entities après on créer les classes suivants :

La Classe AppUser :



```
package ma.fsm.hopital.security.entities;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

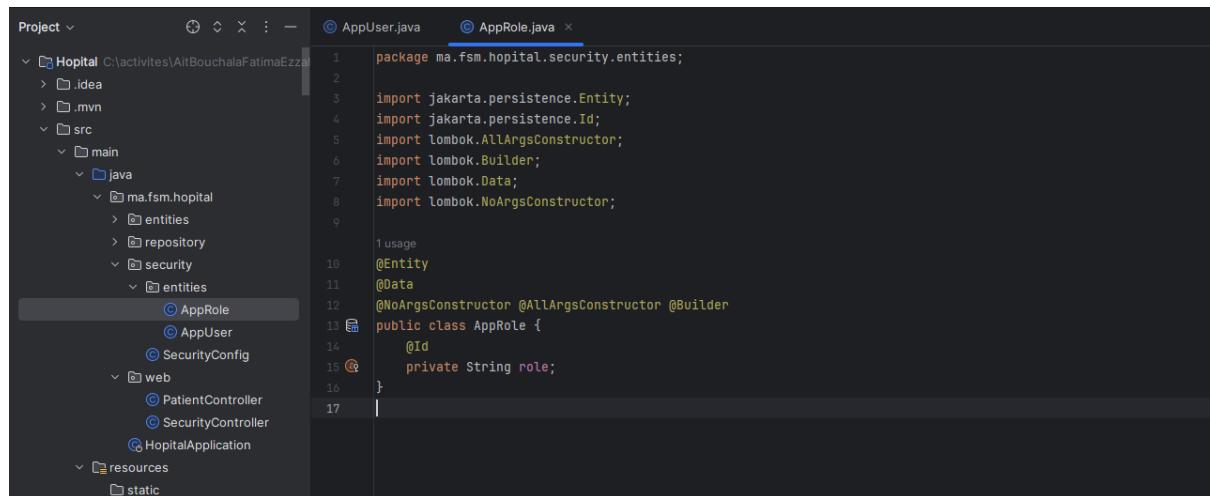
import java.util.List;

no usages
@Entity
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class AppUser {
    @Id
    private String userId;
    @Column(unique=true)
    private String username;
    private String password;
    private String email;

    // il ne charge des role si on a les y besoin
    // @ManyToMany(fetch = FetchType.LAZY)

    // charger listes des utilisateurs avec leurs roles
    // @ManyToMany(fetch = FetchType.EAGER)
    private List<AppRole> roles;
}
```

La Classe AppRole :



```
package ma.fsm.hopital.security.entities;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

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

Création de package repo avec les deux interfaces suivants :

Interface AppUserRepository qui permet de gérer les utilisateurs.

```
1 package ma.fsm.hopital.security.repo;
2
3 import ma.fsm.hopital.security.entities.AppUser;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 no usages
7 public interface AppUserRepository extends JpaRepository<AppUser, String> {
8     AppUser findByUsername(String username);
9
10 }
11
```

Interface AppRoleRepository qui gérer de créer les roles.

```
1 package ma.fsm.hopital.security.repo;
2
3 import ma.fsm.hopital.security.entities.AppRole;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 no usages
7 public interface AppRoleRepository extends JpaRepository<AppRole, String> {
8
9 }
```

La Création du couche Service avec une interface AccountService.

The screenshot shows the IntelliJ IDEA IDE interface. On the left, the project structure is visible, showing packages like 'ma.fsm.hopital' containing 'entities', 'repository', 'service', and 'web' components. The 'service' package contains the 'AccountService' interface and its implementation 'AccountServiceImpl'. The 'AccountService.java' file is currently open in the editor, displaying the following code:

```
1 package ma.fsm.hopital.security.service;
2
3 import ma.fsm.hopital.security.entities.AppRole;
4 import ma.fsm.hopital.security.entities.AppUser;
5
6 1 usage 1 implementation
7 public interface AccountService {
8     no usages 1 implementation
9     AppUser addNewUser(String username, String password, String email, String confirmPassword);
10    no usages 1 implementation
11    AppRole addNewRole(String role);
12    no usages 1 implementation
13    void addRoleToUser(String username, String role);
14    no usages 1 implementation
15    void removeRoleFromUser(String username, String role);
16    no usages
17     AppUser loadUserByUsername(String username);
18 }
```

Et L'interface AccountService implemente la classe AccountServiceImpl. Qui definie les méthodes suivantes.

```
Project ▾
```

```
src
  main
    java
      ma.fsm.hopital
        entities
        repository
        security
          repo
            AppRoleRepository
            AppUserRepository
        service
          AccountService
          AccountServiceImpl
          SecurityConfig
      web
        PatientController
        SecurityController
        HopitalApplication
      resources
        static
        templates
        application.properties
        schema.sql
    test
  target
  .gitignore
```

```
AccountService.java
```

```
AccountServiceImpl.java
```

```
1 package ma.fsm.hopital.security.service;
2 import ...
3 @Service
4 @Transactional //gener les transaction de tout les methodes
5 @AllArgsConstructor
6
7 public class AccountServiceImpl implements AccountService {
8     private AppUserRepository appUserRepository;
9     private AppRoleRepository appRoleRepository;
10    private PasswordEncoder passwordEncoder;
11
12    no usages
13
14    @Override
15    public AppUser addNewUser(String username, String password, String email, String confirmPassword) {
16        AppUser appUser=appUserRepository.findByUsername(username);
17        if (appUser!=null) throw new RuntimeException("this user already exist");
18        if (!password.equals(confirmPassword))throw new RuntimeException("Passwords not match");
19        appUser=AppUser.builder()
20            .userId(UUID.randomUUID().toString())
21            .username(username)
22            .password(passwordEncoder.encode(password))
23            .email(email)
24            .build();
25        AppUser savedAppUser = appUserRepository.save(appUser);
26        return savedAppUser;
27    }
28
29    no usages
30
31    @Override
32    public AppRole addNewRole(String role) {
33        AppRole appRole=appRoleRepository.findById(role).orElse(null);
34        if (appRole!=null)throw new RuntimeException("this role already exist");
35        appRole=AppRole.builder()
36            .role(role)
37            .build();
38    }
39
40    no usages
41
42    @Override
43    public void addRoleToUser(String username, String role) {
44        AppUser appUser=appUserRepository.findByUsername(username);
45        AppRole appRole=appRoleRepository.findById(role).get();
46        appUser.getRoles().add(appRole);
47    }
48
49    no usages
50
51    @Override
52    public void removeRoleFromUser(String username, String role) {
53        AppUser appUser=appUserRepository.findByUsername(username);
54        AppRole appRole=appRoleRepository.findById(role).get();
55        appUser.getRoles().remove(appRole);
56    }
57
58    no usages
59
60    @Override
61    public AppUser loadUserByUsername(String username) {
62        return appUserRepository.findByUsername(username);
63    }
64}
```

```
Project ▾
```

```
src
  main
    java
      ma.fsm.hopital
        entities
        repository
        security
          repo
            AppRoleRepository
            AppUserRepository
        service
          AccountService
          AccountServiceImpl
          SecurityConfig
      web
        PatientController
        SecurityController
        HopitalApplication
      resources
        static
        templates
        application.properties
        schema.sql
    test
```

```
AccountService.java
```

```
AccountServiceImpl.java
```

```
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
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
```

On exécute notre application en ajoutons ces lignes :

```

Project ▾
  AccountServiceImpl
  SecurityConfig
  web
    PatientController
    SecurityController
  HopitalApplication
    resources
      static
        templates
        application.properties
        schema.sql
    test
    target
    .gitignore
    HELP.md
    mvnw
    mvnw.cmd
    pom.xml
  External Libraries
    > C:\Program Files\Java\jdk-17.0.1
    Maven: ch.qos.logback:logback-classic:1.4.14
    Maven: ch.qos.logback:logback-core:1.4.14
Run HopitalApplication
Console Actuator
2024-05-11T17:25:44.970+01:00 INFO 14080 --- [restartedMain] ma.fsm.hopital.HopitalApplication : Started HopitalApplication in 14.363 seconds (process running for 14.363)
2024-05-11T17:27:46.882+01:00 INFO 14080 --- [nio-8084-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2024-05-11T17:27:46.883+01:00 INFO 14080 --- [nio-8084-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2024-05-11T17:27:46.885+01:00 INFO 14080 --- [nio-8084-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms
2024-05-11T17:27:47.287+01:00 WARN 14080 --- [nio-8084-exec-1] o.a.c.util.SessionIdGeneratorBase : Creation of SecureRandom took 47 ms
  
```

Sous phpMyAdmin on aura les tables suivants :

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1 - Base de données: fsm-hospital
- Structure:** Shows the database structure with tables: app_role, app_user, app_user_roles, authorities, patient, users.
- SQL:** Contains a query: `SELECT * FROM `app_role``.
- Rechercher:** Search bar.
- Opérations:** Operations menu.
- Priviléges:** Privileges menu.
- Procédures stockées:** Stored procedures menu.
- Événements:** Events menu.
- Déclencheurs:** Triggers menu.
- Concepteur:** Designer menu.

La Table app_role : contient les roles .

The screenshot shows the phpMyAdmin interface displaying the contents of the app_role table:

Table	Action	Lignes	Type	Interclassement	Taille	Perte
app_role	Parcourir Structure Rechercher Insérer Vider Supprimer	2	InnoDB	utf8mb4_general_ci	16,0 kio	-
app_user	Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8mb4_general_ci	32,0 kio	-
app_user_roles	Parcourir Structure Rechercher Insérer Vider Supprimer	4	InnoDB	utf8mb4_general_ci	48,0 kio	-
authorities	Parcourir Structure Rechercher Insérer Vider Supprimer	4	InnoDB	utf8mb4_general_ci	16,0 kio	-
patient	Parcourir Structure Rechercher Insérer Vider Supprimer	456	InnoDB	utf8mb4_general_ci	48,0 kio	-
users	Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	utf8mb4_general_ci	16,0 kio	-
6 tables	Somme	472	InnoDB	utf8mb4_general_ci	176,0 kio	0 o

Below the table, there is a "Nouvelle table" (New table) form with fields: Nom: and Nombre de colonnes: 4, and an "Exécuter" (Execute) button.

La Table app_user : contient les utilisateurs qui ont le droit d'accès à l'application.

user_id	email	password	username
1cc0883e-ae1f-4845-85a0-6e982ace5ffe	user2@gmail.com	\$2a\$10\$ft06Qrf95U98YlFD3A58bxqwSJyJ1SJy7ASm5GeY..	user2
2b67c330-711f-4423-bf32-30082adde874	user1@gmail.com	\$2a\$10\$mfUkRVjcoAUUDVZMgPHzuuc2UVkP1LxJIWTTBqZqr..	user1
ef7347f8-53f2-46df-af49-631432e2b087	admin@gmail.com	\$2a\$10\$3PUjDDIAzH/ZByKqVFb.HxN9vhE51SZ0E5gYdVCS...	admin

La Table app_user_roles : chaque utilisateur a son rôle.

app_user_user_id	roles_role
2b67c330-711f-4423-bf32-30082adde874	USER
1cc0883e-ae1f-4845-85a0-6e982ace5ffe	USER
ef7347f8-53f2-46df-af49-631432e2b087	USER
ef7347f8-53f2-46df-af49-631432e2b087	ADMIN

La partie la plus importante dans cette stratégie .

```

package ma.fsm.hopital.security.service;

import ma.fsm.hopital.security.entities.AppUser;
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.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;

@Service
public class UserDetailServiceImpl implements UserDetailsService {

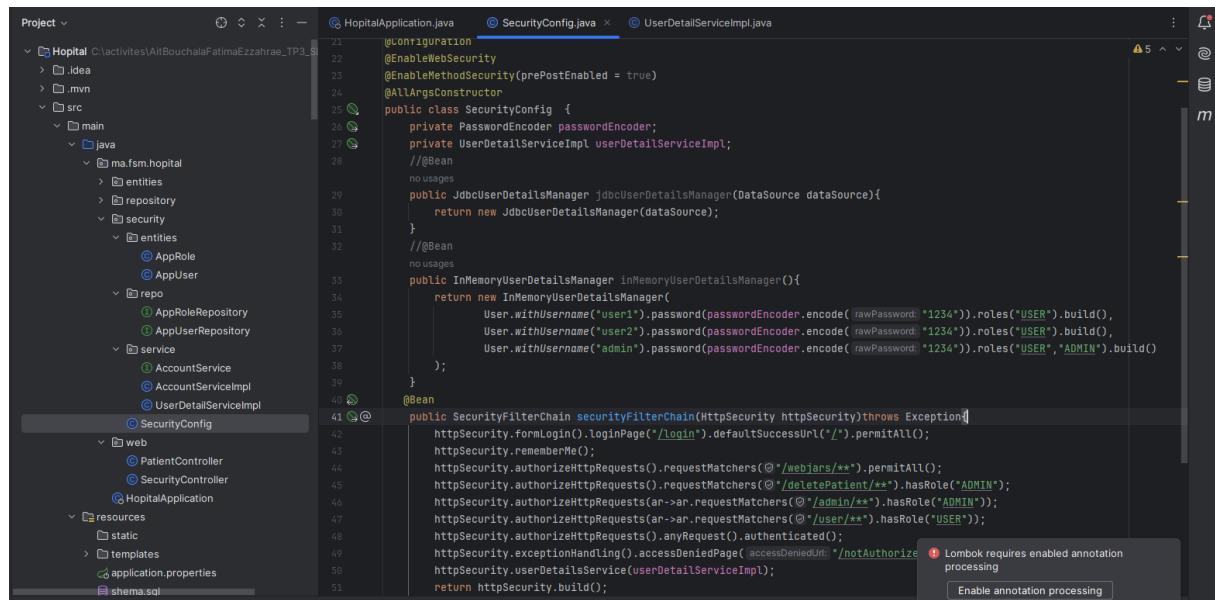
    @usage
    private AccountService accountService;

    @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;
    }
}

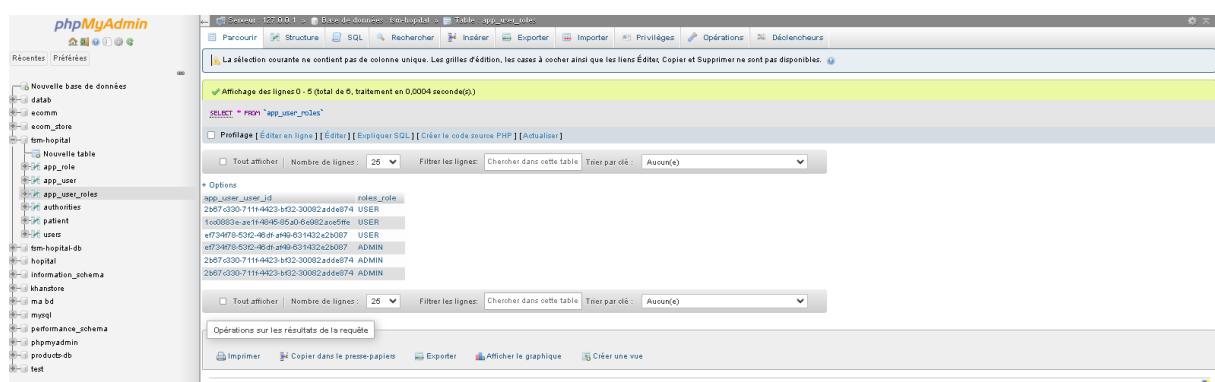
```

On va mettre en commentaire l'objet JdbcUserDetailsManager.



```
21  @Configuration
22  @EnableWebSecurity
23  @EnableMethodSecurity(prePostEnabled = true)
24  @AllArgsConstructor
25  public class SecurityConfig {
26      private PasswordEncoder passwordEncoder;
27      private UserDetailServiceImpl userDetailServiceImpl;
28
29      @Bean
30      no usages
31      public JdbcUserDetailsManager jdbcUserDetailsManager(DataSource dataSource){
32          return new JdbcUserDetailsManager(dataSource);
33      }
34
35      @Bean
36      no usages
37      public InMemoryUserDetailsManager inMemoryUserDetailsManager(){
38          return new InMemoryUserDetailsManager(
39              User.withUsername("user1").password(passwordEncoder.encode("1234")).roles("USER").build(),
40              User.withUsername("user2").password(passwordEncoder.encode("1234")).roles("USER").build(),
41              User.withUsername("admin").password(passwordEncoder.encode("1234")).roles("USER", "ADMIN").build()
42          );
43      }
44
45      @Bean
46      public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception{
47          httpSecurity.formLogin().loginPage("/login").defaultSuccessUrl("/").permitAll();
48          httpSecurity.rememberMe();
49          httpSecurity.authorizeHttpRequests().requestMatchers(/**).permitAll();
50          httpSecurity.authorizeHttpRequests().requestMatchers("/deletePatient/**).hasRole("ADMIN");
51          httpSecurity.authorizeHttpRequests().requestMatchers("/admin/**).hasRole("ADMIN");
52          httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers(/**).hasRole("USER"));
53          httpSecurity.authorizeHttpRequests(ar->ar.requestMatchers("/user/**).hasRole("USER"));
54          httpSecurity.exceptionHandling().accessDeniedPage("/accessDenied").authenticated();
55          httpSecurity.exceptionHandling().accessDeniedPage("/notAuthorized");
56          httpSecurity.userDetailsService(userDetailServiceImpl);
57          return httpSecurity.build();
58      }
59  }
```

Dans cette partie on a Affecter le rôle admin au user1 .



user_id	role
2867430-7114-4c23-b522-300022d4e074	USER
5c003ca5-e119-4b95-85a0-6a0823cc0fde	USER
e734070-532c-494f-af40-631432c2x007	USER
e734070-532c-494f-af40-631332c26087	ADMIN
2867430-7114-4c23-b522-300022d4e074	ADMIN
2867430-7114-4c23-b522-300022d4e074	ADMIN