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

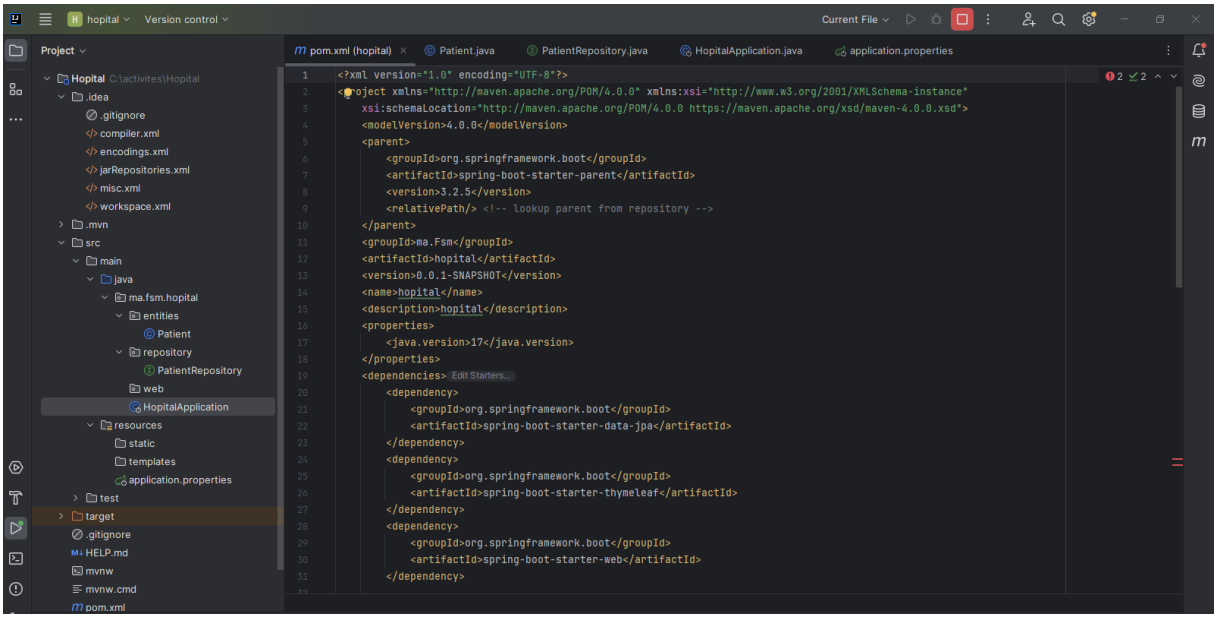
Réalisé par :

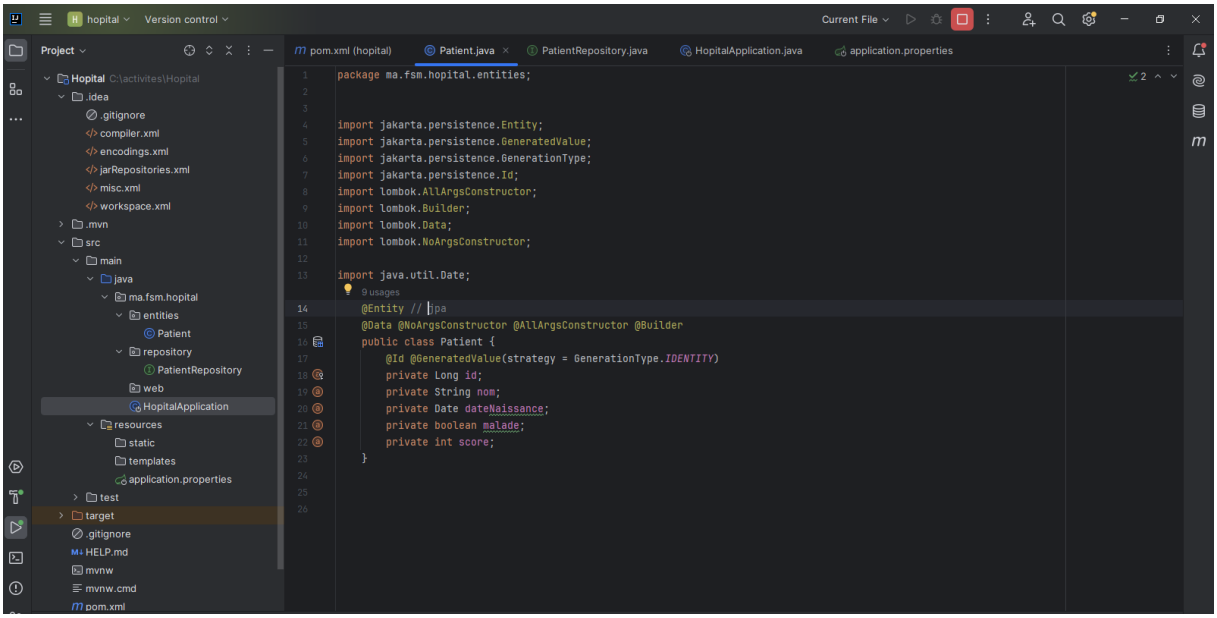
- ***AIT BOUCHALA fatime ezzahrae***

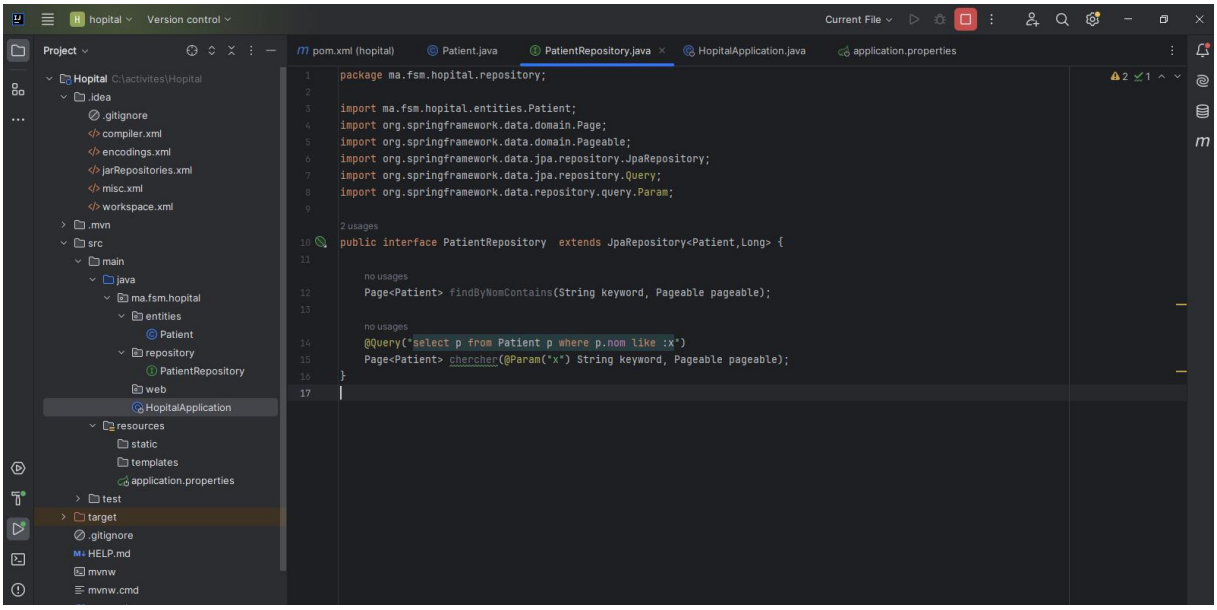
⇒ Partie 1 :

Créer une application Web JEE basée sur Spring MVC, Thylemeaf 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

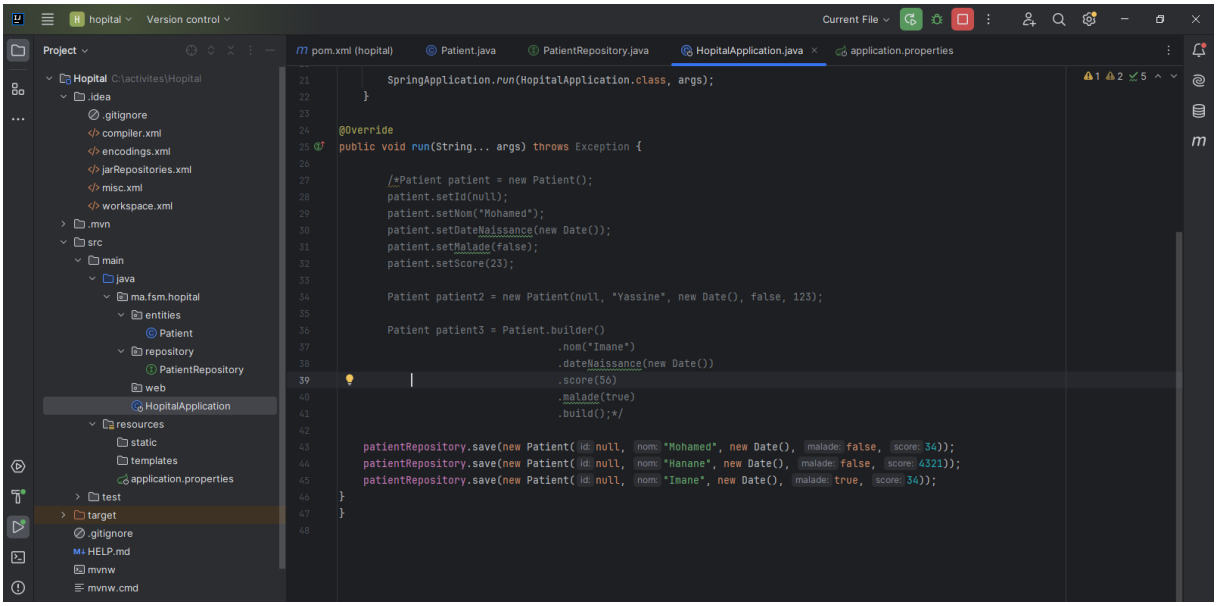


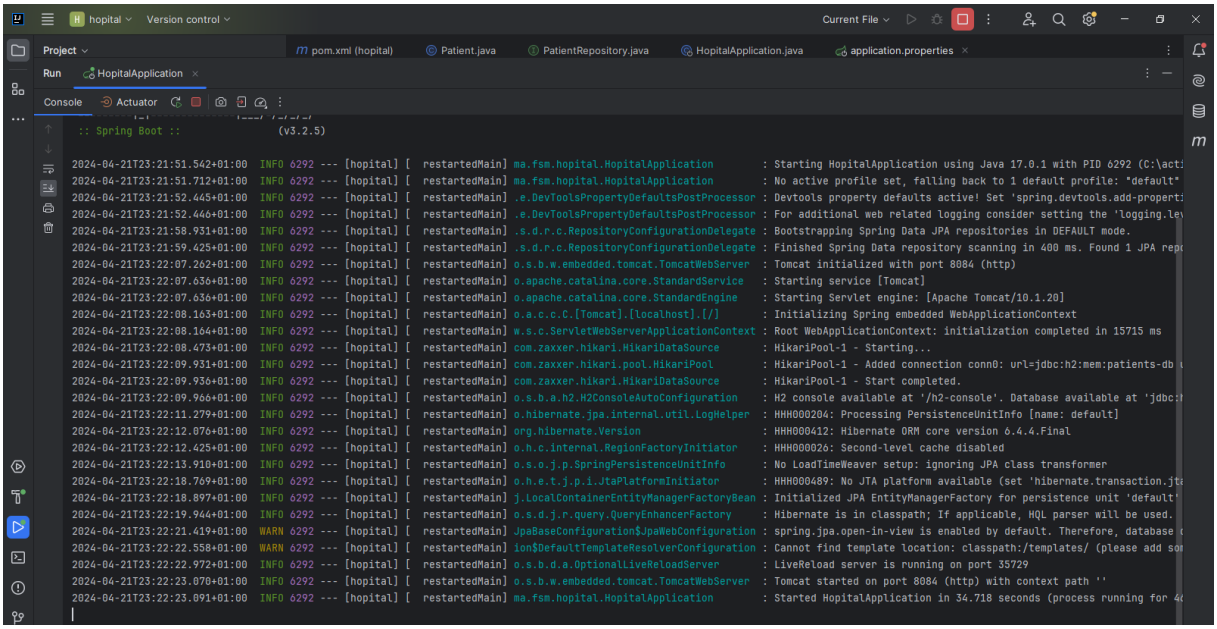


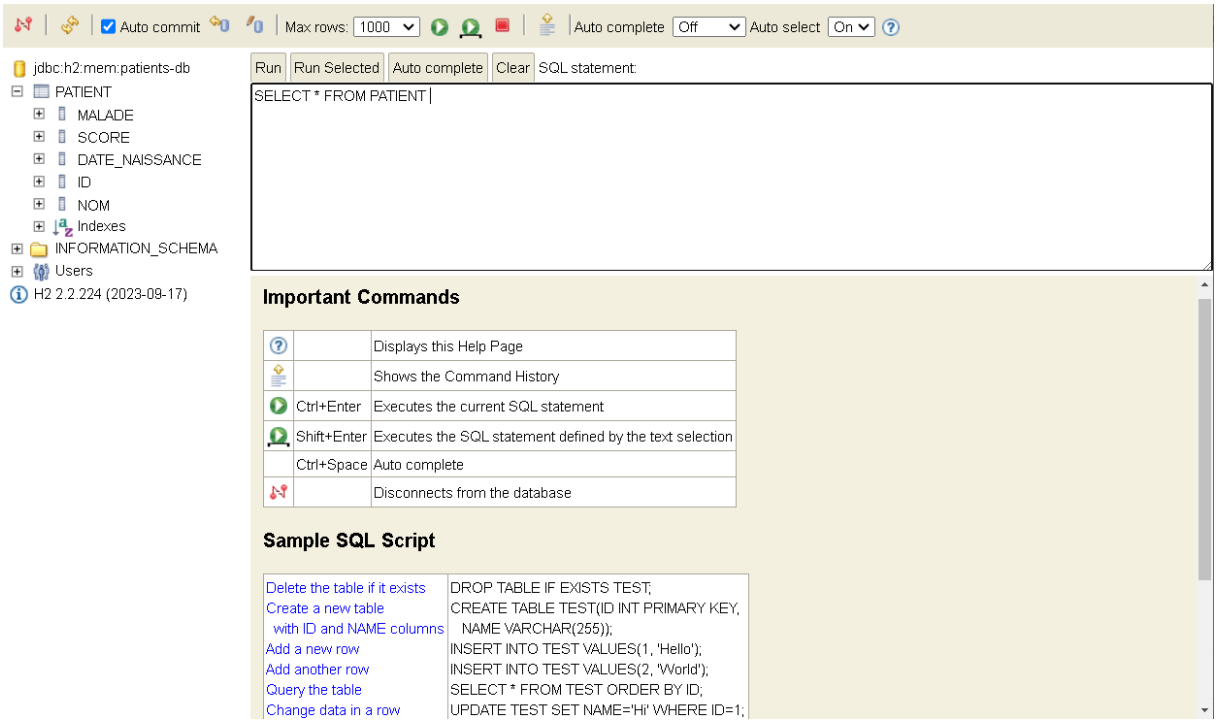


Pour se connecter au base de données h2.

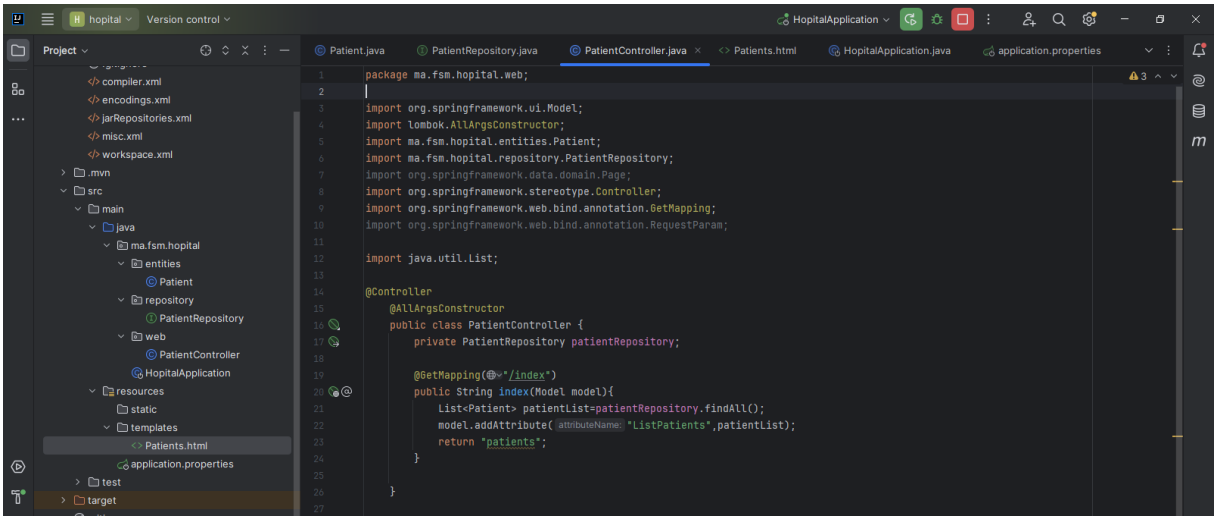
On exécute.


hopital
Version control
Current File
Project
pom.xml (hopital)
Patient.java
PatientRepository.java
HopitalApplication.java
application.properties
Hopital C:\activites\Hopital
.idea
.gitignore
compiler.xml
encodings.xml
jarRepositories.xml
misc.xml
workspace.xml
.mvn
src
main
java
ma.fsm.hopital
entities
Patient
repository
PatientRepository
web
HopitalApplication
resources
static
templates
application.properties
test
target
.gitignore
HELP.md
mvnw
mvnw.cmd
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(23);
Patient patient2 = new Patient(null, "Yassine", new Date(), false, 123);
Patient patient3 = Patient.builder()
.nom("Imane")
.dateNaissance(new Date())
.score(56)
.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));
}
}

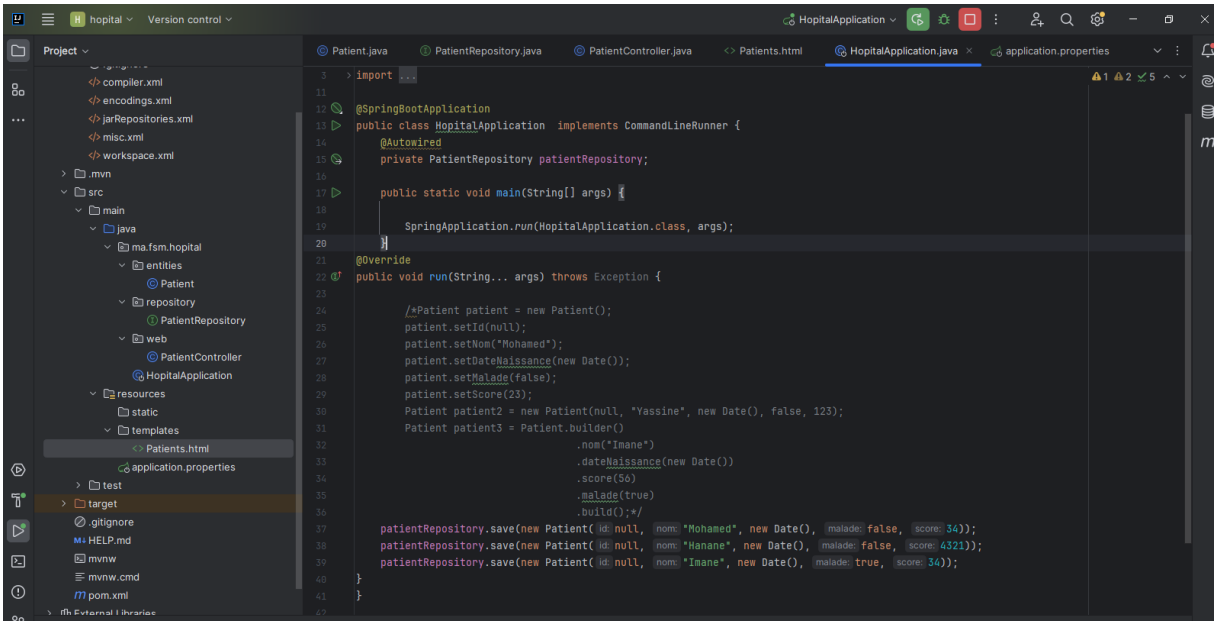

hopital
Version control
Current File
Project
pom.xml (hopital)
Patient.java
PatientRepository.java
HopitalApplication.java
application.properties
Run
HopitalApplication
Console
Actuator
:: Spring Boot :: (v3.2.5)
2024-04-21T23:21:51.542+01:00 INFO 6292 --- [hopital] [restartedMain] ma.fsm.hopital.HopitalApplication : Starting HopitalApplication using Java 17.0.1 with PID 6292 (C:\acti
2024-04-21T23:21:51.712+01:00 INFO 6292 --- [hopital] [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 --- [hopital] [restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properti
2024-04-21T23:21:52.446+01:00 INFO 6292 --- [hopital] [restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.le
2024-04-21T23:21:58.931+01:00 INFO 6292 --- [hopital] [restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-04-21T23:21:59.425+01:00 INFO 6292 --- [hopital] [restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 400 ms. Found 1 JPA rep
2024-04-21T23:22:07.262+01:00 INFO 6292 --- [hopital] [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8084 (http)
2024-04-21T23:22:07.636+01:00 INFO 6292 --- [hopital] [restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2024-04-21T23:22:07.636+01:00 INFO 6292 --- [hopital] [restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.20]
2024-04-21T23:22:08.163+01:00 INFO 6292 --- [hopital] [restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2024-04-21T23:22:08.164+01:00 INFO 6292 --- [hopital] [restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 15715 ms
2024-04-21T23:22:08.473+01:00 INFO 6292 --- [hopital] [restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2024-04-21T23:22:09.931+01:00 INFO 6292 --- [hopital] [restartedMain] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:patients-db
2024-04-21T23:22:09.936+01:00 INFO 6292 --- [hopital] [restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2024-04-21T23:22:09.966+01:00 INFO 6292 --- [hopital] [restartedMain] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:
2024-04-21T23:22:11.279+01:00 INFO 6292 --- [hopital] [restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2024-04-21T23:22:12.076+01:00 INFO 6292 --- [hopital] [restartedMain] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.4.4.Final
2024-04-21T23:22:12.425+01:00 INFO 6292 --- [hopital] [restartedMain] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2024-04-21T23:22:13.910+01:00 INFO 6292 --- [hopital] [restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2024-04-21T23:22:18.769+01:00 INFO 6292 --- [hopital] [restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jt
2024-04-21T23:22:18.897+01:00 INFO 6292 --- [hopital] [restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-04-21T23:22:19.944+01:00 INFO 6292 --- [hopital] [restartedMain] o.s.d.j.r.query.QueryEnhancerFactory : Hibernate is in classpath; If applicable, HQL parser will be used.
2024-04-21T23:22:21.419+01:00 WARN 6292 --- [hopital] [restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database
2024-04-21T23:22:22.558+01:00 WARN 6292 --- [hopital] [restartedMain] ion$DefaultTemplateResolverConfiguration : Cannot find template location: classpath:/templates/ (please add so
2024-04-21T23:22:22.972+01:00 INFO 6292 --- [hopital] [restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2024-04-21T23:22:23.070+01:00 INFO 6292 --- [hopital] [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8084 (http) with context path ''
2024-04-21T23:22:23.091+01:00 INFO 6292 --- [hopital] [restartedMain] ma.fsm.hopital.HopitalApplication : Started HopitalApplication in 34.718 seconds (process running for 4



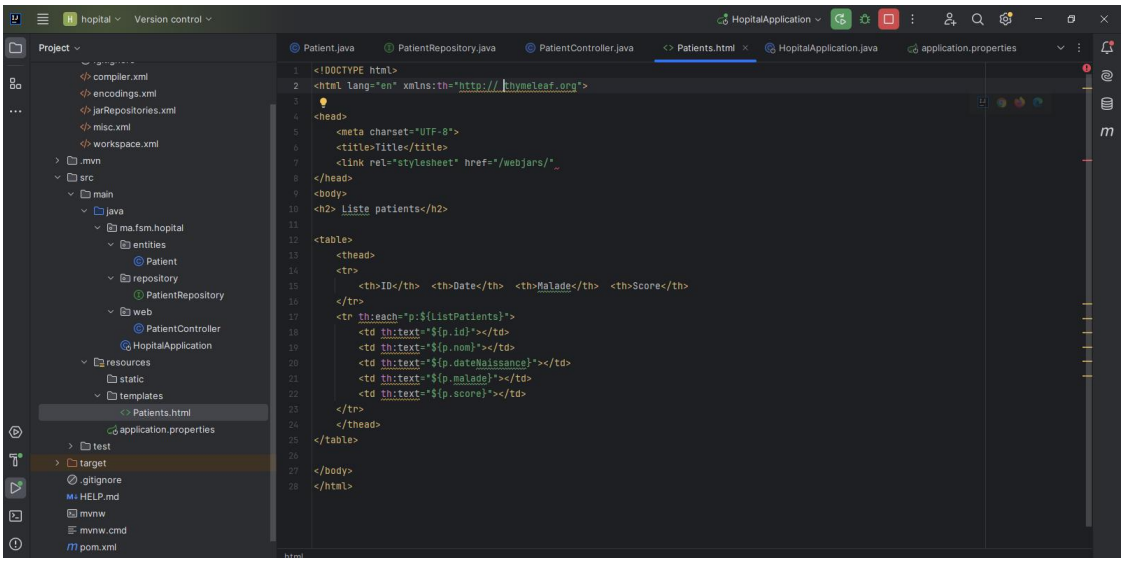
- La partie web :



Si on exécute ce code :



Le Fichier Html :

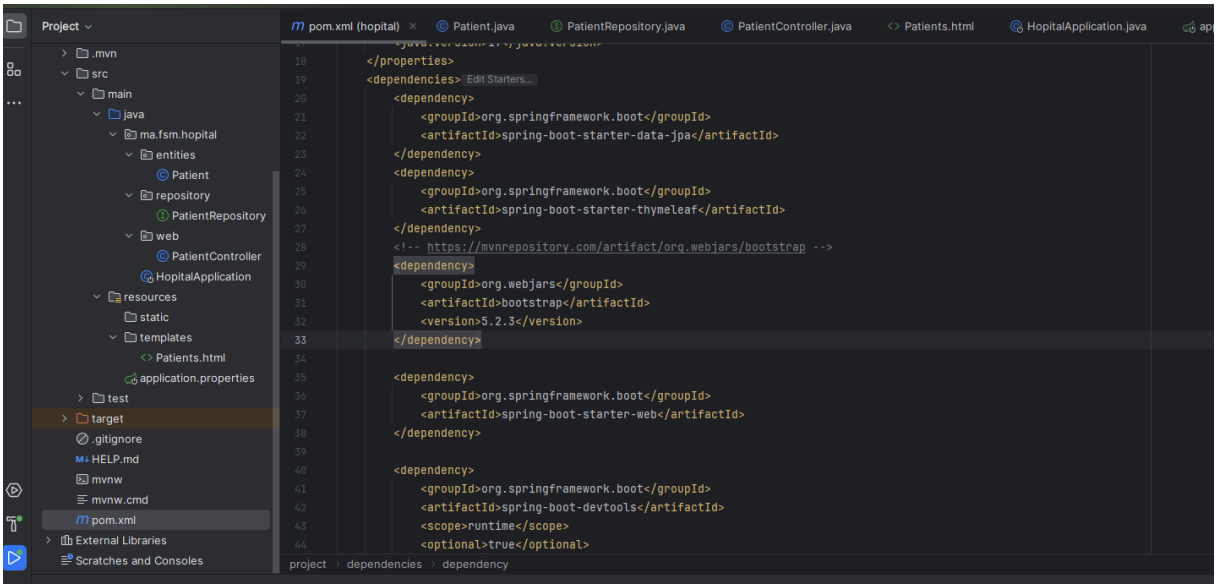


La page html affiche la table suivants :

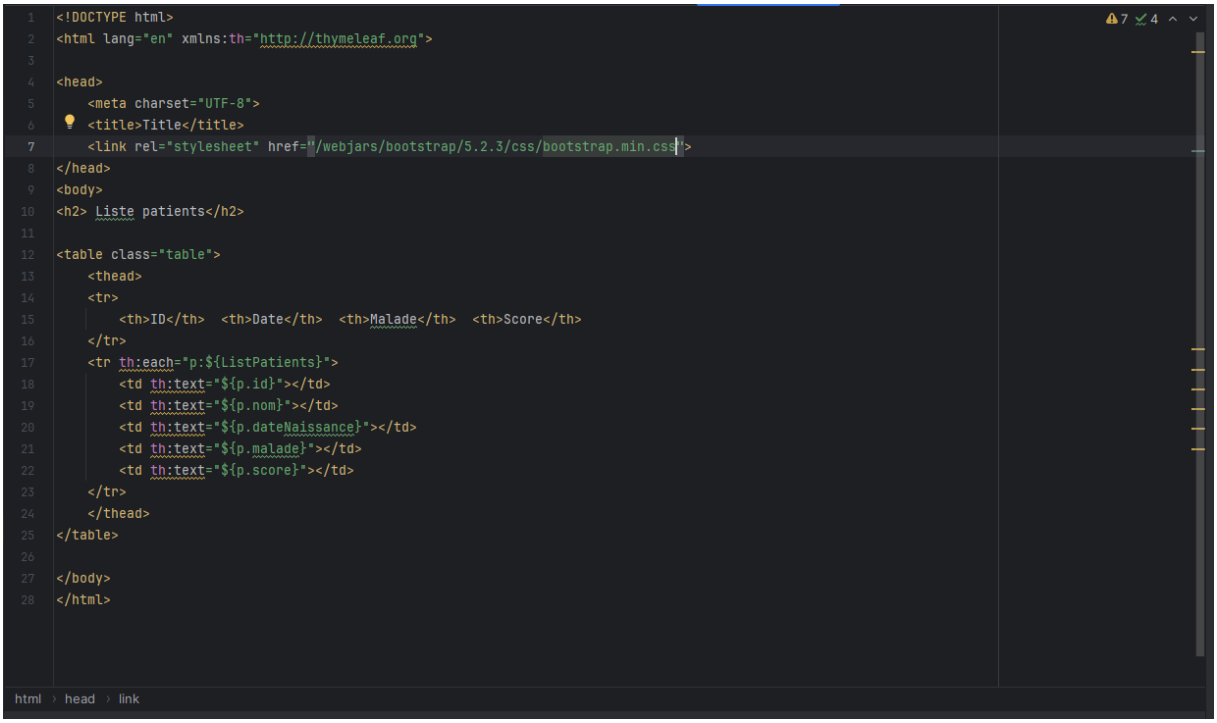
Liste patients

ID	Date	Malade	Score
1	Mohamed 2024-04-22 15:35:11.723	false	34
2	Hanane	2024-04-22 15:35:12.231 false	4321
3	Imane	2024-04-22 15:35:12.233 true	34

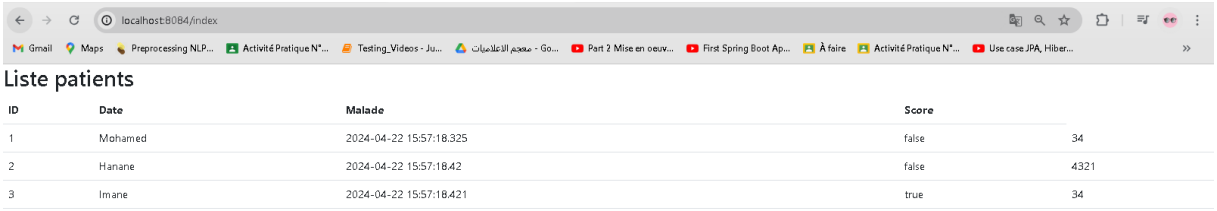
Pour une meilleur apparence de notre liste des patients on va utiliser Bootstrap, donc on va ajouter les dépendances de Bootstrap dans le fichier pom.xml :



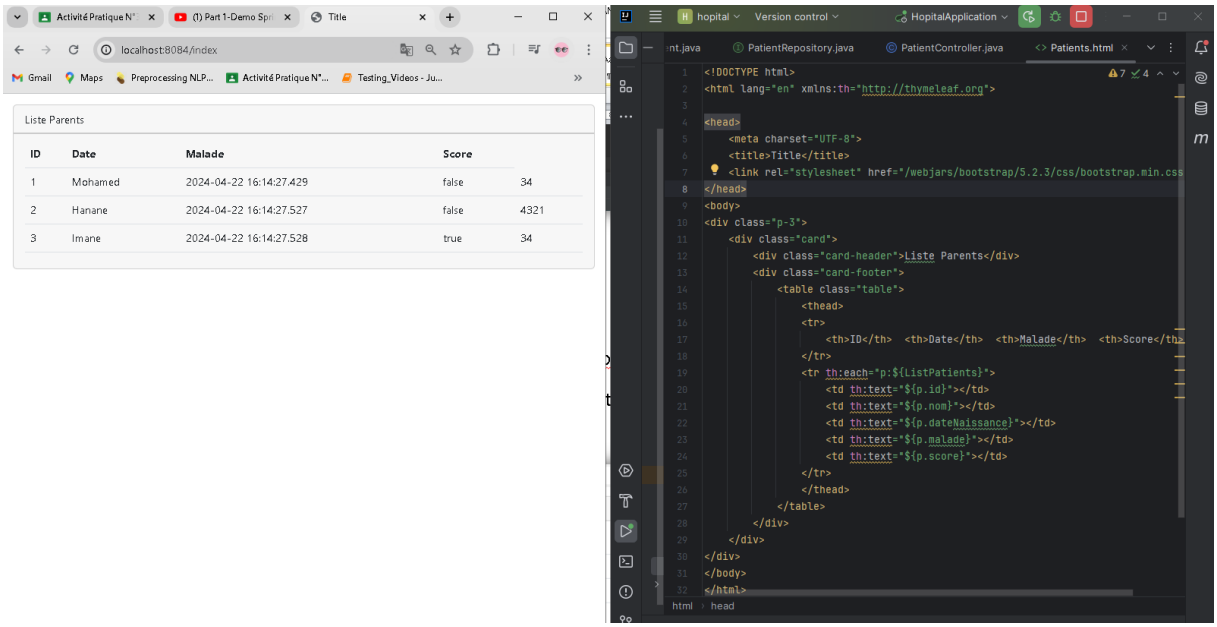
On va utiliser Bootstrap dans notre fichier HTML :



Après exécution on aura comme suivant :

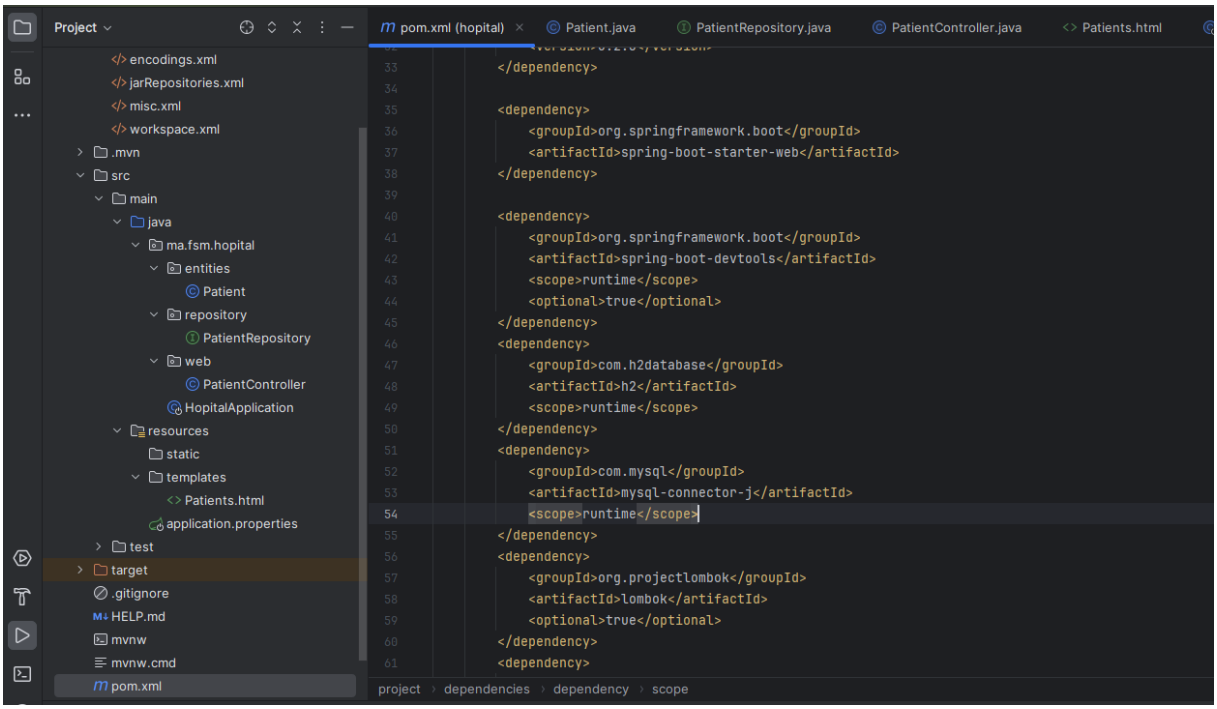


Nous avons effectué quelques changements dans le fichier html pour une meilleure apparence de notre liste.



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

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

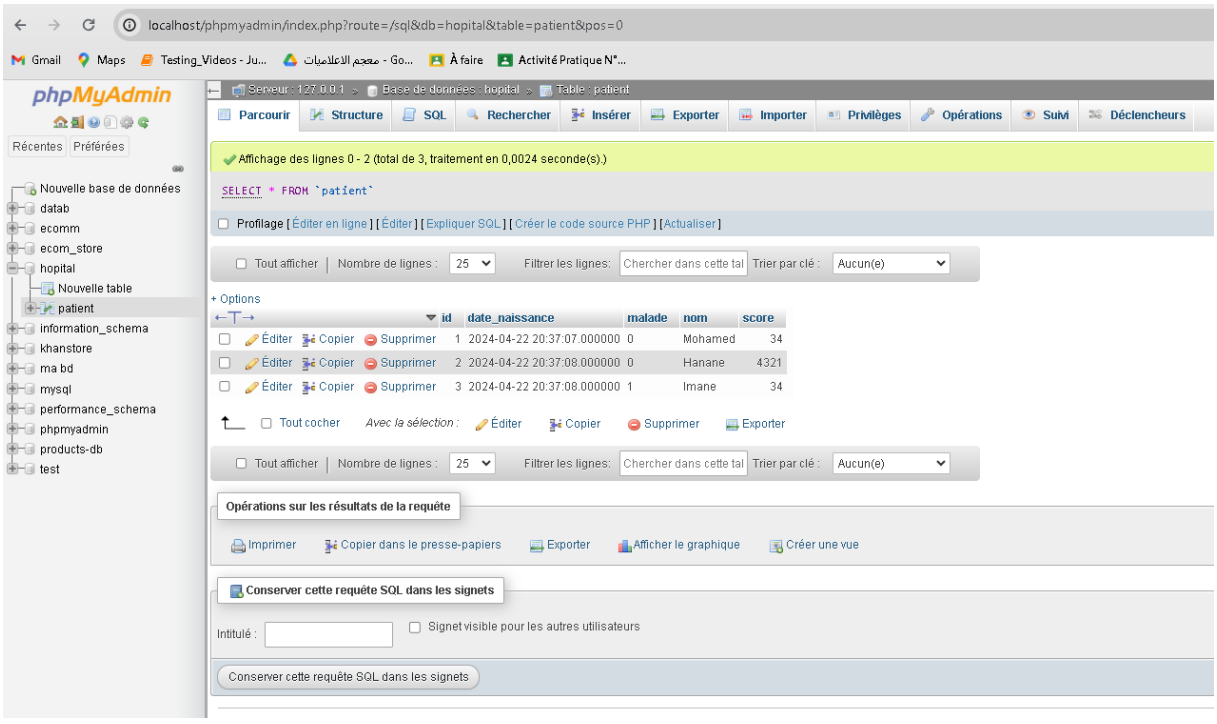


```
spring.application.name=hopital

server.port=8084
# spring.datasource.url=jdbc:h2:mem:patients-db
# spring.h2.console.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
```

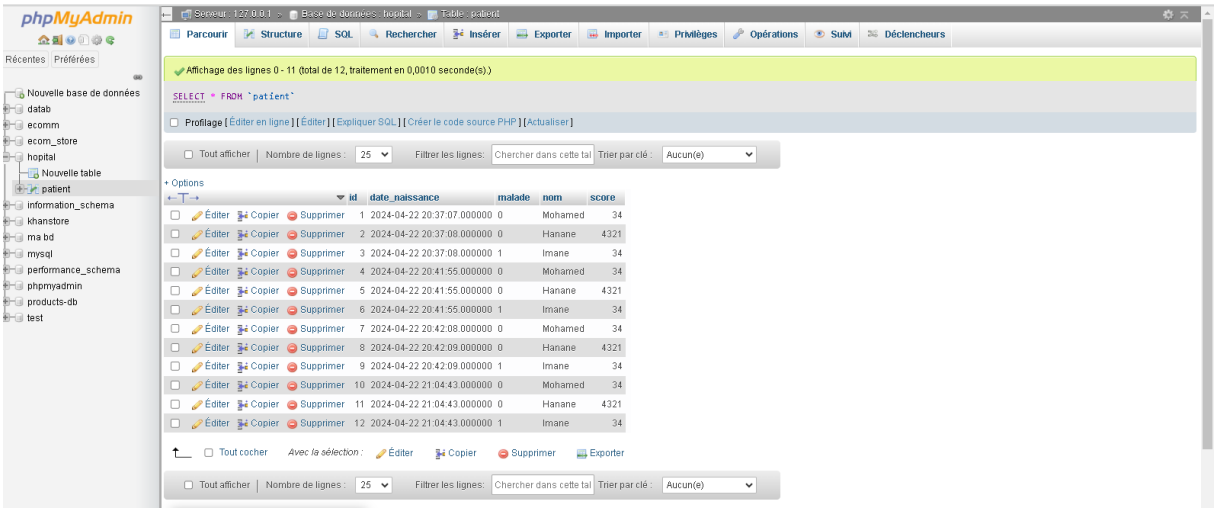
```
package ma.fsm.hopital.web;

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";
        }
```

```
Run  HopitalApplication
Console  Actuator
2024-04-22T20:37:06.119+01:00  WARN 1084 --- [hopital] [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database
2024-04-22T20:37:07.327+01:00  INFO 1084 --- [hopital] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2024-04-22T20:37:07.459+01:00  INFO 1084 --- [hopital] [  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 --- [hopital] [  restartedMain] ma.fsm.hopital.HopitalApplication        : Started HopitalApplication in 11.589 seconds (process running for 1
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 (?,?,?,?)
```

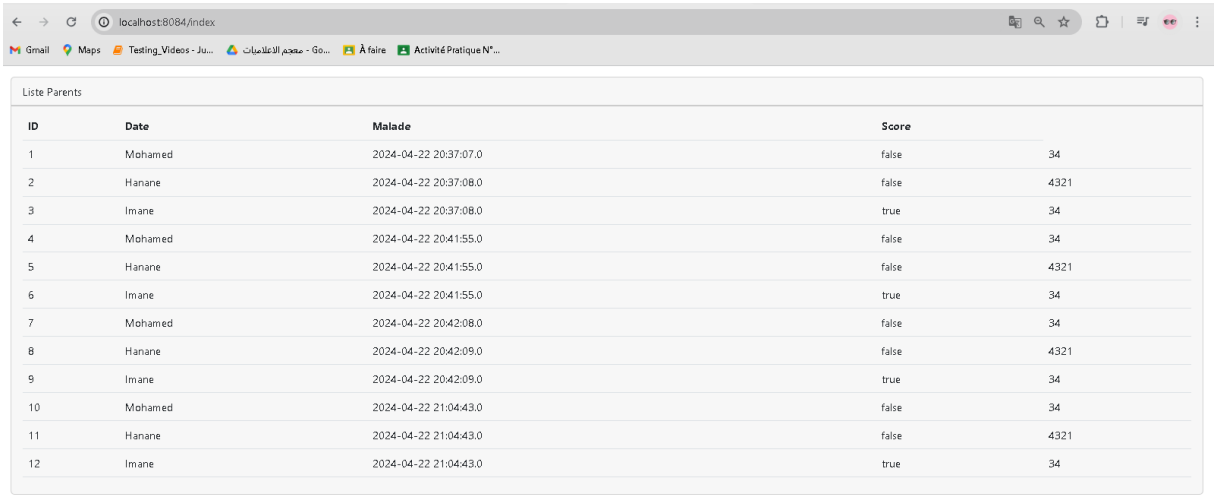
Et on a le suivant dans le phpMyAdmin.



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

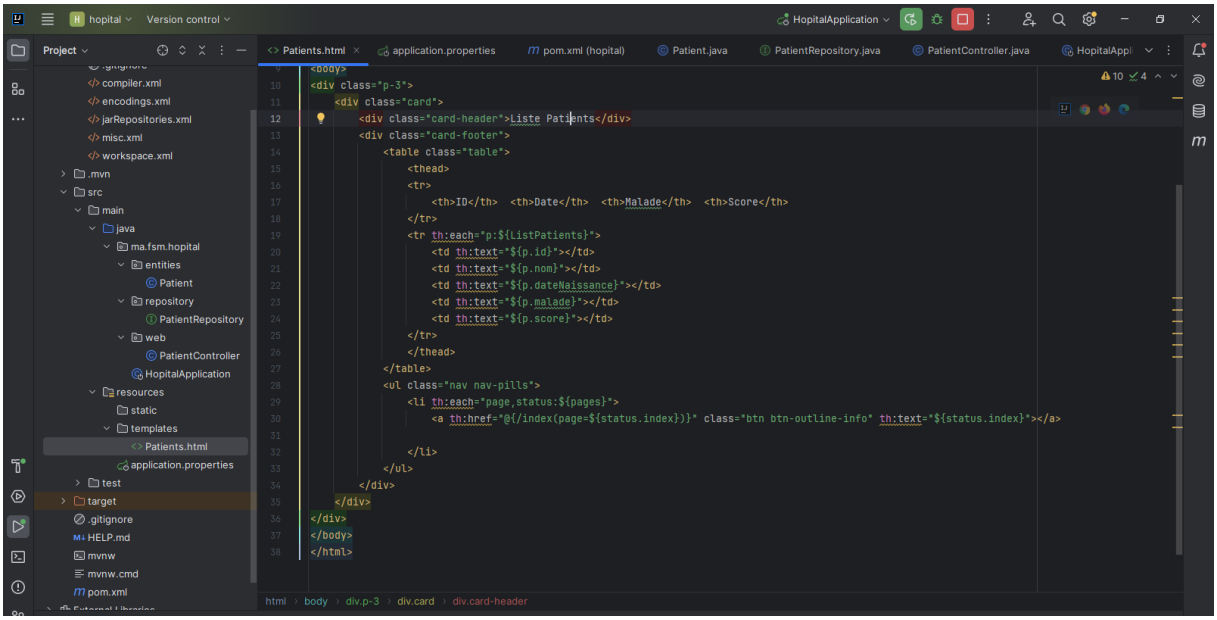


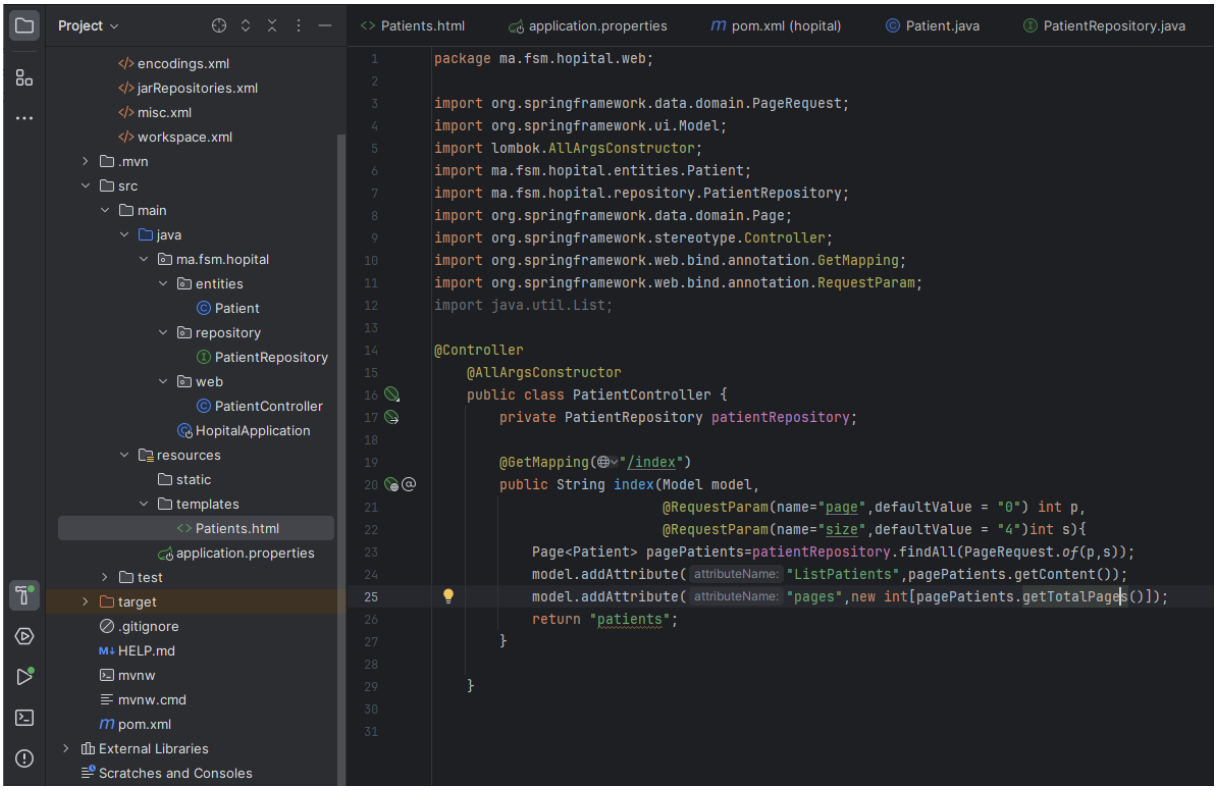
Et le vue se voit comme suivant :



la pagination :

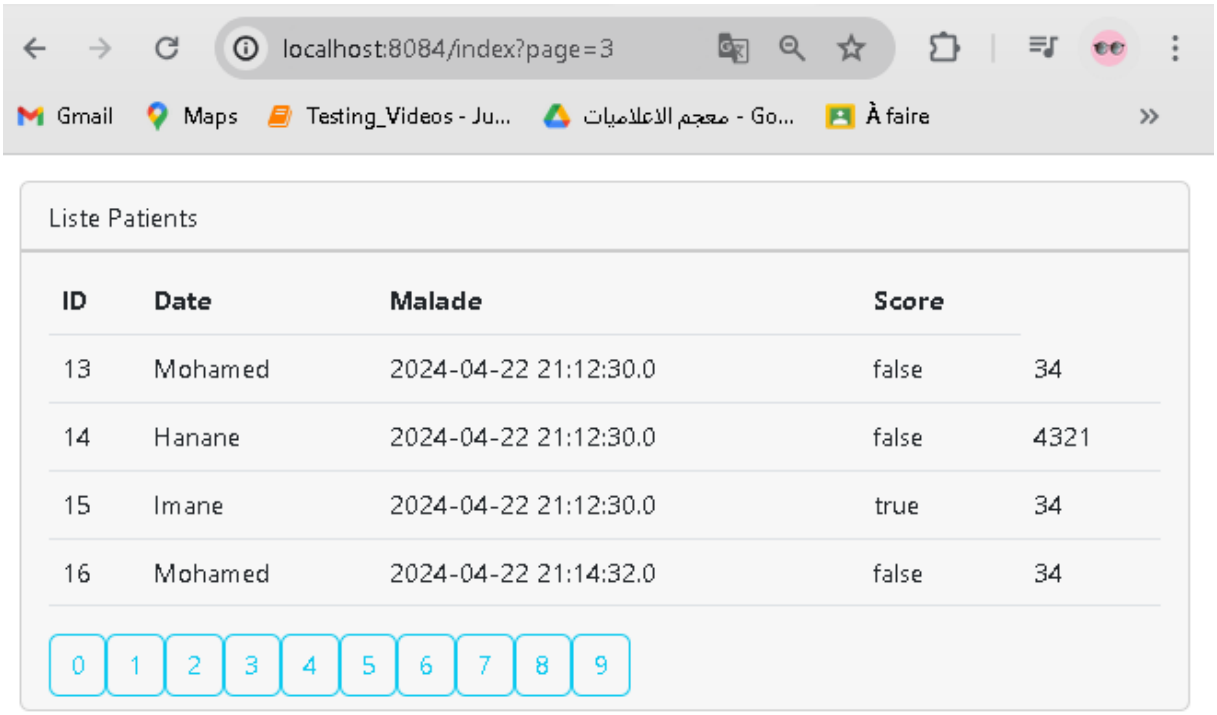
On spécifie dans l'url la page qu'on veut afficher et combien d'éléments .



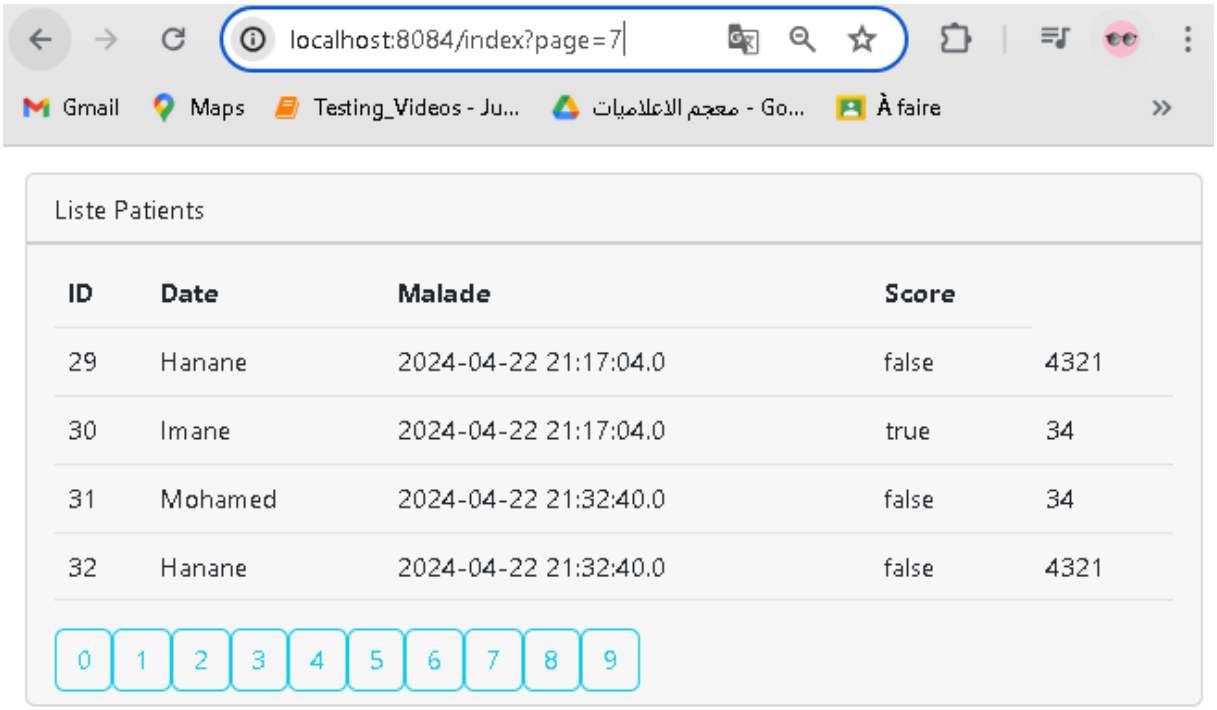


On utilise les boutons pour se déplacer d'une page vers l'autre

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



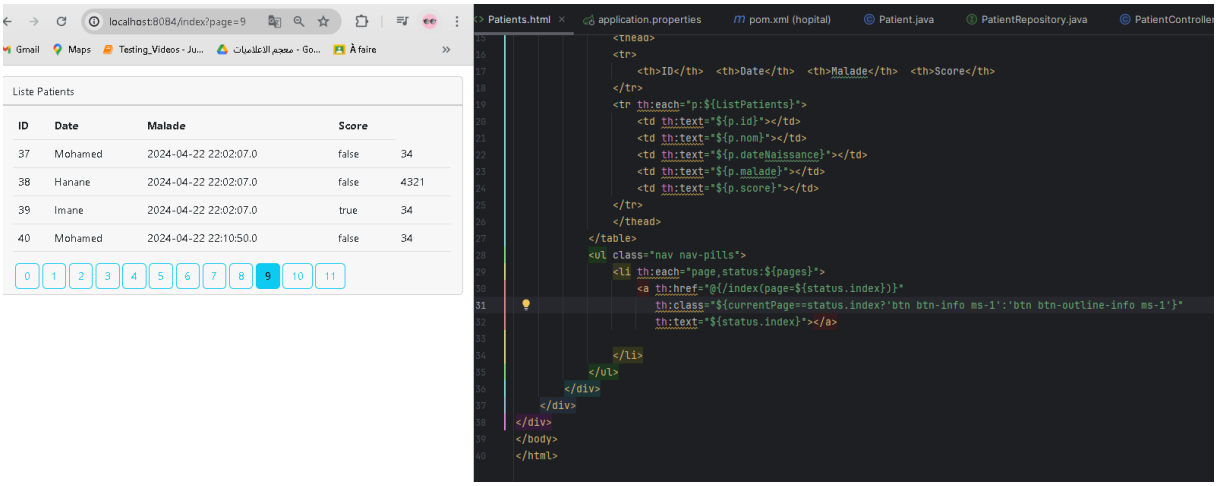
Si en appuyer sur la page 7 elle affiche le suivant :



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

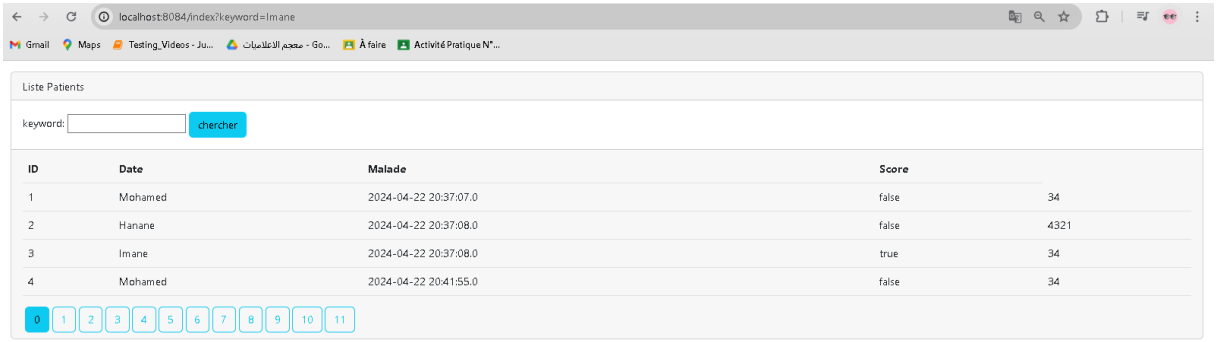
PatientController.java | Patients.html | application.properties | pom.xml (hopital) | Patient.java

```java
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,
                            @RequestParam(name="page",defaultValue = "0") int p,
                            @RequestParam(name="size",defaultValue = "4")int s){
            Page<Patient> pagePatients=patientRepository.findAll(PageRequest.of(p,s));
            model.addAttribute( attributeName: "ListPatients",pagePatients.getContent());
            model.addAttribute( attributeName: "pages",new int[pagePatients.getTotalPages()]);
            model.addAttribute( attributeName: "currentPage",p);
            return "patients";
        }

    }
```



Pour créer un formulaire dans notre vue :

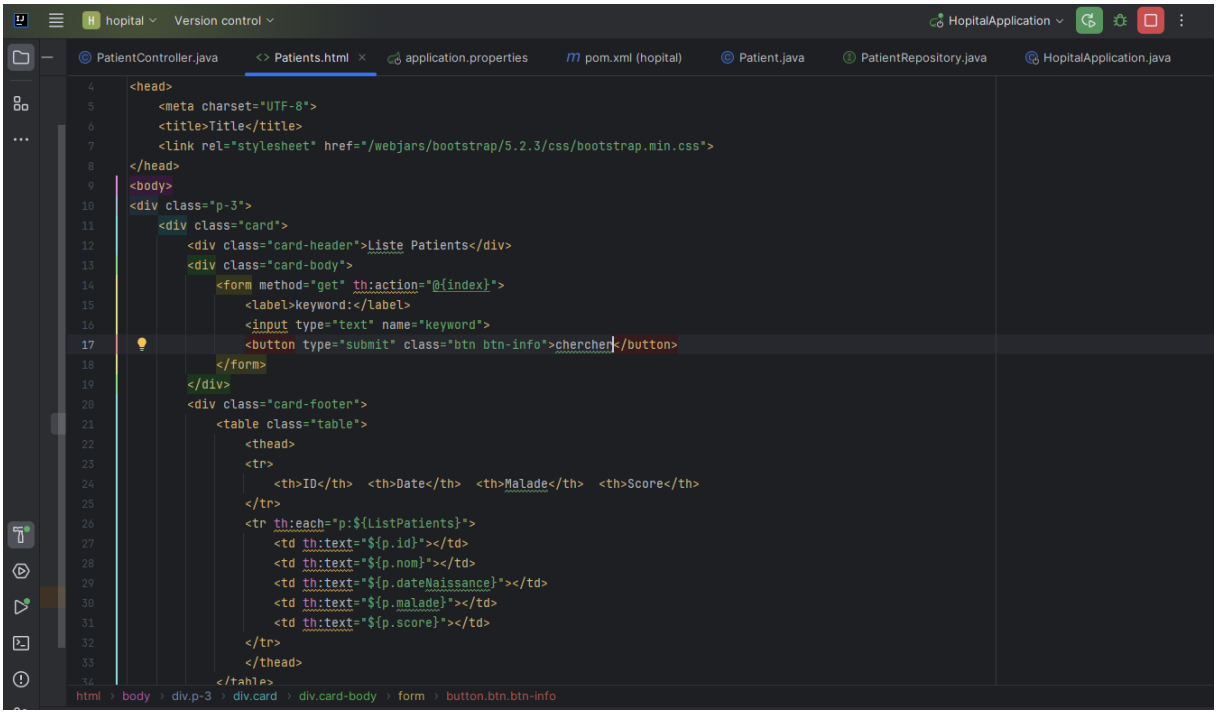
```
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
</head>
<body>
<div class="p-3">
    <div class="card">
        <div class="card-header">Liste Patients</div>
        <div class="card-body">
            <form 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>
                <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>
```



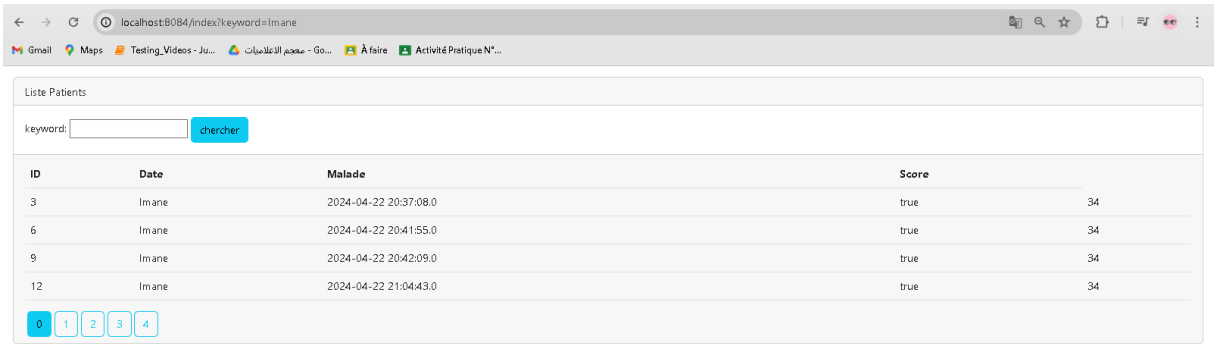
On fait un changement au niveau de PatientController :

```
package ma.fsm.hopital.web;

import org.springframework.data.domain.PageRequest;
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,
                            @RequestParam(name="page",defaultValue = "0") int p,
                            @RequestParam(name="size",defaultValue = "4")int s,
                            @RequestParam(name="keyword",defaultValue = "") String kw)
    {
            Page<Patient> pagePatients=patientRepository.findByNomContains(kw,PageRequest.of(p,s));
            model.addAttribute( attributeName: "ListPatients",pagePatients.getContent());
            model.addAttribute( attributeName: "pages",new int[pagePatients.getTotalPages()]);
            model.addAttribute( attributeName: "currentPage",p);
            return "patients";
        }

    }
```

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>
                <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>
```



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

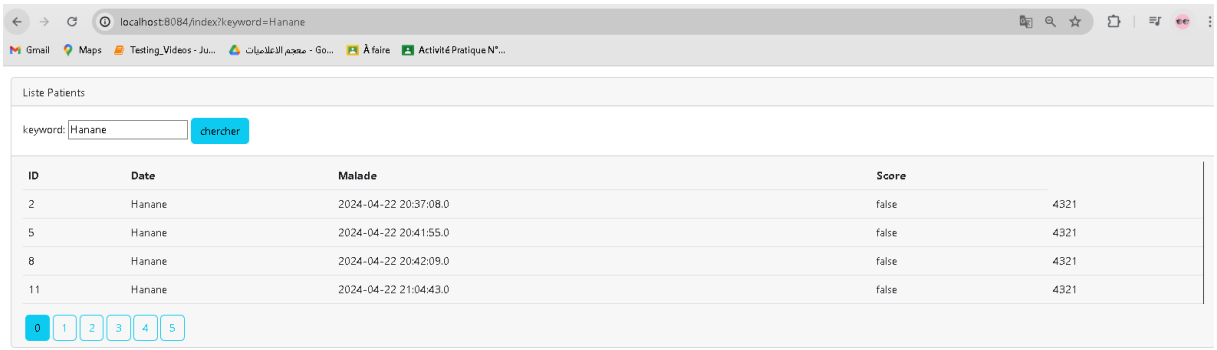
PatientController.java × | Patients.html | application.properties | pom.xml (hopital) | Patient.java | PatientRepository.java

```java
package ma.fsm.hopital.web;

import org.springframework.data.domain.PageRequest;
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,
                            @RequestParam(name="page",defaultValue = "0") int p,
                            @RequestParam(name="size",defaultValue = "4")int s,
                            @RequestParam(name="keyword",defaultValue = "") String kw)
    {
            Page<Patient> pagePatients=patientRepository.findByNomContains(kw,PageRequest.of(p,s));
            model.addAttribute( attributeName: "ListPatients",pagePatients.getContent());
            model.addAttribute( attributeName: "pages",new int[pagePatients.getTotalPages()]);
            model.addAttribute( attributeName: "currentPage",p);
            model.addAttribute( attributeName: "keyword",kw);
            return "patients";
        }
```

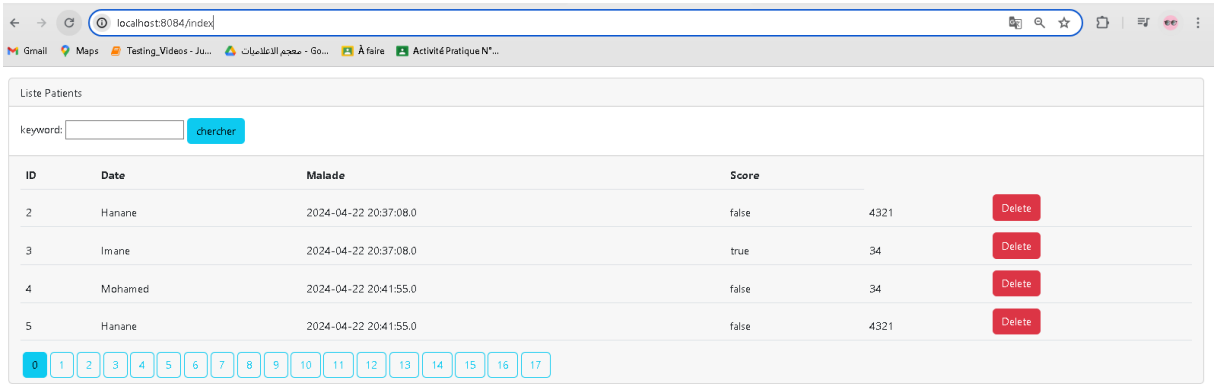
PatientController.java | Patients.html × | application.properties | pom.xml (hopital) | Patient.java | Patien

```html
    <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" th:value="${keyword}" >
                <button type="submit" class="btn btn-info">chercher</button>
            </form>
        </div>
        <div class="card-footer">
            <table class="table">
                <thead>
                <tr>
                    <th>ID</th>  <th>Date</th>  <th>Malade</th>  <th>Score</th>
                </tr>
                <tr th:each="p:${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="${p.score}"></td>
                </tr>
                </thead>
            </table>
            <ul class="nav nav-pills">
                <li th:each="value_item:${pages}">
```

html › body › div.p-3 › div.card › div.card-body › form › input

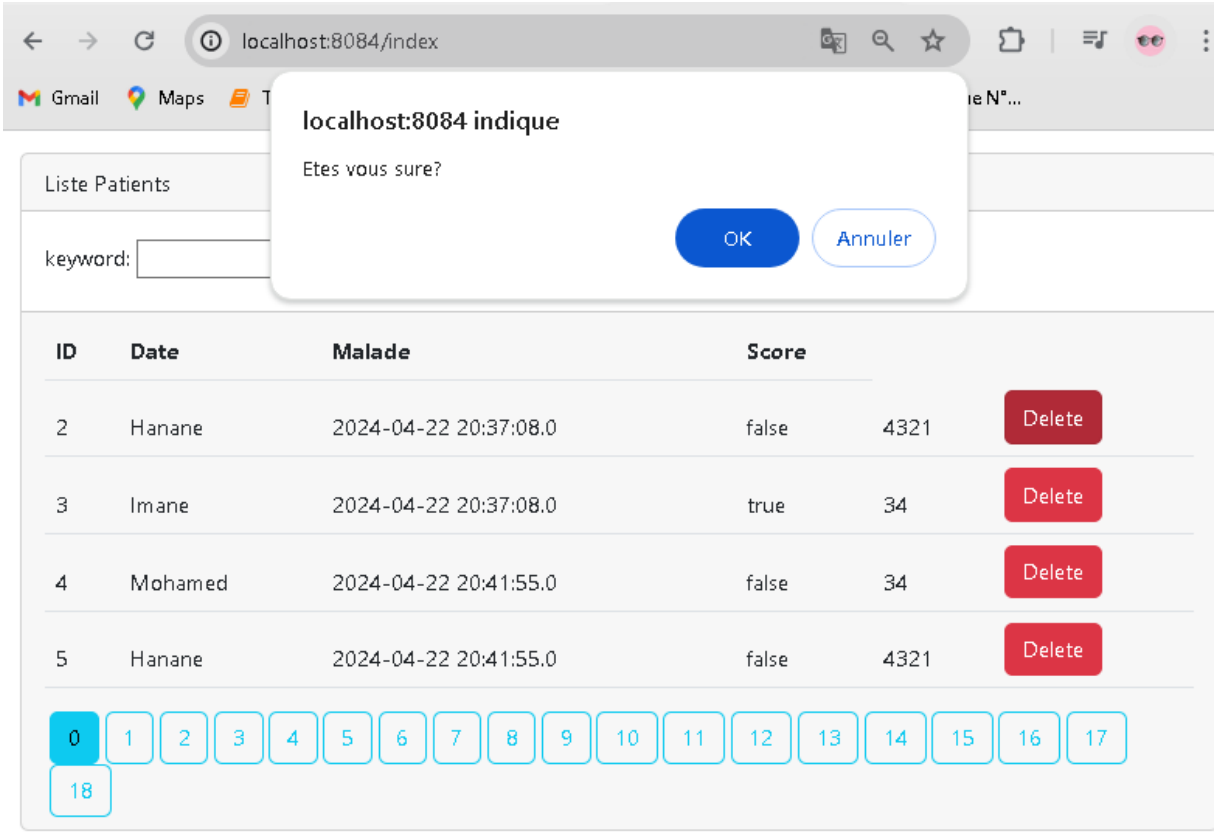


Pour supprimer un patient on va faire le suivant :



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

```
<table class="table">
    <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>
        <td>
            <a onclick="return confirm('Etes vous sure?')" th:href="@{delete(id=${p.id})}" class="btn btn-danger">Delete</a>
        </td>
    </tr>
    </thead>
</table>
<ul class="nav nav-pills">
    <li th:each="value,item:${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>
</div>
</div>
</div>
</body>
</html>
```



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

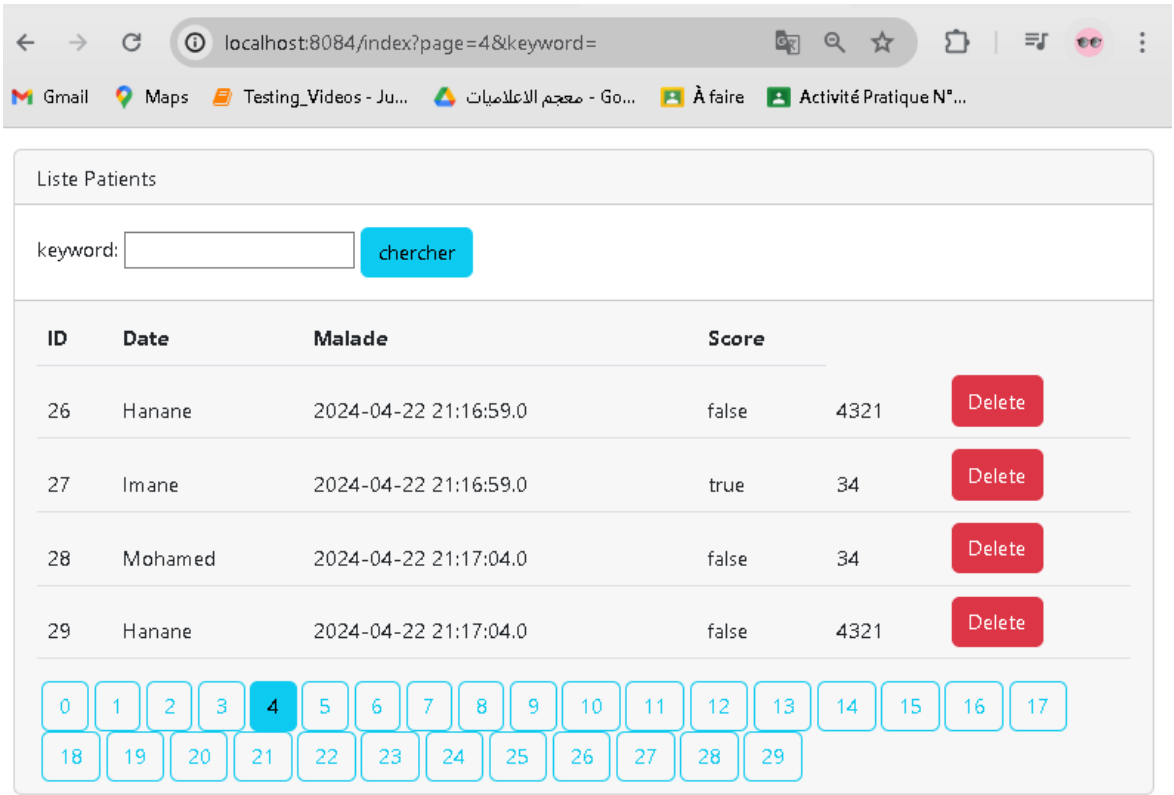
```
<table class="table">
    <thead>
    <tr>
        <th>ID</th>  <th>Date</th>  <th>Malade</th>  <th>Score</th>
    </tr>
    <tr th:each="p:${ListPatients}">
        <td th:text="${p.id}"></td>
        <td th:text="${p.nom}"></td>
        <td th:text="${p.dateNaissance}"></td>
        <td th:text="${p.malade}"></td>
        <td th:text="${p.score}"></td>
        <td>
            <a onclick="javascript:return confirm('Etes vous sure?')"
               th:href="@{delete(id=${p.id},keyword=${keyword},page=${currentPage})}"
               class="btn btn-danger">Delete</a>
        </td>
    </tr>
    </thead>
</table>
<ul class="nav nav-pills">
    <li th:each="value,item:${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>
</div>
</div>
</div>
</body>
</html>
```

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

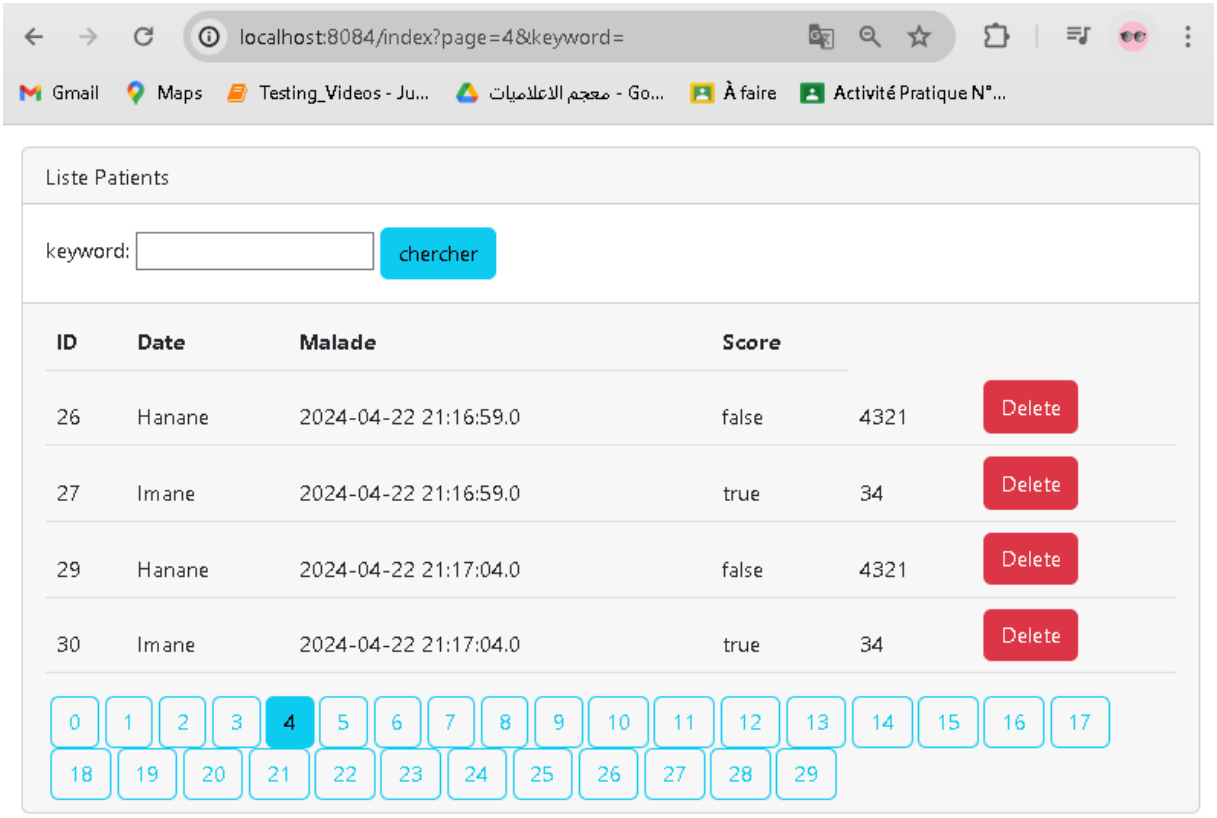
PatientController.java | Patients.html | application.properties | pom.xml (hopital) | Patient.java | PatientRepository.java

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

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


        }

    }

```

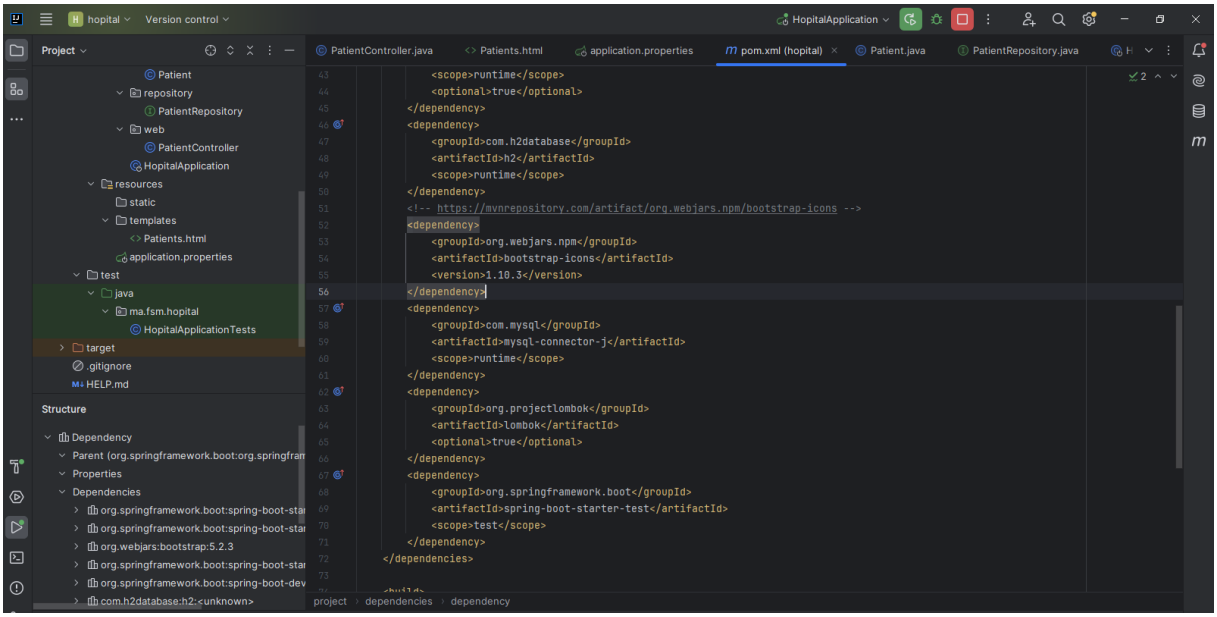
Je doit supprime l'élément mohamed.

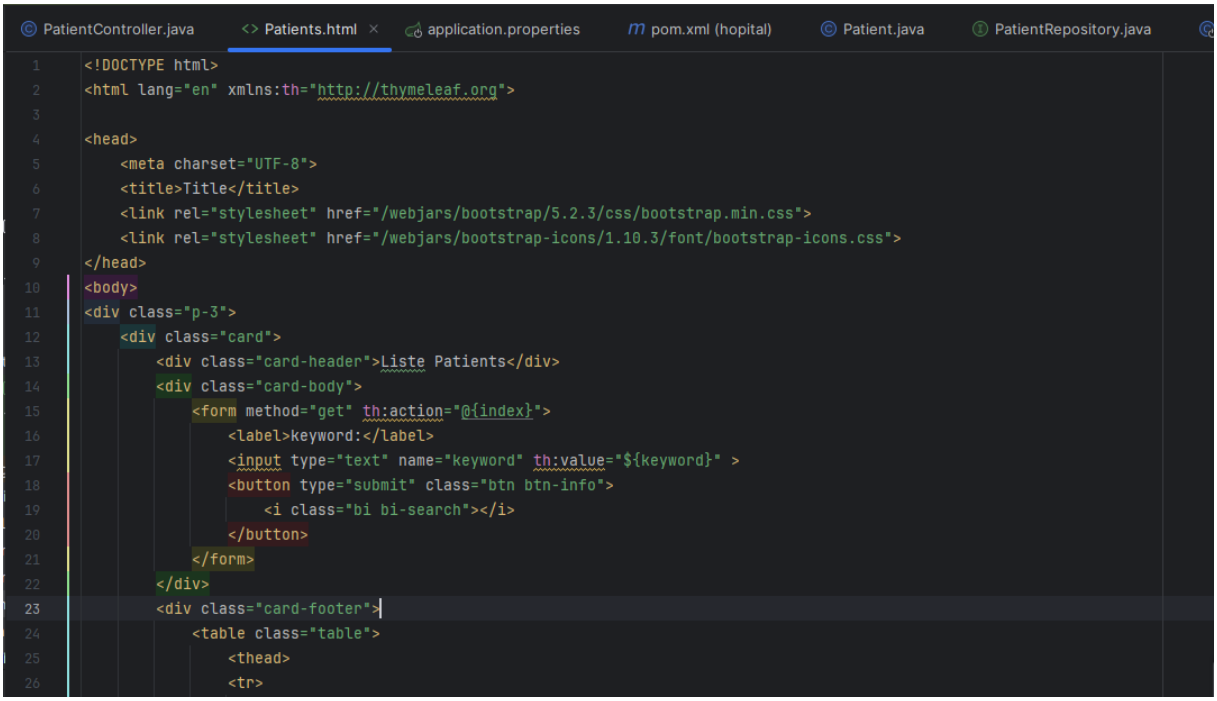


Après la suppression : On a reste dans la page 4 alors le problème est résolu.

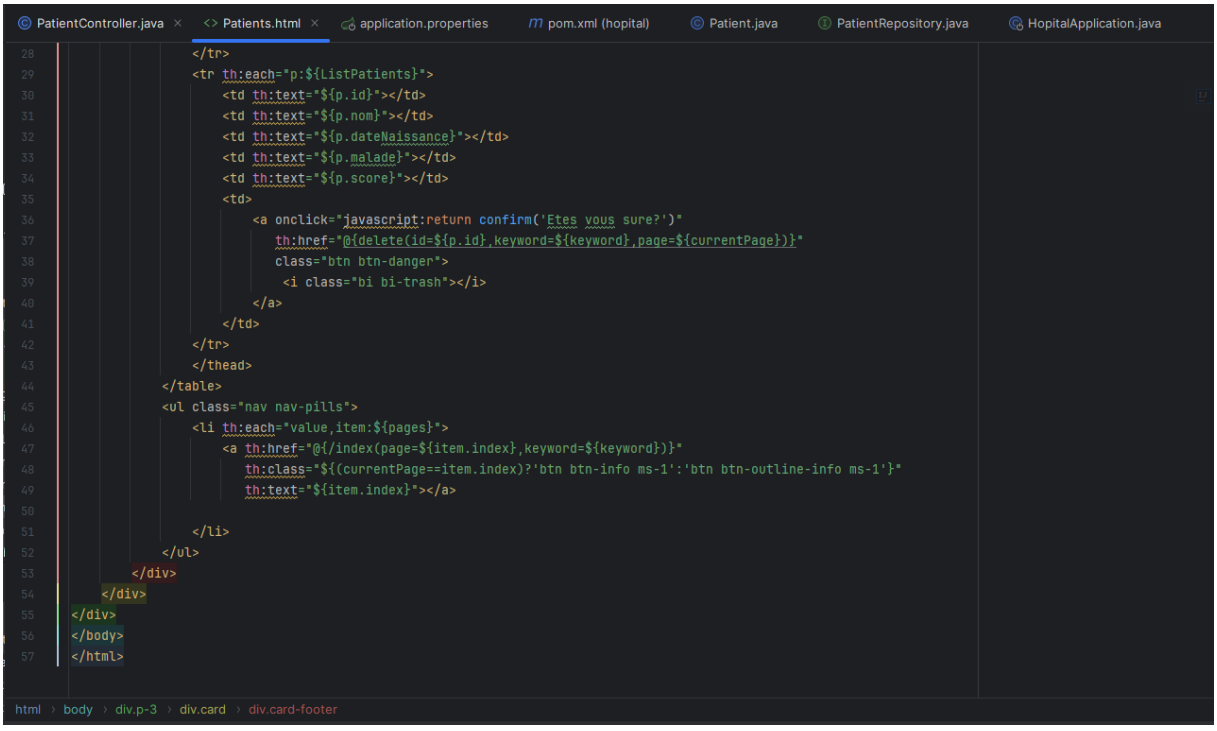


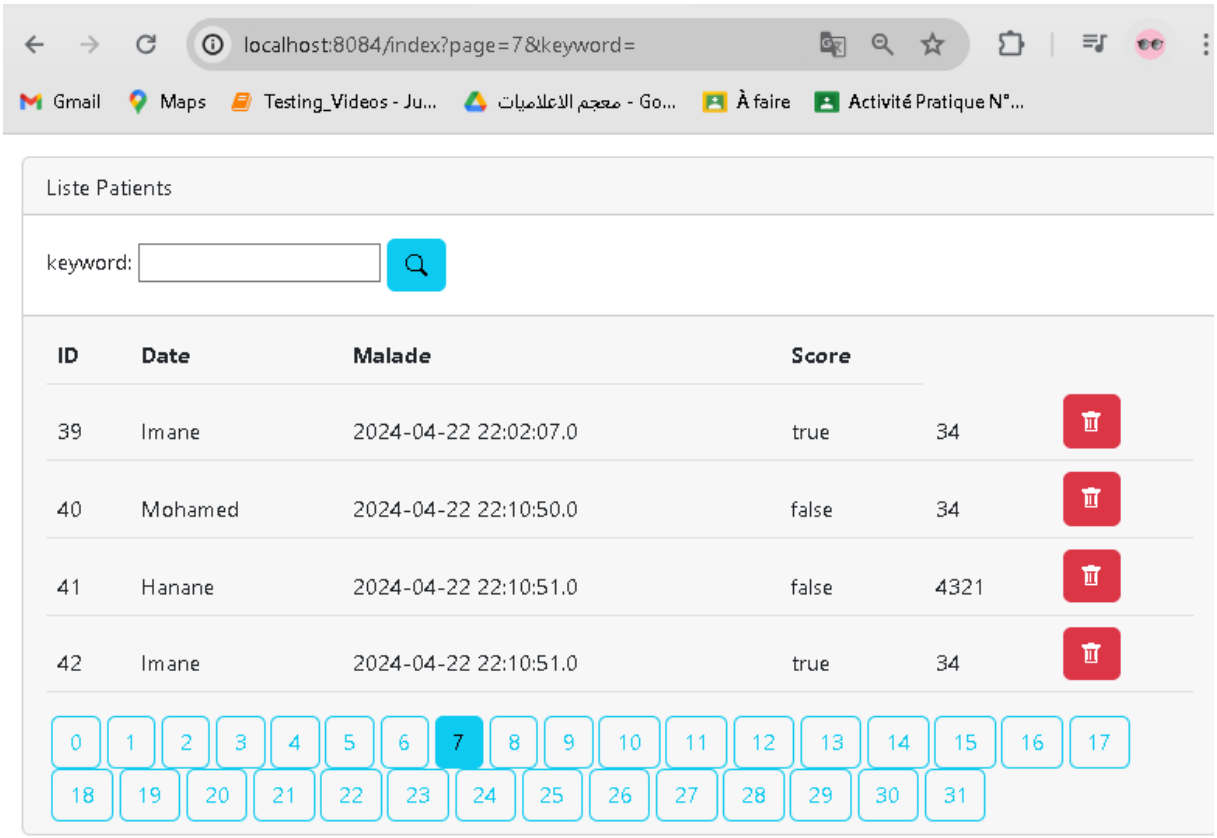
Ajoutons Quelques améliorations supplémentaires : En utilisons bootstrap_icons de search et delete.





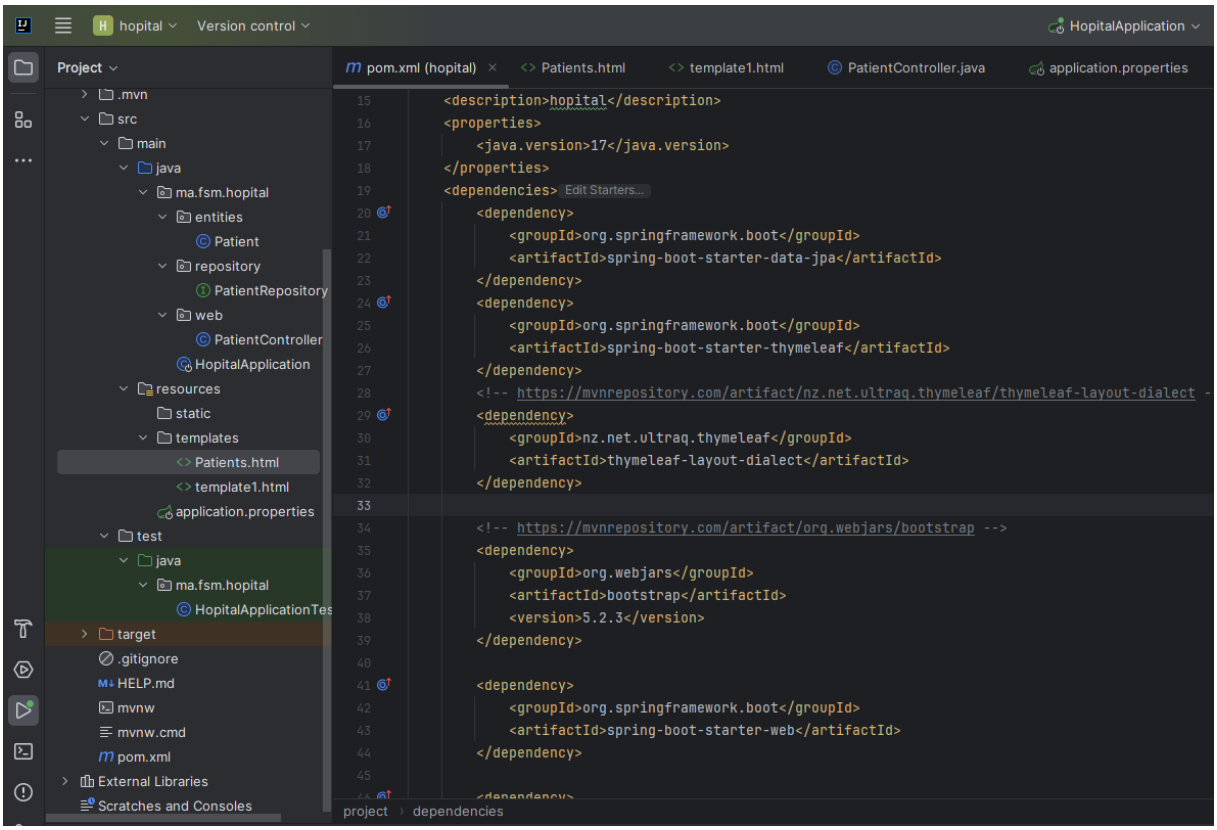
Fichier Patients.html :

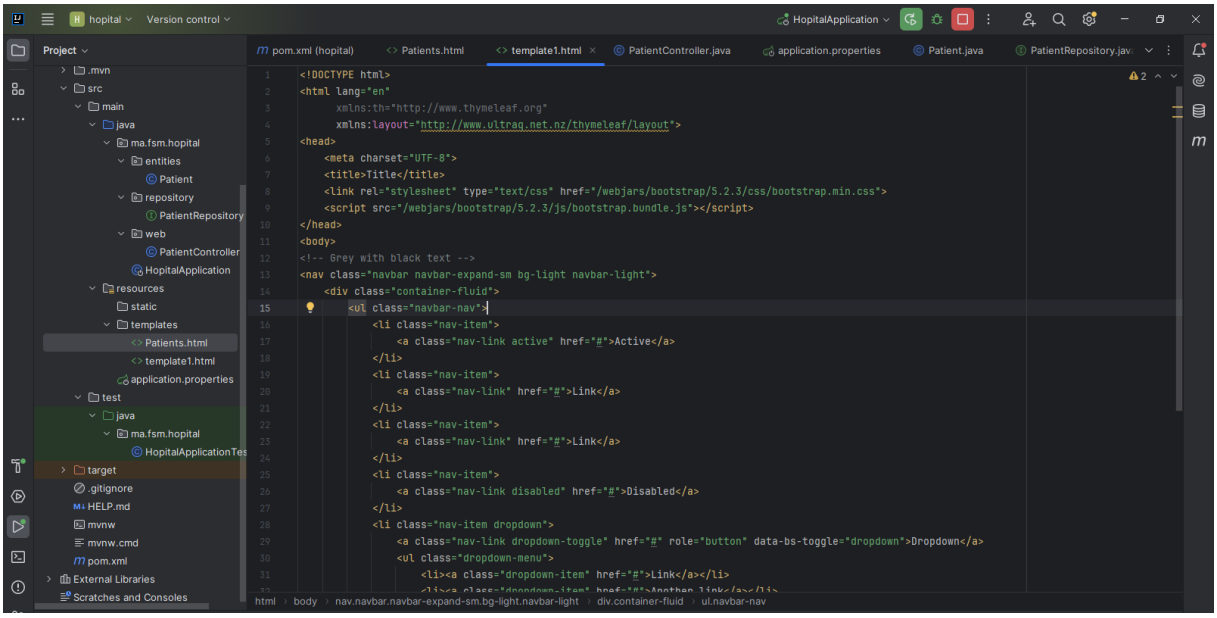




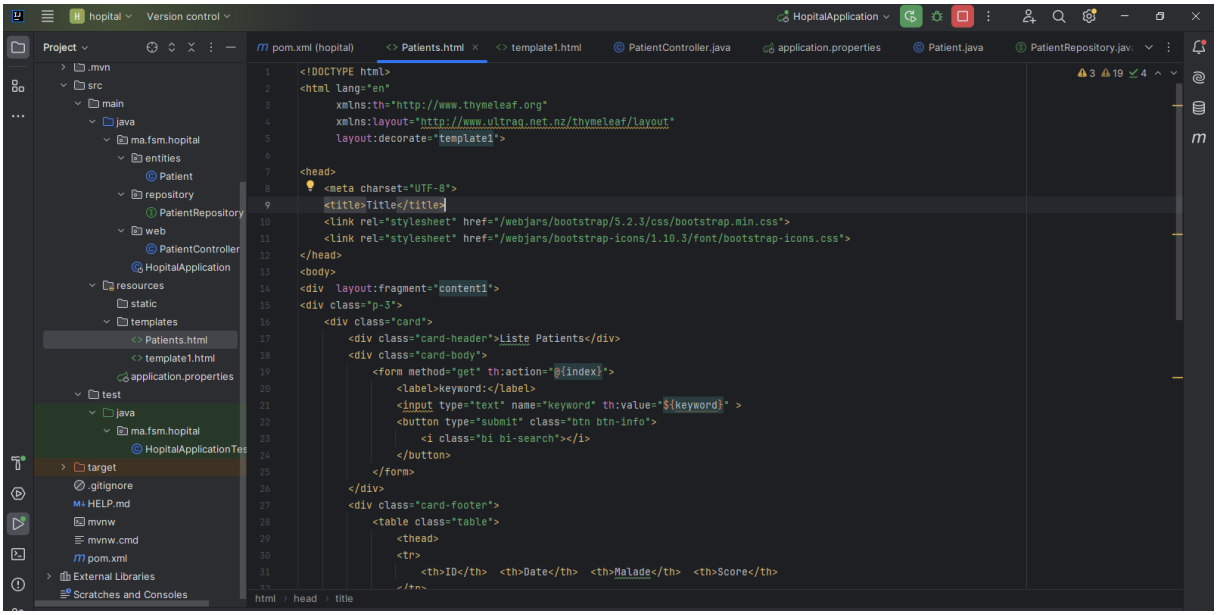
⇒ Partie 2 :

- Créer une page template1 Télécharger les dépendances.

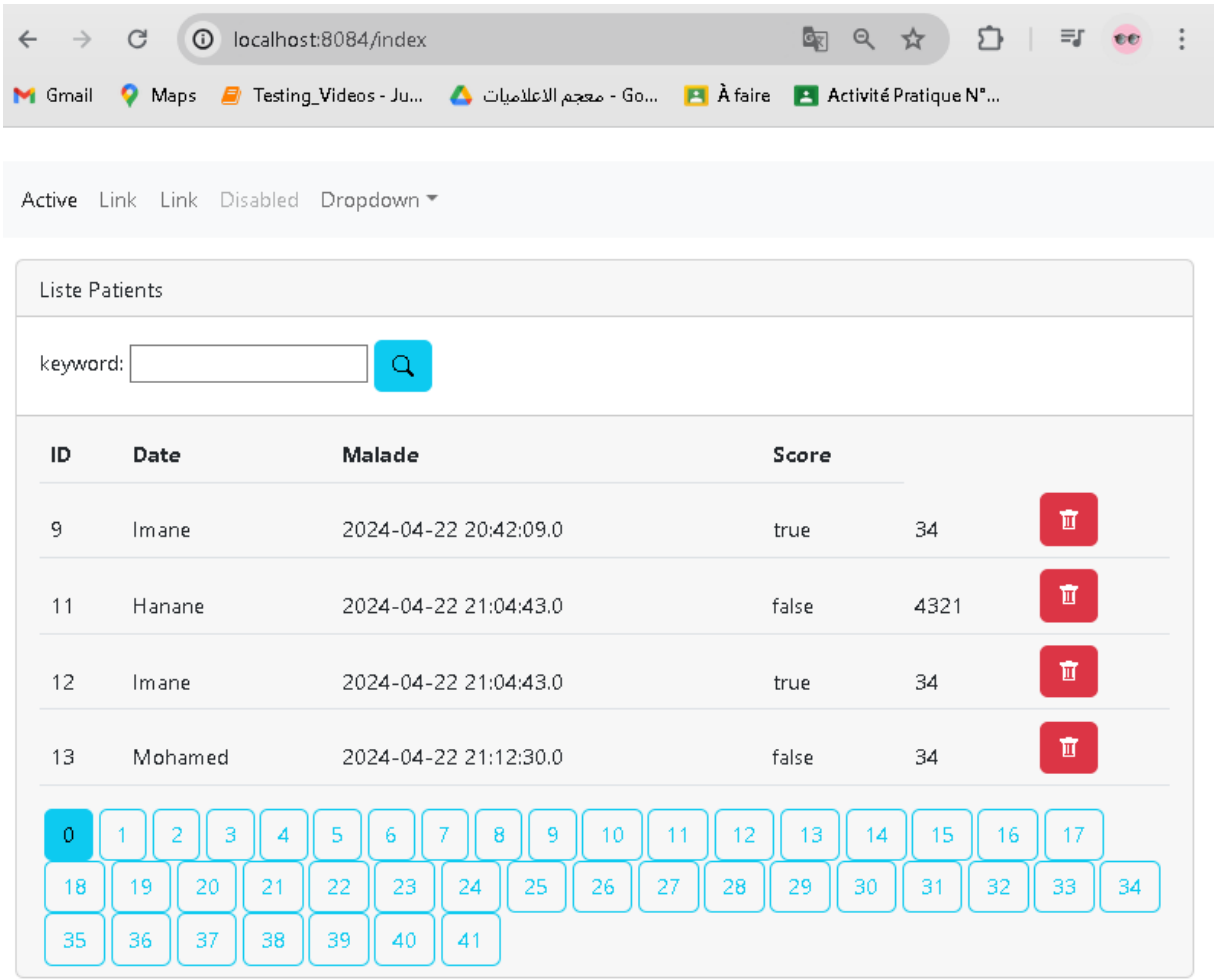




Et utiliser la page template1 dans le fichier html Patients.

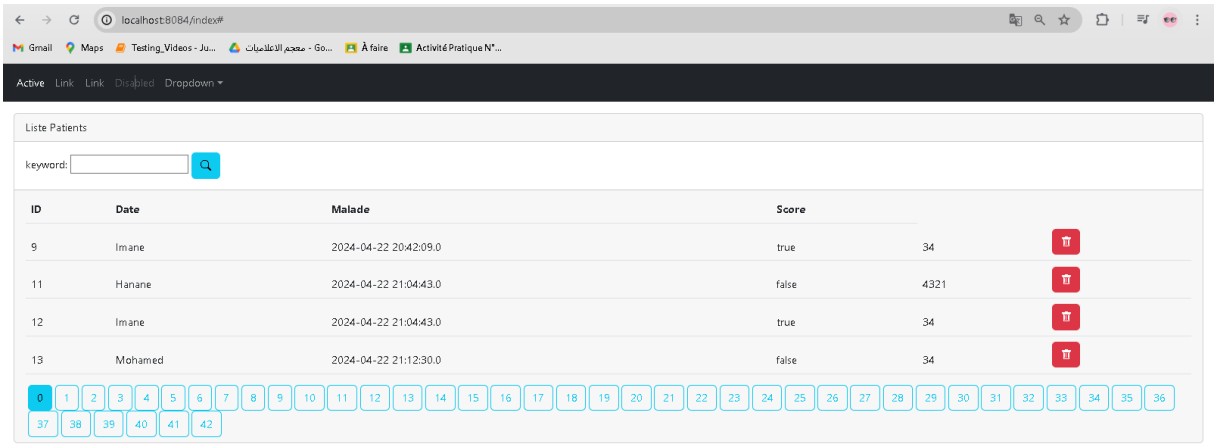


On exécute notre application.

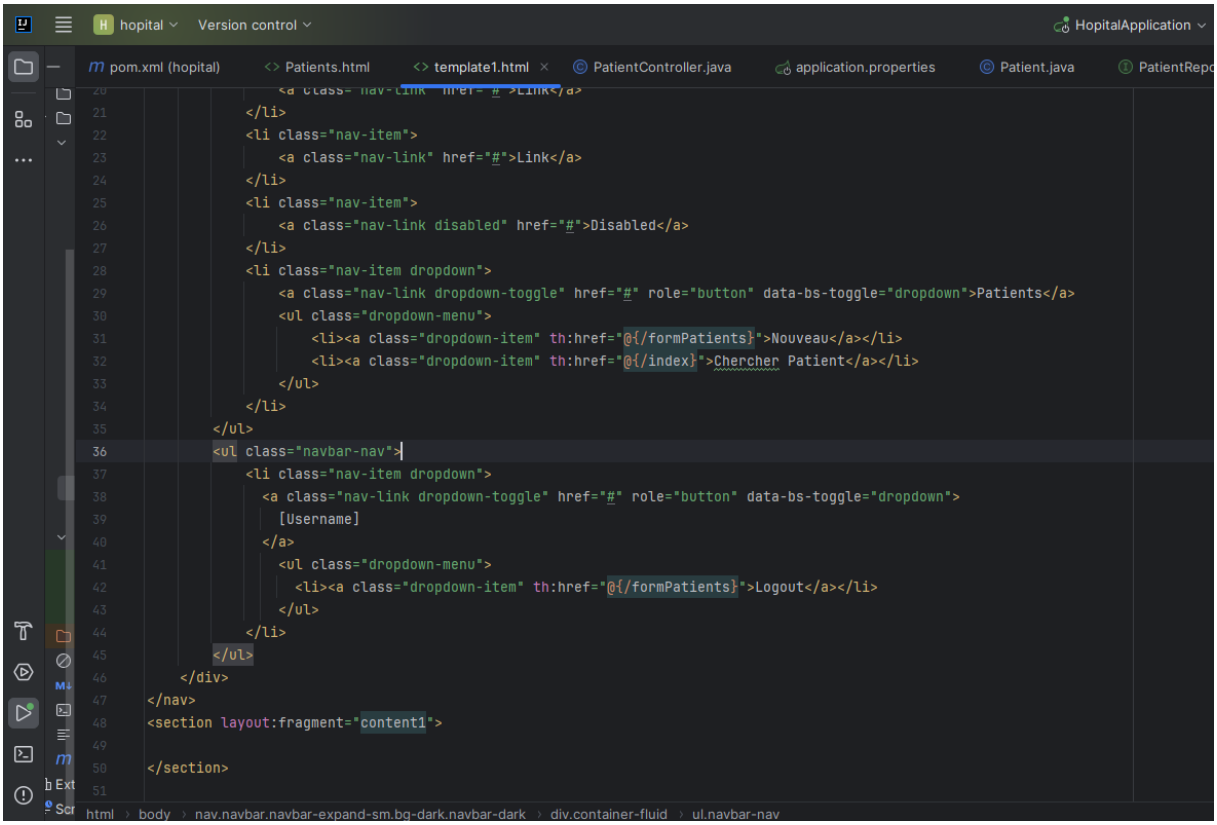


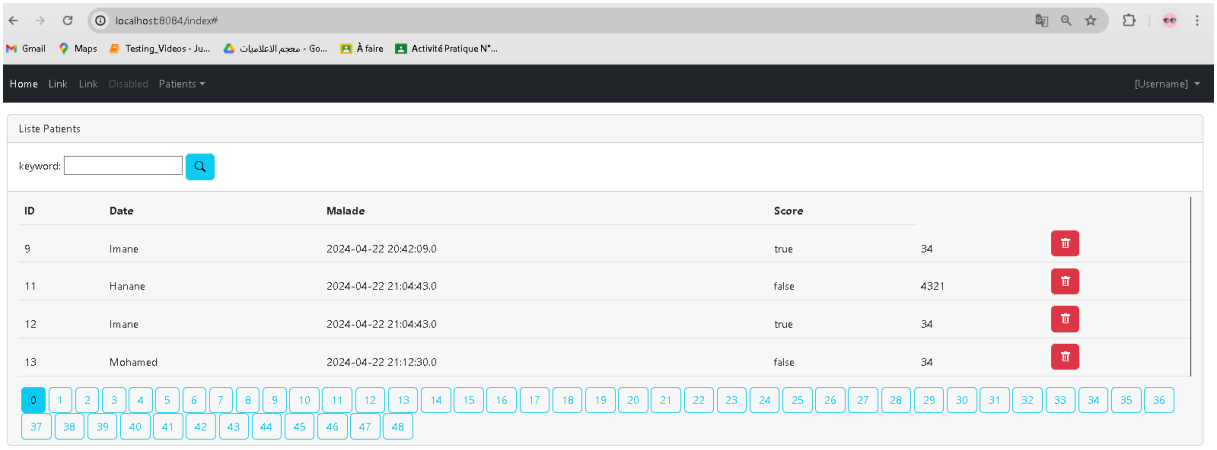
- Colorer la barre de navigation.

```
nts.html   template1.html   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">D
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>
html › body › nav.navbar.navbar-expand-sm.bg-dark.navbar-dark
```

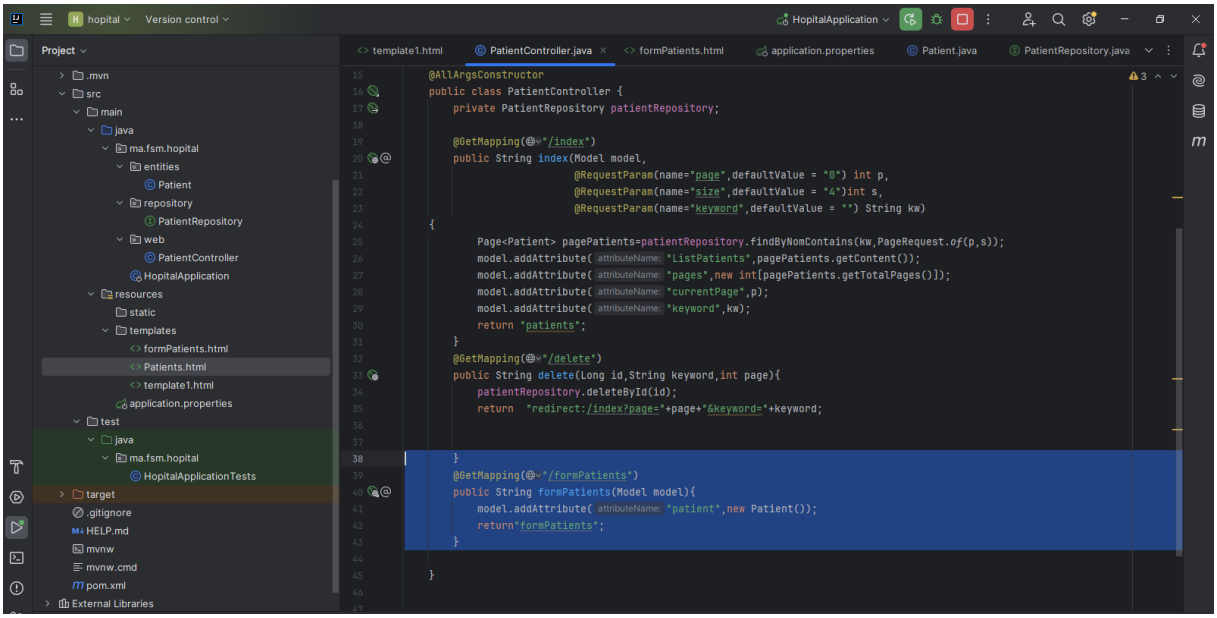


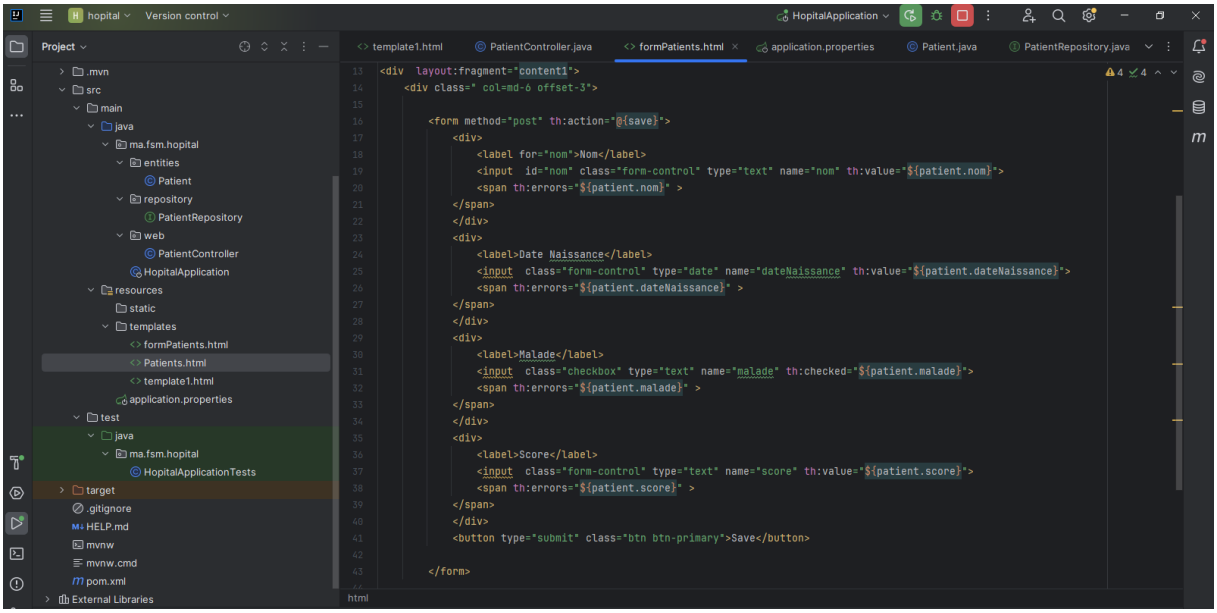
Ajoutons au header les éléments suivants ajouter un nouveau patient, chercher un patient, et se déconnecter .

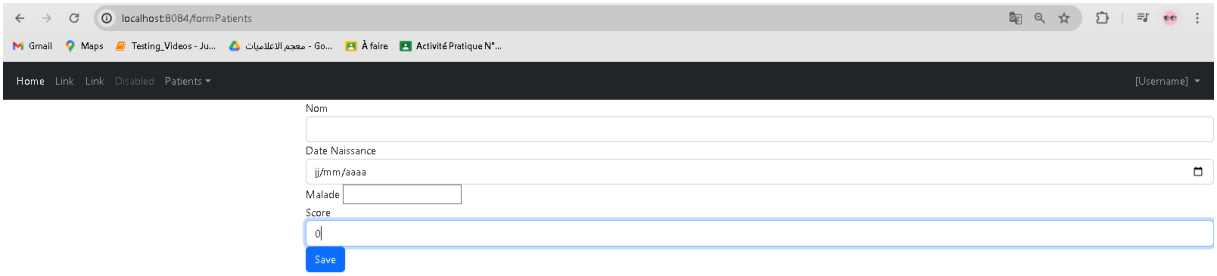




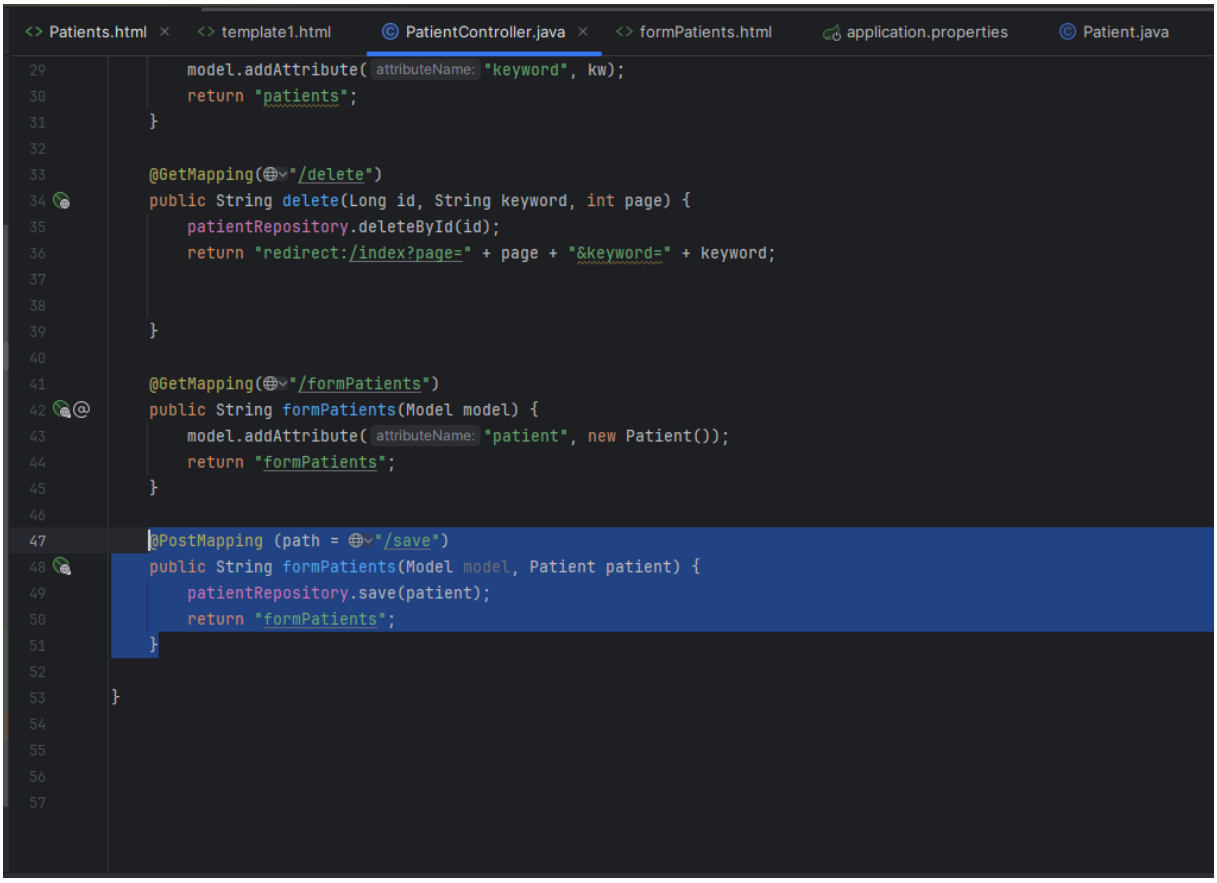
La méthode « formPatients » :On va créer un fichier HTML que l'on appelle formPatients.html, a fin d'afficher le formulaire d'ajout des patients.



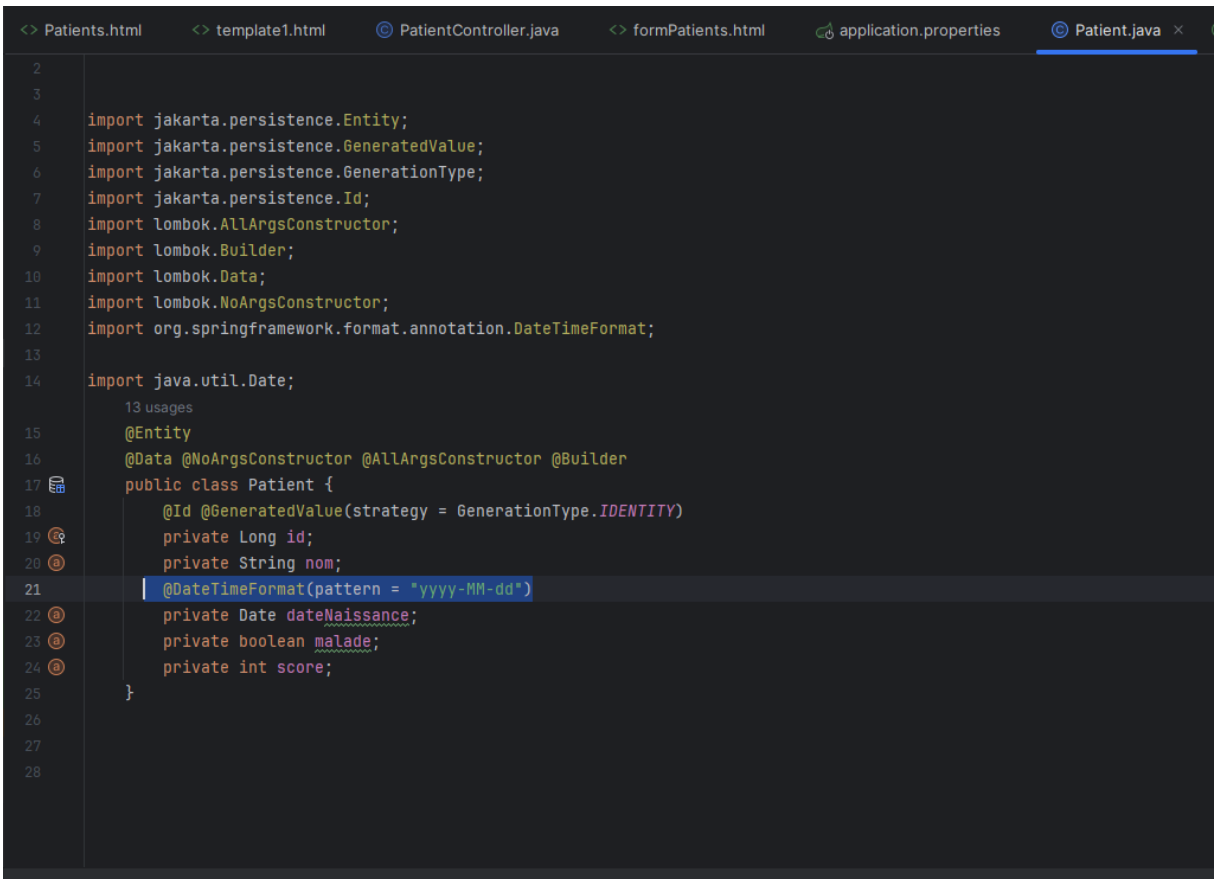




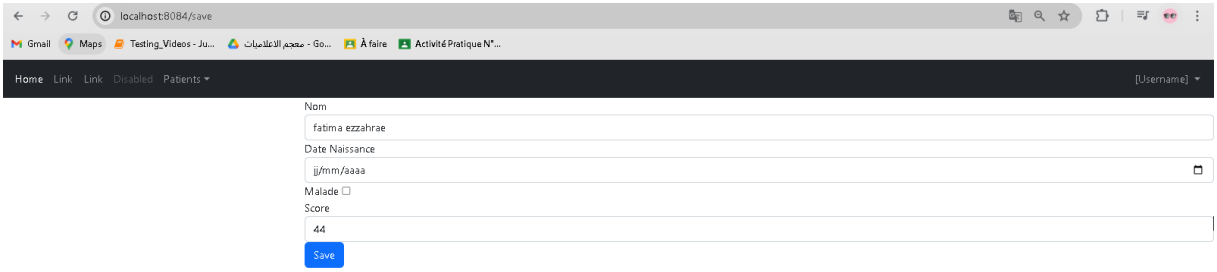
- méthode save() :



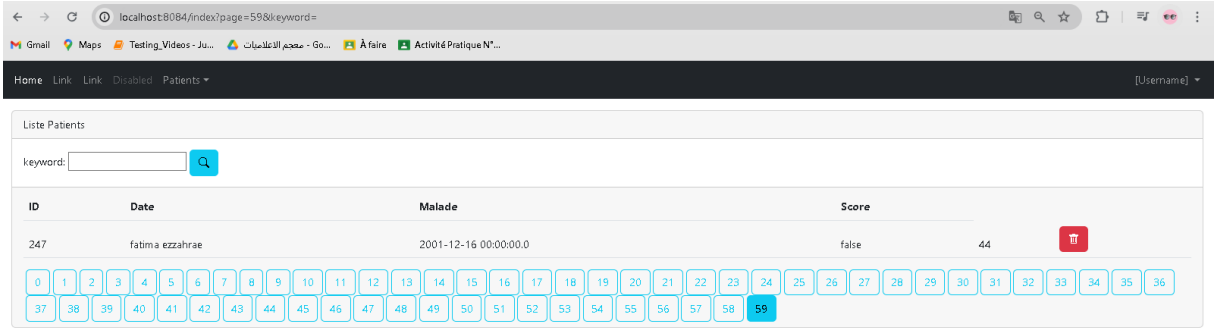
Ajout de annotation @DateTimeFormat au classe Patient :



remplir le formulaire par le patient fatima ezzahrae



Le patient est bien enregistrer dans la listes des patients.



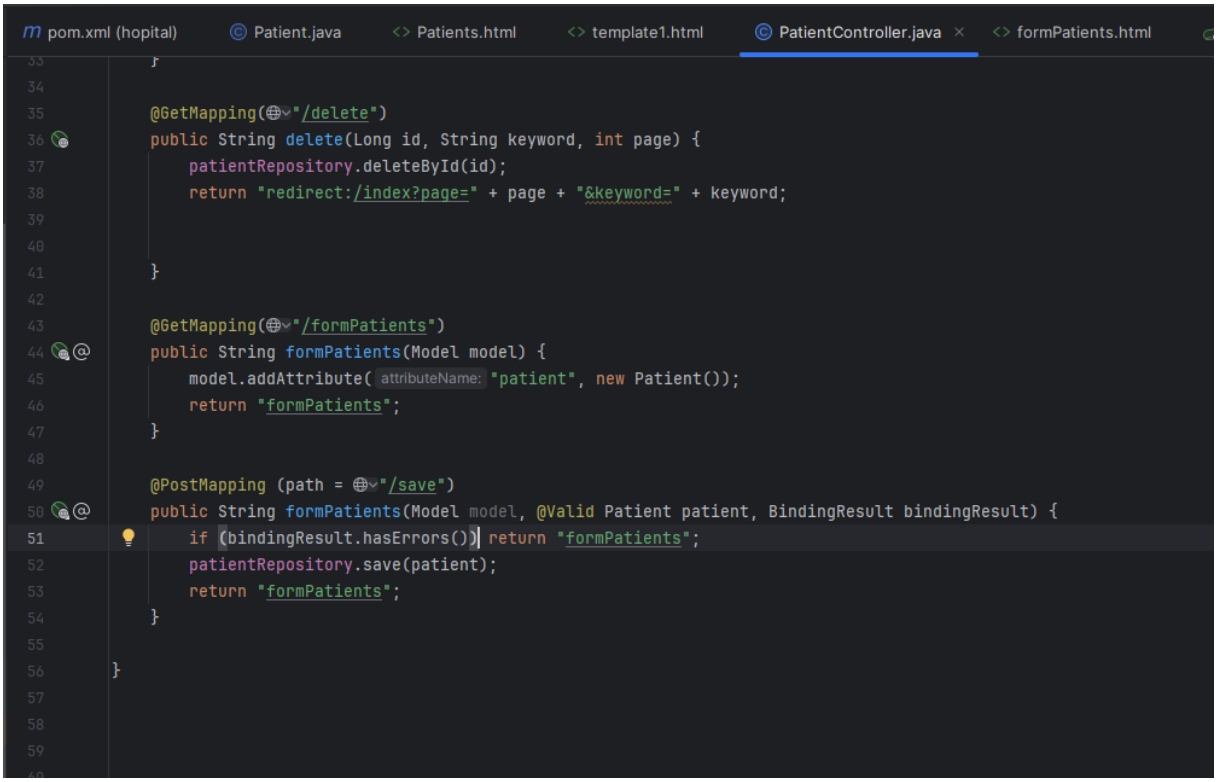
- Faire la validation des formulaires Ajouter les dépendances de spring boot validation.

```xml
<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>
```

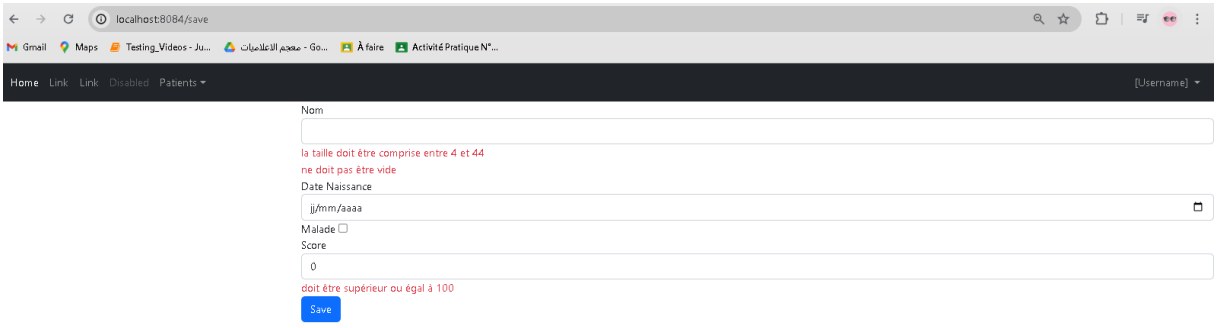
- Ajouter les annotation de validation.

```java
    }

    @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";
    }

}
```

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



D'où la validation est bien effectuée.



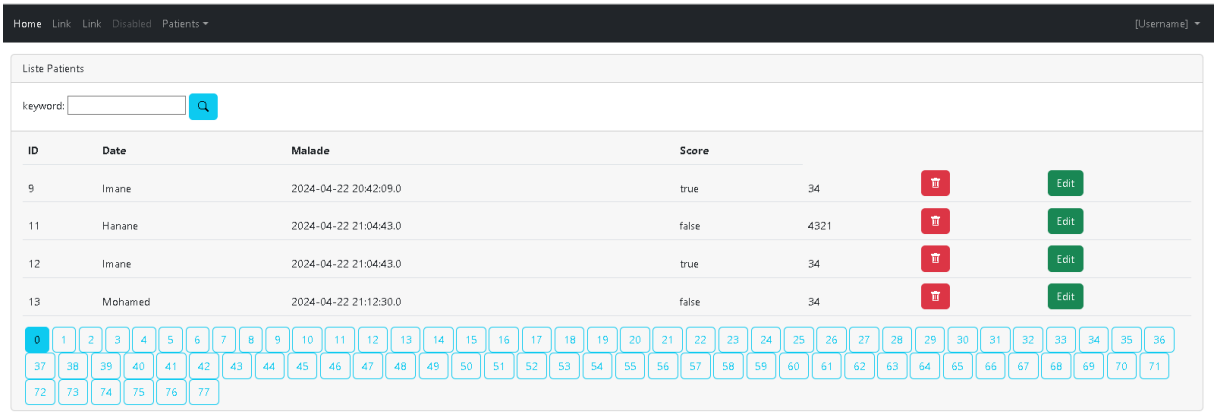
En ajoute le bouton Edit et ainsi la méthode EditPatient.

```
<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>
```

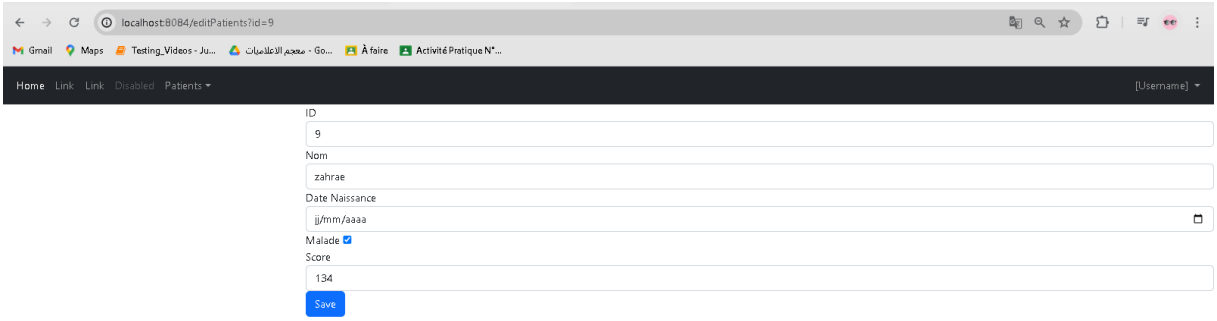
```
}

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

@PostMapping (path = "/save")
public String save(Model model, @Valid Patient patient, BindingResult bindingResult) {
    if (bindingResult.hasErrors()) return "formPatients";
    patientRepository.save(patient);
    return " redirect:/index";
}
@GetMapping("/editPatients")
public String editPatients(Model model , Long id) {
    Patient patient=patientRepository.findById(id).orElse( other: null);
    if (patient==null) throw new RuntimeException("Patient introuvable");
    model.addAttribute( attributeName: "patient",patient);
    return "editPatients";
}
}
```

```
<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}">
            <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>
```

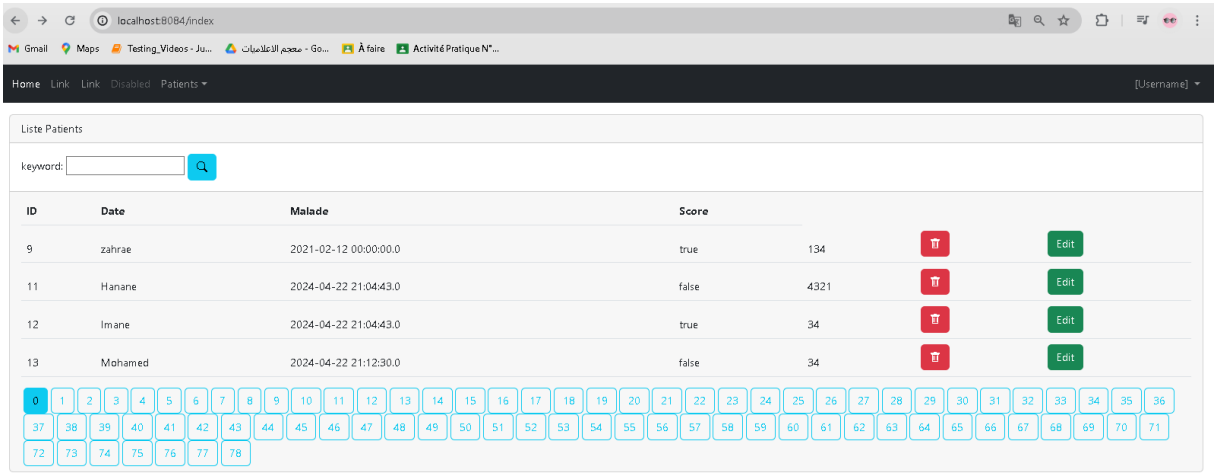
On va prendre id=9 on va essayer d'éditer les informations du patient .



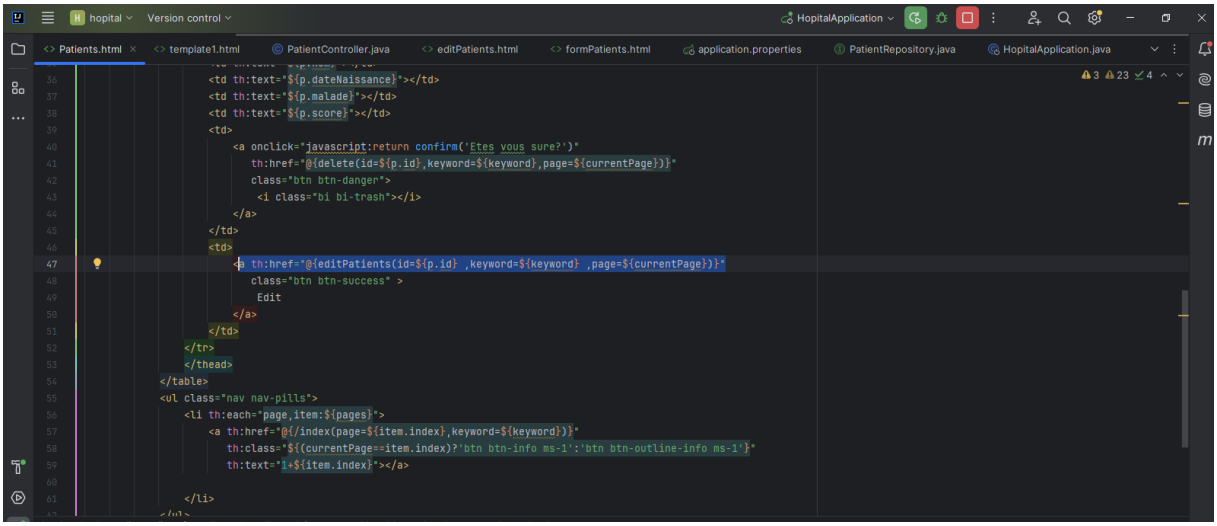
Editer les informations.



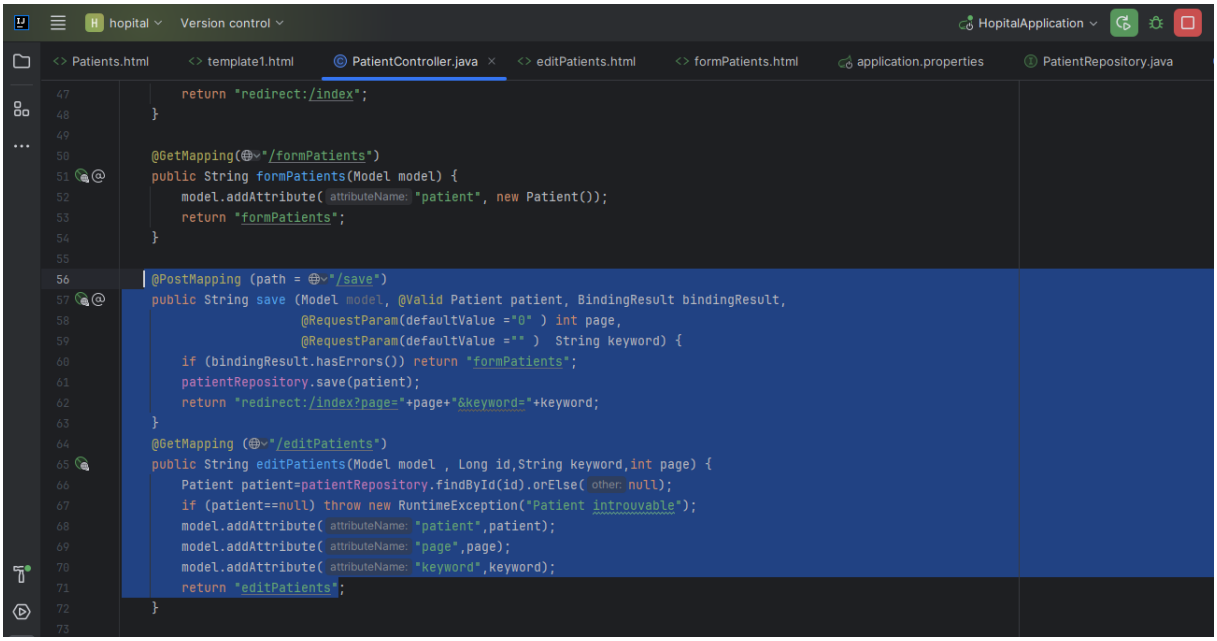
editPatient passe bien.



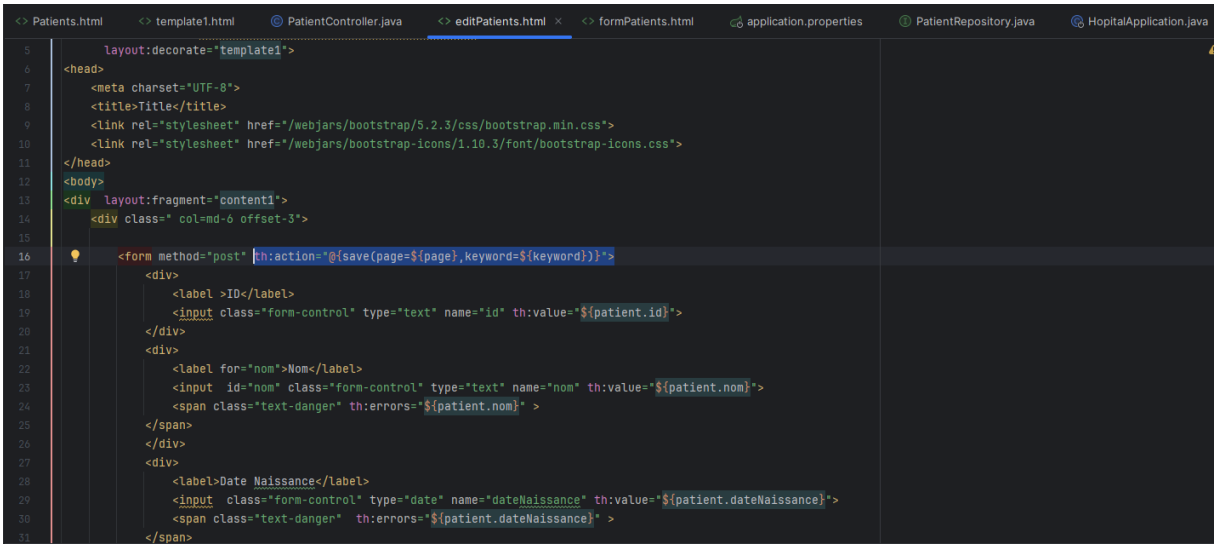
Pour rester dans la même page lorsque on met une modification des donnes de patient.



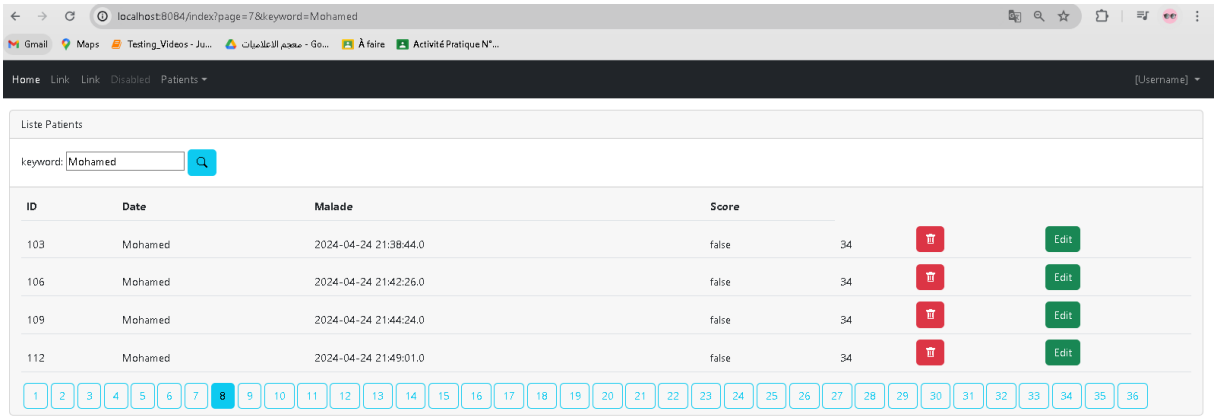
```
<td th:text="${p.dateNaissance}"></td>
<td th:text="${p.malade}"></td>
<td th:text="${p.score}"></td>
<td>
    <a onclick="javascript:return confirm('Etes vous sure?')"
       th:href="@{delete(id=${p.id},keyword=${keyword},page=${currentPage})}"
       class="btn btn-danger">
        <i class="bi bi-trash"></i>
    </a>
</td>
<td>
    <a th:href="@{editPatients(id=${p.id} ,keyword=${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>
```

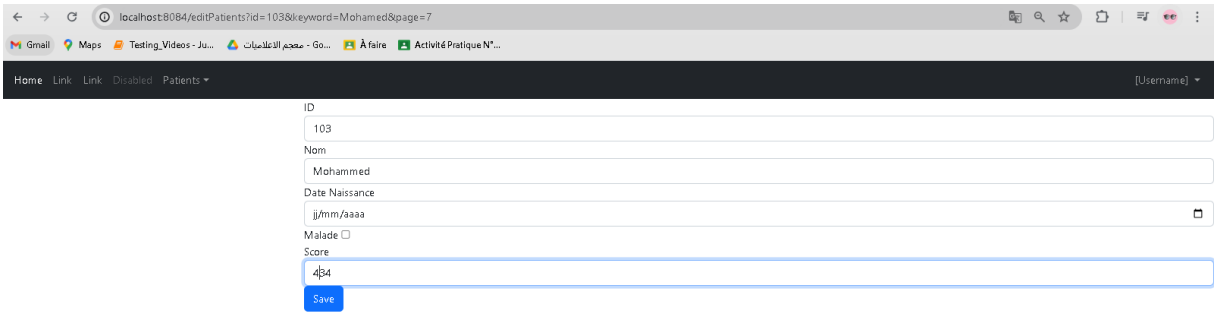


```
    return "redirect:/index";
}

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

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

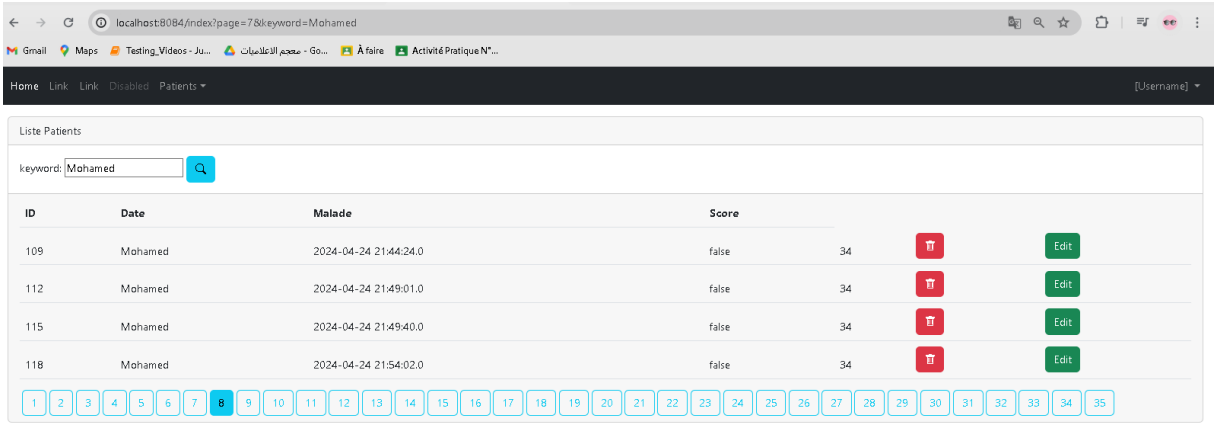


```
    layout:decorate="template1">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
    <link rel="stylesheet" href="/webjars/bootstrap-icons/1.10.3/font/bootstrap-icons.css">
</head>
<body>
<div  layout:fragment="content1">
    <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>
```

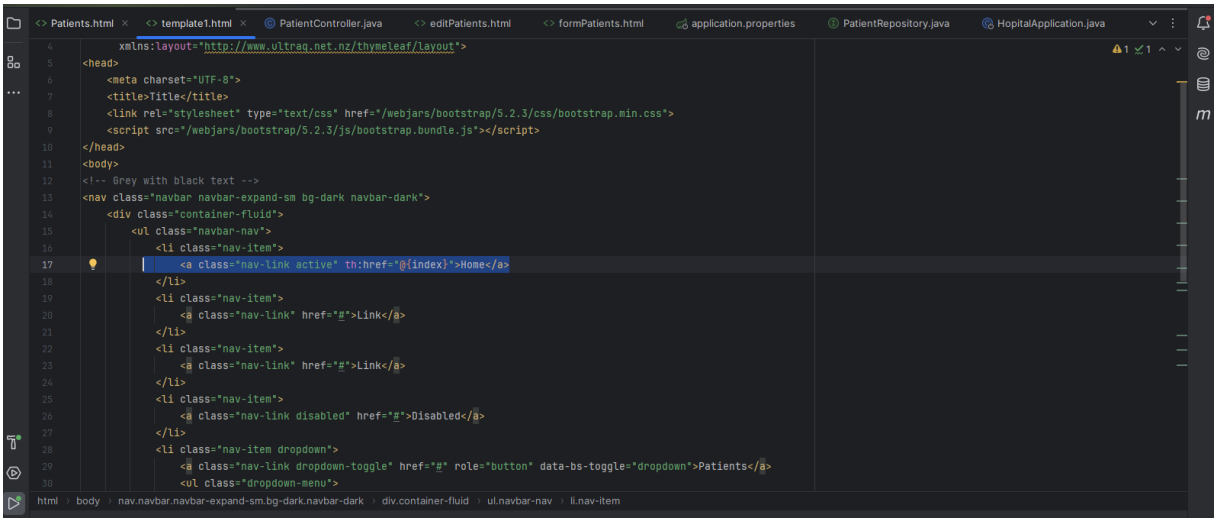




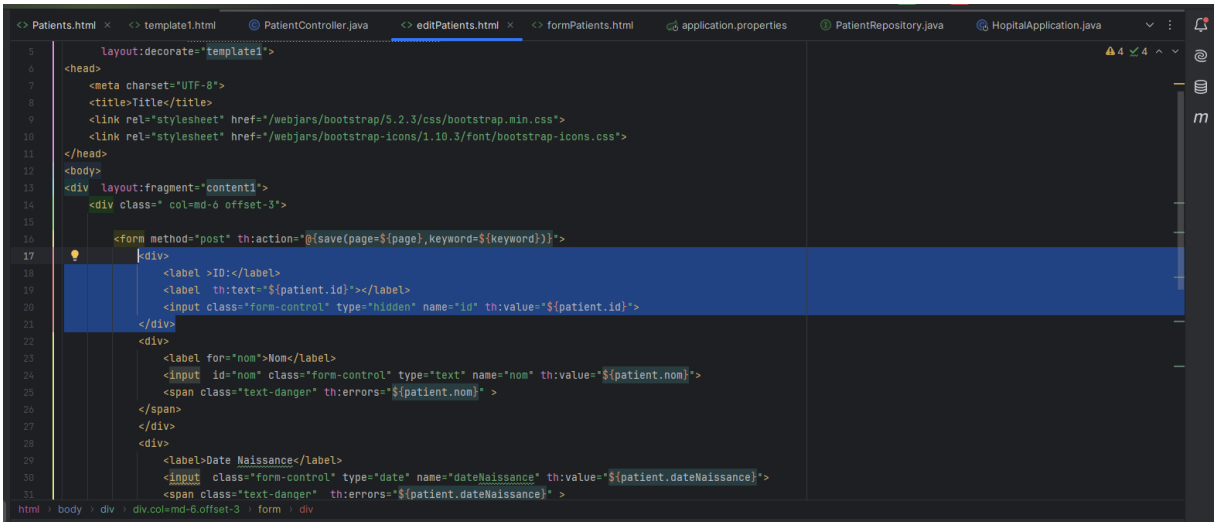
D’où on a rester dans la même page .



On ajouter un lien pour le Home :



On ne peux pas modifier id.

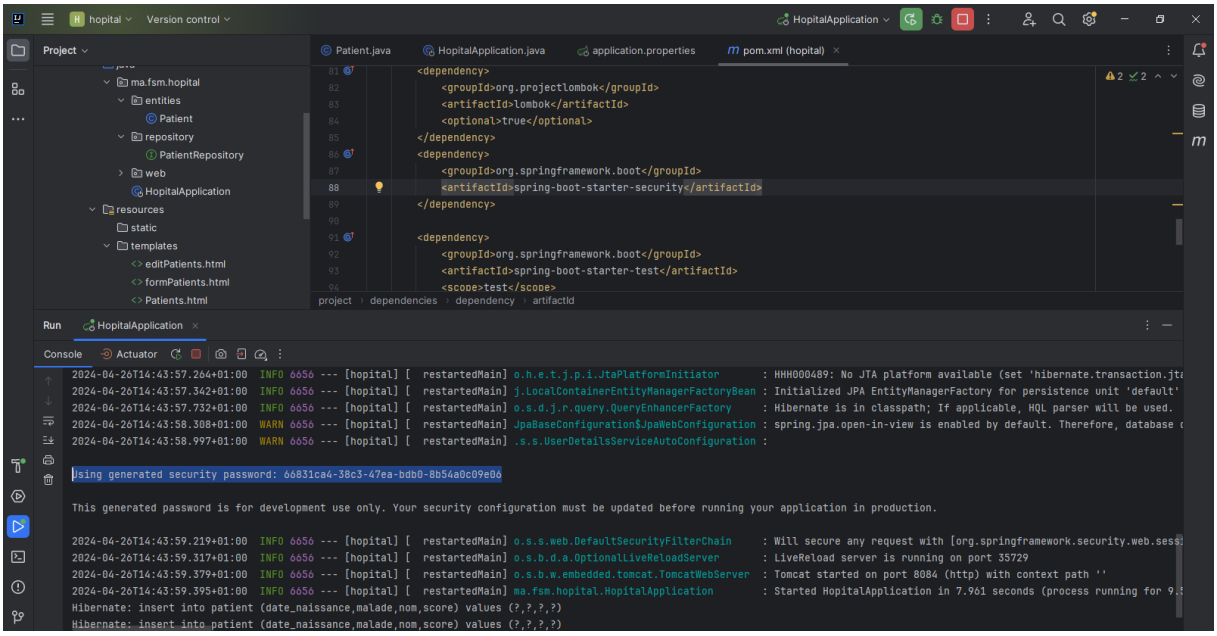


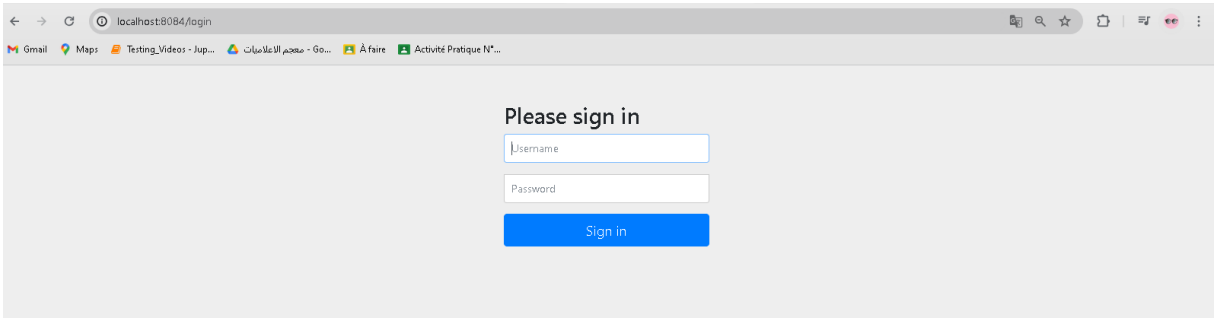
⇒ **Partie 3 :**

Sécurité avec Spring security :

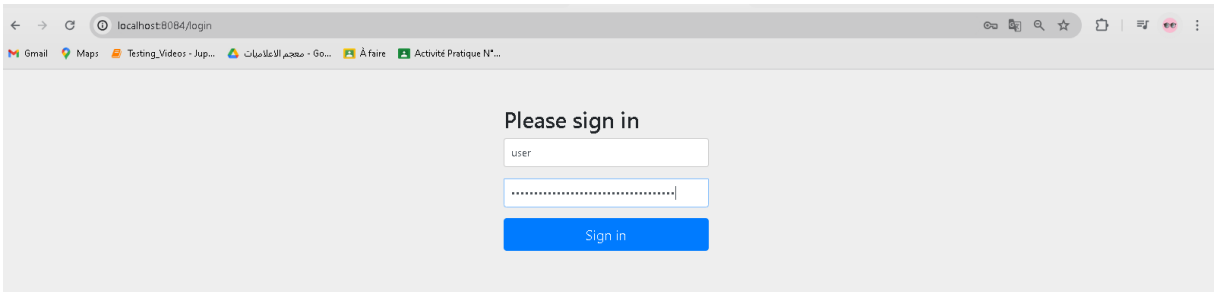
On ajoute les dépendances de spring security au fichier pom.xml.

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

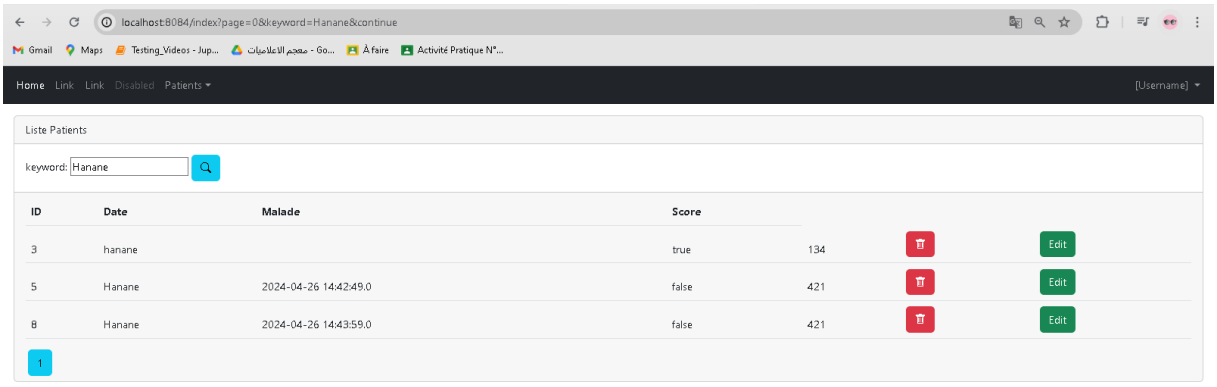




On sign in with username user and password genered by application.

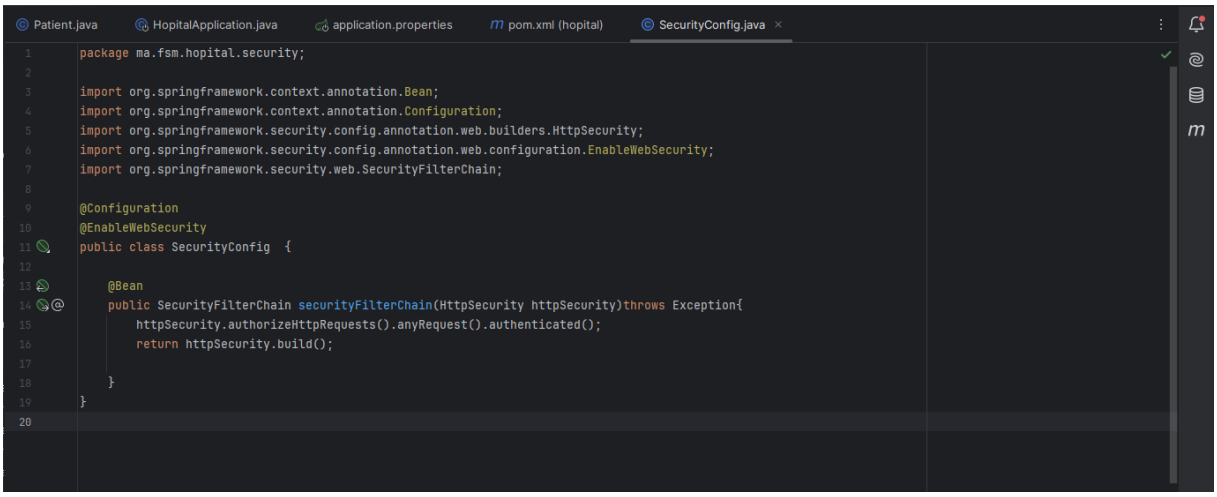


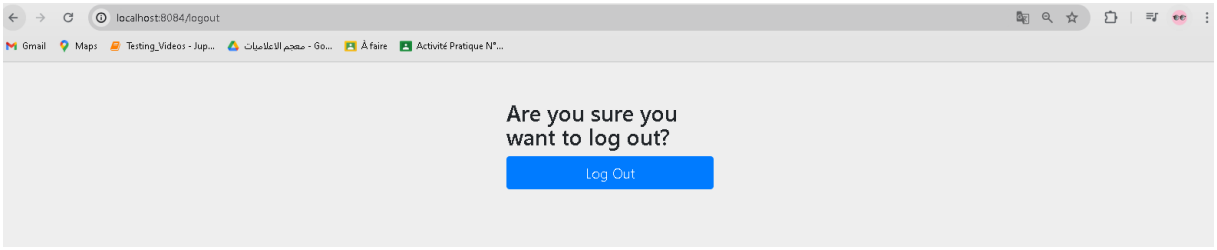
Après en accès a notre application avec succès.



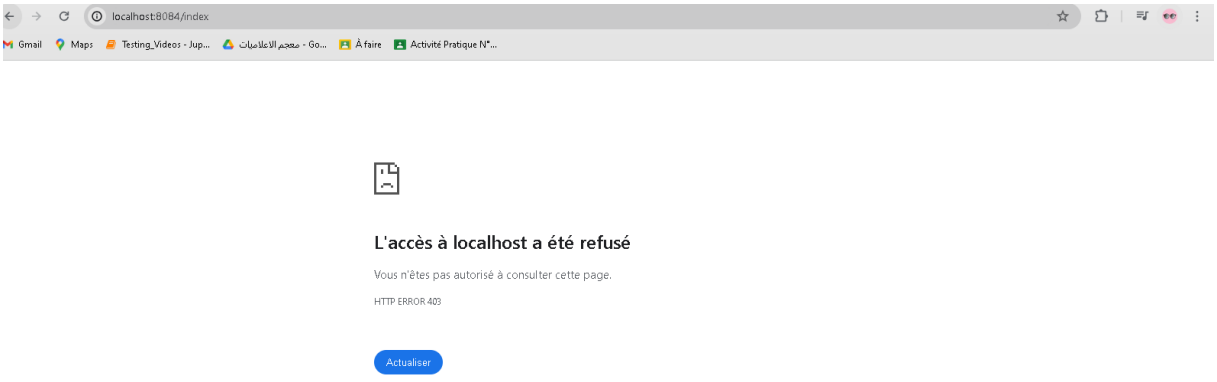
➢ Personnaliser la configuration de security.

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 :



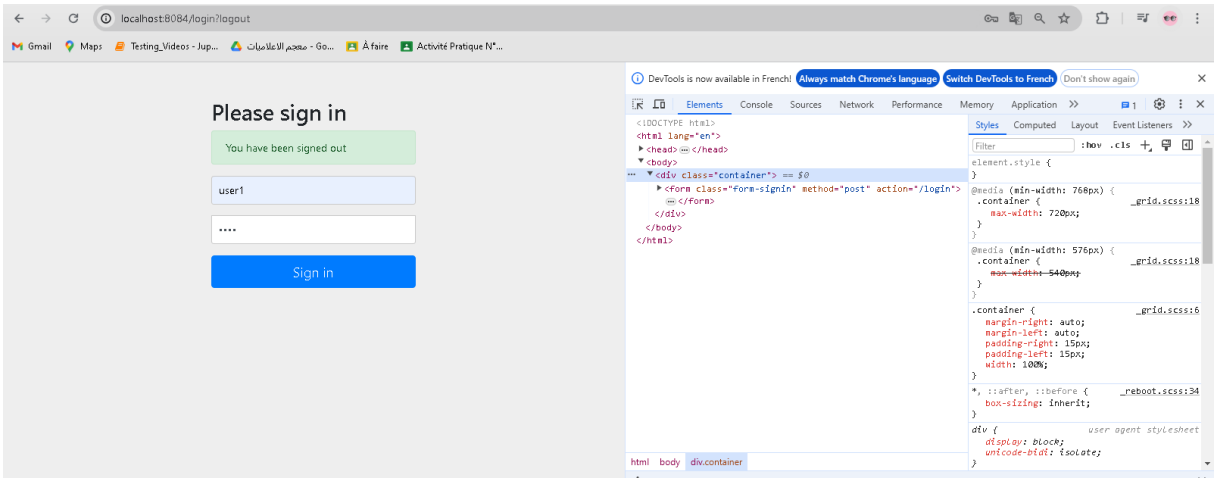


L'accès est refusé.



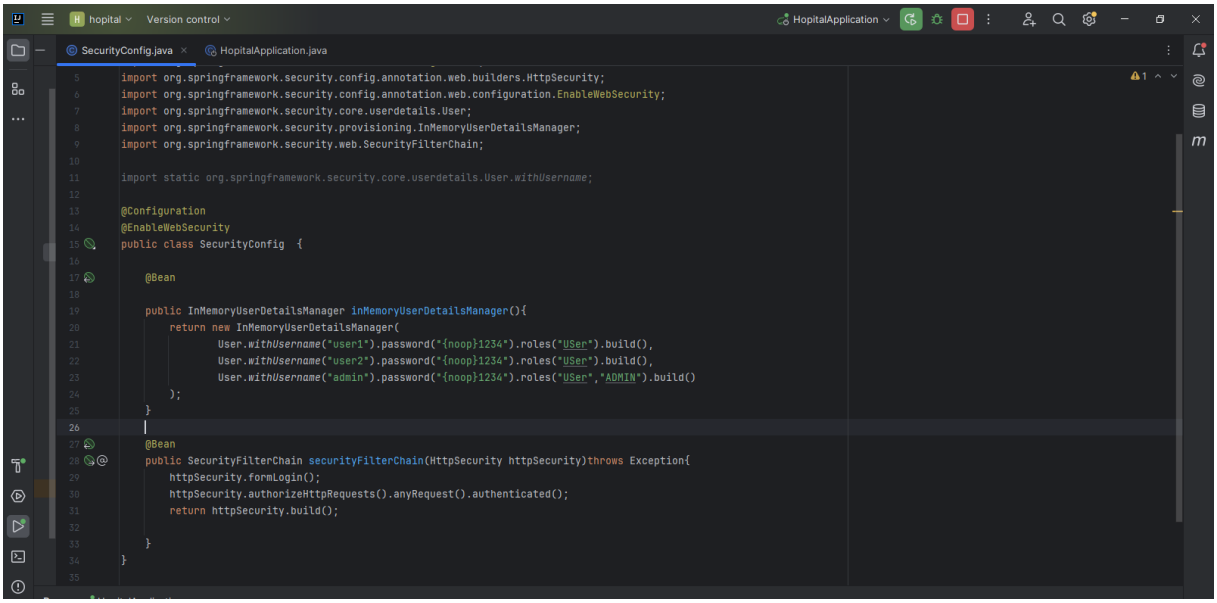
Pour afficher le formulaire d'authentification .

```java
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
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 SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity)throws Exception{
        httpSecurity.formLogin();
        httpSecurity.authorizeHttpRequests().anyRequest().authenticated();
        return httpSecurity.build();

    }
}
```

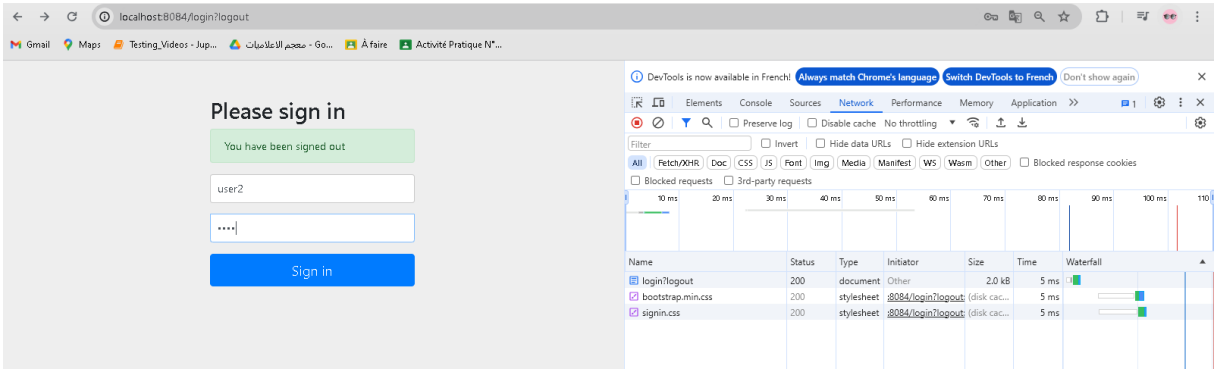


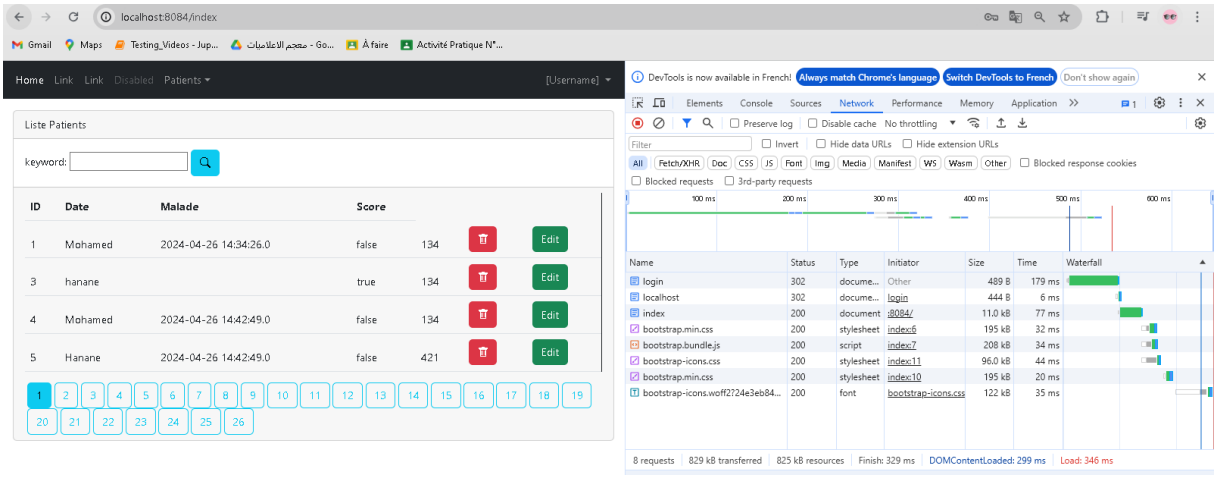
un objet de type InMemoryAuthentication, qui permet de préciser au mémoire les utilisateurs qui ont le droit d'accéder à l'application.

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

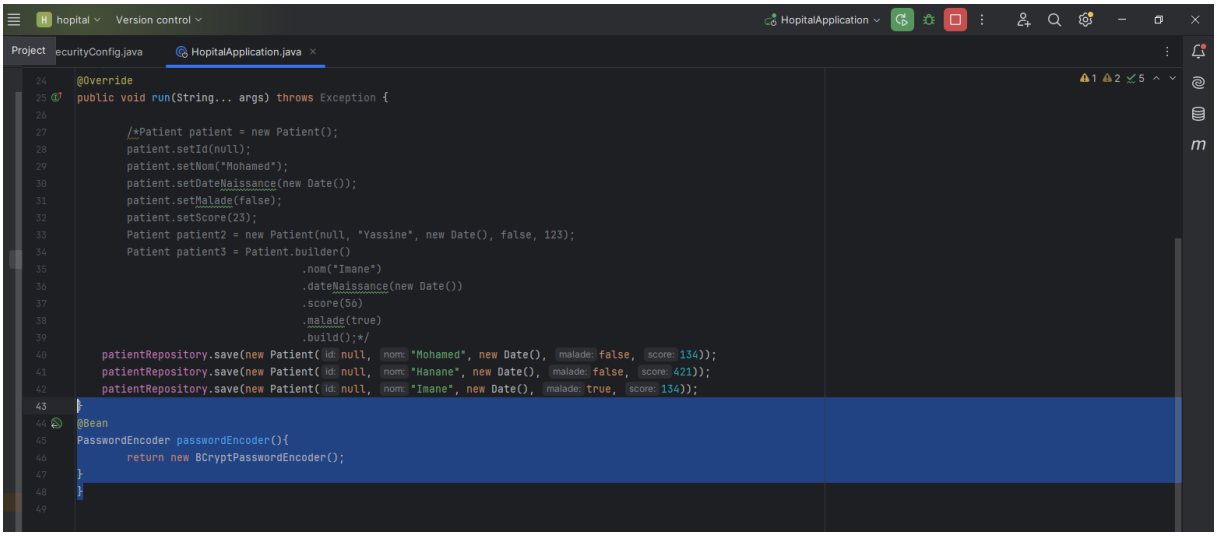


```java
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
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();

    }
}
```

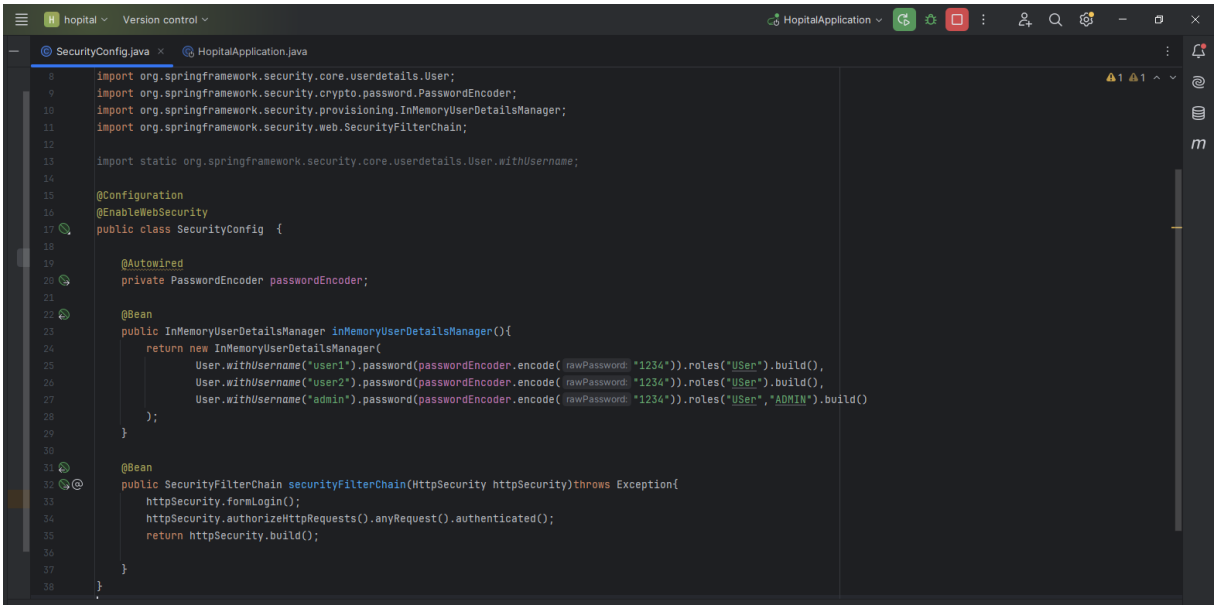




- Utilisant encoder BCrypt :

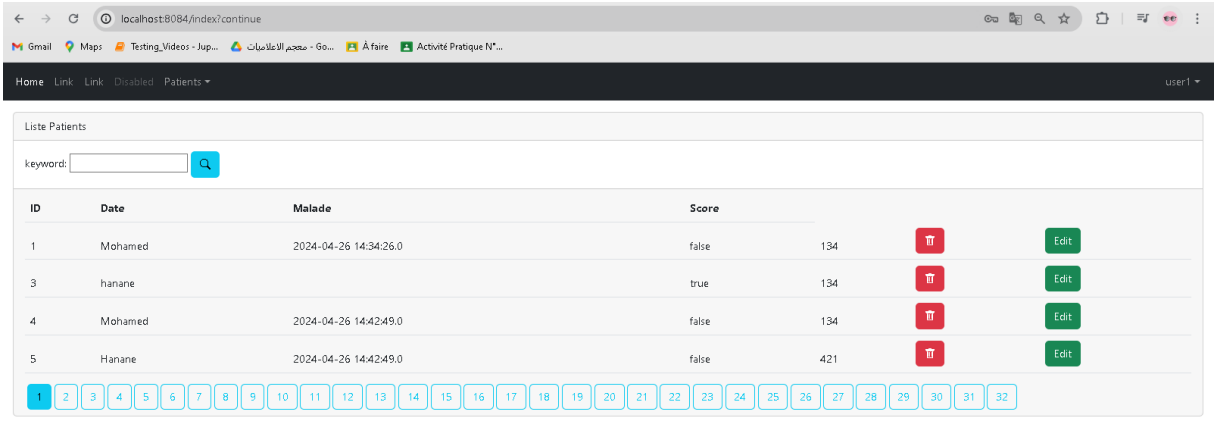


```java
@Override
public void run(String... args) throws Exception {

        /*Patient patient = new Patient();
        patient.setId(null);
        patient.setNom("Mohamed");
        patient.setDateNaissance(new Date());
        patient.setMalade(false);
        patient.setScore(23);
        Patient patient2 = new Patient(null, "Yassine", new Date(), false, 123);
        Patient patient3 = Patient.builder()
                                  .nom("Imane")
                                  .dateNaissance(new Date())
                                  .score(56)
                                  .malade(true)
                                  .build();*/
    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));
}
@Bean
PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder();
}
```



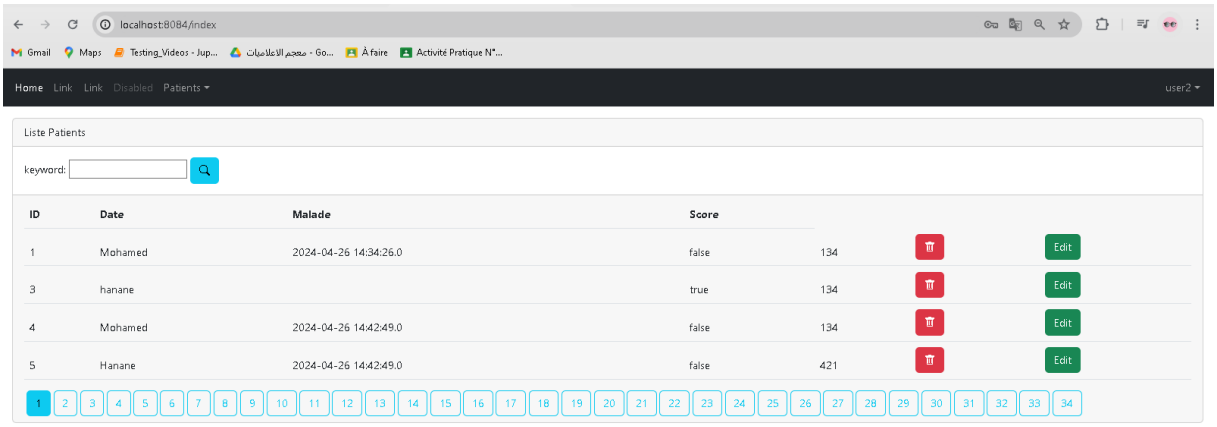
```java
import org.springframework.security.core.userdetails.User;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
import org.springframework.security.web.SecurityFilterChain;

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

@Configuration
@EnableWebSecurity
public class SecurityConfig  {

    @Autowired
    private PasswordEncoder passwordEncoder;

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

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

    }
}
```

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

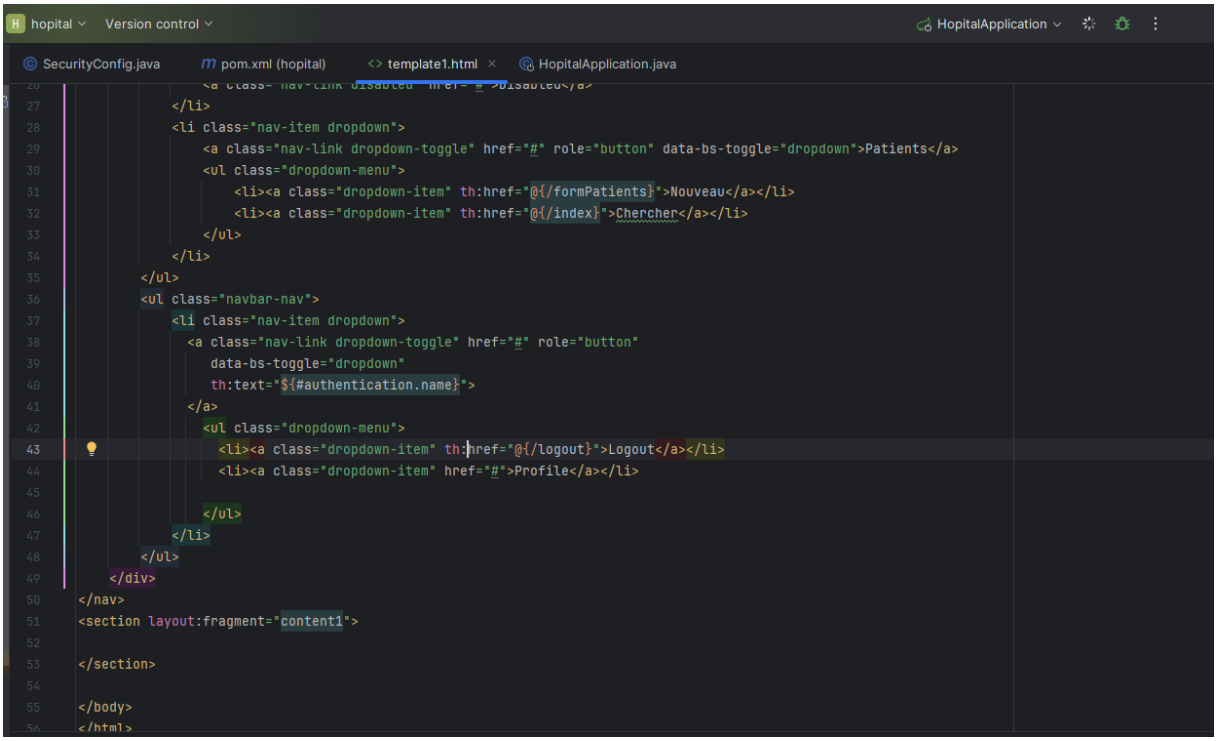


Pour se déconnecter, on a deux méthodes pour faire .


Please sign in
You have been signed out
user2
Sign in


localhost:8084/index
Home Link Link Disabled Patients
user2
Liste Patients
keyword:
ID Date Malade Score
1 Mohamed 2024-04-26 14:34:26.0 false 134 Edit
3 hanane true 134 Edit
4 Mohamed 2024-04-26 14:42:49.0 false 134 Edit
5 Hanane 2024-04-26 14:42:49.0 false 421 Edit

Méthode 1 :


hopital
Version control
HopitalApplication
SecurityConfig.java
pom.xml (hopital)
template1.html
HopitalApplication.java

<li class="nav-item dropdown">
Patients
<ul class="dropdown-menu">
Nouveau
Chercher

<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}">

<ul class="dropdown-menu">
Logout
Profile

</nav>
<section layout:fragment="content1">
</section>
</body>
</html>

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

SecurityConfig.java | pom.xml (hopital) | template1.html | PatientController.java | HopitalApplication.java

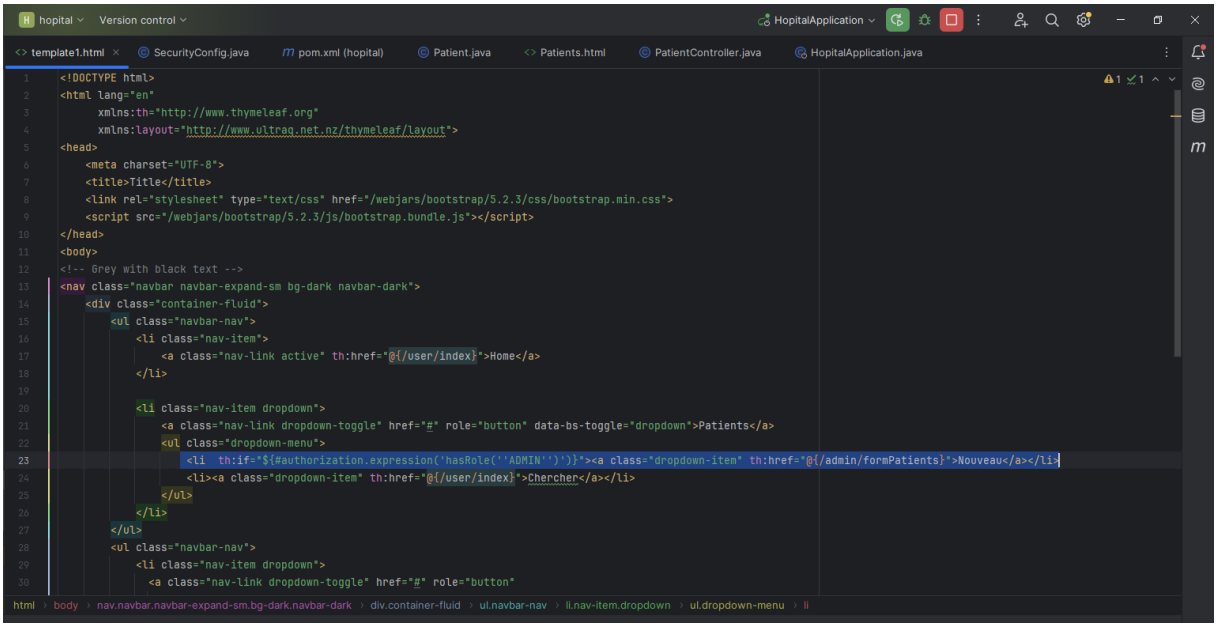
```
            </li>
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">Patients</a>
                <ul class="dropdown-menu">
                    <li><a class="dropdown-item" th:href="@{/formPatients}">Nouveau</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>
</nav>
<section layout:fragment="content1">
```

SecurityConfig.java | pom.xml (hopital) | template1.html | PatientController.java | HopitalApplication.java

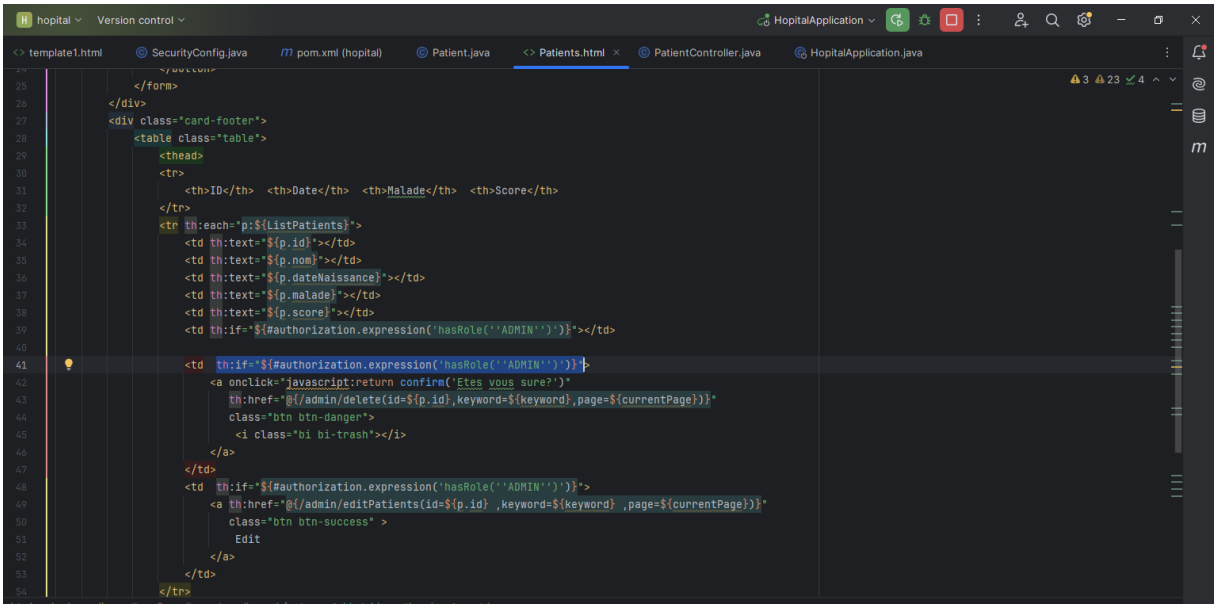
```
        model.addAttribute( attributeName: "patient", new Patient());
        return "formPatients";
    }

    @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.

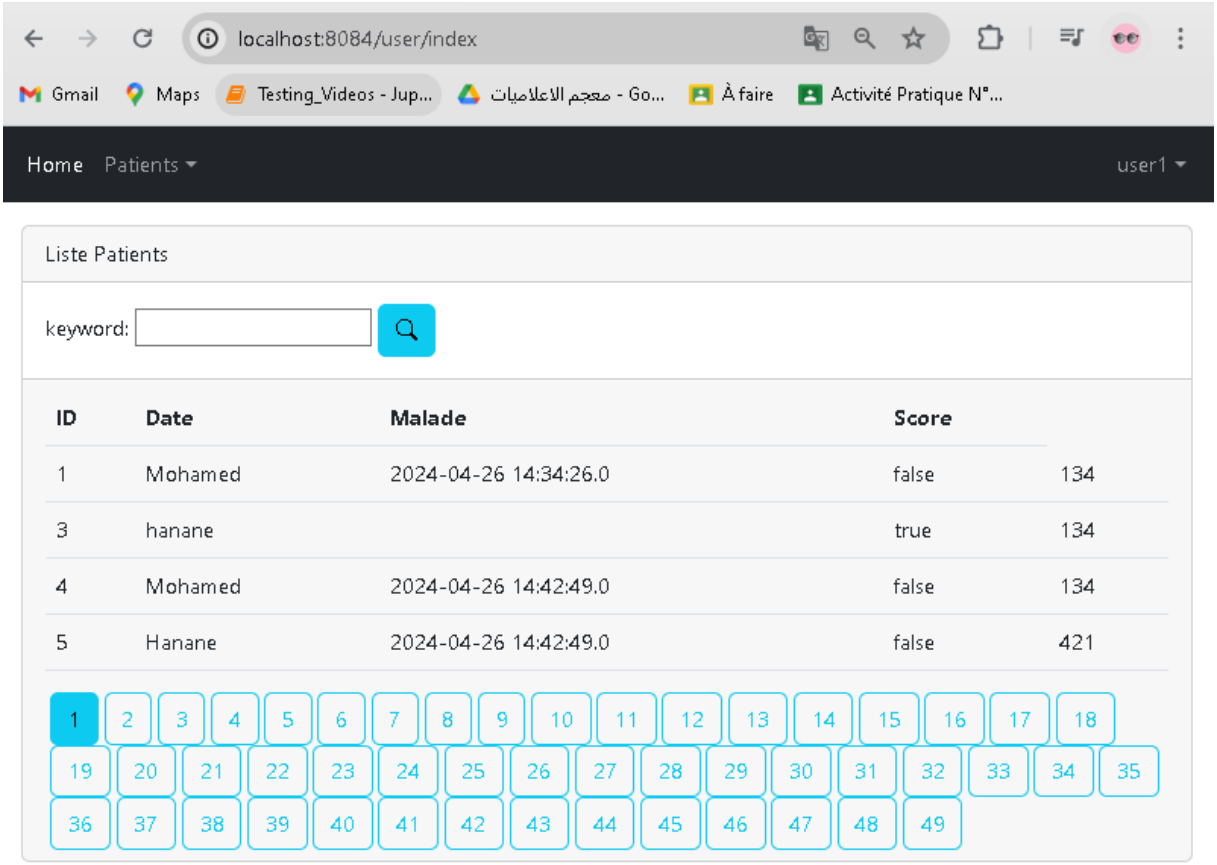
Voilà les boutons sont retirés lorsqu'on se connecte en tant que user.



```
<!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"
```

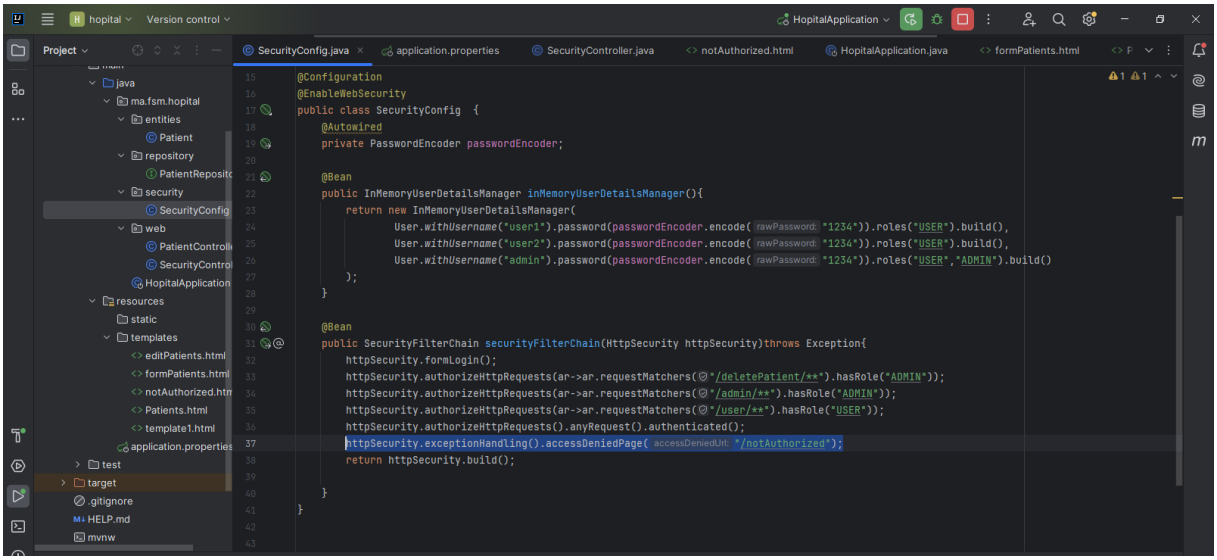


```
        </form>
    </div>
    <div class="card-footer">
        <table class="table">
            <thead>
            <tr>
                <th>ID</th>  <th>Date</th>  <th>Malade</th>  <th>Score</th>
            </tr>
            <tr th:each="p:${ListPatients}">
                <td th:text="${p.id}"></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>
                <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>
```

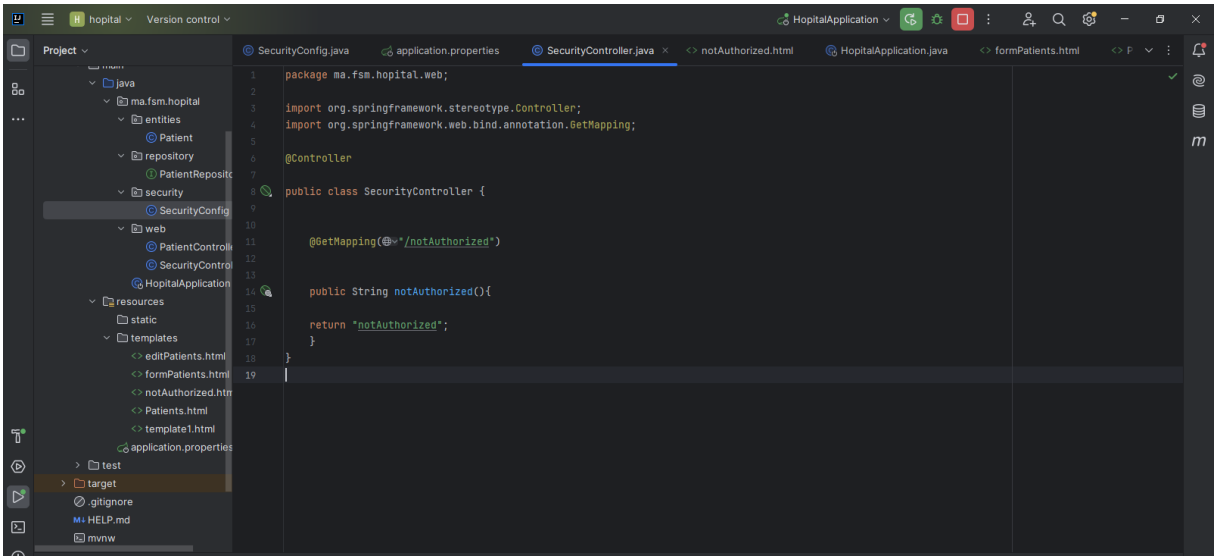


Lorsque user1 veut supprime un patient application génère un message d'erreur comme quoi vous n'avez pas le droit de faire cette fonctionnalité.

En ajoute la configuration suivants.

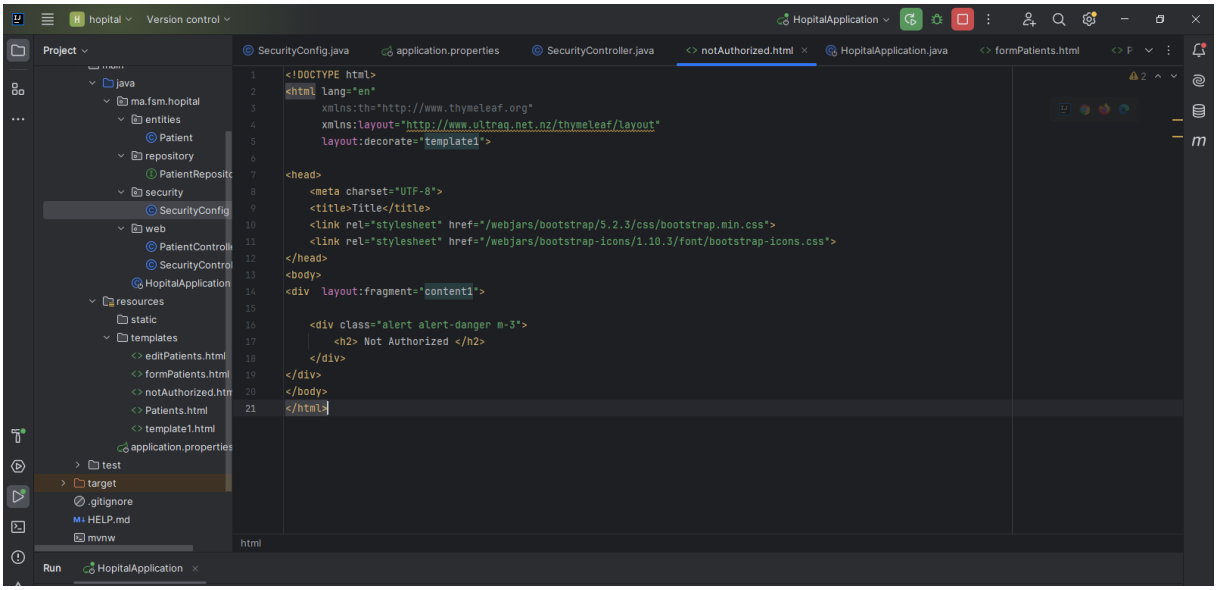


La Classe SecurityController.



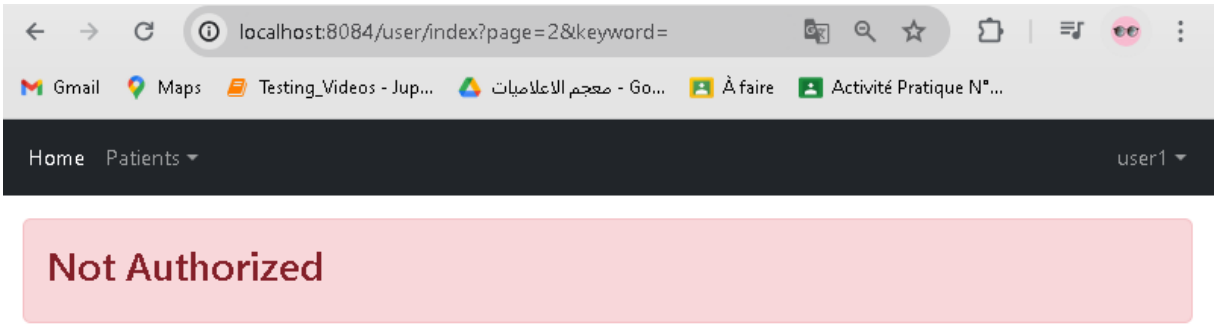
```java
package ma.fsm.hopital.web;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller

public class SecurityController {


    @GetMapping("/notAuthorized")


    public String notAuthorized(){

    return "notAuthorized";
    }
}
```

Le fichier notAuthorized.html.

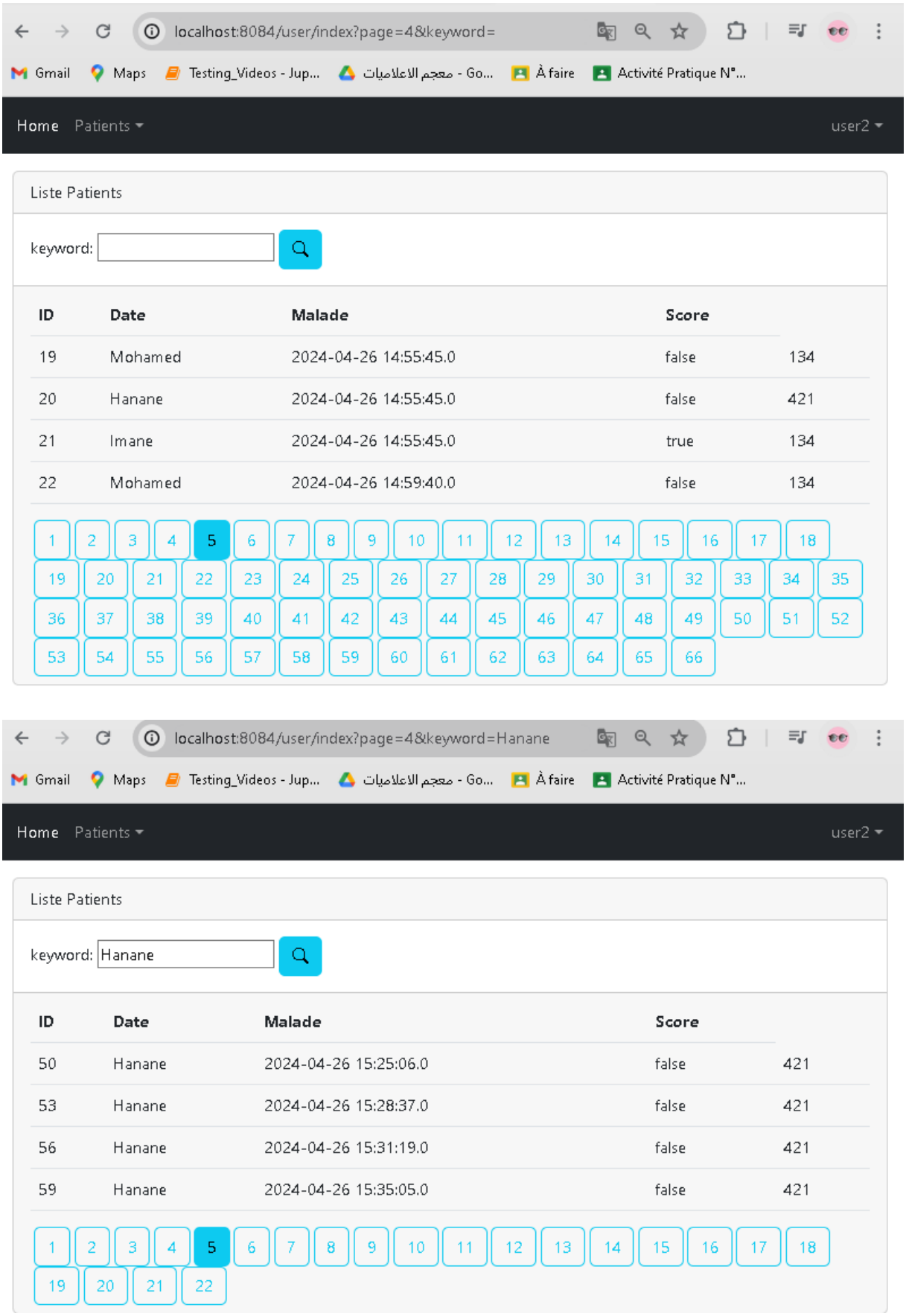


```html
<!DOCTYPE html>
<html lang="en"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="template1">

<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="content1">

    <div class="alert alert-danger m-3">
        <h2> Not Authorized </h2>
    </div>
</div>
</body>
</html>
```

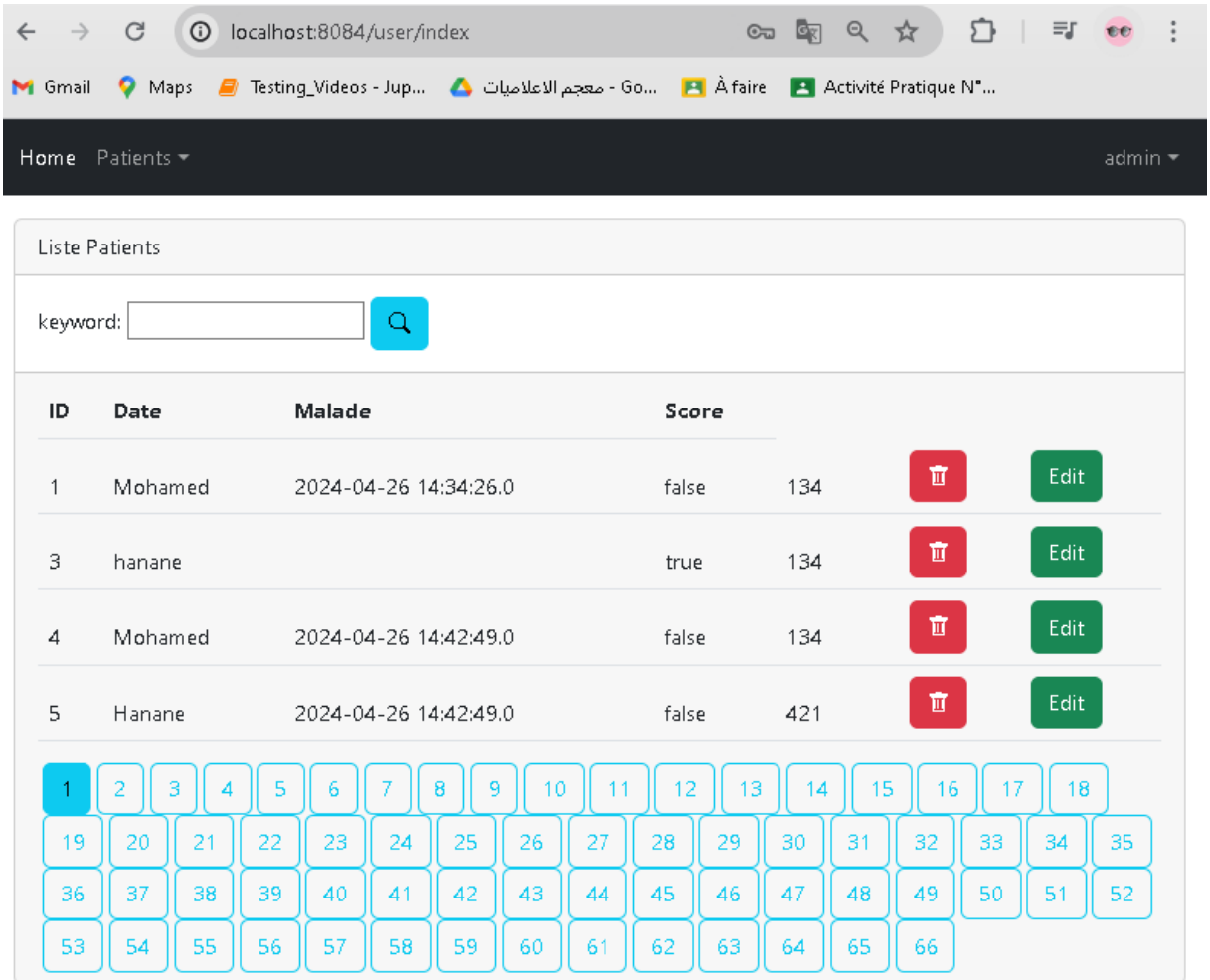
En exécute l'application le message suivant s' affiche.

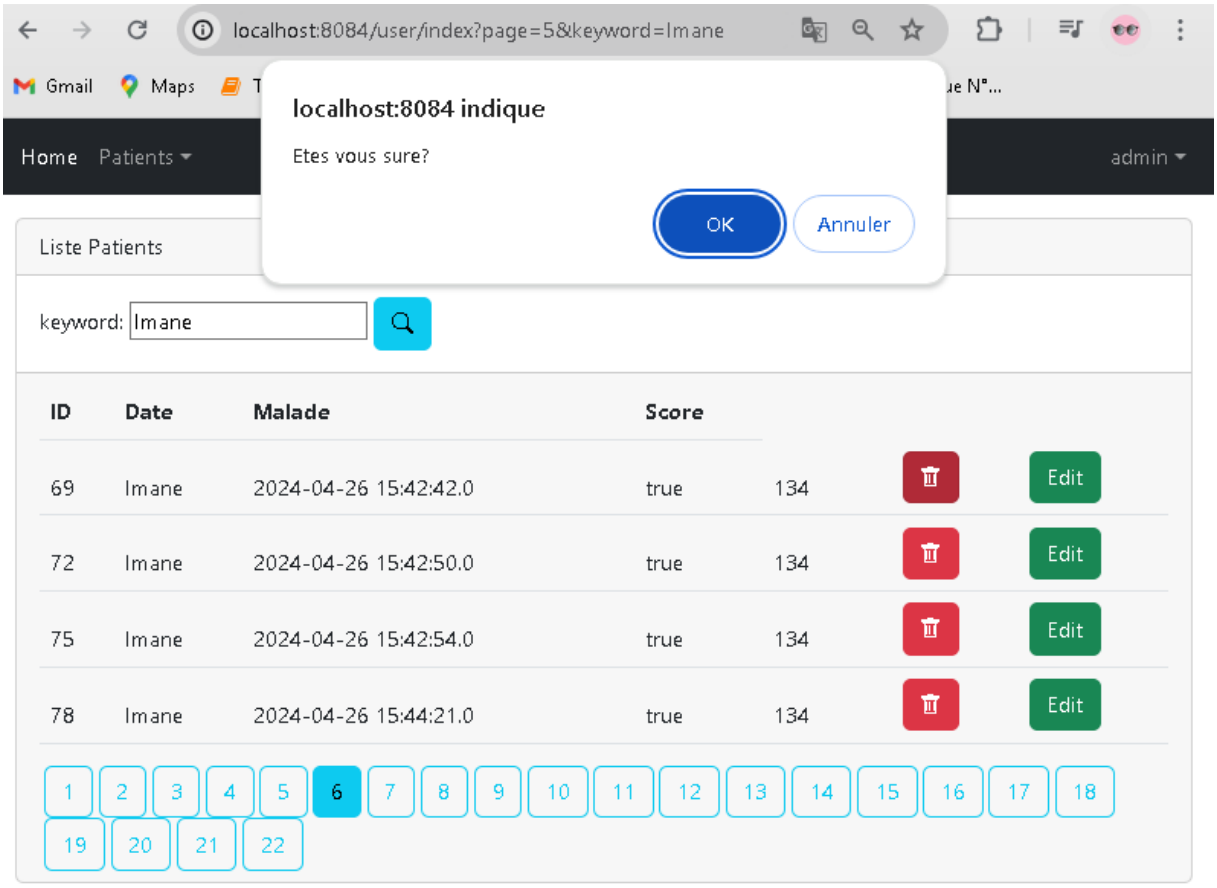


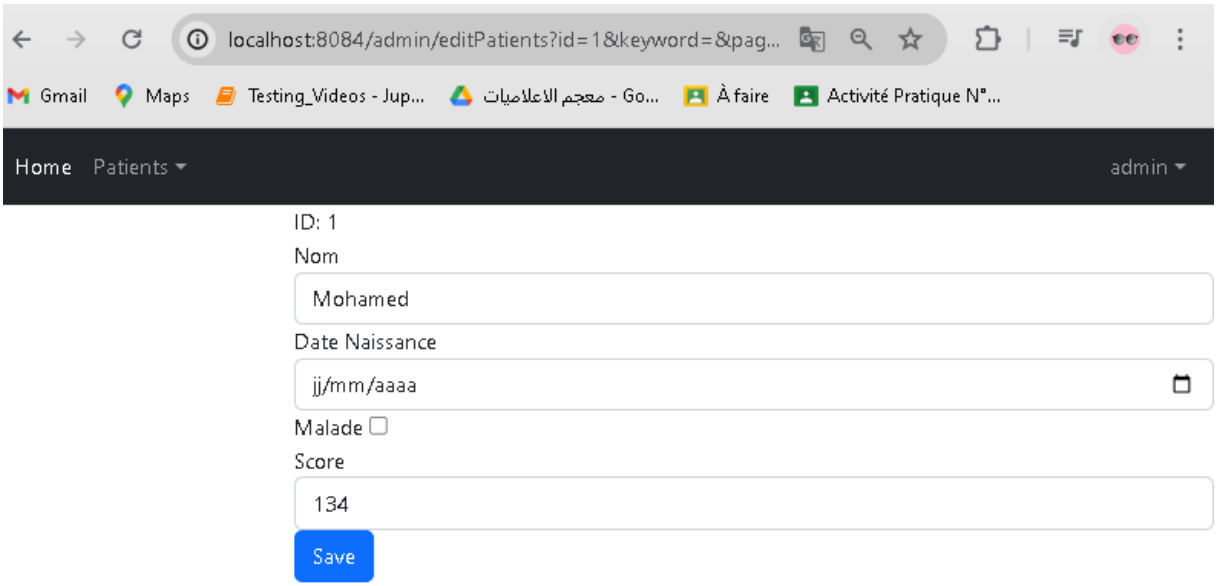
- Lorsque j'authentifier comme user j'ai pas le droit de supprimer ni d'éditer le patient mais je le droit de chercher patient et de faire pagination .

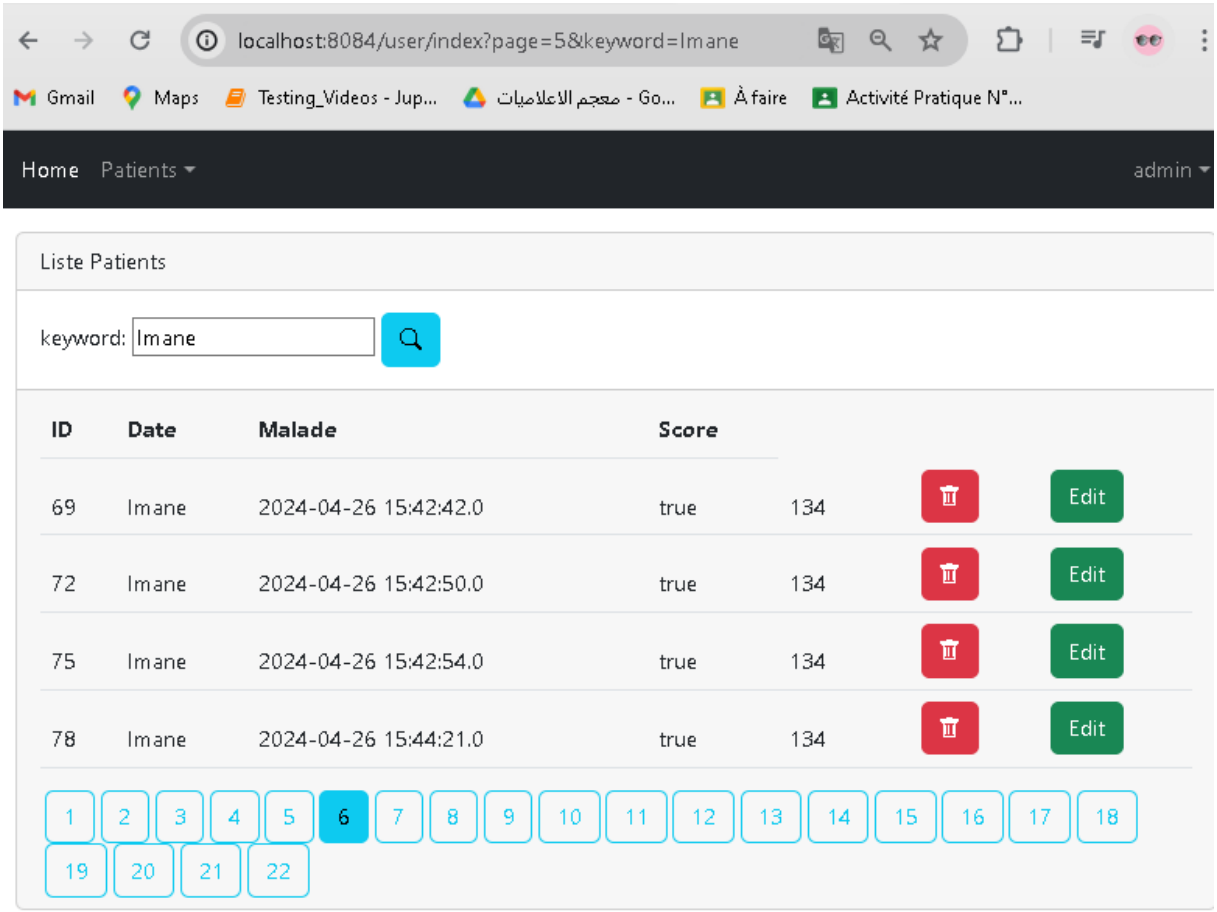


- Lorsque j'authentifier comme admin j'ai le droit de supprimer et éditer et chercher faire pagination .


localhost:8084/user/index
Gmail
Maps
Testing_Videos - Jup...
معجم 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 Edit
3 hanane true 134 Edit
4 Mohamed 2024-04-26 14:42:49.0 false 134 Edit
5 Hanane 2024-04-26 14:42:49.0 false 421 Edit
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
localhost:8084 indique
Etes vous sure?
OK
Annuler
Home
Patients
admin
Liste Patients
keyword: Imane
ID
Date
Malade
Score
69 Imane 2024-04-26 15:42:42.0 true 134 Edit
72 Imane 2024-04-26 15:42:50.0 true 134 Edit
75 Imane 2024-04-26 15:42:54.0 true 134 Edit
78 Imane 2024-04-26 15:44:21.0 true 134 Edit
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
19 20 21 22





- Personnaliser le formulaire d'authentification.

Création du Ficher login.html :

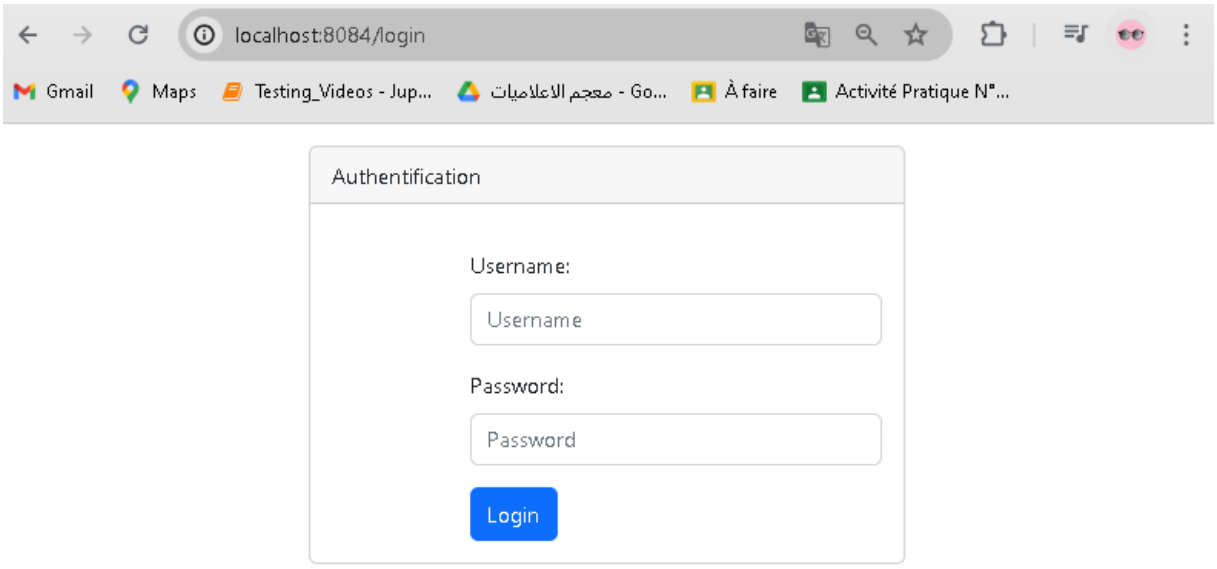
```
SecurityConfig.java × SecurityController.java <> login.html × <> notAuthorized.html HopitalApplication.java <> formPatients.html <> Patients.html
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Authentification</title>
    <link rel="stylesheet" href="/webjars/bootstrap/5.2.3/css/bootstrap.min.css">
    <link rel="stylesheet" href="/webjars/bootstrap-icons/1.10.3/font/bootstrap-icons.css">
</head>
<body>
<div  class="row mt-3">
    <div class="col-md-6 offset-3" >
        <div class="card">
            <div class="card-header"> Authentification </div>
            <div class="card-body">
                <form method="post" th:action="@{/login}">
                    <div class=" col=md-6 offset-3">
                      <form method="post" th:action="@{save}">
                          <div class="mb-3 mt-3">
                            <label for="username" class="form-label">Username:</label>
                            <input  id="username" class="form-control" type="text" placeholder="Username" name="username" >
                          </div>
                          <div class="mb-3 mt-3">
                            <label for="password" class="form-label">Password:</label>
                            <input  id="password" class="form-control" type="text" placeholder="Password" name="password" >
                          </div>
                          <button type="submit" class="btn btn-primary">Login</button>
                      </form>
                    </div>
html › body › div.row.mt-3 › div.col-md-6.offset-3 › div.card › div.card-body › form › div.col=md-6.offset-3
```

La classe SecurityController :

```
SecurityConfig.java  SecurityController.java ×  login.html  notAuthorized.html  HopitalApplication.java

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller

public class SecurityController {


    @GetMapping("/notAuthorized")


    public String notAuthorized(){

    return "notAuthorized";
    }

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

```java
public class SecurityConfig  {
    @Autowired
    private PasswordEncoder passwordEncoder;

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

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity)throws Exception{
        httpSecurity.formLogin().loginPage("/login").permitAll();
        httpSecurity.authorizeHttpRequests().requestMatchers("/webjars/**").permitAll();
        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();

    }
}
```

Notre nouvelle formulaire est le suivant :



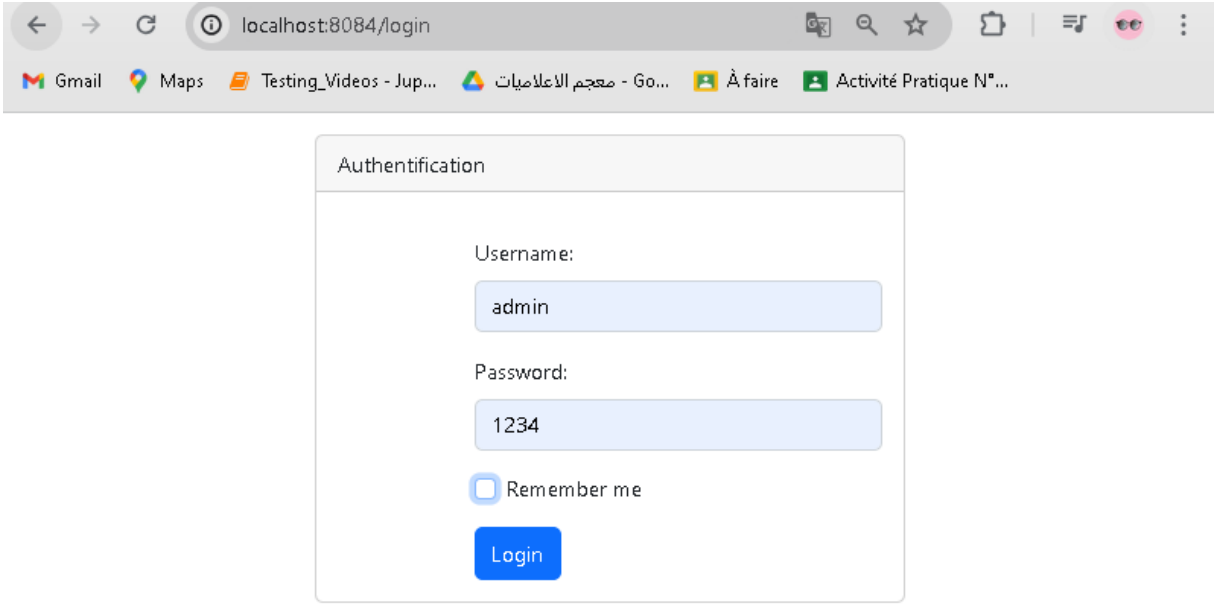
Si on veut ajouter une case à cocher « Remember me » :

Ajoutons ces lignes au Ficher login.html :

```
<form method="post" th:action="@{/login}">
    <div class=" col-md-6 offset-3">
      <form method="post" th:action="@{save}">
        <div class="mb-3 mt-3">
          <label for="username" class="form-label">Username:</label>
          <input  id="username" class="form-control" type="text" placeholder="Username" name="username" >
        </div>
        <div class="mb-3 mt-3">
          <label for="password" class="form-label">Password:</label>
          <input  id="password" class="form-control" type="text" placeholder="Password" name="password" >
        </div>
        <div class="form-check mb-3">
            <label class="form-check-label">
                <input  class="form-check-input" type="checkbox"  name="remember-me"> Remember me
            </label>
        </div>
        <button type="submit" class="btn btn-primary">Login</button>
      </form>
    </div>
  </form>
 </div>
</div>
</div>
</body>
</html>
```

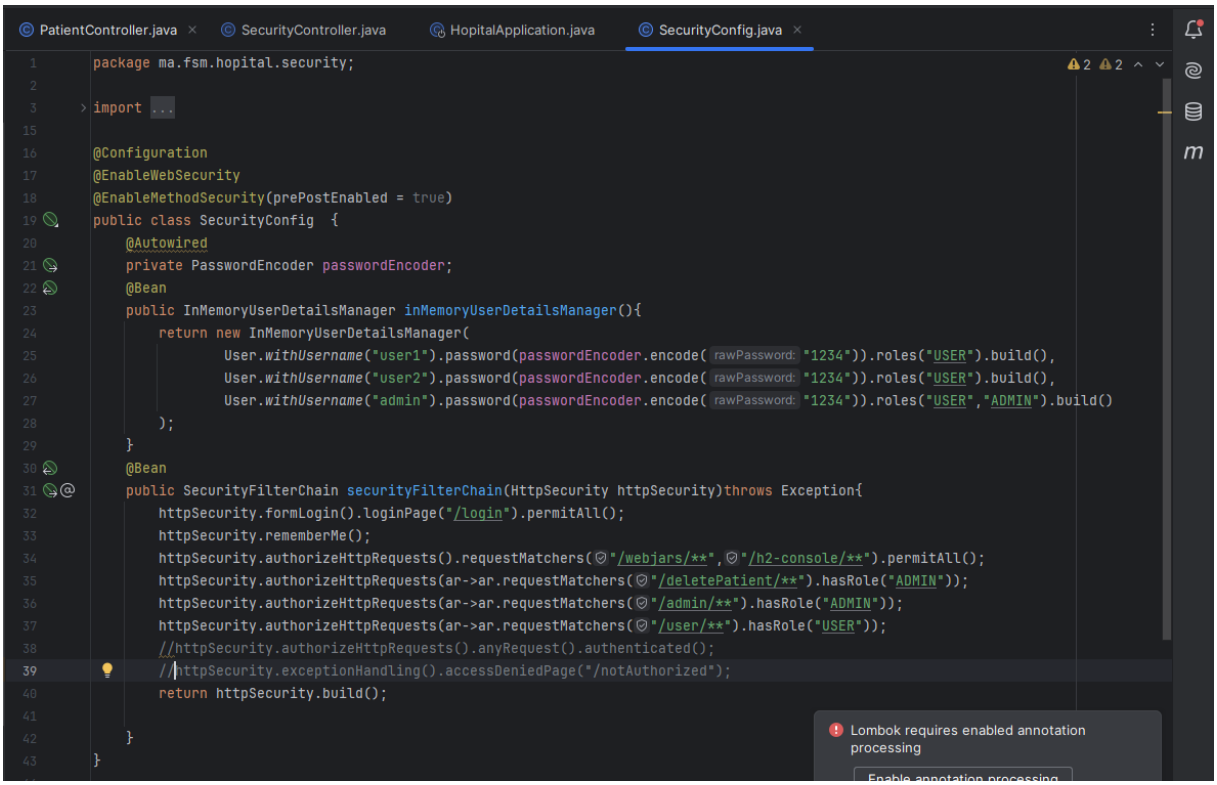
```
public class SecurityConfig  {
    @Autowired
    private PasswordEncoder passwordEncoder;

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

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

    }
}
```

Voilà les résultats :

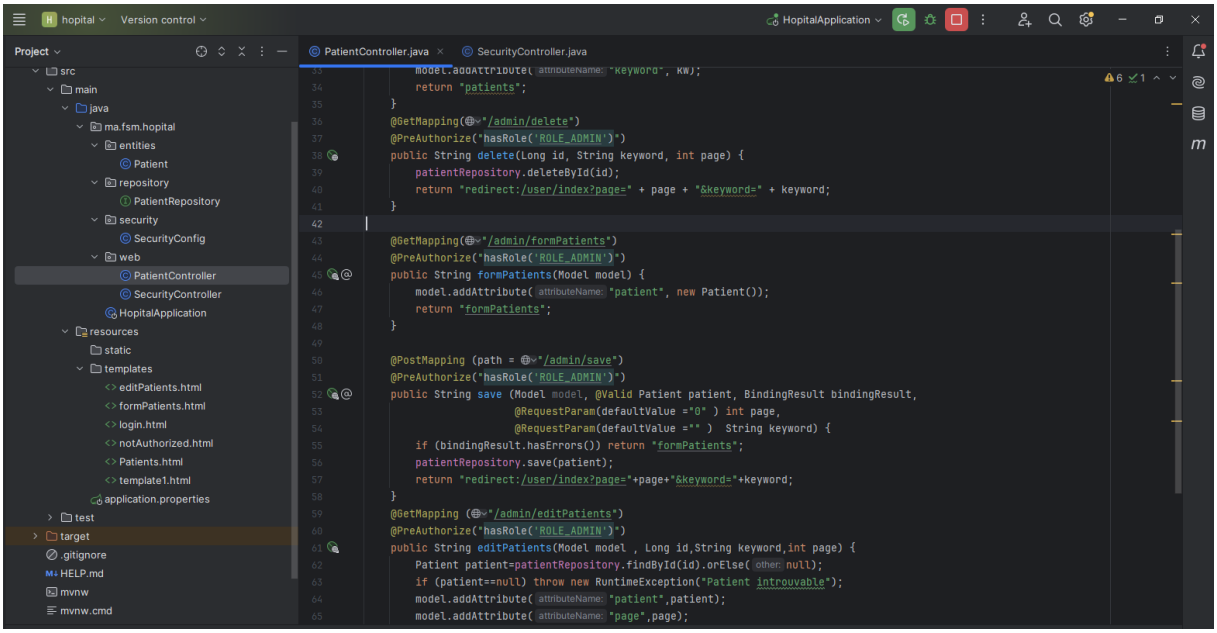


- On va stocker les données dans la base de données au lieu d'une mémoire .

pour protéger les ressources soit par annotation @EnableMethodSecurity :

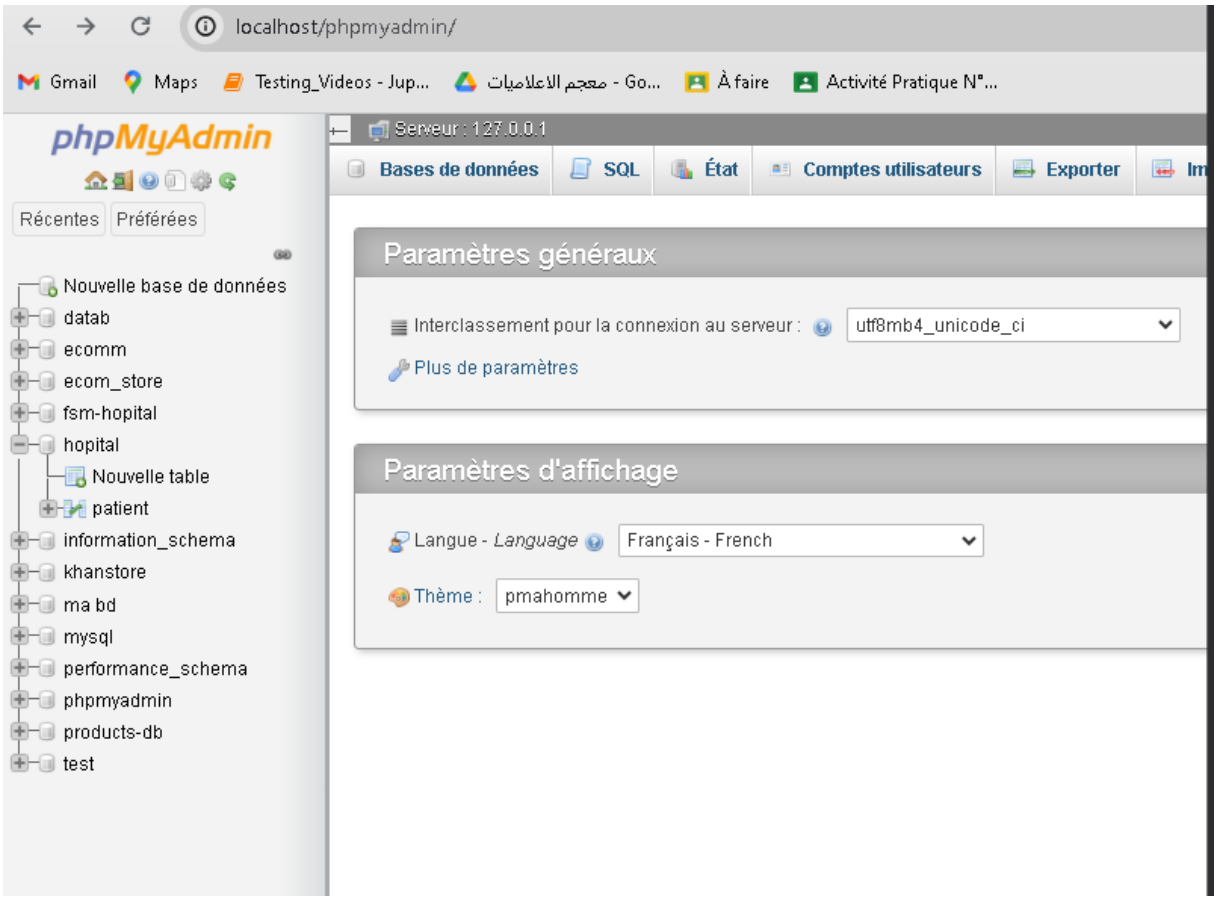


Soit on utilise l'annotation @PreAuthorize :

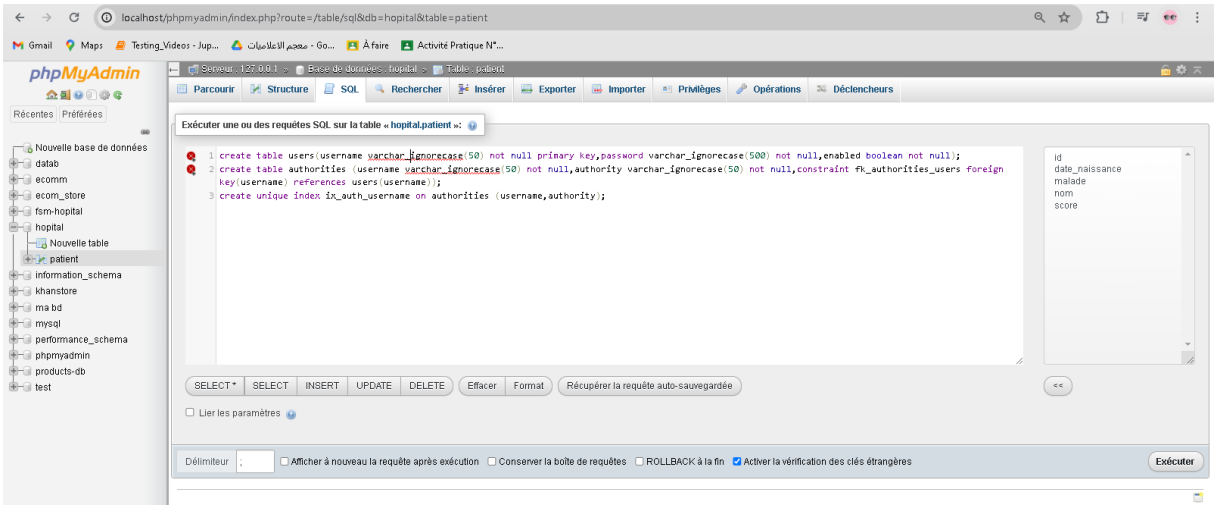


❖ **La stratige JDBC :**

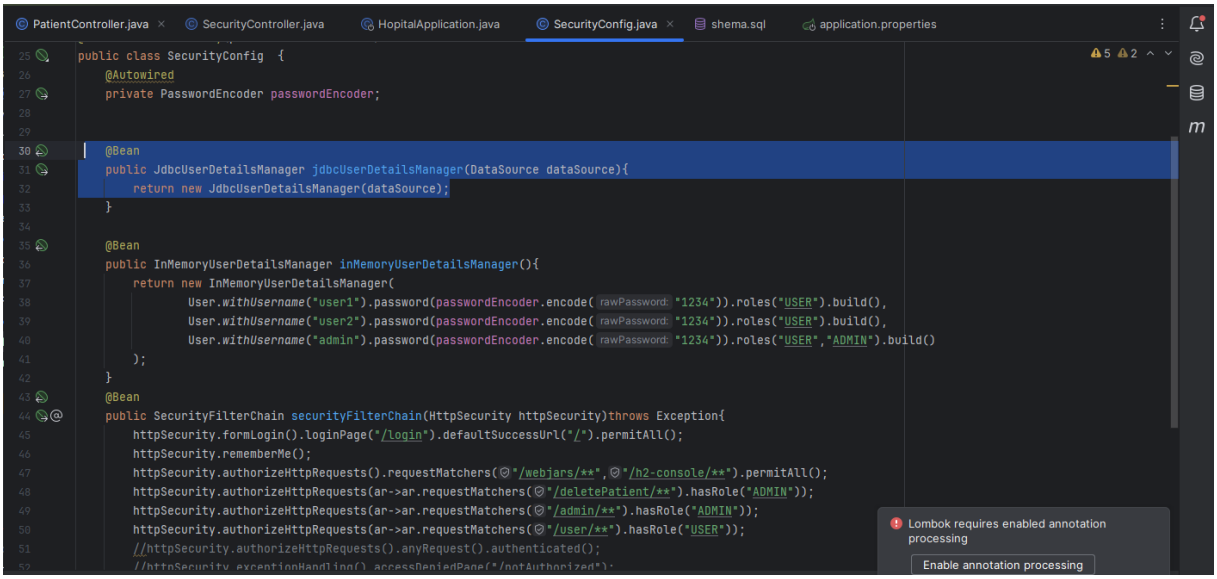
Stoker les utilisateurs dans la base de donnes relationnel.



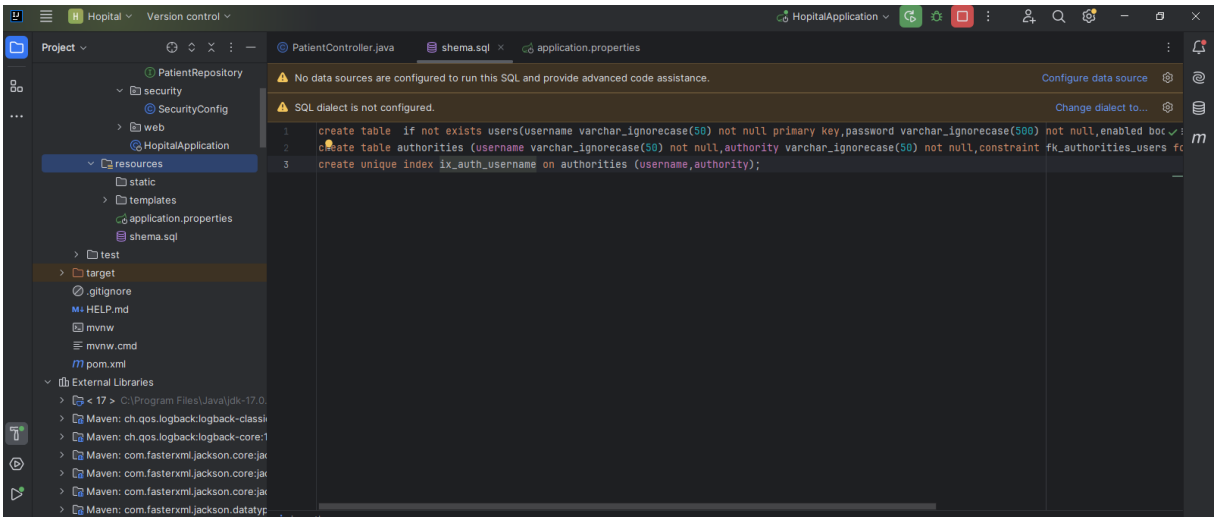
Soit en créer les tables suivants.



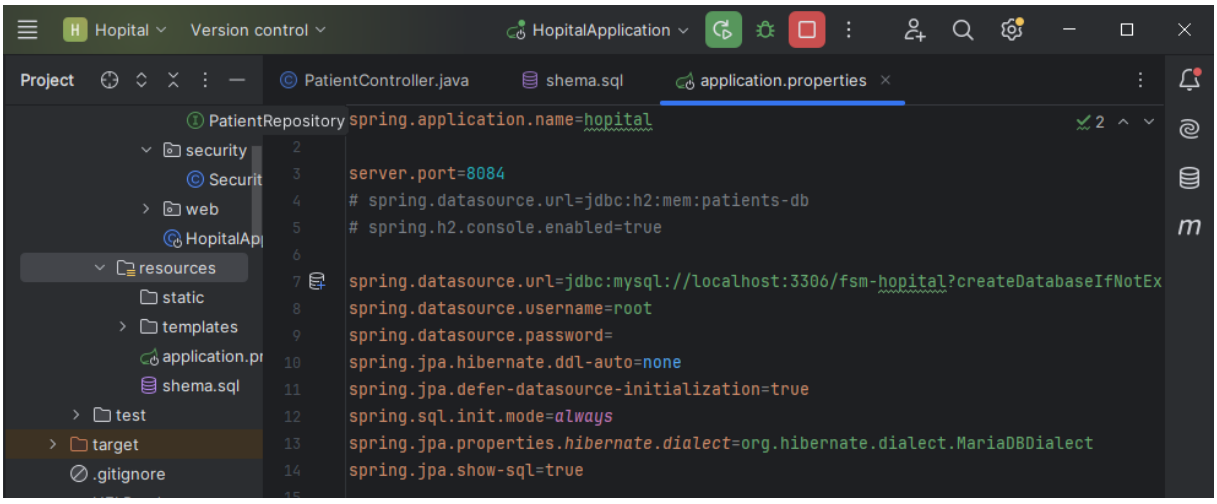
Ou soit par Intelij :



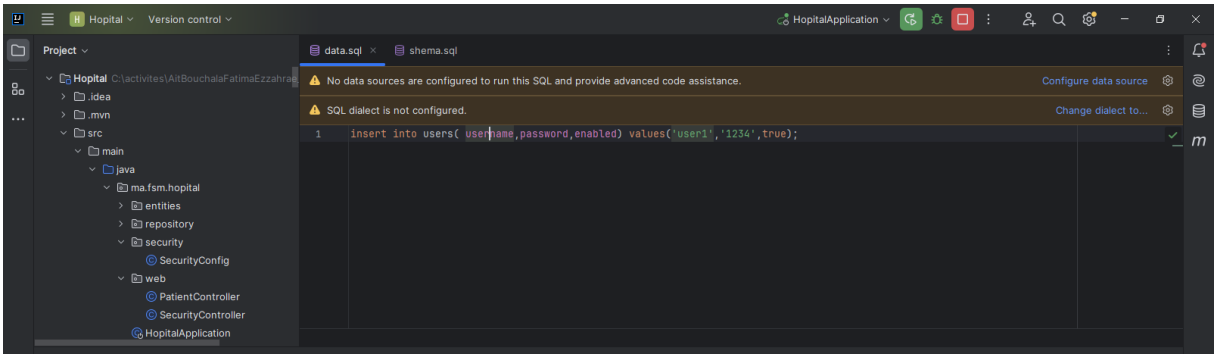
Création de fichier schema.sql :



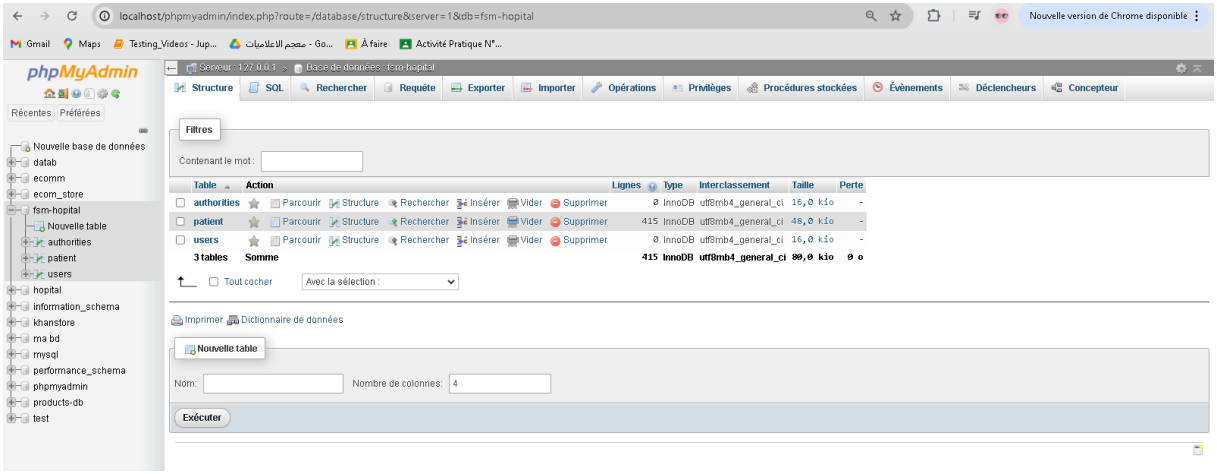
Demander a jpa de ne pas régénérer la base de donnes .



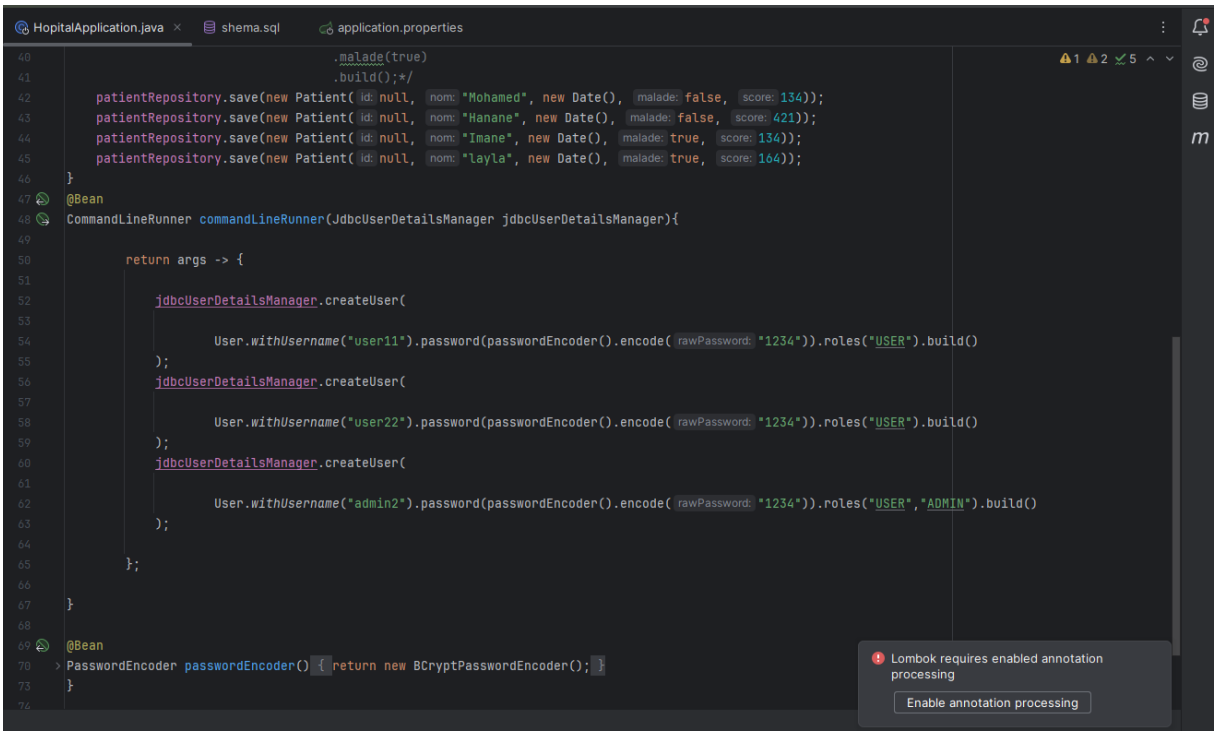
Création de fichier data.sql : Pour alimenter la base des données avec des donnes.

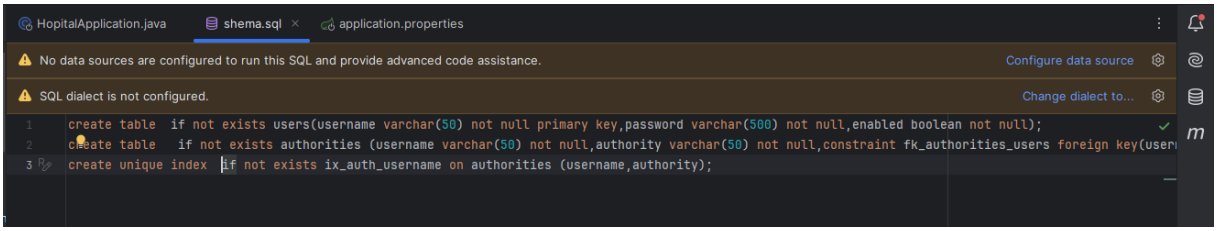


Donc les tableaux sont crée avec succès.



Création des utilisateurs suivants :





La création des tables Sous phpMyAdmin.

Table users :

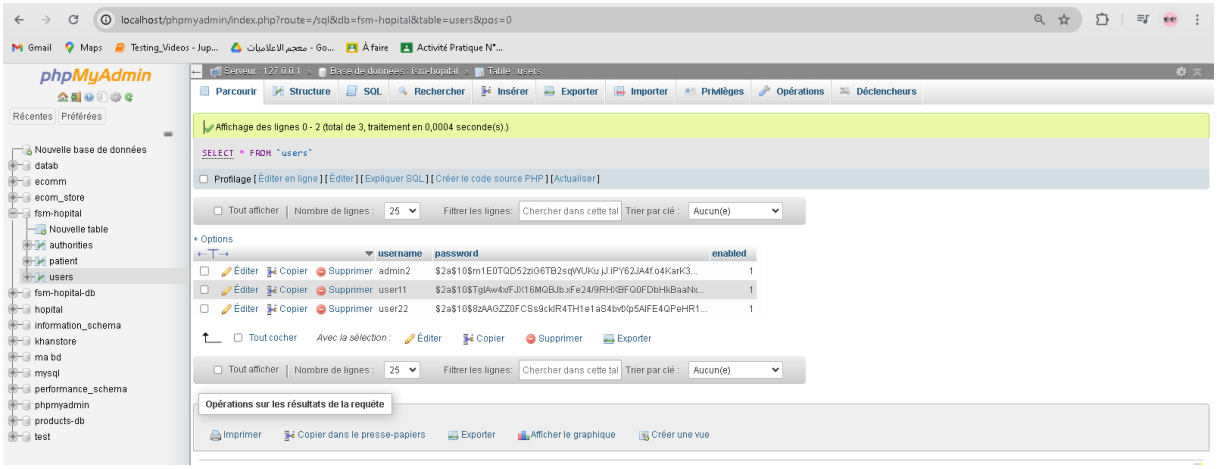
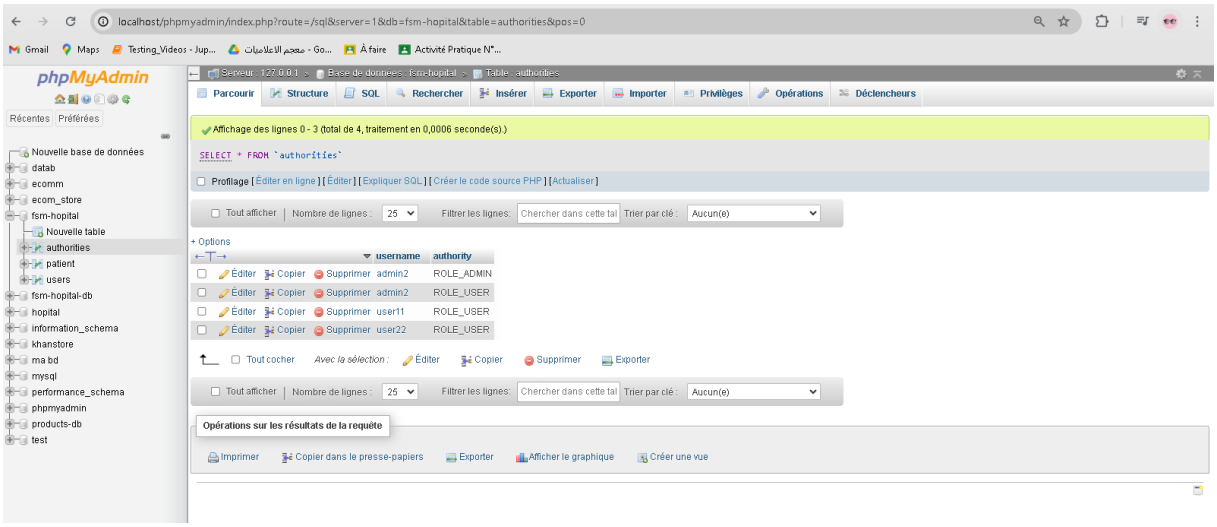
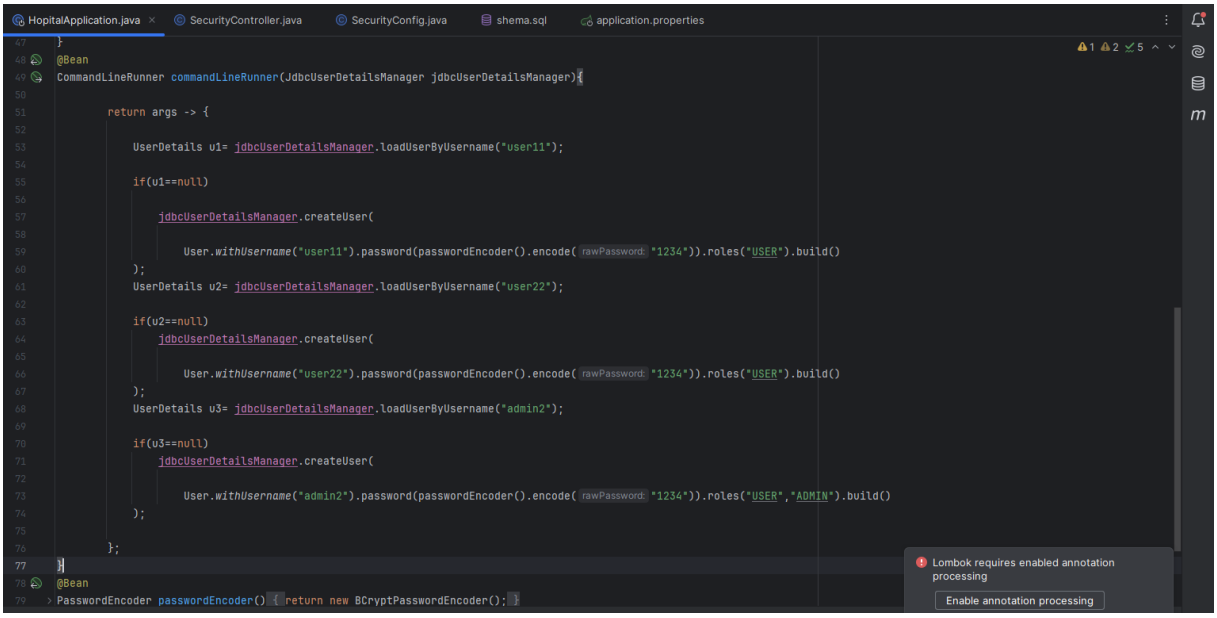


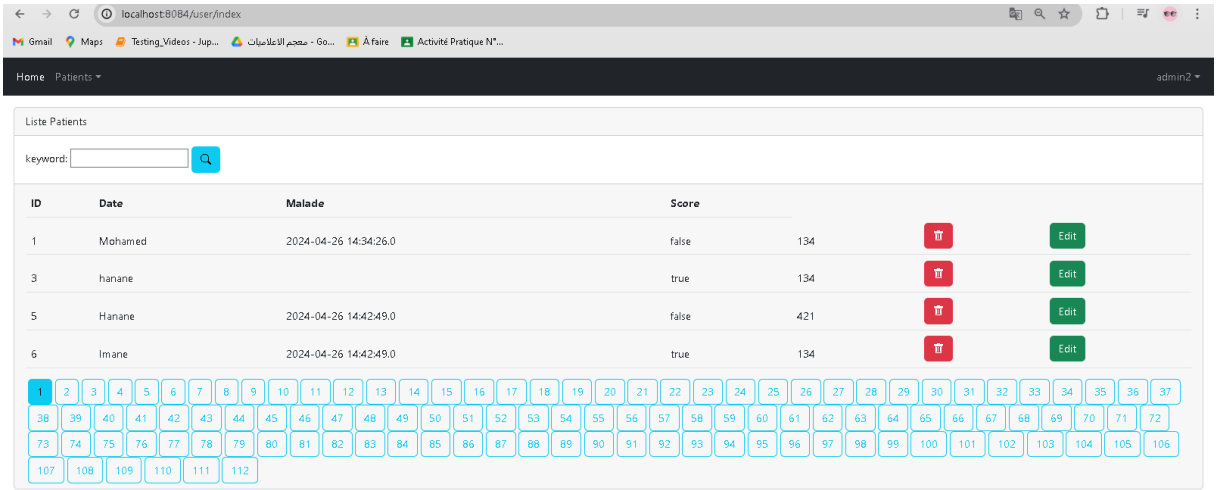
Table authorities :



Pour gérer les utilisateurs et leurs rôles.



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

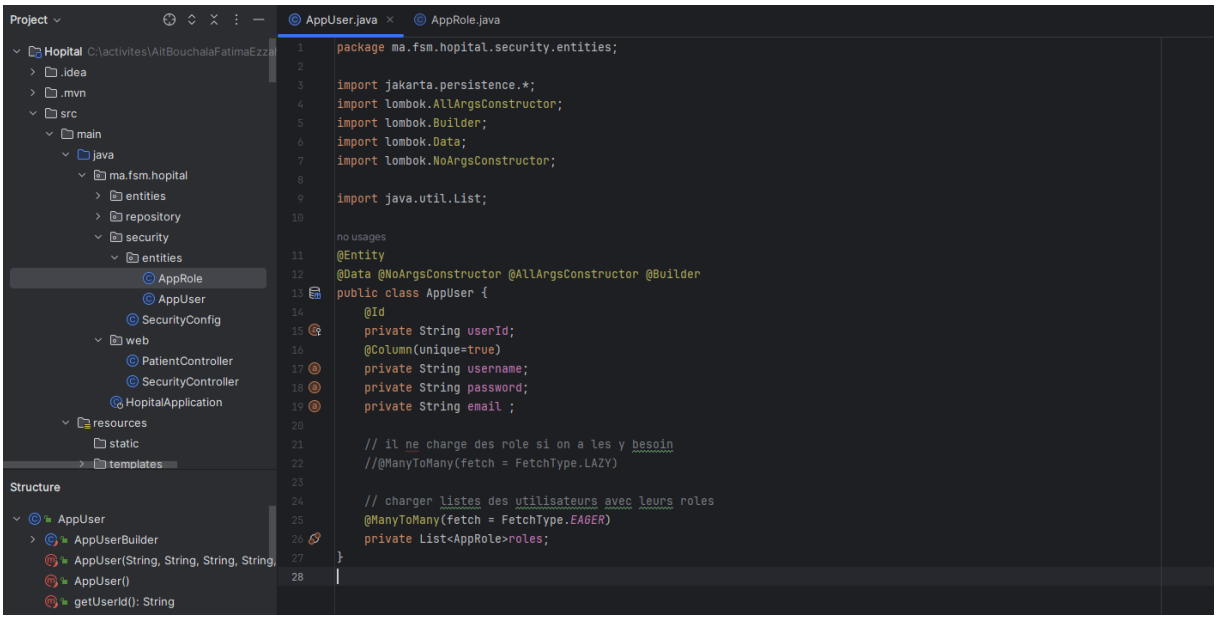


Avec username user1 :

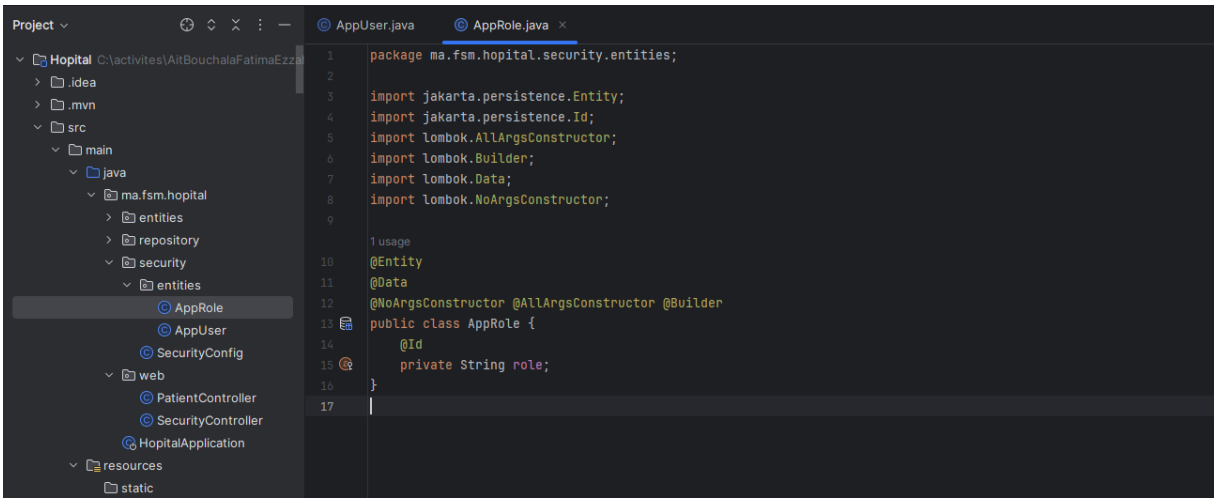
❖ **La stratige UserDetails Service :**

Création de package entities après on créer les classes suivants :

La Classe AppUser :



La Classe AppRole :



Création de package repo avec les deux interfaces suivants :

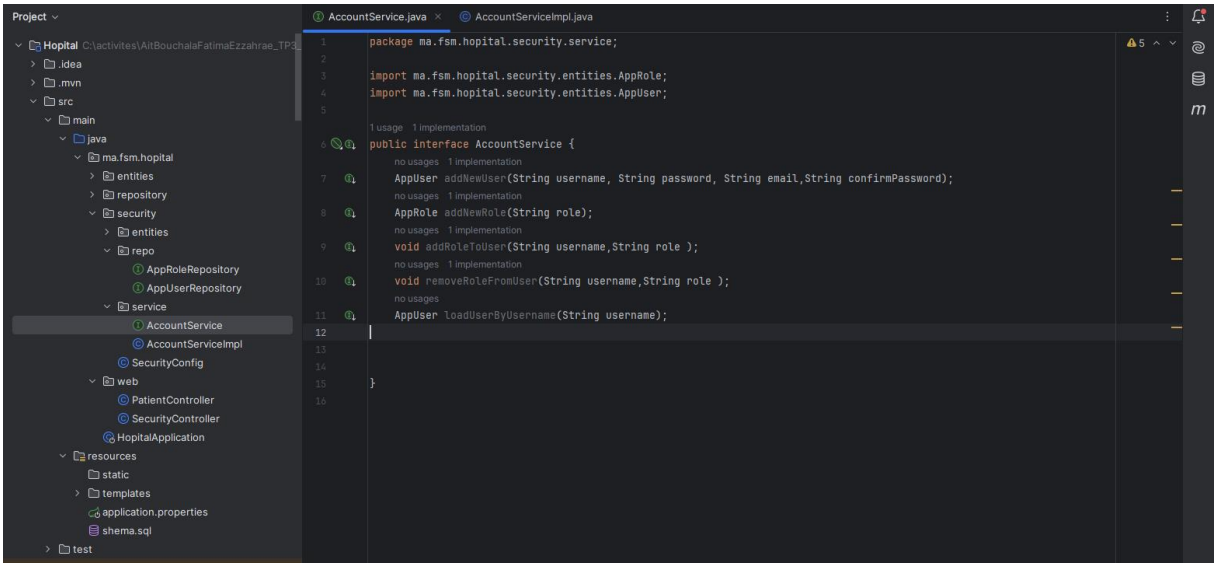
Interface AppUserRepository qui permet de gérer les utilisateurs.

```java
package ma.fsm.hopital.security.repo;

import ma.fsm.hopital.security.entities.AppUser;
import org.springframework.data.jpa.repository.JpaRepository;

no usages
public interface AppUserRepository extends JpaRepository<AppUser,String> {
    no usages
    AppUser findByUsername(String username);


}
```

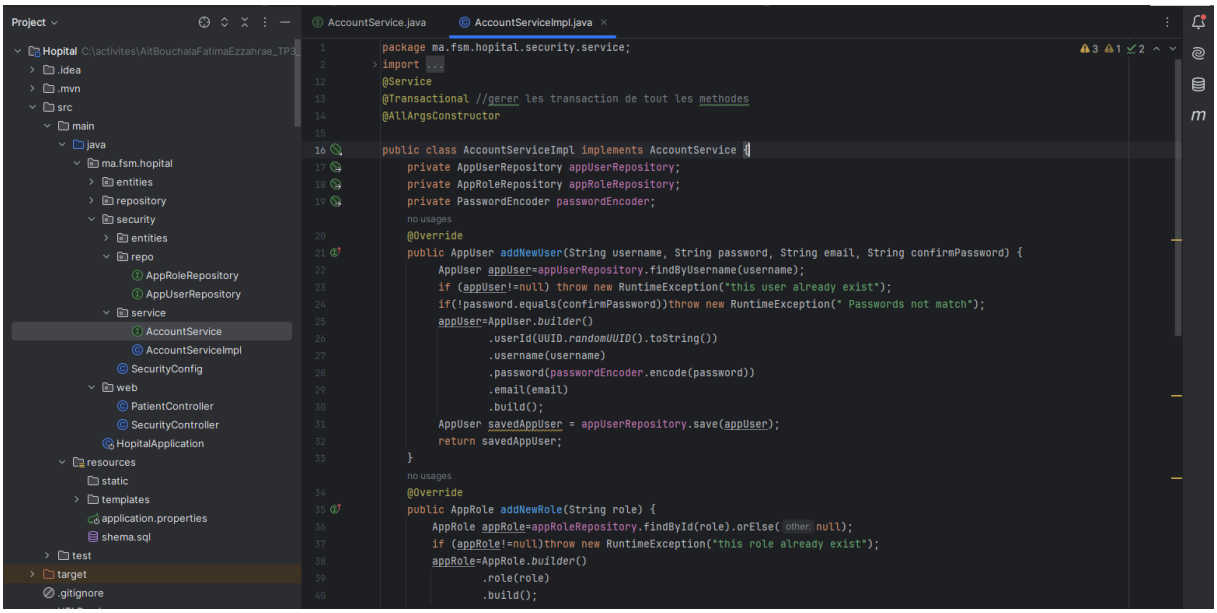
Interface AppRoleRepository qui gérer de créer les roles.

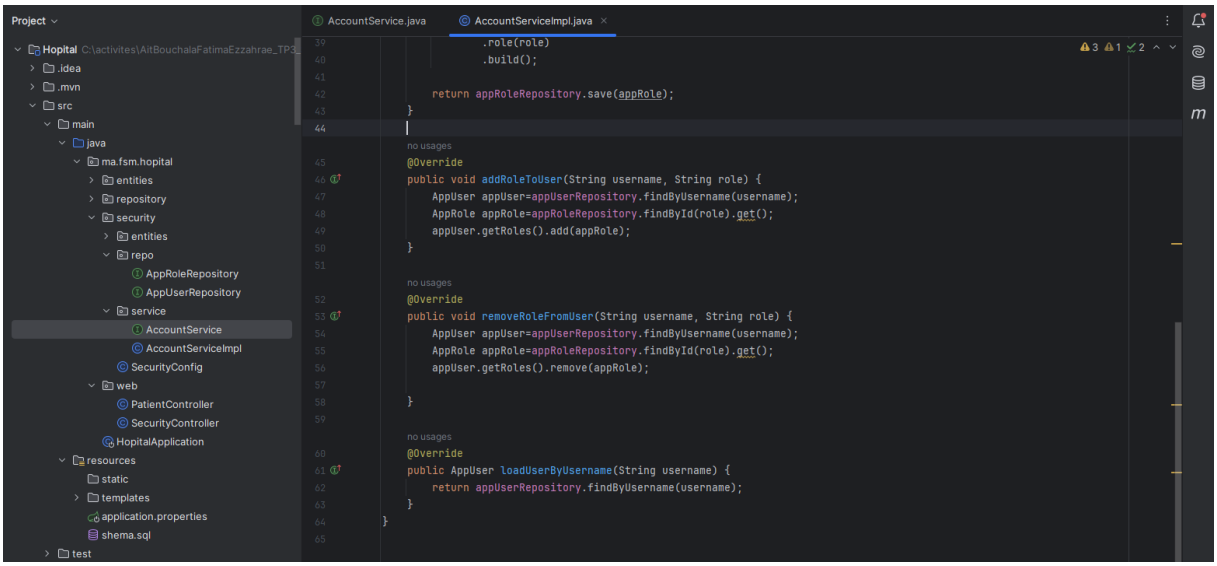
```java
package ma.fsm.hopital.security.repo;

import ma.fsm.hopital.security.entities.AppRole;
import org.springframework.data.jpa.repository.JpaRepository;

no usages
public interface AppRoleRepository extends JpaRepository<AppRole,String> {

}
```

La Création du couche Service avec une interface AccountService.

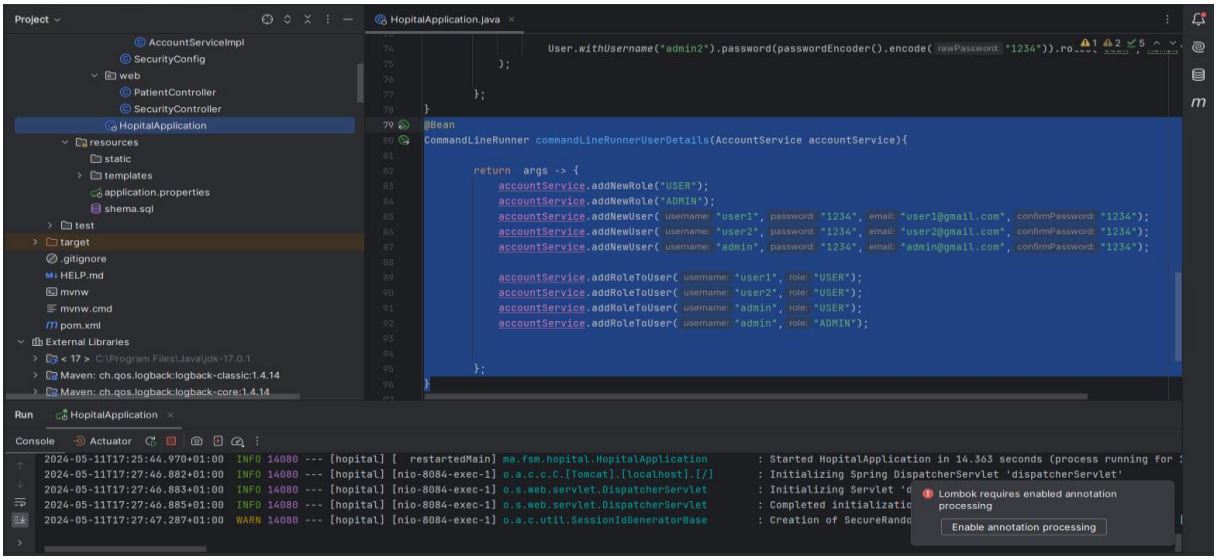


Et L'interface AccountService implemete la classe AccountServiceImpl. Qui définie les méthodes suivantes.

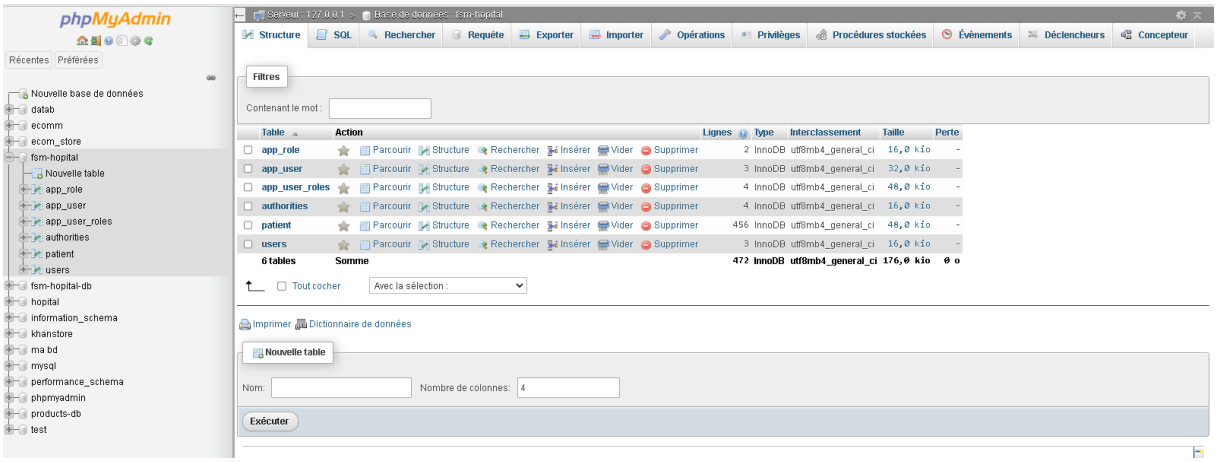




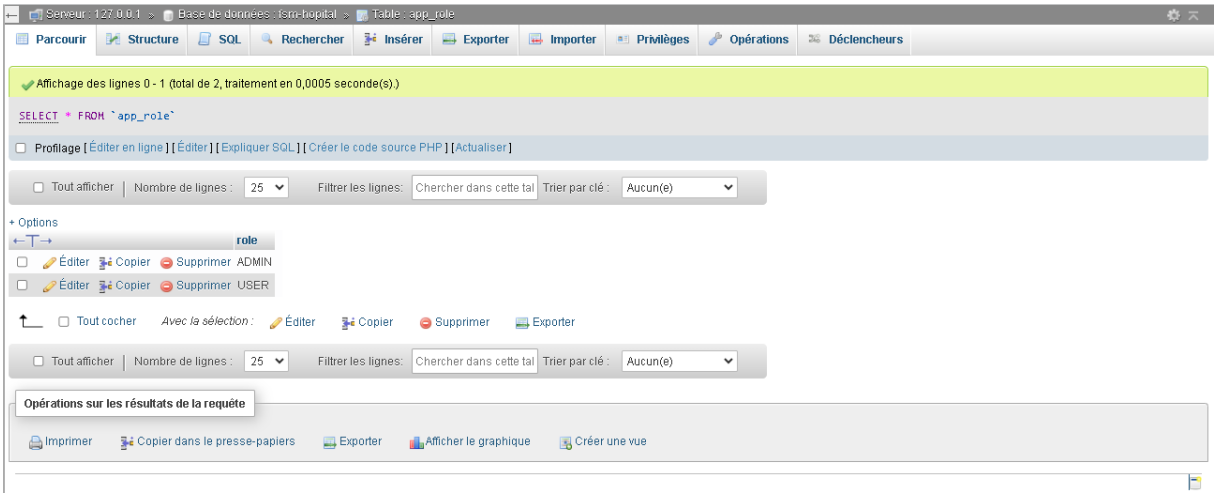
On exécute notre application en ajoutons ces lignes :



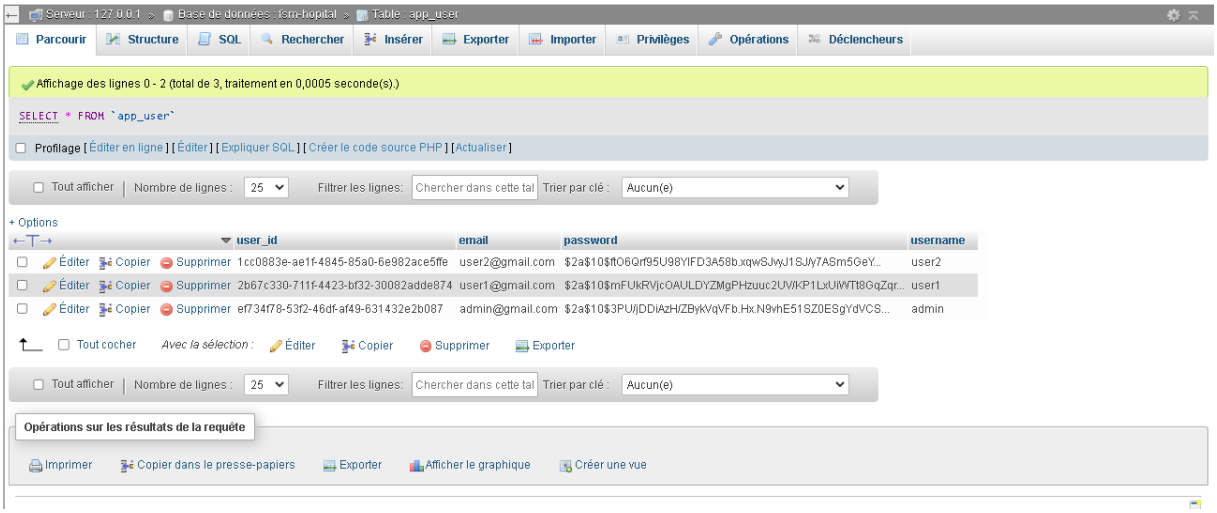
Sous phpMyAdmin on aura les tables suivants :



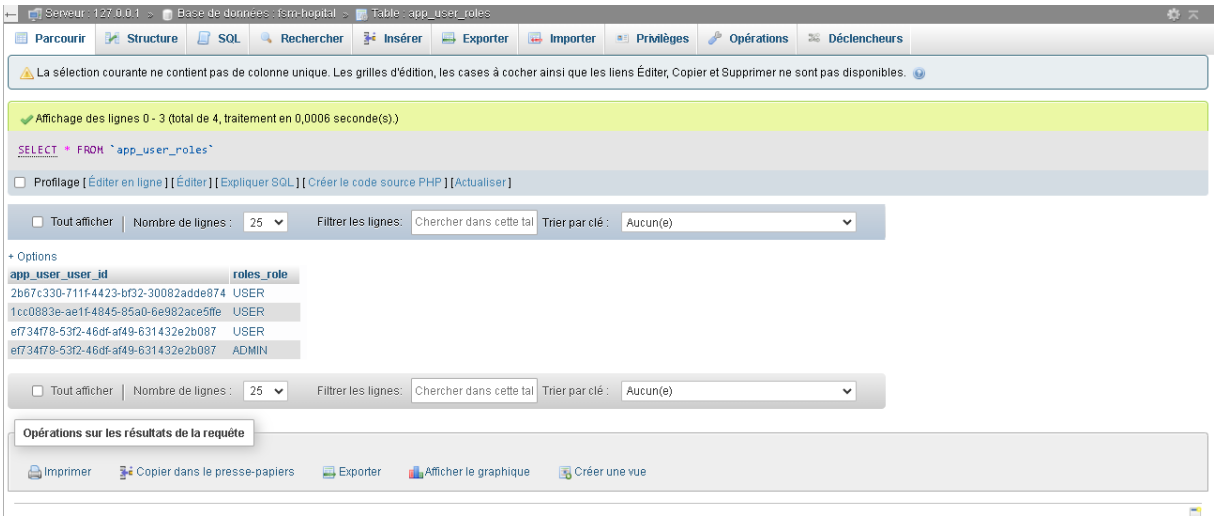
La Table app_role : contient les roles .



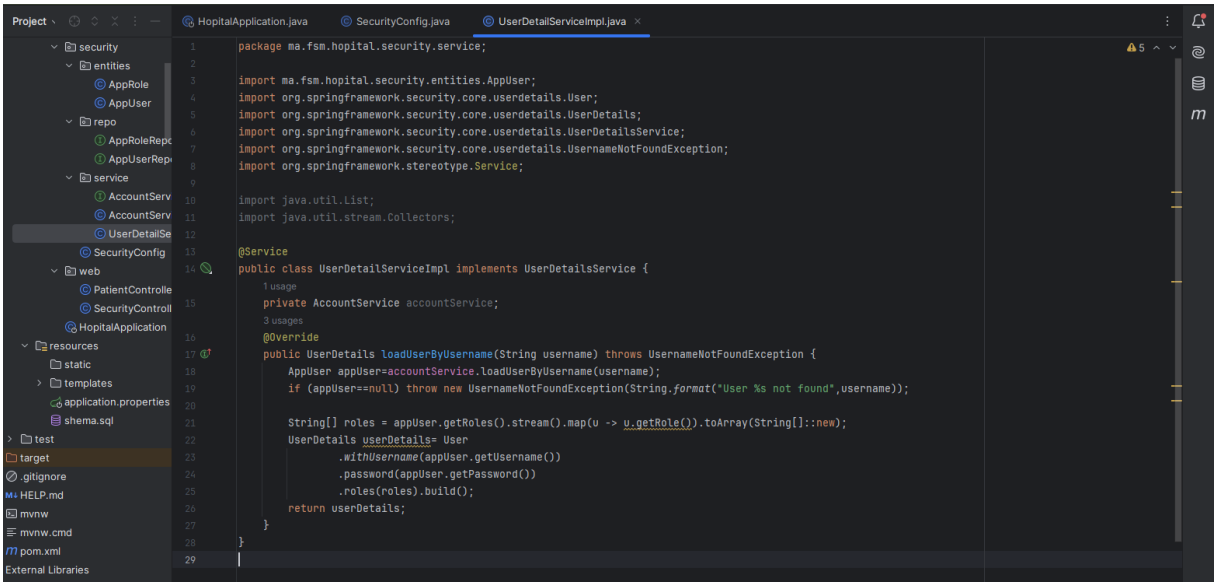
La Table app_user : contient les utilisateurs qui ont le droit d'acces a l'application.



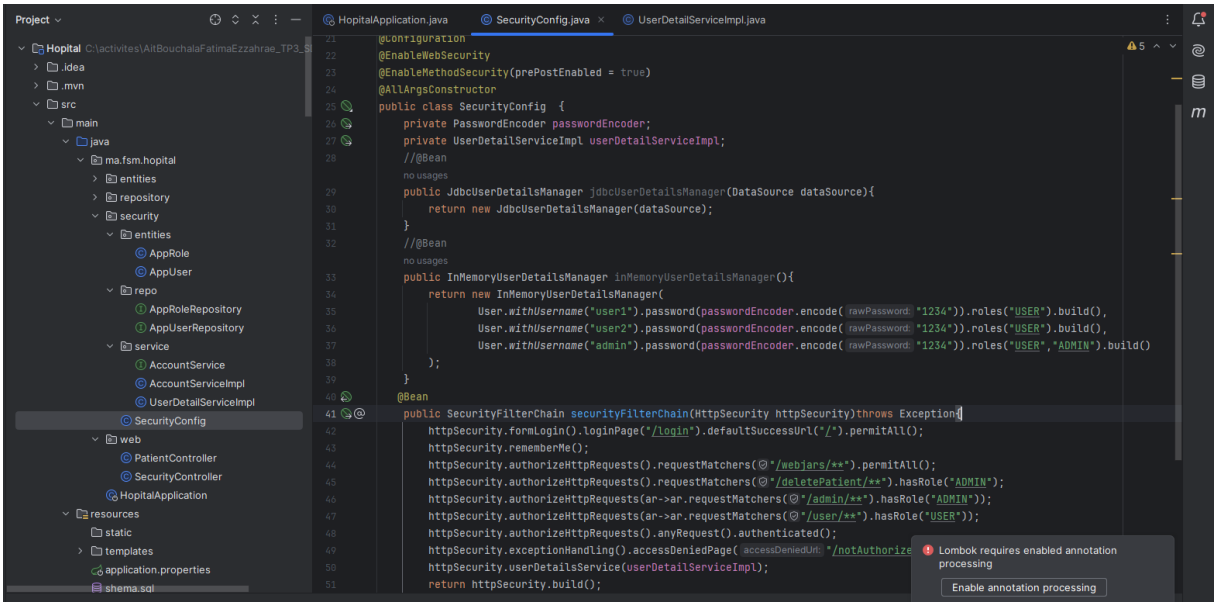
La Table app_user_roles : chaque utilisateur a son role.



La partie la plus importante dans cette strategie .



On va met en commentaire l'objet JbdcUserDetailsManager.



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