Activité Pratique ///

27/04

emsi

Créé par : Yousra Chafik Idrissi

Spring MVC
Thymeleaf
Spring Data

ÉNONCÉ

Partie 1 :

1. **Afficher les Patients**
2. **Faire la Pagination**
3. **Chercher les Patients**
4. **Supprimer un Patient**

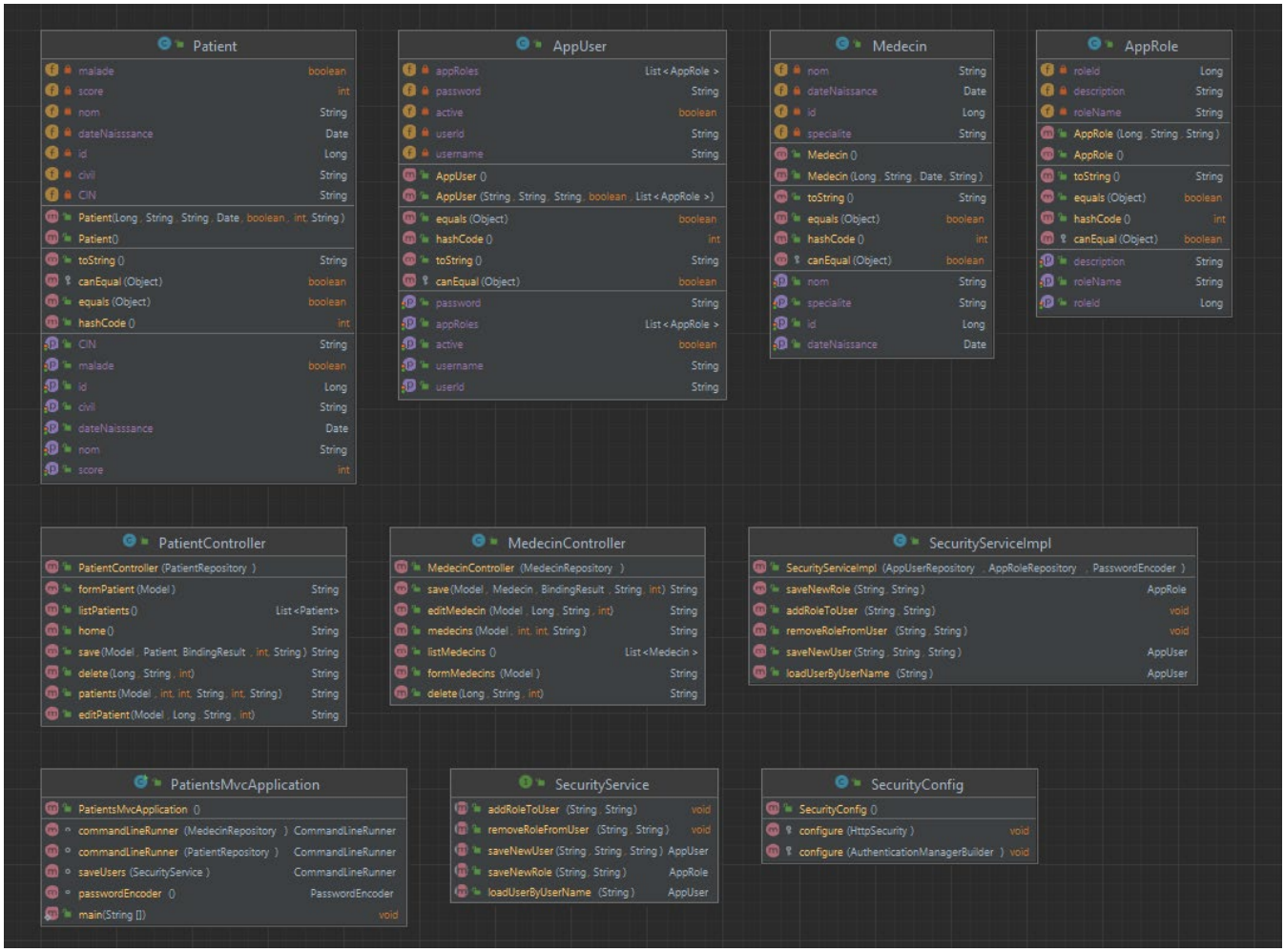
Partie 2 :

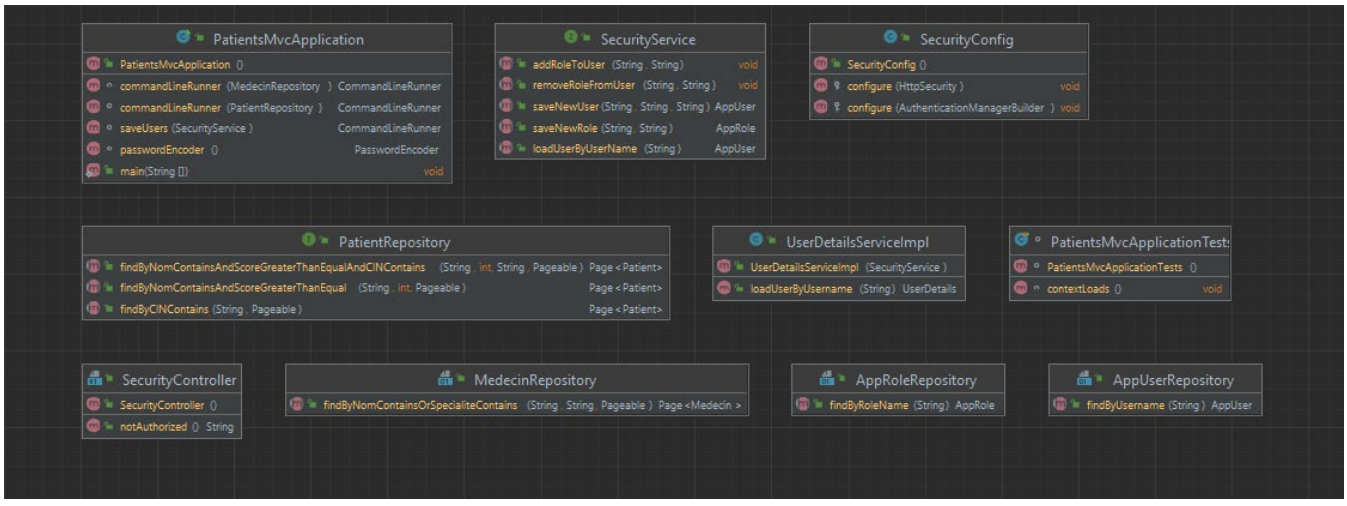
1. **Créer une page Template basée sur Thymeleaf Layout**
2. **Saisir et Ajouter des Patients**
3. **Faire la Validation du Formulaire**
4. **Editer et Mettre à jour un Patient**

Partie 3 & 4:

1. **Ajouter la dépendance Maven de Spring Security**
2. **Personnaliser la configuration de Spring Security**
3. **Basculer de la stratégie authentification**
4. **Basculer vers la stratégie UserDetailsService**

Architecture :


Patient
malade boolean
score int
nom String
dateNaisssance Date
id Long
civil String
CIN String
Patient(Long , String , String , Date , boolean , int, String)
Patient()
toString () String
canEqual (Object) boolean
equals (Object) boolean
hashCode () int
CIN String
malade boolean
id Long
civil String
dateNaisssance Date
nom String
score int
AppUser
appRoles List < AppRole >
password String
active boolean
userId String
username String
AppUser ()
AppUser (String , String , String , boolean , List < AppRole >)
equals (Object) boolean
hashCode () int
toString () String
canEqual (Object) boolean
password String
appRoles List < AppRole >
active boolean
username String
userId String
Medecin
nom String
dateNaissance Date
id Long
specialite String
Medecin ()
Medecin (Long , String , Date , String)
toString () String
equals (Object) boolean
hashCode () int
canEqual (Object) boolean
nom String
specialite String
id Long
dateNaissance Date
AppRole
roleId Long
description String
roleName String
AppRole (Long , String , String)
AppRole ()
toString () String
equals (Object) boolean
hashCode () int
canEqual (Object) boolean
description String
roleName String
roleId Long
PatientController
PatientController (PatientRepository)
formPatient (Model) String
listPatients () List<Patient>
home () String
save (Model , Patient, BindingResult , int, String) String
delete (Long , String , int) String
patients (Model , int, int, String , int, String) String
editPatient (Model , Long , String , int) String
MedecinController
MedecinController (MedecinRepository)
save (Model , Medecin , BindingResult , String , int) String
editMedecin (Model , Long , String , int) String
medecins (Model , int, int, String) String
listMedecins () List<Medecin >
formMedecins (Model) String
delete (Long , String , int) String
SecurityServiceImpl
SecurityServiceImpl (AppUserRepository , AppRoleRepository , PasswordEncoder)
saveNewRole (String , String) AppRole
addRoleToUser (String , String) void
removeRoleFromUser (String , String) void
saveNewUser (String , String , String) AppUser
loadUserByUserName (String) AppUser
PatientsMvcApplication
PatientsMvcApplication ()
commandLineRunner (MedecinRepository) CommandLineRunner
commandLineRunner (PatientRepository) CommandLineRunner
saveUsers (SecurityService) CommandLineRunner
passwordEncoder () PasswordEncoder
main(String [])
SecurityService
addRoleToUser (String , String) void
removeRoleFromUser (String , String) void
saveNewUser (String , String , String) AppUser
saveNewRole (String , String) AppRole
loadUserByUserName (String) AppUser
SecurityConfig
SecurityConfig ()
configure (HttpSecurity) void
configure (AuthenticationManagerBuilder) void


PatientsMvcApplication
PatientsMvcApplication ()
commandLineRunner (MedecinRepository) CommandLineRunner
commandLineRunner (PatientRepository) CommandLineRunner
saveUsers (SecurityService) CommandLineRunner
passwordEncoder () PasswordEncoder
main(String [])
SecurityService
addRoleToUser (String , String) void
removeRoleFromUser (String , String) void
saveNewUser (String , String , String) AppUser
saveNewRole (String , String) AppRole
loadUserByUserName (String) AppUser
SecurityConfig
SecurityConfig ()
configure (HttpSecurity) void
configure (AuthenticationManagerBuilder) void
PatientRepository
findByNomContainsAndScoreGreaterThanEqualAndCINContains (String , int, String , Pageable) Page < Patient>
findByNomContainsAndScoreGreaterThanEqual (String , int, Pageable) Page < Patient>
findByCINContains (String , Pageable) Page < Patient>
UserDetailsServiceImpl
UserDetailsServiceImpl (SecurityService)
loadUserByUsername (String) UserDetails
PatientsMvcApplicationTests
PatientsMvcApplicationTests ()
contextLoads () void
SecurityController
SecurityController ()
notAuthorized () String
MedecinRepository
findByNomContainsOrSpecialiteContains (String , String , Pageable) Page <Medecin >
AppRoleRepository
findByRoleName (String) AppRole
AppUserRepository
findByUsername (String) AppUser

Lien Repo GitHub :

yousracel/JEE-AP3-MVC (github.com)

Captures d'écran :

PARTIE1 :

-Afficher les Patients

ID	NOM	Civilité	DATE NAISSANCE	MALADE	SCORE	N° Carte d'identité		
8	othyy	Mr	2022-04-20	false	70	HU48775	Delete	Edit
9	youssra	Mlle	2022-04-20	true	144	GR12054	Delete	Edit
10	sabah	Mme	2022-04-20	true	12	OK27517	Delete	Edit
11	aliaa	Mme	2022-04-20	true	50	TQ58154	Delete	Edit
12	othmane	Mr	2022-04-20	false	70	DS53710	Delete	Edit

« Previous 0 » Next

-Faire la Pagination

```java
@GetMapping(path = "/user/index")
public String patients(Model model,
                       @RequestParam(name="page",defaultValue = "0") int page,
                       @RequestParam(name="size",defaultValue = "5")int size,
                       @RequestParam(name="keyword",defaultValue = "")String keyword,
                       @RequestParam(name="keyword2",defaultValue = "0") int keyword2,
                       @RequestParam(name="cin",defaultValue = "") String cin){
    Page<Patient> pagePatients=patientRepository.findByNomContainsAndScoreGreaterThanEqual(keyword,keyword2,PageRequest.of(page,size));
    if(!cin.equals(""))
        pagePatients=patientRepository.findByNomContainsAndScoreGreaterThanEqualAndCINContains(keyword,keyword2,cin,PageRequest.of(page,size));
    model.addAttribute( attributeName: "listPatients", pagePatients.getContent());

    model.addAttribute( attributeName: "pages",new int[pagePatients.getTotalPages()]);
    model.addAttribute( attributeName: "currentPage",page);
    model.addAttribute( attributeName: "keyword",keyword);
    model.addAttribute( attributeName: "keyword2",keyword2);
    model.addAttribute( attributeName: "cin",cin);
    model.addAttribute( attributeName: "totalPages",pagePatients.getTotalPages());
    return "patients";
}
```

-Chercher les Patients

Keyword : yousra

Score : 0

CIN :

Chercher

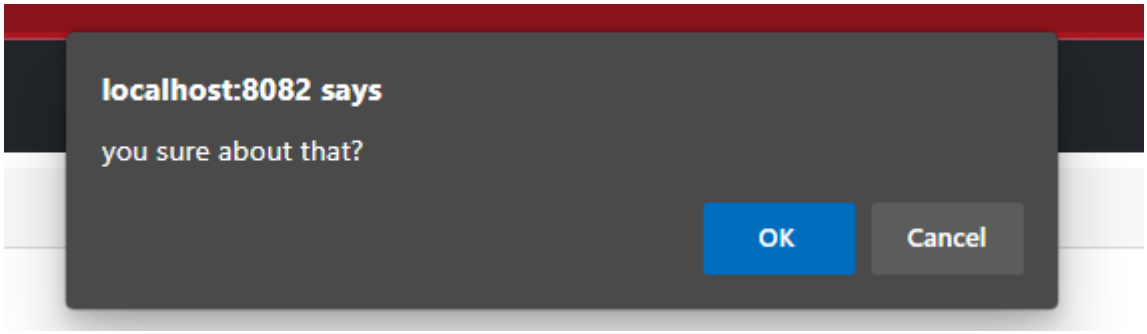
ID	NOM	Civilité	DATE NAISSANCE	MALADE	SCORE	N° Carte d'identité		
13	yousra		2022-04-20	false	10		Delete	Edit
17	yousra		2022-04-20	false	10		Delete	Edit
21	yousra		2022-04-20	false	10		Delete	Edit
25	yousra		2022-04-20	false	10		Delete	Edit
29	yousra		2022-04-20	false	10		Delete	Edit

« Previous 0 » Next

--code :

```java
Page<Patient> findByNomContainsAndScoreGreaterThanEqual(String keyword, int keyword2, Pageable pageable);
Page<Patient> findByNomContainsAndScoreGreaterThanEqualAndCINContains(String keyword, int keyword2, String keyword3, Pageable pageable);
```

-Supprimer un Patient



--code :

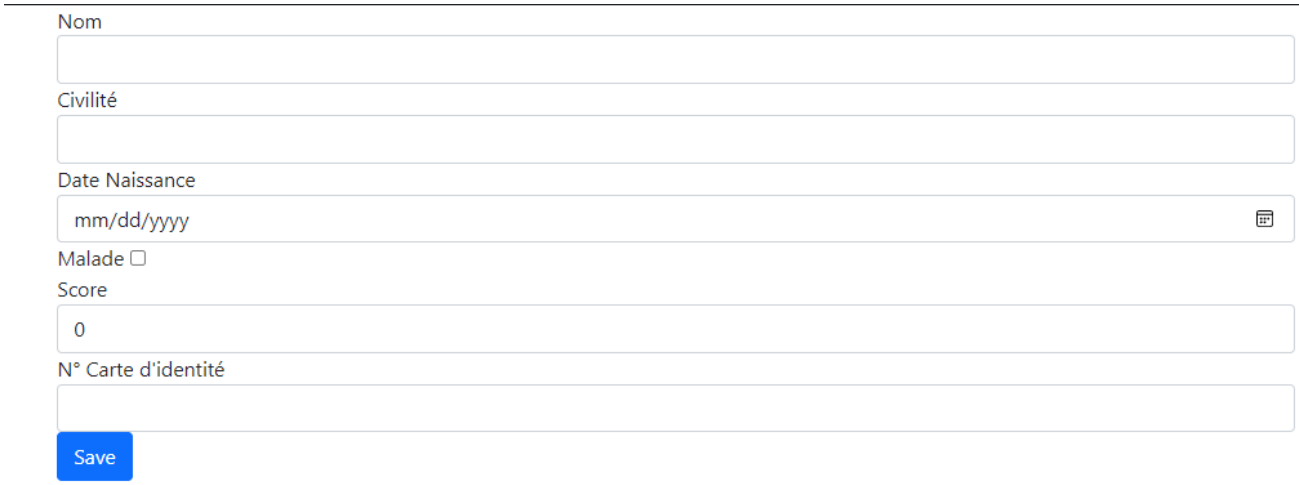
```
@GetMapping("/admin/delete")
public String delete(Long id,String keyword,int page) {
    patientRepository.deleteById(id);
    return "redirect:/user/index?page="+page+"&keyword="+keyword;
}
```

PARTIE2 :

- Créer une page Template basée sur Thymeleaf Layout

```html
<!DOCTYPE html>
<html lang="en"  xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymleaf/layout"
      xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" text="text/css" th:href="@{/webjars/bootstrap/5.1.3/css/bootstrap.min.css}"/>
    <script src="/webjars/bootstrap/5.1.3/js/bootstrap.bundle.js"></script>
</head>
```

- Saisir et Ajouter des Patients



- Faire la Validation du Formulaire

```java
@PostMapping(path = "/admin/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:/user/index?page="+page+ "&keyword="+keyword;
}
```

- Editer et Mettre à jour un Patient

```
@GetMapping("/admin/editPatient")
public String editPatient(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 "editPatient";
}
```

Partie 3 & 4:

- Ajouter la dépendance Maven de Spring Security

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

<dependency>
    <groupId>org.thymeleaf.extras</groupId>
    <artifactId>thymeleaf-extras-springsecurity5</artifactId>
</dependency>
```

- Personnaliser la configuration de Spring Security

```
String encodedPWD=passwordEncoder.encode( rawPassword: "1111");
System.out.println(encodedPWD);
auth.inMemoryAuthentication() InMemoryUserDetailsManagerConfigurer<AuthenticationManagerBuilder>
        .withUser( username: "user1").password(encodedPWD).roles("USER") UserDetailsManagerConfigurer<...>.UserDetailsBuilder
        .and() InMemoryUserDetailsManagerConfigurer<AuthenticationManagerBuilder>
        .withUser( username: "admin").password(passwordEncoder.encode( rawPassword: "1111")).roles("USER","ADMIN");
```

- Basculer de la stratégie authentification

```
auth.jdbcAuthentication()
        .dataSource(dataSource)
        .usersByUsernameQuery("select username as principal,password as credentials, active from users where username=?")
        .authoritiesByUsernameQuery("select username as principal, role as role from users_roles where username=?")
        .rolePrefix("ROLE_")
        .passwordEncoder(passwordEncoder);
```

- Basculer vers la stratégie UserDetailsService

```
auth.userDetailsService(userDetailsService);
```

```java
@Service
public class UserDetailsServiceImpl implements UserDetailsService {
    private SecurityService securityService;

    public UserDetailsServiceImpl(SecurityService securityService) { this.securityService = securityService; }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        AppUser appUser=securityService.loadUserByUserName(username);
        /*Collection<GrantedAuthority> authorities= new ArrayList();
        appUser.getAppRoles().forEach(role->{ //pour chaque role on cree un objet de type Simple..
            SimpleGrantedAuthority authority= new SimpleGrantedAuthority(role.getRoleName()); //smtg about les roles
            authorities.add(authority);// ajout a la collection >>
        });*/

        Collection<GrantedAuthority> authorities1=
                appUser.getAppRoles() List<AppRole>
                        .stream() Stream<AppRole>
                        .map(role->new SimpleGrantedAuthority(role.getRoleName())).collect(Collectors.toList());
        User user= new User(appUser.getUsername(),appUser.getPassword(),authorities1);
        return user;
    }
```