Compte rendu du TP7:

La classe **patient**

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

La classe **patientRepository** qui contient les méthodes pour de traitement de la table patient

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

La classe **patientController** :

La méthode **patients** qui permet de récupérer la liste des patients par mot clé

```java
public class PatientController {
    private PatientRepository patientRepository;

    @GetMapping(path = "/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 ){
        Page<Patient> pagePatients=patientRepository.findByNomContains(keyword, 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);
        return "patients";
    }
```

La méthode **delete** qui permet de supprimer un patient

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

La méthode **home** pour afficher la page accueil

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

La méthode **listPatients** pour afficher la liste des patients en général

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

La classe **properties** qui contient les informations sur la connexion à la base de données

```
spring.datasource.url = jdbc:mysql://localhost:3306/patients?createDatabaseIfNotExist=true
spring.datasource.username = root
spring.datasource.password =
server.port=8083
spring.jpa.hibernate.ddl-auto = create
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MariaDBDialect
spring.jpa.show-sql=true
```

La classe Main

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

@Bean
CommandLineRunner commandLineRunner(PatientRepository patientRepository){
    return args -> {
        patientRepository.save(
                new Patient( id: null, nom: "soukaina", new Date(), malade: false, score: 12)
        );
        patientRepository.save(
                new Patient( id: null, nom: "salma", new Date(), malade: false, score: 12)
        );
        patientRepository.save(
                new Patient( id: null, nom: "soukaina1", new Date(), malade: false, score: 12)
        );
        patientRepository.save(
                new Patient( id: null, nom: "salma1", new Date(), malade: false, score: 12)
        );
        patientRepository.findAll().forEach(p ->{
            System.out.println(p.getNom());
        });
    };
```

Package Security:

Entity AppRole:

```
@Entity
@Data @AllArgsConstructor @NoArgsConstructor
public class AppRole {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long roleID;
    @Column(unique = true)
    private String roleName;
    private String description;
}
```

Entity AppUser:

```java
@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class AppUser {
    @Id
    private String userId;
    @Column(unique = true)
    private String username;
    private String password;
    private boolean active;

    @ManyToMany(fetch = FetchType.EAGER)
    private List<AppRole> appRoles = new ArrayList<>();

}
```

AppRole Repository:

```java
import ...

public interface AppRoleRepository extends JpaRepository<AppRole,Long> {
    AppRole findByRoleName(String roleName);
}
```

AppUser Repository:

```java
@Repository
public interface AppUserRepository extends JpaRepository<AppUser,String> {
    AppUser findByUsername(String username);
}
```

Security Service Interface:

```java
public interface SecurityService {
    AppUser saveNewUser(String username, String password, String rePassword);
    AppRole saveNewRole(String roleName, String description);
    void addRoleToUser(String username, String roleName);
    void RemoveRoleFromUser(String username, String roleName);
    AppUser loadUserByUserName(String username);
}
```

Security Service Implementation:

```java
@Transactional // faire attention d'utiliser celle de spring
public class SecurityServiceImpl implements SecurityService {

    private AppUserRepository appUserRepository;
    private AppRoleRepository appRoleRepository;
    private PasswordEncoder passwordEncoder;

    public SecurityServiceImpl(AppUserRepository appUserRepository, AppRoleRepository appRoleRepository, Password
        this.appUserRepository = appUserRepository;
        this.appRoleRepository = appRoleRepository;
        this.passwordEncoder = passwordEncoder;
    }

    @Override
    public AppUser saveNewUser(String username, String password, String rePassword) {
        if (!password.equals(rePassword)) throw new RuntimeException("mot de pass incorrect");
        String hashedPWD = passwordEncoder.encode(password);
        AppUser appUser = new AppUser();
        appUser.setUserId(UUID.randomUUID().toString());
        appUser.setUsername(username);
        appUser.setPassword(hashedPWD);
        appUser.setActive(true);
        AppUser savedAppUser = appUserRepository.save(appUser);
        return savedAppUser;
    }
```

```java
@Override
public AppRole saveNewRole(String roleName, String description) {
    AppRole appRole = appRoleRepository.findByRoleName(roleName);
    if (appRole!=null) throw new RuntimeException("Role "+roleName+" Already exist");
    appRole=new AppRole();
    appRole.setRoleName(roleName);
    appRole.setDescription(description);
    AppRole savedAppRole = appRoleRepository.save(appRole);
    return savedAppRole;
}

@Override
public void addRoleToUser(String username, String roleName) {
    AppUser appUser = appUserRepository.findByUsername(username);
    if (appUser==null) throw new RuntimeException("User Not Found");
    AppRole appRole = appRoleRepository.findByRoleName(roleName);
    if (appRole==null) throw new RuntimeException("Role Not Found");
    appUser.getAppRoles().add(appRole);
    //appUserRepository.save(appUser);
}
```

```java
@Override
public void RemoveRoleFromUser(String username, String roleName) {
    AppUser appUser = appUserRepository.findByUsername(username);
    if (appUser==null) throw new RuntimeException("User Not Found");
    AppRole appRole = appRoleRepository.findByRoleName(roleName);
    if (appRole==null) throw new RuntimeException("Role Not Found");
    appUser.getAppRoles().remove(appRole);

}

@Override
public AppUser loadUserByUserName(String username) { return appUserRepository.findByUsername(username); }
```

User Details Service Interface:

```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);
        /*
        //Technique classique
        Collection<GrantedAuthority> authorities = new ArrayList<>();
        appUser.getAppRoles().forEach(role -> {
            SimpleGrantedAuthority authority = new SimpleGrantedAuthority(role.getRoleName());
            authorities.add(authority);
        });*/

        // Technique en utilisant l'API streams
        Collection<GrantedAuthority> authorities1=
                appUser.getAppRoles() List<AppRole>
                        .stream() Stream<AppRole>
                        .map(role -> new SimpleGrantedAuthority(role.getRoleName())) Stream<SimpleGrantedAuthority>
                        .collect(Collectors.toList());

        User user = new User(appUser.getUsername(),appUser.getPassword(),authorities1);
        return user;
    }
```

Security Configuration:

```java
@EnableWebSecurity
public class securityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        PasswordEncoder passwordEncoder=passwordEncoder();
        String encodedPWD=passwordEncoder.encode( rawPassword: "1234");
        //stocker en mémoire les utilisateurs qui ont acces à l'application
        auth.inMemoryAuthentication().withUser( username: "user").password(encodedPWD).roles("USER");
        auth.inMemoryAuthentication().withUser( username: "admin").password(passwordEncoder.encode( rawPassword: "5678
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        //damander à spring d'utiliser un formulaire d'authentification
        http.formLogin();
        // cette page ne nécessite pas une permission
        http.authorizeRequests().antMatchers( ...antPatterns: "/").permitAll();
        //ces paths sont accessibles juste pour les utilisateurs qui ont role ADMIN
        http.authorizeRequests().antMatchers( ...antPatterns: "/admin/**").hasRole("ADMIN");
        //ces paths sont accessibles juste pour les utilisateurs qui ont role USER
        http.authorizeRequests().antMatchers( ...antPatterns: "/user/**").hasRole("USER");
        //toute requete http nécessite une authentification
        http.authorizeRequests().anyRequest().authenticated();
        http.exceptionHandling().accessDeniedPage("/403");

    }
```

Security Controller:

```java
@Controller
public class securityController {

    @GetMapping("/403")
    public String notAuthorized() { return "403"; }
}
```

Application:

```java
@SpringBootApplication
public class TpóApplication {

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

    @Bean // au demarrage crée un objet de type PasswordEncoder
    PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

    //@Bean
    CommandLineRunner commandLineRunner(PatientRepository patientRepository){
        return args -> {
            patientRepository.save(
                    new Patient( id: null, nom: "soukaina", new Date(), malade: false, score: 120)
            );
            patientRepository.save(
                    new Patient( id: null, nom: "salma", new Date(), malade: false, score: 120)
            );
            patientRepository.findAll().forEach(p ->{
                System.out.println(p.getNom());
            });
        };
    }
```

Templates:
La page html pour afficher le résultat dans une page web

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="webjars/bootstrap/5.1.3/css/bootstrap.min.css">
</head>
<body>
<div class="container mt-2">
    <div class="card">
        <div class="card-header">Liste des Patients</div>
        <div class="card-body">
            <form method="get" th:action="@{index}">
                <label>Key Word</label>
                <input type="text" name="keyword" th:value="${keyword}">
                <button type="submit" class="bts btn-primary">Chercher</button>
            </form>
            <table class="table">
                <thead>
                <tr>
                    <th>ID</th>
                    <th>Nom</th>
                    <th>DateNaissance</th>
                    <th>Malade</th>
                    <th>Score</th>
                </tr>
```

La forme d’ajout d’un patient

```
<div layout:fragment="content1">
    <div class="col-md-6 offset-3">
    <form method="post" th:action="@{/admin/save}">
        <div class="mb-3">
            <label for="exampleNom" class="form-label">Nom</label>
            <input type="text" class="form-control" id="exampleNom" name="nom" th:value="${patient.nom}">
             <span th:errors="${patient.nom}"></span>
        </div>
        <div class="mb-3">
            <label for="exampleDate" class="form-label">Date de naissance</label>
            <input type="date" class="form-control" id="exampleDate" name="datenaissance" th:value="${patien
            <span th:errors="${patient.datenaissance}"></span>
        </div>
        <div class="mb-3 form-check">
            <input type="checkbox" class="form-check-input" id="exampleCheck1" name="malade"th:checked="${pa
            <label class="form-check-label" for="exampleCheck1">Malade</label>
            <span th:errors="${patient.malade}"></span>
        </div>
        <div class="mb-3">
            <label for="exampleScore" class="form-label">Score</label>
            <input type="text" class="form-control" id="exampleScore"  name="score"th:value="${patient.score
            <span th:errors="${patient.score}"></span>
        </div>
        <button type="submit" class="btn btn-primary">Submit</button>
    </form>
```

Exécution:

Les tables:

Table	Action	Lignes	Type	Interclassement	Taille	Perte
app_role	Parcourir Structure Rechercher Insérer Vider Supprimer	2	InnoDB	latin1_swedish_ci	32,0 kio	-
app_user	Parcourir Structure Rechercher Insérer Vider Supprimer	3	InnoDB	latin1_swedish_ci	32,0 kio	-
app_user_app_roles	Parcourir Structure Rechercher Insérer Vider Supprimer	4	InnoDB	latin1_swedish_ci	48,0 kio	-
patient	Parcourir Structure Rechercher Insérer Vider Supprimer	14	InnoDB	latin1_swedish_ci	16,0 kio	-
4 tables	Somme	23	MyISAM	latin1_swedish_ci	128,0 kio	0 o

Table Patient:

	id	datenaissance	malade	nom	score
Éditer Copier Supprimer	1	2022-04-04	0	soukaina	120
Éditer Copier Supprimer	2	2022-04-04	0	salma	120
Éditer Copier Supprimer	3	2022-04-04	0	soukaina	120
Éditer Copier Supprimer	4	2022-04-04	0	salma	120
Éditer Copier Supprimer	5	2022-04-04	0	soukaina	120
Éditer Copier Supprimer	6	2022-04-04	0	salma	120
Éditer Copier Supprimer	7	2022-04-04	0	soukaina	120
Éditer Copier Supprimer	8	2022-04-04	0	salma	120
Éditer Copier Supprimer	9	2022-04-04	0	soukaina	120
Éditer Copier Supprimer	10	2022-04-04	0	salma	120
Éditer Copier Supprimer	11	2022-04-04	0	soukaina	120
Éditer Copier Supprimer	12	2022-04-04	0	salma	120
Éditer Copier Supprimer	13	2022-04-15	0	soukaina	120
Éditer Copier Supprimer	14	2022-04-15	0	salma	120

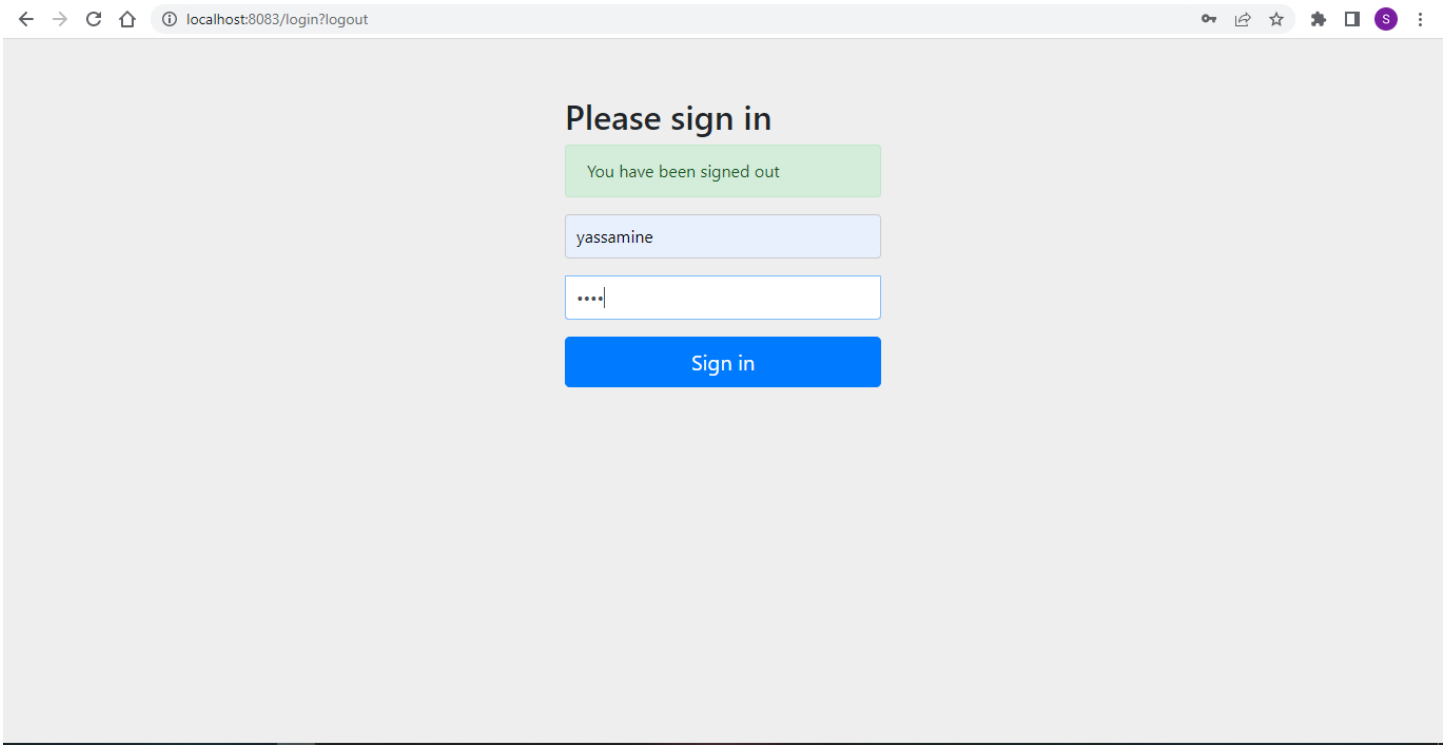
Table AppUser:

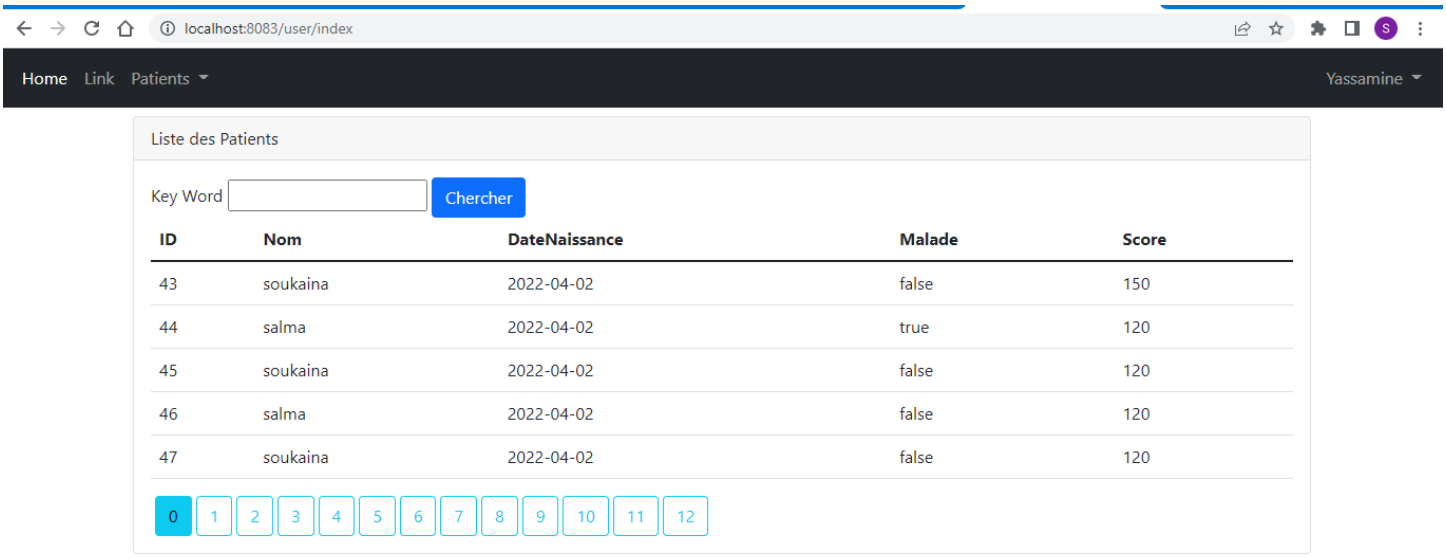
	user_id	active	password	username
Éditer Copier Supprimer	360f8a9c-2558-47f3-87bd-9ead28e1c4d1	1	$2a$10$A1Tvdg4KH4IlrpPI1mrJ2uiz.0dvFXMDAqhCa9pG.eW...	Hassan
Éditer Copier Supprimer	48bf3d1f-960a-448b-9bbb-6ce2bb8cc225	1	$2a$10$e2pkHn99u0Y5ekiSoo.Kc.ezFOCyS4J7QzLtLDfQB3s...	Yassamine
Éditer Copier Supprimer	bf0a2251-8182-4925-8ffe-9b2d9a2b1d1f	1	$2a$10$wC4VpcDlMpMlCvQG3KlvNupQqUw03V99t.apud6wOCc...	Mohamed

Table AppRole:

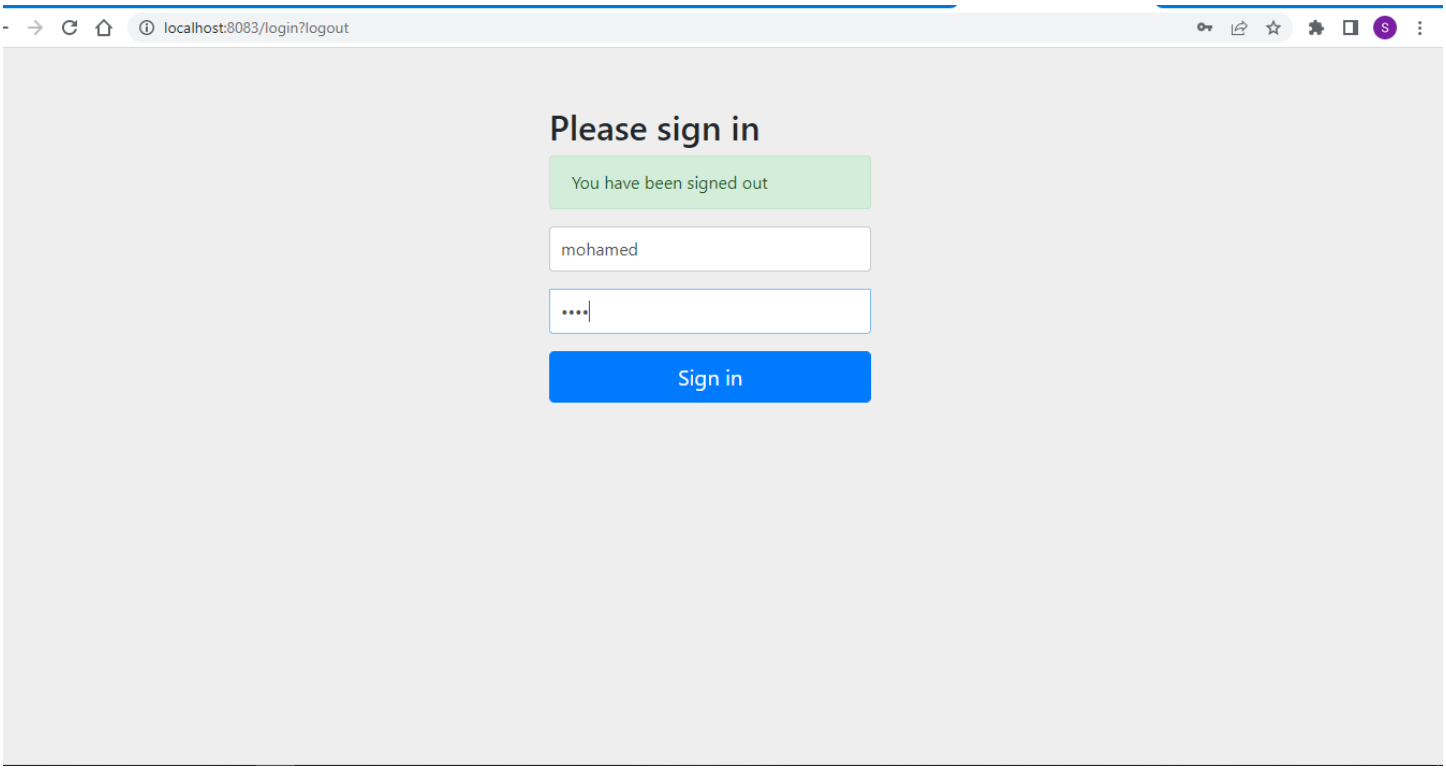
	roleid	description	role_name
Éditer Copier Supprimer	1		USER
Éditer Copier Supprimer	2		ADMIN

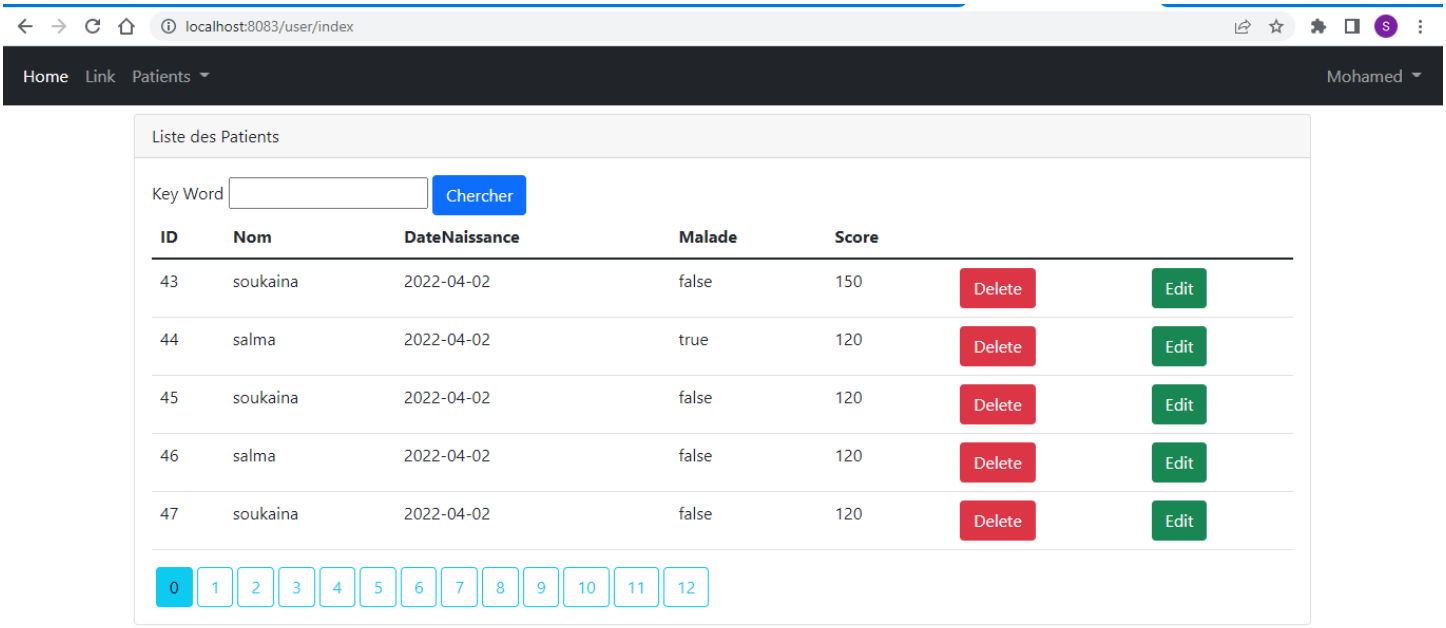
Interface Web:

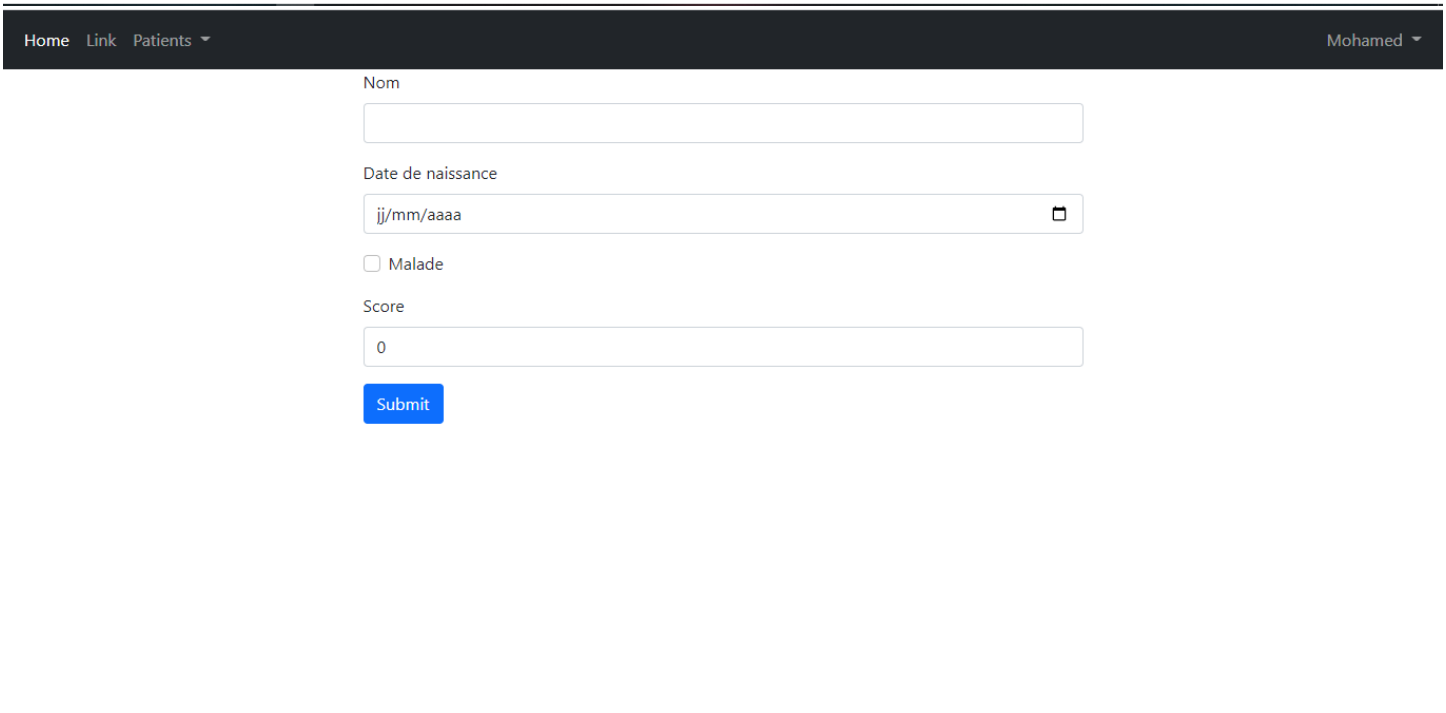




Connexion en tant qu'admin







Mon GitHub: https://github.com/salma-didi/khadidi-salma-JEE