|  |  |
| --- | --- |
|  |  |
| Rapport de TP9 |  |
|  |  |
|  | DATECOURSE TITLE |
|  | STUDENT’S NAMETEACHER’S NAME |

1. **Consignes:**

Créer une application Web basée sur Spring MVC, Spring Data JPA et Spring Security qui permet de gérer des étudiants.  
Chaque étudiant est défini par:  
 - Son id  
 - Son nom  
 - Son prénom  
 - Son email  
 - Sa date naissance  
 - Son genre : MASCULIN ou FEMININ  
 - Un attribut qui indique si il est en règle ou non  
L'application doit offrir les fonctionnalités suivantes :  
  - Chercher des étudiants par nom  
  - Faire la pagination  
  - Supprimer des étudiants en utilisant la méthode (DELETE au lieu de GET)  
  - Saisir et Ajouter des étudiants avec validation des formulaires  
  - Editer et mettre à jour des étudiants  
  - Créer une page template   
  - Sécuriser l'accès à l'application avec un système d'authentification basé sur Spring security en utilisant la stratégie UseDetails Service  
 - Ajouter d'autres fonctionnalités supplémentaires

1. **Exécution :**

On a commencé par créer Entité Etudiant :

package com.example.tp8.Entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import org.springframework.format.annotation.DateTimeFormat;  
  
import javax.persistence.\*;  
import java.util.Date;  
  
@Entity  
@Data @AllArgsConstructor @NoArgsConstructor  
public class Etudiant {  
 @Id @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long Stu\_ID;  
 @Column(unique = true)  
 private String nom;  
 private String prenom;  
 private String email;  
 @Temporal(TemporalType.*DATE*)  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 private Date date\_naissance;  
 private String genre;   
 private boolean regle;  
}

Pour qu’ensuite créer son repository :

package com.example.tp8.Repositories;  
  
import com.example.tp8.Entities.Etudiant;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.Pageable;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface EtudiantRepository extends JpaRepository<Etudiant, Long> {  
 Page<Etudiant> findByNomContains(String kw, Pageable pageable);  
 Page<Etudiant> findAll(Pageable pageable);  
}

Ensuite, on est passé à l’interface web de notre application, en commençant par le Controller d’Etudiant :

package com.example.tp8.Web;  
  
import com.example.tp8.Entities.Etudiant;  
import com.example.tp8.Repositories.EtudiantRepository;  
import lombok.AllArgsConstructor;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.PageRequest;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.validation.BindingResult;  
import org.springframework.web.bind.annotation.\*;  
  
import javax.validation.Valid;  
import java.util.List;  
  
@Controller  
@AllArgsConstructor  
public class EtudiantController {  
 private EtudiantRepository etudiantRepository;  
  
 @GetMapping(path = "/user/index")  
 public String etudiants(Model model,  
 @RequestParam(name = "page", defaultValue = "0") int page,  
 @RequestParam(name = "size", defaultValue = "5") int size,  
 @RequestParam(name = "keyword", defaultValue = "") String keyword ){  
 Page<Etudiant> pageEtudiants=etudiantRepository.findByNomContains(keyword, PageRequest.*of*(page, size));  
 model.addAttribute("listEtudiants",pageEtudiants.getContent());  
 model.addAttribute("pages", new int[pageEtudiants.getTotalPages()]);  
 model.addAttribute("currentPage", page);  
 model.addAttribute("keyword", keyword);  
 return "etudiants";  
 }  
  
 //@GetMapping("/admin/delete")  
 @RequestMapping(value = "/admin/delete", method = RequestMethod.*DELETE*)  
 public String delete(Long id, String keyword, int page) {  
 etudiantRepository.deleteById(id);  
 return "redirect:/user/index?page="+page+"&keyword="+keyword;  
 }  
  
 @GetMapping("/")  
 public String home() {  
 return "home";  
 }  
  
 @GetMapping("/user/etudiants")  
 @ResponseBody  
 public List<Etudiant> listEtudiants(){  
 return etudiantRepository.findAll();  
 }  
  
  
  
 @GetMapping("/admin/formEtudiant")  
 public String formEtudiant(Model model) {  
 model.addAttribute("etudiant",new Etudiant());  
 return "formEtudiant";  
 }  
  
 @PostMapping(path = "/admin/save")  
 public String save(Model model, @Valid Etudiant etudiant, BindingResult bindingResult,  
 @RequestParam(name = "page", defaultValue = "0")int page,  
 @RequestParam(name = "keyword", defaultValue = "")String keyword ) {  
 if(bindingResult.hasErrors()) return "formEtudiant";  
 etudiantRepository.save(etudiant);  
 return "redirect:/user/index?page="+page+"&keyword="+keyword;  
 }  
  
 @GetMapping( "/admin/edit")  
 public String editEtudiant(Model model, Long id, int page,String keyword ) {  
 Etudiant etudiant=etudiantRepository.findById(id).orElse(null);  
 if(etudiant==null) throw new RuntimeException("etudiant introuvable");  
 model.addAttribute("etudiant",etudiant);  
 model.addAttribute("page",page);  
 model.addAttribute("keyword",keyword);  
 return "editEtudiant";  
 }  
}

Et ensuite le Controller de Security :

package com.example.tp8.Web;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
  
@Controller  
public class securityController {  
  
 @GetMapping("/403")  
 public String notAuthorized(){  
 return "403";  
 }  
}

Pour assurer le côté sécurité de l’application, on a créé un package contenant tout le nécessaire.

Tout d’abord, les entités de AppUser et AppRole, et leur repositories :

* AppUser :

package com.example.tp8.security.Entities;  
  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
@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:

package com.example.tp8.security.Entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.\*;  
  
@Entity  
@Data @AllArgsConstructor @NoArgsConstructor  
public class AppRole {  
 @Id @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long roleID;  
 @Column(unique = true)  
 private String roleName;  
 private String description;  
}

* AppUser Repository:

package com.example.tp8.security.Repositories;  
  
import com.example.tp8.security.Entities.AppUser;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface AppUserRepository extends JpaRepository<AppUser,String> {  
 AppUser findByUsername(String username);  
}

* AppRole Repository:

package com.example.tp8.security.Repositories;  
  
import com.example.tp8.security.Entities.AppRole;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface AppRoleRepository extends JpaRepository<AppRole,Long> {  
 AppRole findByRoleName(String roleName);  
}

Pour passer après à security service :

* Interface Security Service :

package com.example.tp8.security.Services;  
  
import com.example.tp8.security.Entities.AppRole;  
import com.example.tp8.security.Entities.AppUser;  
  
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);  
}

* Son implementation:

package com.example.tp8.security.Services;  
  
import com.example.tp8.security.Entities.AppRole;  
import com.example.tp8.security.Entities.AppUser;  
import com.example.tp8.security.Repositories.AppRoleRepository;  
import com.example.tp8.security.Repositories.AppUserRepository;  
import groovy.util.logging.Slf4j;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.UUID;  
  
@Service  
@Slf4j // pour loguer les informations  
@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, PasswordEncoder passwordEncoder) {  
 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;  
 }  
  
 @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);  
 }  
  
 @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 Implementation:

package com.example.tp8.security.Services;  
  
import com.example.tp8.security.Entities.AppUser;  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.authority.SimpleGrantedAuthority;  
import org.springframework.security.core.userdetails.User;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.core.userdetails.UsernameNotFoundException;  
import org.springframework.stereotype.Service;  
  
import java.util.Collection;  
import java.util.stream.Collectors;  
  
@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()  
 .stream()  
 .map(role -> new SimpleGrantedAuthority(role.getRoleName()))  
 .collect(Collectors.*toList*());  
  
 User user = new User(appUser.getUsername(),appUser.getPassword(),authorities1);  
 return user;  
 }  
}

Et la configuration de sécurité :

package com.example.tp8.security;  
  
import com.example.tp8.security.Services.UserDetailsServiceImpl;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.crypto.password.PasswordEncoder;  
  
import javax.sql.DataSource;  
  
@Configuration  
@EnableWebSecurity  
public class securityConfig extends WebSecurityConfigurerAdapter {  
  
 @Autowired  
 private DataSource dataSource;  
 @Autowired  
 private UserDetailsServiceImpl userDetailsService;  
 @Autowired  
 private PasswordEncoder passwordEncoder;  
  
  
 @Override  
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 /\*  
 String encodedPWD=passwordEncoder.encode("1234");  
 //stocker en mémoire les utilisateurs qui ont acces à l'application  
 auth.inMemoryAuthentication().withUser("user").password(encodedPWD).roles("USER");  
 auth.inMemoryAuthentication().withUser("admin").password(passwordEncoder.encode("5678")).roles("USER","ADMIN");  
 \*/  
  
 /\*auth.userDetailsService(new UserDetailsService() {  
 @Override  
 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
 return null;  
 }  
 });  
  
 auth.jdbcAuthentication()  
 .dataSource(dataSource)  
 .usersByUsernameQuery("select username as principal, password as credentials, active from users where username=?") // principal <=> username  
 .authoritiesByUsernameQuery("select username as principal, role as role from users\_roles where username=?")  
 .rolePrefix("Role\_")  
 .passwordEncoder((passwordEncoder));  
  
 \*/  
  
 auth.userDetailsService(userDetailsService);  
 }  
  
 @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("/").permitAll();  
 http.authorizeRequests().antMatchers("/webjars/\*\*").permitAll();  
  
 //ces paths sont accessibles juste pour les utilisateurs qui ont role ADMIN  
 //http.authorizeRequests().antMatchers("/admin/\*\*").hasRole("ADMIN");  
 http.authorizeRequests().antMatchers("/admin/\*\*").hasAuthority("ADMIN");  
  
 //ces paths sont accessibles juste pour les utilisateurs qui ont role USER  
 //http.authorizeRequests().antMatchers("/user/\*\*").hasRole("USER");  
 http.authorizeRequests().antMatchers("/user/\*\*").hasAuthority("USER");  
  
 //toute requete http nécessite une authentification  
 http.authorizeRequests().anyRequest().authenticated();  
 http.exceptionHandling().accessDeniedPage("/403");  
  
 }  
  
}

On passe maintenant à l’affichage web, on a plusieurs fichiers HTML :

* 403.html :

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"  
 layout:decorate="template1"  
 xmlns:sec="http://www.thymeleaf.org/extras/spring-security" >  
<head>  
 <meta charset="UTF-8">  
 <title>Accueil</title>  
</head>  
<body>  
<div layout:fragment="content1">  
 <div class="container">  
 <h2 class="text-danger">vous n'etes pas autorisés à accéder à cette page</h2>  
 </div>  
</div>  
</body>  
</html>

* EditEtudiant.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>Nouveau etudiant</title>  
</head>  
<body>  
 <div layout:fragment="content1">  
 <div class="col-md-6 offset-3">  
 <form method="post" th:action="@{ /admin/save(page=${page},keyword=${keyword}) }">  
 <div class="mb-3">  
 <label for="exampleId" class="form-label">ID</label>  
 <input type="text" class="form-control" id="exampleId" name="id" th:value="${etudiant.stu\_ID}">  
 </div>  
 <div class="mb-3">  
 <label for="exampleNom" class="form-label">Nom</label>  
 <input type="text" class="form-control" id="exampleNom" name="nom" th:value="${etudiant.nom}">  
 <span th:errors="${etudiant.nom}"></span>  
 </div>  
 <div class="mb-3">  
 <label for="exampleNom" class="form-label">Prenom</label>  
 <input type="text" class="form-control" id="examplePrenom" name="prenom" th:value="${etudiant.prenom}">  
 <span th:errors="${etudiant.prenom}"></span>  
 </div>  
 <div class="mb-3">  
 <label for="exampleNom" class="form-label">EMAIL</label>  
 <input type="text" class="form-control" id="exampleEmain" name="email" th:value="${etudiant.email}">  
 <span th:errors="${etudiant.email}"></span>  
 </div>  
 <div class="mb-3">  
 <label for="exampleDate" class="form-label">Date de naissance</label>  
 <input type="date" class="form-control" id="exampleDate" name="datenaissance" th:value="${etudiant.date\_naissance}">  
 <span th:errors="${etudiant.date\_naissance}"></span>  
 </div>  
 <div class="mb-3 form-check">  
 <input type="checkbox" class="form-check-input" id="exampleCheck1" name="regle"th:checked="${etudiant.regle}">  
 <label class="form-check-label" for="exampleCheck1">En Regle</label>  
 <span th:errors="${etudiant.regle}"></span>  
 </div>  
 <div class="mb-3">  
 <label for="exampleScore" class="form-label">Genre</label>  
 <input type="text" class="form-control" id="exampleScore" name="genre"th:value="${etudiant.genre}">  
 <span th:errors="${etudiant.genre}"></span>  
 </div>  
 <button type="submit" class="btn btn-primary">Submit</button>  
 </form>  
 </div>  
 </div>  
</body>  
</html>

* Etudiants.html:

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"  
 layout:decorate="template1"  
 xmlns:sec="http://www.thymeleaf.org/extras/spring-security" >  
<head>  
 <meta charset="UTF-8">  
 <title>Accueil</title>  
</head>  
<body>  
<div layout:fragment="content1">  
<div class="container mt-2">  
 <div class="card">  
 <div class="card-header">Liste des Etudiants</div>  
 <div class="card-body">  
 <form method="get" th:action="@{/user/index}">  
 <label>Key Word</label>  
 <input type="text" name="keyword" th:value="${keyword}">  
 <button type="submit" class="btn btn-primary">Chercher</button>  
 </form>  
 <table class="table">  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>Nom</th>  
 <th>Prenom</th>  
 <th>Email</th>  
 <th>DateNaissance</th>  
 <th>En Regle</th>  
 <th>Genre</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="p:${listEtudiants}">  
 <td th:text="${p.stu\_ID}"></td>  
 <td th:text="${p.nom}"></td>  
 <td th:text="${p.prenom}"></td>  
 <td th:text="${p.email}"></td>  
 <td th:text="${p.date\_naissance}"></td>  
 <td th:text="${p.regle}"></td>  
 <td th:text="${p.genre}"></td>  
 <td sec:authorize="hasAuthority('ADMIN')">  
 <a onclick="return confirm('etes vous sure?')" class="btn btn-danger" th:href="@{/admin/delete(id=${p.stu\_ID}, keyword=${keyword}, page=${currentPage} )}">  
 Delete  
 </a>  
 </td>  
 <td sec:authorize="hasAuthority('ADMIN')">  
 <a class="btn btn-success" th:href="@{/admin/edit(id=${p.stu\_ID} , keyword=${keyword}, page=${currentPage} )}">  
 Edit  
 </a>  
 </td>  
 </tr>  
 </tbody>  
 </table>  
 <ul class="nav nav-pills">  
 <li th:each="page,status:${pages}">  
 <a th:class="${status.index==currentPage?'btn btn-info ms-1' : 'btn btn-outline-info ms-1'}"  
 th:text="${status.index}"  
 th:href="@{/user/index(page=${status.index}, keyword=${keyword})}"  
 > </a>  
 </li>  
 </ul>  
 </div>  
 </div>  
</div>  
</div>  
</body>  
</html>

* FormEtudiant.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>Nouveau etudiant</title>  
</head>  
<body>  
 <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="${etudiant.nom}">  
 <span th:errors="${etudiant.nom}"></span>  
 </div>  
 <div class="mb-3">  
 <label for="exampleNom" class="form-label">Prenom</label>  
 <input type="text" class="form-control" id="examplePrenom" name="prenom" th:value="${etudiant.prenom}">  
 <span th:errors="${etudiant.prenom}"></span>  
 </div>  
 <div class="mb-3">  
 <label for="exampleNom" class="form-label">Nom</label>  
 <input type="text" class="form-control" id="exampleEmail" name="email" th:value="${etudiant.email}">  
 <span th:errors="${etudiant.email}"></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="${etudiant.date\_naissance}">  
 <span th:errors="${etudiant.date\_naissance}"></span>  
 </div>  
 <div class="mb-3 form-check">  
 <input type="checkbox" class="form-check-input" id="exampleCheck1" name="regle"th:checked="${etudiant.regle}">  
 <label class="form-check-label" for="exampleCheck1">En Regle</label>  
 <span th:errors="${etudiant.regle}"></span>  
 </div>  
 <div class="mb-3">  
 <label for="exampleScore" class="form-label">Genre</label>  
 <input type="text" class="form-control" id="exampleScore" name="genre"th:value="${etudiant.genre}">  
 <span th:errors="${etudiant.genre}"></span>  
 </div>  
 <button type="submit" class="btn btn-primary">Submit</button>  
 </form>  
 </div>  
 </div>  
</body>  
</html>

* Home.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>Home</title>  
</head>  
<body>  
 <div layout:fragment="content1"></div>  
</body>  
</html>

* Template1.html:

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"  
 xmlns:sec="http://www.thymeleaf.org/extras/spring-security">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
 <link rel="stylesheet" type="text/css" href="/webjars/bootstrap/5.1.3/css/bootstrap.min.css">  
 <script src="/webjars/bootstrap/5.1.3/js/bootstrap.bundle.min.js"></script>  
</head>  
 <body>  
 <nav class="navbar navbar-expand-sm bg-dark navbar-dark">  
 <div class="container-fluid">  
 <ul class="navbar-nav">  
 <li class="nav-item">  
 <a class="nav-link active" th:href="@{/user/index}">Home</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link" href="#">Link</a>  
 </li>  
 <li class="nav-item dropdown">  
 <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">  
 Etudiants  
 </a>  
 <ul class="dropdown-menu">  
 <li sec:authorize="hasAuthority('ADMIN')"><a class="dropdown-item" th:href="@{/admin/formEtudiant}">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" sec:authorize="isAuthenticated()">  
 <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">  
 <span sec:authentication="name"></span>  
 </a>  
 <ul class="dropdown-menu">  
 <li><a class="dropdown-item" th:href="@{/logout}">Logout</a></li>  
 </ul>  
 </li>  
 <li class="nav-item" >  
 <a class="nav-link active" sec:authorize="!isAuthenticated()" th:href="@{/login}">Login</a>  
 </li>  
 </ul>  
 </div>  
 </nav>  
 <section layout:fragment="content1">  
  
 </section>  
</body>  
</html>

Et pour finir, on a notre Application Main et propriété de l’application :

* Main :

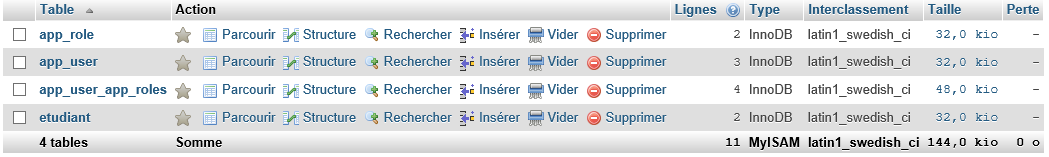
package com.example.tp8;  
  
import com.example.tp8.Entities.Etudiant;  
import com.example.tp8.Repositories.EtudiantRepository;  
import com.example.tp8.security.Services.SecurityService;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.annotation.Bean;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
  
import java.util.Date;  
  
@SpringBootApplication  
public class Tp8Application {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Tp8Application.class, args);  
 }  
  
 @Bean // au demarrage crée un objet de type PasswordEncoder  
 PasswordEncoder passwordEncoder(){  
 return new BCryptPasswordEncoder();  
 }  
  
 //@Bean  
 CommandLineRunner commandLineRunner(EtudiantRepository etudiantRepository){  
 return args -> {  
 etudiantRepository.save(  
 new Etudiant(null, "elkamouni","soukaina","soukaina@gmail.com",new Date(),"F",true)  
 );  
 etudiantRepository.save(  
 new Etudiant(null, "khadidi","salma","salma@gmail.com",new Date(),"F",true)  
 );  
 etudiantRepository.findAll().forEach(p ->{  
 System.*out*.println(p.getNom());  
 });  
 };  
 }  
  
 //@Bean  
 CommandLineRunner saveUsers(SecurityService securityService){  
 return args -> {  
 securityService.saveNewUser("Mohamed", "1234", "1234");  
 securityService.saveNewUser("Yassamine", "1234", "1234");  
 securityService.saveNewUser("Hassan", "1234", "1234");  
  
 securityService.saveNewRole("USER","");  
 securityService.saveNewRole("ADMIN","");  
  
 securityService.addRoleToUser("Mohamed","USER");  
 securityService.addRoleToUser("Mohamed","ADMIN");  
 securityService.addRoleToUser("Yassamine","USER");  
 securityService.addRoleToUser("Hassan","USER");  
  
 };  
 }  
}

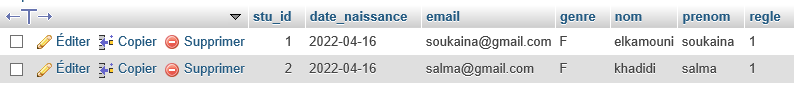
* Application Properties :

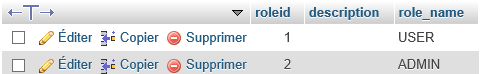
spring.datasource.url = jdbc:mysql://localhost:3309/etudiants?createDatabaseIfNotExist=true  
spring.datasource.username = root  
spring.datasource.password =  
server.port=8085  
spring.jpa.hibernate.ddl-auto = update  
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MariaDBDialect  
spring.jpa.show-sql=true  
#spring.main.allow-circular-references=true

1. **Affichage :**

On va voir tout d’abord notre base de données :



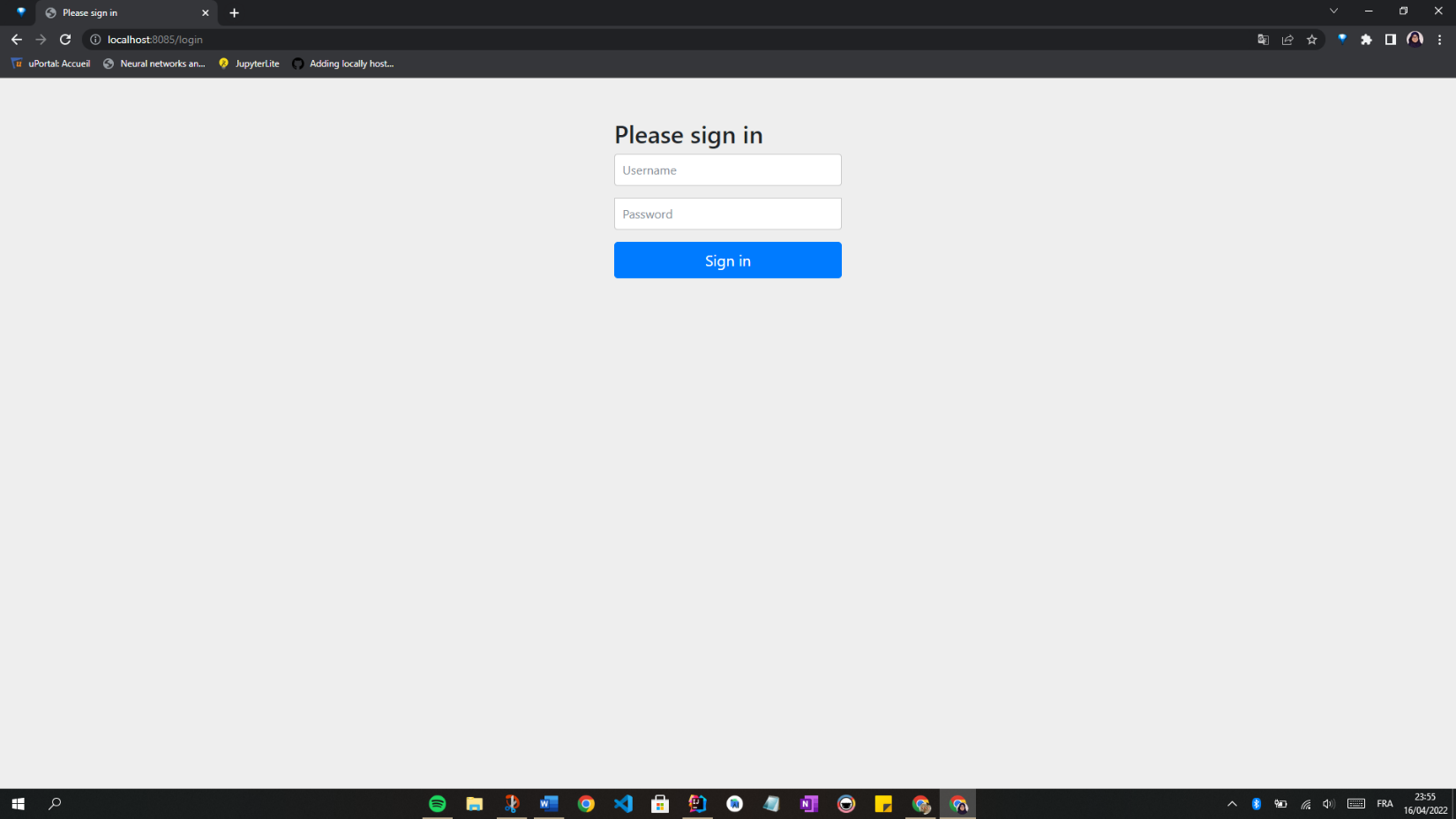




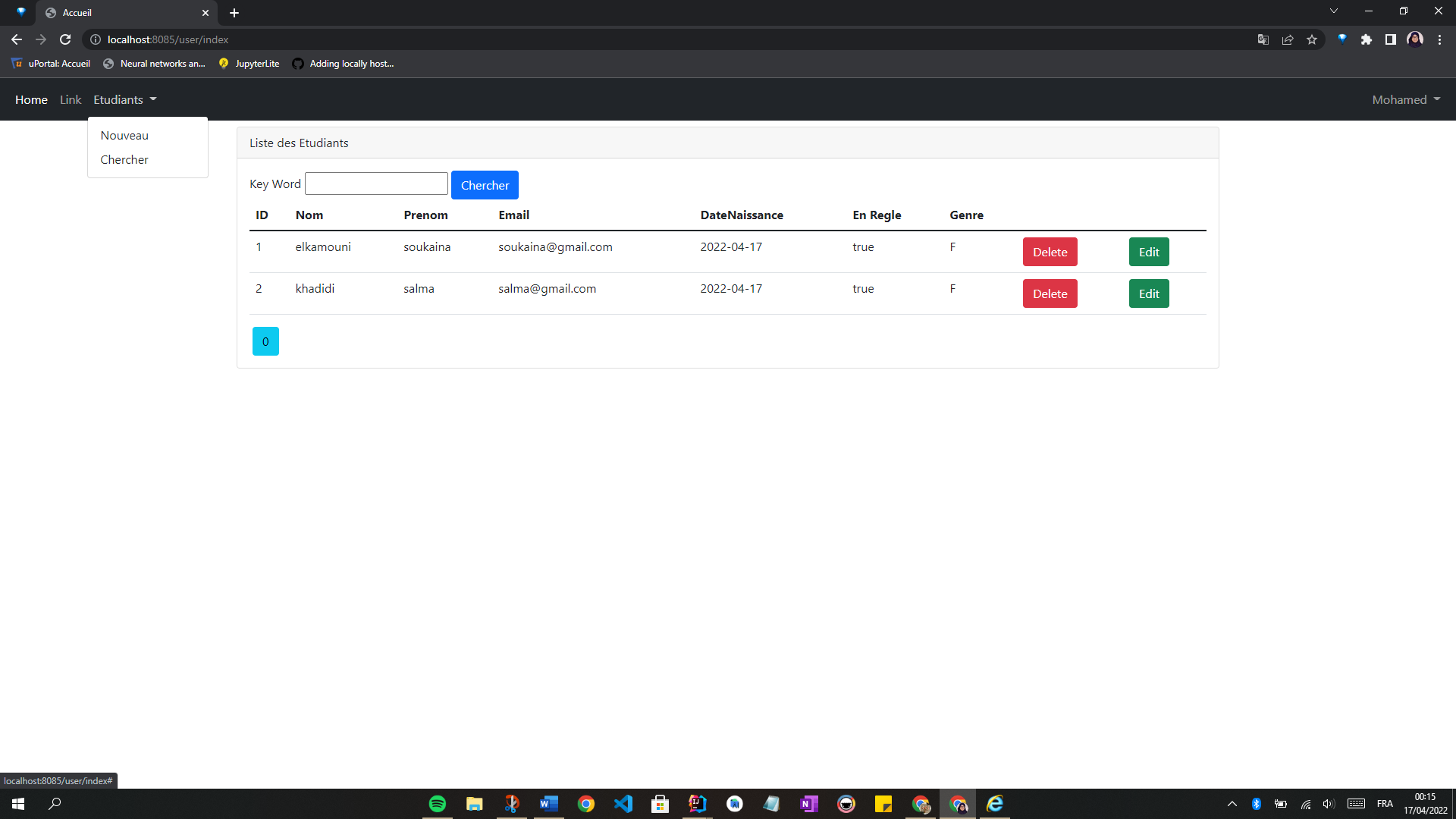




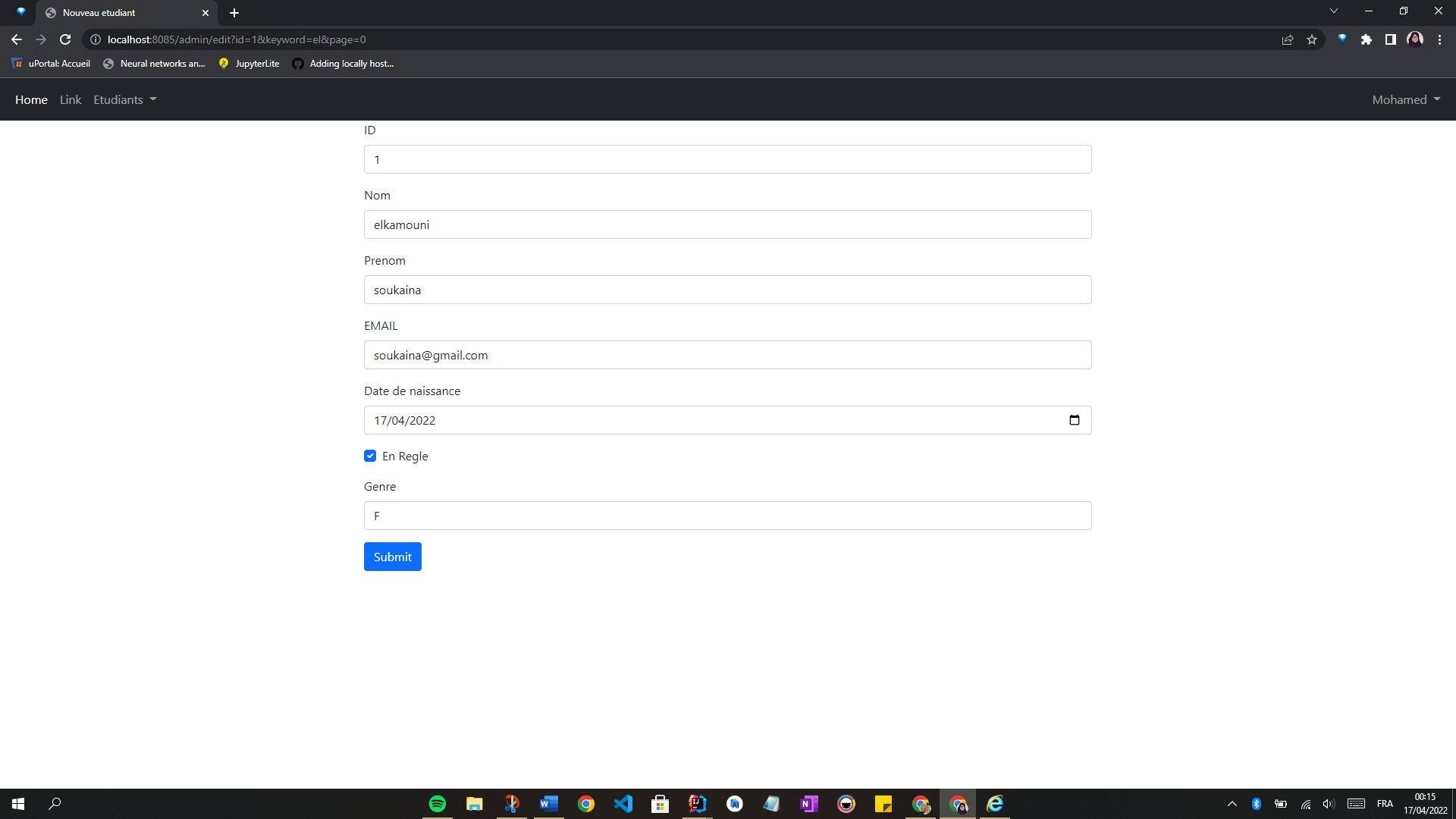
Et maintenant, l’interface Web :

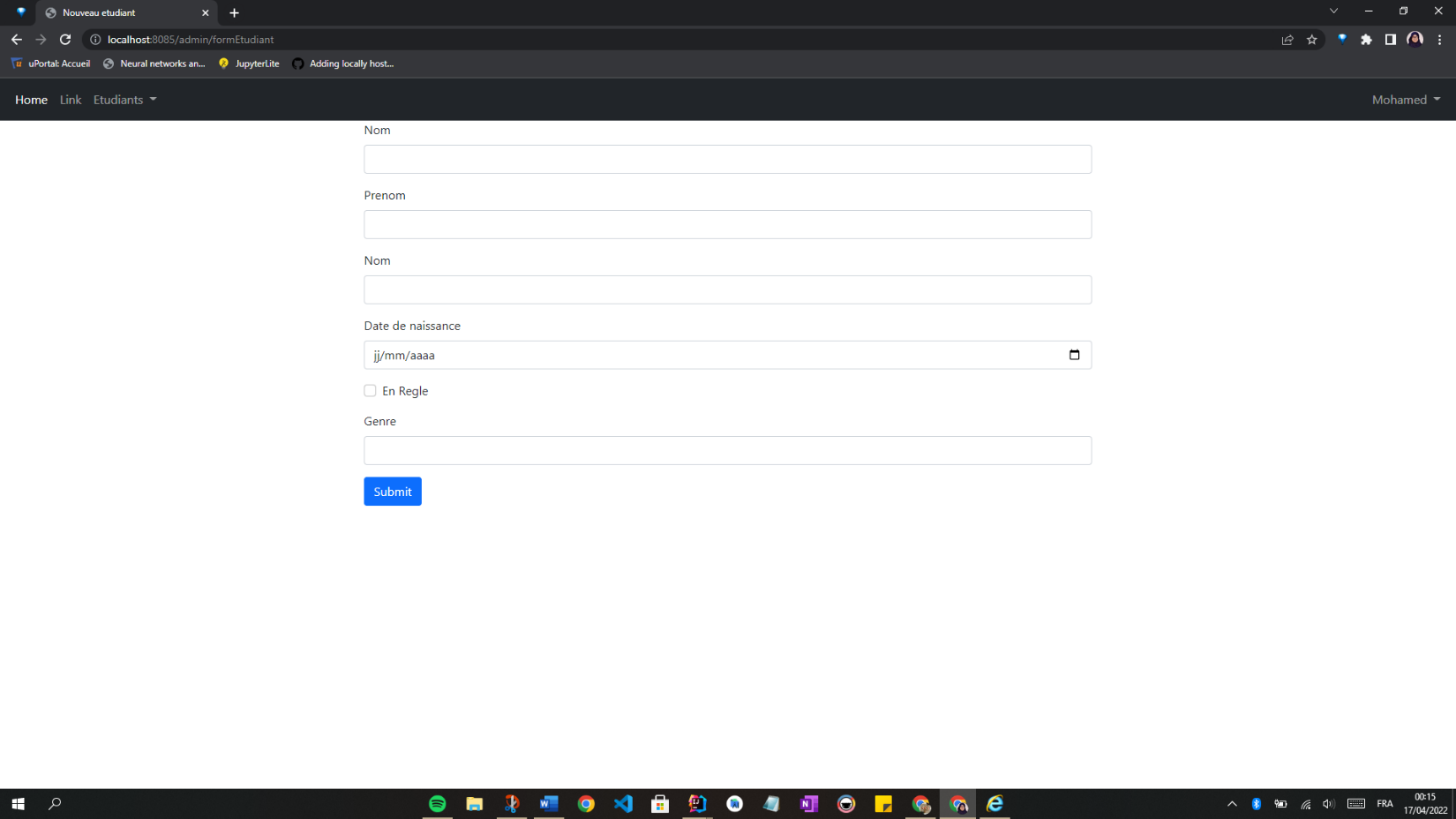


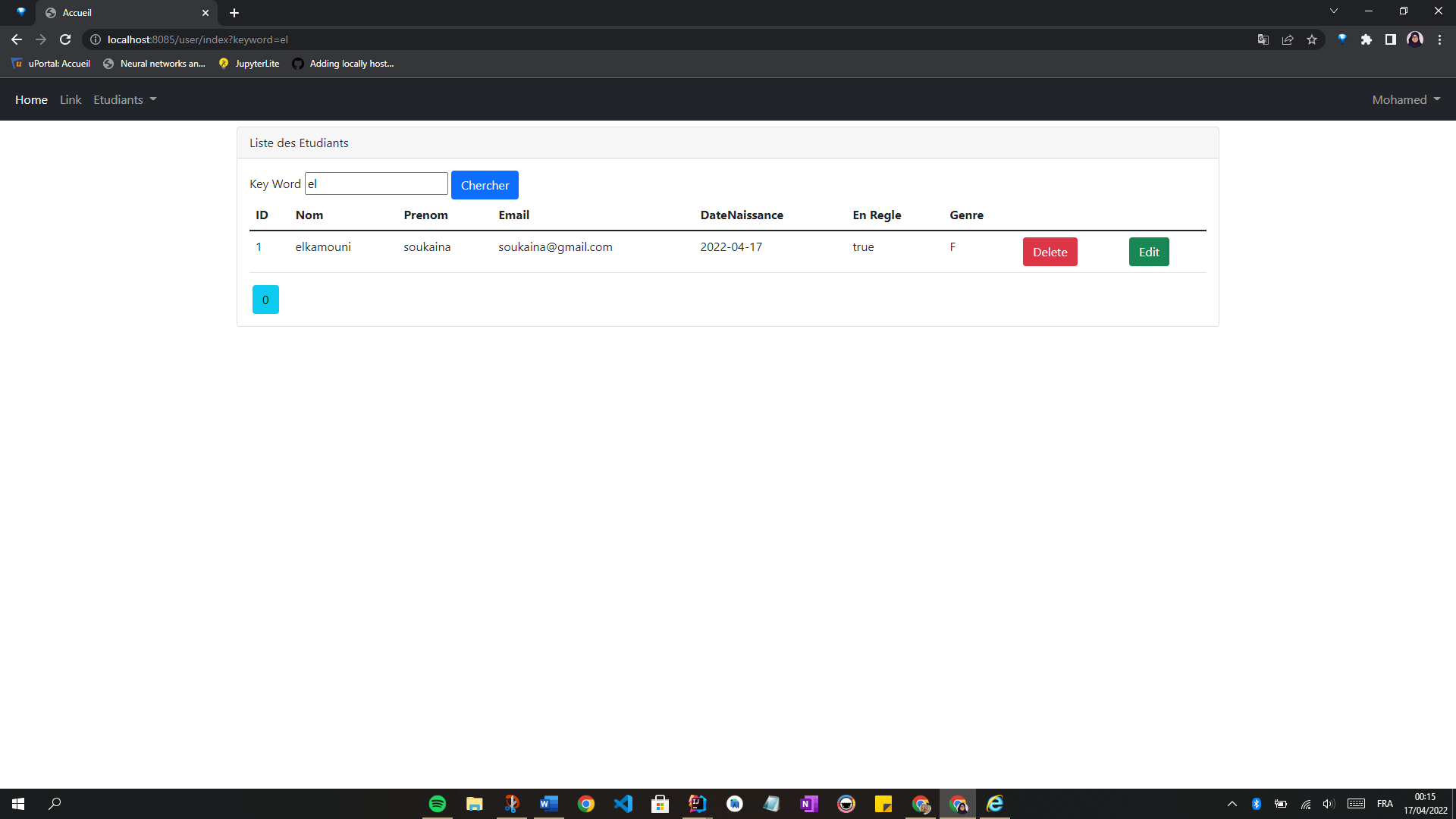
On se connecte tout d’abord à un compte ADMIN :



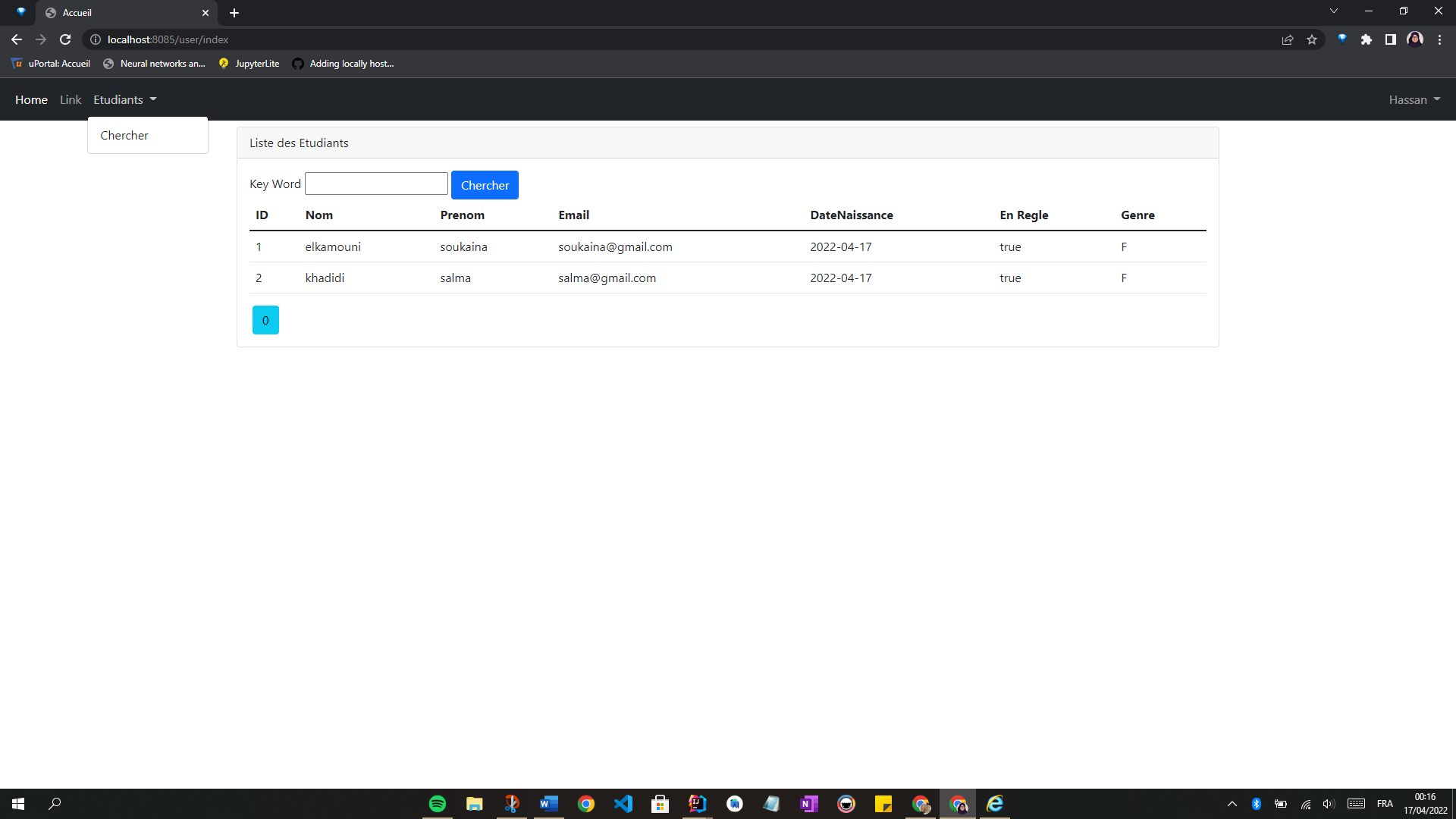
Le site nous affiche la liste des étudiants avec la possibilité de cherche, supprimer, éditer et ajouter un nouvel étudiant.







En se connectant à un compte USER normal, on peut seulement chercher et afficher la liste des étudiants :



**Mon GitHub :** https://github.com/SoukainaElkm/Soukaina-ELKAMOUNI.JEE