

Compte Rendu

Spring MVC, Spring Data JPA et
Spring Security

Students Management

Tarik FERTAH
II-BDCC2

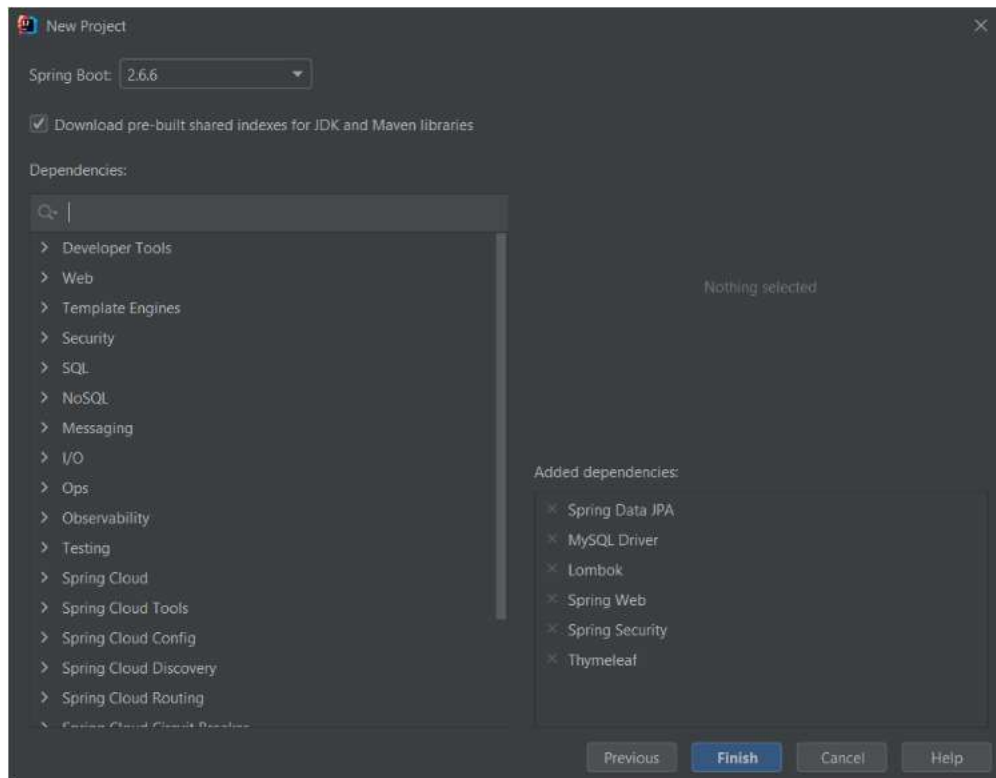
Année Universitaire : 2022 - 2023

Enoncé de l'App

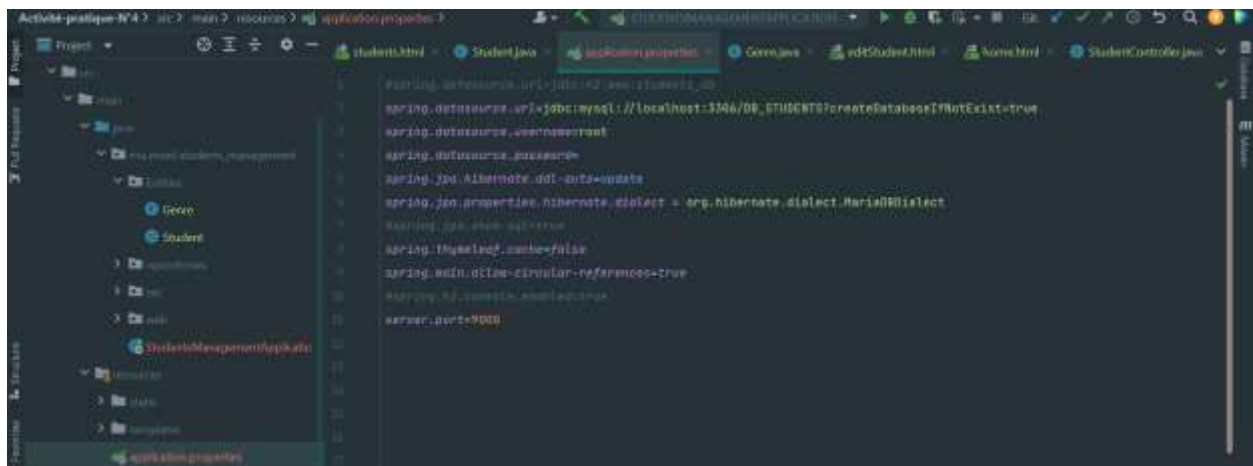
Créer une application Web basée sur Spring MVC, Spring Data JPA et Spring Security qui permet de gérer des étudiants. Chaque étudiant est défini par :

- Son id
- Son nom
- Son prénom
- Son email
- Sa date naissance
- Son genre: MASCULIN ou FEMININ
- Un attribut qui indique s'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 UserDetails Service

1 Dépendances :

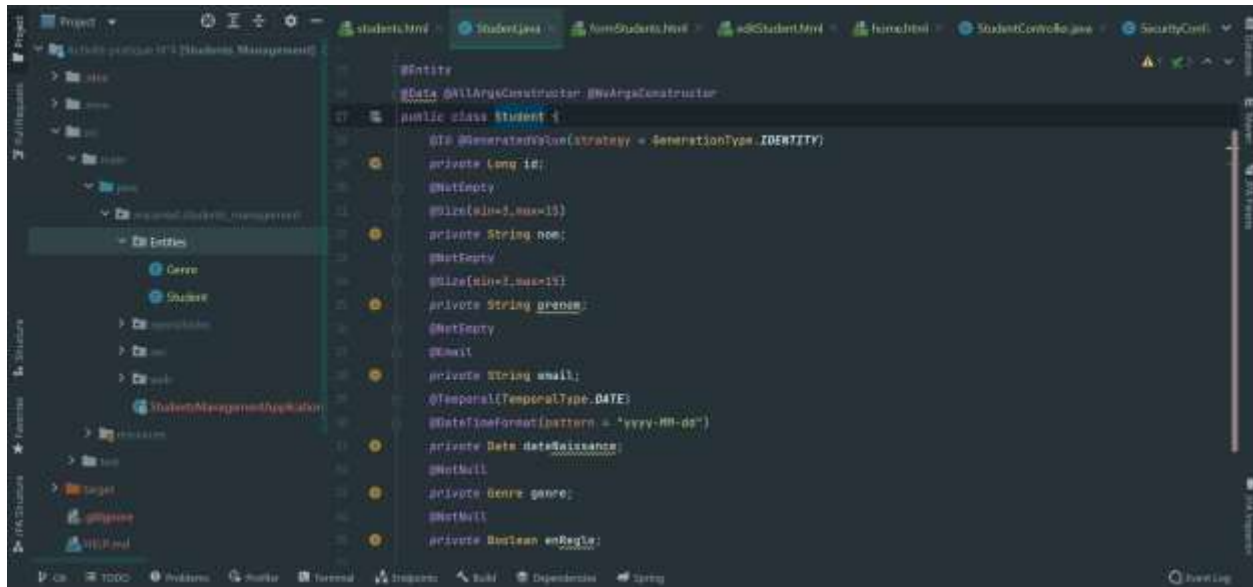


2 Configuration

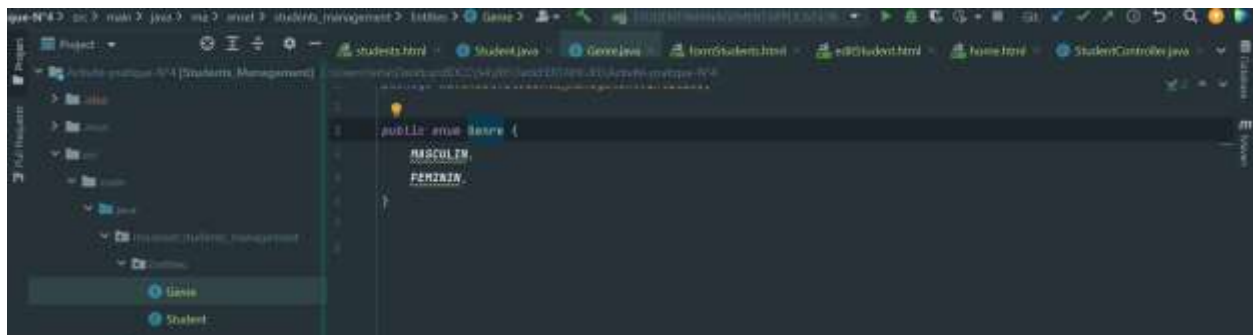


3 Création des entités

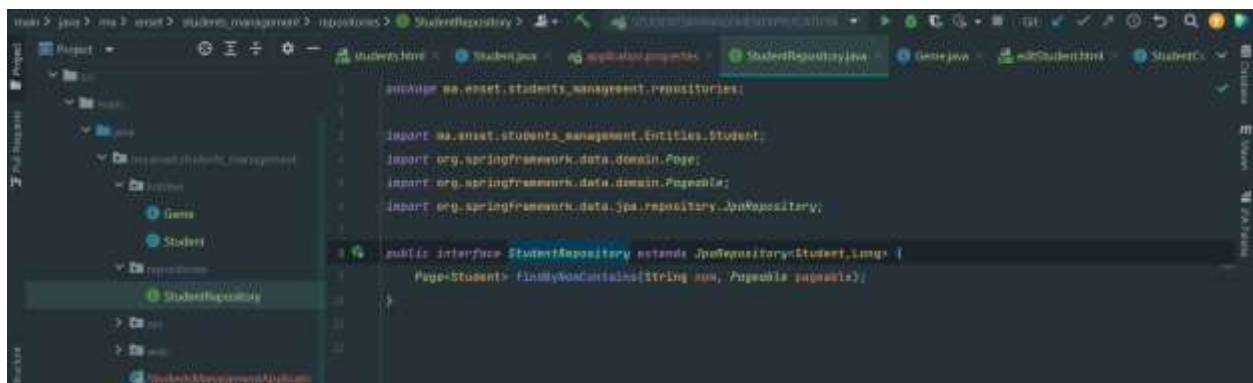
Entité « **student** » => entities/Student



Entité « **genre** » => entities/Genre



4 Création du Repository de l'entité « student » => repositories/ StudentRepository



5 Création du Controller => web/StudentController

- Home

```
@GetMapping(path = "${~}/")
public String Home() { return "home"; }
```

- Liste des étudiant
- Recherche par keyword
- Pagination

```
@GetMapping(path = "${~}/user/index")
public String students(Model model,
    @RequestParam(name = "page", defaultValue = "0") int page,
    @RequestParam(name = "size", defaultValue = "5") int size,
    @RequestParam(name = "keyword", defaultValue = "") String keyword){
    Page<Student> pageStudents = studentRepository.findByNomContains(keyword, PageRequest.of(page, size));
    model.addAttribute( attributeName: "listStudents", pageStudents.getContent());
    model.addAttribute( attributeName: "pages", new int[] {pageStudents.getTotalPages()});
    model.addAttribute( attributeName: "currentPage", page);
    model.addAttribute( attributeName: "keyword", keyword);
    return "students";
}
```

- Delete Student By Id:

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

- Add new Student

```
@GetMapping("${~}/admin/formStudents")
public String formStudents(Model model){
    model.addAttribute( attributeName: "student", new Student());
    return "formStudents";
}
```

- Edit Student

```

@GetMapping("/{id}/admin/editStudent")
public String editStudent(Model model, Long id, String keyword, int page){
    Student student = studentRepository.findById(id).orElse( null);
    if (student == null) throw new RuntimeException("Student introuvable");
    model.addAttribute( attributeName: "student", student);
    model.addAttribute( attributeName: "page", page);
    model.addAttribute( attributeName: "keyword", keyword);
    return "editStudent";
}

```

- Insertion des étudiants dans la base de données

```

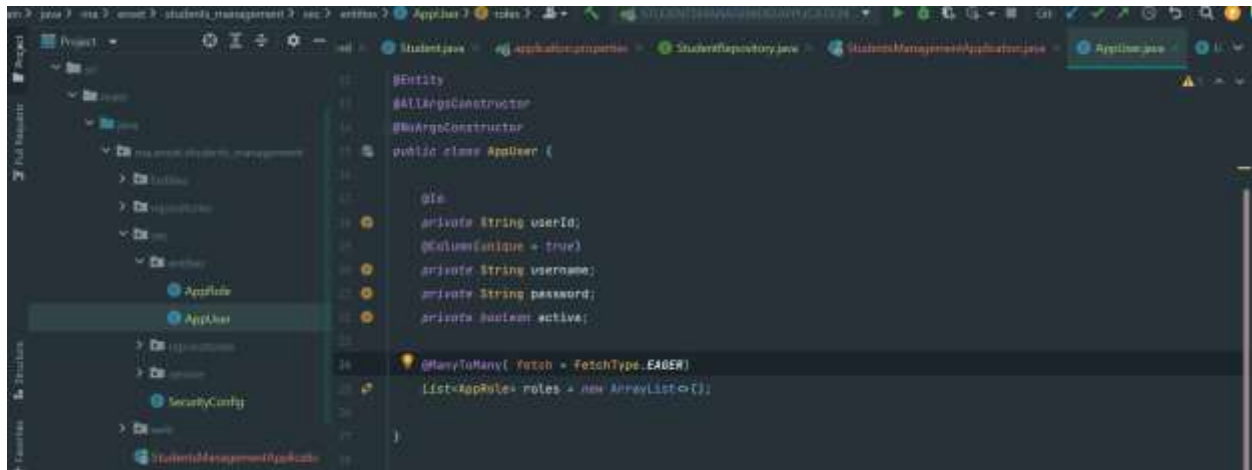
public static void main(String[] args) { SpringApplication.run(StudentsManagementApplication.class, args); }

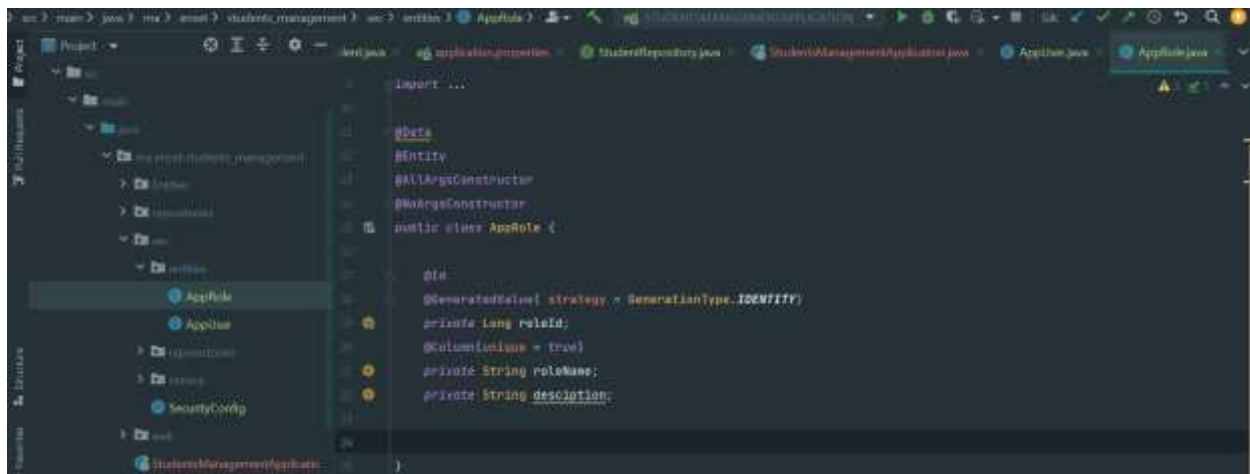
@Bean
CommandLineRunner commandLineRunner(StudentRepository studentRepository) {
    return args ->{
        studentRepository.save(new Student( id null, nom: "khalil", prenom: "khalil", email: "khalil.kha@gmail.com", new Date()));
        studentRepository.save(new Student( id null, nom: "saloua", prenom: "saloua", email: "saloua@gmail.com", new Date()));
        studentRepository.save(new Student( id null, nom: "ayoub", prenom: "ayoub", email: "ayoub@gmail.com", new Date()));
        studentRepository.save(new Student( id null, nom: "sara", prenom: "sara", email: "sara@gmail.com", new Date()));
    };
}

```

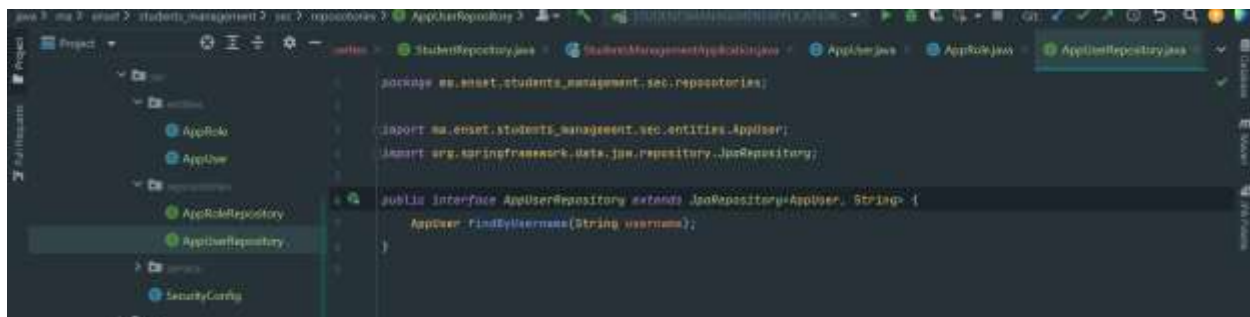
6 Spring Security

- Création des entités : Entité « User » et Entité « Role »

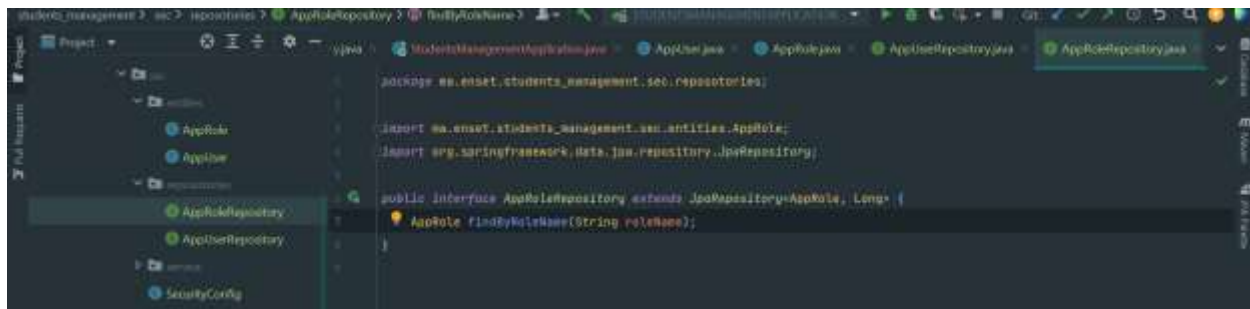




- UserRepository



- RoleRepository



- SecurityConfig


```

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 {
        auth.userDetailsService(userDetailsService);
        //auth.jdbcAuthentication()
        //dataSource(dataSource);
    }
}

```

```

@Override
protected void configure(HttpSecurity http) throws Exception {
    http.authorizeRequests().antMatchers("/").permitAll();
    http.formLogin().loginPage("/login").permitAll();
    http.logout().logoutSuccessUrl("/");
    http.authorizeRequests().antMatchers("/css/**").permitAll();
    http.authorizeRequests().antMatchers("/static/**").permitAll();
    http.authorizeRequests().antMatchers("/table/css/**").permitAll();
    http.authorizeRequests().antMatchers("/webjars/**").permitAll();
    http.authorizeRequests().antMatchers("/admin/**").hasAuthority("ADMIN");
    http.authorizeRequests().antMatchers("/user/**").hasAuthority("USER");
    http.authorizeRequests().anyRequest().authenticated();
    http.exceptionHandling().accessDeniedPage("/403");
}

```

Définition de la fonction BEAN qui va fournir à l'application toujours un **PasswordEncoder** de type **BCryptPasswordEncoder** pour **Hasher** les mots de passe, et une deuxième fonction BEAN pour créer deux premiers utilisateurs avec deux rôles **ADMIN** et **USER** :

```

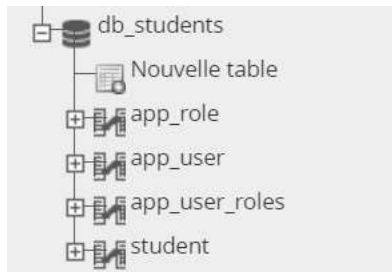
@Bean
CommandLineRunner saveUsers (SecurityService securityService) {
    return args -> {
        securityService.saveNewUser( username: "admin", password: "admin", rePassword: "admin");
        securityService.saveNewUser( username: "user1", password: "user", rePassword: "user");

        securityService.saveNewRole( roleName: "ADMIN", description: "This is the admin role !");
        securityService.saveNewRole( roleName: "USER", description: "This is the user role !");

        securityService.addRoleToUser( userName: "admin", roleName: "ADMIN");
        securityService.addRoleToUser( userName: "admin", roleName: "USER");
        securityService.addRoleToUser( userName: "user1", roleName: "USER");
    };
}

```


Au relancement de l'application, trois nouvelles tables sont créées, avec les données dedans :



- Table « app_user »

✓ Affichage des lignes 0 - 1 (total de 2, traitement en 0,0012 seconde(s))

```
SELECT * FROM "app_user"
```

☐ Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

☐ Afficher les données | Sélectionner lignes: 25 | Chercher dans cette table | Aucun(e)

+ Options

	user_id	active	password	username
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	63c120fc-825d-4cd5-b67e-b6a3b0227496	1	\$2a\$10\$esZOINh8PVIDY8u85pEHOY3MawXjGMT9LgMx5g...	admin
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	c4106a88-916d-4d8d-8c1e-18c24d68e8a	1	\$2a\$10\$ihrtgOL6kh9D4CKlk9PO54sMhuFLFHTK0F6e9Ize...	user1

- Table « app_role »

✓ Affichage des lignes 0 - 1 (total de 2, traitement en 0,0006 seconde(s))

```
SELECT * FROM "app_role"
```

☐ Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

☐ Afficher les données | Sélectionner lignes: 25 | Chercher dans cette table | Aucun(e)

+ Options

	role_id	description	role_name
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	1	This is the admin role!	ADMIN
<input type="checkbox"/> Éditer <input type="checkbox"/> Copier <input type="checkbox"/> Supprimer	2	This is the user role!	USER

- Table app_user_roles

✓ Affichage des lignes 0 - 2 (total de 3, traitement en 0,0008 seconde(s).)

```
SELECT * FROM `app_user_roles`
```

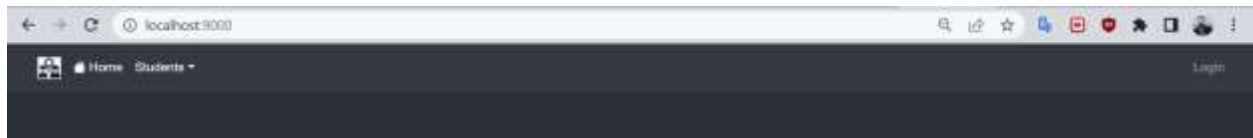
Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

25

Options

app_user_id	roles_role_id
63c120fc-821d-4cd5-b67e-b6a3b0227496	1
63c120fc-821d-4cd5-b67e-b6a3b0227496	2
r4106a88-916d-4d8d-8c1e-18c34df68e8a	2

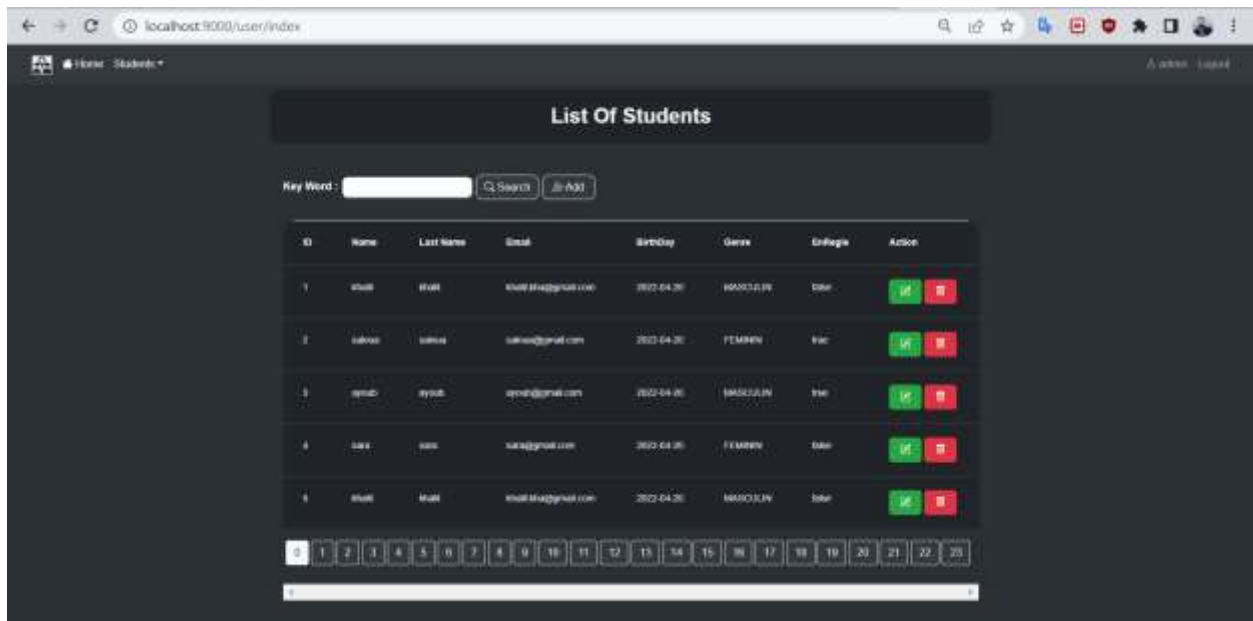
7 Page Home



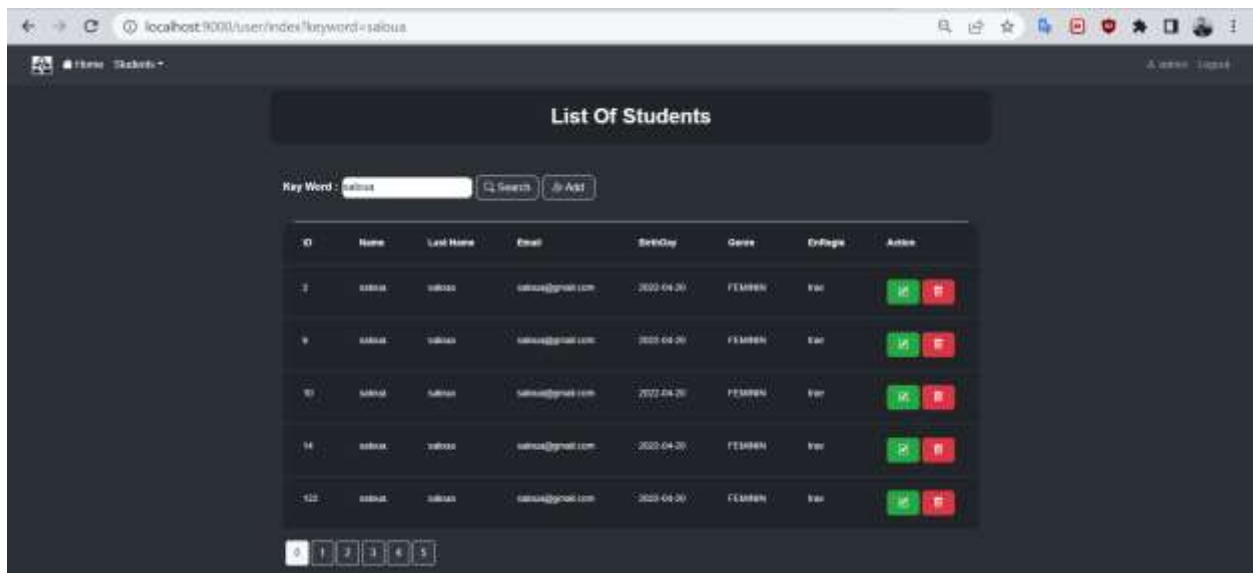
8 Page login (avec le role Admin)



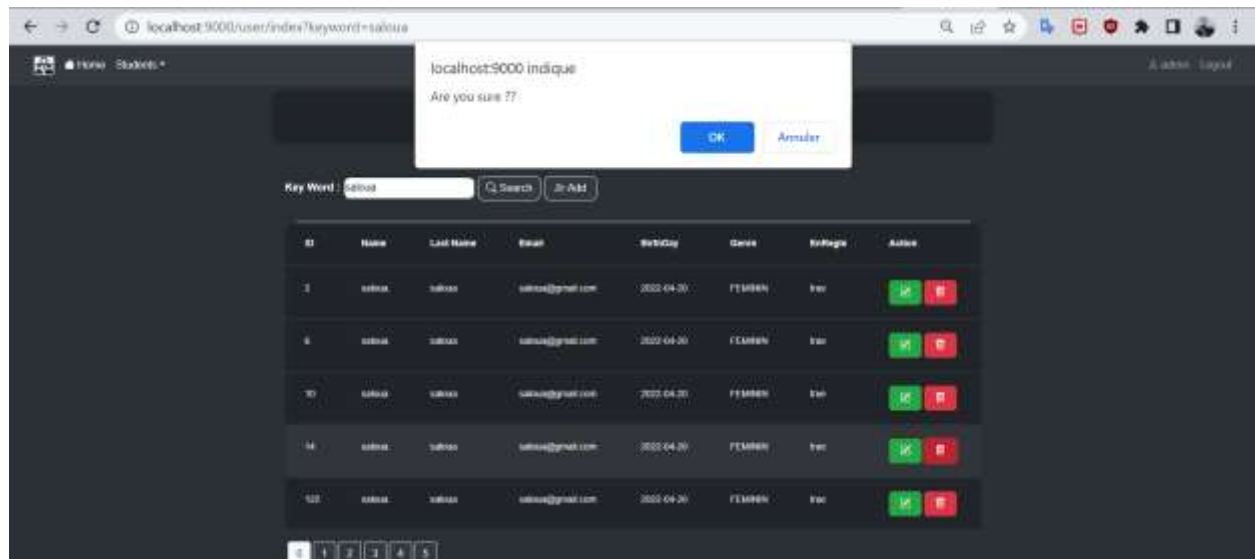
8 Authentification en tant que Admin (rechercher, ajouter, modifier et supprimer)



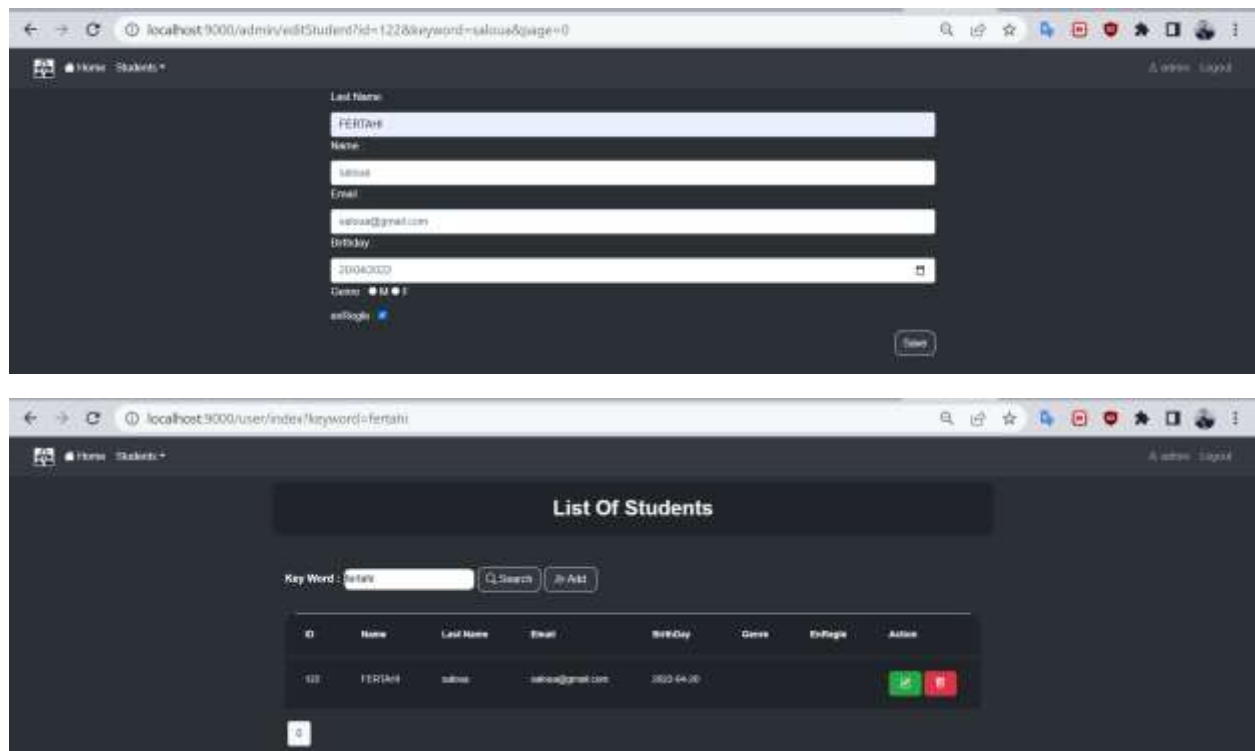
- Rechercher (étudiante Saloua)



- Supprimer l'étudiante Saloua ayant comme Id 14



- Modifier l'étudiante Saloua ayant comme Id 122



- Ajouter l'étudiant « Mohammed »

localhost:9000/admin/formStudents

Home Students admin Logout

Last Name
Sayfi

Name
Mohammed

Email
mohammed.sayfi@gmail.com

Birthday
24/04/2022

Genre : ☒ M ☐ F

enRegle : ☒

Save

localhost:9000/user/index?page=23&keyword=

Home Students admin Logout

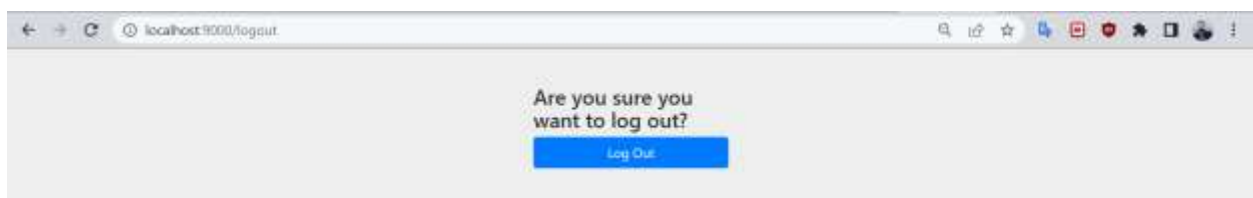
List Of Students

Key Word : Search Add

ID	Name	Last Name	Email	Birthday	Genre	EnRegle	Action
223	ayoub	ayoub	ayoub@gmail.com	2022-04-20	MASCULIN	True	<input checked="" type="checkbox"/> <input type="checkbox"/>
224	lara	lara	lara@gmail.com	2022-04-20	FEMMIN	True	<input checked="" type="checkbox"/> <input type="checkbox"/>
225	Sayfi	Mohammed	mohammed.sayfi@gmail.com	2022-04-24	MASCULIN	True	<input checked="" type="checkbox"/> <input type="checkbox"/>

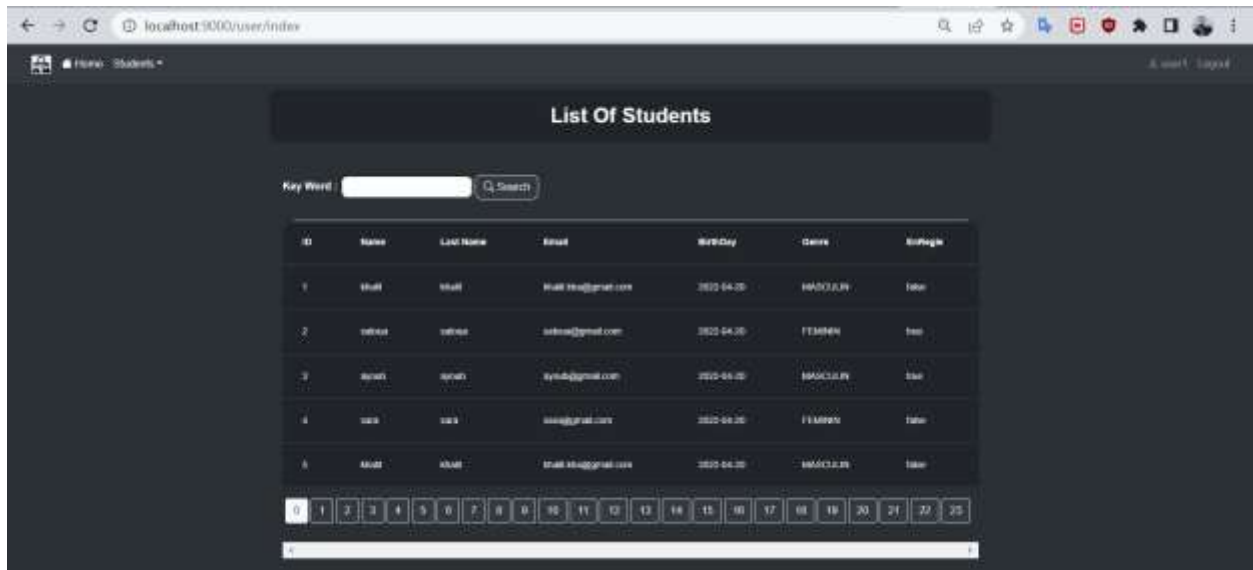
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

9 Logout

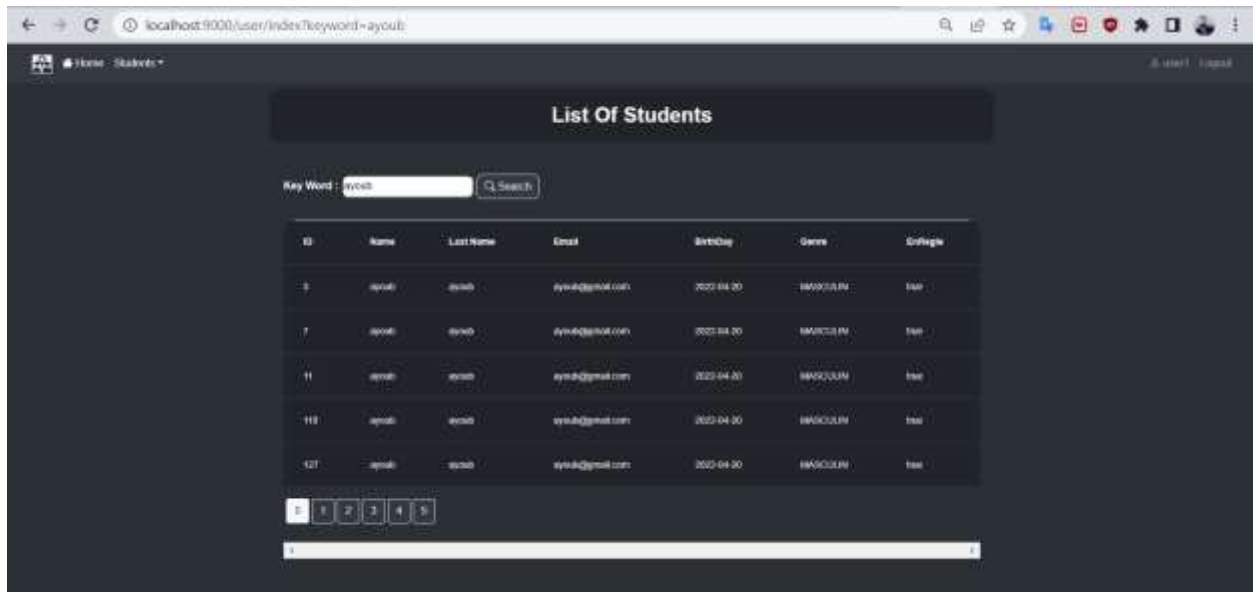


10 Authentification en tant qu'USER

- Consulter la liste des étudiants



- Rechercher un étudiant



- Si l'utilisateur USER1 test l'une des fonctionnalités de l'Administrateur

