

Activité pratique : Gestion des étudiants

Créé par : Mohamed Ait Lahcen

L'énoncé de l'activité :

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 UserDetails Service
- Ajouter d'autres fonctionnalités supplémentaires

1. Les dépendances :

pour construire cette application on a utilisé un ensemble des dépendances suivants :

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

```
>   <dependency>
>     <groupId>org.springframework.boot</groupId>
>     <artifactId>spring-boot-starter-security</artifactId>
>   </dependency>
>   <dependency>
>     <groupId>org.springframework.boot</groupId>
>     <artifactId>spring-boot-starter-test</artifactId>
>     <scope>test</scope>
>   </dependency>
>   <!-- https://mvnrepository.com/artifact/nz.net.ultraq.thymeleaf/thymeleaf-layout-
>   <dependency>
>     <groupId>nz.net.ultraq.thymeleaf</groupId>
>     <artifactId>thymeleaf-layout-dialect</artifactId>
>   </dependency>
```

```

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-validation</artifactId>
  <version>2.6.6</version>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <version>2.6.6</version>
</dependency>
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
</dependency>

```

```

<!-- https://mvnrepository.com/artifact/org.webjars/bootstrap -->
<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>bootstrap</artifactId>
  <version>5.1.3</version>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
</dependency>
</dependencies>

```

2. Le fichier configuration :

```
server.port=8086
spring.datasource.url = jdbc:mysql://localhost:3306/students-management
spring.datasource.username = root
spring.datasource.password =
spring.datasource.driverClassName = com.mysql.jdbc.Driver
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
spring.thymeleaf.cache=false
spring.main.allow-circular-references=true
```

3. L'entité Etudiant :

```
package com.example.students_management.entities;
import ...

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Etudiant {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank(message = "saisi le nom !")
    @Size(min = 3 ,max = 35 , message = "la longueur du nom doit etre entre 3 et 35")
    private String nom;

    @NotBlank(message = "saisi le prenom !")
    @Size(min = 3 ,max = 35 , message = "la longueur du prenom doit etre entre 3 et 35")
    private String prenom;
```

- **La table Etudiant dans la base de données :**

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/>	1	id	bigint(20)			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer ▼
<input type="checkbox"/>	2	date_naissance	datetime			Oui	NULL			Modifier Supprimer ▼
<input type="checkbox"/>	3	email	varchar(255) utf8mb4_general_ci			Oui	NULL			Modifier Supprimer ▼
<input type="checkbox"/>	4	genre	int(11)			Oui	NULL			Modifier Supprimer ▼
<input type="checkbox"/>	5	nom	varchar(255) utf8mb4_general_ci			Oui	NULL			Modifier Supprimer ▼
<input type="checkbox"/>	6	prenom	varchar(255) utf8mb4_general_ci			Oui	NULL			Modifier Supprimer ▼
<input type="checkbox"/>	7	regle	bit(1)			Non	Aucun(e)			Modifier Supprimer ▼

- **Le type énumération : Genre**

```
package com.example.students_management.entities;
```

± mohamed-ait

```
public enum Genre {
```

2 usages

MASCULIN,

2 usages

FEMININ

```
}
```

4. L'interface EtudiantRepository :

Cette interface contient des fonction prédéfinie qui permet d'interagir avec la base de donnée (ajout,suppression,modification...).

```
package com.example.students_management.repositories;

import ...

4 sages mohamed-ait
public interface EtudiantRepository extends JpaRepository<Etudiant,Long> {
    mohamed-ait
    Etudiant findByNom(String nom);
    1 usage mohamed-ait
    Page<Etudiant> findByNomContains(String kw, Pageable pageable);
}
```

5. EtudiantController :

cette classe représente le contrôleur de l'application qui s'occupe de répondre aux requêtes du client en lui transmettant les vues.

```
mohamed-ait *
@Controller
public class EtudiantController {
    4 usages
    private EtudiantRepository etudiantRepository;
    mohamed-ait
```

➤ Les fonctionnalités de l'application :

◆ La recherche des étudiants par nom :

Voici la fonction déclarée dans le contrôleur et qui permet de retourner les étudiants par mot clé en fait il prend le mot clé

```
@GetMapping("/index")
public String etudiants(Model model, @RequestParam(name = "page",defaultValue = "0") int page, @RequestParam(name = "size",def
    @RequestParam(name = "keyWord",defaultValue = "") String keyWord){
    Page<Etudiant> pageEtudiants = etudiantRepository.findByNomContains(keyWord, PageRequest.of(page,size));
    model.addAttribute( attributeName: "listEtudiants",pageEtudiants.getContent());
```

comme argument et il stocke le résultat dans le modèle:

2. La pagination :

voici la partie de code de la fonction index qui est occupée de gérer la pagination.

```
mohamed-ait *
@GetMapping("/index")
public String etudiants(Model model, @RequestParam(name = "page",defaultValue = "0") int page, @RequestParam(name = "size",def
    @RequestParam(name = "keyWord",defaultValue = "") String keyWord){
    Page<Etudiant> pageEtudiants = etudiantRepository.findByNomContains(keyWord, PageRequest.of(page,size));
    model.addAttribute( attributeName: "listEtudiants",pageEtudiants.getContent());
    int nbrPage;
    if(pageEtudiants.getTotalPages()>10)nbrPage=10;
    else nbrPage=pageEtudiants.getTotalPages();
    int pages[]=new int[nbrPage];
    if(page<7){
        for(int i=0;i<nbrPage;i++){
            pages[i]=i;
        }
    }
    else if(page>=7 && page<pageEtudiants.getTotalPages()-5){
        for(int j=0;j<nbrPage;j++){
            pages[j]=page-5+j;
        }
    }
}
```



```

else if(page>=pageEtudiants.getTotalPages()-5){
    for(int k=0;k<nbrPage;k++){
        pages[k]=pageEtudiants.getTotalPages()-10+k;
    }
}

model.addAttribute( attributeName: "pages", pages);
model.addAttribute( attributeName: "currentPage", page);
model.addAttribute( attributeName: "keyWord", keyWord);
model.addAttribute( attributeName: "totalPages", pageEtudiants.getTotalPages());
return "etudiants";
}

```

Dans l'interface graphique(vue) on affiche dix pages et deux bouton un pour passer à la page suivante et un autre bouton pour la page précédente.



3. La suppression des étudiants :

```

@DeleteMapping("/{delete/id}")
public String delete(@PathVariable("id") Long id,int page,String keyWord){
    etudiantRepository.deleteById(id);
    return "redirect:/index?page="+page+"&keyWord="+keyWord;
}

```

— mohamed-ait

4. L'ajout d'un étudiant :


Voici la méthode qui permet d'ajouter un nouveau étudiant :

```
//methode to save student
@ mohamed-ait
@PostMapping("/save")
public String save(Model model, @Valid Etudiant etudiant, BindingResult bindingResult, @RequestParam(name = "page", defaultValue = "1") Integer page) {
    if(bindingResult.hasErrors()) return "addFormEtudiant";
    etudiantRepository.save(etudiant);
    return "redirect:/index?page="+page+"&keyWord="+keyWord;
}
```

La page html qui contient le formulaire d'ajout :

```
<form method="post" th:action="@{/save}" >
    <div class="row my-4">
        <div class="col-md-6">
            <input type="text" class="form-control" id="nom" placeholder="Enter nom" name="nom" th:value="${etudiant.nom}">
            <span class="text-danger" th:errors="${etudiant.nom}"></span>
        </div>
        <div class="col-md-6">
            <input type="text" class="form-control" id="prenom" placeholder="Enter le prénom" name="prenom" th:value="${etudiant.prenom}">
            <span class="text-danger" th:errors="${etudiant.prenom}"></span>
        </div>
    </div>
    <div class="row my-4">
        <div class="col-md-6">
            <input type="text" class="form-control" id="email" placeholder="Enter email" name="email" th:value="${etudiant.email}">
            <span class="text-danger" th:errors="${etudiant.email}"></span>
        </div>
        <div class="col-md-6">
            <input type="date" class="form-control" id="dateNaissance" placeholder="Enter la date de naissance" name="dateNaissance">
        </div>
    </div>
</form>
```

```
<input type="date" class="form-control" id="dateNaissance" placeholder="Enter la date de naissance" />
<span class="text-danger" th:errors="${etudiant.dateNaissance}"></span>
</div>
</div>
<div class="row my-4">
  <div class="col-md-6">
    <select class="form-select" id="genre" name="genre" th:value="${etudiant.getGenre()}">
      <option value="MASCULIN"> Masculin</option>
      <option value="FEMININ"> Féminin</option>
    </select>
  </div>
  <div class="col-md-6">
    <label for="regle" class="form-label mt-2 me-2">Règle:</label>
    <input class="form-check-input" style="margin-top:12px" type="checkbox" id="regle" name="regle" th:checked="${etudiant.getRegle()}" />
    <span class="text-danger" th:errors="${etudiant.regle}"></span>
  </div>
</div>
<button type="submit" class="btn btn-success w-75 offset-2 mb-3"><i class="bi bi-person-plus-fill" style="font-size: 1.5em; margin-right: 5px;"></i> Sauvegarder</button>
</form>
```

 Ajouter un étudiant

Enter nom


Enter le prénom

Enter email

jj/mm/aaaa

Genre

Règle: ☐


 Sauvegarder

5. La modification d'un étudiant :

Voici la méthode qui permet de modifier un étudiant :

```
//edit function :  
@ mohamed-ait *  
@GetMapping("/editEtudiant")  
public String editEtudiant(Model model, Long id, int page, String keyWord){  
    Etudiant etudiant = etudiantRepository.findById(id).orElse(null);  
    model.addAttribute("etudiant", etudiant);  
    model.addAttribute("page", page);  
    model.addAttribute("keyWord", keyWord);  
    if(etudiant == null) throw new RuntimeException("etudiant introuvable!");  
    return "editFormPage";  
}
```

Modifier un étudiant

Règle: ☐ Modifier

6. La page template :

a) nameSpace de thymleaf et de la spring-security :

```
<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">
```

b) les librairies utilisés dans la page template (bootstrap,bootstrap-icons,fontAwsome,bootstrap-bundle) :

```
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.3.0/font/bootstrap-icons.css">
  <link rel="stylesheet" type="text/css" href="webjars/bootstrap/5.1.3/css/bootstrap.min.css"/>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@4.7.0/css/font-awesome.min.css">
  <script src="webjars/bootstrap/5.1.3/js/bootstrap.bundle.js"></script>
</head>
```

c) navBar :

```
<nav class="navbar navbar-expand-sm bg-secondary navbar-light">
  <div class="container-fluid">
    <ul class="navbar-nav">
      <li class="nav-item me-5">
        <a class="nav-link active text-white fw-bold" th:href="@{/}"><i class="fa fa-mortar-board fa-2x fw-bold"></i></a>
      </li>

      <li class="nav-item ml-5 me-3">
        <a class="nav-link text-white fw-bold fs-5" th:href="@{/index}"><i class="fa fa-home me-1"></i>Accueil</a>
      </li>

      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle text-white fw-bold fs-5" href="#" role="button" data-bs-toggle="dropdown"><i class="fa fa-bars me-1"></i></a>
        <ul class="dropdown-menu bg-secondary">
          <li><a th:href="@{/addEtudiant}" class="dropdown-item text-white fw-bold">nouveau<i class="fa fa-user-plus"></i></a>
          <li><a class="dropdown-item text-white fw-bold" th:href="@{/index}">chercher<i class="fa fa-magnifying-glass"></i></a>
        </ul>
      </li>
    </ul>
  </div>
</nav>
```



Accueil

Etudiants

admin

d) footer :

```
<!-- Footer -->
<footer
  class="text-center text-lg-start text-white bg-secondary mt-auto"
>
  <!-- Grid container -->
  <div class="container p-4 pb-0">
    <!-- Section: Links -->
    <section class="">
      <!-- Grid row -->
      <div class="row">
        <!-- Grid column -->
        <div class="col-md-3 col-lg-3 col-xl-3 mx-auto mt-3">
          <h6 class="text-uppercase mb-4 font-weight-bold">
            MOHAMED AIT LAHCEN
          </h6>
          <p>Elève ingénieur en deuxième année à l'enset mohammedia filière big data et cloud computing
```

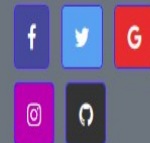
MOHAMED AIT LAHCEN

Elève ingénieur en deuxième
année à l'enset mohammedia
filiale big data et cloud computing

CONTACT

🏠 Kelaâ m'gouna tinghir
✉️ mohamedaitlahcen62@gmail.com
📞 0628020585

FOLLOW US



© 2020 Copyright: enset.ma

7. La sécurité de l'application :

Pour sécuriser l'application on a utilisé un système d'authentification qui est basé sur la stratégie userDetails service voici la structure suivi :



◆ Les entités de la système d'authentification :

1. AppUser :

```
@Entity
@Data @AllArgsConstructor @NoArgsConstructor
public class AppUser {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(unique = true, length = 20)
    private String userName;
    private String password;
    private boolean active;
    @ManyToMany(fetch = FetchType.EAGER)
    private List<AppRole> roles=new ArrayList<>();
}
```

2. AppRole :

```
@Entity
@Data @AllArgsConstructor @NoArgsConstructor
public class AppRole {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(unique = true, length = 20)
    private String roleName;
    private String description;
}
```

◆ La classe SecurityConfig :

Dans cette méthode on spécifie la stratégie de la sécurité à utiliser dans l'application :

```
private DataSource dataSource;

//dans cette methode on specifie comment spring chercher les roles des utilisateurs:
/* mohamed-ait */

@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    PasswordEncoder passwordEncoder=passwordEncoder();
    /*
```


- ◆ Dans cette méthode on gère les routes de l'application et les droit d'accès :

```
 mohamed-ait
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.formLogin().loginPage("/login").defaultSuccessUrl( defaultSuccessUrl: "/", alwaysUse: false);
    http.csrf().disable();
    http.authorizeHttpRequests().antMatchers( ...antPatterns: "/", "/login/**").permitAll();
    http.authorizeHttpRequests().antMatchers( ...antPatterns: "/index/**", "/addEtudiant/**", "/delete/**", "/save/**", "/editEtudiant/**");
    http.authorizeHttpRequests().antMatchers( ...antPatterns: "/index/**").hasAuthority("USER");
    http.authorizeHttpRequests().antMatchers( ...antPatterns: "/webjars/**").permitAll();
    http.exceptionHandling().accessDeniedPage( accessDeniedUrl: "/403");
    http.authorizeHttpRequests().anyRequest().authenticated();
}
```

8. Personnalisé l'authentification :

1. La méthode login :

```
 mohamed-ait
@GetMapping
@RequestMapping("/login")
public String login() {
    return "login";
}
```

2. La page login :

```
<div layout:fragment="content1">
<div class="container-fluid text-center p-5">
  <div class="card w-50 offset-3">
    <div class="card-header bg-primary text-white">
      <h3>Connexion</h3>
    </div>
    <div class="card-body p-3 bg-light">
      <form th:action="@{/login}" method="post" >

        <div th:if="${param.error}">
          <p class="text-danger">[[${session.SPRING_SECURITY_LAST_EXCEPTION.message}]]</p>
        </div>

        <div th:if="${param.logout}">
          <p class="text-warning">You have been logged out.</p>
        </div>

        <div class="form-group my-3">
```

```
        <div th:if="${param.logout}">
          <p class="text-warning">You have been logged out.</p>
        </div>

        <div class="form-group my-3">
          <input type="username" name="username" class="form-control" placeholder="username" required autofocus/>
        </div>
        <div class="form-group my-3">
          <input type="password" name="password" class="form-control" placeholder="Password" required />
        </div>
        <div class="form-group my-3">
          <input type="checkbox" name="remember-me" /> Remember Me
        </div>
        <div class="form-group mt-3">
          <input type="submit" value="Se connecter" class="btn btn-primary w-50 fw-bold" />
        </div>
      </form>
    </div>
  </div>
</div>
```

Connexion

☐ Remember Me

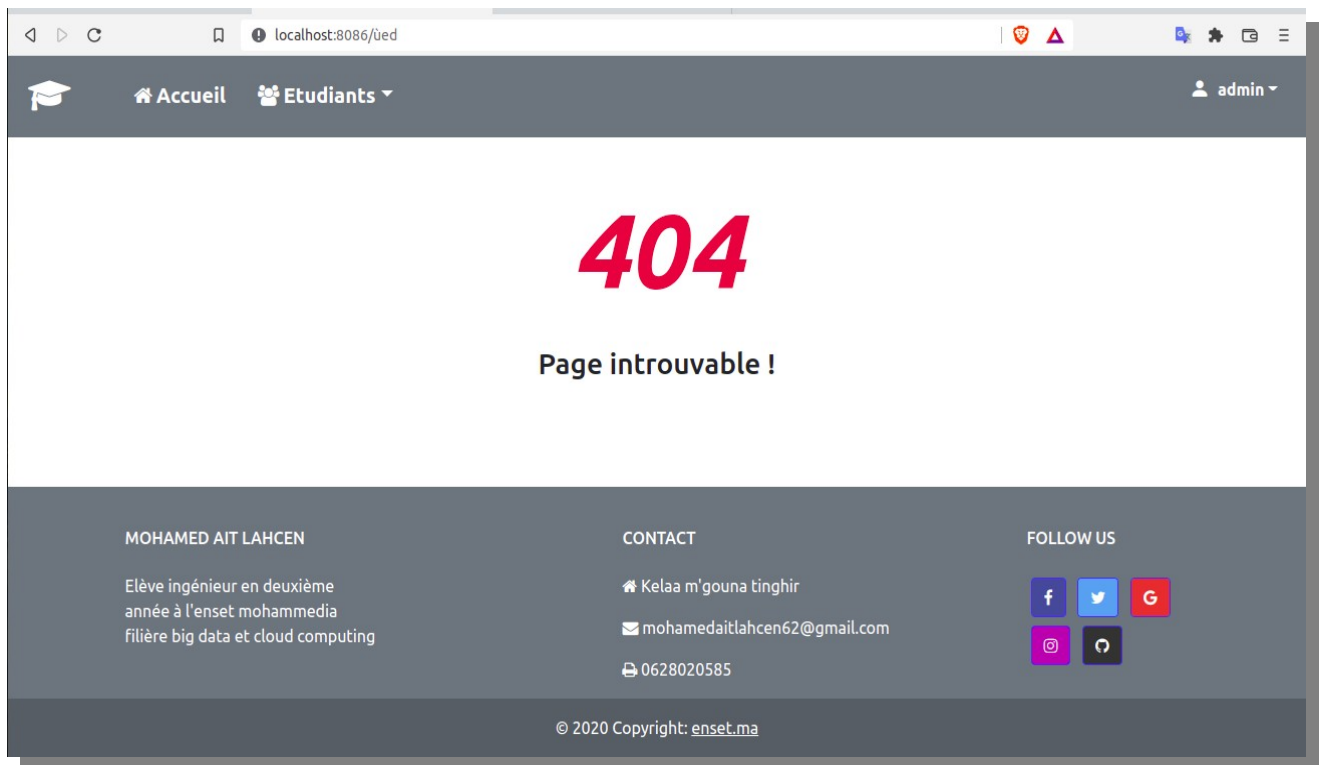
Se connecter

9. La page error 404 :

Cette page apparaît quand une ressource demandée est indisponible :

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="template"
      xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
  <meta charset="utf-8"/>
  <title>patient-mvc</title>

</head>
<body class="d-flex flex-column min-vh-100">
<div layout:fragment="content1">
  <div class="container p-5">
    <h1 class="text-center fw-bold fst-italic text-danger " style="font-size:100px">404</h1>
    <h3 class="text-center ml-2">Page introuvable !</h3>
  </div></div>
</body>
</html>
```



10. La page error 403 :

Cette page apparaît quand une ressource demandée n'est pas autorisée :

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="template"
      xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
  <meta charset="utf-8"/>
  <title>patient-mvc</title>
</head>
<body class="d-flex flex-column min-vh-100">
<div layout:fragment="content1">
<div class="container p-5">
  <h1 class="text-center fw-bold fst-italic text-danger " style="...">403</h1><br>
  <h3 class="text-center ml-2">Vous n'êtes pas autorisés !</h3>
</div></div>
</body>
</html>
```

[Accueil](#)[Etudiants](#) ▾[user](#) ▾

403

Vous n'êtes pas autorisés !

MOHAMED AIT LAHCEN

Elève ingénieur en deuxième
année à l'enset mohammedia
filière big data et cloud computing

CONTACT

🏠 Kelaâ m'gouna tinghir

✉ mohamedaitlahcen62@gmail.com

📞 0628020585

FOLLOW US



© 2020 Copyright: enset.ma


Les pages de l'application :

1) La page d'accueil :

Cette page est accessible à tous :



2) La page d'authentification :

[Accueil](#) [Etudiants](#) [connexion](#)

Connexion

☐ Remember Me

Se connecter

MOHAMED AIT LAHCEN

Elève ingénieur en deuxième
année à l'enset mohammedia
filière big data et cloud computing






CONTACT

🏠 Kelaa m'gouna tinghir

✉ mohamedaitlahcen62@gmail.com


☎ 0628020585

FOLLOW US














© 2020 Copyright: [enset.ma](#)

3) La page de la recherche :

 [Accueil](#) [Etudiants](#) admin

La liste des étudiants



ID	Nom	Prenom	Email	Date de Naissance	Genre	Regle	Delete	Edit
3	nom3	prenom3	prenom3nom3@gmail.com	2022-04-18	MASCULIN	×		
4	nom4	prenom4	prenom4nom4@gmail.com	2022-04-18	FEMININ	✓		
5	nom1	prenom1	prenom1nom1@gmail.com	2022-04-18	MASCULIN	✓		
6	nom2	prenom2	prenom2nom2@gmail.com	2022-04-18	FEMININ	✓		
7	nom3	prenom3	prenom3nom3@gmail.com	2022-04-18	MASCULIN	×		

0

1

2

3

4

5

6

7

8




9

▶






MOHAMED AIT LAHCEN

Elève ingénieur en deuxième
année à l'enset mohammedia
filère big data et cloud computing

CONTACT



 Kelaa m'gouna tinghir
 mohamedaitlahcen62@gmail.com
 0628020585


FOLLOW US


  
 

© 2020 Copyright: [enset.ma](#)

4) La page d'ajout :

[Accueil](#) [Etudiants](#) 


admin 


Ajouter un étudiant

Enter nom


Enter le prénom

Enter email

jj/mm/aaaa 

Masculin 


Règle: ☐


Sauvegarder


MOHAMED AIT LAHCEN

Elève ingénieur en deuxième
année à l'enset mohammedia
filière big data et cloud computing






CONTACT

 Kelaa m'gouna tinghir

 mohamedaitlahcen62@gmail.com

 0628020585

FOLLOW US



© 2020 Copyright: [enset.ma](#)

5) La page de modification :



Modifier un étudiant

Règle: ☐

Modifier

MOHAMED AIT LAHCEN

Elève ingénieur en deuxième
année à l'enset mohammedia
filière big data et cloud computing

CONTACT

🏠 Kelaa m'gouna tinghir
✉ mohamedaitlahcen62@gmail.com
☎ 0628020585

FOLLOW US

