

Rapport Projet JEE

Application pour la gestion des notes

Réalisé Par:

MALHOUNI YOUNESS

Encadré Par :

Pr.GHERABI Noreddine

Sommaire

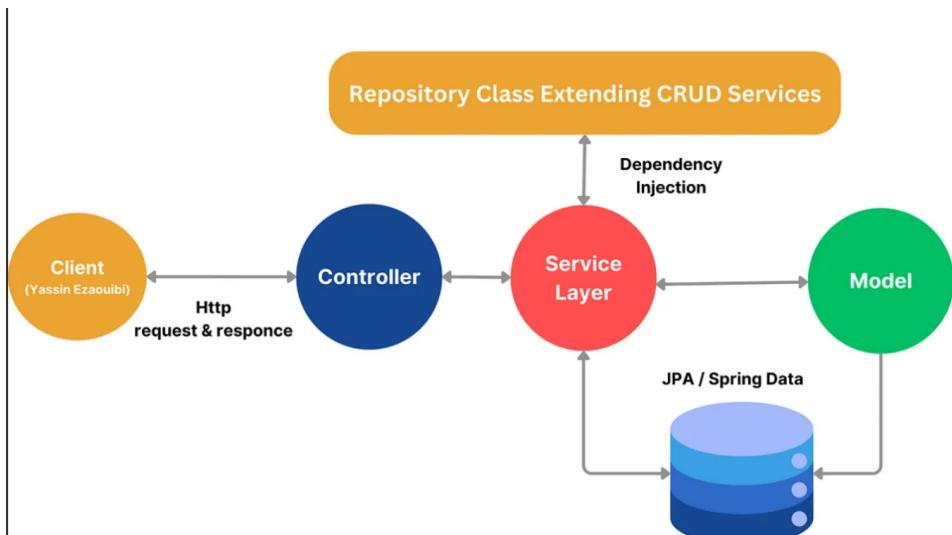
- ▼ Introduction
- ▼ Technologie utilisees
- ▼ Conception et Base de donnees
 - ▼ Modelisation UML
 - ▼ Diagramme de Classes
 - ▼ Base De Donnees
- ▼ Traitements de l'Application
 - ▼ Traitements De Connexions
 - ▼ Traitements Administrateurs
 - ▼ Traitements Professeurs
- ▼ Conclusion

Introduction

Dans le contexte des systèmes d'information académiques, la gestion des notes et des évaluations représente un enjeu crucial nécessitant des solutions logicielles robustes et performantes. Ce projet a pour objectif le développement d'une application web de gestion des notes, intégrant un ensemble de fonctionnalités permettant l'administration des composantes pédagogiques et le suivi des évaluations.

L'architecture technique de notre solution repose sur **Spring Boot**, implémentant une architecture en couches qui assure une organisation optimale du code et une séparation claire des responsabilités. Cette architecture se compose de plusieurs couches distinctes, chacune ayant un rôle spécifique :

- La couche **présentation (View)**, développée avec **JSP** (JavaServer Pages), met en œuvre une interface utilisateur intuitive et responsive grâce à l'intégration de **Bootstrap**. L'utilisation de **JavaScript** permet d'enrichir l'expérience utilisateur en offrant des interactions dynamiques et des validations côté client, améliorant ainsi la réactivité de l'application.
- La couche **Controller** agit comme point d'entrée de l'application, interceptant et gérant les requêtes HTTP. Elle assure la coordination entre la couche présentation et les services métier, tout en gérant la validation des données et la gestion des sessions utilisateurs. Les contrôleurs sont annotés avec `@Controller`.
- La couche **Service** encapsule la logique métier complexe de l'application. Elle implémente les règles de gestion spécifiques, comme le calcul des moyennes, la validation des notes, et la gestion des droits d'accès. Cette couche joue un rôle crucial dans la maintenance de l'intégrité des données et l'application des règles métier.
- La couche **Repository**, basée sur Spring Data JPA, gère l'accès aux données et la persistance. Elle fournit une abstraction de la base de données en utilisant les interfaces JPA Repository, simplifiant ainsi les opérations CRUD (Create, Read, Update, Delete) tout en permettant l'écriture de requêtes personnalisées lorsque nécessaire.
- La couche **Model** définit les entités du domaine, représentant la structure des données de l'application. Ces entités sont enrichies d'annotations JPA pour la persistance et de validations Bean Validation pour garantir l'intégrité des données.



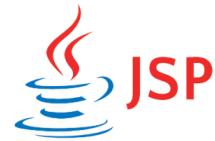
Le système implémente une gestion fine des autorisations basée sur deux profils distincts : **administrateur** et **professeur**. Cette ségrégation des rôles, implémentée via Spring Security, permet un contrôle granulaire des accès aux différentes fonctionnalités. L'administrateur peut gérer l'ensemble des aspects configuration du système, tandis que les professeurs ont accès à des fonctionnalités spécifiques liées à la gestion des notes.

La gestion des données est optimisée grâce à l'utilisation de Hibernate comme implémentation JPA, permettant une manipulation efficace des relations entre les différentes entités : professeurs, filières, modules, éléments de modules, et notes. Les transactions sont gérées de manière déclarative via les annotations Spring, assurant l'intégrité des données lors des opérations critiques.

L'interface utilisateur a été conçue en mettant l'accent sur l'ergonomie et la facilité d'utilisation. L'utilisation de Bootstrap garantit une expérience cohérente et responsive , tandis que l'intégration de JavaScript permet d'implémenter des fonctionnalités avancées comme la validation en temps réel des formulaires et l'actualisation dynamique des données.

Ce rapport détaille la démarche technique adoptée, depuis la phase de conception jusqu'à l'implémentation, en mettant l'accent sur les choix architecturaux et les solutions techniques mises en œuvre. Nous aborderons également les défis rencontrés et les solutions apportées pour garantir la performance, la sécurité et la maintenabilité de l'application.

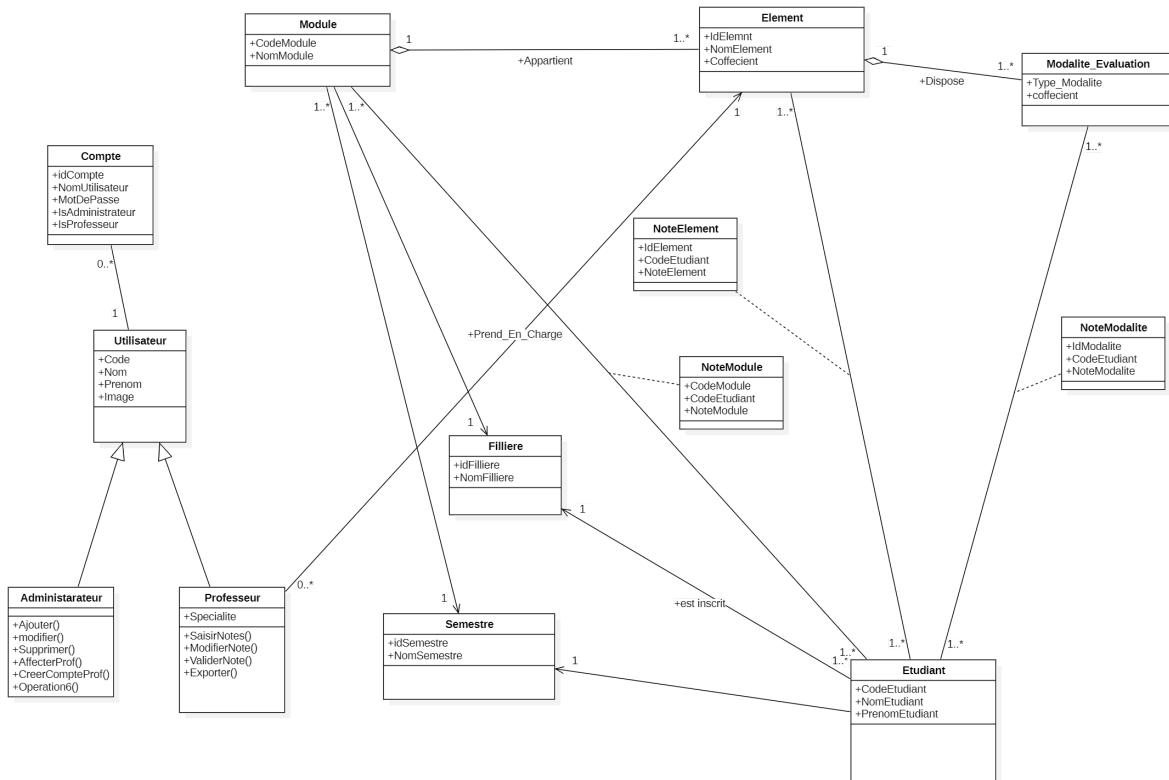
Technologies Utilisees



Conception et Base de Donnees

Modelisation UML

Diagramme de classes



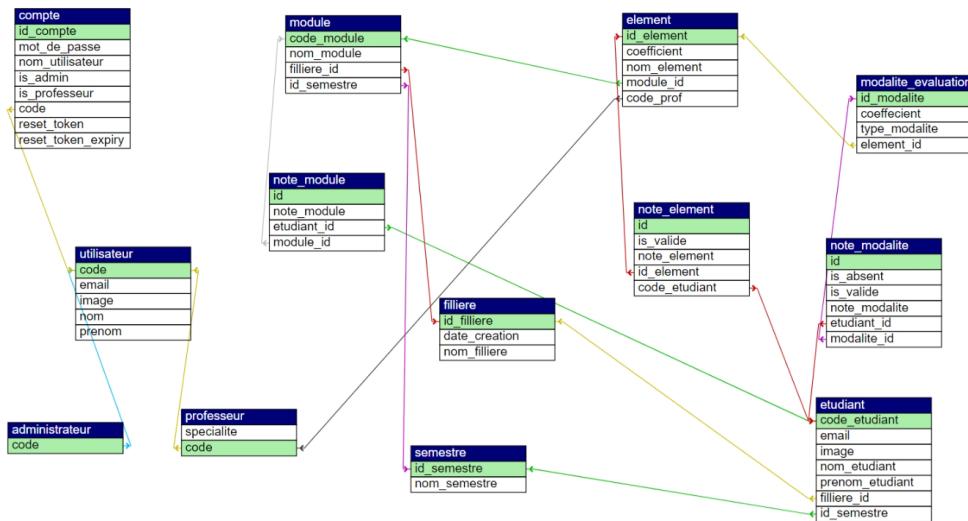
Base De Donnees

Comme il est decrits dans le diagramme de classe deja presente , notre base de donnees contient les 10 Tables suivantes : **Utilisateur** , **Administrateur** , **Professeur** , **Compte**

, Module , Element, Modalite_Evaluation , Filiere , Semestre et Etudiant

Avec les associations suivantes : NoteModalite , NoteElement

Le schcema Relationnel du base de donnees se presente comme ceci :

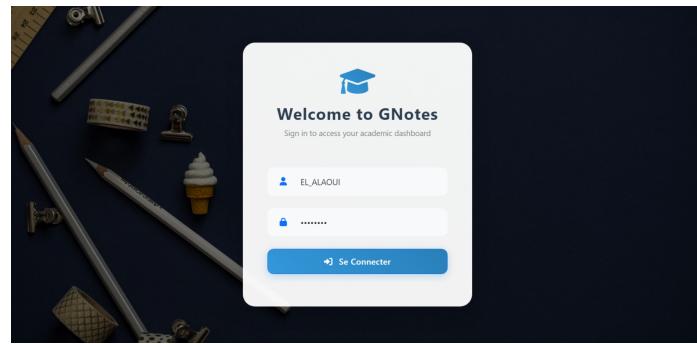


Traitements de l'Application

Traitements De Connexions

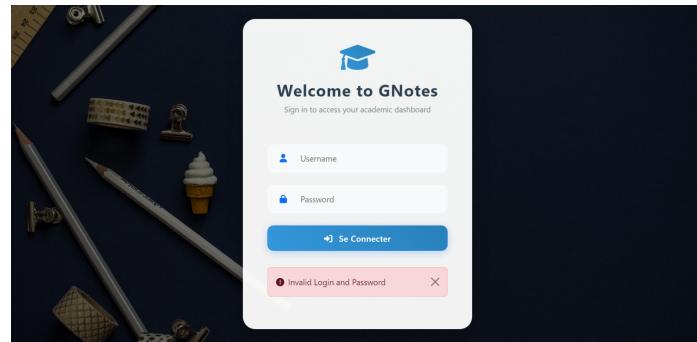
Le premier traitement de l'application est la connexion de l'utilisateur. L'application assure la connexion sécurisée des administrateurs et des professeurs, tout en respectant les autorisations et les droits d'accès pour chaque rôle.

L'application détecte le rôle de l'utilisateur à partir de ses informations de connexion (nom d'utilisateur, mot de passe), puis affiche les tâches autorisées selon ce rôle.



L'application cherche dans la base de données sur le compte avec les informations citées et détecte son rôle après l'attribut IsProfesseur, IsAdmin puis renvoie le Dashboard.

- Cas de fausses informations



- Cas D'un Administrateur

- Cas D'un Professeur

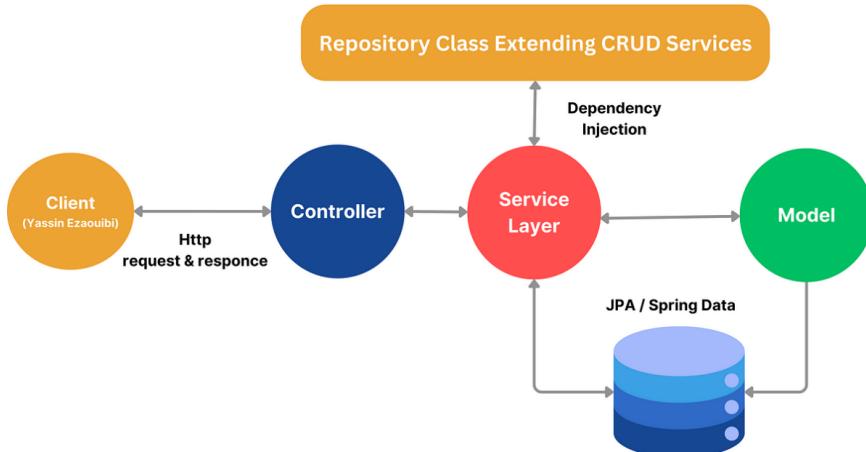
Bienvenue, Mohammadi

Voici un aperçu de votre espace professeur

24 Étudiants Actifs	3 Modules Enseignés	6 Éléments de Module	85% Notes Saisies
-------------------------------	-------------------------------	--------------------------------	-----------------------------

- Mes Étudiants**
Gérer la liste de vos étudiants par classe et filière
[Accéder →](#)
- Mes Éléments**
Consulter et gérer vos modules et éléments de module
[Accéder →](#)
- Gestion des Notes**
Saisir et modifier les notes des étudiants
[Accéder →](#)

On assure cette procédure du Login en suivant l'archecture de **Spring Boot**



le controller est le suivant :

```

public class UtilisateurController {
    @GetMapping("/loginPage")  ~ younessmalhouni <younes.malhouni03@gmail.com>
    public String loginPage() { return "login"; }
    @PostMapping("/loginUser") ~ younessmalhouni <younes.malhouni03@gmail.com> *
    public String loginUser(@ModelAttribute("form") Compte compte, Model model, HttpSession session) {
        Compte existingUser = compteService.login(compte.getNomUtilisateur(), compte.getMotDePasse());
        if (existingUser == null) {
            model.addAttribute( attributeName: "msg", attributeValue: "Invalid Login and Password");
            return "login";
        } else {
            session.setAttribute( s: "loggedInUser", existingUser);
            model.addAttribute( attributeName: "nomUtilisateur", existingUser.getNomUtilisateur());
            Integer totalEtudiants = administrateurService.NbreEtudiants();
            Integer totalProfesseurs = administrateurService.NbreProfesseurs();
            Integer totalFillieres = administrateurService.NbreFillieres();
            Integer totalModules = administrateurService.NbreModules();
            model.addAttribute( attributeName: "totalEtudiants", totalEtudiants);
            model.addAttribute( attributeName: "totalProfesseurs", totalProfesseurs);
            model.addAttribute( attributeName: "totalFillieres", totalFillieres);
            model.addAttribute( attributeName: "totalModules", totalModules);
            if (Boolean.TRUE.equals(existingUser.getIsAdmin())) {
                return "adminTemplate"; // Admin-specific JSP
            } else if (Boolean.TRUE.equals(existingUser.getIsProfesseur())) {
                return "professorTemplate"; // Professor-specific JSP
            } else {
                model.addAttribute( attributeName: "msg", attributeValue: "User role not recognized.");
                return "login";
            }
        }
    }
}

```

Le contrôleur Utilise plusieurs Service qui implémente les méthodes nécessaires pour le traitement

par exemple : ici on récupère le compte à partir du nom d'utilisateur et le mot de passe à l'aide de service `compteService` qui implemente la méthode
`login(compte.getNomUtilisateur(), compte.getMotDePasse());`

```
Compte existingUser = compteService.login(compte.getNomUtilisateur(),
(), compte.getMotDePasse());
```

`CompteService.java`

```

package com.example.GNotesAPP12.Service;
import com.example.GNotesAPP12.Model.Compte;
import com.example.GNotesAPP12.Model.Utilisateur;
import com.example.GNotesAPP12.Repo.CompteRepo;
import com.example.GNotesAPP12.Repo.UtilisateurRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.SimpleMailMessage;
```

```

import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.stereotype.Service;
import java.time.LocalDateTime;
import java.util.UUID;
@Service
public class CompteService {
    @Autowired
    private CompteRepo compteRepo;

    public Compte login(String nomUtilisateur, String motDePasse)
    {
        return compteRepo.findByNomUtilisateurAndMotDePasse(nomUtilisateur, motDePasse);
    }

    public void saveCompte(Compte compte) {
        compteRepo.save(compte);
    }
}

```

CompteRepo.java

```

package com.example.GNotesAPP12.Repo;

import com.example.GNotesAPP12.Model.Compte;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

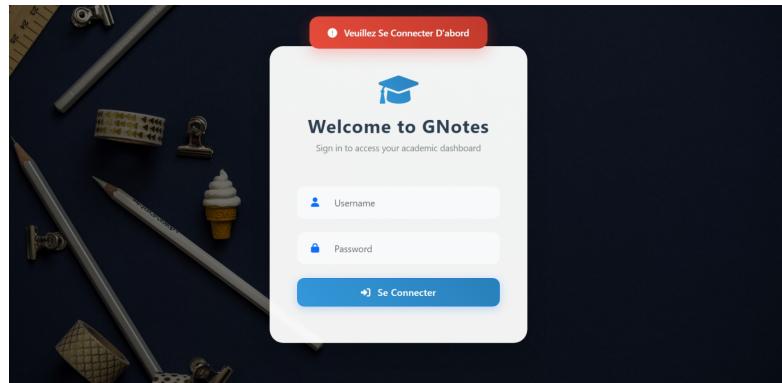
import java.util.Optional;

public interface CompteRepo extends JpaRepository<Compte, Long> {
    Compte findByNomUtilisateurAndMotDePasse(String nomUtilisateur, String motDePasse);

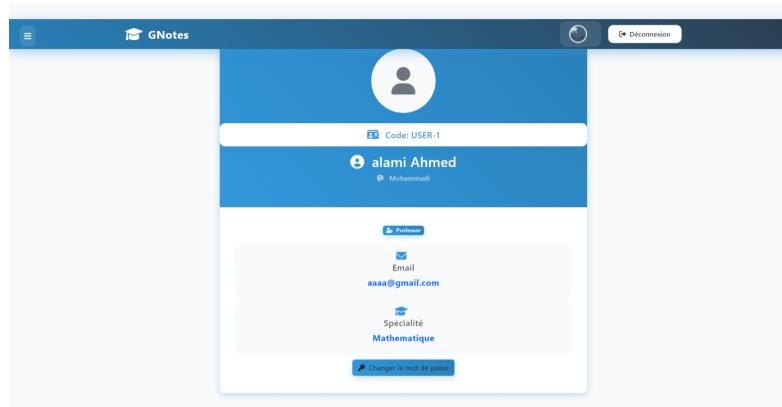
    @Query("SELECT c FROM Compte c " +
            "JOIN c.utilisateur u " +
            "WHERE u.email = :email"
    )
    Optional<Compte> findByEmail(String email);
    Optional<Compte> findByResetToken(String resetToken);
}

```

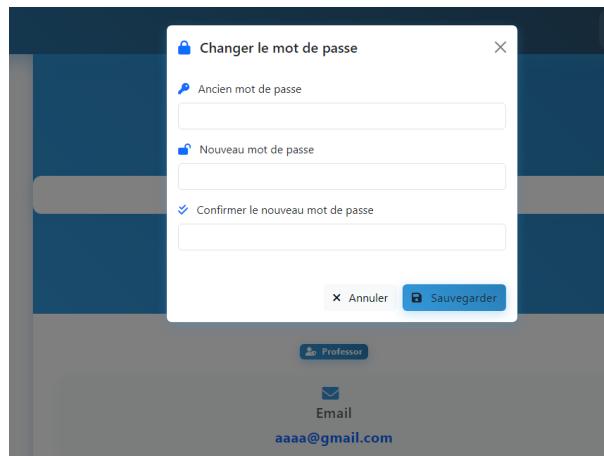
- Si l 'Utilisateur Essaye d'accéder une des tâches Sans se Connecter une Alerta est Déclarée



On peut aussi accéder au profil d 'utilisateur Connecté, en cliquant l icone de profil en tout haut droite :



l application permet au utilisateurs de modifier leurs mot de passe :



Traitements Administrateur

L'administrateur joue un rôle central dans la gestion et l'organisation de l'application. Il est responsable de la configuration globale du système éducatif en assurant la gestion complète des différentes entités. Ses responsabilités comprennent la gestion des comptes professeurs, la création et la maintenance des filières d'études, la configuration des modules et leurs éléments constitutifs, ainsi que la définition des modalités d'évaluation. L'administrateur est également chargé d'une tâche cruciale : l'affectation des éléments d'enseignement aux professeurs correspondants. Cette position lui confère un rôle de superviseur qui garantit le bon fonctionnement et l'organisation structurée de l'ensemble du système de gestion des notes.

Pour Assurer le deroulement de ces procedures On implemente les Traitements Suivants :

1. Gestion Des Professeurs

l'administrateur gère les opérations de l'ajout, la lecture, la modification, et la suppression d'un professeur, aussi son affectation à un ou plusieurs éléments d'un ou plusieurs modules

The screenshot shows a user interface for managing teachers. At the top, there is a header bar with the logo 'GNotes', a search bar, a dropdown menu for 'Tous Les Spécialités', and a 'Déconnexion' button. Below the header, there is a large 'Ajouter un Professeur' button. The main area displays a grid of six teacher profiles, each with a thumbnail, name, subject, code, status, and three action buttons: 'Modifier', 'Supprimer', and 'Créer Compte'. The profiles are as follows:

Code	Nom	Spécialité	Statut	Action
12	EL MALKI YOUSSEF	Mathématiques	Actif	Modifier, Supprimer, Créer Compte
13	BOUZIANE AMINA	Physique	Actif	Modifier, Supprimer, Créer Compte
14	CHAFIK SALMA	Chimie	Inactif	Modifier, Supprimer, Créer Compte
15	EL JADIDI NOUREDDINE	Biologie	Actif	Modifier, Supprimer, Créer Compte
16	ABBOUDI HIND	Informatique	Actif	Modifier, Supprimer, Créer Compte
17	OUAZZANI REDA	Histoire	Actif	Modifier, Supprimer, Créer Compte

Ajouter Professeur

Add Professor

Nom

Prénom

Spécialité

Image
 No file chosen

Pour Afficher le formulaire d ajout:

```
@GetMapping("/add")
public String showAddForm(Model model) {
    model.addAttribute("professeur", new Professeur());
    return "professeur-form";
}
```

Pour Enregistrer :

```
public class AdministrateurController {
    @PostMapping("/save")
    public String saveProfesseur(@ModelAttribute("professeur") Professeur professeur,
                                  @RequestParam(value = "imageFile", required = false) MultipartFile imageFile)
        throws IOException {

        if (professeur.getCode() != null) {
            // Get existing professor
            Professeur existingProfesseur = professeurService.getProfesseurById(professeur.getCode().toString());
            if (existingProfesseur != null) {
                // Preserve existing image if no new image is uploaded
                if (imageFile == null || imageFile.isEmpty()) {
                    professeur.setImage(existingProfesseur.getImage());
                } else {
                    processAndSetImage(professeur, imageFile);
                }
            }
        } else if (imageFile != null && !imageFile.isEmpty()) {
            processAndSetImage(professeur, imageFile);
        }
        professeurService.saveProfesseur(professeur);
        return "redirect:/professeurs";
    }
}
```

On suit la même logique pour la modification.

On peut créer ou ajouter un compte pour un professeur pour lui affecter un rôle,

On peut Affecter un Role Administrateur pour un professeur en lui ajoutant un compte admin.

Le status de compte devient **actif**

2. Gestion Des Modules

La procédure de création ou de modification d'un module suit les étapes et règles suivantes :

L'administrateur ajoute ou modifie un module en cliquant sur le bouton "Ajouter Module" ou "Modifier Module". Lors de la création, il peut ajouter les éléments (et leurs modalités) avec leurs coefficients respectifs. Les coefficients doivent être compris entre 1 et 100. La somme des coefficients, que ce soit pour les modalités d'un élément ou pour les éléments d'un module, ne doit pas dépasser 100. Si ces conditions ne sont pas respectées, l'enregistrement du module est bloqué et des alertes s'affichent pour indiquer les erreurs à corriger.

Ajouter Un Module

Module Information

Module Code *	Module Name *
Filière *	Semestre *
Select a Filière	Select a Semestre

Elements

Element ID *	Element Name *	Coefficient *
		1
Professeur *	Sélectionner un Professeur	
Evaluation Modalities		
Type *	Coefficient *	
	1	
Add Modality		
Remove Element		

Cancel **Save Module**

Chaque fois qu'on clique sur "Ajouter Élément", un nouveau formulaire s'affiche pour renseigner les informations et les modalités de cet élément. Il est également possible de supprimer une modalité ou un élément à tout moment.

Chaque module doit être assigné à une filière et un semestre, et chaque élément doit être affecté à un professeur.

Après avoir renseigné toutes les informations, l'application vérifie si ces données respectent les règles suivantes :

▼ les coefficients doivent être entre 0 et 100

Element ID *	Element Name *	Coefficient *
	JAVA	120
Professeur *	NOUREDDINE	
Evaluation Modalities		

Le Coefficient doit être entre 1 et 100 et la somme ne doit pas dépasser 100.

▼ la somme des coefficients des éléments doit être inférieur à 100

▼ la somme des coefficients des modalités d'évaluation doit être inférieur à 100

Element ID *	Element Name *	Coefficient *
	JAVA	10 ✓
Professeur *	NOUREDDINE ✓	
Evaluation Modalities		
Type *	Coefficient *	
Exam	70 ⓘ	Le Coefficient doit être entre 1 et 100 et la somme ne doit pas dépasser 100.
Remove Modality		
Type *	Coefficient *	
tp	40 ⓘ	Le Coefficient doit être entre 1 et 100 et la somme ne doit pas dépasser 100.
Remove Modality		

L'ajout du module ne se fait correctement que si toutes ces règles sont respectées.

Les informations saisies dans les formulaires de l'application sont traitées à travers une architecture en couches respectant le paradigme MVC. Le contrôleur capture les données via des requêtes HTTP et les transmet à la couche service

`ModuleController`

```
public class ModuleController {
    @GetMapping("/add")
    public String showAddForm(Model model) {
        Module module = new Module();
        module.setElements(new ArrayList<>());
        model.addAttribute("module", module);
        model.addAttribute("fillieres", filliereService.getAllFillieres());
        model.addAttribute("semestres", semestreService.getAllSemestres());
        model.addAttribute("professeurs", professeurService.getAllProfesseurs());
        return "module-form";
    }
}
```

`ShowAddForm` permet de retourner le formulaire d'ajout au dessus .

```

public class ModuleController {
    @PostMapping("/save")
    public String saveModule(@ModelAttribute Module module) {
        if (module.getElements() == null) {
            module.setElements(new ArrayList<>());
        }
        for (Element element : module.getElements()) {
            element.setModule(module);
            if (element.getProfesseur() != null) {
                element.setProfesseur(professeurService.getProfesseurById(element.getProfesseur().getCode().toString()));
            }
            if (element.getModalites() == null) {
                element.setModalites(new ArrayList<>());
            }
            if (element.getNoteElements() == null) {
                element.setNoteElements(new ArrayList<>());
            }

            for (Modalite_Evaluation modalite : element.getModalites()) {
                modalite.setElement(element);
            }
        }
        moduleService.saveModule(module);
        return "redirect:/modules";
    }
}

```

`saveModule` permet, comme son nom l'indique, d'enregistrer le nouveau module saisi dans le formulaire

Le service délègue ensuite les opérations de persistance à la couche repository, qui interagit directement avec la base de données via Spring Data JPA

`ModuleService`

```

14     @Service 4 usages  ✘ younessmalhouni <younes.malhouni03@gmail.com> *
15     public class ModuleService {
16         @Autowired
17         private ModuleRepo moduleRepo;
18         @Autowired
19         private NoteElementRepo noteElementRepo;
20
21         public List<Module> getAllModules() { return moduleRepo.findAll(); }
24
25         public Module getModuleById(String id) { return moduleRepo.findById(id).orElse( other
28
29         public void saveModule(Module module) { moduleRepo.save(module); }
32
33         public void deleteModule(String id) { moduleRepo.deleteById(id); }
36         public Double SommeCoefModule(Long CodeModule) { return moduleRepo.SommeCoefModule(0
39
40         public Double SommeCoefModuleExceptElement(Long CodeModule, Element element) { no us
41             return moduleRepo.SommeCoefModule(CodeModule) - element.getCoefficient();
42         }
43         public Double CalculerNoteModule(Long CodeModule, Long CodeEtudiant) { 1 usage new *
44             Module module = getModuleById(CodeModule.toString());
45             List<Element> elements = module.getElements();
46             Double somme = 0.0;

```

`ModuleRepo`

```

1 package com.example.GNotesAPP12.Repo;
2 > import ...;
3
4 public interface ModuleRepo extends JpaRepository<Module, String> {
5
6     @Query("select sum(el.coefficient) from Element el where el.module.codeModule = :moduleCode")
7     Double SommeCoefModule(Long moduleCode);
8
9
10    @Query("SELECT DISTINCT e FROM Etudiant e " + 1 usage new *
11        "JOIN e.filliere f " +
12        "JOIN f.modules m " +
13        "WHERE m.codeModule = :codeModule AND " +
14        "(:search IS NULL OR LOWER(e.nomEtudiant) LIKE LOWER(CONCAT('%', :search, '%')) " +
15        "OR LOWER(e.prenomEtudiant) LIKE LOWER(CONCAT('%', :search, '%')))) AND " +
16        "(:filliereId IS NULL OR e.filliere.idFilliere = :filliereId) AND " +
17        "(:semestreId IS NULL OR e.semestre.id_Semestre = :semestreId)");
18    List<Etudiant> EtudiantsOfModule(Long codeModule,
19                                         @Param("search") String search,
20                                         @Param("filliereId") Long filliereId,
21                                         @Param("semestreId") Long semestreId);
22
23
24
25
26
27
28
29
30

```

Modifier un Module

Modifier Module

Module Information

Module Code *	Module Name *
	Programmation Avancée
Filliere *	Semestre *
IID	S1

Elements

Element ID *	Element Name *	Coefficient *
	Structures de Données Avancées	50.0

Professeur *

YOUSSEF

Evaluation Modalities

Type *	Coefficient *
Examen	60.0
<button>Remove Modality</button>	
Type *	Coefficient *
TP	40.0
<button>Remove Modality</button>	
<button>Add Modality</button>	
<button>Remove Element</button>	

On trouve les information déjà renseignée du Module ,on peut les modifier , ajouter d autre éléments (respectivement Modalités d'évaluation), changer leurs coefficients mais tout en respectant les règles déjà introduites précédemment

La modification inclut tous les opérations concernant l'affectation des éléments à des nouveaux professeurs , ou l'assignement des Module à d'autre filière dans d'autre semestre

```

@GetMapping("edit/{id}")
public String showEditForm(@PathVariable("id") String id, Model model) {
    Module module = moduleService.getModuleById(id);
    if (module.getElements() == null) {
        module.setElements(new ArrayList<>());
    }
    Double SommeCoefModule = moduleService.SommeCoefModule(module.getCodeModule());
    model.addAttribute( attributeName: "SommeCoefModule", SommeCoefModule);
    model.addAttribute( attributeName: "module", module);
    model.addAttribute( attributeName: "fillieres", filliereService.getAllFillieres());
    model.addAttribute( attributeName: "semestres", semestreService.getAllSemestres());
    model.addAttribute( attributeName: "professeurs", professeurService.getAllProfesseurs());
    return "module-form";
}

```

L'enregistrement des modifications se fait par la méthode `SaveModule` déjà expliquée précédemment.

Supprimer Un Module

```

@GetMapping("/delete/{id}")
public String deleteModule(@PathVariable("id") String id) {
    moduleService.deleteModule(id);
    return "redirect:/modules";
}

```

Consultation Des Notes

En plus de cela , un administrateur , a le droit a de consulter les notes des etudiants dans Module, mais les Notes ne sont pas affiches que si les professeurs responsables des elements de ce module ont valides les notes pour un Etudiant , Les Notes sont affichees par la methode suivante :

The screenshot shows a web application interface titled 'GNotes'. At the top, there is a navigation bar with icons for profile, logout, and a search bar. Below the navigation bar, the title 'Notes du Module' is displayed, followed by 'Programmation Avancée (1)'. On the right side of the header, there are buttons for 'Exporter PDF' and 'Retour aux Modules'. The main content area features a table with student data:

Code Étudiant	Nom Complet	Filière	Éléments	Note Finale
1	ALAOUI Ahmed	IID	Structures de Données Avancées: 15.4 Conception d'Algorithmes: 15.5	15.45
2	BENJELLOUN Sara	IID	Structures de Données Avancées: 14.6 Conception d'Algorithmes: 18.5	16.55
3	CHAQUI Karim	IID	Structures de Données Avancées: 8.6 Conception d'Algorithmes: 7.5	8.05
5	EL OMARI Youssef	IID	Structures de Données Avancées: 10.4 Conception d'Algorithmes: 15.0	12.7
6	FASSI Leila	IID	Structures de Données Avancées: N/A Conception d'Algorithmes: N/A	N/A
7	GHAZALI Mehdi	IID	Structures de Données Avancées: N/A Conception d'Algorithmes: N/A	N/A

Dans cet exemple, les professeurs des éléments **Structures de Données Avancées et Conception d'Algorithmes** ont saisi et validé les notes pour les quatre premiers étudiants. Ces notes sont donc affichées. Pour les autres étudiants, les professeurs n'ont pas encore validé les notes.

Une note en vert indique que l'étudiant a validé ce module (≥ 12), sinon elle apparaît en rouge.

Les notes sont calculées automatiquement après la validation de tous les éléments par les professeurs, à l'aide de la méthode suivante :

```
public Double CalculerNoteModule(Long CodeModule, Long CodeEtudiant) { 1 usage new *
    Module module = getModuleById(CodeModule.toString());
    List<Element> elements = module.getElements();
    Double somme = 0.0;
    Double coef = 0.0;
    for (Element element : elements) {
        NoteElement noteElement = noteElementRepo.findNoteElementByElementAndEtudiant(element.getIdElement(), CodeEtudiant);
        if (noteElement == null) {
            return null;
        }
        somme += noteElement.getNoteElement() * element.getCoefficient();
    }
    Double noteModule = somme / 100;
    return noteModule;
}

public List<Etudiant> EtudiantsOfModule(Long moduleId, Long semestreId, Long filiereId, String searchTerm) { 1 usage new *
    return moduleRepo.EtudiantsOfModule(moduleId, searchTerm, semestreId, filiereId);
}
```

!!! une Note d'un element n'est inseree a la table NoteElement qu'apres la validation de l element, ce qui assure que le calcule de la note de module ne se fait que si les elements sont validees

Exporter Notes

Un administrateur peut exporter les notes des étudiants (selon les filtres choisis), en cliquant sur la boutonne **Exporter**.

Notes du Module

Module: Programmation Avancée (1)

Code Étudiant	Nom Complet	Filière	Note Finale
1	ALAOUI Ahmed	IID	15.45
2	BENJELLOUN Sara	IID	16.55
3	CHAOUI Karim	IID	8.05
5	EL OMARI Youssef	IID	12.7
6	FASSI Leila	IID	N/A
7	GHAZALI Mehdi	IID	N/A
8	HASSANI Nada	IID	N/A

3. Gestion Des Fillieres

Liste Des Fillieres, avec l option d ajouter , modifier , ou supprimer une filiere

The screenshot shows a list of academic programs (Fillieres) in a software interface. Each program card includes its name, description, student count, module count, and creation date. There are 'Edit' and 'Delete' buttons for each entry.

Filliere	Description	Students	Modules	Created On
IID	Informatique et Ingénierie de données	75 Students	0 Modules	Created on 2016-06-02 01:33:45.0
GI	Genie Informatique	75 Students	0 Modules	Created on 2015-01-08 01:33:45.0
IRIC	Réseaux Intelligents et CyberSécurité	75 Students	0 Modules	Created on 2017-10-25 01:35:58.0
MGSI	Management Des Systèmes Informatiques	0 Students	0 Modules	Created on 2024-01-02 01:35:58.0

L ajout d une filiere

This is a form for adding a new academic program. It requires the input of the program name (G_INDUS), a description (Genie Industriel), and a creation date (01/11/2020). There are 'Save' and 'Cancel' buttons at the bottom.

Filliere Name	G_INDUS
Description	Genie Industriel
Date of Creation	01/11/2020

The screenshot shows the updated list of academic programs. The 'G_INDUS' program has been successfully added, appearing alongside the others with its details: 0 students, 0 modules, and a creation date of 01/11/2020.

Filliere	Description	Students	Modules	Created On
GPEE	Genie Des Procedes d Energies Renouvelables	0 Students	0 Modules	Created on 2015-01-21 01:35:58.0
GE	Genie Electrique	0 Students	0 Modules	Created on 2017-01-17 01:35:58.0
G_INDUS	Genie Industriel	0 Students	0 Modules	Created on 2024-01-09 00:00:00.0

Aussi pour la modification : par exemple on va changer la date de creation :

This is an edit form for the 'G_INDUS' program. It allows changing the program name (G_INDUS), description (Genie Industriel), and creation date (06/11/2020). There are 'Save' and 'Cancel' buttons at the bottom.

Filliere Name	G_INDUS
Description	Genie Industriel
Date of Creation	06/11/2020

The screenshot shows the 'G_INDUS' program detail page after the creation date was modified. The creation date is now listed as 06/11/2020, reflecting the change made in the edit form.

Filliere	G_INDUS
Description	Genie Industriel
Students	0 Students
Modules	0 Modules
Created On	Created on 2020-06-11 00:00:00.0

4. Gestion Des Etudiants

La liste Des Etudiants :

Rechercher		Filière	Semestre					
<input type="text"/>	<input type="text"/>	Toutes les filières	Tous les semestres					
IMAGE	ID	CNE	NOM	PRÉNOM	EMAIL	FILIÈRE	SEMESTRE	ACTIONS
	1		ALAOUI	Ahmed	aloui.ahmed1@gmail.com	IID	S1	
	2	P100002	BENJELLOUN	Sara	benjelloun.sara1@gmail.com	IID	S1	
	3	P100003	CHAoui	Karim	chaoui.karim1@gmail.com	IID	S1	
	4	P100004	DRISSI	Amina	driSSI.amina1@gmail.com	IID	S1	

avec l'option de l'ajout , la suppression ou la modification :

Modifier Etudiant

Nom
ALAOUI

Prénom
Ahmed

Email
aloui.ahmed1@gmail.com

Image de l'étudiant
Choose File: Student.png

Filière
IID

Semestre
S1

Enregistrer Annuler

5. Gestion Des Elements

Gestion des Éléments			Liste	Nouveau
<input type="text"/> Rechercher un élément <input type="button"/> Filtrer par module <input type="button"/> Tous les modules <input type="button"/> Filtrer par professeur <input type="button"/> Tous les professeurs				
Structures de Données Avancé 50.0	Conception d'Algorithmes 50.0	Structures de Contrôle 100.0		
Module: Programmation Avancée Professeur: YOUSSEF EL MALKI Code: 1	Module: Programmation Avancée Professeur: AMINA BOUZIANE Code: 2	Module: Algorithmes et Structures Professeur: SALIMA CHAFIK Code: 3		
Introduction aux Réseaux de Neurones 100.0	Préparation des Données 70.0	Statistiques Descriptives 30.0		
Module: Introduction à l'IA Professeur: NOUREDDINE EL JADIDI Code: 4	Module: Analyse des Données Professeur: HIND ABOUDI Code: 5	Module: Analyse des Données Professeur: REDA OUAZZANI Code: 6		

Cette Fenetre Permet aux utilisateurs de d'ajouter , Modifier ,ou Supprimer un element
Modifier

Dans cet Exemple On a change le Professeur de l'element, et on va supprimer une Modalite en modifiant les cofficients:



Formulaire de Modification:

Modifier Élément

Nom de l'Élément
Conception d'Algorithmes

Coefficient
50.0

Module
Programmation Avancée

Professeur
NOUREDDINE EL JADIDI

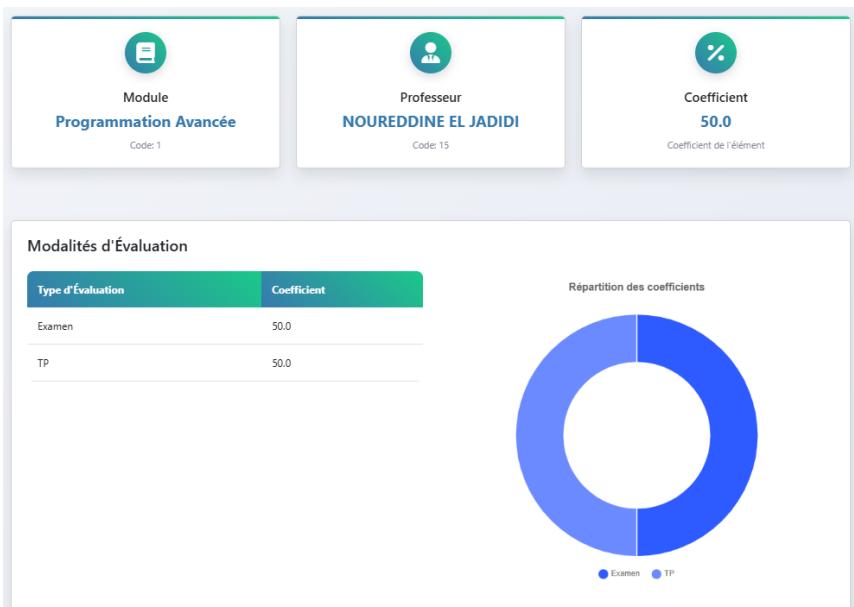
Modalités d'Évaluation

Examen	50.0	Supprimer
TP	50	Supprimer

Ajouter Modalité

Enregistrer Annuler

Apres l'enregistrement , les Modification sont Enregistrees avec succes



On fait la même chose pour l'ajout et la suppression.

Pour l'implementation, on suit la même structure déjà citée.

Pour voir les détails de l'élément, veuillez cliquer sur [Voir Details](#)

DÉTAILS DE L'ÉLÉMENT: CONCEPTION D'ALGORITHMMES

Type d'Évaluation	Coefficient
Examen	50.0
TP	25.0
Projet	25.0

Répartition des coefficients

Type	Coefficient
Examen	50.0
TP	25.0
Projet	25.0

Traitements Professeurs

Le professeur dispose d'un ensemble de fonctionnalités spécifiques liées à la gestion des notes des éléments qui lui sont attribués. Il peut accéder à la liste des étudiants qui lui sont assignés pour saisir leurs notes, les modifier tant que le module n'est pas validé, et procéder à la validation finale des notes d'un élément donné. Le système impose un cadre rigoureux pour la validation des notes : toutes les notes doivent être saisies et comprises entre 0 et 20, avec une distinction claire entre les absents (notés 0) et les notes nulles obtenues lors d'une évaluation. Le professeur peut également sauvegarder des notes en brouillon, annuler une saisie en cours, et une fois l'élément validé, exporter les relevés de notes au format numérique. Le système lui permet aussi de calculer automatiquement les moyennes des éléments et des modules, facilitant ainsi le suivi de la progression des étudiants.

Le Tableau de Bord d'un Professeur

Bienvenue, yassine_prof

Voici un aperçu de votre espace professeur

24 Étudiants Actifs	3 Modules Enseignés	6 Éléments de Module	85% Notes Saisies
-------------------------------	-------------------------------	--------------------------------	-----------------------------

Mes Étudiants

Gérer la liste de vos étudiants par classe et filière

[Accéder →](#)

Mes Éléments

Consulter et gérer vos modules et éléments de module

[Accéder →](#)

Gestion des Notes

Saisir et modifier les notes des étudiants

[Accéder →](#)

Gestion Des Etudiants

Photo	Code	Nom	Prénom	Email	Filière	Semestre	Actions	Saisir Note
	1	ALAOUI	Ahmed	alaoui.ahmed1@gmail.com	IID	S1	Voir Notes	Saisir Note
	2	BENJELLOUN	Sara	benjelloun.sara1@gmail.com	IID	S1	Voir Notes	Saisir Note
	3	CHAOUI	Karim	chaoui.karim1@gmail.com	IID	S1	Voir Notes	Saisir Note
	5	EL OMARI	Youssef	elomari.youssef1@gmail.com	IID	S1	Voir Notes	Saisir Note
	6	FASSI	Leila	fassi.leila1@gmail.com	IID	S1	Voir Notes	Saisir Note
	7	GHAZALI	Mehdi	ghazali.mehdi1@gmail.com	IID	S1	Voir Notes	Saisir Note
	8	HASSANI	Nada	hassani.nada1@gmail.com	IID	S1	Voir Notes	Saisir Note

Lorsque le professeur accède à cette tache l'application lui affiche seulement les étudiants attribués à ce professeur , avec les options de saisir les notes soit par :

Voir Notes : cette fonctionnalité permet au prof d'accéder aux notes de l'étudiant sélectionné pour tous les éléments enseignés par ce professeur à cet étudiant, il permet aussi de modifier ou valider , ou annuler l'opération de saisie

Notes de l'étudiant

ALAOUI Ahmed

Code: 1
Filière: IID
Semestre: S1

Modèles de Régression		
<input checked="" type="button"/> Modifier <input checked="" type="button"/> Valider		Note: Non saisie ^
Modalité	Coefficient	Note
Examen	50.0%	Non saisie
TP	50.0%	Non saisie

Lors de la saisie des Notes , les validations suivantes sont vérifiées :

1- les Notes doivent être entre 0 et 20

Modification des notes - Modèles de Régression

ALAOUI Ahmed

Code: 1
Filière: IID
Semestre: S1

Modalité	Coefficient	Note actuelle	Nouvelle note	Absent
Examen	50.0%	Non saisie	<input type="text" value="21"/> <small>La note doit être comprise entre 0 et 20</small>	/20 <input type="checkbox"/> Absent
TP	50.0%	Non saisie	<input type="text"/> /20	<input type="checkbox"/> Absent

Annuler Enregistrer les notes

2- La Note d'un étudiant Absent dans une modalité est 0 même si je tape une autre note

Modification des notes - Modèles de Régression

ALAOUI Ahmed

Code: 1
Filière: IID
Semestre: S1

Modalité	Coefficient	Note actuelle	Nouvelle note	Absent
Examen	50.0%	Non saisie	<input type="text" value="18"/> /20 <input type="checkbox"/> Absent	
TP	50.0%	Non saisie	<input type="text" value="17"/> /20 <input checked="" type="checkbox"/> Absent	

Annuler Enregistrer les notes

Notes de l'étudiant

ALAOUI Ahmed

Code: 1
Filière: IID
Semestre: S1

Modèles de Régression

Note: Non saisie

Modalité	Coefficient	Note
Examen	50.0%	18.0/20
TP	50.0%	0.0/20

On a toujours la possibilité d'annuler une opération de saisie tant qu'on n'a pas encore validé

3-La note d'élément est calculée automatiquement après la saisie mais n'est enregistrée qu'après la validation

ALAOUI Ahmed

Code: 1
Filière: IID
Semestre: S1

Modèles de Régression

Note: 15.5

Confirmation

Êtes-vous sûr de vouloir valider cet élément ? Cette action est irréversible.

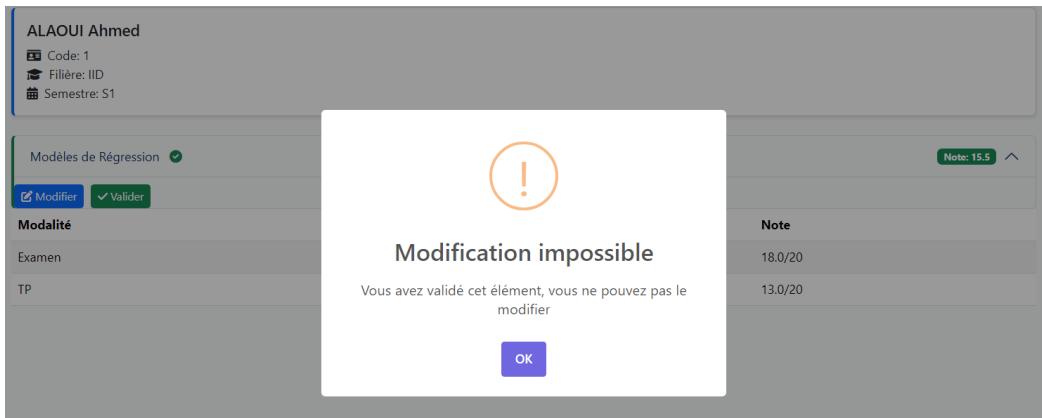
Oui, valider Annuler

Modèles de Régression ✓

Note: 15.5

Modalité	Coefficient	Note
Examen	50.0%	18.0/20
TP	50.0%	13.0/20

4-Apres la validation le professeur n'a plus le droit de modifier la note



Tous ces Operations sont assures par :

```
//pour avoir la liste des etudiants d un professeur
@GetMapping("/etudiants")
    public String listEtudiantsProfesseur(Model model, HttpSession
session,
                                         @RequestParam(required =
false, defaultValue = "") String searchTerm,
                                         @RequestParam(required =
false, defaultValue = "") Long filliereId,
                                         @RequestParam(required =
false, defaultValue = "") Long semestreId,
                                         @RequestParam(required =
false, defaultValue = "") Long codeModule,
                                         @RequestParam(required =
false, defaultValue = "") Long idElement) {

    {
        if (idElement!=null){
            Element element = elementService.getElementById(id
Element.toString());
            List<Modalite_Evaluation> modalites = modalite_eva
luationService.getAllModalite_Evaluations_byElement(element.getIdE
lement());
            model.addAttribute("modalites", modalites);
        }

        model.addAttribute("fillieres", filliereService.getAll
Fillieres());
        model.addAttribute("semestres", semestreService.getAll
Semestres());
        model.addAttribute("modules", moduleService.getAllModu

```

```

les());
model.addAttribute("elements", elementService.getAllElements());

        // Récupérer le professeur connecté depuis la session
        Compte Compte = (Compte) session.getAttribute("loggedInUser");
        Professeur professeur = professeurService.getProfesseurById(Compte.getUtilisateur().getCode().toString());

        if (professeur == null) {
            return "redirect:/utilisateur/loginPage";
        }

        // Récupérer uniquement les étudiants associés à ce professeur
        List<Etudiant> etudiantsProfesseur = professeurService.getEtudiantsByProfesseur(professeur.getCode(), searchTerm, filiereId, semestreId, codeModule, idElement);

        model.addAttribute("etudiants", etudiantsProfesseur);
        //model.addAttribute("filieres", professeur.getFilières());
        // Si le professeur a des filières spécifiques
        return "prof_etudiants-list";
    }
}

```

la Modification de la Note

```

@GetMapping("/etudiants/notes/{idEtudiant}/{idElement}/ModifierNotes")
public String modifierNotesEtudiant(@PathVariable("idEtudiant") String idEtudiant,@PathVariable("idElement") String idElement,
Model model, HttpSession session) {
    Compte compte = (Compte) session.getAttribute("loggedInUser");
    Professeur professeur = professeurService.getProfesseurById(compte.getUtilisateur().getCode().toString());
    Etudiant etudiant = etudiantService.getEtudiantByCode(idEtudiant);
    Element element = elementService.getElementById(idElement);
}

```

```

        model.addAttribute("etudiant", etudiant);
        model.addAttribute("element", element);
        NoteElement noteElement = element.getNoteElementByEtudiant
(etudiant);

        if (noteElement != null) {
            model.addAttribute("noteElement", noteElement);
            if (!noteElement.getIsValid()) {
                return "notes_form-modifier";
            }
            else {
                return "redirect:/professeur/etudiants/notes/" + i
dEtudiant;
            }
        }
        return "notes_form-modifier";
    }
}

```

La Validation d une Note

```

@GetMapping("/etudiants/notes/{idEtudiant}/{idElement}/Valider")
    public String validerNotesEtudiant(@PathVariable("idEtudiant")
String idEtudiant,@PathVariable("idElement") String idElement, Mod
el model,HttpSession session) {
    Compte compte = (Compte) session.getAttribute("loggedInUse
r");
    Professeur professeur = professeurService.getProfesseurByI
d(compte.getUtilisateur().getCode().toString());
    Etudiant etudiant = etudiantService.getEtudiantByCode(idEt
udiant);
    Element element = elementService.getElementById(idElemen
t);
    model.addAttribute("etudiant", etudiant);
    model.addAttribute("element", element);
    NoteElement noteElement = element.getNoteElementByEtudiant
(etudiant);

    if (noteElement != null) {
        model.addAttribute("noteElement", noteElement);
        noteElement.setValid(true);
        noteElementService.saveNoteElement(noteElement);
        return "redirect:/professeur/etudiants/notes/" + idEtu
}
}

```

```

diant;
    }
    Module module = element.getModule();
    boolean isElementofModuleValid = noteModuleService.isElementsOfModuleValid(module, etudiant);
    if (isElementofModuleValid==false) {
        NoteModule noteModule = noteModuleService.getNoteModuleByModule_IdAndEtudiant_Id(module.getCodeModule(), etudiant.getCodeEtudiant());
        if (noteModule == null) {
            noteModule = new NoteModule();
            noteModule.setEtudiant(etudiant);
            noteModule.setModule(module);
            noteModule.setNoteModule(noteModuleService.calculateModuleNoteForEtudiant(module, etudiant));
        }
        noteModuleService.saveNoteModule(noteModule);
    }

    return "redirect:/professeur/etudiants/notes/" + idEtudiant;
}

```

L'enregistrement et le calcul des Notes(on enregistre les note de modalites , puis en calcule la note de l element a partir des notes saisis et les cofficients)

```

@PostMapping("/etudiants/notes/{idEtudiant}/{idElement}/save")
public String saveNotesEtudiant(
    @PathVariable("idEtudiant") String idEtudiant,
    @PathVariable("idElement") String idElement,
    Model model,
    HttpSession session,
    @RequestParam Map<String, String> allParams) {

    // Get required entities
    model.addAttribute("elements", filliereService.getAllFillieres());
    Etudiant etudiant = etudiantService.getEtudiantByCode(idEtudiant);
    Element element = elementService.getElementById(idElement);
    List<Modalite_Evaluation> modalites = modalite_evaluations

```

```

service.getAllModalite_Evaluations_byElement(element.getIdElement());
}

double Noteelement = 0.0;
double totalCoefficients=0.0;

for (Modalite_Evaluation modalite : modalites) {
    String noteKey = "note_" + modalite.getId_Modalite();
    String absentKey = "absent_" + modalite.getId_Modalite();
}

// Check if student was marked as absent
boolean isAbsent = allParams.containsKey(absentKey) &&
allParams.get(absentKey).equals("on");

// trouver la note de la modalité pour cet étudiant
// s'il existe déjà ou créer une nouvelle
NoteModalite noteModalite = noteModaliteService.getNoteModaliteByModaliteAndEtudiant(modalite.getId_Modalite(), etudiant.getCodeEtudiant());

if (noteModalite == null) {
    // Create new note if doesn't exist
    noteModalite = new NoteModalite();
    noteModalite.setEtudiant(etudiant);
    noteModalite.setModaliteEvaluation(modalite);
}

// Update note values
if (isAbsent) {
    noteModalite.setAbsent(true);
    noteModalite.setNoteModalite(0.0);
} else {
    String noteValue = allParams.get(noteKey);
    if (noteValue != null && !noteValue.isEmpty()) {
        double note = Double.parseDouble(noteValue);
        noteModalite.setAbsent(false);
        noteModalite.setNoteModalite(note);
    }
}

// Add to weighted sum for element note calcul

```

```

        ation
                    Noteelement += note * (modalite.getCoeffecient
        () / 100.0);
                    totalCoefficients += modalite.getCoeffecient
        ();
                }
            }

        // Save or update note modalité
        noteModaliteService.saveNoteModalite(noteModalite);
    }

    // Calculate and save element note
    if (totalCoefficients == 100) {
        // Trouver la note de l'élément pour cet étudiant s'il
        existe déjà ou créer une nouvelle
        NoteElement noteElement = noteElementService.getNoteEl
        ementByElementAndEtudiant(element.getIdElement(), etudiant.getCode
        Etudiant());
        if (noteElement == null) {
            noteElement = new NoteElement();
            noteElement.setEtudiant(etudiant);
            noteElement.setElement(element);
            noteElement.setNoteElement(Noteelement);
        }
        noteElementService.saveNoteElement(noteElement);
    }

    return "redirect:/professeur/etudiants/notes/" + idEtudian
t;
}

```

Saisir Notes: Approche plus rapide pour la saisie des notes de tous les etudiants a la fois dans la meme page sans acceder une autre page

The screenshot shows the 'GNotes - Espace Professeur' dashboard. At the top, there are search filters for 'Rechercher par nom ou prénom', 'Toutes les filières', 'Tous les semestres', 'Apprentissage Autor.', and 'Tous les éléments'. Below is a table of students:

Photo	Code	Nom	Prénom	Email	Filière	Semestre	Actions	Saisir Note
	1	ALAOUI	Ahmed	alaoui.ahmed1@gmail.com	IID	S1	Voir Notes	Saisir Note
	2	BENJELLOUN	Sara	benjelloun.sara1@gmail.com	IID	S1	Voir Notes	Saisir Note

A modal window titled 'Saisie des notes - BENJELLOUN Sara' is open, showing the note entry form for student ID 2. It includes fields for 'Modalité', 'Coefficient', 'Note actuelle', 'Nouvelle note', and 'Absent'. The 'Absent' checkbox is unchecked. Buttons at the bottom are 'Annuler' and 'Enregistrer les notes'.

Les même validations et règles présentes précédemment reste valable

Gestion Des Elements

The screenshot shows the 'Mes Éléments d'Enseignement' page. At the top, there are search filters for 'Rechercher un élément', 'Toutes les filières', 'Tous les semestres', and 'Tous les modules'.

Two course cards are displayed:

- Modèles de Régression** (Coef: 60.0)
 - Apprentissage Automatique
 - Modalités d'évaluation:
 - Examen (50.0%)
 - TP (50.0%)
 Statistics: 32 Étudiants, 75% Réussite, 16 Moyenne
- Optimisation Combinatoire** (Coef: 100.0)
 - Optimisation et Recherche
 - Modalités d'évaluation:
 - Examen (50.0%)
 - TP (50.0%)
 Statistics: 32 Étudiants, 80% Réussite, 14 Moyenne

Each card has a 'Gérer les Notes' button at the bottom.

La page ci-dessus affiche les éléments affectés au professeur connecté. Chaque carte d'élément comporte les informations sur les modalités, le nombre d'étudiants, la moyenne et le taux de réussite actuels. De plus, elle contient deux boutons :

Gérer Notes pour la saisie des notes de cet élément , il redirige vers la page suivante :

GNotes - Espace Professeur

Rechercher par nom ou prénom: Toutes les filières: Tous les semestres: Apprentissage Autor: Tous les éléments:

Photo	Code	Nom	Prénom	Email	Filière	Semestre	Actions	Saisir Note
	1	ALAOUI	Ahmed	alaoui.ahmed1@gmail.com	IID	S1	Voir Notes	Saisir Note
	2	BENJELLOUN	Sara	benjelloun.sara1@gmail.com	IID	S1	Voir Notes	Saisir Note
	3	CHAOUI	Karim	chaoui.karim1@gmail.com	IID	S1	Voir Notes	Saisir Note
	5	EL OMARI	Youssef	elomari.youssef1@gmail.com	IID	S1	Voir Notes	Saisir Note
	6	FASSI	Leila	fassi.leila1@gmail.com	IID	S1	Voir Notes	Saisir Note
	7	GHAZALI	Mehdi	ghazali.mehdi1@gmail.com	IID	S1	Voir Notes	Saisir Note
	8	HASSANI	Nada	hassani.nada1@gmail.com	IID	S1	Voir Notes	Saisir Note

Exporter pour exporter la liste des étudiants avec leurs notes saisies

Liste des Étudiants à Exporter - Modèles de Régression

[Exporter en PDF](#)

Rechercher par nom ou prénom: Toutes les filières: S1: Apprentissage Autor:

Photo	Code	Nom	Prénom	Email	Filière	Semestre	Element	Note
	1	ALAOUI	Ahmed	alaoui.ahmed1@gmail.com	IID	S1	Modèles de Régression	15.5/20
	2	BENJELLOUN	Sara	benjelloun.sara1@gmail.com	IID	S1	Modèles de Régression	13.5/20
	3	CHAOUI	Karim	chaoui.karim1@gmail.com	IID	S1	Modèles de Régression	14.0/20
	5	EL OMARI	Youssef	elomari.youssef1@gmail.com	IID	S1	Modèles de Régression	12.0/20
	6	FASSI	Leila	fassi.leila1@gmail.com	IID	S1	Modèles de Régression	9.0/20
	7	GHAZALI	Mehdi	ghazali.mehdi1@gmail.com	IID	S1	Modèles de Régression	11.0/20
	8	HASSANI	Nada	hassani.nada1@gmail.com	IID	S1	Modèles de Régression	15.5/20
	9	IDRISSI	Omar	idrissi.omar1@gmail.com	IID	S1	Modèles de Régression	16.0/20
	10	JEBLI	Salma	jebli.salma1@gmail.com	IID	S1	Modèles de Régression	16.0/20

En cliquant sur "Exporter", un fichier PDF est téléchargé contenant les notes des étudiants avec les filtres sélectionnés

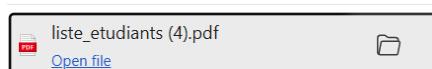


Photo	Code	Nom	Prénom	Email	Filière	Semestre	Element	Note
	1	ALAOUI	Ahmed	alaoui.ahmed1@gmail.com	IID	S1	Modèles de Régression	15.5/20
	2	BENJELLOUN	Sara	benjelloun.sara1@gmail.com	IID	S1	Modèles de Régression	13.5/20
	3	CHAOUI	Karim	chaoui.karim1@gmail.com	IID	S1	Modèles de Régression	14.0/20
	5	EL OMARI	Youssef	elomari.youssef1@gmail.com	IID	S1	Modèles de Régression	12.0/20
	6	FASSI	Leila	fassi.leila1@gmail.com	IID	S1	Modèles de Régression	9.0/20
	7	GHAZALI	Mehdi	ghazali.mehdi1@gmail.com	IID	S1	Modèles de Régression	11.0/20
	8	HASSANI	Nada	hassani.nada1@gmail.com	IID	S1	Modèles de Régression	15.5/20

le code suivant suivant permet de préparer la liste des à exporter

```

21 public class ProfesseurController {
22     public String listEtudiantsProfesseurexport(@PathVariable Long idElement, Model model, HttpSession session,
23                                                 @RequestParam(required = false, defaultValue = "") String searchTerm,
24                                                 @RequestParam(required = false, defaultValue = "") Long filiereId,
25                                                 @RequestParam(required = false, defaultValue = "") Long semestreId,
26                                                 @RequestParam(required = false, defaultValue = "") Long codeModule){
27
28     // Récupérer le professeur connecté depuis la session
29     Compte Compte = (Compte) session.getAttribute("loggedInUser");
30     Professeur professeur = professeurService.getProfesseurById(Compte.getUtilisateur().getCode().toString());
31     model.addAttribute("professeur", professeur);
32     model.addAttribute("filiieres", filiereService.getAllFiliieres());
33     model.addAttribute("semestres", semestreService.getAllSemestres());
34     model.addAttribute("modules", moduleService.getAllModules());
35     model.addAttribute("element", elementService.getElementById(idElement.toString()));
36
37     if (professeur == null) {
38         return "redirect:/utilisateur/loginPage";
39     }
40     // Récupérer uniquement les étudiants associés à ce professeur
41     List<Etudiant> etudiantsProfesseur = professeurService.getEtudiantsByProfesseur(professeur.getCode(), searchTerm, filiereId);
42     model.addAttribute("etudiants", etudiantsProfesseur);
43     //model.addAttribute("filiieres", professeur.getFiliieres()); // Si le professeur a des filières spécifiques
44     return "prof_etudiants-listAexport";
45 }
46
47
48 }
```

et l'action de télécharger le fichier pdf est implémentée avec javascript en utilisant la fonction suivante :

```

function exportToPDF() {
    const element = document.getElementById('exportTable');
    const opt = {
        margin: 1,
        filename: 'liste_etudiants.pdf',
        image: { type: 'jpeg', quality: 0.98 },
        html2canvas: { scale: 2 },
        jsPDF: { unit: 'in', format: 'a4', orientation: 'landscape' }
    };

    html2pdf().set(opt).from(element).save();
}
</script>
```

Conclusion

En conclusion, ce projet de gestion des notes représente une réalisation significative dans le domaine des applications de gestion académique. L'utilisation des technologies Java EE, combinée à une architecture robuste et modulaire, a permis de créer une solution qui répond efficacement aux besoins complexes de la gestion des notes universitaires.

Les accomplissements principaux de ce projet comprennent une interface utilisateur intuitive et responsive qui facilite la saisie et la gestion des notes pour les professeurs. Le système intègre un calcul automatique sophistiqué prenant en compte les différentes modalités d'évaluation, les coefficients associés et les règles de validation spécifiques. De plus, il offre des fonctionnalités avancées comme l'export des données au format PDF, la validation des notes avec des contrôles stricts et la gestion des absences.

Du point de vue technique, l'application démontre une mise en œuvre réussie des concepts clés de Java EE avec une architecture MVC bien structurée, une gestion efficace des sessions utilisateur, une interaction optimisée avec la base de données et une sécurisation appropriée des données et des accès. Les aspects de sécurité ont été particulièrement bien traités, incluant des contrôles d'accès basés sur les rôles, une validation des données côté serveur, une protection contre les modifications non autorisées et une gestion sécurisée des sessions.

Les perspectives d'évolution de ce projet sont nombreuses, notamment l'ajout d'une interface mobile, l'intégration de fonctionnalités de statistiques avancées, la mise en place d'un système de notification et l'extension des capacités d'export vers d'autres formats.

En définitive, ce projet constitue non seulement une solution fonctionnelle pour la gestion des notes, mais aussi une base solide pour de futures améliorations et extensions. Il démontre une maîtrise approfondie des technologies Java EE et une compréhension claire des besoins métier dans le contexte académique.