Rapport de Projet:

Gestion d’une scolarité

Java Enterprise Edition

CY Tech 2024/2025 S1
M. Haddache
21/10/2024 - 30/11/2024

Ingénieur 2e année
Génie des systèmes d’informations
Groupe 3

Membres:

- ARMANNO Yannis
- CHEN Kevin
- GERMAIN Baptiste
- KAU Pascal
- LE COADOU Clément

Sommaire

Introduction

Ce rapport accompagne un projet JEE (Java Enterprise Edition) réalisé dans le cadre des cours à CYTech, en ING2, semestre 1. Il a pour objectif de détailler la partie conception du projet et d'expliquer le schéma de la partie implémentation. Ce projet a été réalisé par un groupe de cinq personnes : ARMANNO Yannis, CHEN Kevin, GERMAIN Baptiste, KAU Pascal et LE COADOU Clément. Nous appartenons tous au même groupe, en Génie des Systèmes d'Information (GSI), groupe 3.

Le projet a débuté le 21 octobre 2024, date à laquelle les sujets ont été communiqués sur Teams, et se termine le 30 novembre 2024. Par ailleurs, une présentation orale de notre travail, d'une durée de 10 à 15 minutes, est demandée afin de démontrer notre compréhension du sujet. Cette présentation aura lieu le 16 décembre 2024 à 11h10.

L'architecture de ce projet est disponible sur un dépôt GitHub accessible via le lien suivant : https://github.com/clement-le-coadou/gestion-scolarite_v2

Pour réaliser ce projet, nous avons utilisé plusieurs outils et technologies adaptés à la création d'une application web. Tout d'abord, nous avons travaillé avec le langage Java pour gérer la logique du site et connecter le tout à une base de données. Cette base de données, créée avec **MySQL**, permet de stocker toutes les informations importantes, comme les étudiants, les enseignants, ou encore les notes. Pour que Java et la base de données puissent "dialoguer" facilement, nous avons utilisé un outil appelé **Hibernate**, qui simplifie ces échanges.

Le site web en lui-même a été conçu avec des pages spéciales appelées JSP (Java Server Pages), qui permettent d'afficher les données sous forme de pages web. Ces pages ont été mises en forme grâce à **Bootstrap**, qui constitue un style CSS déjà préparé et adapté à tous les écrans (ordinateur, tablette, téléphone).

Pour faire fonctionner le site, nous avons utilisé un serveur nommé Apache Tomcat (v10.1). Nous avons respecté la modélisation MVC Modèle vue contrôleur. Le **MVC** permet de séparer les différents traitements et fonctionnalités de l'application. En effet le **Modèle** correspond aux **classes java** et leur intéractions, qui est modifiée et lue par le **Contrôleur** (les servlets) qui fabrique et envoie au client la **Vue** en fonction des données récupérées (page JSP).

Nous avons organisé le projet de manière structurée grâce à un système appelé **Maven**, qui nous aide à bien ranger et gérer tout le code. Tout le développement a été fait sur un logiciel IDE appelé Eclipse, qui est un outil pratique pour écrire et tester notre travail.

Pour travailler en équipe, nous avons aussi utilisé des outils de collaboration. Par exemple, Github Desktop nous a permis de partager facilement notre code entre les membres de l'équipe, tandis que Figma a servi à dessiner des schémas pour planifier notre site. Enfin, Monday nous a aidés à organiser les tâches de chacun et à suivre notre progression pour s'assurer que nous respections les délais.

Grâce à cette combinaison d'outils, nous avons pu créer un site web fonctionnel et répondre aux besoins du projet tout en travaillant efficacement en équipe.

La partie conception :

Présentation du diagramme de classes qui représente les différentes interactions entre les entités (classes) de l'application ainsi que l'organisation des servlets .

Ce projet consiste en la création d'une application web dédiée à la gestion de la scolarité. Elle vise à centraliser et faciliter la gestion des informations académiques pour trois types d'utilisateurs : les étudiants, les enseignants et les administrateurs. L'application offre un ensemble de fonctionnalités essentielles pour répondre aux besoins de chaque groupe.

Tout d'abord, la gestion des étudiants permet d'enregistrer, mettre à jour, rechercher et afficher leurs informations personnelles, ainsi que de supprimer des éléments si nécessaire. De même, la gestion des enseignants comprend l'enregistrement et la modification de leurs informations, ainsi que leur affectation aux cours.

L'application intègre également un module de gestion des cours, permettant leur création, modification ou suppression, ainsi que l'attribution des enseignants et des étudiants aux différents cours. En parallèle, un système d'inscriptions facilite l'enregistrement des étudiants aux cours et permet à chacun de consulter les cours auxquels il est inscrit.

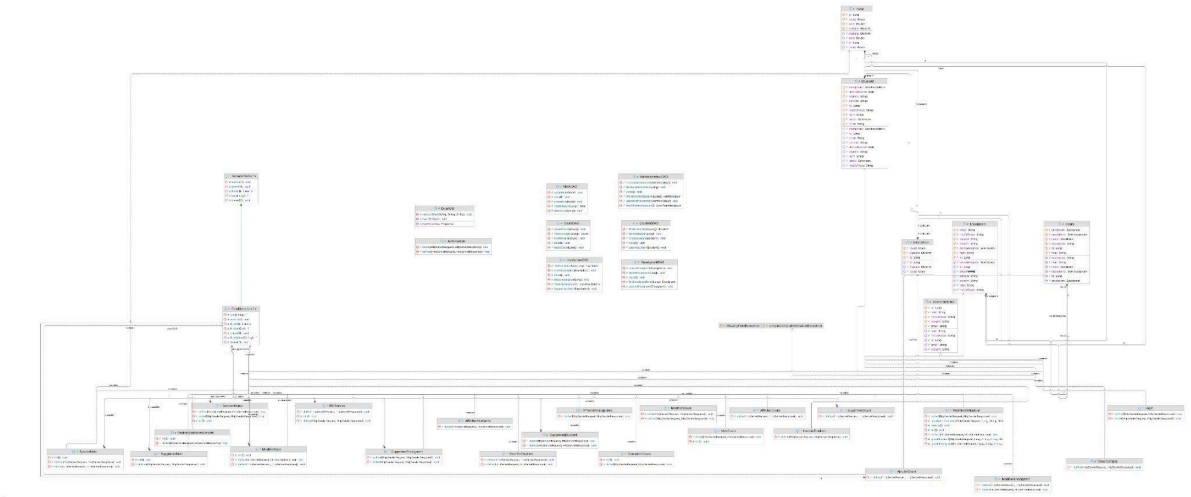
Pour évaluer les performances académiques, les enseignants disposent d'une interface dédiée à la saisie des notes des étudiants. Les résultats sont ensuite accessibles via un module de gestion des résultats, qui offre des fonctionnalités telles que l'affichage des notes, et la génération de relevés de notes individuels en format (.pdf).

Enfin, un système d'authentification et de gestion des rôles assure la sécurité et la personnalisation de l'accès en fonction des profils utilisateurs : étudiant, enseignant ou administrateur.

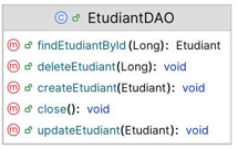
Cette application a pour ambition d'améliorer la gestion académique en centralisant les données et en simplifiant les interactions entre les différents acteurs de l'institution.

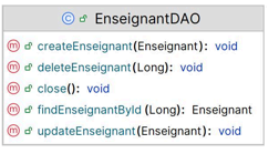
Après une discussion en classe (au dernier cours soit 4 jours avant le rendu) nous avons compris que vous attendiez un système prenant en charge un emploi du temps avec une salle, un horaire, une durée. Nous avons donc ajouté cette fonctionnalité en urgence uniquement sur le projet Spring Boot.
Il y a donc une page qui permet la création d'un créneau pour un cours. Attention, le cours doit être créé auparavant. Il y a plusieurs sécurité qui empêchent l'ajout d'un créneaux d'un cours qui soit n'est pas enseigné par l'enseignant sélectionné, soit l'élève sélectionné n'y est pas inscrit. De même, il n'est pas possible de choisir un horaire en dehors de la plage horaire 8h00 à 17h00, ou encore de sélectionner une salle déjà utilisée sur cet horaire. Une autre sécurité permet d'éviter que plusieurs cours se chevauchent sur la même plage horaire pour un même élève ou enseignant.

Diagramme de classe:



classe DAO et Mail





Ces DAO possèdent les méthodes de gestion de la base de données, ils seront appelés dans les servlets.

Relation entre les JPA:

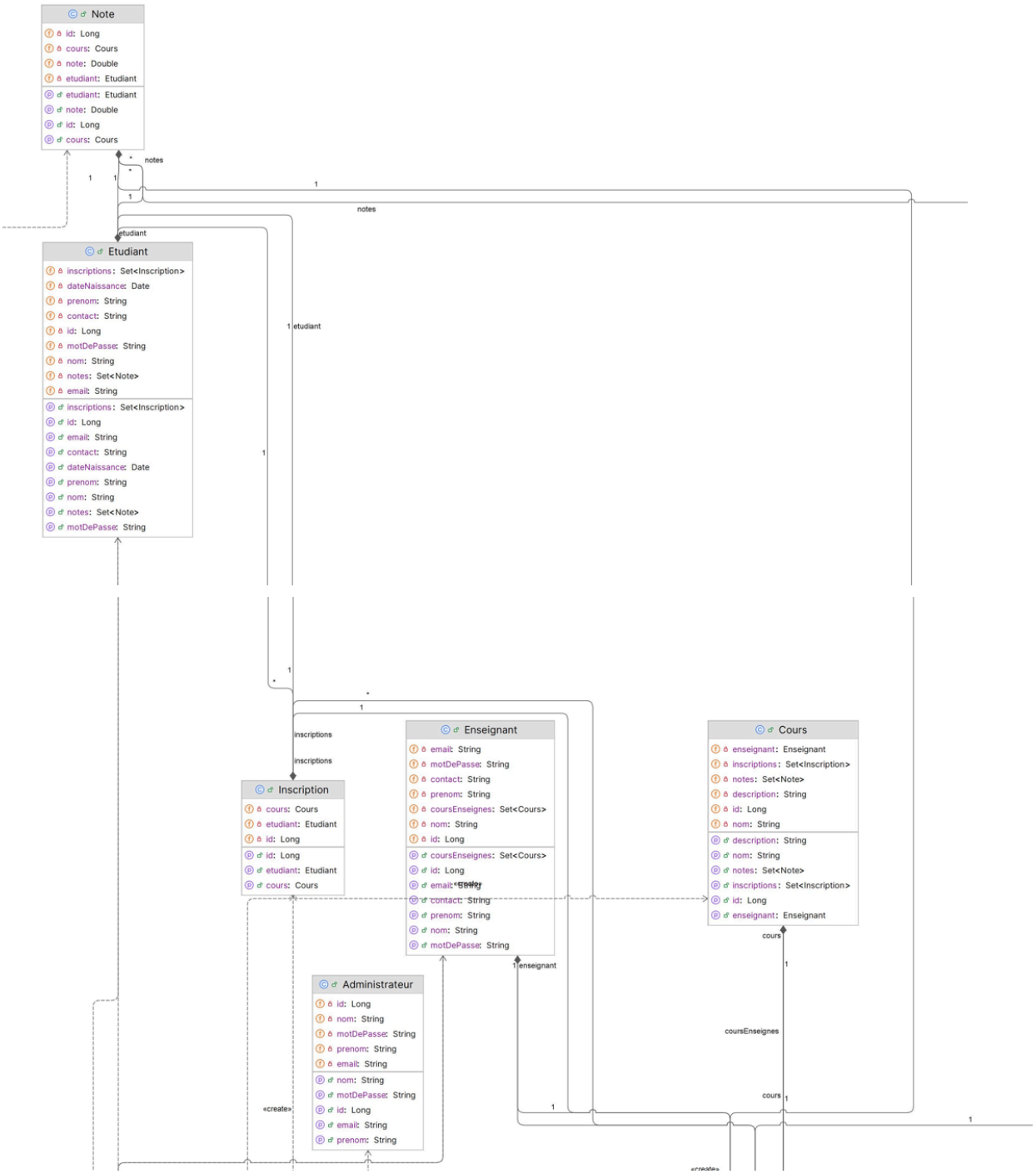
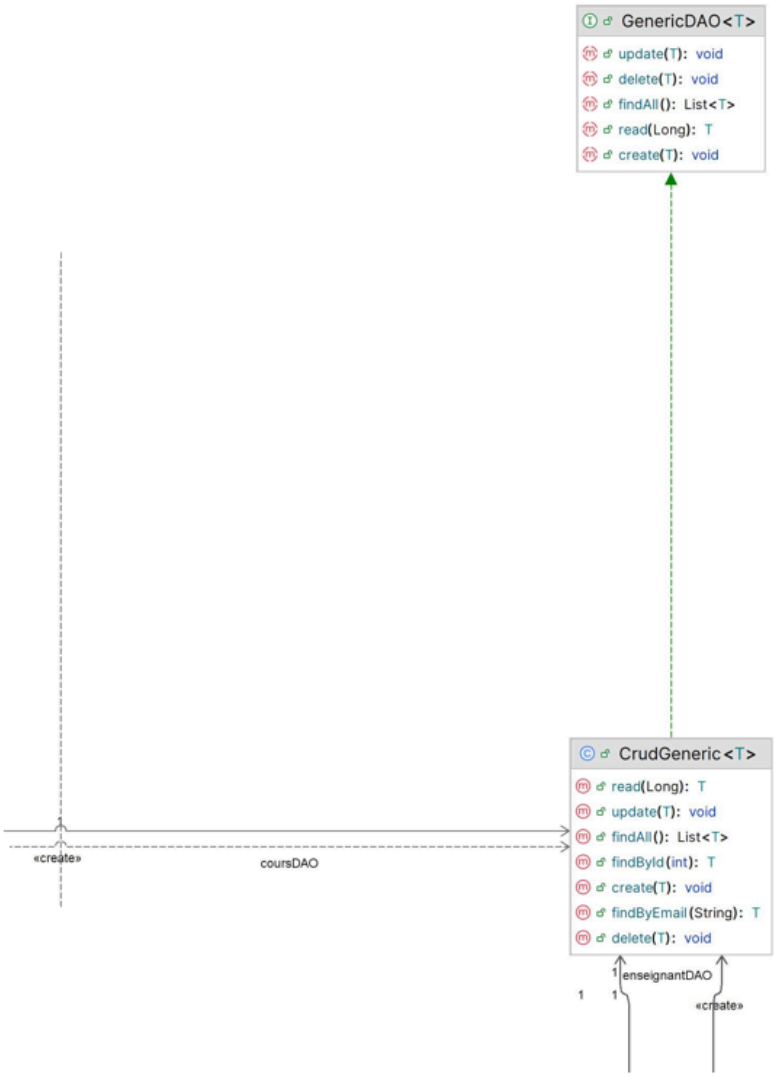


Diagramme principale qui permet de comprendre les relations entre les différentes classes du modèle

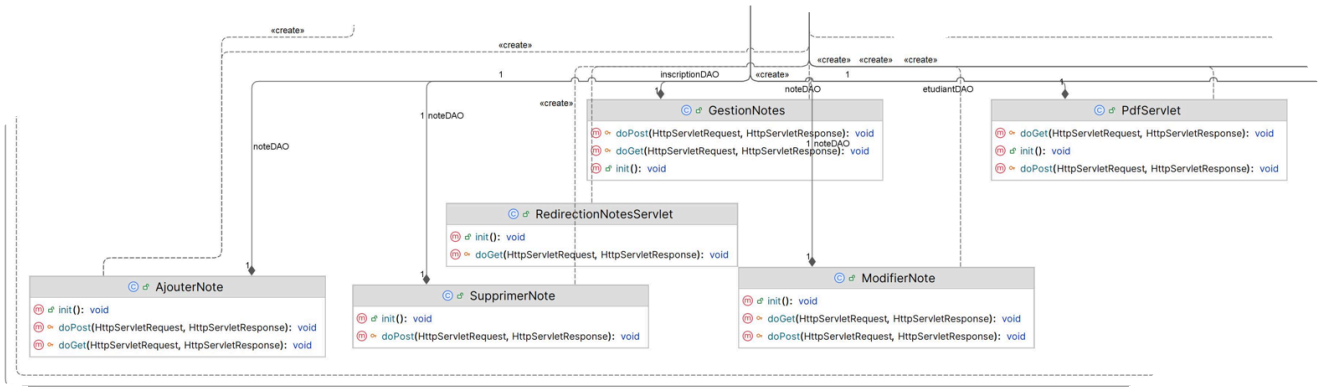
Généralisation des DAO par héritage



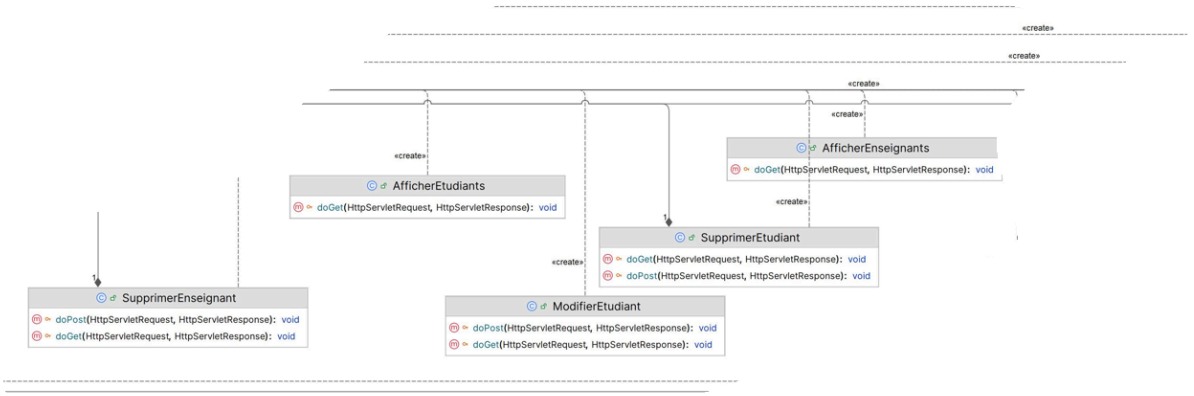
Cette généralisation permet de centraliser la logique commune (dans GenericDAO<T>) et de la réutiliser dans des classes plus spécifiques (CrudGeneric<T>) .Cela suit le principe de programmation orientée objet pour éviter la duplication du code.

Servlet qui hérite des CRUD :

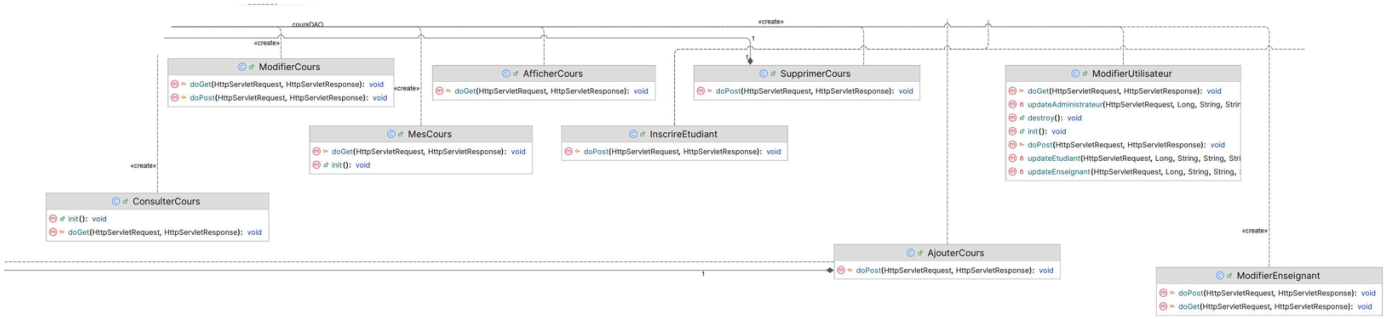
Note:



Enseignant et étudiant:



Cours et authentification



Modification, création, suppression et affichage pour les 3 types d'utilisateurs, ainsi que les cours, inscription et les notes.
Remarquons que la création d'un étudiant, enseignant se fait par la page de connexion.

Implémentation

Pour l'implémenter, nous avons identifié trois rôles principaux pour les utilisateurs de l'application : **Étudiants**, **Enseignants** et **Administrateurs**. Chaque rôle dispose de privilèges spécifiques en fonction de ses responsabilités au sein du système, tout en garantissant une gestion centralisée et efficace des données académiques.

Gestion des étudiants

Les étudiants peuvent accéder à des fonctionnalités essentielles liées à leur parcours académique. Ils ont la possibilité de consulter la liste des cours auxquels ils sont inscrits, d'afficher leurs résultats comprenant les notes, les moyennes et les relevés détaillés, ainsi que de gérer leurs propres informations personnelles, telles que leur nom, prénom, date de naissance et coordonnées.

Gestion des enseignants

Les enseignants, quant à eux, jouent un rôle central dans la gestion pédagogique. Ils peuvent visualiser les cours qu'ils enseignent et disposer d'une interface pour saisir les notes des étudiants. En outre, ils ont la capacité de mettre à jour leurs propres informations personnelles. Afin de mieux suivre leurs étudiants, les enseignants ont également accès à des fonctionnalités leur permettant de consulter la liste des étudiants inscrits à leurs cours, d'accéder aux informations académiques des étudiants et de vérifier les cours auxquels ces derniers sont inscrits.

Gestion des administrateurs

Enfin, les administrateurs détiennent les droits les plus étendus dans l'application, leur permettant d'assurer la gestion globale du système. Leur rôle consiste principalement à créer, modifier ou supprimer des cours, à inscrire les étudiants aux cours et à affecter les enseignants à ces derniers. Ils peuvent également gérer les informations des étudiants et des enseignants en effectuant des mises à jour ou des suppressions. De plus, les administrateurs ont accès à toutes les données du système, notamment la liste des étudiants et des enseignants, leurs détails académiques, ainsi que les cours auxquels ils sont associés. Ils sont également responsables de la gestion des rôles et des autorisations pour garantir une utilisation sécurisée et adaptée des fonctionnalités de l'application.

Cette répartition claire des rôles et responsabilités contribue à une gestion efficace et organisée des données académiques, tout en répondant aux besoins spécifiques de chaque utilisateur.

Authentification

Pour gérer l'accès à l'application, nous avons mis en place un système d'authentification et d'autorisations basé sur des comptes utilisateurs liés à la base de données. Comme dit précédemment, le système repose sur trois rôles distincts : étudiants, enseignants et administrateurs, chacun ayant des droits spécifiques au sein de l'application.

Toutefois, contrairement aux administrateurs, les étudiants et les enseignants ne peuvent pas créer eux-mêmes de compte. L'administrateur est le seul habilité à créer ou gérer les comptes des utilisateurs, qu'il s'agisse d'étudiants ou d'enseignants.

Cette organisation simplifie l'accès et renforce la sécurité, car les identifiants sont fournis directement par l'administrateur. Pour cette raison, notre application dispose uniquement d'une page de connexion, similaire à celle utilisée à CY Tech pour accéder à l'ENT. Lors de leur première connexion, les utilisateurs, qu'ils soient étudiants ou enseignants, utilisent les identifiants transmis pour accéder à leurs fonctionnalités spécifiques.

Gestion des cours & Inscription

Pour implémenter les cours et l'inscription au cours, il nous a d'abord fallu penser à l'association cours-enseignant-étudiant, comme un cours peut être associé à un enseignant mais plusieur élève, nous avons donc pensé à associer le professeur directement au cour puis utilisé une classe inscription qui prenait l'identifiant du cours et qui lui associera un étudiant, ainsi la table inscription contiendra le lien étudiant-cours.

Pour la gestion des cours et donc des inscriptions par ailleurs, les droits de ces derniers ne peuvent être accordés qu'à l'admin par souci de sécurité et de gestion.

Saisie des notes

Il faut identifier 3 cas d'accès aux notes:

- étudiant: accès à ses notes et génération de son bulletin.
- enseignant: accès à ses cours uniquement, possibilité de CRUD sur la page de gestions et d'affichage qui affiche tous les étudiant inscrit à son cours.
- administrateur: accès à tous les cours et CRUD comme enseignant.

Pour cela il nous faut une page de redirection en fonction du type de rôle, puis une page de choix de cours pour choisir quel cours on veut gérer les notes.

Génération des résultats

La gestion des résultats est un aspect crucial de l'application. Nous avons défini l'accès pour l'Étudiants, accès restreint à leurs propres notes, avec la possibilité de générer un bulletin détaillé sous forme PDF.

Elle se base sur iText en raison de sa robustesse et de ses fonctionnalités avancées pour manipuler des PDF. Cela nous permet d'inclure des styles complexes comme :

- Des couleurs pour les en-têtes.
- Une mise en page cohérente avec des alignements et marges spécifiques.
- La génération dynamique de contenu comme les noms des cours et les notes.

Email

L'utilisation d'un système de notification par email s'impose comme une solution simple et efficace pour communiquer avec les utilisateurs. Cette fonctionnalité permet, par exemple, d'informer les étudiants des changements dans leurs cours ou notes.

Nous avons choisi un système basé sur un serveur SMTP pour sa compatibilité et sa simplicité. En utilisant des outils standardisés, nous assurons une configuration facile et une compatibilité avec la plupart des fournisseurs d'emails. Le choix d'un tel système garantit également une sécurité des échanges via des protocoles chiffrés (TLS/SSL).

DAO Générique

La création d'un DAO (Data Access Object) générique nous permet de centraliser et d'uniformiser l'accès aux données dans le système. Cette approche s'inscrit dans une optique de maintenabilité et de réutilisation du code. En ayant un DAO générique capable de gérer toutes les entités (cours, étudiants, enseignants, inscriptions, etc.), nous évitons la redondance et pouvons appliquer des modifications globales plus facilement.

Ce choix facilite également l'intégration avec des frameworks comme JPA, qui s'appuie sur des entités Java mappées directement sur les tables de la base de données. Nous avons pensé à inclure des méthodes génériques telles que findAll, findById, save, update et delete, tout en les personnalisant pour répondre aux besoins spécifiques de certaines entités.

JPA

Pour interagir avec la base de données, nous avons privilégié JPA (Java Persistence API) pour ses capacités d'abstraction et sa compatibilité avec Hibernate. JPA permet de mapper les entités Java directement sur les tables, simplifiant ainsi les opérations de persistance. Ce choix réduit la complexité de gestion des requêtes SQL brutes tout en améliorant la lisibilité et la maintenabilité du code.

La réflexion derrière ce choix est de tirer parti des annotations de JPA pour gérer les relations entre les entités (comme les relations @OneToMany entre cours et étudiants, ou @ManyToOne pour l'enseignant associé à un cours). Cela permet de définir un modèle de données clair et aligné avec les besoins métier, tout en respectant les contraintes d'intégrité de la base.

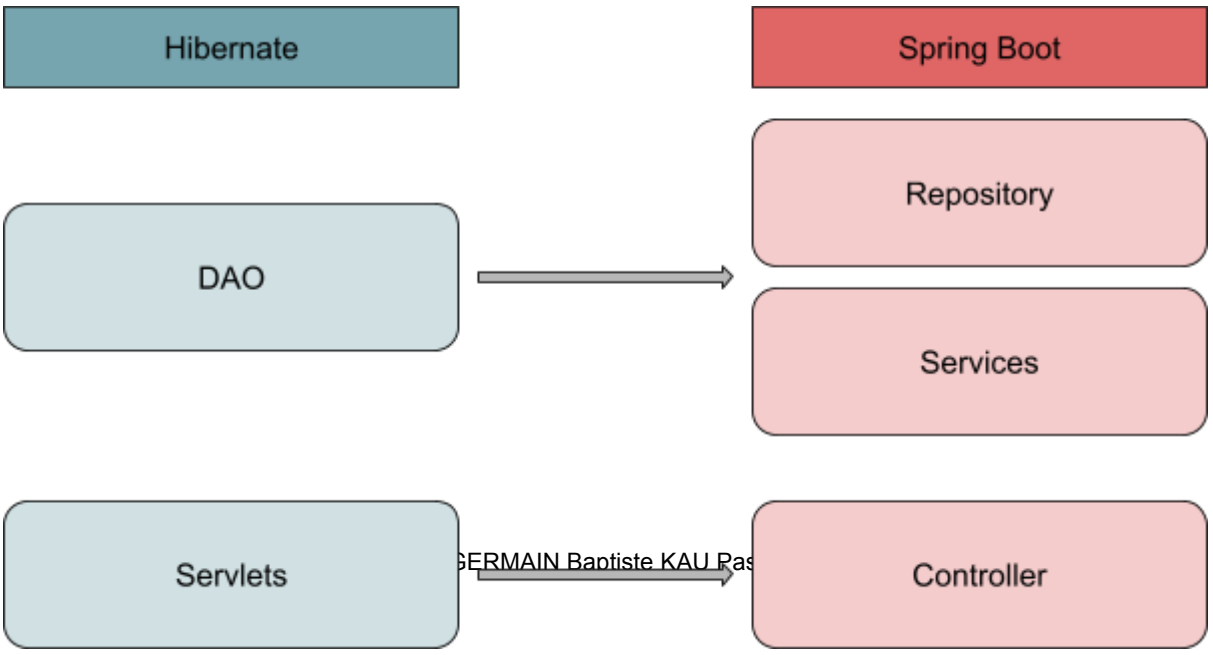
Application en utilisant Spring Boot

La migration vers le framework Spring Boot se fait suivant les étapes suivantes :

1. Il faut passer des DAO aux Repository Java qui vont tous hériter des classes JpaRepository :

Original DAO	New Repository
AdministratorDAO	AdministratorRepository
CoursDAO	CoursRepository
EnseignantDAO	EnseignantRepository
EtudiantDAO	EtudiantRepository
InscriptionDAO	InscriptionRepository
NoteDAO	NoteRepository

2. Le DAO Générique ainsi que le CRUD générique ne sont plus nécessaires.
3. Les entités JPA sont regroupées dans le package model, avec annotations pour les relations et contraintes.
4. La logique métier des précédents DAO sont maintenant relayés à des classes services réservées.
5. Enfin, il faut revoir les servlets et les adapter aux modifications précédentes.
6. Il faut également, un fichier application.properties pour les configurations de la relation avec les bases de données.



Le tout est lancé avec un fichier MainApplication.java qui a l'annotation @SpringBootApplication.

Le grand avantage de Spring Boot est qu'il élimine la complexité des configurations manuelles et fournit un framework robuste et évolutif.

Notons également que Spring Boot inclut un serveur Tomcat, Jetty ou Undertow intégré, éliminant la nécessité de déployer les applications sur des serveurs externes. Cela simplifie le développement, le test et le déploiement.

Le fait d'avoir des packages dédiés dao, model, service permet d'avoir une meilleur visibilité et surtout permet de décomposer les tâches.

Le diagramme de classe de la version Spring Boot est le suivant :