Université de CY Cergy Paris Université


# Projet Final JEE

Groupe :

LIMAM Mohamed Limam (GSI-1)

SEMGAT Ayoub (GSI-2)

FILALI Amine (GSI-2)

BELGHITHI Mohamed Amine (GSI-2)

THOYER Roman (GSI-2)

Professeur référent : HADDACHE Mohamed

Année Universitaire : 2024/2025

# I- Introduction

## 1) Contexte et Motivation

Le projet de fin de semestre constitue une étape importante dans le cursus du cycle ingénieur de la classe ING2-GSI. Cette étape permet d'acquérir non seulement une nouvelle expérience précieuse en gestion de projet, mais nous a aussi permis de coder dans un nouveau langage de programmation, alliant le Java et divers langages de programmation de création de site internet. Il nous a permis d'exercer notre autonomie, de relever de nouveaux défis, de trouver des solutions à divers problèmes, d'envisager plusieurs options pour chaque cas d'usage, et de gérer la cohésion de groupe. Toutes ces compétences sont essentielles dans la formation d'un futur ingénieur.

## 2) Description du projet

Ainsi, dans le cadre du module de développement web avancé, notre équipe à entrepris la réalisation d'une application web de gestion de scolarité. Ce projet s'inscrit dans un contexte académique visant à simplifier et à centraliser les processus liés à la gestion des étudiants, enseignants, cours, inscriptions, et résultats. Il a pour but de fournir une solution efficace et moderne répondant aux besoins spécifiques des utilisateurs finaux : les administrateurs, les enseignants et les étudiants.

L'objectif principal de ce projet est de développer une plateforme intuitive et robuste permettant d'automatiser des tâches essentielles telles que l'enregistrement et la mise à jour des données, la gestion des notes, ou encore la génération de relevés. Par ailleurs, l'application intègre des mécanismes d'authentification et de gestion des rôles pour garantir la sécurité et la confidentialité des données.

## 3) Technologies et outils utilisés

Pour répondre à ces exigences, nous avons utilisé des technologies actuelles et performantes, notamment les « Servlets » pour le backend, « Hibernate » pour la gestion des données, et une base de données relationnelles comme « MySQL ». L'architecture « MVC » (Modèle-Vue-Contrôleur) a été adoptée pour assurer une séparation claire des responsabilités et améliorer la maintenabilité de l'application.

Ce rapport détaille les différentes étapes de réalisation, de l'analyse des besoins à l'implémentation finale, en passant par la conception des modèles et la mise en œuvre des fonctionnalités principales. Nous présentons également les résultats obtenus et les perspectives d'amélioration pour une évolution future de l'application.

# II- Méthodologie de développement

## 1) Organisation

Durant la première partie du projet, l'organisation a été principalement effectuée de manière orale, basée sur une réunion quotidienne au cours de laquelle un point était fait sur le travail accompli et celui à venir. Par la suite, nous nous assignions chacun une tâche à réaliser, seul ou en groupe, en fonction de sa complexité et de sa durée. Nous restions ensuite tous connectés sur le même groupe Discord, ce qui nous permettais d'interagir directement avec tous les membres du groupe en cas de besoin. De plus, la possibilité de partager nos écrans et d'utiliser un dépôt de code communautaire sur IntelliJ par exemple, facilitait la mise en commun de nos travaux et l'assistance mutuelle en cas de nécessité. À la suite d'une réunion, nous avons amélioré notre système d'organisation en utilisant Google Agenda ainsi que Jira, ce qui nous a permis de mieux visualiser toutes les tâches en cours, celles déjà réalisées, ainsi que de savoir qui travaillait sur quelles tâches.

## 2) Analyse des besoins

L'analyse des besoins a permis de définir les fonctionnalités essentielles de l'application, en tenant compte des attentes des différents types d'utilisateurs. Ces utilisateurs sont :

- Administrateurs : ils ont un contrôle total sur la gestion des données, y compris la création et la mise à jour des informations des utilisateurs, des cours, et des inscriptions.
- Enseignants : leur rôle est centré sur l'attribution des cours, la saisie des notes et la consultation des résultats.
- Étudiants : ils peuvent consulter leurs inscriptions, leurs notes, et leurs relevés académiques.

Les fonctionnalités clé identifiées sont :

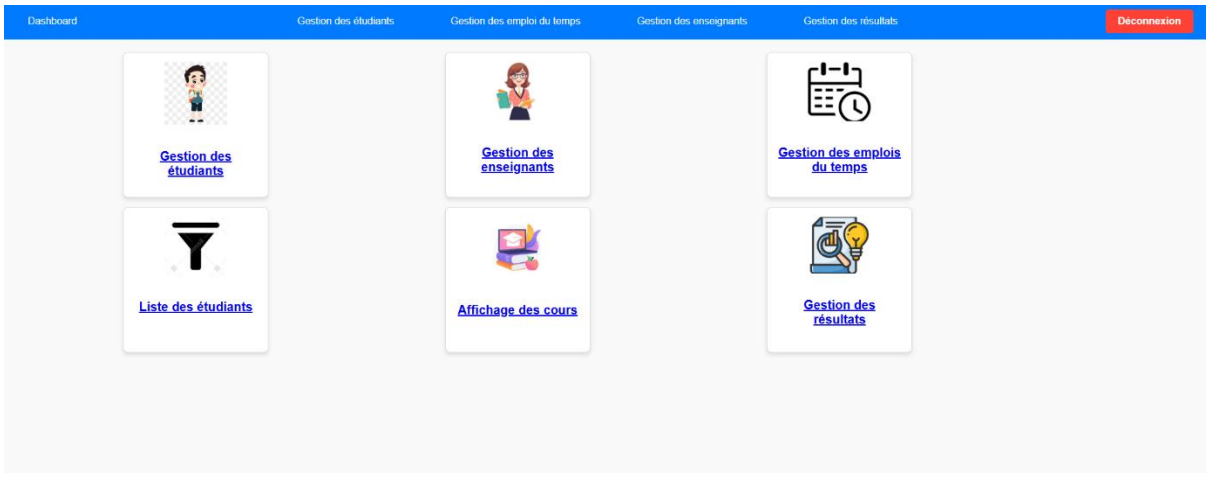
- Gestion des étudiants : ajout, mise à jour, suppression et affichage des informations.
- Gestion des enseignants et des cours : attribution et modification des cours.
- Gestion des inscriptions : association des étudiants aux cours.
- Gestion des résultats : saisie des notes par les enseignants, calcul des moyennes et génération de relevés.
- Authentification : sécurisation par rôles d'accès.

## 3) Choix des technologies

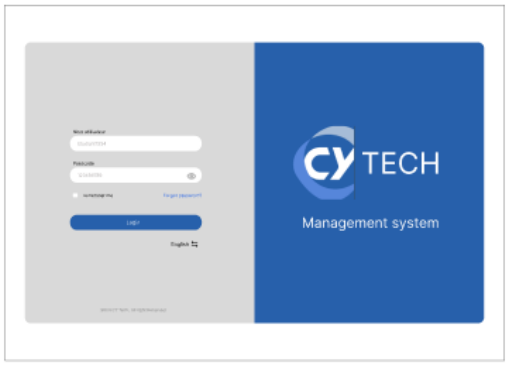
Suite à ce travail préliminaire, nous avons choisi (ou avons été imposé) les technologies suivantes pour répondre aux besoins identifiés :

- Backend : les Servlets Java ont été utilisées pour gérer les requêtes http envoyées par le client.
- ORM (Object-Relational Mapping) : Hibernate pour la gestion des données relationnelles.
- Base de données : MySQL pour sa robustesse et sa compatibilité avec les frameworks choisis.
- Frontend : JSP, HTML, CSS et JavaScript pour une interface utilisateur moderne et dynamique.
- Serveur web : Apache Tomcat pour le déploiement.

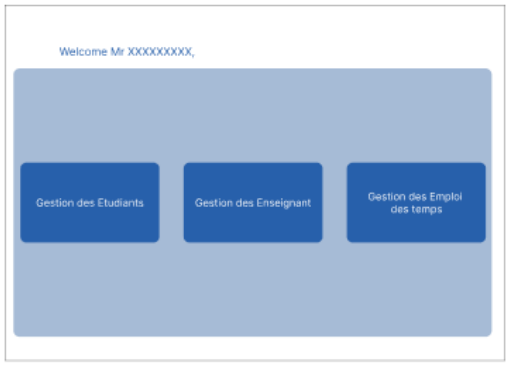
## 4) Exemple de travail visuel



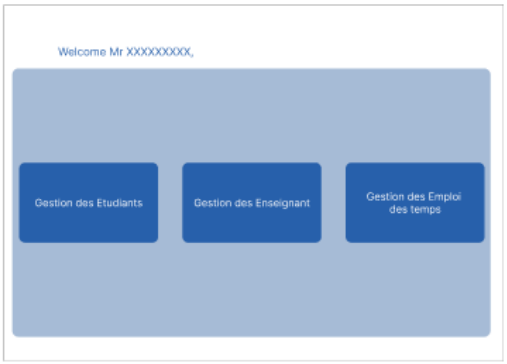
Acceuil




Admin Page : Dashboard




Admin Page : Gestion Etudiant




# III- Implémentation

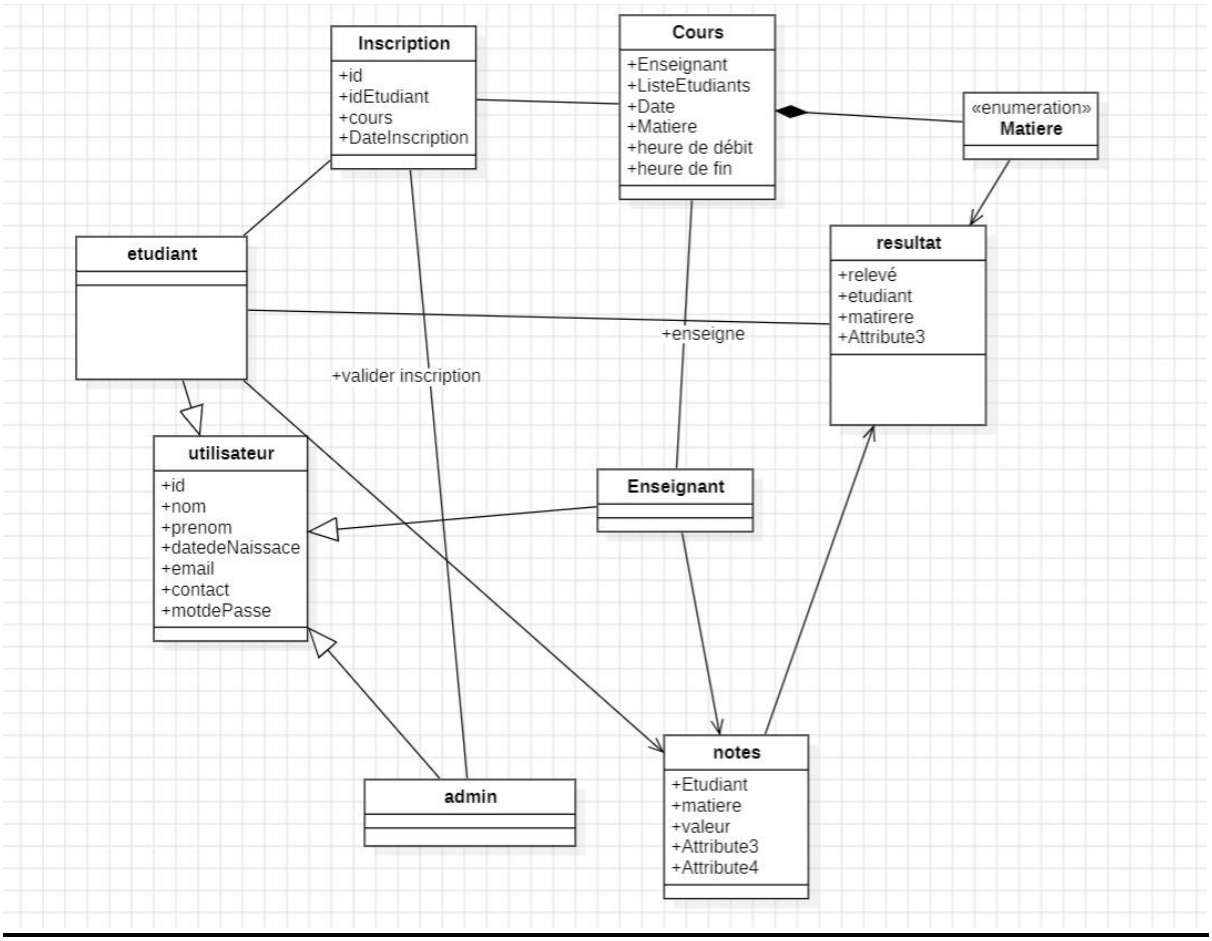
## 1) Architecture de l'application

L'application a été conçue en suivant le modèle MVC (Modèle-Vue-Contrôleur), qui sépare les différentes responsabilités de manière claire pour améliorer la maintenabilité et l'évolutivité du projet. Voici une description des trois couches principales :

- Modèle : gère la logique métier et les interactions avec la base de données. Par exemple, les entités telles que Etudiant, Cours et Inscription sont implémentées ici en tant que classe Java.
- Vue : représente l'interface utilisateur. Les pages JSP sont utilisées pour afficher les données dynamiques sous forme de HTML/CSS et JavaScript, permettant aux utilisateurs de naviguer et d'interagir avec l'application internet.

- Contrôleur : traite les requêtes http, gère la logique métier et orchestre les interactions entre le modèle et la vue. Les Servlets remplissent ce rôle.

## 2) Prototype de MLD




# IV- Difficultées rencontrées

## 1) Configuration des Mappings avec Hibernate

La configuration des mappings objet-relationnel avec Hibernate s'est avérée complexe, en particulier pour définir les relations entre les entités. Des erreurs comme « LazyInitializationException » ou des incohérences dans les données ont également été rencontrées. Ainsi, nous avons porté une attention particulière à la configuration des annotations pour les relations. Nous avons activé le chargement « eager » pour certaines relations critiques afin de limiter les problèmes liés à l'accès différé des

données. Enfin, les transactions ont été gérées avec soin pour éviter des incohérences dans la base. Cela impliquait de s'assurer que chaque opération impliquait un « commit() » approprié.

## 2) Difficultés avec les fichiers JSP

Les fichiers JSP, utilisés pour générer les pages web, ont parfois rendu difficile la séparation entre la logique de programmation et l'affichage. Cela a conduit à des problèmes tels que des erreurs inattendues dans les pages ou des difficultés à afficher correctement les données.

Nous avons donc amélioré notre manière de travailler avec les JSP en réduisant la complexité dans ces fichiers. Nous avons mieux structuré notre code pour que les données soient préparées avant d'être envoyées à l'affichage. Pour faciliter la gestion des données dynamiques, nous avons utilisé des outils standard qui simplifient l'insertion des informations dans les pages sans encombrer le code.

## 3) Absence de Spring Boot

Nous n'avons pas réussi à utiliser Spring Boot pour le projet, ce qui a rendu certaines tâches plus complexes, comme la gestion des configurations et des dépendances. Cela a également exigé de nombreuses opérations manuelles pour structurer l'application.

Nous avons donc opté pour une approche classique en utilisant des servlets, JSP, et Hibernate. Bien que cela demande plus de configuration manuelle, cela nous a permis de maintenir une architecture claire et d'apprendre à organiser le projet sans l'aide d'un framework moderne. Nous avons tiré parti des outils standards pour structurer l'application et gérer les interactions entre les différentes parties.

## 4) Résumé des problèmes

Bien que l'absence de Spring Boot ait ralenti le développement, cela nous a permis de mieux comprendre les bases des servlets, JSP, et Hibernate. Chaque défi technique a renforcé notre compréhension des outils utilisés, tout en nous poussant à optimiser nos configurations et notre gestion des données. Ces

problèmes ont été une opportunité d'apprentissage et d'amélioration pour notre équipe.

# V- Conclusion et Améliorations

## 1) Conclusion

La réalisation de ce projet de gestion de scolarité a représenté un défi stimulant et formateur. Malgré les contraintes techniques et les imprévus, notamment l'impossibilité d'utiliser Spring Boot, nous avons su adapter notre approche en exploitant les technologies classiques comme les Servlets, JSP et Hibernate. Ce choix nous a permis de développer une application fonctionnelle respectant les exigences de base et répondant aux besoins des utilisateurs finaux :

- La conception et la modélisation des bases de données.
- La structuration d'une application web en respectant l'architecture MVC.
- L'utilisation des ORM pour simplifier les interactions entre les objets Java et la base de données.
- La gestion de la sécurité et des autorisations dans un contexte multi-utilisateur.

Bien que le projet ne soit pas dispensé de limitations, il représente une base solide sur laquelle des améliorations futures peuvent être apportées.

## 2) Améliorations

Pour l'avenir, plusieurs améliorations et évolutions peuvent être envisagées pour perfectionner l'application.

La première serait d'intégrer Spring Boot, ce qui permettrait d'automatiser une grande partie des configurations et de simplifier le développement, notamment en introduisant des fonctionnalités modernes et une meilleure gestion des dépendances.

La seconde serait de remplacer les pages JSP par des « frameworks frontend » modernes comme React ou Angular, ce qui offrirait une expérience utilisateur plus dynamique et fluide.

De plus, il serait toujours possible de mettre en œuvre des optimisations comme la pagination pour les listes volumineuses d'étudiants ou de cours par exemple, ou la mise en cache des données souvent consultées.

Enfin, il serait aussi possible d'ajouter de nouvelles fonctionnalités, comme l'ajout d'un tableau de bord pour les administrateurs, ou la mise en place de système de message (même si cela sors, dans ce cadre, du sujet du projet).

La réalisation de ce projet a présenté non seulement un intérêt pédagogique, mais il a aussi permis de développer des compétences techniques variées, allant de la programmation à la conception d'interfaces utilisateur, en passant par la gestion de bases de données et l'intégration de divers langages de programmation. Ce projet a ainsi été une excellente opportunité de mettre en pratique nos connaissances et de les enrichir dans un contexte concret et stimulant. Les compétences acquises et les leçons tirées de cette expérience constituent un atout précieux pour nos futurs projets, que ce soit dans un cadre académique ou professionnel. Nous sommes confiants que, avec de telles bases solides, nous pourrons relever des défis plus complexes à l'avenir.

# VI- Bibliographie

## 1) Documentation officielle

- Hibernate : https://hibernate.org/
- Servlets et JSP : https://docs.oracle.com/javaee/
- Apache Tomcat : https://tomcat.apache.org/

## 2) Tutoriels et ressources en ligne

- Baeldung, *Introduction to Hibernate* : https://www.baeldung.com/hibernate
- GeeksforGeeks, *Servlet Basics* : https://geeksforgeeks.org/servlets/
- Marco Codes Playlist : https://www.youtube.com/playlist?list=PLIRBol92yMam1HaUYrMAaPdbZKV1BFW0F
- Oracles : https://www.oracle.com/java/technologies/jee-tutorials.html

## 3) Outils utilisés

- IntelliJ IDEA : https://www.jetbrains.com/idea/
- MySQL Workbench : https://dev.mysql.com/workbench/
- Eclipse : https://eclipseide.org/