RAPPORT

Conception et implémentation de l'application

PROJET JEE : Application de Gestion de Scolarité

HAUTECOURT Julien
MAESTRI Adrien
KENMOGNE Loïc
RIVIERRE Hugo
ELISABETH Louis
HADDACHE Mohamed

Table des Matières

1. **Introduction**

2. **Conception**

    1. Analyse des besoins

        1. Besoins fonctionnels
        2. Besoins non fonctionnels

    2. Modèle Conceptuel des Données (MCD)

3. **Implémentation**

    1. Technologies utilisées

    2. Architecture du projet

        1. Modèle
        2. Vue
        3. Controlleurs

1. Introduction

Le projet a pour objectif de développer un système de gestion de scolarité simplifiant la gestion des cours, des emplois du temps et des résultats. Les utilisateurs incluent des administrateurs, des enseignants, et des étudiants, chacun ayant des fonctionnalités spécifiques.

Le système repose sur une base de données relationnelle structurée et une interface utilisateur intuitive permettant une interaction fluide.

2. Conception

2.1 Analyse des Besoins :

2.1.1 Besoins fonctionnels ;

Les besoins fonctionnels décrivent les services que l'application doit fournir aux utilisateurs :

- Pour les administrateurs :

- Gérer les utilisateurs (ajouter, modifier ou supprimer les étudiants, enseignants et autres administrateurs).
- Planifier les cours en associant un enseignant, une salle et un horaire spécifiques.
- Consulter et modifier les emplois du temps globaux ou individuels.

- **Pour les enseignants :**

- Consulter leur emploi du temps personnalisé.
- Accéder aux informations relatives aux cours attribués, notamment les horaires, les salles et les étudiants inscrits.
- Éventuellement, consulter et gérer les notes des étudiants.

- **Pour les étudiants :**

- Consulter leur emploi du temps et les détails des cours auxquels ils sont inscrits.
- Vérifier les informations relatives aux salles et aux horaires.
- Accéder à leurs notes et résultats.

2.1.2 Besoins non fonctionnels :

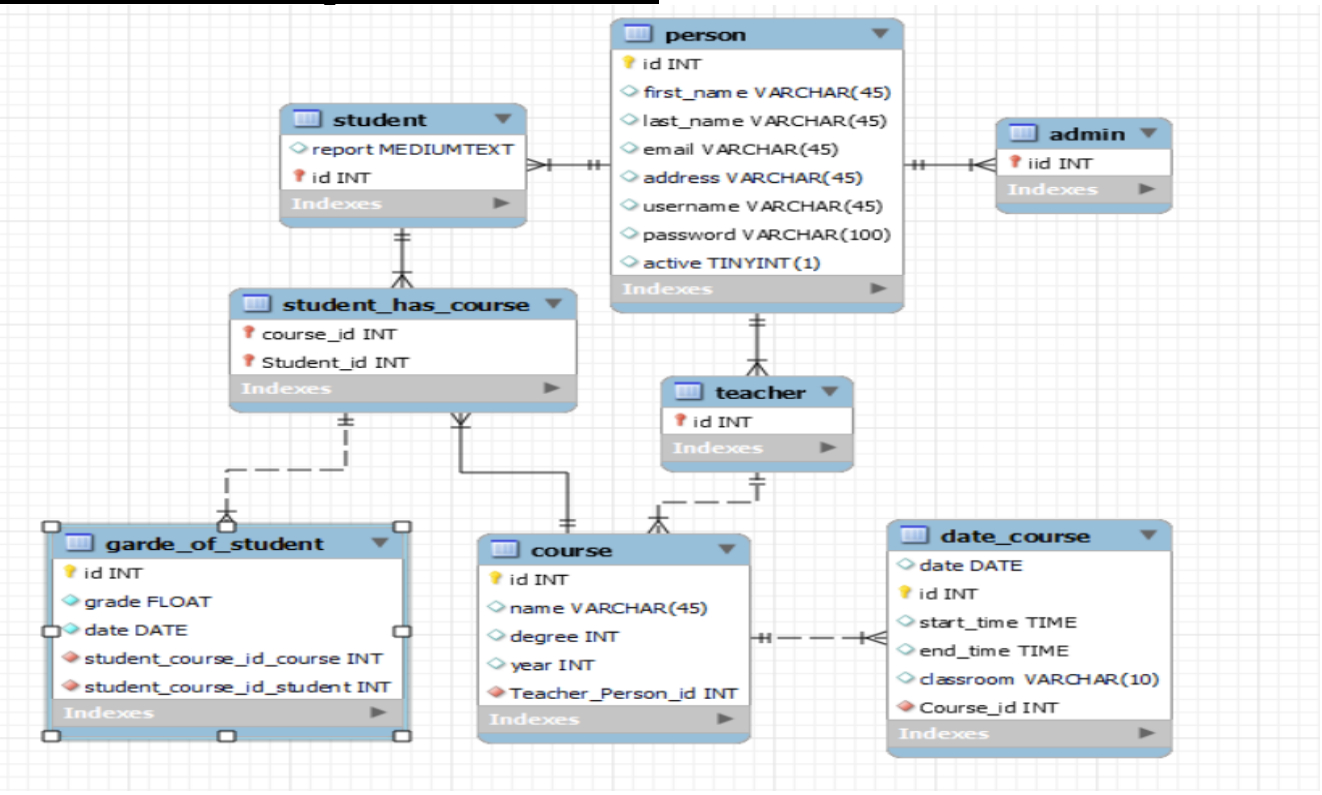
Les besoins non fonctionnels concernent la qualité et les contraintes techniques du système :

- **Facilité d'utilisation** : Interface claire et intuitive pour tous les profils d'utilisateurs.

- **Sécurité** : Authentification obligatoire pour tous les utilisateurs avec des rôles définis (administrateur, enseignant, étudiant).

- **Scalabilité** : Capacité à gérer un grand nombre d'utilisateurs, cours et données associées.

2.2 Modèle Conceptuel de Données :



Explications :

Entités principales et leurs attributs :

1. **person :**
   - Représente une personne générale dans le système.
   - Attributs :
     - id : Identifiant unique.
     - first_name et last_name : Prénom et nom de la personne.
     - email : Adresse email.
     - address : Adresse physique.
     - username et password : Informations de connexion.
     - active : Statut de la personne (actif/inactif).

2. **admin :**
   - Hérite de l'entité person (relation entre person et admin).

- Identifie les administrateurs du système.

3. **teacher :**
   - Hérite également de l'entité person.
   - Représente les enseignants qui peuvent être liés à des cours.

4. **student :**
   - Hérite de l'entité person.
   - Représente les étudiants inscrits dans le système.
   - Attribut supplémentaire :
     - report : Un champ texte permettant de stocker un rapport ou des remarques pour l'étudiant.

5. **course :**
   - Représente un cours enseigné dans le système.
   - Attributs :
     - id : Identifiant unique du cours.
     - name : Nom du cours.
     - degree : Niveau ou diplôme associé au cours.
     - year : Année du cours.
     - Teacher_Person_id : Identifie l'enseignant responsable du cours.

6. **date_course :**
   - Représente les plages horaires d'un cours.
   - Attributs :
     - id : Identifiant unique.
     - date : Date de la séance.
     - start_time et end_time : Heures de début et de fin.
     - classroom : Salle de cours.
     - Course_id : Référence au cours correspondant.

7. **student_has_course :**
   - Table associative entre student et course.
   - Attributs :
     - Student_id : Référence à l'étudiant.
     - course_id : Référence au cours.

8. **garde_of_student :**
   - Représente les notes obtenues par les étudiants pour un cours donné.

- Attributs :
  - id : Identifiant unique.
  - grade : Note obtenue.
  - date : Date de la note.
  - student_course_id_course et student_course_id_student : Références combinées à un étudiant et un cours.

3. Implémentation

3.1. Technologies Utilisées

- **Langages :** Java, SQL, HTML, CSS, JavaScript.
- **Frameworks :** Jakarta EE
- **Base de données :** MySQL
- **Serveur :** Apache Tomcat.

3.2. Architecture du Projet

Le projet suit l'architecture MVC (Model-View-Controller) :

3.2.1 Modèle (Model) :

Le projet contient un package «models » où sont stockés toutes les classes qui définissent nos entités. On y retrouve :

Admin : Cette classe représente un administrateur dans le système. Elle hérite toutes les propriétés générales d'une personne (définies dans Person) mais peut également être étendue pour inclure des fonctionnalités spécifiques à un administrateur.

Course : La classe Course représente une entité de cours avec des propriétés et des méthodes pour manipuler ces informations. Elle contient des informations sur :

- L'identifiant du cours (id).
- Le nom du cours (name).
- Le diplôme ou degré associé (degree).
- L'année d'étude associée (year).

- L'identifiant de l'enseignant responsable (teacherId).
- Le nom de la salle de classe (classroom).
- Le nom de l'enseignant (teacherName), récupéré à partir d'une base de données

DateCourse : La classe DateCourse modélise une session d'un cours, en incluant des informations essentielles comme :

- L'identifiant unique de la session (id).
- L'identifiant du cours associé (courseId).
- La date de la session (date).
- La salle de classe où la session a lieu (classroom).
- Les heures de début et de fin de la session (startTime et endTime).

DateCourseDetail : Cette classe fournit un constructeur pour initialiser ces champs et des getters pour accéder à leurs valeurs. Elle est utile pour transmettre ou afficher des informations complètes sur un cours planifié.

Grade : La classe Grade sert à encapsuler les informations liées à une note, en particulier :

- La valeur de la note (grade, de type double).
- La date associée à cette note (date, de type String).

Person : La classe Person sert de classe de base pour toutes les entités qui partagent des propriétés communes d'une personne, comme Admin, Teacher, ou Student. Ces classes peuvent hériter de Person et éventuellement ajouter leurs propres propriétés ou comportements spécifiques.

Student : La classe Student est une spécialisation de la classe Person. Elle représente une entité étudiant en ajoutant des fonctionnalités supplémentaires (ici, un attribut report) tout en héritant des caractéristiques générales d'une personne.

StudentCourse : La classe StudentCourse modélise la relation entre un étudiant et un cours, en ajoutant des détails spécifiques à cette relation :

1. Les identifiants de l'étudiant et du cours concernés.
2. Une liste de notes (grades) obtenues par l'étudiant dans ce cours.
3. Une propriété optionnelle (degree) pour indiquer la moyenne ou le niveau atteint dans ce cours.

Teacher : La classe Teacher est une sous-classe de Person. Elle utilise l'héritage pour réutiliser les attributs et les méthodes de Person.

3.2.2 Vue (View) :

Dans le package “webapp”, se trouvent les fichiers JSP pour générer les pages HTML nécessaires :

admin.jsp : Sert à afficher l’interface d'administration permettant de gérer les utilisateurs (élèves et professeurs), avec des options d'ajout, de filtrage et de déconnexion.

studentForm.jsp : Permet d’afficher le formulaire pour créer ou modifier un étudiant, en fonction de la présence d'un identifiant. Inclut également des mécanismes pour afficher des messages d'erreur ou de succès et permet de revenir à la page d'administration.

teacherForm.jsp : Permet soit de créer soit de modifier un professeur en fonction de l'existence ou non d'un id passé dans la requête. Inclut également des mécanismes pour afficher des messages d'erreur ou de succès et permet de revenir à la page d'administration.

login.jsp : Sert à afficher le formulaire de connexion. Elle vérifie d'abord si l'utilisateur est déjà connecté et, si c'est le cas, le redirige vers la page appropriée (administration ou accueil). Si l'utilisateur n'est pas connecté, il peut saisir ses informations de connexion, et un message d'erreur sera affiché si nécessaire.

courseCalendar.jsp : Sert à afficher un tableau de bord personnalisé pour un étudiant ou un professeur. Elle ajuste son apparence et ses options de navigation en fonction du rôle de l'utilisateur, et elle affiche un calendrier de l'emploi du temps avec les cours et leurs détails.

courseServlet.jsp : Permet aux utilisateurs de charger la liste complète des cours, rechercher des détails d'un cours via son ID et de Créer de nouveaux cours en envoyant un formulaire avec des informations nécessaires.

CourseList.jsp : Permet à un enseignant de sélectionner un cours parmi une liste pour consulter les notes associées.

Enrollment.jsp : Permet à l'utilisateur de s'inscrire à un cours de manière simple et directe, tout en affichant des messages de succès ou d'erreur selon le résultat de l'inscription. Le côté client est géré avec JavaScript pour interagir dynamiquement avec le backend (par exemple, pour remplir la liste des cours), et le formulaire est soumis à un servlet ou un contrôleur qui gère l'inscription proprement dite.

enrollmentSuccess.jsp : Affiche un message de confirmation après qu'un utilisateur (étudiant) se soit inscrit avec succès à un cours. Il indique que l'inscription a été réussie et fournit un bouton pour revenir à la page d'accueil.

gradeEntry.jsp : Permet à un enseignant d'enregistrer les notes des étudiants pour un cours spécifique.

gradeEntry2.jsp : Permet à un enseignant d'assigner une note à un étudiant inscrit à un cours.

gradeList.jsp : Affiche la liste des notes attribuées à un étudiant dans un cours spécifique.

StudentDetails.jsp : Affiche les informations détaillées d'un étudiant spécifique. Elle est conçue pour présenter des données récupérées via un contrôleur et transmises comme attributs à la page.

studentList.jsp : Permet de rechercher des étudiants en fonction de leur nom et prénom, et d'afficher les résultats sous forme de tableau avec un bouton pour consulter les détails de chaque étudiant.

teacherDetails.jsp : Affiche les détails d'un enseignant.

teacherList.jsp : Implémente une fonctionnalité de recherche et d'affichage des enseignants en fonction des critères fournis par l'utilisateur.

home.jsp : Affiche la page d'accueil personnalisée pour un utilisateur en fonction de son rôle (étudiant ou professeur), avec des options adaptées à chaque rôle, telles que consulter les notes ou gérer des cours.

A cela viennent s'ajouter des fichiers CSS et JS qui permettent de styliser les pages et de pouvoir interagir.

3.2.3 Contrôleur (Controller) :

Enfin, dans un package "controllers", on retrouve les servlets qui permettent le bon fonctionnement de l'application :

AdminServlet : Le Servlet AdminServlet permet à un administrateur de créer, modifier, supprimer et récupérer des informations sur les utilisateurs ayant le rôle de student ou teacher. Le Servlet interagit avec une base de données via la classe DbConnect pour obtenir ou manipuler les informations des utilisateurs.

AuthServlet : Le servlet AuthServlet gère l'authentification des utilisateurs d'une application web. Il vérifie les identifiants (nom d'utilisateur et mot de passe) fournis par un utilisateur via une requête POST. En cas de succès, il crée une session et redirige l'utilisateur vers la page correspondante à son rôle (admin ou utilisateur standard). Sinon, il retourne un message d'erreur à la page de connexion.

CalendarServlet : Le servlet CalendarServlet est conçu pour afficher le calendrier des cours d'un enseignant. Il récupère les informations des cours associés à un enseignant à partir d'une base de données et les transmet à une page JSP pour les afficher.

CourseServlet : Le servlet CourseServlet gère la création et la récupération des cours au format JSON via des requêtes HTTP.

EnrollementServlet : Ce servlet gère les inscriptions des étudiants à des cours et récupère les informations des cours et enseignants.

GradeCourseListServlet : Ce servlet gère les interactions entre les cours et les notes d'un étudiant.

GradeServlet : Ce servlet est destiné à gérer la récupération et la gestion des cours associés à un enseignant.

LogoutServlet : Ce servlet sert à gérer la déconnexion d'un utilisateur en invalidant la session et en redirigeant l'utilisateur vers la page de connexion.

StudentListServlet : Ce servlet gère les requêtes pour afficher une liste d'étudiants et les détails d'un étudiant spécifique.

StudentServlet : Ce servlet a pour but de traiter la création et la mise à jour des étudiants via un formulaire soumis par un administrateur.

TeacherListServlet : Ce servlet gère les opérations liées à la recherche et à l'affichage des enseignants.

TeacherServlet : Ce servlet permet de gérer les interactions entre l'utilisateur et les opérations liées aux enseignants.