

# RAPPORT DU PROJET

## JAVA

**Réaliser par :**

- ✚ EL MOSTAFA FADILI
- ✚ MOAD EL KHAYATY
- ✚ OUSSAMA ABOUZID
- ✚ BEL MEKNASSI IBRAHIM

Encadré par:

Mr IMAD HAFIDI

# SOMMAIRE

<b>1-Introduction de la problématique.....</b>	<b>2</b>
<b>2-Cahier de charge.....</b>	<b>3</b>
<b>3-Analyse et conception.....</b>	<b>4</b>
a-Dictionnaire de donnée.....	4
b-Modèle conceptuel des données.....	5
c-Modèle logique des données.....	6
<b>3-Réalisation.....</b>	<b>7</b>
a-Couche persistance.....	7
b-Couche Métier.....	7
c-Couche Présentation.....	8
<b>4-Test et validation.....</b>	<b>19</b>
a-Les tests unitaires.....	19
b-Les tests globaux.....	19
<b>5-La maintenance.....</b>	<b>20</b>
<b>6-Conclusion .....</b>	<b>20</b>
Annexe.....	21

## INTRODUCTION DE LA PROBLMATIQUE:

Avant l'informatisation des écoles, toutes les tâches étaient réalisées manuellement, et indépendamment les unes des autres. Les administrateurs faire inscrire des professeurs et des étudiants afin de rédiger des formulaires étaient signalés sur un registre d'inventaire manuscrit, qui ne quittait plus le meuble destiné à la conserver.

L'objectif de ce projet est de réaliser une application pour la gestion des projets en utilisant JAVA. Cette application offre un nombre de fonctionnalités de base à savoir la consultation, l'ajout, la modification et la suppression d'un professeur ou étudiant ou bien des rapports projets documents et des rendez .

### a. Critique de l'existant

L'analyse de l'existant met l'accent sur plusieurs difficultés telles que :

- Le travail de certaines écoles se fait encore manuellement.
- Perte de temps
- Manque d'organisation du travail dans l'écoles.
- Volume important des informations traitées manuellement, ce qui provoque parfois des erreurs dans l'établissement des documents.
- Recherche difficile sur les registres.
- Insécurité des informations.
- Possibilité d'erreur dans le remplissage des différents documents et registres.
- Nombre important des archives qui engendre une difficulté de stockage.
- Détérioration des archives à force de leur utilisation trop fréquente.
- Mauvaise codification sur quelques objets dans la gestion d'information.

### b. Solutions :

Afin de corriger les problèmes présentés ci-dessus, Nous sommes appelées à réaliser une application qui assure les points suivants :

- Automatiser les tâches qui se traitent manuellement.
- Faciliter la recherche et l'accès aux informations.
- Sauvegarder toutes les données relatives à la gestion des projets sur des supports informatiques ce qui assurera leur sécurité.
- Minimiser les supports papiers utilisés.
- Faire toute modification (ajout, suppression, modification) automatiquement.
- Plus d'organisation dans le travail du l'écoles.
- Faciliter la recherche de l'information.
- Rapidité dans l'établissement des différents documents.

## Cahier de charge :

Ce projet consiste à développer une application de gestion des projets académiques. Les projets académiques sont de trois types : -Projet de Fin année ( durée 2 mois ) - Projet Fin d'étude ( durée 5 à 6mois ) – Doctorat ( 3 à 6 ans).

Gérer les utilisateurs ( projets ,étudiants et professeurs) :

- Ajout, modification, suppression et Edition des professeurs
- Ajout, modification, suppression et édition ( par groupe ou par personne) des étudiants ( les étudiants du 4 et 5 année peuvent être ajoutés par un fichier excel)
- Création, affectation (étudiant et professeur), modification, édition des projets
- Front office : Partie de gestion des projets et partage de documentation
- Le professeur définit les étapes et livrables de chaque projet, valide l'avancement du projet et sa cloture, définit les rendez et valide leurs rapports
- L'étudiant consulte ses rendez vous, dépose les livrables ainsi que les rapports
- Le professeur et les étudiants ont le droit de déposer des documents (chargement, mots clés...). Chaque document dépend du projet et la recherche s'effectue par des mots clés. L'étudiant à le droit de consulter que les documents de son projet.
- Reporting : Partie des statistiques
- Le professeur consulte les statistiques d'avancement de chaque projet ainsi que tous les projets ou par type de projet.
- L'administrateur peut consulter en plus les statistiques d'avancement de tous les projets ou par type de projet

## Analyse et conception

### Description de la méthode de conception utilisée

Dans le cadre de ce projet nous avons utilisé la méthode **MERISE** (Méthode d'Etude de Réalisation Informatique pour les Systèmes d'Entreprise). Le but de cette méthode est d'arriver à concevoir un système d'information.

Cette dernière est basée sur la séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et physiques, vu qu'elle assure une longévité au modèle.

En effet, l'agencement des données n'a pas à être souvent remanié, tandis que les traitements le sont plus fréquemment.

La méthode MERISE s'appuie sur 2 points principaux :

- **Le cycle de vie** (très variable selon les projets)
  - Gestation et Conception
  - Réalisation et Exploitation
  - Maintenance (évolution, adaptation, mort)
- **Le cycle de spécification** (ou d'abstraction) du système d'information (SI)
  - Domaine des données : la mémorisation de l'information
  - Domaine des traitements : les processus de traitement de l'information

Merise étant une méthode de conception et de développement de système d'information, l'objectif de ce projet donc est de se familiariser avec cette méthode et connaître la notion de système d'information et d'en avoir une description formelle.

### 1-Dictionnaire De Donnée :

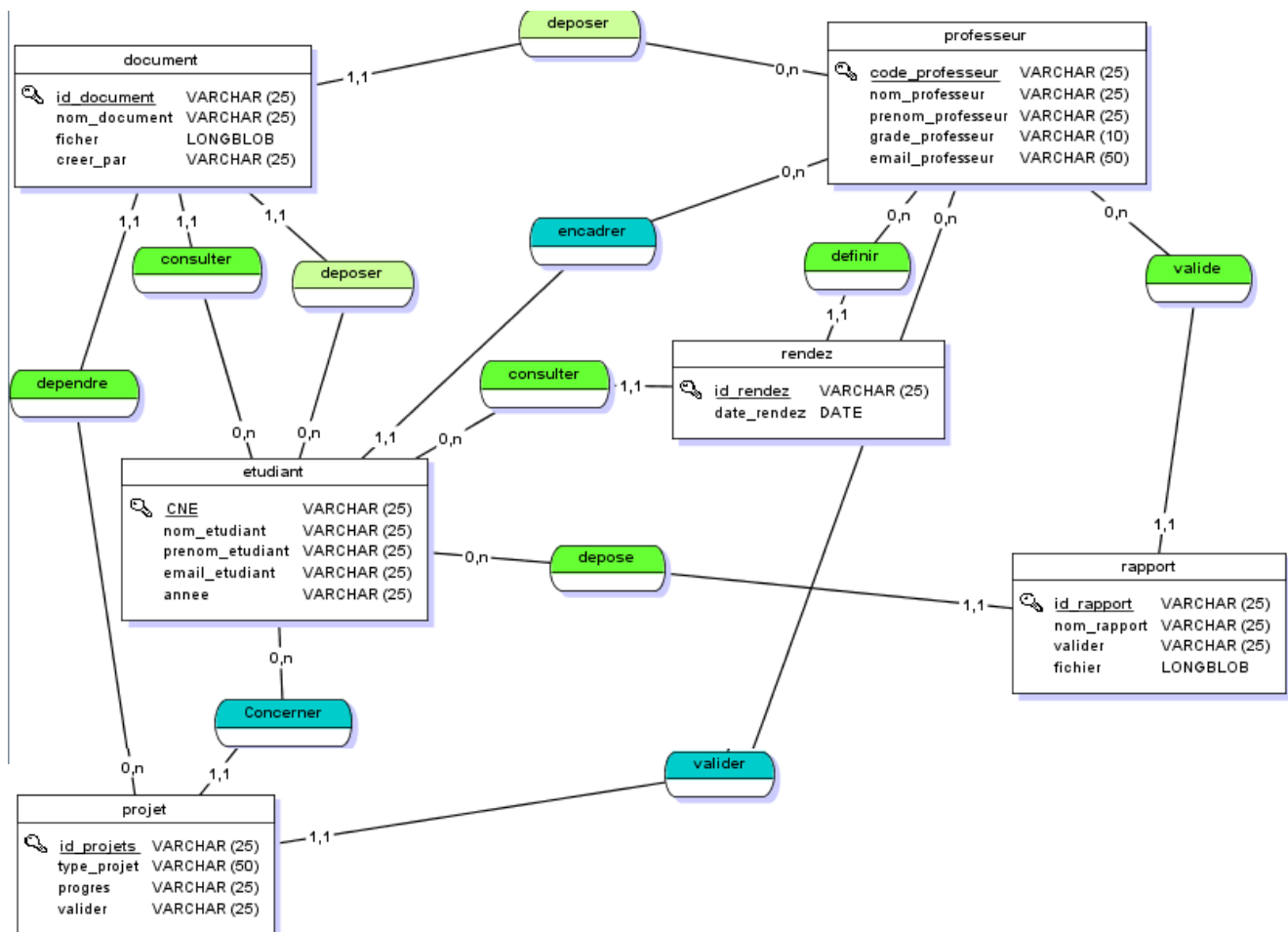
Nom	Code	type	taille
CNE	CNE	Varchar	25
nom_etudiant	NOM_ETUDIANT	Varchar	25
prenom_etudiant	PRENOM_ETUDIANT	Varchar	25
email_etudiant	EMAIL_ETUDIANT	Varchar	25
annee	ANNEE	Varchar	25
Id_projets	ID_PROJETS	Varchar	25
id_document	ID_DOCUMENT	Varchar	25
id_rapport	ID_RAPPORT	Varchar	25
code_professeur	CODE	Varchar	25
date_rendez	DATE_RENDEZ	Date	
nom_document	NOM_DOCUMENT	Varchar	25
nom_rapport	NOM_RAPPORT	Varchar	25
nom_professeur	NOM_PROFESSEUR	Varchar	25
prenom_professeur	PRENOM_PROFESSEUR	Varchar	25
grade_professeur	GRADE_PROFESSEUR	Varchar	10
email_professeur	EMAIL_PROFESSEUR	Varchar	50
type_projet	TYPE_PROJET	Varchar	50
valider	VALIDER	Varchar	25
creer_par	CREER_PAR	Varchar	25
ficher	FICHER	Longblob	
fichier	FICHER	Longblob	
progres	PROGRES_PROJET	Varchar	25
id_rendez	ID_RENDEZ	Varchar	25
id_rapport	ID_RAPPORT	Int	25
id_document	ID_DOCUMENT	Int	25
id_projets	ID_PROJETS	Int	25

## 2- Modèle conceptuel des données

Le modèle conceptuel des données (MCD) a pour but de représenter de façon structurée les données qui seront utilisées par le système d'information. En effet, il décrit la sémantique c'est à dire le sens attaché à ces données et à leurs rapports et non à l'utilisation qui peut en être faite.

On établit le MCD après avoir recensé et donné un nom à l'ensemble des données du domaine étudié. Ensuite on étudie les relations existantes entre ces données (les dépendances fonctionnelles), pour aboutir au MCD.

En ce qui concerne notre étude on établit le MCD suivant :



### 3- Modèle logique des données

La modélisation logique des données conduit aux opérations suivantes:

La transformation du MCD, en un MLD (Modèle Logique des données) exprimé dans un formalisme logique adapté au SGBD envisagé optimisation générale (notamment du coût induit par le mode de gestion) Le MLD sera ensuite transformé et adapté en fonction des spécificités du langage de définition des données On dit qu'une association binaire (entre deux entités) est de type :

- 1 : 1 (un à un) si aucune des 2 cardinalités maximales n'est n
- 1 : n (un à plusieurs) si une des 2 cardinalités maximales est n
- n : m (plusieurs à plusieurs) si les 2 cardinalités maximales sont n

Pour passer d'un modèle conceptuel de données MCD à un modèle logique des données MLD, on applique les règles suivantes :

*Règle 1 : Toute entité devient une table dans laquelle les attributs deviennent les colonnes.*

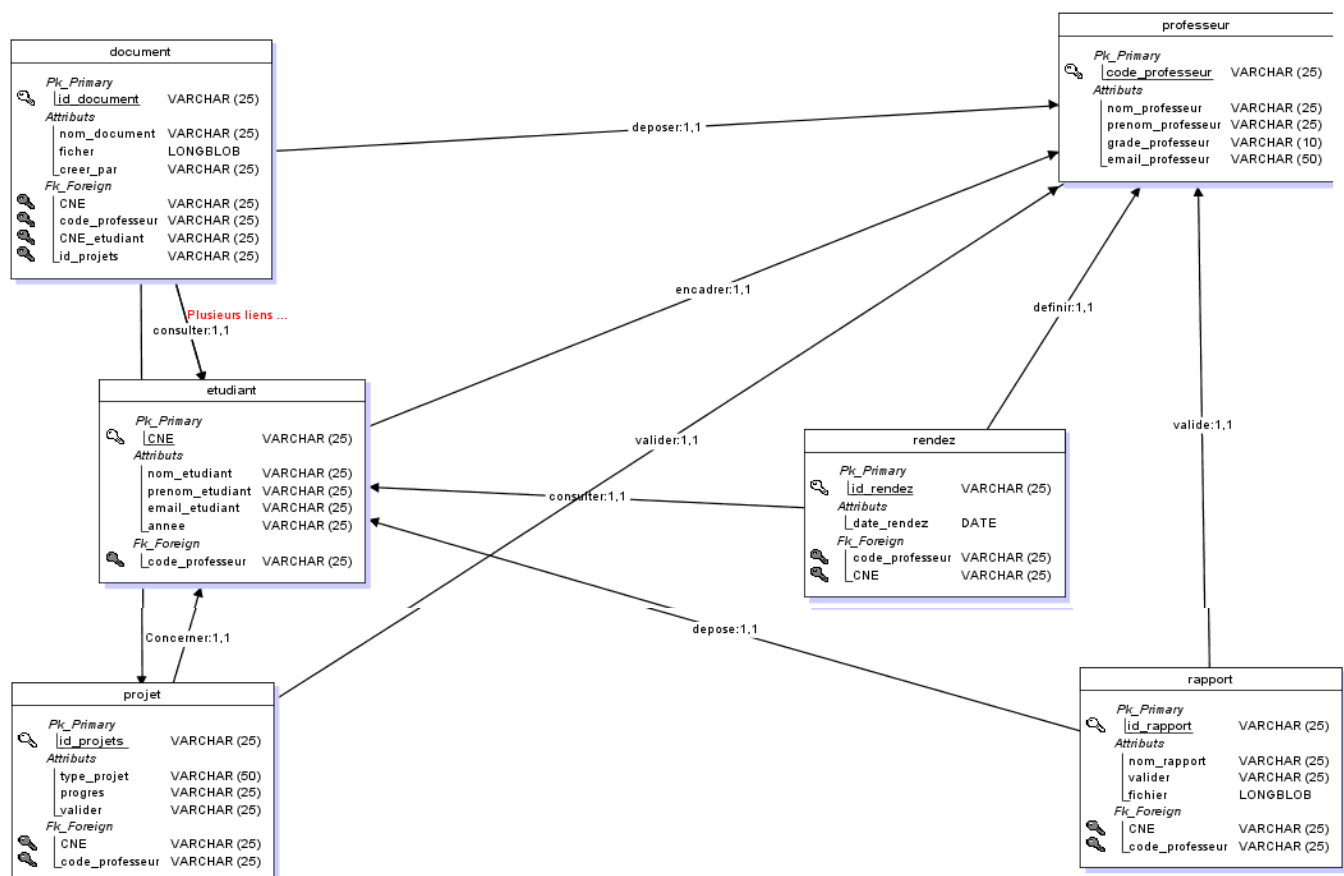
*L'identifiant de l'entité constitue alors la clé primaire de la table.*

*Règle 2 : Une association binaire de type 1 : n disparaît, au profit d'une clé étrangère dans la table coté 0,1 ou 1,1 qui référence la clé primaire de l'autre table. Cette clé étrangère ne peut pas recevoir la valeur vide si la cardinalité est 1,1*

*Règle 3 : Une association binaire de type n : m devient une table supplémentaire (table de jonction) dont la clé primaire est composée des deux clés étrangères.*

*Règle 4 : Une association binaire de type 1 : 1 est traduite comme une association binaire de type 1:n sauf que la clé étrangère se voit imposer une contrainte d'unicité en plus d'une éventuelle contrainte de non vacuité (cette contrainte d'unicité impose à la colonne correspondante de ne prendre que des valeurs distinctes).*

En appliquant les règles ci-dessus et A l'aide de JMERIS , On retrouve le modèle logique des données donné dans la page suivant





## REALISATION :


Nous proposons de réaliser une application en trois couches :



### Couche persistance :






Cette couche est responsable de la communication avec la base des données.

On a créé des classes en détailles (attributs et méthodes) de la couche persistance en respectant les règles suivantes :

-  **persistance.DAO**
  -  **ConnectionBD.java**

Chaque table dans la base de données est représentée par une classe DAO
  -  **DAODocument.java**



Chaque table DAO offre les services CRUD (ajouter, rechercher, modifier, supprimer) au minimum de la table qu'elle représente.
  -  **DAOEtudiant.java**
  -  **DAOInterface.java**


On Ajoute une classe qui est responsable de la connexion avec la BD avec le pattern singleton
  -  **DAOProfesseur.java**
  -  **DAOProjet.java**
  -  **DAORapport.java**
  -  **DAORendez.java**
  -  **DAOUser.java**



### Couche Métier :




Cette couche est responsable du logique métier et communique avec la couche persistance pour lire et effectuer les mises à jour dans la base des données.









On a créé des classes en détailles (attributs et méthodes) de la couche métier en respectant les règles suivantes :

-  **Metier.Gestion**
  -  **GestionDocument.java**

Cette couche est divisée en deux packages : POJO et Gestion
  -  **GestionEtudiant.java**

POJO : classe Métier = Table BD avec getteurs et setteurs
  -  **GestionProfesseur.java**
  -  **GestionProjet.java**

Gestion : classe qui gère les opération CRUD et la logique métier des POJO (Collections du POJO + Méthodes CRUD)
  -  **GestionRapport.java**
  -  **GestionRendez.java**
  -  **GestionUser.java**

Les classe Gestion sont les seuls qui communiquent avec la couche persistance.
-  **Metier.POJO**
  -  **Document.java**
  -  **Etudiant.java**
  -  **Professeur.java**
  -  **Projet.java**
  -  **Rapport.java**
  -  **Rendez.java**
  -  **User.java**



## Couche Présentation :

Cette couche contient tous les interfaces graphiques et la logique de navigation. Cette couche communique avec la couche métier (Plus de détail est au-dessous).

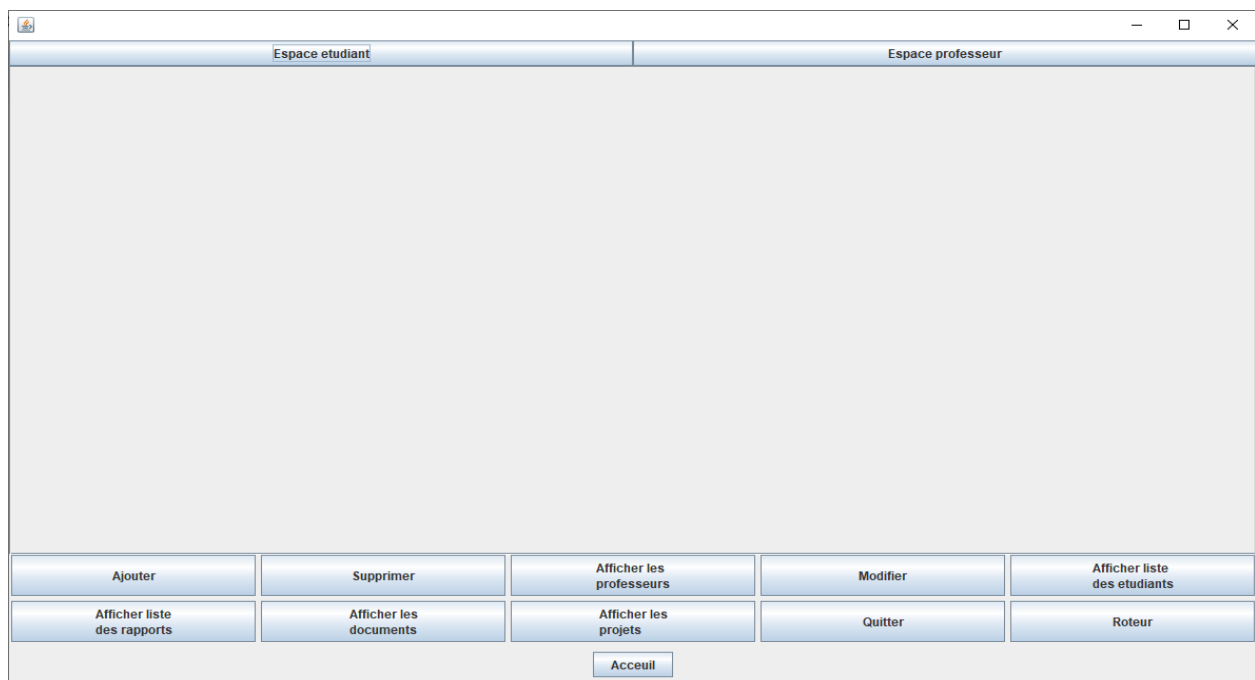
A l'aide du programme [Eclipse](#) on utilise la notion de JFRAME(SWING) pour créer des différents interfaces

## Interface principale :



## Partie Administrateur :

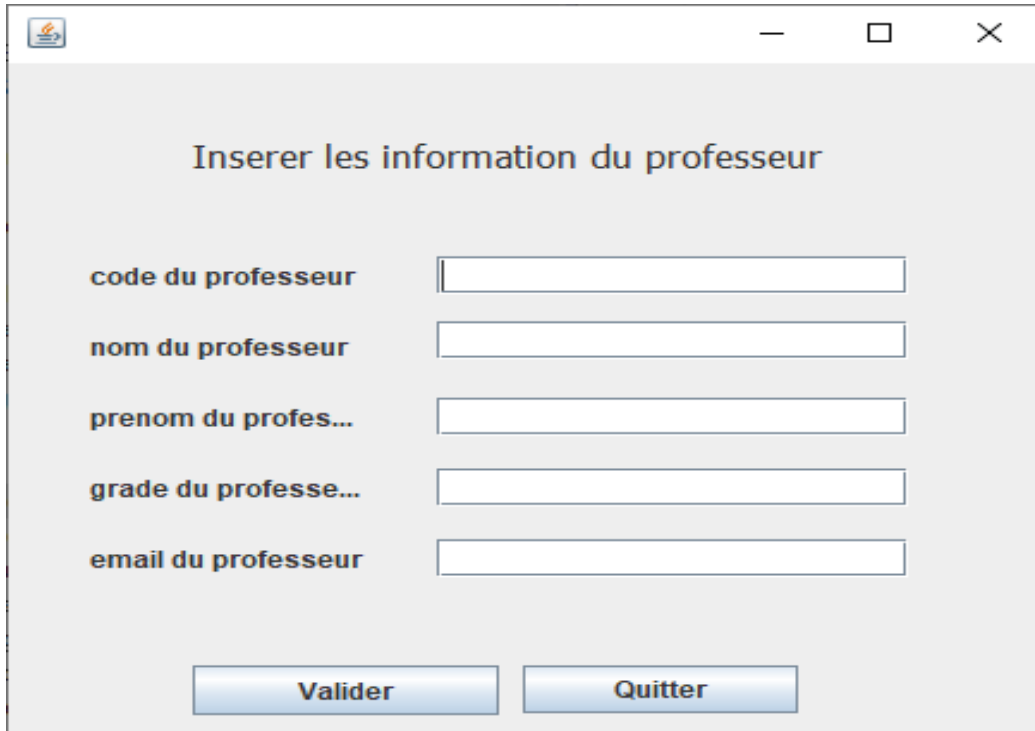
Dans notre projet, on a mis par défaut un administrateur (user Name: admin / password:1234), ce administrateur a plusieurs droits. Ce dernier a l'accès à deux espaces (étudiant ou professeur).



## Espace professeur :

L'administrateur a les droits suivants :

- Ajouter les professeurs :  
En cliquant sur le bouton <Ajouter> qui va par la suite donner à l'administrateur un formulaire Pour ajouter le nouveau professeur à la base de donne en respectant les règles d'intégrité :



**Insérer les information du professeur**

**code du professeur**

**nom du professeur**

**prenom du profes...**

**grade du professe...**

**email du professeur**

- supprimer le(s) professeurs :  
En sélectionnant les professeur a supprimer dans le JTable a la droite de la fenêtre principale  
Et en cliquant sur la bouton <Supprimer>
- Afficher les professeurs  
En cliquant sur le bouton <Afficher> qui va afficher le contenu du tableau professeur au niveau de notre base de données
- Modifier un professeur  
En cliquant sur le bouton <Modifier> qui va donner au administrateur une fenêtre de modification en respectant les règles d'intégrité
- Afficher les étudiants d'un professeur :  
En cliquant sur le bouton <Afficher liste des étudiants> qui va donner à l'administrateur une fenêtre qui demande l'id du professeur pour afficher ses étudiants
- Afficher les rapports gère par un professeur :  
En cliquant sur le bouton <Afficher liste des rapports> qui va donner à l'administrateur une fenêtre qui demande l'id du professeur pour afficher les rapports encadre par lui
- Afficher les documents d'un professeur :  
En cliquant sur le bouton <Afficher liste des documents> qui va afficher les documents rendu par un professeur selon son id
- Afficher les étudiants d'un professeur :

En cliquant sur le bouton <Afficher liste des étudiants> qui va donner à l'administrateur une fenêtre qui demande l'id du professeur pour afficher ses étudiants

- Afficher les projets gère par un professeur :

En cliquant sur le bouton <Afficher liste des projets> qui va donner à l'administrateur une fenêtre qui demande l'id du professeur pour afficher les projets (PFE/PFA/DOCTORANT) encadre par lui.

## Espace étudiant :

L'administrateur a les droits suivants :

- Ajouter les étudiants :

En cliquant sur le bouton <Ajouter> qui va par la suite donner à l'administrateur un formulaire Pour ajouter les nouveaux étudiants à la base de donne en respectant les règles d'intégrité (presque la même fenêtre donne au niveau de l'insertion d'un professeur).

- supprimer un étudiant :

En sélectionnant les professeur à supprimer dans le JTable a la droite de la fenêtre principale Et en cliquant sur le bouton <Supprimer>

- Afficher les étudiants

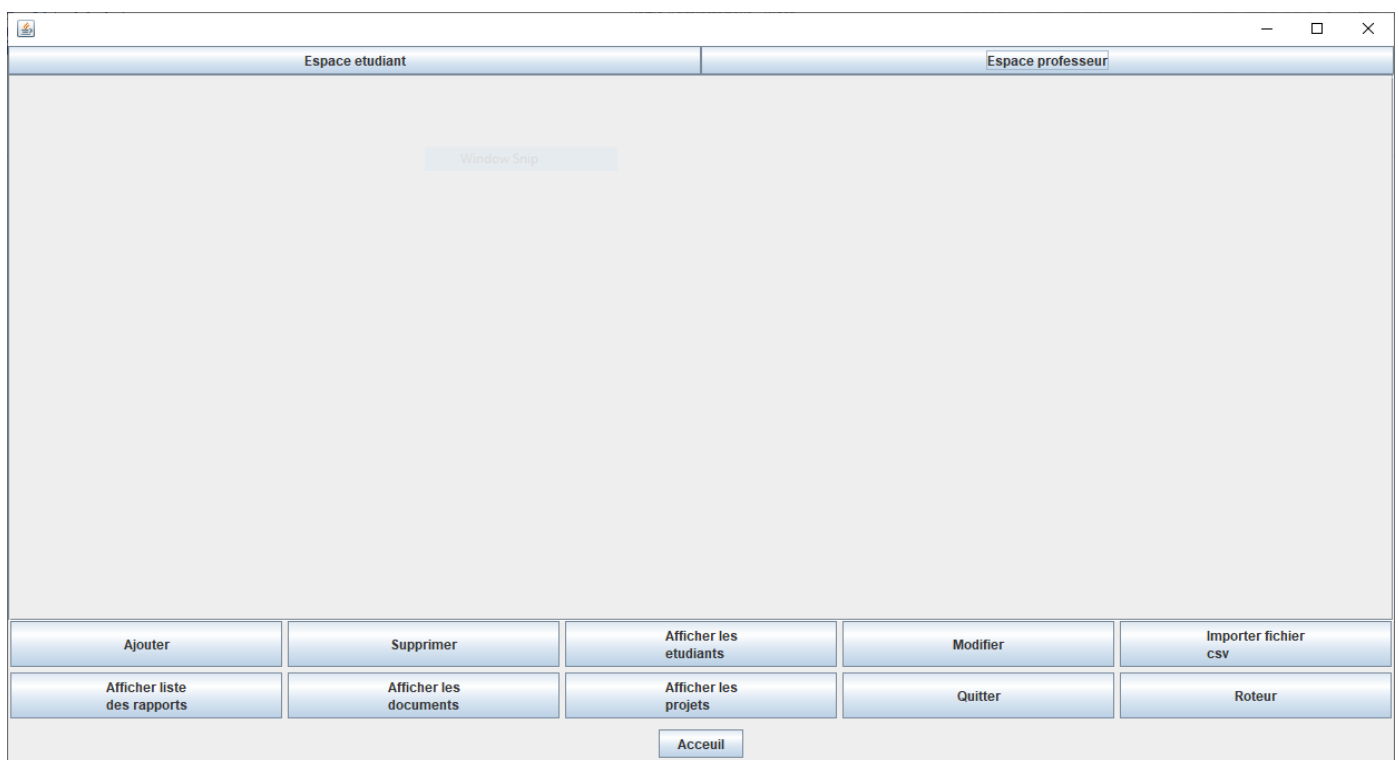
En cliquant sur le bouton <Afficher> qui va afficher le contenu du tableau étudiant au niveau de notre base de données

- Modifier un étudiant

En cliquant sur le bouton <Modifier> qui va donner à l'administrateur une fenêtre de modification en respectant les règles d'intégrité

- Afficher les rapports d'un étudiant :

En cliquant sur le bouton <Afficher liste des rapports> qui va donner à l'administrateur une fenêtre qui demande l'id d'étudiant pour afficher ses rapports.



- Afficher les documents d'un étudiant:  
En cliquant sur le bouton <Afficher liste des documents> qui va afficher les documents rendu par un étudiant selon son CNE.
- Afficher les projets gère par un professeur :  
En cliquant sur le bouton <Afficher les projets> qui va donner à l'administrateur une fenêtre qui demande le CNE d'un étudiant pour afficher ses projets (PFE/PFA/DOCTORANT).
- Ajouter les étudiants par un fichier csv :  
En cliquant sur le bouton <Importer fichier csv> qui va donner à l'administrateur une fenêtre qui demande le chemin du fichier csv qui va ajouter seulement les étudiants de la 4eme et la 5eme année(le fichier doit contenir les mêmes coulons que la table étudiant dans la base de données) .

## Partie professeur :

Lorsque le professeur se connecter il va s'afficher l'interface du Professeur.

L'interface Professeur contient 6 buttons (Retour, Afficher Etudiants, Afficher Rapports, Afficher Documents, Déposer un document, Etablir un rendez-vous), les informations personnelles du professeur et un espace notifications.

**Espace Professeur**

**Informations Personnelles :**

Nom : Ahmed

Prenom : Mohammed

Email : mohammed@ahmed.com

Id : 154

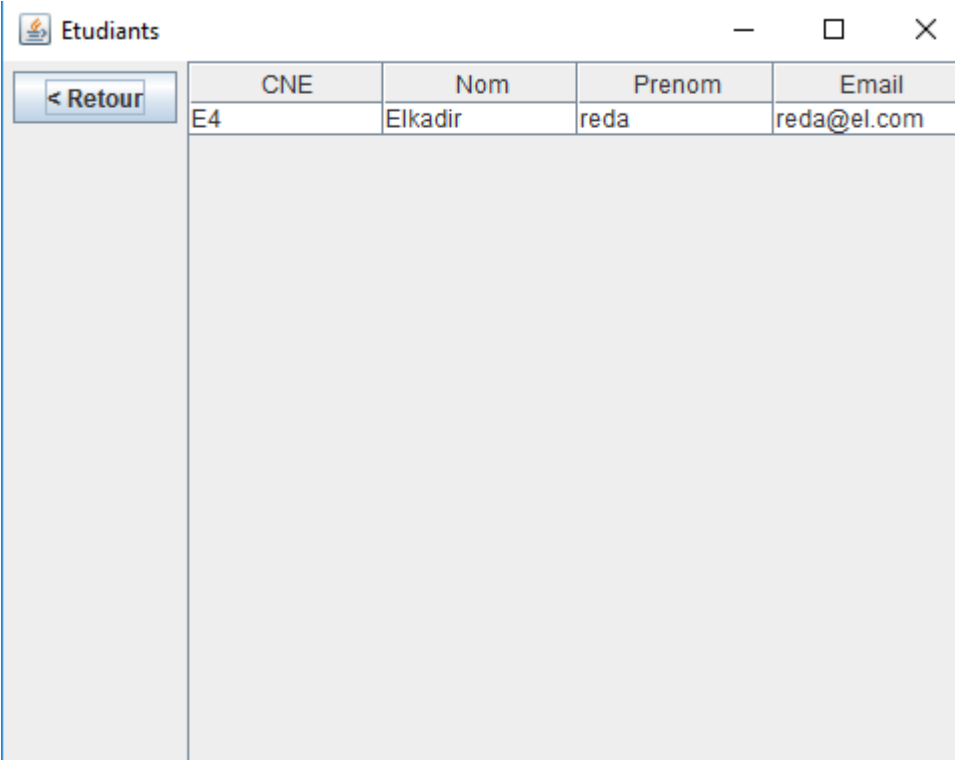
**Notifications**

< Retour    Afficher les Etudiant    Afficher les Rapports

Afficher les Documents    Deposer un document    Etablir un rendez-vous

L'espace des notifications est réservé pour recevoir les messages des étudiants et les rendez-vous qui sont déjà déterminés.

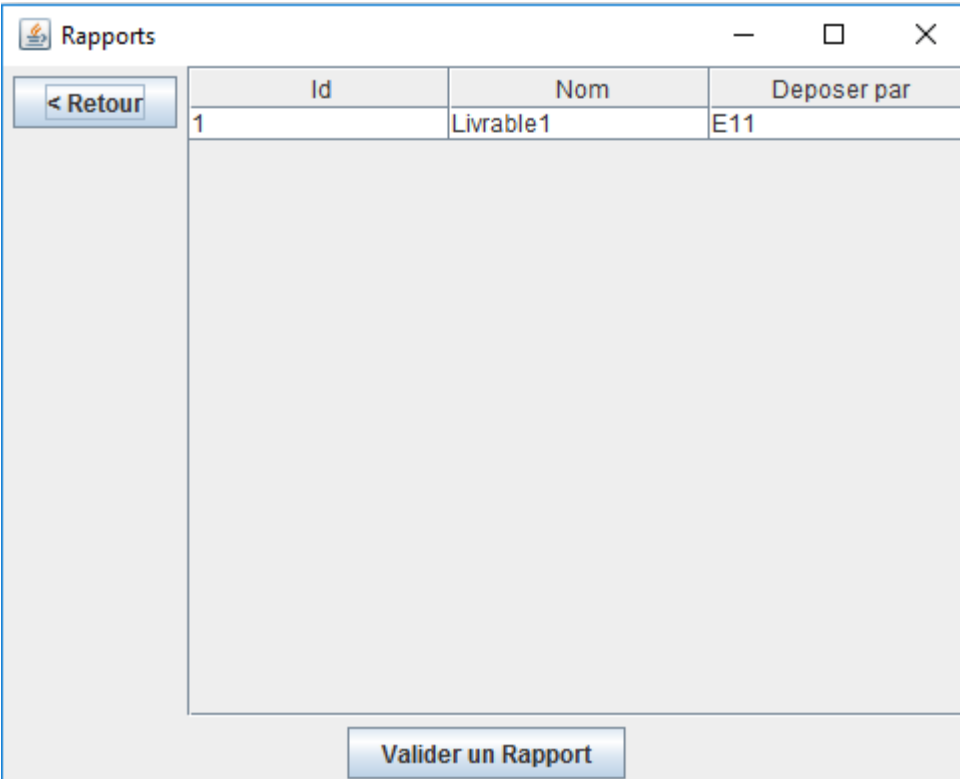
Lorsque le professeur clique sur le bouton Afficher Etudiants il va s'afficher une fenêtre qui contient un tableau des étudiants et un bouton (< Retour) réservé pour revenir à la page précédente.



The screenshot shows a window titled 'Etudiants' with a table containing student information. The table has four columns: CNE, Nom, Prenom, and Email. The first row contains the values E4, Elkadir, reda, and reda@el.com. A button labeled '< Retour' is located on the left side of the window.

CNE	Nom	Prenom	Email
E4	Elkadir	reda	reda@el.com

Lorsque le professeur clique sur le buttons Afficher Rapports il va s'afficher une fenêtre qui contient un tableau des rapports et deux buttons le premier (< Retour) est réserve pour revenir à la page précédente, et l'autre (Valider un rapport) qui est réserve pour la validation des projets.

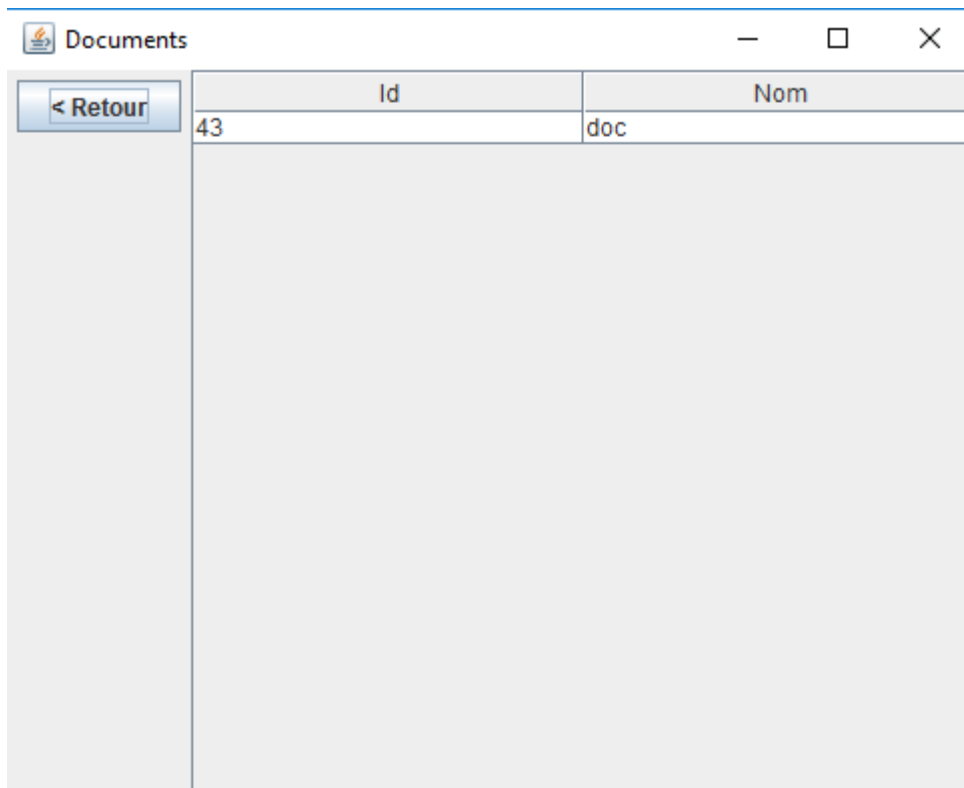


The screenshot shows a window titled 'Rapports' with a table containing report information. The table has three columns: Id, Nom, and Deposer par. The first row contains the values 1, Livable1, and E11. A button labeled '< Retour' is located on the left side of the window, and a button labeled 'Valider un Rapport' is located at the bottom center of the window.

Id	Nom	Deposer par
1	Livable1	E11

Lorsque le professeur clique sur le bouton Afficher Documents il va s'afficher une fenêtre qui contient un tableau des Documents et un bouton (< Retour) réservé pour revenir à la page précédente.

Le bouton Déposer un document est utilisé par le professeur pour importer un fichier.



Le bouton Etablir un rendez-vous est utilisé pour la création des rendez-vous (réunions) avec les étudiants.

## Partie étudiant :

Lorsque l'étudiant se connecte il va s'afficher l'interface étudiant.

L'interface étudiant contient 4 boutons (documents, rapports, mon projet, accueil), les informations personnelles de l'étudiant et un espace notifications.

L'espace des notifications est réservé pour recevoir les messages et les rendez-vous qui sont déjà déterminés par les professeurs.

**Bienvenue !!**

Nom :	FADILI	<b>Notifications :</b> vous avez un rendez-vous le 09-05-2020
Prénom :	EL MOSTAFA	
CNE :	L125202548	
Email :	mustafa@gmail.com	
Anné :	3 ème Anné	

Documents    Rapports    mon Projet    Accueil

Lorsque l'étudiant clique sur le bouton mon projet il va s'afficher une fenêtre qui contient les informations du projet de l'étudiant.

**Mon Projet :**

ID :	Projet_01
Type :	Fin année
Sujet :	Analyse
L'Avancement :	75%
Validation :	Non Validé

Retour    Accueil

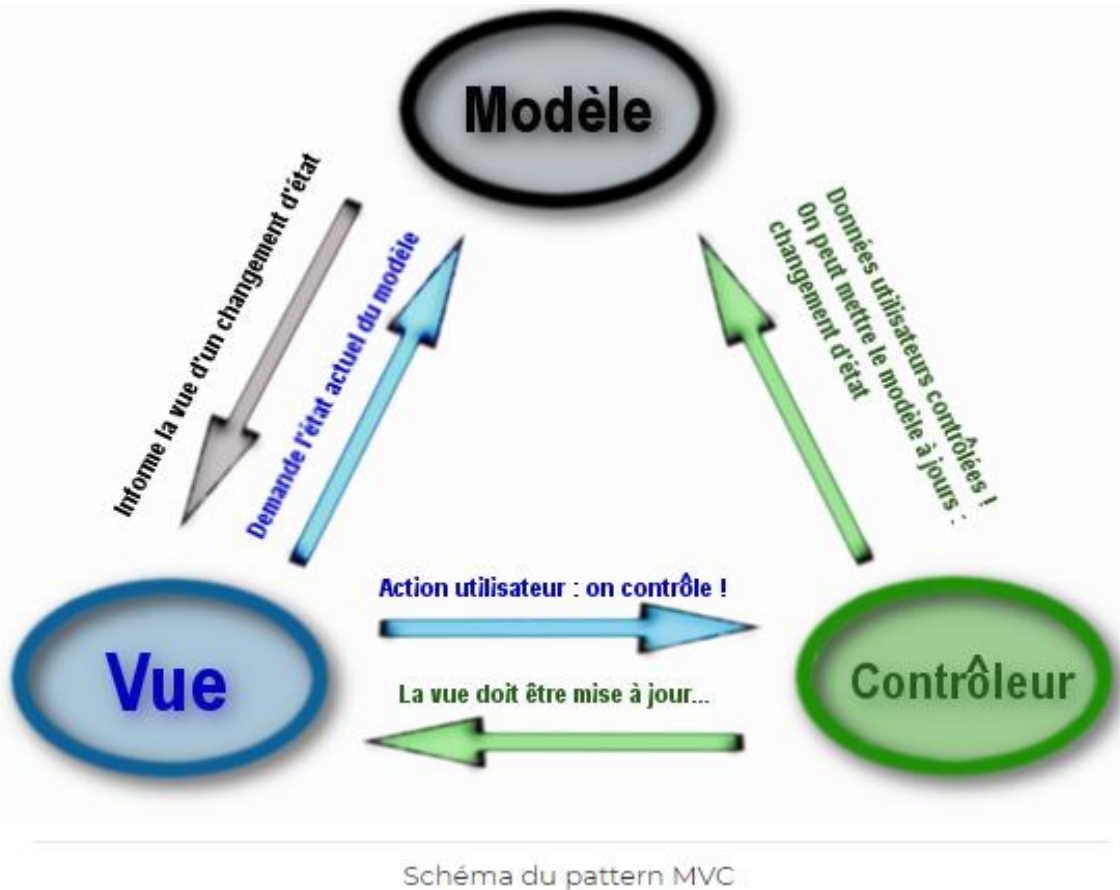
Lorsque l'étudiant cliquer sur le buttons <documents > (resp <rapports >) il va s'afficher une fenêtre qui contient un tableau des information des documents (resp rapports) et 3 buttons <ajouter>, (<Retour) réserver pour revenir à la page précédente et <accueil >réserver pour revenir à la page principale.



## Problème rencontré et solution :

Après la réalisation des déferents interfaces graphiques à l'aide du JAVA(SWING) et connaitre les déferents procédures des fonctionnalités du programme, le problème est comment relier entre les interfaces graphiques d'une façon organiser et plus efficace avec le moins consommation du RAM.

Le problème était résolu grâce à l'utilisation le modèle MVC(modèle, vue, contrôleur)



































- Le pattern MVC est un pattern composé du pattern observé et du pattern stratégie.
- Avec ce pattern, le code est découpé en trois parties logiques qui communiquent entre elles :
  - La vue (fenêtre)
  - Le modèle (données)
  - Le contrôleur qui lie les deux.
- L'implémentation du pattern observer permet au modèle de tenir informés ses observateurs.
- L'implémentation du pattern stratégie permet à la vue d'avoir des contrôles différents.
- Utiliser ce pattern permet de découpler trois acteurs d'une application, ce qui permet plus de souplesse et une maintenance plus aisée du code.

Ce qui concerne notre projet :

## LA VUE :

La vue représente ce que l'utilisateur a sous les yeux. La vue peut donc être :




















 Presentation.Vues  
 >  AddEtudiant.java  
 >  AddProfesseur.java  
 >  AfficherDocumentsFromProfesseur.jav  
 >  AfficherEtudiant.java  
 >  AfficherProjetsFromEtudiant.java  
 >  AfficherRapportFromEtudiant.java  
 >  AjoutDocument.java  
 >  AjoutProjet.java  
 >  AjoutRapport.java  
 >  ChoiceForAdmin.java  
 >  EtudiantAdministration.java  
 >  FirstWindow.java  
 >  ModifierEtudiant.java  
 >  ModifierProfesseur.java  
 >  ProfesseurAdministration.java  
 >  Test.java  
 >  VueAjtDocumentPr.java  
 >  VueDocument.java  
 >  VueDocumentPr.java  
 >  VueEtudiant.java  
 >  VueEtudiant2.java  
 >  VueEtudiantPr.java  
 >  VueEvaluerProgression.java  
 >  VueLoginPr.java  
 >  VueMainProfesseurPr.java  
 >  VueModifierRendezPr.java  
 >  VueProjet.java  
 >  VueProjetPr.java  
 >  VueRapport.java  
 >  VueRapportPr.java  
 >  VueRendezPr.java

- une page web ;
  - un terminal Linux ou une console Windows ;
  - une application graphique Swing, AWT, SWT pour Java
- On a créé un package nommer **Presentation** puis un package Vue qui contient tous les interfaces graphiques.

Chaque écran dans la maquette est présenté avec une classe

## LE MODÈLE :

C'est là que se trouvent les données. Il s'agit en général d'un ou plusieurs objets Java. Ces objets s'apparentent généralement à ce qu'on appelle souvent « la couche métier » de l'application et effectuent des traitements absolument transparents pour l'utilisateur. Par exemple, on peut citer des objets dont le rôle est de gérer une ou plusieurs tables d'une base de données.






 Presentation.Modules  
 >  ModelAddEtudiant.java  
 >  ModelAddProfesseur.java  
 >  ModelDocument.java  
 >  ModelDocumentPr.java  
 >  ModelEtudiantEtudiant.java  
 >  ModelEtudiantPr.java  
 >  ModelEtudiantProjet.java  
 >  ModelEtudiantRapport.java  
 >  ModelFirstWindow.java  
 >  ModelLoginPr.java  
 >  ModelMainProfesseurPr.java  
 >  ModelProfesseurDocument.java  
 >  ModelProfesseurProfesseur.java  
 >  ModelProjet.java  
 >  ModelProjetPr.java  
 >  ModelRapport.java  
 >  ModelRapportPr.java  
 >  ModelRendezPr.java

On a créé un package **Modules** et on Ajoute un classe modèle pour chaque écran (cette classe contient les données de l'écran avec getteurs et setteurs)

## LE CONTRÔLEUR

Le contrôleur permet de faire le lien entre la vue et le modèle lorsqu'une action utilisateur est intervenue sur la vue. C'est cet objet qui aura pour rôle de contrôler les données.

### ■ Presentation.Controlleur

- ▷  ControlleurEtudiantAd.java
- ▷  ControlleurFirstWindow.java
- ▷  ControlleurOpenAUser.java
- ▷  ControlleurProfesseurAdd.java
- ▷  ControlleurProfesseurEtudiantAd.java

On a créé un package **Controlleur** puis des classes qui contient une liste des méthodes. puisque notre contrôleur doit interagir avec le modèle, il faudra qu'il possède une instance de notre modèle

On va expliquer plus précisément la façon dont il travaille une application structurée en MVC, voici ce qu'il peut se passer :

- L'utilisateur effectue une action sur notre programme (un clic sur un bouton) ;
- L'action est captée par le contrôleur, qui va vérifier la cohérence des données et éventuellement les transformer afin que le modèle les comprenne. Le contrôleur peut aussi demander à la vue de changer ;
- Le modèle reçoit les données et change d'état (une variable qui change, par exemple) ;
- Le modèle notifie la vue (ou les vues) qu'il faut se mettre à jour ;
- L'affichage dans la vue (ou les vues) est modifié en conséquence en allant chercher l'état du modèle.

## Test et validation :

### LES TESTS UNITAIRES :

On a testé chaque méthode de la classe des différents packages soit pour la communication avec la base de données par exemple la connexion ou bien modèle CRUD (ajouter, rechercher, modifier, supprimer) soit les différentes vues (interfaces graphiques) .

On prend par exemple la méthode qui se trouve dans le package **persistance.DAO** dans la classe **DAOProfesseur** insert(Professeur p) pour ajouter un professeur dans la base de données avec **P=( P200,nom\_200,prenom\_200,PH,P200@ensak.com)**

On a appelé la méthode et on observe le résultat dans le programme de gestion de la base de données (exemple POSTGRESQL 10)

	code_professeur [PK] character varying (25)	nom_professeur character varying (25)	prenom_professeur character varying (25)	grade_professeur character varying (10)	email_professeur character varying (50)
1	P200	nom_200	prenom_200	PH	P200@ensak.com

On peut exporter les résultats sous forme de fichier CSV qui sépare les attributs par des points-virgules comme cet exemple **P200;nom\_200;prenom\_200;PH;P200@ensak.com;**

### LES TESTS GLOBAUX :

On a testé l'interface administrateur (espace professeur-- >ajouter un professeur)

la figure ci-dessous :

On remplit le formulaire par **(P201,nom\_201,prenom\_201,PA,P201@ensak.com)** et on clique sur <valider> puis on observe le résultat dans le programme de gestion de la base de données (exemple POSTGRESQL 10)

	code_professeur [PK] character varying (25)	nom_professeur character varying (25)	prenom_professeur character varying (25)	grade_professeur character varying (10)	email_professeur character varying (50)
1	P200	nom_200	prenom_200	PH	P200@ensak.com
2	P201	nom_201	prenom_201	PA	P201@ensak.com

## La maintenance :

Après avoir réalisé notre programme qui est capable de gérer les projets des étudiants dans l'école, on peut proposer des évolutions pour notre programme comme la gestion des ID d'une manière automatique, c-à-d le utilisateur (administrateur, professeur, étudiant) peut pas ajouter ID du objet (document, projet, étudiant, professeur, ...) chaque fois et vérifier si existe ou non car le problème est si il y a plusieurs objets avec différents ID entrés par différents utilisateurs.

Cette proposition est facile à réaliser grâce à une vérification dans la base de données et donner automatique ID qui n'existe pas à l'utilisateur donc ce dernier a le droit d'écrire les informations comme (nom, prénom, type, email...etc) sans problème de répétition et vérification chaque fois ID, dans le but de minimiser le temps.

## Conclusion :

Ce projet a pour objectif la conception et réalisation d'un programme permettant d'aider à la gestion des projets des étudiants. Au cours de ce projet, on a présenté les différentes étapes de la conception et la réalisation du programme, afin de satisfaire les besoins des utilisateurs on a commencé par la conception en utilisant le formalisme de Merise, ensuite la réalisation à l'aide du JAVA.

Ce projet a fait l'objet d'une expérience intéressante, qui nous permet d'améliorer nos connaissances et nos compétences dans le domaine de la programmation. En fin nous souhaitons que ce travail soit satisfaisant et notre objectif sera atteint.

De l'avis général, nous avons consolidé nos connaissances générales et plus orientées pour que le travail soit réussir.

Au niveau de la gestion du projet en équipe, nous avons réussi à bien nous répartir les tâches afin de réaliser nos objectifs.

## ANNEXE

### Un guide d'industrialisation

