

**Conception et implémentation d’un site E-learning**





**Réalisé par :**

**Tchikou Lakhdar**

**Groupe 03 / ISII**

22 juin 2019

Université Alger I – Benyoucef Benkhada-

Année Univ. 2018/2019

**Table des matières**

[**Liste des figures :** 2](#_Toc13637091)

[**Liste des Tableaux :** 4](#_Toc13637092)

[**Introduction :** 5](#_Toc13637093)

[**Etude des besoins :** 6](#_Toc13637094)

[ **Besoins fonctionnels :** 6](#_Toc13637095)

[ **Besoins non fonctionnels :** 6](#_Toc13637096)

[**Conception :** 7](#_Toc13637097)

[ **Cycle de vie :** 7](#_Toc13637098)

[ **Méthodologie de conception :** 8](#_Toc13637099)

[ **Conception générale :** 8](#_Toc13637100)

[ **Conception détaillée :** 9](#_Toc13637101)

[**1.** **Diagramme de cas d’utilisation :** 9](#_Toc13637102)

[**2.** **Diagramme de classe :** 12](#_Toc13637103)

[ **La Maquette de notre site web :** 13](#_Toc13637104)

[**1)** **Structure du site :** 13](#_Toc13637105)

[**2)** **La charte graphique :** 13](#_Toc13637106)

[**Implémentation** 15](#_Toc13637107)

[ **Environnement de travail :** 15](#_Toc13637108)

[**1.** **Environnement Hard :** 15](#_Toc13637109)

[**2.** **Outils et logiciels de développement :** 15](#_Toc13637110)

[**3.** **Technologies :** 16](#_Toc13637111)

[ **Les Architectures :** 18](#_Toc13637112)

[**i.** **ARCHITECTURE APPLICATIVE :** 18](#_Toc13637113)

[**ii.** **ARCHITECTURE TECHNIQUE :** 19](#_Toc13637114)

[ **Description du code :** 20](#_Toc13637115)

[**1-** **Utilisation du framework Spring :** 20](#_Toc13637116)

[**2-** **Les configurations :** 21](#_Toc13637117)

[**3-** **L’injection des dépendances :** 23](#_Toc13637118)

[**4-** **Les contrôleurs (servelet) et les pages JSP :** 31](#_Toc13637119)

[**5-** **Spring sécurité :** 37](#_Toc13637120)

[ **Démonstrations des interfaces :** 42](#_Toc13637121)

[**1.** **L’interface d’Authentification dans le site :** 42](#_Toc13637122)

[**2.** **Espace Enseignant :** 43](#_Toc13637123)

[**3.** **Espace Etudiant :** 47](#_Toc13637124)

[**Conclusion et perspectives :** 49](#_Toc13637125)

[**Bibliographie** 50](#_Toc13637126)

# **Liste des figures :**

[**Figure 1: Modèle de cycle de vie en V** 7](#_Toc13637002)

[**Figure 2: L'architecture MVC adapté** 8](#_Toc13637003)

[**Figure 3: Architecture globale du système** 9](#_Toc13637004)

[**Figure 4: Diagramme de cas d'utilisation d'un enseignant** 10](#_Toc13637005)

[**Figure 5: Diagramme de cas d'utilisation d'un étudiant** 11](#_Toc13637006)

[**Figure 6: Diagramme de classe** 12](#_Toc13637007)

[**Figure 7: structure en évolution de site "e-learning"** 13](#_Toc13637008)

[**Figure 8: la charte graphique du site 'E\_Leraning'** 14](#_Toc13637009)

[**Figure 9: L'architecture applicative du 'e-learning'** 18](#_Toc13637010)

[**Figure 10:L'architecture technique du 'e-learning'** 19](#_Toc13637011)

[**Figure 11: Création d'un projet spring mvc** 21](#_Toc13637012)

[**Figure 12: le fichier persistence.xml** 21](#_Toc13637013)

[**Figure 13: le fichier ContextApplication.xml** 22](#_Toc13637014)

[**Figure 14: Les dépendances ajoutées dans le fichier pom.xml** 23](#_Toc13637015)

[**Figure 15: fichier web.xml** 23](#_Toc13637016)

[**Figure 16: Le code de l’entité Enseignant.java** 26](#_Toc13637017)

[**Figure 17: Les tables dans notre base** 27](#_Toc13637018)

[**Figure 18: l’interface I\_ElearningDAO.java** 27](#_Toc13637019)

[**Figure 19: code d’ElearningDao\_impl.java** 28](#_Toc13637020)

[**Figure 20: L'interface de EnseignantDAO.java** 29](#_Toc13637021)

[**Figure 21: Le code d'implémentation d’EnseignantDAO.java** 29](#_Toc13637022)

[**Figure 22: Création d'interface Enseingnant\_Service.java** 29](#_Toc13637023)

[**Figure 23: Code d'Imp\_Enseignant\_Service.java** 30](#_Toc13637024)

[**Figure 24: Le fichier beans\_config.xml** 31](#_Toc13637025)

[**Figure 25: Déclaration des beans dans ApplicationContext.xml** 31](#_Toc13637026)

[**Figure 26: le contrôleur cours** 32](#_Toc13637027)

[**Figure 27: La méthode ajouterCours.java** 32](#_Toc13637028)

[**Figure 28: La méthode ModifierCours.java** 33](#_Toc13637029)

[**Figure 29: La méthode supprimerCours.java** 33](#_Toc13637030)

[**Figure 30: le fonctionnement du modèle MVC** 34](#_Toc13637031)

[**Figure 31: la page include.jsp** 34](#_Toc13637032)

[**Figure 32: le fichier servelet-context.xml** 35](#_Toc13637033)

[**Figure 33: la page jsp d'Etudiant** 37](#_Toc13637034)

[**Figure 34: configuration du fichier applicationContext.xml** 38](#_Toc13637035)

[**Figure 35: le fichier data-source-config.xml** 39](#_Toc13637036)

[**Figure 36: security-config.xml** 40](#_Toc13637037)

[**Figure 37: erreurController.java** 40](#_Toc13637038)

[**Figure 38: dépendances spring ajoutées dans pom.xml** 41](#_Toc13637039)

[**Figure 39: l'ajout de filtre dans web.xml** 41](#_Toc13637040)

[**Figure 40: spécification de spring sécurité dans la page login.jsp** 41](#_Toc13637041)

[**Figure 41: url spring sécurité\_logout** 42](#_Toc13637042)

[**Figure 42: l’interface login** 42](#_Toc13637043)

[**Figure 43: login d'un enseignant** 43](#_Toc13637044)

[**Figure 44: interface de page d'accueil** 43](#_Toc13637045)

[**Figure 45: consultation la liste des étudiants (leurs informations)** 44](#_Toc13637046)

[**Figure 46: La liste des cours** 44](#_Toc13637047)

[**Figure 47: la modification d'un cours** 45](#_Toc13637048)

[**Figure 48: la suppression d'un cours** 45](#_Toc13637049)

[**Figure 49: la mise à jour des examens** 46](#_Toc13637050)

[**Figure 50: mise à jour des questions** 46](#_Toc13637051)

[**Figure 51: interface de page d'accueil** 47](#_Toc13637052)

[**Figure 52: consultation l'affichage des cours** 47](#_Toc13637053)

[**Figure 53: Consultation l'affichage des notes.** 48](#_Toc13637054)

# **Liste des Tableaux :**

[**Tableau 1: Les annontations JPA de base** 24](#_Toc12963278)

[**Tableau 2: Les annontations de relations entre les tables** 24](#_Toc12963279)

# **Introduction :**

Durant ces dernières décennies, l’utilisation d’Internet est devenue incontournable pour les entreprises, qui évoluent dans une compétition de plus en plus rude dans un environnement en évolution continuelle. Pour répondre à de tels besoins, certaines entreprises ont adopté des organisations distribuées de leurs organisations.

A l’heure où tout doit aller très vite, les applications d’entreprises doivent faire face aux modifications et à la complexité de leur environnement, tout en étant robustes et en permettant la réutilisabilité et la portabilité; C’est précisément pour ces raisons qu’a été mise en place la plate-forme Java Entreprise Edition (JEE), référence pour le développement d'applications d'envergure et supportant la programmation, la persistance des données, les transactions, la sécurité et le déploiement de projets. La contribution principale de ce présent travail porte sur la création d’un site web dynamique qui vise à réaliser un système de formation à distance accessible aux enseignants et aux étudiants depuis n’importe quel ordinateur et plus précisément d’approvisionnement, afin de faciliter et de faire évoluer la communication avec ces derniers .

De ce fait, nous avons utilisé la plateforme JEE, ceci, afin de pouvoir profiter de la puissance du langage Java et des avantages de la plateforme JEE discutées plus lois dans ce document.

Le rapport est composé de 3 parties :

* La première partie propose étude des besoins.
* La deuxième partie présente la conception en proposant les diagrammes de cas d’utilisation et de classe.
* Enfin la troisième partie l’implémentation et l’environnement matériel et logiciel de l’application, en mettant un aperçu sur les interfaces réalisées pour montrer l’utilisation de l’application

# **Etude des besoins :**

Nous intéressons aux besoins des utilisateurs traités dans notre projet c’est à dire l’accès des enseignants et les étudiants à la plateforme, en assurant les différentes taches des chacun à travers les spécifications fonctionnelles et non fonctionnelles pour aboutir à un site de qualité qui répond aux besoins demandés.

## **Besoins fonctionnels :**

Les besoins fonctionnels se présentent en sept grandes parties :

* **L’inscription d’utilisateur (enseignant ou étudiant) :** Jusqu’à ce stade, l’enseignant et l’étudiant sont toujours anonyme mais pour pouvoir passer à un stade plus rigoureux, il faut qu’ils s’inscrivent, cela se fait uniquement pour le premier accès à la plateforme, après, ils s’authentifient avec leurs E-mail et leur mot de passe pour passer à d’autres taches.
* **Exposition des cours ainsi que leurs caractéristiques.**
* **Ajout des cours et examens.**
* **Consultation des cours par les étudiants et les enseignants.**
* **Consultation des réponses et des notes des étudiants.**
* **Mis à jour des documents par les enseignants.**
* **Affichage des questions d’examens.**

## **Besoins non fonctionnels :**

Les besoins non fonctionnels sont importants car ils agissent de façon indirecte sur le résultat et sur le rendement de l’utilisateur, ce qui fait qu’ils ne doivent pas être négligés, pour cela il faut répondre aux exigences suivantes :

* **Fiabilité :** L’application doit fonctionner de façon cohérente sans erreurs et doit être satisfaisante.
* **Les erreurs :** Les ambigüités doivent être signalées par des messages d’erreurs (feedback) bien organisés pour bien guider l’utilisateur et le familiariser avec notre site web.
* **Ergonomie de l’Interface :** L’application doit être adaptée à l’utilisateur sans qu’il ne fournisse aucun effort (utilisation claire et facile) de point de vue navigation entre les différentes pages, couleurs et mise en textes utilisés.
* **Sécurité :** Notre solution doit respecter surtout la confidentialité des données personnelles des utilisateurs qui reste l’une des contraintes les plus importantes dans les sites web.
* **Aptitude à la maintenance et la réutilisation :** Le système doit être conforme à une architecture standard et claire permettant sa maintenance et sa réutilisation.

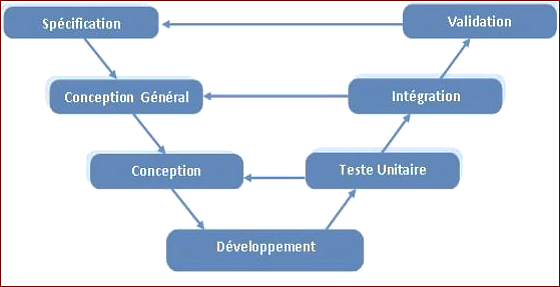
# **Conception :**

Tout au long le cycle de vie de notre projet, la conception représente une phase primordiale et déterminante pour produire une application de haute qualité. C’est dans ce stade que nous devons clarifier en premier lieu la vue globale, en décrivant l’architecture générale que nous allons suivre dans la partie réalisation de notre projet. Puis, nous allons détailler notre choix conceptuel à travers des diagrammes UML.

## **Cycle de vie :**

Le cycle de vie d'une application comprend toutes les étapes depuis sa conception et sa réalisation jusqu’à sa mise en œuvre. L'objectif d'un tel découpage est de permettre de définir des jalons intermédiaires permettant la validation du développement du logiciel et la vérification de son processus de développement.

Afin de concevoir et développer notre application, nous avons opté pour le modèle de cycle de vie en **V**. Ce choix reviens au fait que ce cycle est le plus efficace avec son principe de travail qui nécessite la vérification de chaque étape et la possibilité de corriger les fautes avant de se lancer vers l’étape suivante ( Deux types de tâches sont réalisées en parallèle : Verticalement : on prépare l’étape suivante et Horizontalement : on prépare la vérification de la tâche en cours).



**Figure 1: Modèle de cycle de vie en V**

## **Méthodologie de conception :**

Notre projet consiste à concevoir et réaliser un système de formation à distance en se basant sur le modèle **MVC** constitué de trois parties.

Bien évidemment, les deux parties connues qui sont les vues V (les interfaces IHM) et le modèle M (le serveur de données) et une troisième partie représenté comme contrôleur de trafic C, (le serveur d’application).

Cette architecture a pas mal d’avantages pour qu’elle reste toujours la plus utilisée dans le monde de développement Web comme notre cas étant donnée qu’elle se caractérise par :

* L’allégement du poste de travail.
* La prise en compte de l'hétérogénéité des plates-formes (serveurs, clients, langages, etc).
* L’introduction de clients dits " légers " (plus liée aux technologies Intranet/HTML qu'au 3-tiers proprement dit).
* Une meilleure répartition de la charge entre les différents entités clients et serveurs.

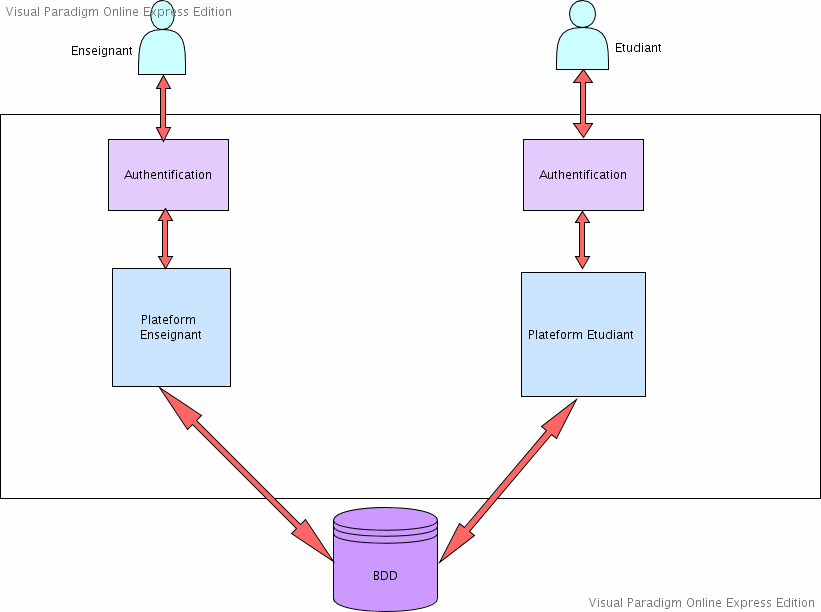


**Figure 2: L'architecture MVC adapté**

## **Conception générale :**

Nous avons présenté les deux plates-formes de notre application, une plate-forme enseignant et une plate-forme étudiant.

La figure suivante (figure-3) représente l’architecture globale de système :



**Figure 3: Architecture globale du système**

## **Conception détaillée :**

Pour bien faciliter notre tâche encore, nous avons adapté le langage de modélisation unifié **UML** (Unified Modelling Language) qu’est une notation qui permet de modéliser un problème de façon standard à un tel point que sa connaissance devienne indispensable pour un développeur.

Dans notre projet, nous avons basées sur deux diagrammes pour faciliter notre conception ; c’est **le diagramme de cas d’utilisation** et **le diagramme de classe**.

### **Diagramme de cas d’utilisation :**

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés.

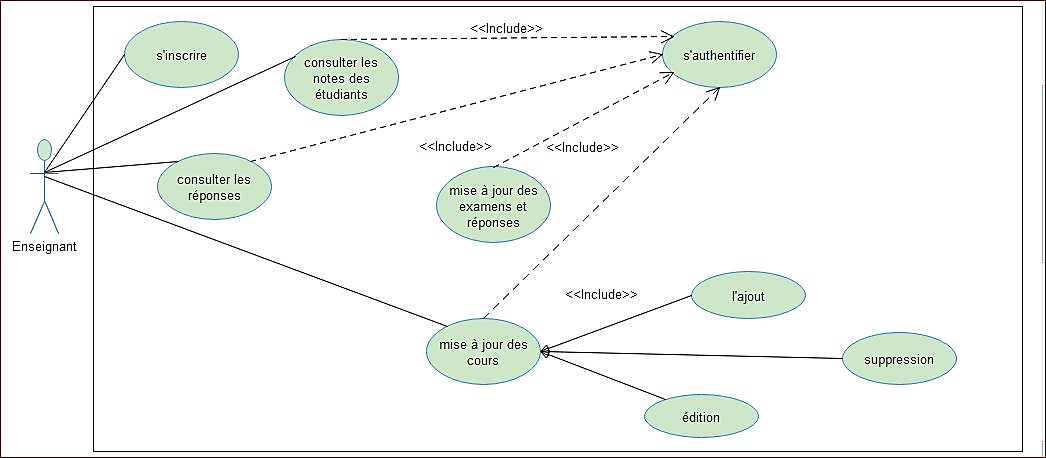
Pour illustrer les cas d'utilisation de notre système, nous vous présentons les différents acteurs qui interviennent dans notre application qui sont l’enseignant et l’étudiant.

* **Enseignant :**

Un enseignant est un visiteur de site qui peut réaliser les taches suivantes :

* Inscription dans le site.
* Authentification.
* Mise à jour des cours (l’ajout, modification et la suppression).
* Mise à jour des examens (l’ajout, modification et la suppression).
* Consultation des réponses des étudiants.
* Consultation des notes des étudiants.

La figure suivante (figure-4) représente le diagramme de cas d'utilisation d’un enseignant :



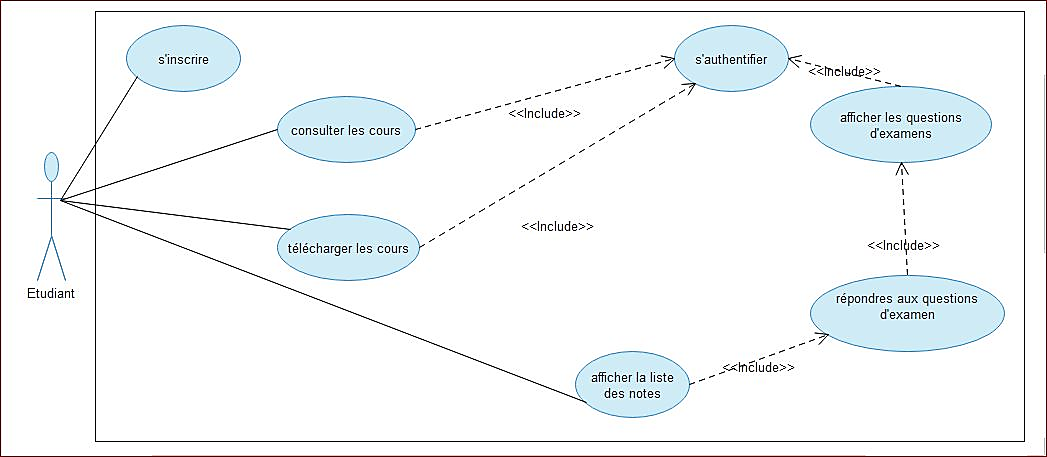
**Figure 4: Diagramme de cas d'utilisation d'un enseignant**

* **Etudiant :**

Un étudiant est un visiteur aussi de site qui peut réaliser les taches suivantes :

* Inscription dans le site.
* Authentification.
* Consultation des cours.
* Téléchargement des cours.
* Affichage des questions d’examen.
* Répondre aux questions.
* Consultation l’affichage des notes.

La figure suivante (figure-5) représente le diagramme de cas d'utilisation d’un étudiant :



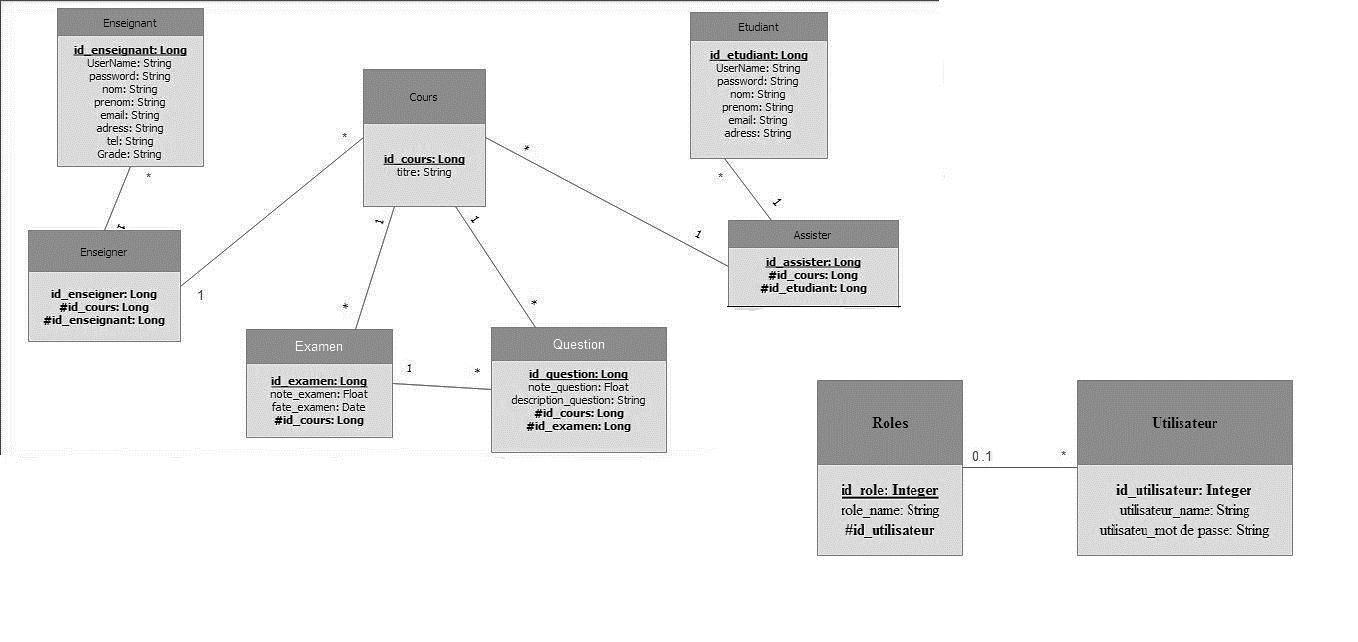
**Figure 5: Diagramme de cas d'utilisation d'un étudiant**

### **Diagramme de classe :**

Le diagramme de classes est un schéma utilisé en UMl pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci.

* Il permet de déterminer les concepts manipulés dans le site ainsi sa structure.

Cette figure (figure-6) représente le diagramme de classe, pour démontrer la structure des tables utilisées dans la base de données et les relations entre eux.

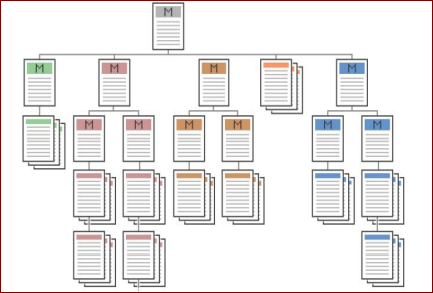


**Figure 6: Diagramme de classe**

## **La Maquette de notre site web :**

### **Structure du site :**

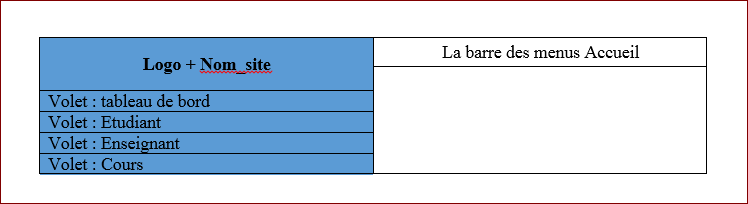
Dans un site web e-learning, le faite maintenir une hiérarchisation équilibrée permet l'accès rapide à l'information et une compréhension intuitive de la façon dont les pages sont organisées tout en donnant la possibilité d’évoluer est un objectif préalable. Pour cela nous avons choisis la structure en évolution.



**Figure 7: structure en évolution de site "e-learning"**

### **La charte graphique :**

Une charte graphique aboutit généralement à la création de modèles de pages (en anglais Template) servant comme des gabarits pour la création du site web. Les Template sont des images créées sous forme de calques ou bien des pages web représentant le squelette graphique des pages types dans notre site web comme par exemple : la page d’accueil, la page Enseignant et la page Etudiant dans notre projet .



**Figure 8: la charte graphique du site 'E\_Leraning'**

Nous venons de terminer la partie de conception, qui consiste à déterminer aussi bien les méthodes de travail que les chartes graphiques de notre site web avec ses parties statiques et dynamiques.

Nous passerons alors à aborder la dernière partie qui représente la partie d’implémentation de notre site web, en se basant sur les mécanismes et les solutions déterminés dans la phase de conception.

# **Implémentation**

## **Environnement de travail :**

### **Environnement Hard :**

Hôte : Assus

Microprocesseur : intel core i7

RAM : 16.00 Go

### **Outils et logiciels de développement :**

#### **Outil de Conception :**

 **: VisualParadigm** est un outil de modélisation UML. Il permet d'analyser, de dessiner, de coder, de tester et de déployer.Cette application permet de dessiner tous les types de diagrammes UML notamment le diagramme de cas d’utilisation et digramme de classe que nous avons adopté pour notre projet.

#### **Environnement de développement :**

*  : **Eclipse Spring Tool Suit** **(version 4.3.4)** est un IDE, décliné et organisé en un ensemble de sous-projets de développement logiciels, de la Fondation Eclipse visant à développer un environnement de production de logiciels libres qui soit extensible, universel et polyvalent, en s’appuyant principalement sur Java et JEE.

Le choix d'STS repose essentiellement sur sa gratuité, sa facilité d'utilisation, sa puissance de développement et surtout ses nombreux plugins (bibliothèques additives).

#### **Serveur d’application :**

L'utilisation d'un serveur Java EE est obligatoire pour le développement de pages Web dynamiques en Java EE. Un serveur HTTP classique reçoit des requêtes HTTP et renvoie des réponses mais il ne connait pas les Servlets, les JSP... Il est donc essentiel d'utiliser un programme appelé moteur de Servlets qui est contenu dans le serveur Java EE et qui permet de pallier ce manque.

*  **Serveur TOMCAT (version 9.0) :** est un conteneur de Servlet J2EE qui implémente la spécification des Servlets et des JSP.

Nous avons choisi d'utiliser Tomcat, car c'est un serveur léger, gratuit, libre, multiplateforme et assez complet pour notre sujet. De plus, Tomcat à bien répondu à notre besoin contrairement GlasFish par ces multiple versions.

*  **Apache** est le serveur Web le plus utilisé sur Internet. Dans une architecture en production, il est recommandé d'utiliser un serveur Web en frontal d'un serveur d'applications. Ces recommandations sont également appliquées dans le cas de l'utilisation d'un conteneur Web comme Tomcat. L'utilisation d'un serveur Web en frontal est nécessaire dans ce projet pour des raisons de performance, de sécurité et de flexibilité.

#### **Système de gestion de base de données :**

*  : **My Structured Query Language (MySQL connector-java .0.16) :**

Langage de requêtes structuré est un système de gestion de bases de données relationnelles dédiées Open source. Il est très rapide, fiable et facile à utiliser et gratuit et supporté par un large éventail d'outils.

MySQL est surtout installé pour les applications Web. Ce SGBD est solide et utilisé par de grands groupes spécialisés dans l'Internet.

### **Technologies :**

#### **Langage de programmation :**

Java Entreprise Edition (JEE) a été conçu comme un environnement pour développer, déployer et exécuter des applications réparties pour le monde de l'entreprise. Ce contexte de l'entreprise se caractérise généralement par la nécessité d'assurer des niveaux de qualité de service tels que la sûreté de fonctionnement, la résistance à des charges d'exécution importantes ou encore la sécurité.

Pour cela, nous avons eu recours, pour le développement de notre application au langage de programmation J2EE. C’est la garantie de portabilité qui fait la réussite de Java dans les architectures client-serveur en facilitant la migration entre serveurs, très difficile pour les gros systèmes. D’autre part JAVA est sécurisée, il a été conçu pour être exploité dans des environnements serveur et distribués. Dans ce cadre, la sécurité n’a pas été négligeable. C’est le langage le plus adopté par les développeurs grâce à sa fiabilité et sa performance élevé.

#### **Environnement de développement :**

* **JDK** : Java Développent Kit Java est l'environnement dans lequel le code Java est compilé pour être transformé en byte-code afin que la machine virtuelle JAVA (JVM) puisse l'interpréter. Les composants primaires du JDK sont une sélection d'outils de programmation.
* **Javac :** le compilateur, qui convertit le code source en fichier .class (contenant le bytecode Java).
* **Jar :** c’est lui qui se charge de mettre l'ensemble des fichiers class sous forme d'un paquetage unique dans un fichier JAR pour les archiver.
* **Javadoc :** c’est bien le générateur de documentation, qui génère automatiquement la documentation à partir des commentaires du code source.
* **Jdb :** le débogueur des applications java.
* **JPA :** c’est une API Java Persistance qui repose sur des entités annotés et sur un gestionnaire de ces entités (EntityManager) qui propose des fonctionnalités pour les manipuler (ajout, modification suppression, recherche). Ce gestionnaire est responsable de la gestion de l'état des entités et de leur persistance dans la base de données.
* **EJB :** Les Entreprise Java Bean ou EJB sont des composants serveurs donc non visuels qui respectent les spécifications d'un modèle édité par Sun. Ces spécifications définissent une architecture, un environnement d'exécution et un ensemble d'API.
*  : **Spring** est un framework open source J2EE pour applications 3-tiers. Il facilite l'intégration des différentes technologies qui a priori sont pour le moins hétérogènes.
*  **Java Server Faces** est un framework de développement d’application Web en Java permettant de respecter le modèle d’architecture MVC et basé sur des composants côté présentation.
*  : **Hibernate** est un framework open source gérant la persistance des objets en base de données relationnelle.
*  **: JSP** est l’acronyme de Java Server Page. C’est une technologie java qui permet la génération des pages web dynamiques. La technologie JSP permet de séparer la présentation sous forme de code HTML et les traitements sous formes de classes.
* **HTML** (HyperText Markup Language) c’est un langage de balisage servant à écrire des pages pour le World Wide Web. Conçu à l'origine comme le successeur d'HTML, XHTML se fonde sur la syntaxe définie par XML, plus récente, mais plus exigeante que celle définie par SGML sur laquelle repose HTML : il s'agit en effet de présenter un contenu affichable non seulement par les ordinateurs classiques, mais également sans trop de dégradation par des PDA bien moins puissants.
*  : **CSS** (Cascading Style Sheets) c’est un langage informatique qui sert à décrire la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.

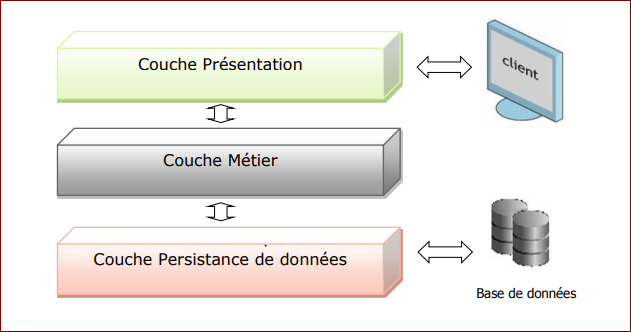
## **Les Architectures :**

### **ARCHITECTURE APPLICATIVE :**

Notre application est constituée de trois couches. Le concept de partitionner une application en couches et de garder toute la logique de l’application dans ces couches distinctes et séparées, a été introduite bien avant l’approche orientée objet. Ainsi une application est divisée en trois couches logiques, chacune traitant des fonctions spécifiques :

* **Présentation** : interface usager et présentation.
* **Logique du logiciel à produire** (besoins, services de l’entreprise) : les règlements de l’entreprise et la logique de l’application.
* **Logique des données** : Base de données et intégration des services de l’entreprise.

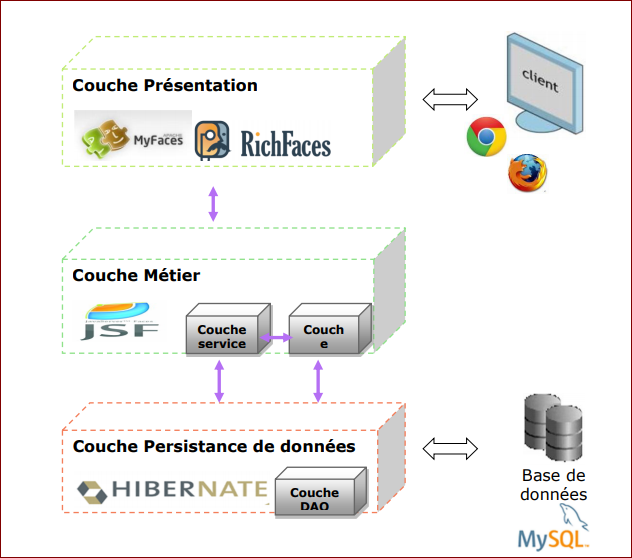
Ce concept nous permet de créer des composants indépendants et de les déployer sur des plates-formes différentes.



**Figure 9: L'architecture applicative du 'e-learning'**

### **ARCHITECTURE TECHNIQUE :**

Ici, nous avons représenté l’architecture technique de site c’est à dire chaque couche avec sa technologie et son environnement de développement



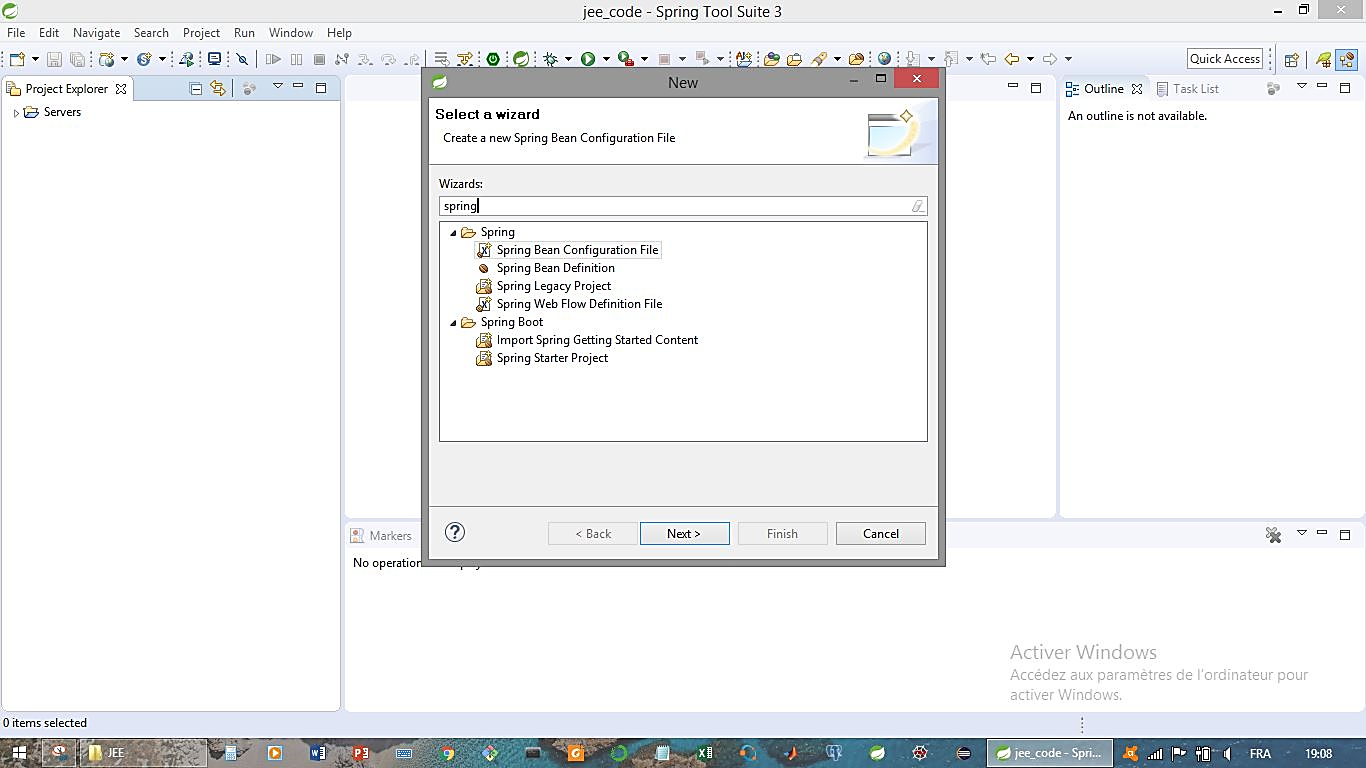
**Figure 10:L'architecture technique du 'e-learning'**

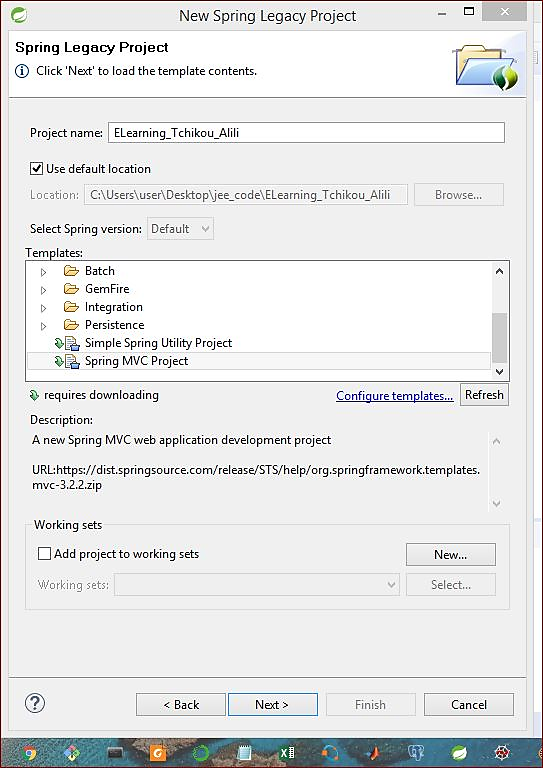
## **Description du code :**

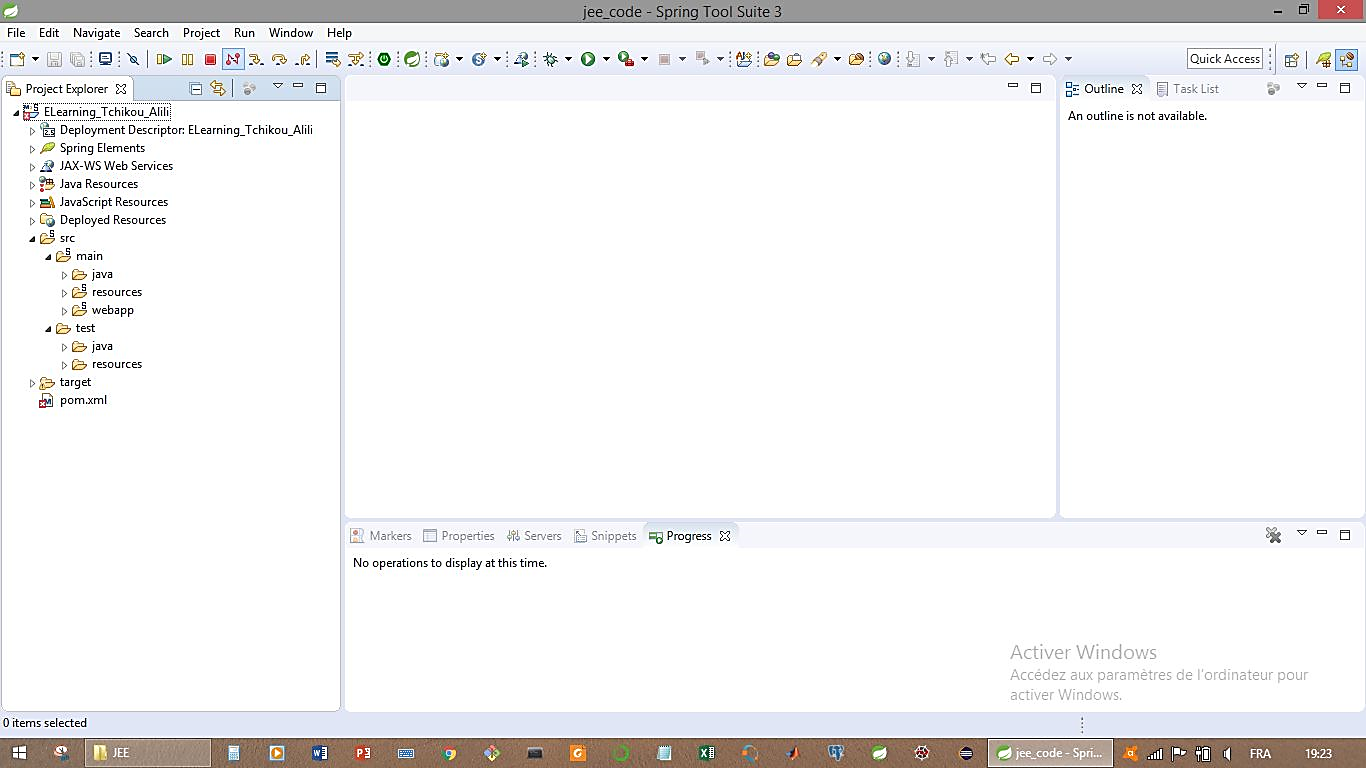
La démarche suivie est celle des TPs présentés durant le semestre comme suit :

### **Utilisation du framework Spring :**

* Création d’un projet Spring mvc en suivants les étapes citées dans le TP1 :



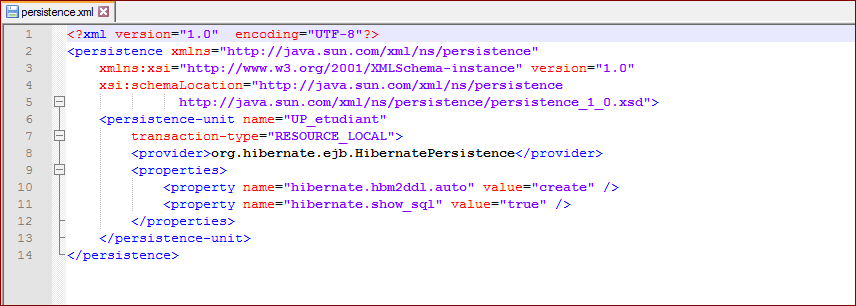




**Figure 11: Création d'un projet spring mvc**

### **Les configurations :**

* 1. Création du fichier ***persistence.xml*** dans le dossier **src/main/ressources/META-INF *;*** ce fichier contientl’unité de notre persistance et les types de transaction qui sont local. De plus, il contient 2 propriétés qui ont une relation avec d’hibernante lors de démarrage de notre application.

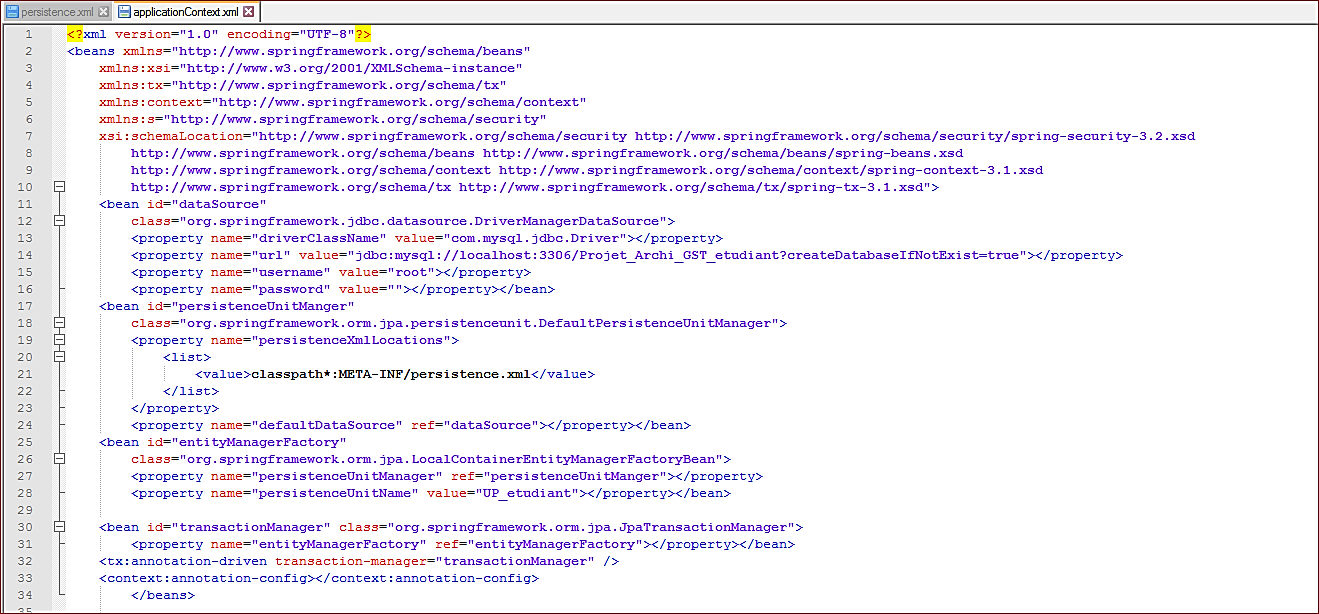


**Figure 12: le fichier persistence.xml**

* 1. Création de fichier ***ApplicationContext.xml*** dans le dossier **src/main/ressources**.

Ce fichier comprend notre définition, déclaration et configurations des **beans**, qui sont :

* Bean  ‘dataSource’ : qui contient la description et les propriétés de notre base comme l’url, nom de BDD, le mot de passe, ...etc.
* Bean ‘persistenceUnitManager’ c’est le gestionnaire de l’unité de persistance.Nous lui définissons l’unité de persistance quand a créé et lui fournir le dataSource.
* Bean ‘entityManaggerFactory’ : que nous lui fournit le persistenceUnitManager et l’unité de persistance.
* Bean ‘transactionManager’ : que nous lui fournit entityManaggerFactory.



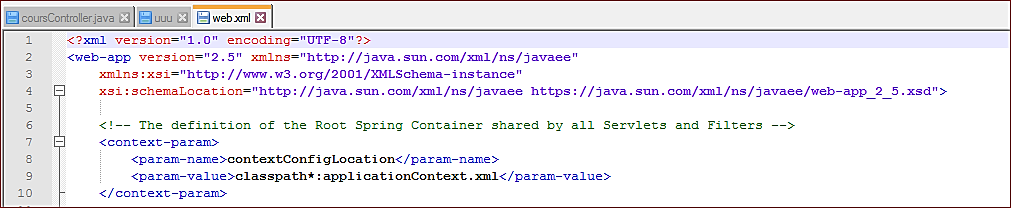
**Figure 13: le fichier ContextApplication.xml**

* 1. Configuration du fichier ***pom.xml :*** dans cette étape nous ajoutons les dépendances nécessaires en respectant les versions associées qui sont comme suit :



**Figure 14: Les dépendances ajoutées dans le fichier pom.xml**

* 1. Et pour cela, dans le fichier ***web.xml,*** nous devons changer le chemin de fichier de configuration dans le context param à notre fichier de configuration personnalisé comme suit :



**Figure 15: fichier web.xml**

### **L’injection des dépendances :**

#### **3-1- Création des entités :**

* Création des différentes classes dans un package nommé **com.Elearning.mvc.entites** en se basant sur le diagramme de classe qui sont : cours.java, Assister.java, Enseignant.java, Enseigner.java, Etudiant.java, Examen.java, Question.java, Role.java et Utilisateur.java
* Effectuer le Mapping Objet/Relationnel de la structure par l’utilisation des annotations JPA assez complet qui sont :

|  |  |
| --- | --- |
| Annotations | Rôles |
| @javax.persistence.Table | Préciser le nom de la table concernée par le mapping |
| @javax.persistence.Column | Associer un champ de la table à la propriété (à utiliser sur un getter) |
| @javax.persistence.Id | Associer un champ de la table à la propriété en tant que clé primaire (à utiliser sur un getter) |
| @javax.persistence.GeneratedValue | Demander la génération automatique de la clé primaire au besoin |

**Tableau 1: Les annontations JPA de base**

* L'annotation **@javax.persistence.Table** permet de lier l'entité à une table de la base de données. Par défaut, l'entité est liée à la table de la base de données correspondant au nom de la classe de l'entité. Si ce nom est différent alors l'utilisation de l'annotation @Table est obligatoire.
* Dans notre cas, nous avons laissé par défaut.
* L'annotation **@javax.persistence.Column** permet d'associer un membre de l'entité à une colonne de la table. Par défaut, les champs de l'entité sont liés aux champs de la table dont les noms correspondent. Si ces noms sont différents alors l'utilisation de l'annotation @Column est obligatoire.
* Dans notre cas, nous avons laissé par défaut.
* Il faut obligatoirement définir une des propriétés de la classe avec l'annotation @Id pour déclarer comme étant la clé primaire de la table. Cette annotation peut marquer soit le champ de la classe concernée soit le getter de la propriété. L'utilisation de l'un ou l'autre précise au gestionnaire s'il doit se baser sur les champs ou les getters pour déterminer les associations entre l'entité et les champs de la table.

La clé primaire peut être générée automatiquement en utilisant l'annotation @javax.persistence.GeneratedValue. Cette annotation possède plusieurs attributs, le plus important est strategy qui précise le type de générateur à utiliser : TABLE, SEQUENCE, IDENTITY ou AUTO.

* La valeur par défaut est AUTO. Il laisse l'implémentation générer la valeur de la clé primaire.
* Les annotations de gestion des relations entre les entités sont 4 types nécessite une table d'association entre les 2 tables et utilisera @JoinTable pour préciser cette table comme suit :

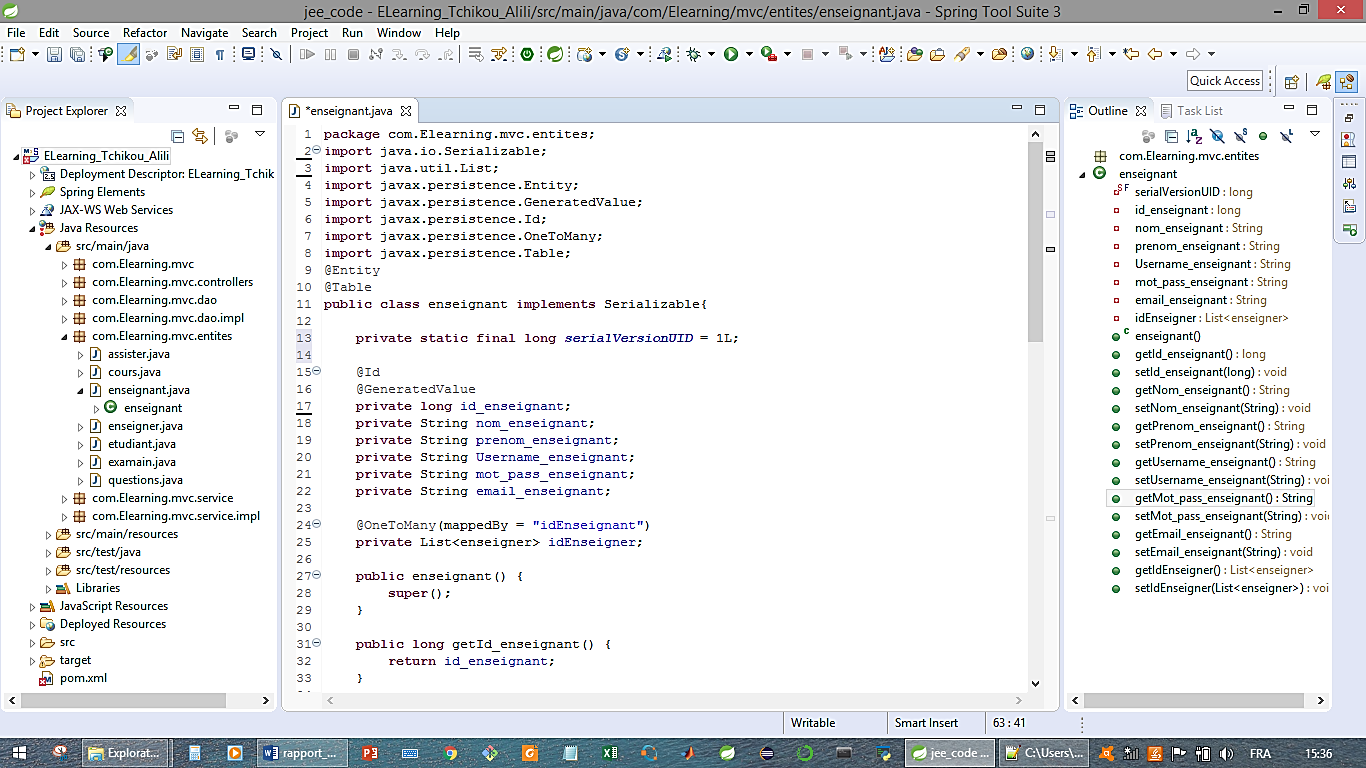
|  |
| --- |
| Annotations |
| @OneToOne 1 --- 1 |
| @OneToMany 1 --- \* |
| @ManyToOne \* --- 1 |
| @ManyToMany \* --- \* |

**Tableau 2: Les annontations de relations entre les tables**

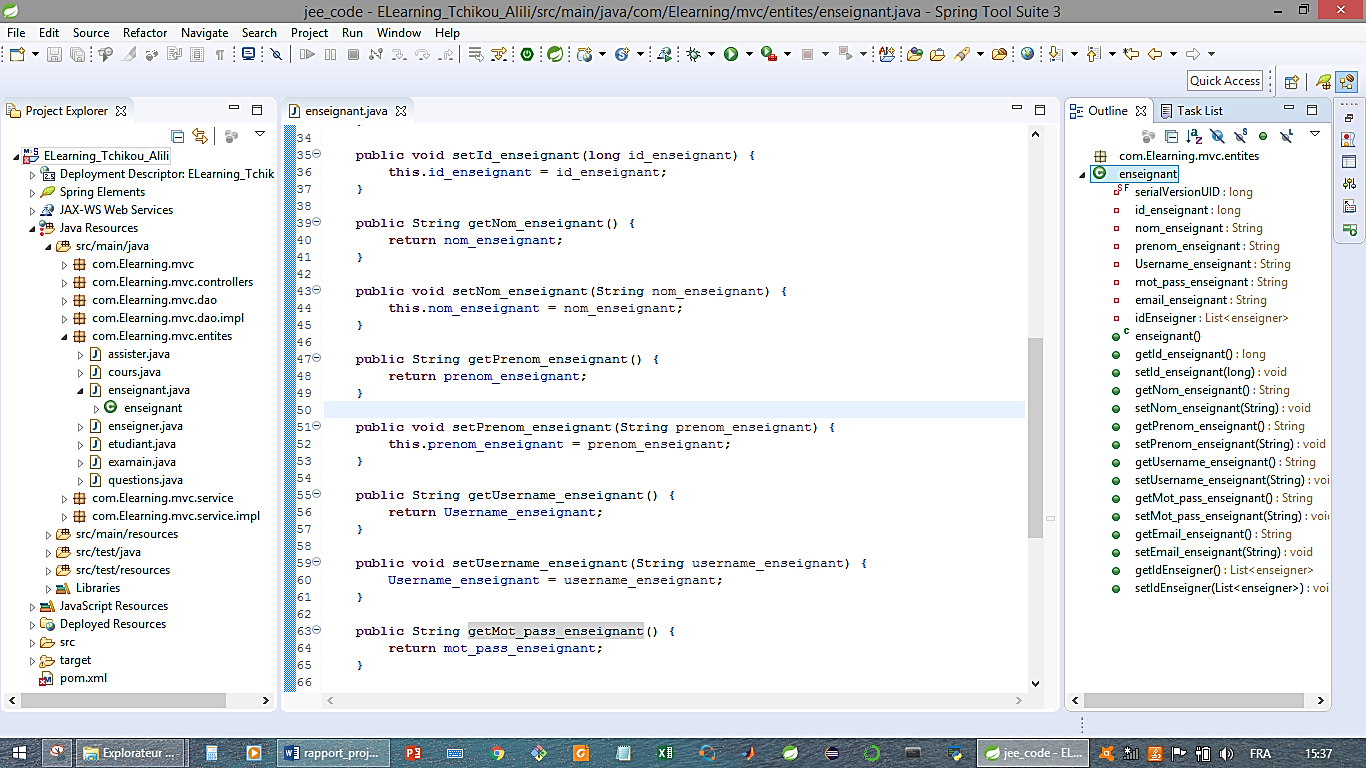
* @JoinColumn : Sur un attribut, précisera avec quelle clé étrangère est faite la jointure.

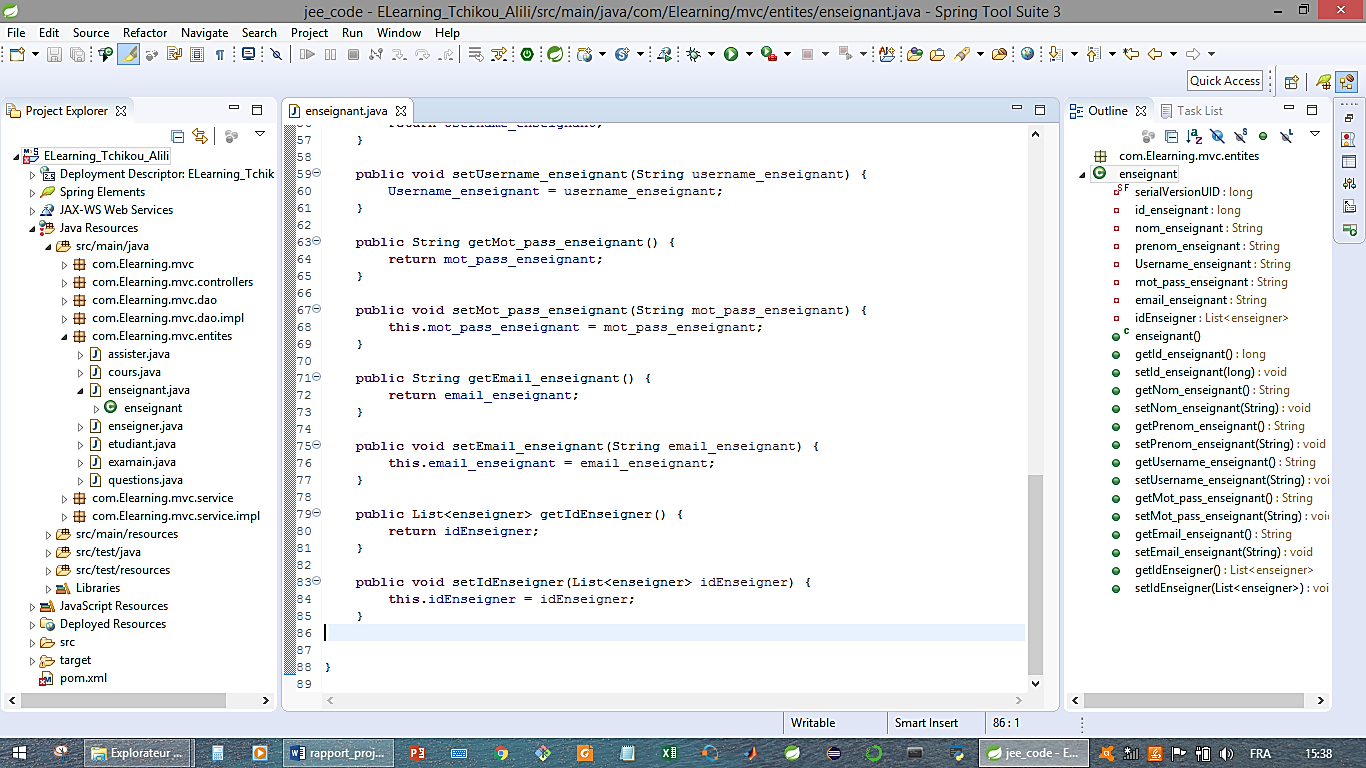
NB : Toutes les classes Implémentent la classe **Serializable** (java.io.Serializable).

A titre d’exemple, les différentes captures suivantes (figure -14) représentent le code de l’entité Enseignant.java :



* Les setters et les getters :

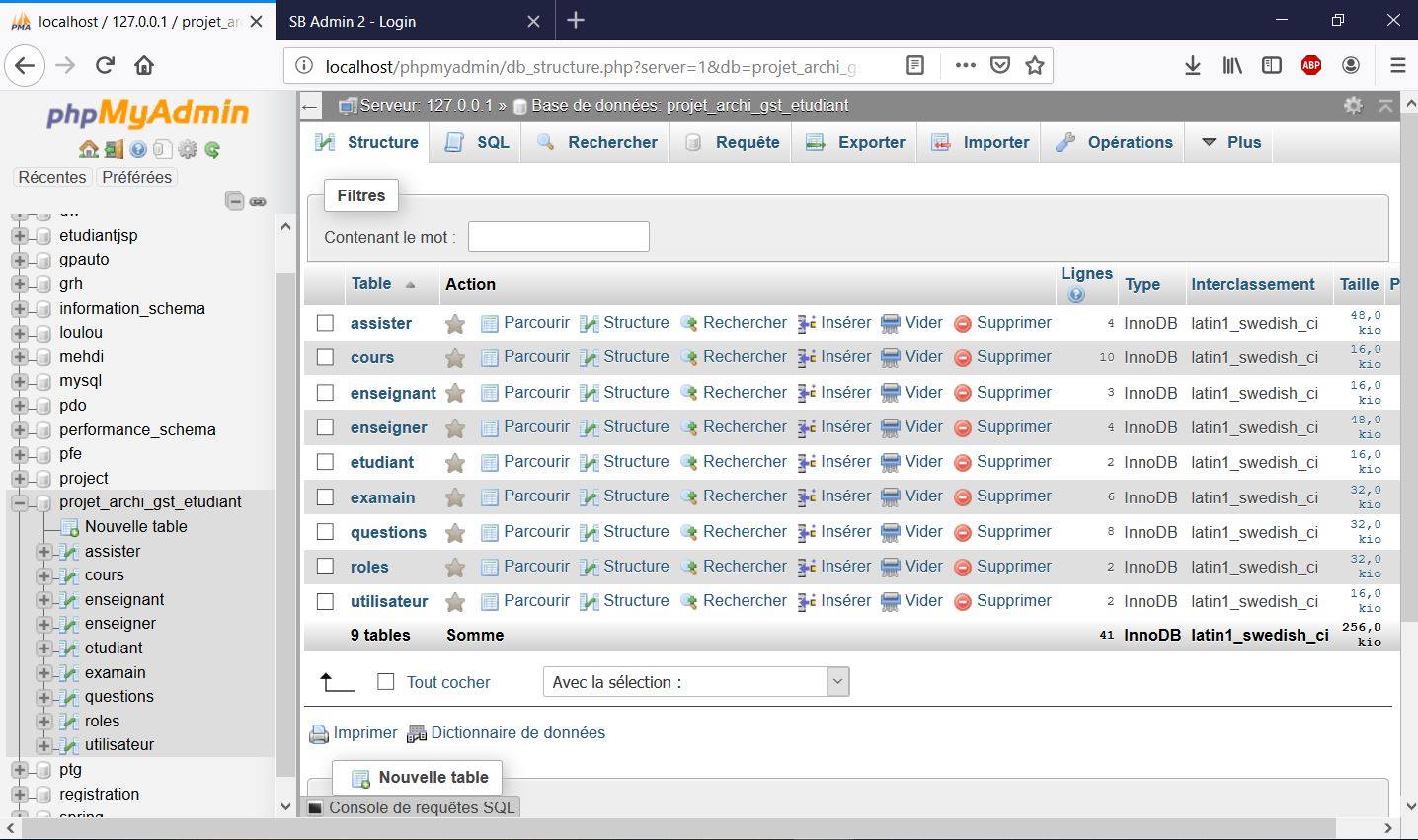




**Figure 16: Le code de l’entité Enseignant.java**

#### **3-2- Exécution et génération des tables :**

En exécutant notre projet avec le serveur Tomcat, nous générons les tables comme suit :

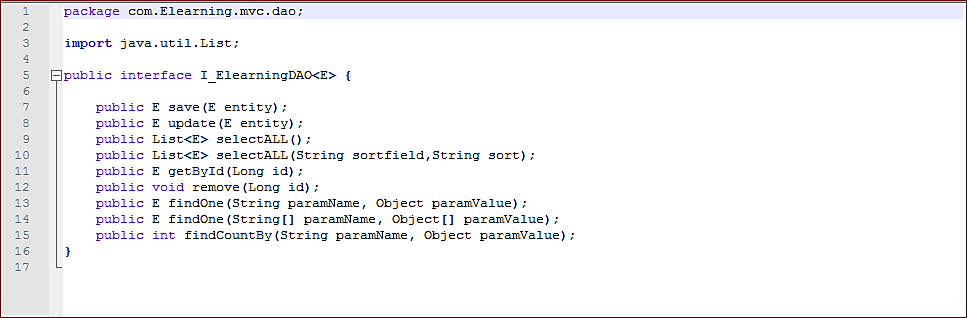
****

**Figure 17: Les tables dans notre base**

#### **3-3- création de la couche DAO :**

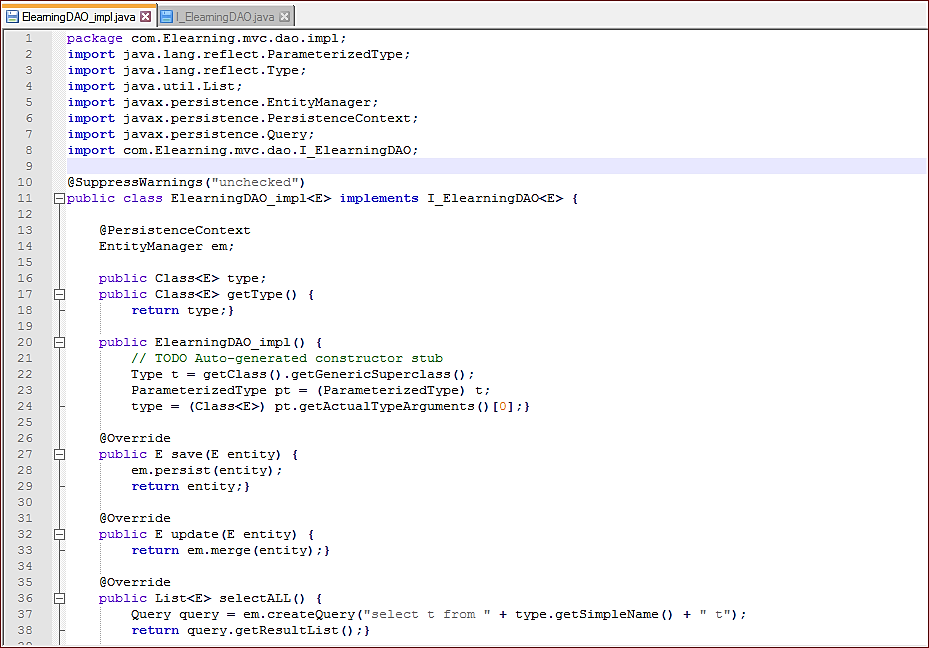
Cette étape permet de créer des classes contenant les opérations d’accès à la base de données. La conception consiste à créer une interface définissant les méthodes d’accès à la base de données.

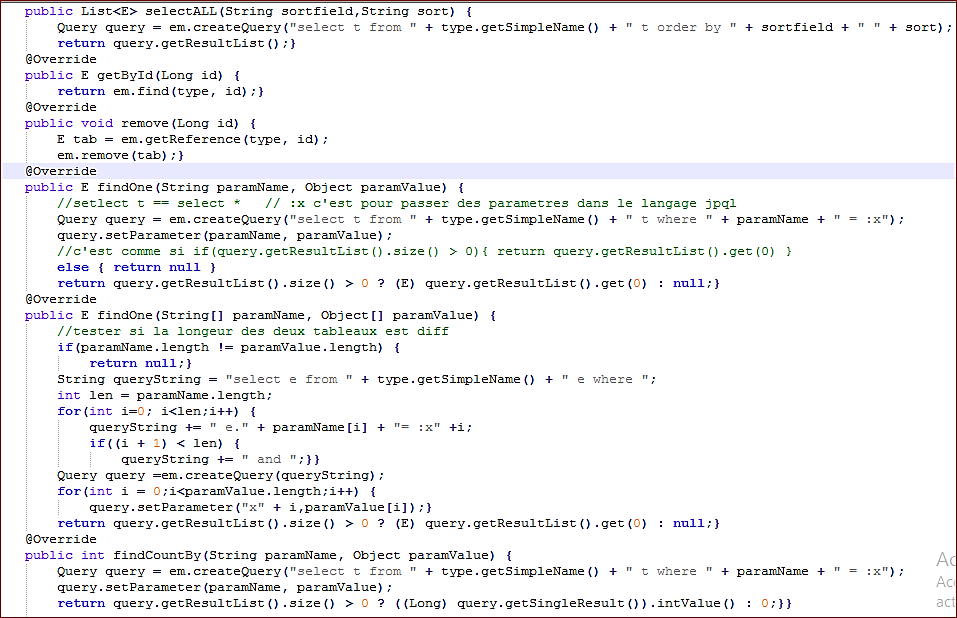
D’abord, nous commençons par la création du package : **com.Elearning.mvc.dao**. Ensuite, nous créons l’interface générique I\_ElearningDAO qui regroupe les méthodes communes entre les DAO que nous avons.



**Figure 18: l’interface I\_ElearningDAO.java**

L’implémentation de cette interface en créant une classe générique ElearningDAO\_impl.java qui utilise un objet de l’interface « Entity Manager » qui définit les méthodes permettant la gestion des persistances des entités. On y trouve : persist(),find(), createQuery(), remove() et merge().

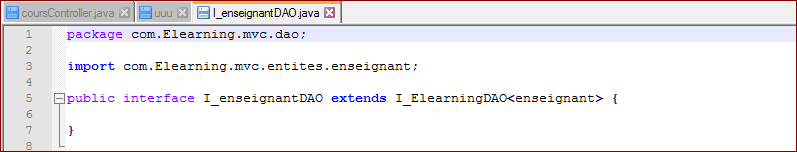




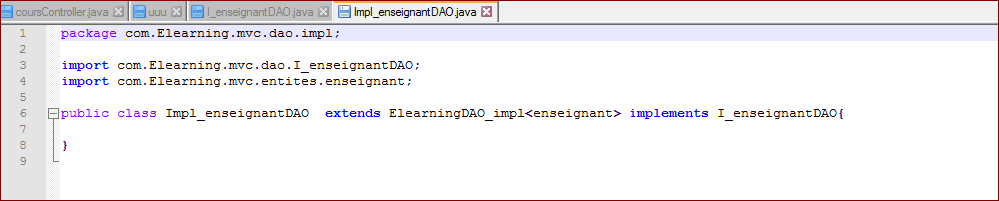
**Figure 19: code d’ElearningDao\_impl.java**

Par la suite, nous allons créer un DAO par entité (9 DAO pour les 9 entités).

par exemple, l’interface de DAO de l’entité enseignant et son implémentation qu’elle représentée par la figure suivante (figure 19) :



**Figure 20: L'interface de EnseignantDAO.java**



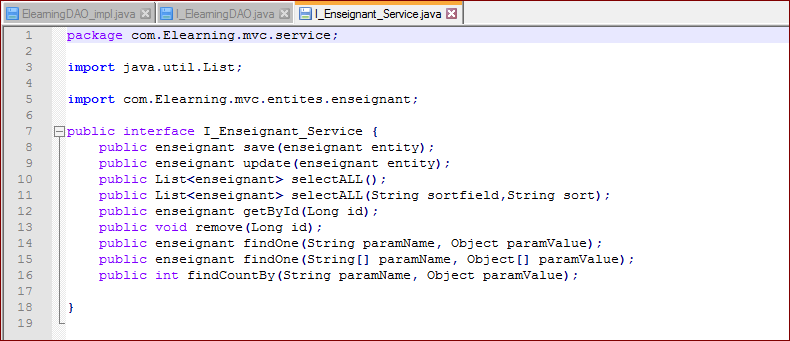
**Figure 21: Le code d'implémentation d’EnseignantDAO.java**

#### **3-4- Création de la couche métier et injection de dépendance :**

La couche métier est définie par un ensemble de classes permettant de réaliser les différents traitements de l’application et définir les règles métier. Ces classes font appel aux méthodes de la couche DAO.

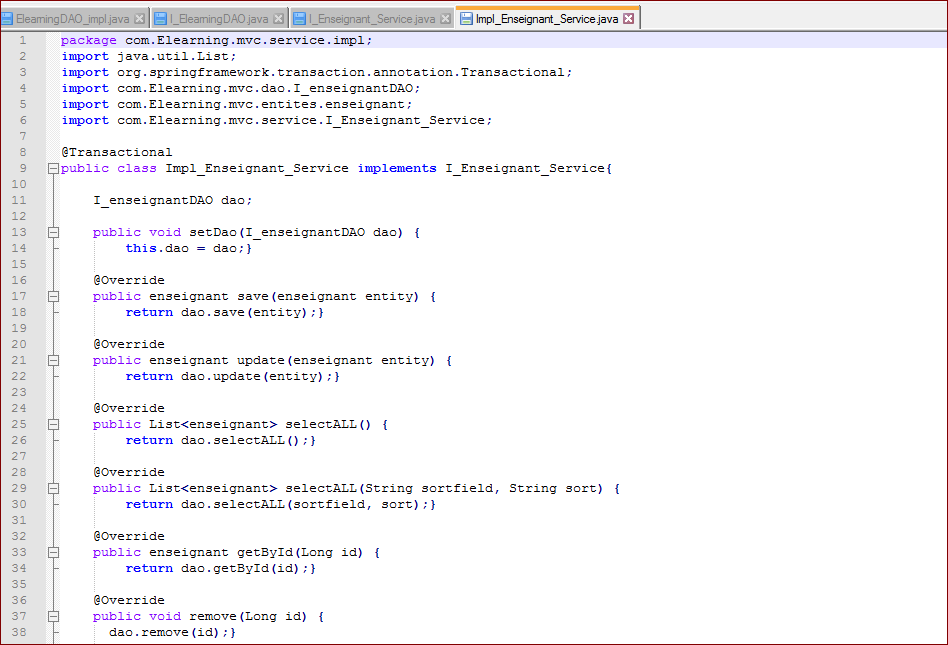
Nous commençons par la création du package **com.Elearning.mvc.service** et des interfaces tels que une interface par DAO qui définit les différents traitements,

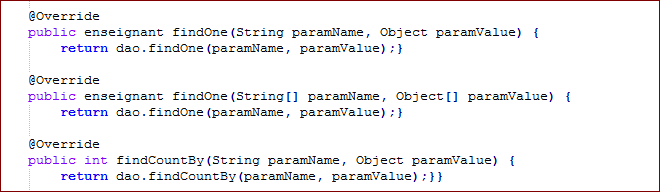
La figure suivante (figure 20) représente un exemple l’interface qu’elle est Enseignant\_Service.java :



**Figure 22: Création d'interface Enseingnant\_Service.java**

Puis nous implémentons cette interface, par la création d’une variable de type dao, définition d’un setter pour le DAO car elle sera utilisée par spring pour gérer les dépendances. Aussi, nous ajoutons des différentes méthodes transactionnelles.

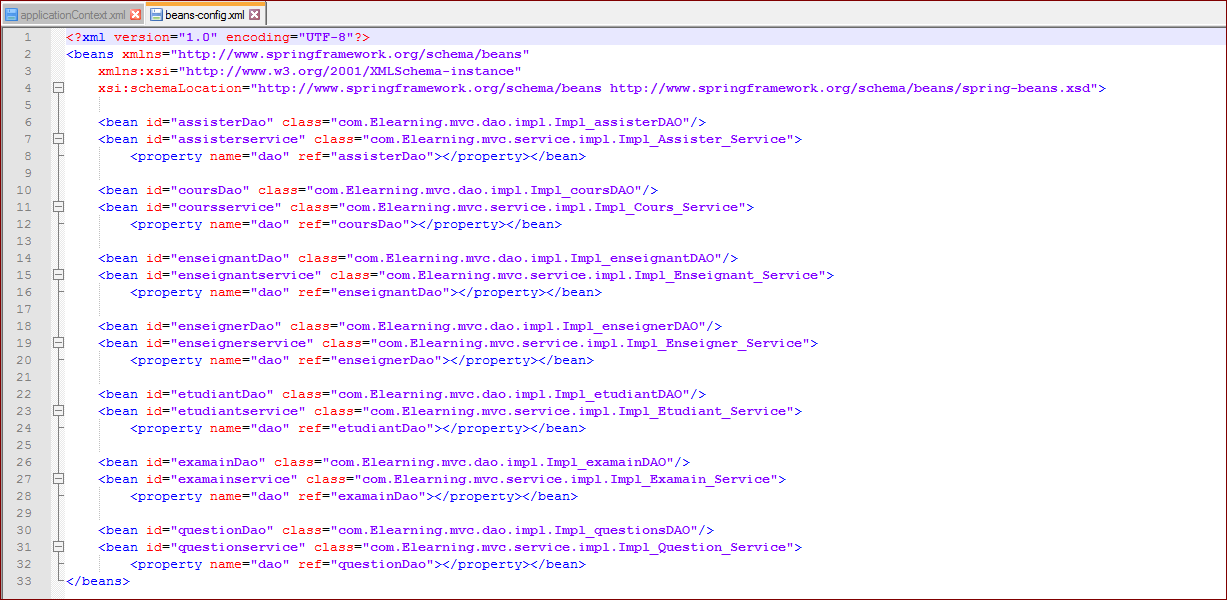




**Figure 23: Code d'Imp\_Enseignant\_Service.java**

Et à la fin, nous avons ajouté une configuration comme suit :

* Création d’un dossier ‘config’ dans ***src/main/ressources*.**
* Création d’un fichier ‘beans\_config.xml’ dans ***src/main/ressources/config.***
* Définition des différents beans qui contient en lui donnant un id et la classe de DAO et de service qui servent à faire injection automatiquement des propriétés lorsque notre projet démarre.( spring nous facilite beaucoup notre modifications).



**Figure 24: Le fichier beans\_config.xml**

* Importer le path des beans dans **ApplicationContext.xml :**



**Figure 25: Déclaration des beans dans ApplicationContext.xml**

### **Les contrôleurs (servelet) et les pages JSP :**

Dans cette partie nous entamons la partie web de notre système E-learning. Cette couche est constituée de deux parties : la partie front-end et la partie backend. Dans la première partie nous définissons les contrôleurs et dans la deuxième partie les pages web (pages JSP).

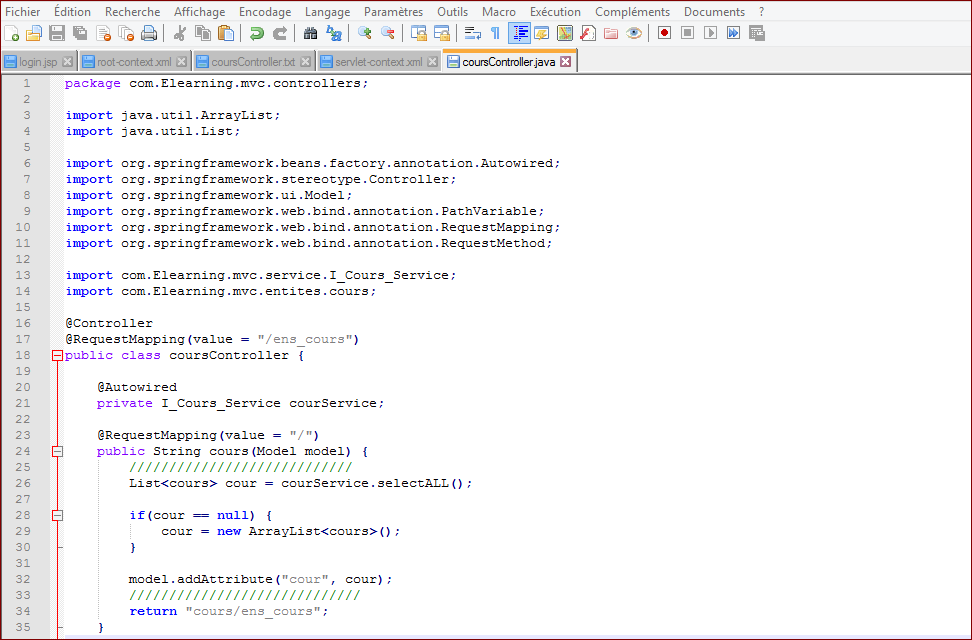
#### **4-1- les contrôleurs :**

 Nous créons un contrôleur chargé des traitements des requêtes. Chaque méthode du contrôleur se chargera de traiter les requêtes déclenchées par les URL spécifiées par l’annotation :

« @RequestMapping ».

 Ce contrôleur utilise un objet de la couche métier pour effectuer quelques traitements (l’ajout, la modification et la suppression d’un cours). En utilisant l’injection de dépendance de Spring pour injecter une valeur à cet objet.

* La méthode décrit ci-dessus va charger le modèle (la page avec une liste d'objet cour sélectionné par courService.selectALL() ; depuis notre base de donnée. Puis injecter au modèle par model.addAttribute("cour", cour).
* De plus, la méthode public String cours(Model model) est faite pour le remplissage de tableau.



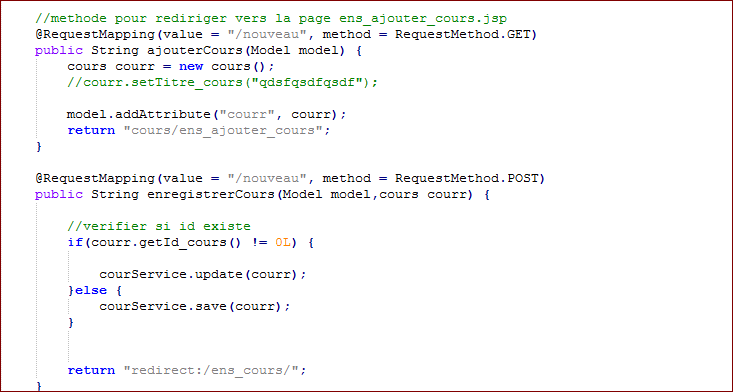
**Figure 26: le contrôleur cours**

* **L’ajout d’un cours :** pour cela, la méthode décrite va charger le modèle avec un nouvel objet qui sera rempli automatiquement.

Par exemple l'attribut de cette ligne :

**<f:input path="titre\_cours" class="form-control" placeholder="Titre du cour" />**

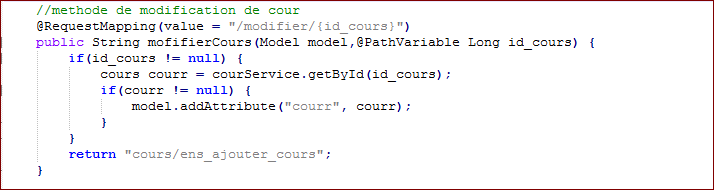
Elle charge automatiquement l'attribut titre\_cours de l'objet cour par path="titre\_cours"



**Figure 27: La méthode ajouterCours.java**

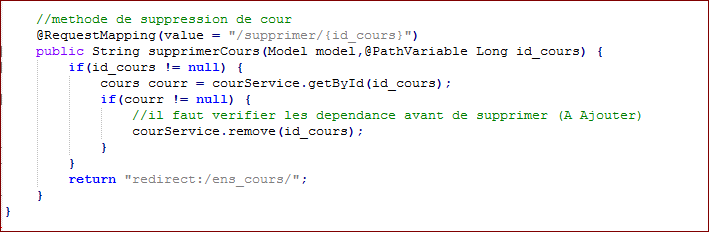
Et nous continuons avec le même principe pour la modification et la suppression des cours

* **Modifier un cours :**



**Figure 28: La méthode ModifierCours.java**

* **Supprimer cours :**



**Figure 29: La méthode supprimerCours.java**

 Les méthodes après avoir exécuté leurs instructions, nous afficherons la page «cours.jsp»

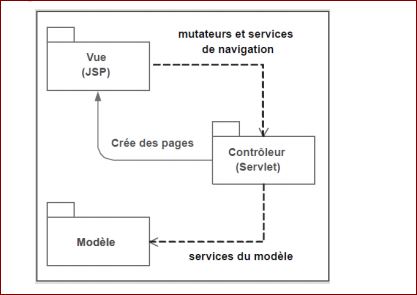
#### **4-2- Les pages JSP :**

La technologie des pages Jsp est considérée la recommandation du modèle MVC. Une page jsp est un fichier texte contenant des balises html et des balises jsp qui permettent l’utilisation direct des objets.

 EL (Expression Langage) : Le langage jsp est défini par un ensemble de balise. Un type particulier de balise est les EL (Expression Langage). Ces balises permettent via une syntaxe très épurée d’effectuer des tests basiques sur des expressions et de manipuler simplement des objets et des attributs dans une page sans nécessité d’utilisation du code java (le but est de faciliter la maintenance).

 Les objets implicites : sont des objets que nous pouvons manipuler dans une page jsp dans initialisation explicite. Le conteneur propose toute une série d’objets implicites : out, session, pageContext, Application, Request, Response, Etc.

 Bibliothèque JSTL : propose des balises préconçues pour permettre ‘ajouter plus de fonctionnalités aux pages jsp (par exemple la bibliothèque core).



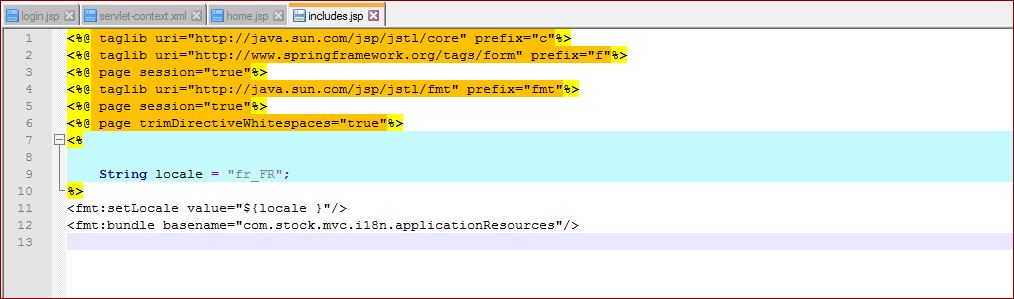
**Figure 30: le fonctionnement du modèle MVC**

* Configuration :

Nous avons créé un dossier **include** dans le dossier **views**, et nous avons mis tous nos includes (tag lib) dans un fichier ‘include.jsp’

La page ‘include.jsp’ comprend :

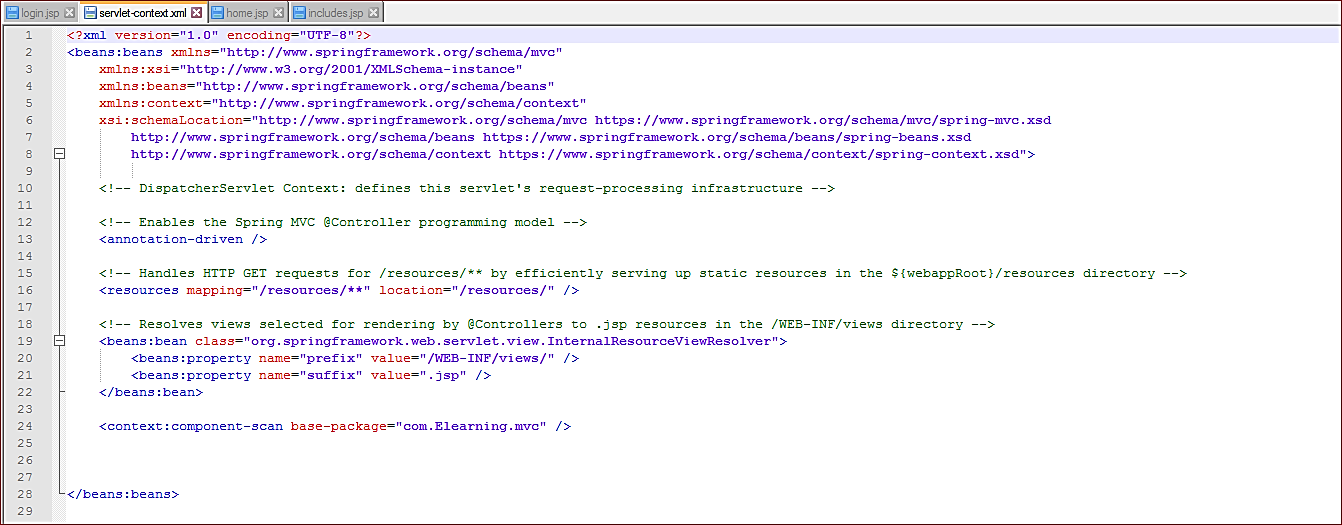
* Page Jstl core
* Jstl de spring framework
* Page session : pur stocker les informations dans nos sessions
* Le ‘prefix= fmt’ : pour l’internationalisation. (i18n ségnifie internationalisation)

****

**Figure 31: la page include.jsp**

Et pour cela, à chaque création d’une nouvelle page jsp, nous ajoutons le path d’include au début du code de page.

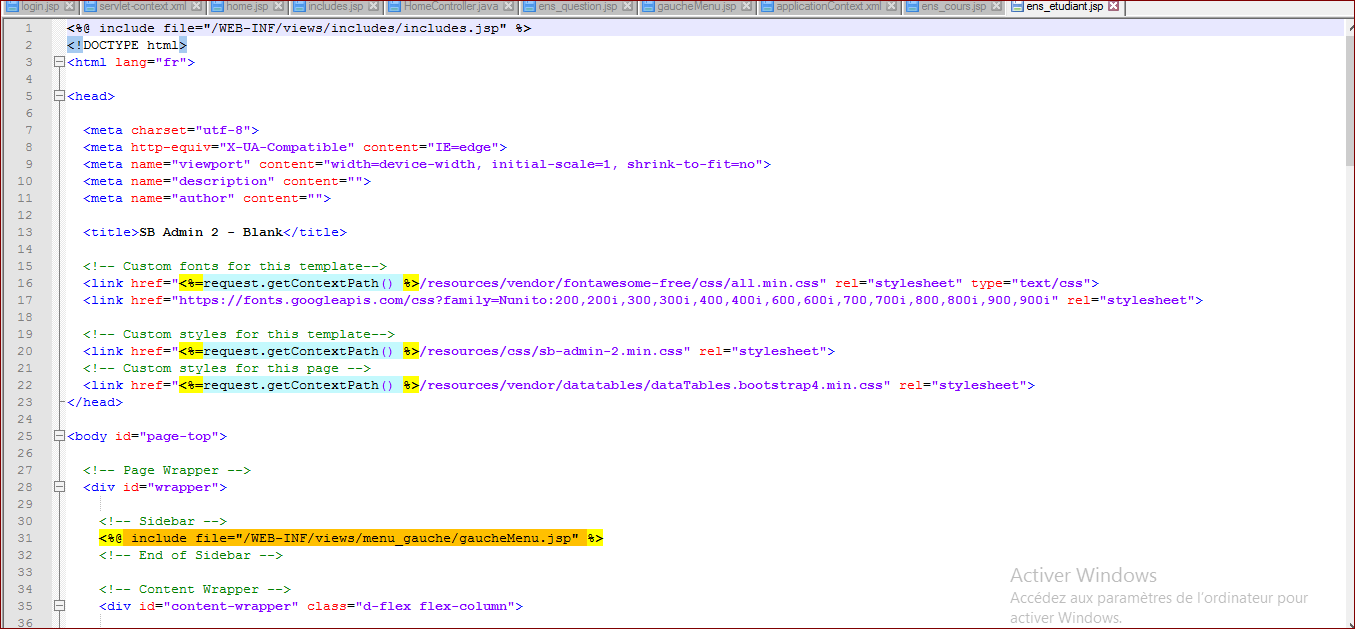
De plus, Le fichier servlet-context.xml : qui représente la programmation dynamique, c'est-à-dire on peut écrire des pages JSTL dans nos pages. Dans ce fichier, nous spécifions valeur de suffix en « **.jsp** »



**Figure 32: le fichier servelet-context.xml**

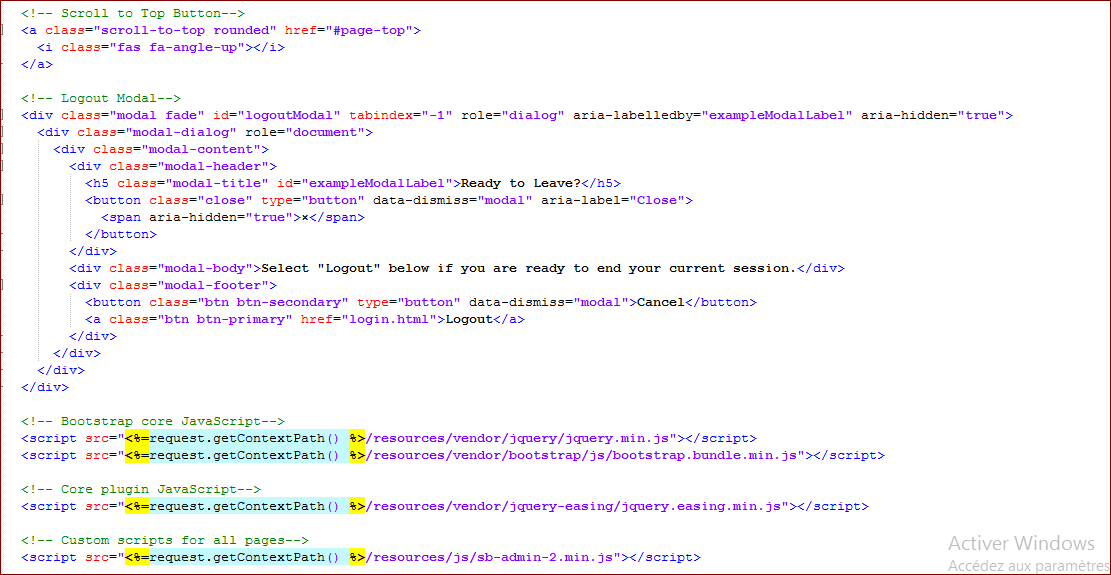
Dans notre système, prenons l’exemple de la page « Etudiant.jsp » qui affiche un formulaire d’ajout d’un étudiant et la liste des étudiants déjà inscrits.

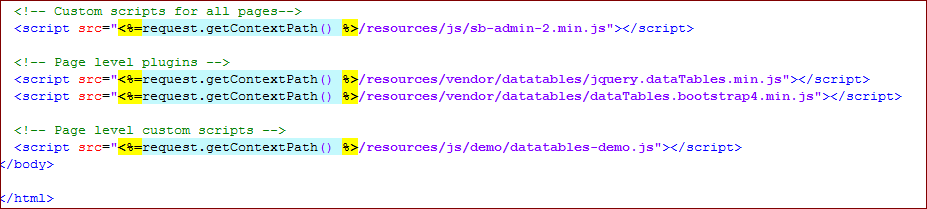
Les différentes captures suivantes (figure -28) représentent le code de la page en\_Etudiant.jsp :









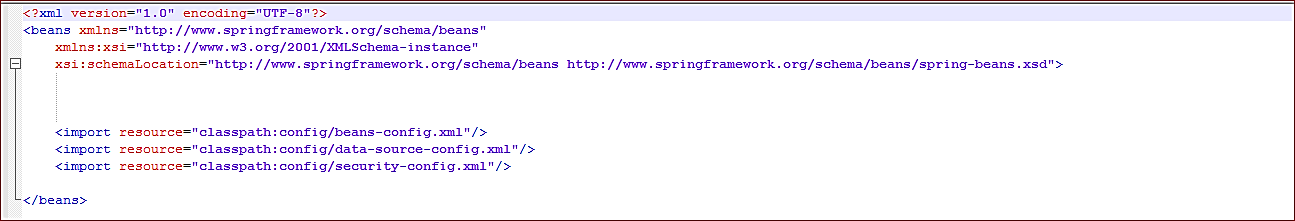


**Figure 33: la page jsp d'Etudiant**

### **Spring sécurité :**

Cette partie comprenne la sécurité pour quand on puisse gérer les utilisateurs et leurs rôles.

5-1. nous commençons par la configuration du fichier applicationContextext.xml : nous avons factorisé ce fichier par la séparation de ces définitions : ‘beans\_config’ , ‘data\_source\_config’ et la partie sécurité ‘security\_config’.



**Figure 34: configuration du fichier applicationContext.xml**

5-2. Ensuite, nous passerons à la configuration du fichier security-config.xml qui se trouve dans **src\main\resources\config**tel que :

* Nous avons définie au premier lieu la partie **http** en ajoutant les url quand nous voulons les intercepté.
* Pour chaque url, nous avons défini les autorisations c'est-à-dire chaque utilisateur qu’est ce qu’il fait.
* En deuxième lieu, nous avons spécifié la forme **login** par son url et le fail bien sûr en cas le nom d’utilisateur ou le mot de passe erroné.
* Par la suite, nous définissons l’access denied handler : c’est la spécification de la page d’erreur (/403) c'est-à-dire le droit d’accès à cette page est refusé.
* La page logout : c’est à quelle page on va le rediriger si nous voulons déconnecter.
* A la fin, nous arrivons à la définition de l’authentification manager tel que : nous avons spécifie le fournisseur d’authentification en utilisant un service jdbc= data-source qui un fichier xml dans **src\main\resources\config**, qui contient la connection vers la base de donnée.



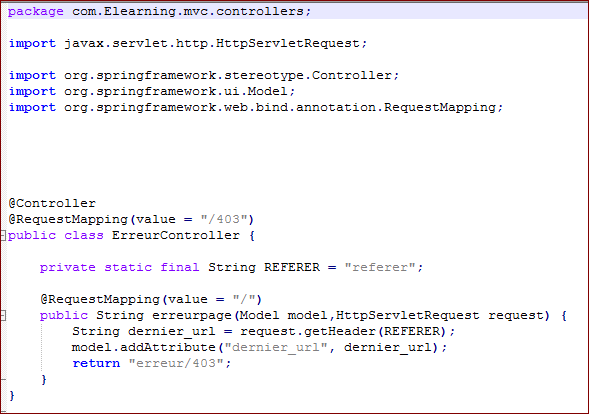
**Figure 35: le fichier data-source-config.xml**

* Et la requête de récupération des utilisateurs (il faut bien ajouter le mot actived pour spring sécurité fonctionne) et la liste des rôles par utilisateur.



**Figure 36: security-config.xml**

5-3. Création d’un autre Controller dans **src\main\java\com\Elearning\mvc\controllers** qui est ErreurController.java qui permet de gérer encore les erreur dont le mapping est 403.



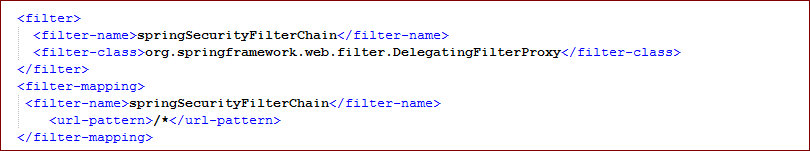
**Figure 37: erreurController.java**

**5-4.** Et pour bien assurer le fonctionnement de spring sécurité, nous devons ajouter 4 dépendances dans le fichier pom.xml



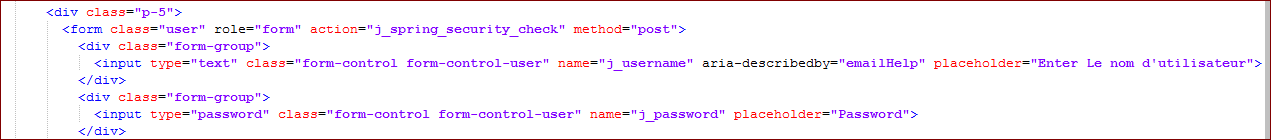
**Figure 38: dépendances spring ajoutées dans pom.xml**

5-5. ensuite, nous avons modifié un peu le fichier web.xml par l’ajout de filter c’est à dir faire le filtrage des urls.



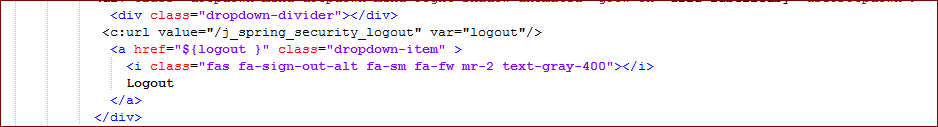
**Figure 39: l'ajout de filtre dans web.xml**

**5**-6. spécification de spring sécurité dans la page login.jsp



**Figure 40: spécification de spring sécurité dans la page login.jsp**

5-7. Enfin, pour le logout, nous avons créé un url qui est spring\_security\_logout

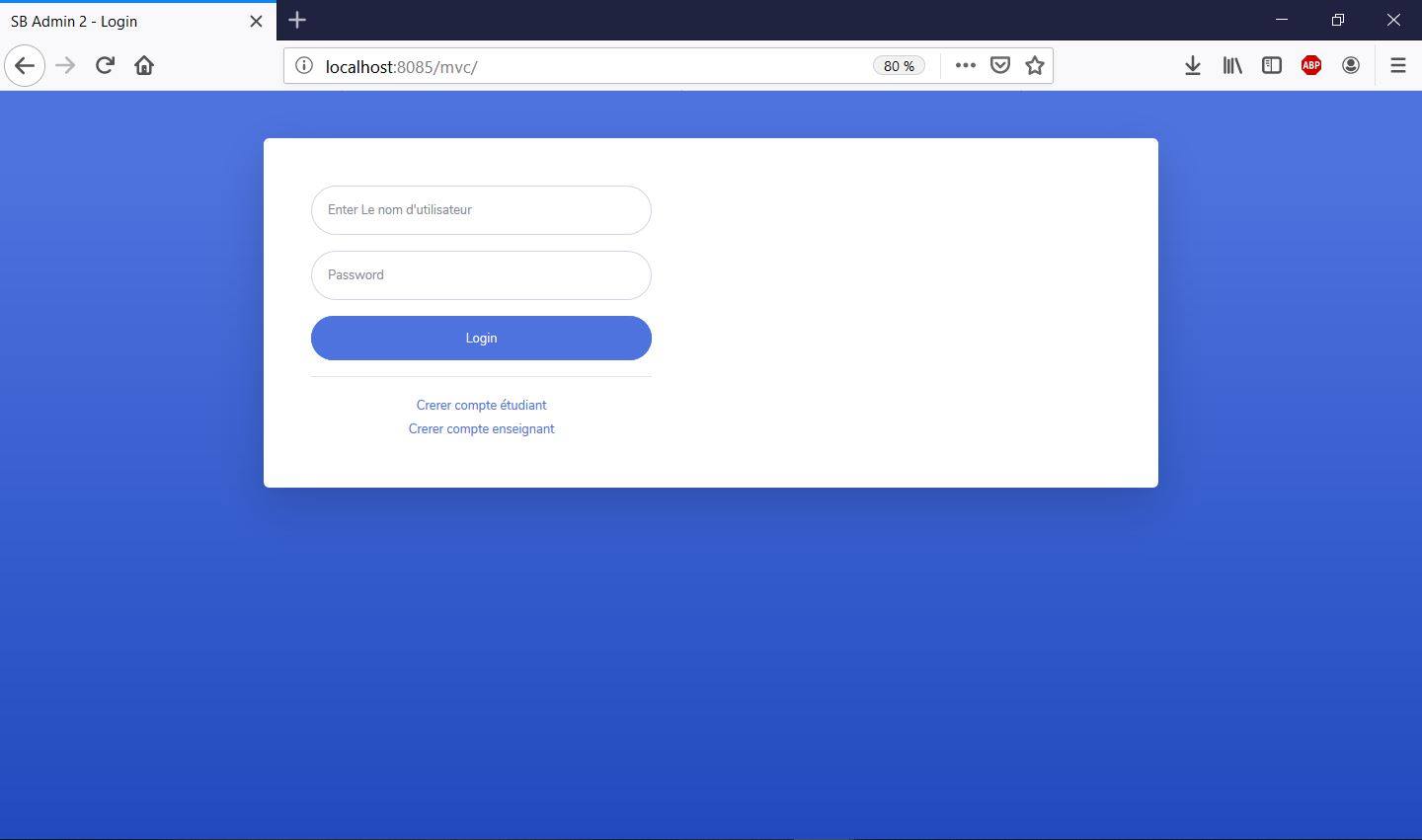


**Figure 41: url spring sécurité\_logout**

## **Démonstrations des interfaces :**

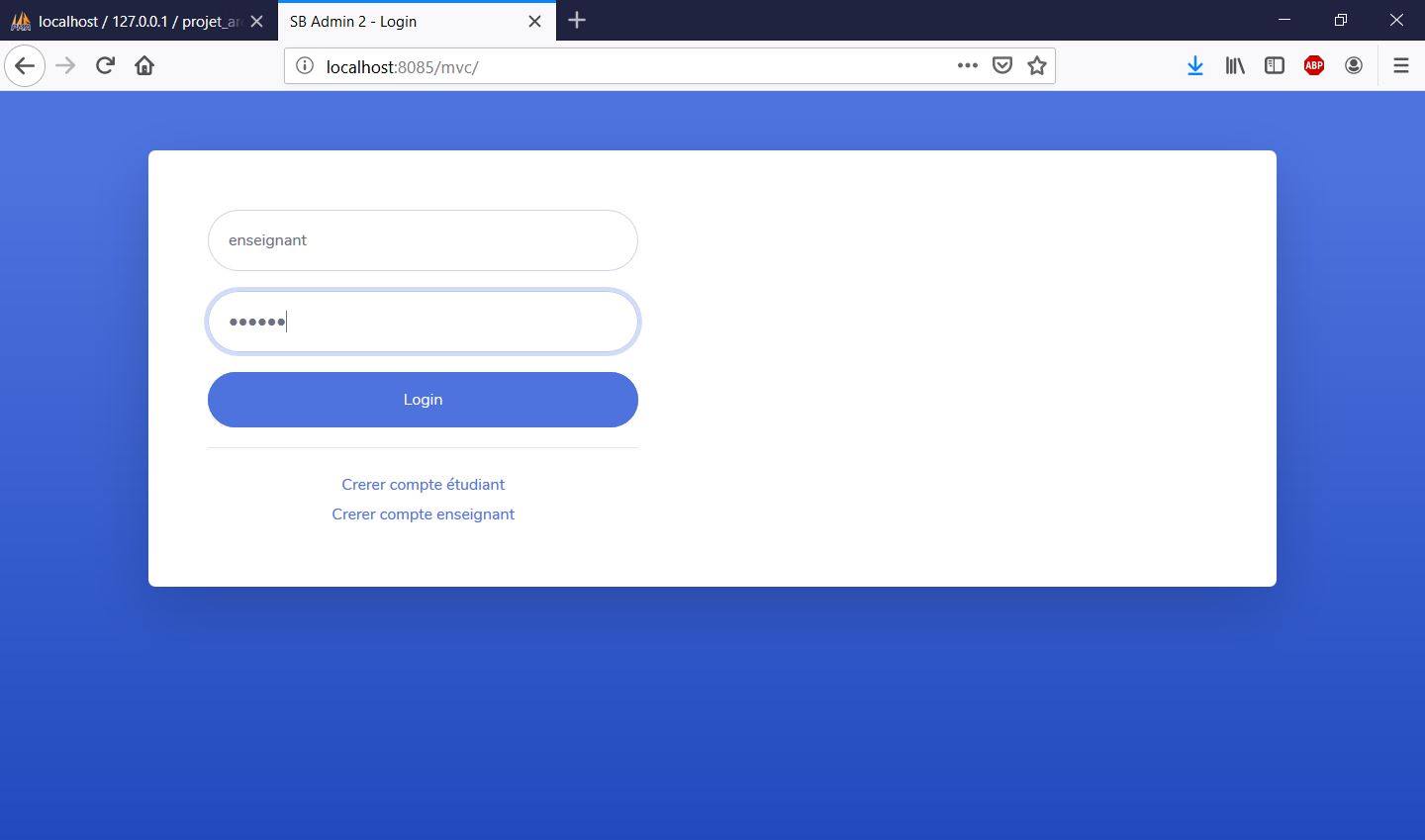
Nous allons présenter les différents scénarios applicatifs de l’application par des captures écran des principales interfaces réalisées dans notre site web.

### **L’interface d’Authentification dans le site :**

****

**Figure 42: l’interface login**

Et par celle nous introduit l’utilisateur et son mot de passe comme suit :

****

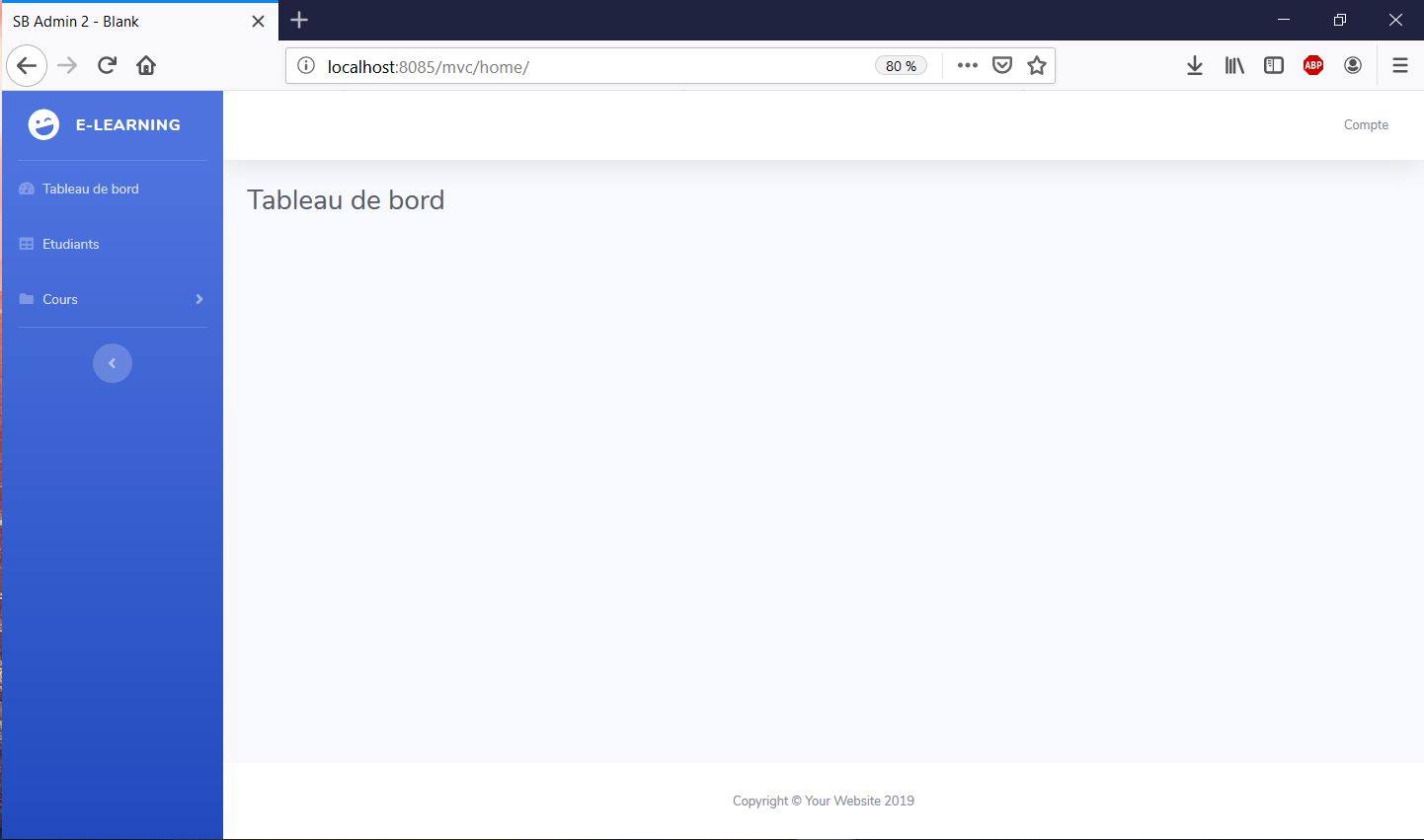
**Figure 43: login d'un enseignant**

Si le nom ou le mot de passe est erroné le système affiche un message d’erreur sur la même page, sinon

Ilva ouvrir l’espace de l’enseignant ou l’étudiant identifié.

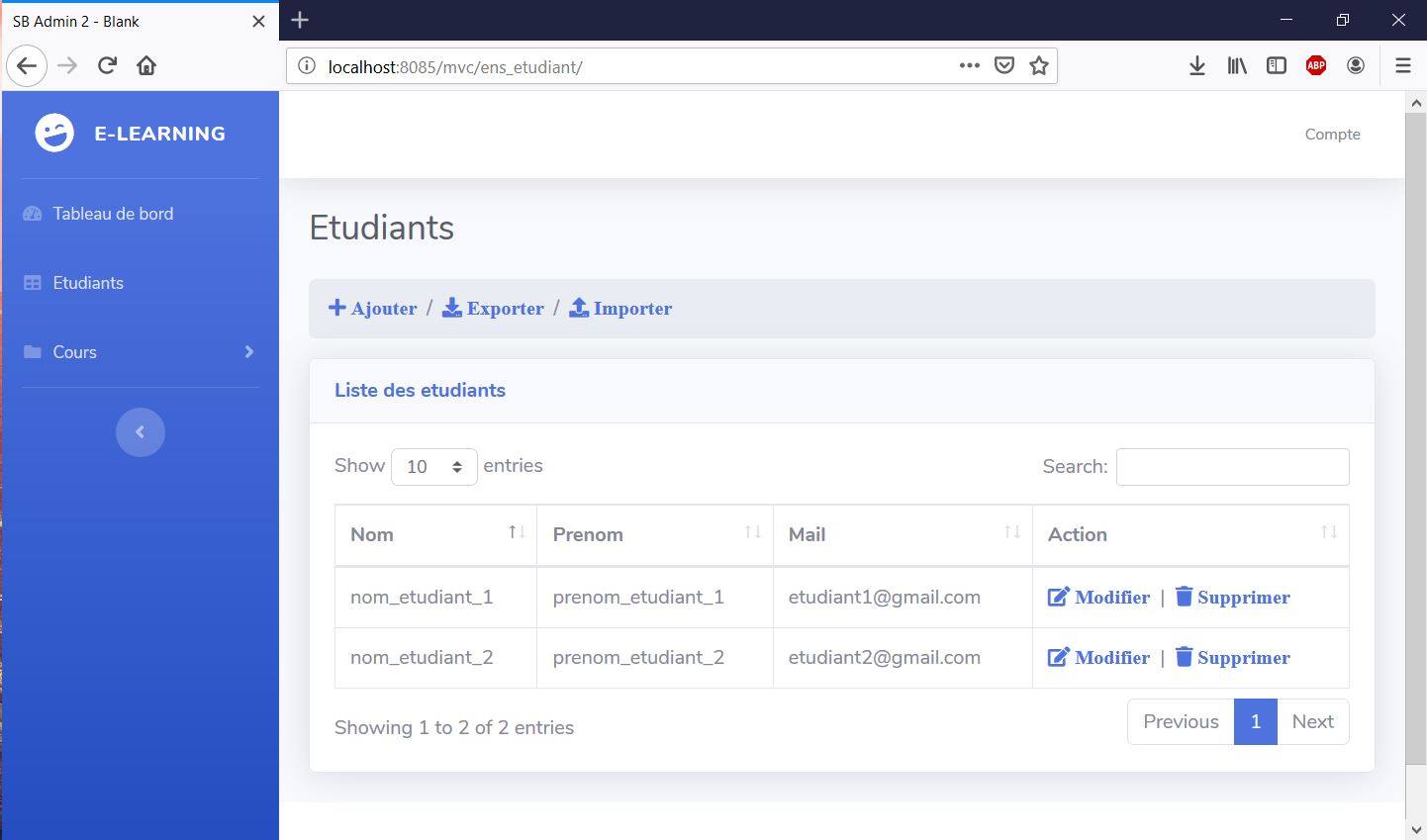
### **Espace Enseignant :**

#### **Interface principale d’un enseignant :**



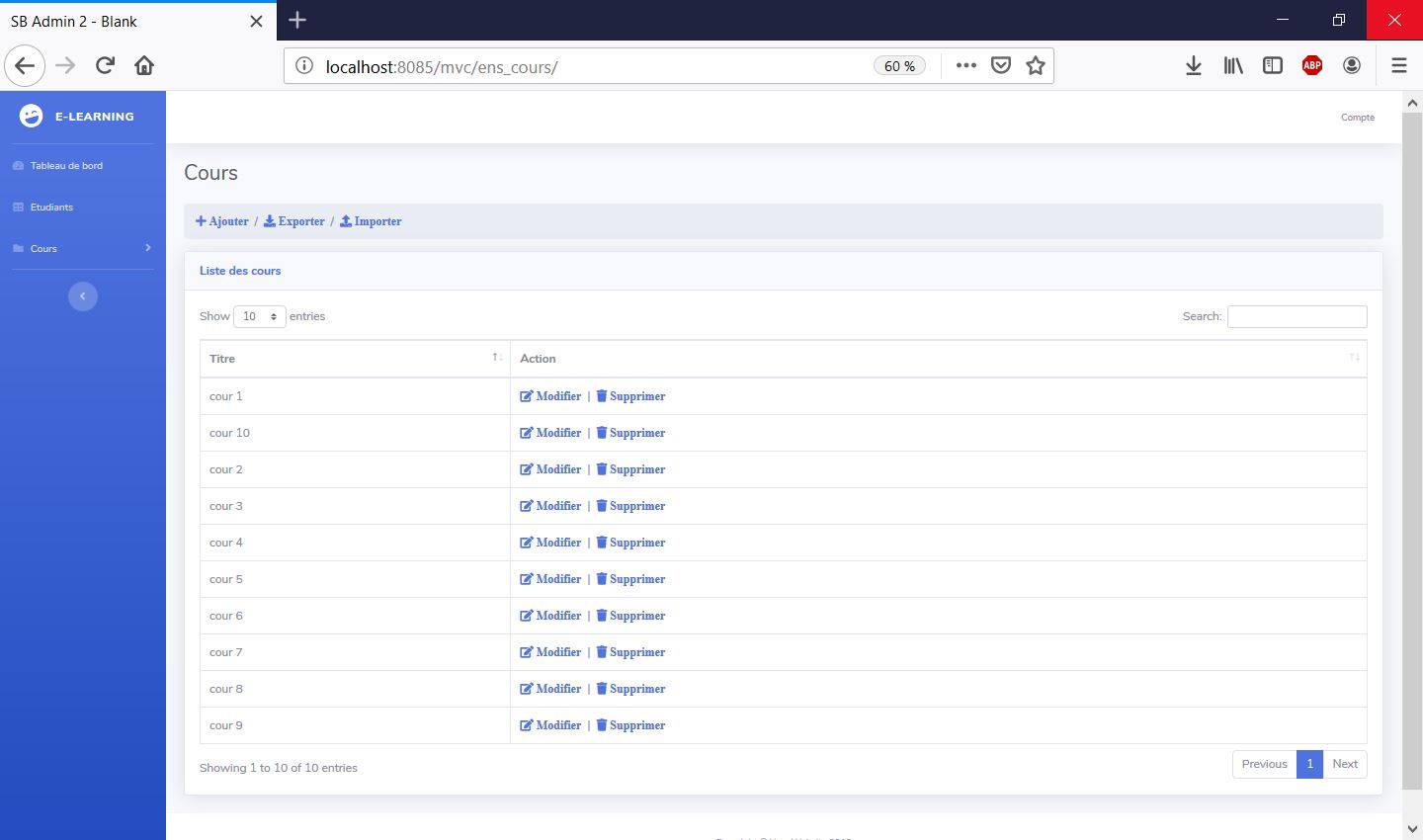
**Figure 44: interface de page d'accueil**

#### **Consultation la liste des étudiants :**



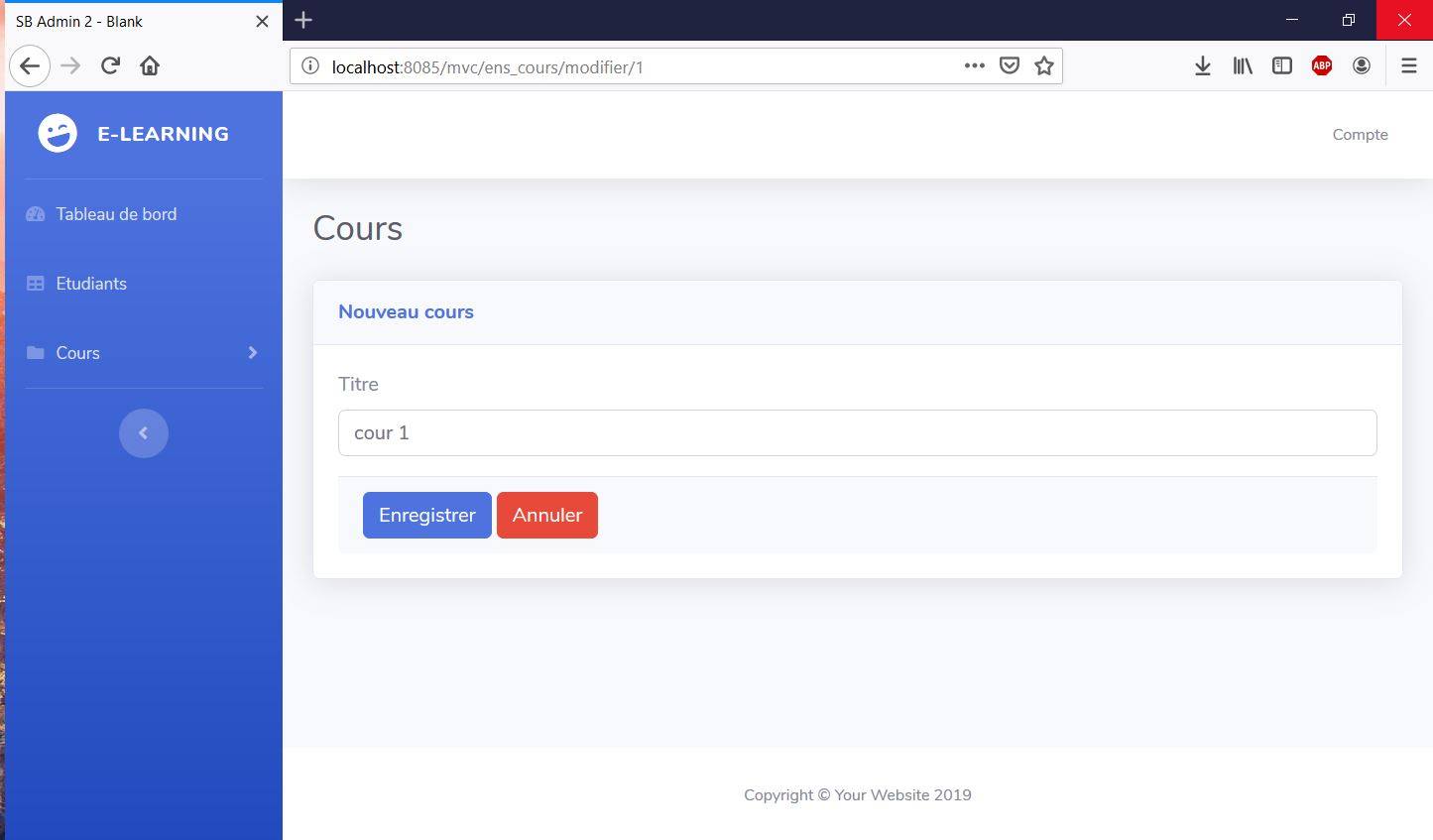
**Figure 45: consultation la liste des étudiants (leurs informations)**

#### **Affichage la liste des cours :**



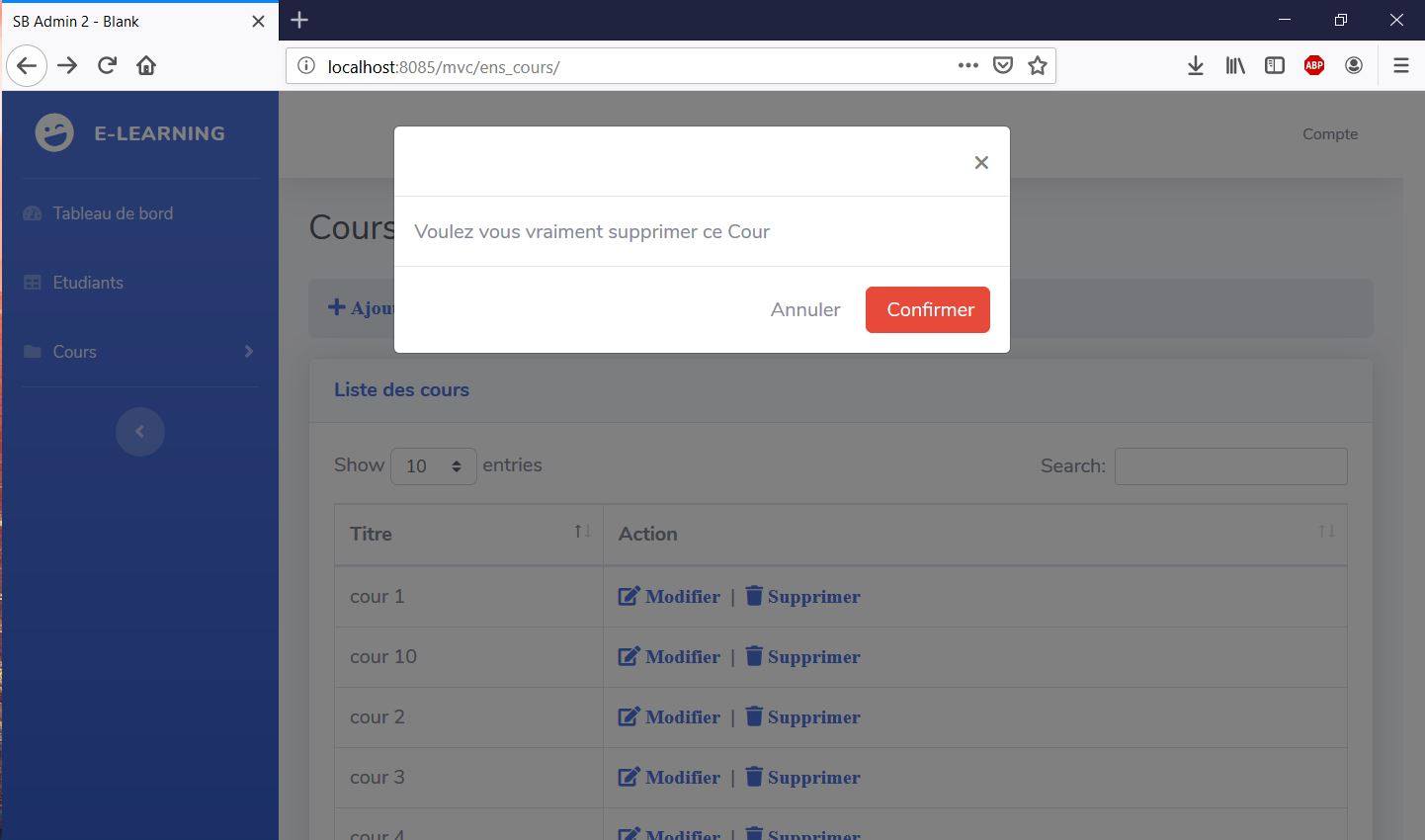
**Figure 46: La liste des cours**

#### **La modification d’un cours :**



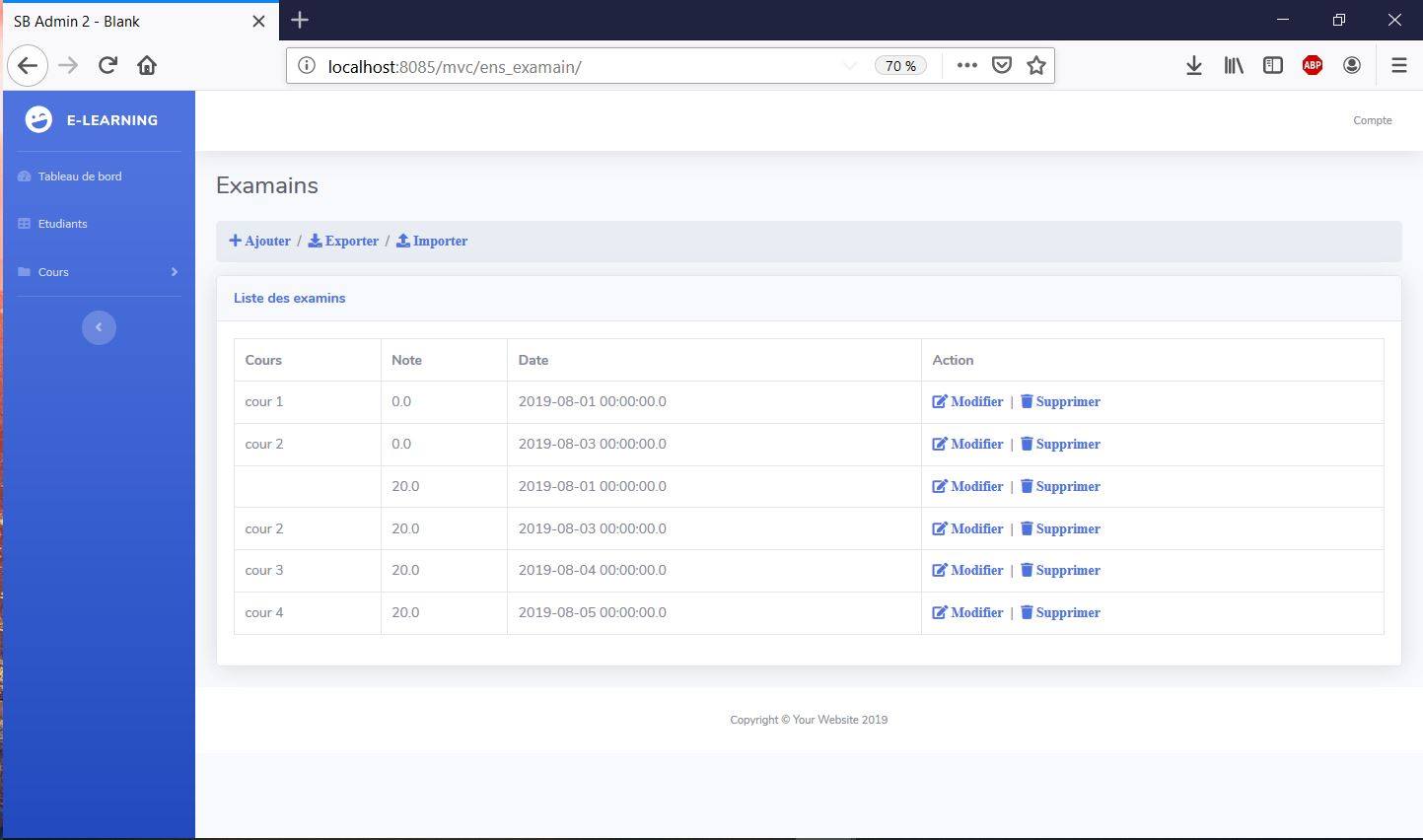
**Figure 47: la modification d'un cours**

#### **La suppression d’un cours :**



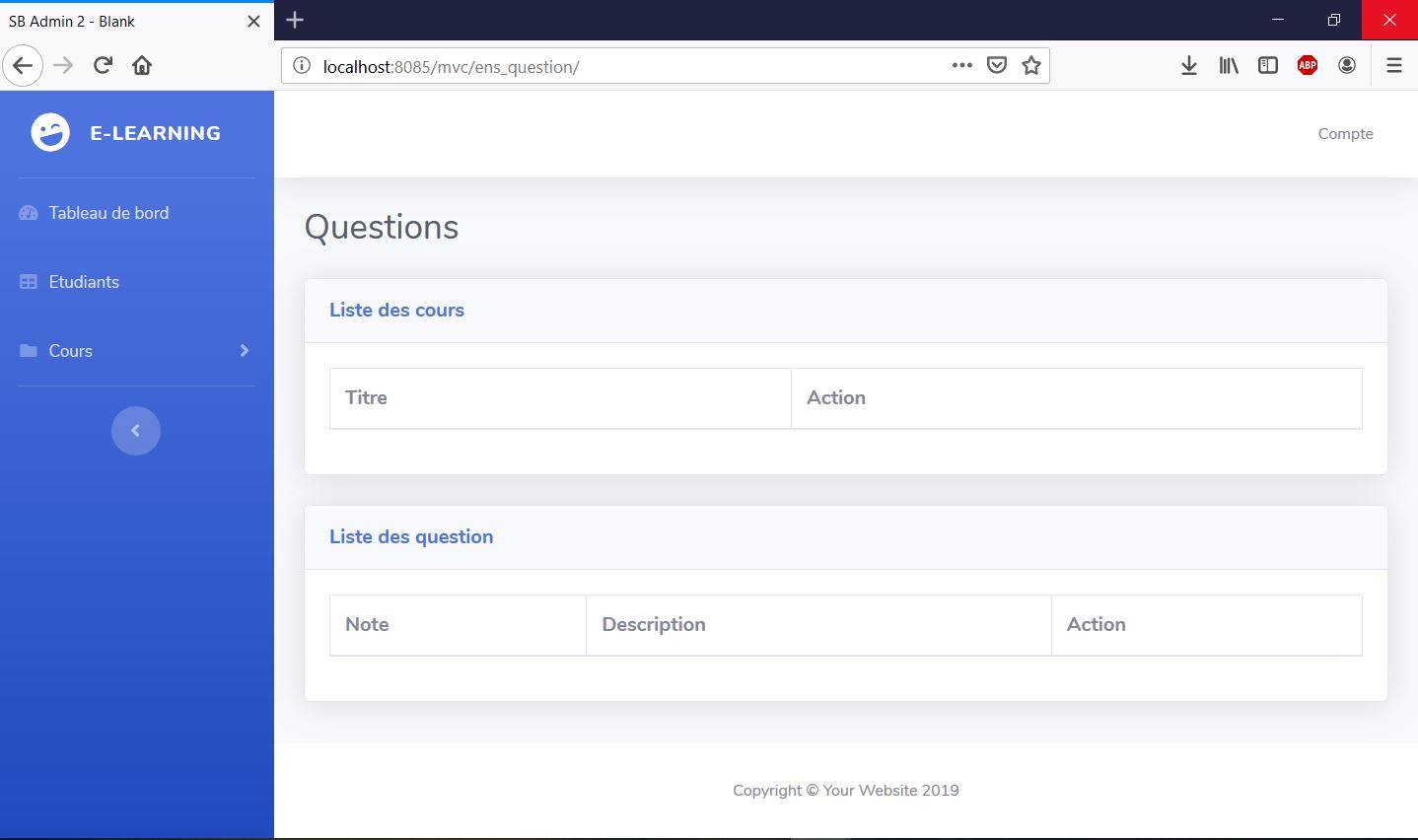
**Figure 48: la suppression d'un cours**

#### **Mise à jour des examens + consultation des notes des étudiants :**



**Figure 49: la mise à jour des examens**

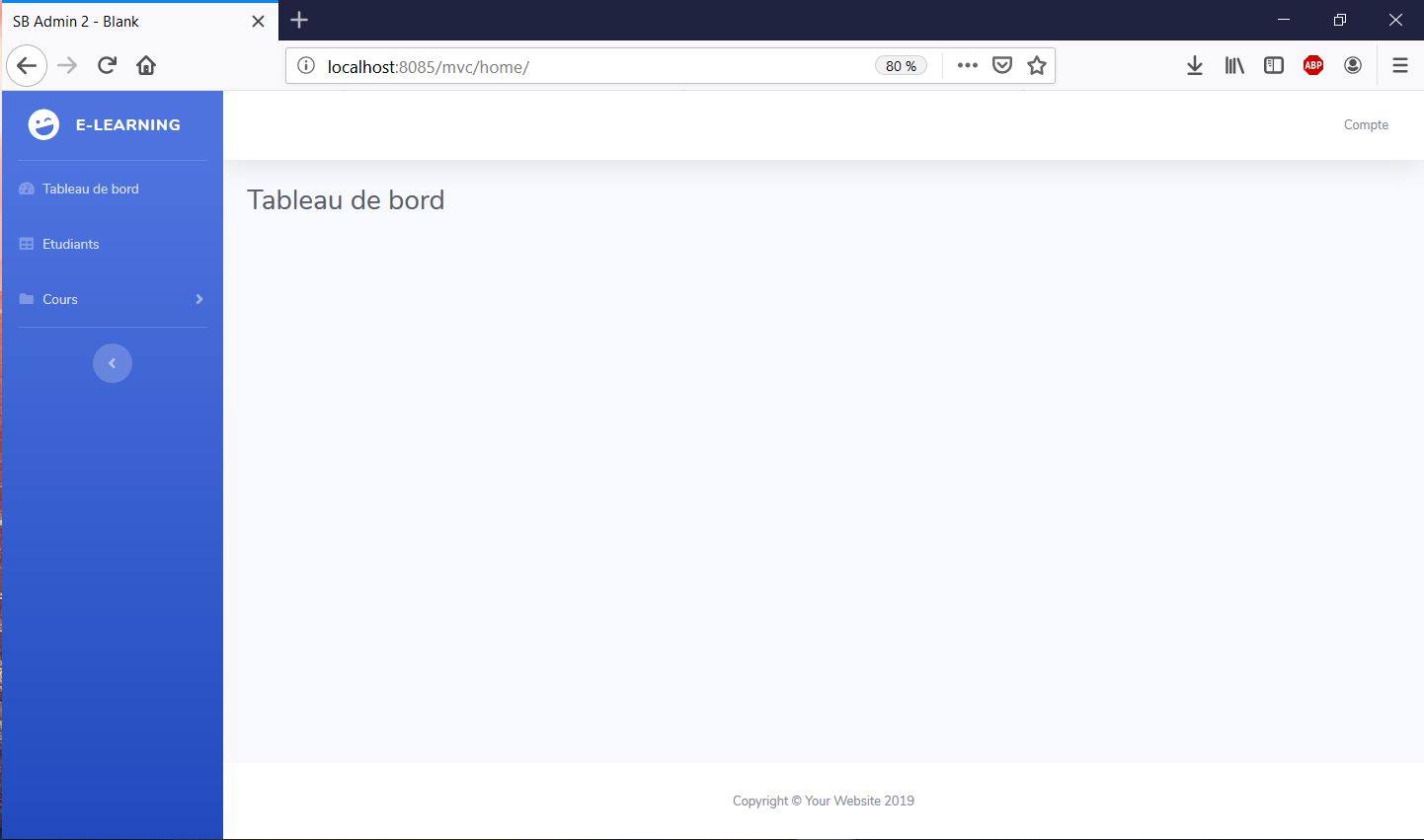
#### **Mise à jour des questions d’examen :**



**Figure 50: mise à jour des questions**

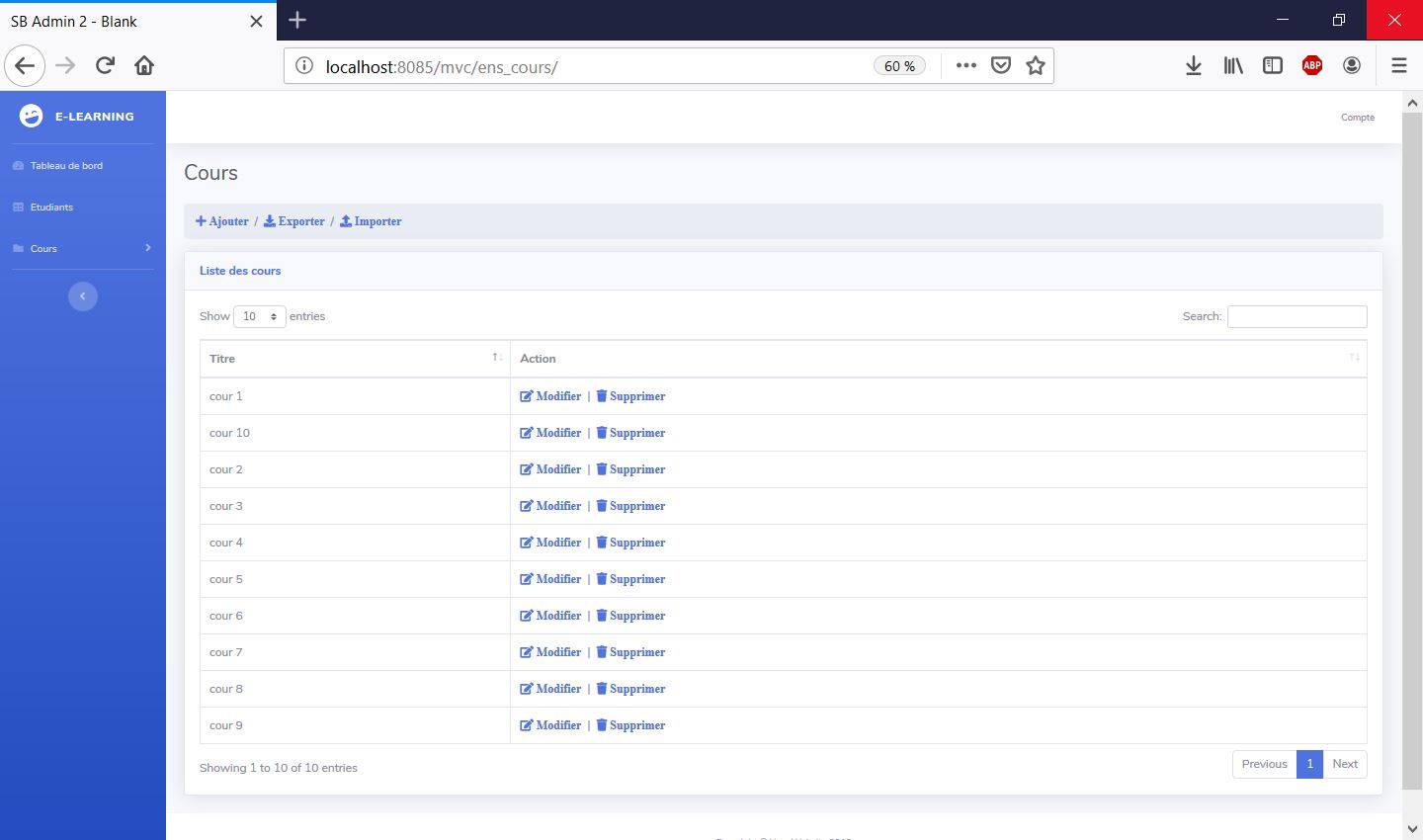
### **Espace Etudiant :**

#### **Interface principale d’un étudiant :**



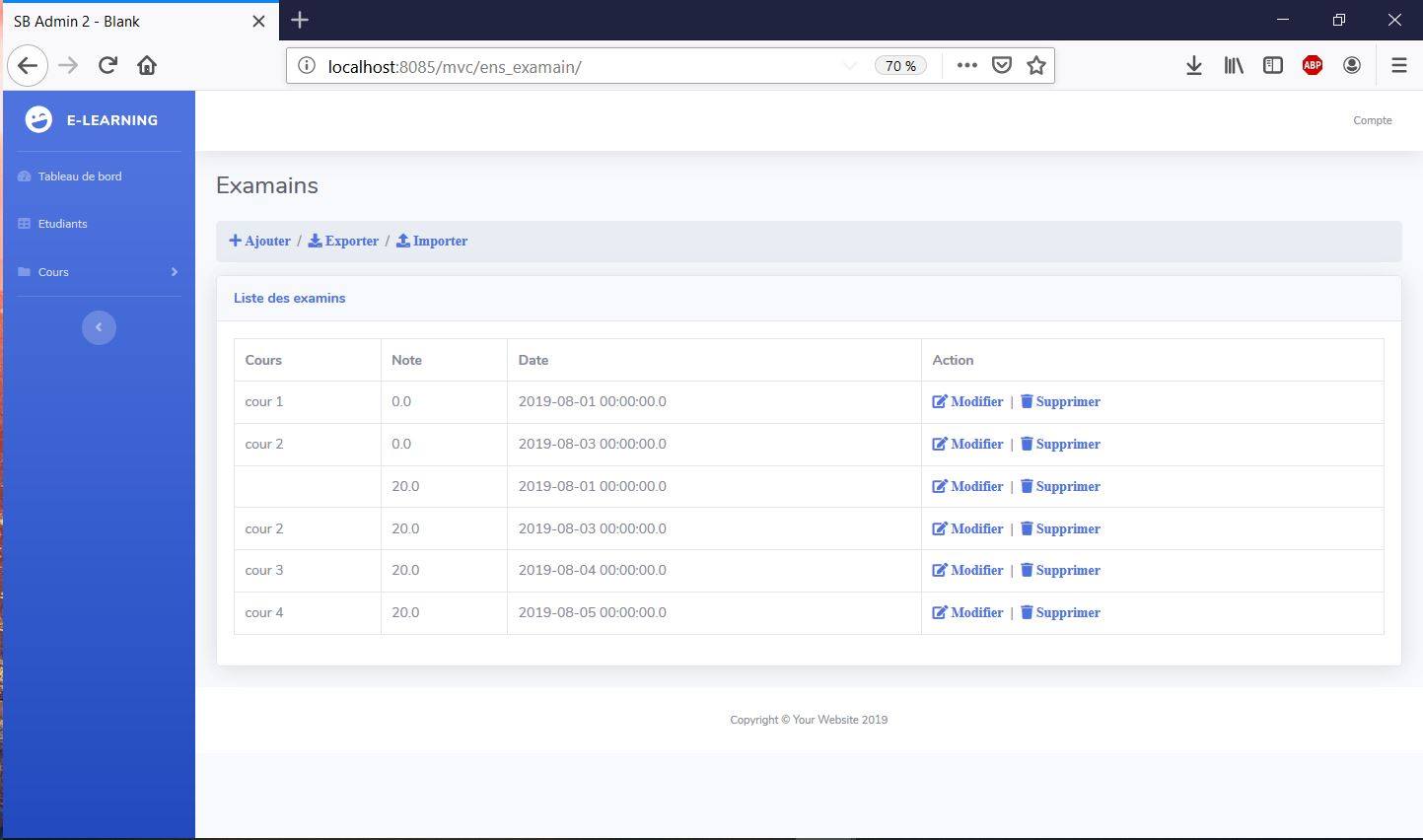
**Figure 51: interface de page d'accueil**

#### **Affichage la liste des cours :**



**Figure 52: consultation l'affichage des cours**

#### **Consultation des notes :**



**Figure 53: Consultation l'affichage des notes.**

# **Conclusion et perspectives :**

Le développement de l’internet et les outils de communication ont permis de développer l’enseignement par la naissance du nouveau mot : E-Learning ou bien la formation à distance.

L’objectif de E-Learning est de remplacer les anciennes façons temps, place, contenu de l’apprentissage prédéterminé avec des processus d’apprentissage rapides, ouverts, personnalisés.

Notre travail consiste à la « conception et réalisation d’une plateforme web pour la formation à distance ». Donc, nous avons réalisé une plateforme E-Learning grâce à la plateformeJ2EE et la technologie MYSQL qui nous permet de gérer notre base de données.

Notre plateforme permet aux enseignants de s’authentifie, créer et publier des cours et des examens et suivre leurs étudiants. L’étudiant aussi peut s’authentifie sur cette dernière, et consulte les cours affichés et ses notes des examens.

Le travail que nous avons réalisé peut être amélioré et enrichi afin d’en faire un système plus performant. Parmi les perspectives à prendre en compte pour améliorer le fonctionnement du système, nous citons :

1. Inscription des étudiants et des enseignants.
2. Créer un espace de travail collaboratif pour faire les TD et les TP.
3. Amélioration des taches manquantes et non marchante comme : le téléchargement des fichiers,...etc.
4. Faire un agenda pour la plateforme.
5. Amélioration du Template du site.

# **Bibliographie**

|  |  |
| --- | --- |
| [1] | A. E. Mouelhi, «JPA». |
| [2] | spring-mvc-controller-example, «javacodegeeks,» [En ligne]. Available: https://examples.javacodegeeks.com/enterprise-java/spring/mvc/spring-mvc-controller-example/?fbclid=IwAR0OBrB4qluz8zX-RmRgo-lTDmNwiS9kc-S1UX0TgPoaazqrGqnyakZiwpQ. |
| [3] | M. Youssfi, «dveloppement d' un site web jee». |
| [4] | «C:\Users\user\Desktop\Global M1\_S2\JEE\projet\conception d'un site web elearning». |
| [5] | W. W. &. J. White, Spring in practice. |