

**Université Assane Seck de Ziguinchor**

**-------------**

**UFR des Sciences et Technologies**

---------

**Département d’Informatique**

---------

**Génie Logiciel**

**-----------**

Mémoire de Master 2 Génie Logiciel

Développement d’une Application de gestion de la Scolarité et d’un site Web pour Ziguinchor Institut Polytechnique (ZIP)

**Par : Mr ISMAÏLA DIALLO**

Encadreur : **Dr KHALIFA GAYE**  Maître de Stage : **Pr. Baba Issa CAMARA** : ***Maître de conférences université de Lorraine, France***

**Membres du Jury :**

**Dr Alassane Diédhiou :** *(****Président du jury****)*

*Département Mathématique*

**Dr Marie Ndiaye :** *Département Informatique*

**Dr Ibrahima Diop :** *Département Informatique*

**Dr KHALIFA GAYE :** *Département Informatique*

*2015-2016*

**DEDICACES**

***Je dédie ce mémoire aux personnes qui me sont très chères et qui m’ont toujours soutenu et qui continuent à le faire.***

***Je veux citer :***

***Mes chers parents :*** *Salimatou BA et Mamadou DIOULDE*

***Mes frères et sœurs:*** *Issagha, Saliou, Korka, Abdoulaye et Awa Diallo*

***Mes camarades de Classe :*** *Avéwé Bassène, Mass Samba Gueye, Mr Mboup, Rahime Diallo****,*** *Babacar Ka, Mohamed Badji, Diouldé*

***Mes deux nièces chéries et mes deux neveux :*** *Fana, Binta, Benté et Azedine Ka*

**Hommage**

**Pour toi ma très chère Maman *Salimatou Ba* :**

*Maman, c’est toi qui m’as donné la vie*

*Ta voix, j’entends encore ta douce mélodie*

*Du haut du ciel, entends tes enfants chéris*

*Maman, pourquoi tu n’es pas là ?*

*J’ai besoin de toi aujourd’hui.*

*Ce que je veux te dire, de mon cœur je le crie*

*Je t’aime maman, j’’aurais dû mieux te le dire.*

*Car tu m’as donné la meilleure éducation*

*Dirigée vers le chemin de la conviction*

*Aussi tracé la route de ma profession*

*Dans mes tracas tu fus grande consolation*

*En m’aidant, me soutenant sans modération.*

*Si tu étais là, je te dirais ma tristesse*

*Je te parlerais, tu calmerais ma détresse*

*J’irai vers toi et me blottirais dans tes bras*

*La vie est si dure et la solitude est là*

*Ton amour me redonnerait goût et soleil*

*A toutes ces joies jadis de bonheur sans pareil*

*Les enfants débordés, pas besoin de mes larmes*

*Je dois me ressaisir car plus rien n’a de charme*

*Protège-moi Maman du haut du ciel, je t’en prie*

*Redonne-moi la force, le courage, la joie de vivre et d’aller de l’avant.*

*Je t’aime ma Maman Chérie, Reposes en Paix !!!*

**REMERCIEMENTS**

A Allah le Miséricordieux le Tout Miséricordieux, qui par Sa Grâce et Sa Miséricorde nous a donné la santé d’achever ce travail.

Ce mémoire a été possible grâce à l’appui d’un certain nombre de personnes, qui par leur soutien, disponibilité, conseils m’ont permis de mener à terme ce travail. Ainsi, je formule mes remerciements :

* A mon maître de stage **Pr Baba Issa CAMARA**, Maître de conférences à l’université de Loraine France;
* A mon encadreur le **Docteur Khalifa GAYE**
* A **Amadou Tidiane Ly**
* **Au Dr Marie NDIAYE** du **Département Informatique**
* **Au Dr Ibrahima DIOP du département Informatique**
* Au **Dr Youssou DIENG** chef du **Département Informatique**
* A **Mr Gorgoumague SAMB**
* Au **Dr Youssou FAYE** responsable du **Master Génie Logiciel**

A tous les enseignants chercheurs du **Département** **d’Informatique**, **Mathématiques**, **Physique** et **Chimie**, à toutes les personnes qui m’ont soutenu de près comme de loin sans oublier de nommer le corps professoral de **ZIP** Alassane Tamboura, Dr Léna Tendeng, Mr Aïdara, Mr Sambou, Mr Sarr …

**Liste des figures**

[Figure 1 formation gea 13](#_Toc478597051)

[Figure 2 cycle de vie en V 16](#_Toc478597052)

[Figure 3 : Cas d’utilisation étudiant 19](#_Toc478597053)

[Figure 4 : diagramme de cas d’utilisation de l’acteur professeur 20](#_Toc478597054)

[Figure 5 : Diagramme de cas d’utilisation du gestionnaire 21](#_Toc478597055)

[Figure 6 diagramme de séquence du scénario ajouter d’un utilisateur 22](#_Toc478597056)

[Figure 7 diagramme de séquence du scénario consulté le relevé de notes 23](#_Toc478597057)

[Figure 8 Diagramme de classe 25](#_Toc478597058)

[Figure 9 Architecture Client-serveur 27](#_Toc478597059)

[Figure 10 Diagramme de cas d’utilisation du site Web 31](#_Toc478597060)

[Figure 11 Structure de la couche métier 35](#_Toc478597061)

[Figure 12 Entité Classe 36](#_Toc478597062)

[Figure 13 Entité Comptes 37](#_Toc478597063)

[Figure 14 Interfaces Local : IScolariteLocal 37](#_Toc478597064)

[Figure 15 Interface Remote : IScolariteRemote 38](#_Toc478597065)

[Figure 16Implémentation des interfaces IScolariteLocal et IScolariteRemote 38](#_Toc478597066)

[Figure 17 ajout de la librairie JUnit 41](#_Toc478597067)

[Figure 18 librairie JUnit 2 41](#_Toc478597068)

[Figure 19 Librairi JUnit 3 42](#_Toc478597069)

[Figure 20 Librairie JUnit 4 42](#_Toc478597070)

[Figure 21 classe de test de la classe Comptes 43](#_Toc478597071)

[22 43](#_Toc478597072)

[Figure 23 classe de test 44](#_Toc478597073)

[Figure 24 Définition des méthodes de test 45](#_Toc478597074)

[Figure 25 les méthodes de tests pour la classe Comptes 46](#_Toc478597075)

[Figure 26 Exécution du test 46](#_Toc478597076)

[Figure 27 Résultat de l’exécution du test 47](#_Toc478597077)

[Figure 28 Menu principal du gestionnaire 48](#_Toc478597078)

[Figure 29 menu Administrateur 48](#_Toc478597079)

[Figure 30 interface de gestion des professeurs 49](#_Toc478597080)

[Figure 31 Gestions des utilisateurs 49](#_Toc478597081)

[Figure 32 Gestion de l’école 50](#_Toc478597082)

[Figure 33 Gestion Scolarité 50](#_Toc478597083)

[Figure 34 Menu « front-office » du site web 51](#_Toc478597084)

[Figure 35 Menu « back-office » du site web 51](#_Toc478597085)

[Figure 36 Interface Accueil 51](#_Toc478597086)

[Figure 37 Formation 52](#_Toc478597087)

[Figure 38 Inscription d’un étudiant 52](#_Toc478597088)

**Cigles et Abréviations**

**UP** : Unified Process

**Stick-man** : Dessin représentant un acteur.

**Tiers** : partie, couche.

**SGBDR** : Système de gestion des bases de données relationnelles.

**Client** : dispositif pouvant utiliser notre application.

**Serveur de SGBD** : machine hébergeant le SGBD

**Serveur web** : machine hébergeant la partie web de notre application.

**Serveur d’application** : machine hébergeant la partie métier de notre application

**Classe** : Objet virtuel simulant un objet de la vie réelle.

**Méthode** : action, comportement d’une classe.

**Java** : Langage de programmation objet.

**Persistance** : mécanisme responsable de la sauvegarde des données.

**Transaction** : Une transaction est une succession d’opérations, le plus souvent vers une base de données, dont leurs exécutions doivent toutes réussir ou aucune ne doit être exécutée. Elle permet d'assurer l'intégrité des données.

**Déploiement** : processus selon lequel une application ou un composant fini est distribué en vue de son installation sur d'autres ordinateurs.

**Injection de dépendances** : Il consiste à créer dynamiquement (injecter) les dépendances entre les différentes classes en s'appuyant sur une description (fichier de configuration ou métadonnées) ou de manière programmatique

**Open Source** : libre

**Framework** : ensemble d’outils disponibles pour développer mieux et plus vite. .

**Managed Bean** : Classe Java qui fait la liaison entre les pages et le code métier.

**Langage EL** : Petits bouts de code utilisés sur les pages jsf pour lier un composant à un attribut du Managed Bean.

Conteneur :

**JVM** : Java Virtual Machine

**API** : Application Programming Interface

**JPA** : Java Persistence API

**Data source** : Source de données.

**ZIP** : Ziguinchor Institut Polytechnique

**RT** : Réseaux et Télécommunications

**ISD** : Informatique et Statistique Décisionnelle

**GEA** : Gestion des Entreprises et des Administrations

**GLT** : Gestion Logistique et Transport

**EMRT : E-Commerce – Marketing et Réseaux de Communications**

**DUT** : Diplôme Universitaire de Technologie

**LP** : Licence Professionnelle

**UML** : Unified Modeling Language

**Résumé**

Du 15 Février 2015 au 15 Décembre 2015, pour une durée de 9 mois, j’ai effectué un stage au sein de l’entreprise **ZIP** qui est un institut d’enseignement supérieur polytechnique (située au quartier de ***Diabir/Ziguinchor*** en face de l’université ***Assane Seck de Ziguinchor***). Au cours de ce stage au département Informatique de **ZIP**, j’ai pu m’intéresser essentiellement sur le développement et la mise en place de deux plates-formes

* Un site Web <http://zip-edu.sn> : pour augmenter la visibilité de l’institut et permettre aux étudiants de connaître les différentes formations dispensées dans l’école.
* Une application de gestion de scolarité **ZipScolarWeb** moderne sécurisé dont les informations sont rapidement accessibles, qui facilitera la bonne gestion de l’institut en offrant une large gamme de services aux personnels administratifs, aux enseignants intervenant dans l’institut mais aussi aux étudiants.

**Abstract**

From February 15th 2015 to December 15th 2015, for a period of 9 months, I completed an internship at the ZIP company, which is a polytechnic higher education institute (located in the Diabir / Ziguinchor district opposite the Assane Seck University of Ziguinchor). During this internship at the IT department of ZIP, I was able to focus mainly on the development and implementation of two platforms

* A website <http://zip-edu.sn> : to increase the visibility of the institute and to allow the students to know the different training provided in the school.

# A modern secure ZipScolarWeb school management application with fast access to information, which will facilitate the proper management of the institute by offering a wide range of services to administrative staff, teachers working in the institute but also to students.

# Table des matières

[Table des matières 1](#_Toc478596945)

[Introduction Générale 11](#_Toc478596946)

[Chapitre 1 : Présentation de Ziguinchor Institut Polytechnique 12](#_Toc478596947)

[1 Introduction 12](#_Toc478596948)

[2 Formations offertes par ZIP 12](#_Toc478596949)

[2.1 DUT en Réseaux et Télécommunications (DUT-RT) 12](#_Toc478596950)

[2.2 DUT en Informatique et Statistique Décisionnelle (DUT-SID) 12](#_Toc478596951)

[2.3 Dut en Gestion des Entreprises et des Administrations (DUT-GEA) 13](#_Toc478596952)

[2.4 DUT en Gestion Logistique et Transport (DUT-GLT) 13](#_Toc478596953)

[2.5 Licence Professionnelle en RT (LP-RT) 13](#_Toc478596954)

[2.6 Licence Professionnelle en ISD 13](#_Toc478596955)

[2.7 Licence Professionnelle en E-commerce – Marketing et Réseaux de Communication (LP-EMRC) 13](#_Toc478596956)

[Chapitre 2 : Spécifications Fonctionnelles Détaillées 15](#_Toc478596957)

[1 Introduction 15](#_Toc478596958)

[2 Conception de la plate-forme de Gestion de Scolarité 15](#_Toc478596959)

[2.1 Processus de développement 15](#_Toc478596960)

[2.2 Définition et Analyse des besoins 16](#_Toc478596961)

[2.2.1 Définition des besoins 16](#_Toc478596962)

[2.2.2 Définition des cas d’utilisations 18](#_Toc478596963)

[2.3 Architecture de l’application 26](#_Toc478596964)

[3 Conception du site Web 28](#_Toc478596965)

[3.1 Introduction 28](#_Toc478596966)

[3.1.1 But de cette seconde partie 28](#_Toc478596967)

[3.1.2 Contexte de l’application 28](#_Toc478596968)

[3.2 Spécifications Générales 28](#_Toc478596969)

[3.2.1 Présentation du logiciel 28](#_Toc478596970)

[3.2.2 Environnement du produit 28](#_Toc478596971)

[3.3 Spécifications fonctionnelles Générales 29](#_Toc478596972)

[3.3.1 Accès à l’application 29](#_Toc478596973)

[3.3.2 Gestionnaire du droit d’accès à l’application et définitions des besoins 29](#_Toc478596974)

[Chapitre 3 : Développement (codage) 32](#_Toc478596975)

[1 Introduction 32](#_Toc478596976)

[2 Outils de développements 32](#_Toc478596977)

[2.1 Eclipse 32](#_Toc478596978)

[2.2 Joomla 32](#_Toc478596979)

[2.3 SGBDR Wamp Server 32](#_Toc478596980)

[2.4 La plateforme Java EE 33](#_Toc478596981)

[2.4.1 La couche Web : JavaServer Faces (JSF) 33](#_Toc478596982)

[2.4.2 Couche métier : Entreprise JavaBeans (EJB) 33](#_Toc478596983)

[2.4.3 Serveur : JBoss as 7.1.1. Final 34](#_Toc478596984)

[3 Codage de la couche métier EJB 34](#_Toc478596985)

[3.1 Structure de la couche métier 35](#_Toc478596986)

[3.2 Codage des Entités 35](#_Toc478596987)

[3.3 Codage des interfaces 37](#_Toc478596988)

[3.4 Implémentation des interfaces 38](#_Toc478596989)

[Chapitre 4 : Intégration Continue 39](#_Toc478596990)

[1 Introduction 39](#_Toc478596991)

[2 Les différentes étapes du test des logiciels 39](#_Toc478596992)

[3 Test unitaire de la plateforme 40](#_Toc478596993)

[3.1 Les outils de test 40](#_Toc478596994)

[3.2 Installation de JUnit 40](#_Toc478596995)

[3.3 Codage des Classes de tests 42](#_Toc478596996)

[3.3.1 Exemple de la classe TestCase 43](#_Toc478596997)

[3.3.2 Test des classes métier 43](#_Toc478596998)

[Chapitre 5 : Interface Homme Machine (IHM) 48](#_Toc478596999)

[1 Présentation de l’application Gestion de Scolarité 48](#_Toc478597000)

[1.1 Menus de l’application 48](#_Toc478597001)

[1.2 Les interfaces de l’application 48](#_Toc478597002)

[2 Présentation du site Web http://zip-edu.sn 51](#_Toc478597003)

[2.1 Les menus 51](#_Toc478597004)

[2.2 Les interfaces 51](#_Toc478597005)

[Conclusion 53](#_Toc478597006)

# Introduction Générale

Mon stage s’est déroulé à Ziguinchor Institut Polytechnique (ZIP) pour une durée de neuf mois.

Fondé en 2015, ZIP qui est une école d’enseignement supérieur polytechnique, forme des ingénieur(e)s généralistes des Sciences et Technologies Numériques, adaptables à leur environnement technique et humain, ainsi qu’aux exigences des secteurs d’activité très diversifiés au Sénégal et à l’international. Zip offre une gamme de formation très riche et très variée avec quatre Diplômes Université de Technologie (DUT) à savoir :

* Dut en Réseaux et Télécommunications (DUT-RT)
* Dut en Informatique et Statistique Décisionnelle (DUT-ISD)
* Dut en Gestion Logistique et Transport (DUT-GLT)
* Dut en Gestion des Entreprises et des Administrations (DUT-GEA)

Trois Licences Professionnelles (LP) :

* Licence Professionnelle en Réseaux et Télécommunications (LP-
* Licence Professionnelle en Informatique et Statistique Décisionnelle
* LP en E-Commerce – Marketing et Réseaux de Communications (LP-E-MRC)

Et des Masters Professionnels (MP) qui seront ouverts sous-peu

L’objectif du présent document est de décrire l’ensemble des fonctionnalités à mettre en place en vue de la réalisation et de la mise en œuvre de l’application Web baptisée ZIP … sur les épaules d’un géant et de la plate-forme de Gestion de Scolarité nommé **ScolarZipWeb**.

Pour atteindre cet objectif on a divisé notre travail en 5 Chapitres

* Chapitre 1 : Présentation de Ziguinchor Institut Polytechnique
* Chapitre 2 : Spécifications Fonctionnelles Détaillées (SFD)
* Chapitre 3 : Développement
* Chapitre 4 : Intégration continue
* Chapitre 5 : Démonstration Interface Homme Machine (IHM)

Chapitre 1 : Présentation de Ziguinchor Institut Polytechnique

# Introduction

Fondé en 2015, Ziguinchor Institut Polytechnique (ZIP) forme des ingénieur(e)s généralistes des Sciences et Technologies Numérique, adaptables à leur environnement technique et humain, ainsi qu'aux exigences des secteurs d'activité très diversifiés au Sénégal et à l'international. Le promoteur est natif de la raison de Ziguinchor, il s’agit **Mr Baba Issa Camara** *Maître de conférence à université de Lorraine région de Metz France* dans le domaine des mathématiques.

**ZIP** est une institution multinationale qui a pour vocation de former des diplômés (Techniciens supérieurs, Licences Professionnelles, Ingénieurs, Masters, Mastères spécialisés) dans les domaines de nouvelles technologies de l'information et de la communication.

# Formations offertes par ZIP

Zip offre une gamme de formation très riche et très variée avec quatre Diplômes Université de Technologie (DUT) et trois Licences Professionnelles  (LP).

## DUT en Réseaux et Télécommunications (DUT-RT)

L’objectif du DUT en Réseaux et Télécommunication est de former les étudiants dans les différentes disciplines des RT. Les futurs diplômés seront compétents pour :

* Installer et faire évoluer des architectures de réseaux informatiques et téléphoniques et de télécoms (câbles, antennes, connecteurs, décodeurs, etc.)
* Mettre en œuvre un réseau à intégration de systèmes routage (NAT, VPN, xDSL) et de gestion de réseaux voix-données et images

## DUT en Informatique et Statistique Décisionnelle (DUT-SID)

L’objectif du DUT en Informatique et Statistique Décisionnelle est de former des techniciens en traitements des données.

## Dut en Gestion des Entreprises et des Administrations (DUT-GEA)



Figure 1 formation gea

## DUT en Gestion Logistique et Transport (DUT-GLT)

Le DUT Gestion Logistique et Transport propose de vous préparer aux fonctions de cadres intermédiaire capable d'apporter aux entreprises des compétences générales solides, ainsi que des savoir-faire dans le domaine de la Logistique et du Transport

## Licence Professionnelle en RT (LP-RT)

Les futurs diplômés seront compétents pour:

* Analyser et comprendre le réseau d'une entreprise en vue de son exploitation, son amélioration et sa sécurisation
* Concevoir et déployer un réseau informatique sécurisé

## Licence Professionnelle en ISD

La licence Professionnelle est tournée vers les nouvelles technologies du Système d'Information. Elle aborde tous les aspects permettant de répondre aux nombreuses questions que se posent les entreprises soucieuses d'optimiser leur Système d'Information (SI).

## Licence Professionnelle en E-commerce – Marketing et Réseaux de Communication (LP-EMRC)

L'objectif est de former des collaborateurs ayant une double compétence «  **commerce - technologie de la communication** » qui leur permette:

* D'utiliser le net comme outil de communication, search marketing, e-publicité, e-mailing, e-newsletter...
* De participer à la conception et au développement d'outils de gestion de la relation et de la fidélisation client

Chapitre 2 : Spécifications Fonctionnelles Détaillées

# Introduction

Actuellement, le monde connaît une avancée technologie considérable dans tous les secteurs et cela grâce à l’informatique qui est une science qui étudie les techniques de traitements automatiques de l’information. Elle joue un rôle important dans le développement de l’entreprise ainsi que d’autres établissements.

De ce fait l’objectif de ce projet dans cette partie est de concevoir une plateforme de gestion de scolarité sécurisée dont les informations sont rapidement accessibles, qui facilitera la bonne gestion de l’institut en offrant une large gamme de services aux personnels administratifs, aux enseignants intervenants dans l’institut mais aussi aux étudiants et d’un site Web moderne sécurisé dont les informations sont rapidement accessibles, qui augmentera la visibilité de l’institut et qui donnera la possibilité aux étudiants de découvrir l’ensemble des formations dispensées et aussi d’y s'inscrire.

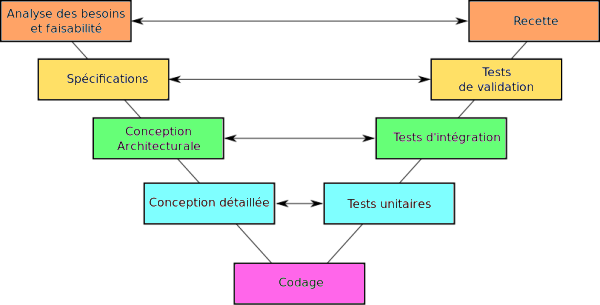
# Conception de la plate-forme de Gestion de Scolarité

## Processus de développement

Un projet se démarque par son cycle de vie, qui est généralement présenté comme étant constitué de phases (processus). Un processus est un savoir-faire associé à une discipline (par exemple : le développement de logiciel). C’est un ensemble structuré :

* d’acteurs (concepteur, chef de projet…)
* d’activités pour chaque acteur (code, planifier…)
* d’artefacts pour chaque activité (exécutable, planning…) un artefact est le résultat concret d’une activité
* de workflows (un workflow = un ensemble d’activités)

Dans notre cas on va essayer d’adopter un processus capable de répondre à la gestion des besoins et des exigences (***traçabilité***), au souci permanent de la qualité (***recettes fréquentes de versions intermédiaires, automatisation des tests, revus par les pairs***), à la gestion des risques permanente, à la gestion des demandes de changement qui est le principe **UP** en passant d’abord par la **capture des besoins**, puis sur **les spécifications fonctionnelles**, ensuite la **conception** et le **codage** et enfin aux **tests**.



Source : http://www.2dsi.fr/blog/blog\_agile.html

Figure 2 cycle de vie en V

## Définition et Analyse des besoins

Cette partie détaille la spécification et l’analyse de l’application dans le cadre de ce projet. Il se présente comme étant le cahier des charges de l’application à concevoir et à développer. Il présente aussi une partie analytique qui précède la phase de conception et correspond aux diagrammes UML.

### Définition des besoins

L'objectif de ce projet est de concevoir une plateforme web moderne sécurisé dont les informations sont rapidement accessibles, qui facilitera la bonne gestion de l’institut en offrant une large gamme de services aux personnels administratifs, aux enseignants intervenants dans l’institut mais aussi aux étudiants.

#### Présentation des acteurs

Un acteur représente une personne, un matériel ou un logiciel qui interagit directement avec le système en question. Les acteurs qui participent à ce système sont :

* les utilisateurs (gestionnaire, étudiant ou professeur) qui ont des droits d’accès limités.
* L’administrateur possédant tous les droits d’accès pour interagir avec le système.

#### Les besoins non fonctionnels

Les besoins non fonctionnels représentent les exigences implicites auquel le système doit répondre. Parmi ces besoins on cite :

* Le système doit être simple à utiliser.
* Le déplacement entre les pages doit être souple et facile.

Le système doit assurer une bonne interface qui donne aux administrateurs et aux internautes l’envie d’y visiter.

#### Les besoins fonctionnels

Les services qu’offrira cette plateforme seront :

* **Gestion des étudiants :** ce service permettra entre autre de gérer
  + Les inscriptions des étudiants
  + Leur paiement mensuel pour leur formation
  + Mais aussi la création automatique de leur carte étudiante
* **Gestion des notes :** 
  + les enseignants pourront gérer les notes de leurs étudiants directement sur la plateforme en renseignant les notes de contrôle continu ainsi que les notes d’examen pour chacun des étudiants. Les notes ne peuvent être modifiées que par l’enseignant.
  + Les étudiants aussi auront la possibilité de consulter leurs relevés de notes directement via la plateforme.
  + La plateforme permettra aussi de faire une délibération des notes d’examens
* **Gestion des emplois du temps**
  + Pour faciliter le bon planning des cours :
    - Les enseignants intervenants dans l’institut pourront renseigner directement sur la plateforme leur disponibilité
    - A partir de ces informations on peut créer facilement des emplois du temps.
  + Les étudiants ainsi que même les enseignants auront la possibilité de voir leurs emplois du temps à partir de la plateforme
* **Gestion de partage de fichiers**
  + Ici les enseignants pourraient poster leurs cours, TD, TP sur la plateforme enfin que les étudiants puissent les télécharger,
  + En retour les étudiants peuvent y mettre directement les projets qu’ils auront à réaliser enfin que ça soient accessibles par leurs professeurs.
* **La gestion des comptes utilisateurs**
  + Permettra de gérer les différents comptes utilisateurs de la plateforme
  + L’administrateur
  + Le secrétariat
  + Le directeur général
  + Les enseignants
  + Les étudiants
* **Gestion de suivi des étudiants dans leur stage en entreprise**
  + Permettra d’avoir un suivi des étudiants dans leur stage
  + De connaitre leur sujet de stage, dans qu’elle entreprise le stage se fait ainsi que la date du début et celle de la fin du stage.
* **Gestion des alertes**
  + Permettra d’envoyer des messages à l’ensemble des utilisateurs soit par
    - **Mailing**
    - **SMS** en ayant notre propre serveur de sms
* **Statistiques**

Là nous auront toutes les statistiques de l’institut. Notamment sur le taux de réussite des étudiants, le nombre d’étudiants que constitue l’école

### Définition des cas d’utilisations

Pour développer une application il faut d’abord organiser ses idées, les documenter, pour organiser la réalisation en définissant les modules et les étapes de la réalisation. On appelle cette démarche modélisation. Pour bien définir les fonctionnalités de cette application, on a choisi le langage UML qui parait la meilleure solution pour modéliser les différents objets du système.

#### Le langage de modélisation UML

**UML** (***Unified modeling Language***) [1], que l’on peut traduire par langage de modélisation unifiée est une notation permettant de modéliser un problème de façon standard. Ce langage est né de fusion de plusieurs méthodes existantes auparavant et est devenu désormais la référence en termes de modélisation objet. Le langage **UML** fourni une panoplie d’outils permettant de représenter l’ensemble des éléments du mode objet (classes, objets,...) ainsi que les liens qui les relient. Toutefois, étant donnée qu’une seule représentation est trop subjective, **UML** fourni un moyen astucieux permettant de représenter diverses projections d’une même représentation grâce aux vues.

Pour la conception et la réalisation de notre application, nous avons choisis de modéliser avec le formalisme **UML** (***Unified Modeling Language***) qui offre une flexibilité marquante qui s’exprime par l’utilisation des diagrammes. UML définit neuf types de diagrammes devisés en deux catégories :

* **Diagrammes statiques (structurels) :** 
  + diagramme de classe
  + d’objet
  + de composant
  + de déploiement
  + de diagramme de cas d’utilisation.
* **Diagrammes dynamique (comportementaux) :** 
  + diagramme d’activité
  + de séquence
  + d’état-transition
  + de diagramme de collaboration.

Pour la modélisation des besoins, nous utilisons les diagrammes UML suivant : Diagramme de cas d’utilisation, diagramme de séquence et diagramme de classe.

#### Diagramme de cas d’utilisation

Un diagramme de cas d’utilisation est un graphe d’acteurs, un ensemble de cas d’utilisation englobés par la limite du système, des associations de communication entre les acteurs et les cas d’utilisation, et des généralisations entre cas d’utilisation. [8] Il est destiné à représenter les besoins des utilisateurs par rapport au système. [7] Identification des acteurs Les acteurs d’un système sont les entités externes à ce système qui interagissent avec lui.

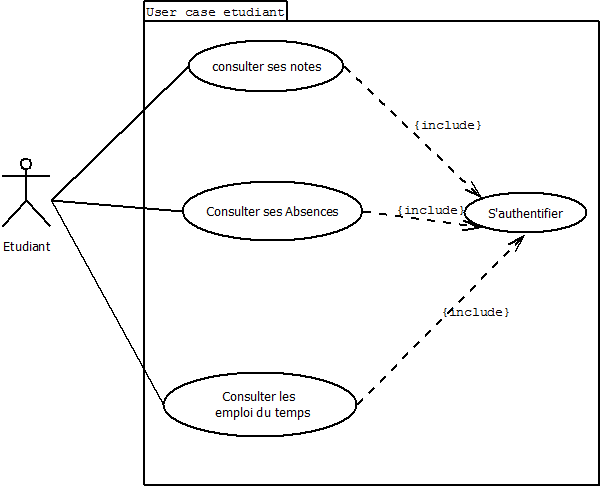
Dans notre application, les acteurs qui interagissent avec le système sont le gestionnaire(ou directeur général), les professeurs, les étudiants et l’administrateur du système.

##### Identification des cas d’utilisations

Un cas d’utilisation est utilisé pour définir le comportement d’un système ou la sémantique de toute autre entité sans révéler sa structure interne. Chaque cas d’utilisation spécifie une séquence d’action, y compris des variantes, que l’entité réalise, en interagissant avec les acteurs de l’entité. La responsabilité d’un cas d’utilisation est de spécifier un ensemble d’instances, où une instance de cas d’utilisation représente une séquence d’actions que le système réalise et qui fournit un résultat observable par l’acteur. [8]

Voici les cas d’utilisation de notre application:

* Etudiant
  + Consulter

Figure 3 : Cas d’utilisation étudiant

* Professeur
  + Gérer les notes
  + Consulter l’emploi du temps
  + Renseigner sa disponibilité
  + Ajouter des cours

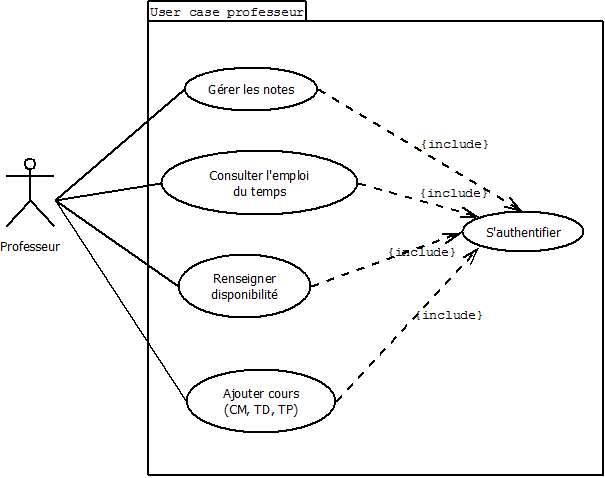


Figure 4 : diagramme de cas d’utilisation de l’acteur professeur

* Gestionnaire
  + Gérer les étudiants
  + Gérer les absences
  + Gérer les professeurs
  + Gérer les notes
  + Gérer les paiements
  + Gérer les modules

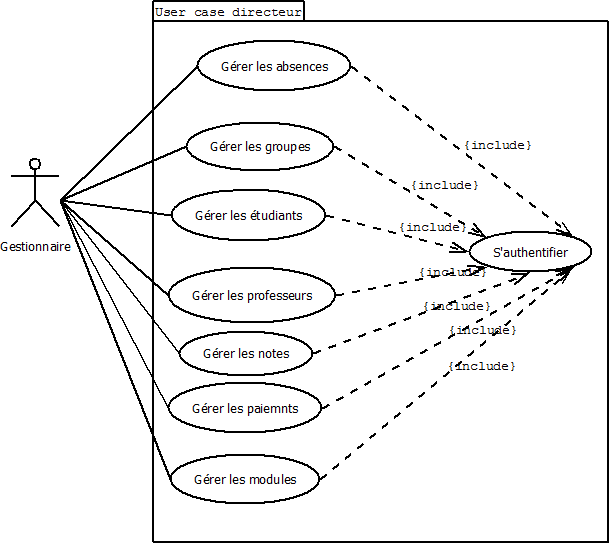


Figure 5 : Diagramme de cas d’utilisation du gestionnaire

#### Diagramme de séquence

Il permet de décrire les scénarios de chaque cas d’utilisation en mettant l’accent sur la chronologie des opérations en interaction avec les objets. [7] Un diagramme de séquence montre une interaction présentée en séquence dans le temps. En particulier, il montre aussi les objets qui participent é l’interaction par leur “ligne de vie ”et les messages qu’ils échangent présentés en séquence dans le temps. [8]

Voici quelques notions de base du diagramme : [4]

* Scénario : une liste d’actions qui décrivent une interaction entre un acteur et le système.
* Interaction : un comportement qui comprend un ensemble de messages échangés par un ensemble d’objets dans un certain contexte pour accomplir une certaine tâche.
* Message : Un message représente une communication unidirectionnelle entre objets qui transporte de l’information avec l’intention de déclencher une réaction chez le récepteur.

##### Diagramme de séquence du cas d’utilisation du scénario « **Ajouter un utilisateur** »

1. Le système demande à l’utilisateur de d’authentification.
2. L’utilisateur saisit le mot de passe.
3. Le système vérifie la validité du mot de passe dans la base de données.
4. Si c’est ok le système affiche la liste des opérations
5. L’utilisateur demande le formulaire d’ajout d’un utilisateur
6. Le système affiche le formulaire
7. L’utilisateur saisit les informations
8. Le système envoi les informations dans le SGBD (Système de gestion de base de données)
9. Le SGBD confirme l’enregistrement au système
10. Le système envoie un message de confirmation

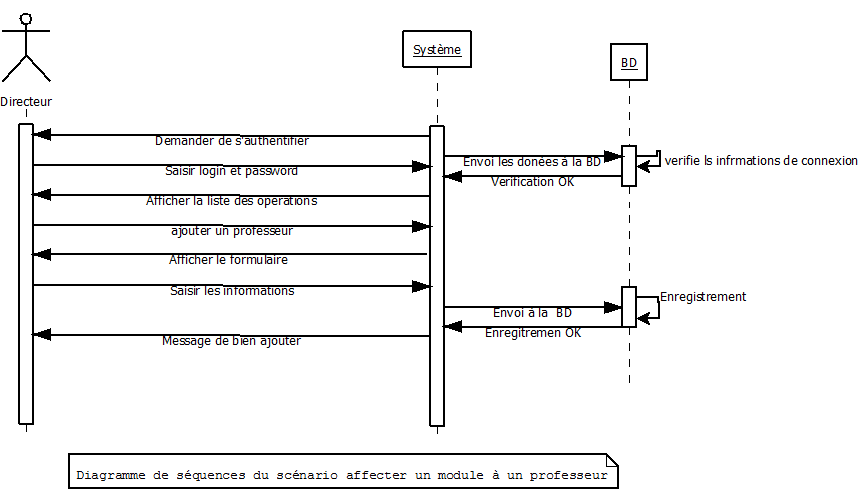


Figure 6 diagramme de séquence du scénario ajouter d’un utilisateur

##### Diagramme de séquence du cas d’utilisation du scénario « **Consulter son relevé de notes** »

1. Le système demande à l’utilisateur de d’authentification.
2. L’utilisateur saisit le mot de passe.
3. Le système vérifie la validité du mot de passe dans la base de données.
4. Si c’est ok le système affiche la liste des opérations
5. L’utilisateur demande le relevé de notes
6. Le système affiche la liste des notes

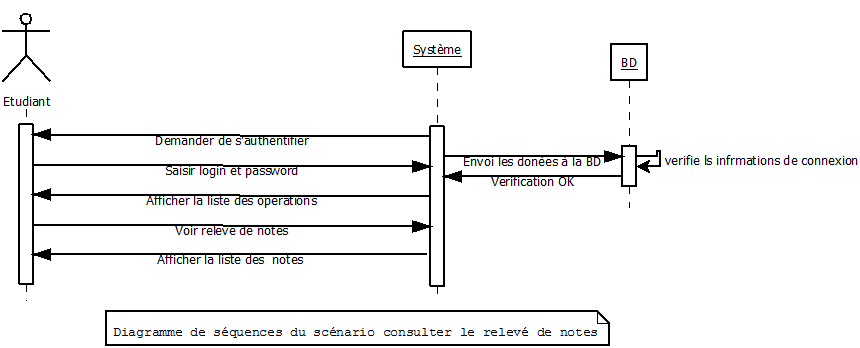


Figure 7 diagramme de séquence du scénario consulté le relevé de notes

##### Diagramme de séquence du cas d’utilisation du scénario « **Affecter un module à un professeur** »

1. Le système demande à l’utilisateur de d’authentification.
2. L’utilisateur saisit le mot de passe.
3. Le système vérifie la validité du mot de passe dans la base de données.
4. Si c’est ok le système affiche la liste des opérations
5. L’utilisateur demande le formulaire d’affectation de module
6. Le système affiche le formulaire
7. L’utilisateur saisit les informations
8. Le système envoi les informations dans le SGBD
9. Le SGBD confirme l’enregistrement au système
10. Le système envoie un message de confirmation

#### Diagramme de classe

C’est une collection d’éléments de modèle statique, tels que des classes, des interfaces et leurs relations, connectés entre eux comme un graphe. *[8]* Il représente la description statique du système en intégrant dans chaque classe la partie dédiée aux données et celle consacrée aux traitements. C’est le diagramme pivot de l’ensemble de la modélisation d’un système. *[7]*

**Identification des classes**

Une classe est une description d’un groupe d’objets partageant un ensemble commun de propriétés (les attributs), de comportements (les opérations) et de relations avec d’autres objets (les associations et les agrégations). *[6]*

Une classe contient : *[2]*

* Méthodes (ou opérations de la classe) : Les méthodes décrivent les opérations  
  qui sont applicables aux instances de la classe.
* Attributs (ou champs, ou variables d’instances) : Les attributs d’une classe décrivent la structure de ses instances (les objets).
* Agrégation : qui est une association correspondant à une relation qui lorsqu’elle est lue dans un sens signifie « est une partie de » et lorsqu’elle est lue dans l’autre sens elle signifie « est composé de ». Le diagramme de classe correspondant à notre application est la figure ci-dessous :

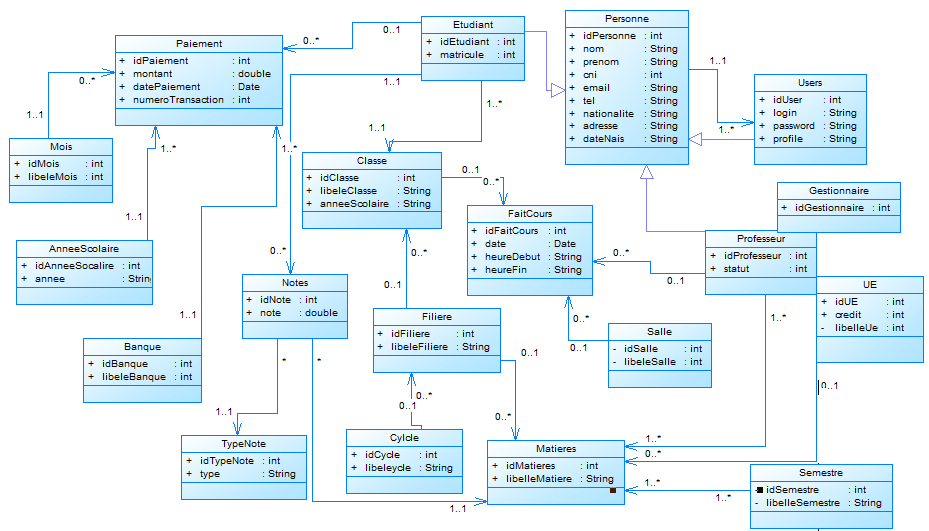


Figure 8 Diagramme de classe

## Architecture de l’application

L’architecture consiste à bien structurer un projet, pour se faire nous avons choisi d’utiliser le modèle MVC dont son objectif global est de séparer les aspects traitement, données et présentation, et de définir les interactions entre ces trois aspects. En simplifiant, les données sont gérées par le modèle, la présentation par la vue, les traitements par des actions et l’ensemble est coordonné par les contrôleurs.

* **Le modèle**

Le modèle est responsable de la préservation de l’état d’une application entre deux requêtes HTTP, ainsi que des fonctionnalités qui s’appliquent à cet état. Toute donnée persistante doit être gérée par la couche modèle. Cela concerne les données de session (le panier dans un site de commerce électronique par exemple) ou les informations contenues dans la base de données (le catalogue des produits en vente, pour rester dans le  
même exemple). Cela comprend également les règles, contraintes et traitements qui s’appliquent `à ces données, souvent désignées collectivement par l’expression logique de l’application.

* **La vue**

La vue est responsable de l’interface, ce qui recouvre essentiellement les fragments (X) HTML assemblés pour constituer les pages du site. Elle est également responsable de la mise en forme des données (pour formater une date par exemple) et doit d’ailleurs se limiter à cette tâche. Il faut prendre garde à éviter d’introduire des traitements complexes dans la vue,   
même si la distinction est parfois difficile. En principe la vue ne devrait pas accéder au modèle et obtenir ses données uniquement de l’action (mais il s’agit d’une variante possible du MVC). La vue est souvent implantée par un moteur de Template (que l’on peut traduire par gabarit), dont les caractéristiques, avantages et inconvénients donnent lieu à de nombreux débats. Nous utiliserons un de ces moteurs dans notre MVC, ce qui vous  
permettra de vous former votre propre opinion.

* **Contrôleurs et actions**

Le rôle des contrôleurs est de récupérer les données utilisateur, de les filtrer et les contrôler, de déclencher le traitement approprié (via le modèle), et finalement de déléguer la production du document de sortie à la vue. Comme nous l’avons indiqué précédemment, l’utilisation de contrôleurs a également pour effet de donner une structure hiérarchique à l’application, ce qui facilite la compréhension du code et l’accès rapide aux parties à modifier. Indirectement, la structuration logique d’une application MVC en contrôleurs et actions induit une organisation physique adaptée.



Source:http://theses.ulaval.ca/archimede/fichiers/24629/ch04.html shttp://theses.ulaval.ca/archimede/fichiers/24629/ch04.html http://theses.ulaval.ca/archimede/fichiers/24629/ch04.htmle : fsdffdsfsdffddqsddqsdfhttp://theses.ulaval.ca/archimede/fichiers/24629/ch04.htmlhttp://theses.ulaval.ca/archimede/fichiers/24629/ch04.htmlhttp://theses.ulaval.ca/archimede/fichiers/24629/ch04.html

Figure 9 Architecture Client-serveur

# Conception du site Web

## Introduction

### But de cette seconde partie

L’objectif de cette seconde partie est de décrire l’ensemble des fonctionnalités à mettre en place en vue de la réalisation et de la mise en œuvre de l’application Web baptisée Ziguinchor Institut Polytechnique (ZIP) sur les épaules d‘un géant.

### Contexte de l’application

L'objectif de ce stage est de concevoir un site web moderne sécurisé dont les informations sont rapidement accessibles, qui augmente la visibilité du site dans le monde entier et qui donnent la possibilité aux étudiants de découvrir l’ensemble des formations dispensées et aussi de s'inscrire ou de faire des payements en ligne.

## Spécifications Générales

### Présentation du logiciel

#### Service à offrir

Le site est accessible via le lien <http://zip-edu.sn> et sa prestation comprend :

* + Le développement de l’application (partie « front office » et « back office »
  + La maintenance corrective de l’application
  + La maintenance évolutive de l’application

### Environnement du produit

#### Environnement serveur

L’environnement de développement (Limoux) est le suivant :

* architecture WAMP (Windows, Apache, MySQL, PHP).
* développement en PHP, version 5.5.12
* OS du serveur HTTP : <http://www.ovh.com>
* serveur HTTP : Apache 2.4.9
* base de données : MySQL 5.6.17 (client API version).
* Joomla 3.3.6

De manière à disposer d’une application facilement maintenable et évolutive, la conception et le développement seront fait à l’aide du CMS Joomla version 3.3.6.

#### Environnement client

La compatibilité de l’application est assurée avec les navigateurs correspondant à l’Internet moderne et Mac (HTML5, CSS, DOM), à savoir :

* Mozilla 1.x sous Windows,
* Netscape 7,
* I.E. 6.0,
* Safari sous MacOs X.
* Google Chrome
* Navigateur mobile (Smartphone)

Il s’agit des navigateurs où nous testons que tous les éléments de décoration (images, typo, filets ou puces) apparaissent conformément à la création graphique.

## Spécifications fonctionnelles Générales

### Accès à l’application

L’accès au site Web Ziguinchor Institut Polytechnique ***ZIP*** se fait au travers de l’espace projet, par l’URL suivant : htpp://www.zip-edu.sn

Seul l’administrateur déjà définis dans l’espace « ***back office*** » du projet peut accéder à ***Ziguinchor Institut Polytechnique*** pour la maintenance ainsi que pour une mise à jour éventuelle de l’application Web. Les candidats ne peuvent accéder qu’aux données relatives à l’espace « **front office**» dans lequel ils peuvent avoir accès à toutes les informations définies dans cet espace et ainsi pouvoir faire des demandes de candidature.

### Gestionnaire du droit d’accès à l’application et définitions des besoins

La gestion des droits d’accès à l’outil est définie par l’administrateur, via le « ***back office*** » ou via le front office par un utilisateur simple, grâce à la notion de profils. Les différents types de profils et les fonctionnalités auxquels ils donnent accès sont les suivants :

* **Profil administrateur** : générique, ayant accès à toutes les fonctionnalités du back-office ainsi que :
  + lister les différentes formations
  + ajouter ou modifier une formation
  + supprimer une formation
  + afficher la liste de demande candidature
  + Faire une demande de candidature pour un candidat
  + Accepter ou refuser une candidature
  + Supprimer une candidature
  + Afficher la liste des candidats acceptés
  + Gérer les différents partenaires
  + Activer ou désactiver un commentaire posté dans le forum
  + Publier des articles
  + Ajouter un utilisateur
  + Supprimer ou modifier un utilisateur
* **Profile utilisateur** : Spécifique, ayant accès uniquement à la partie « front office » à savoir :
  + Afficher les informations relatives à l’école
  + Voir les différentes formations qui existent
  + Afficher les conditions d’admissibilité à Ziguinchor Institut polytechnique
  + Faire une demande de candidature
  + Faire une recherche sur les différentes informations du site

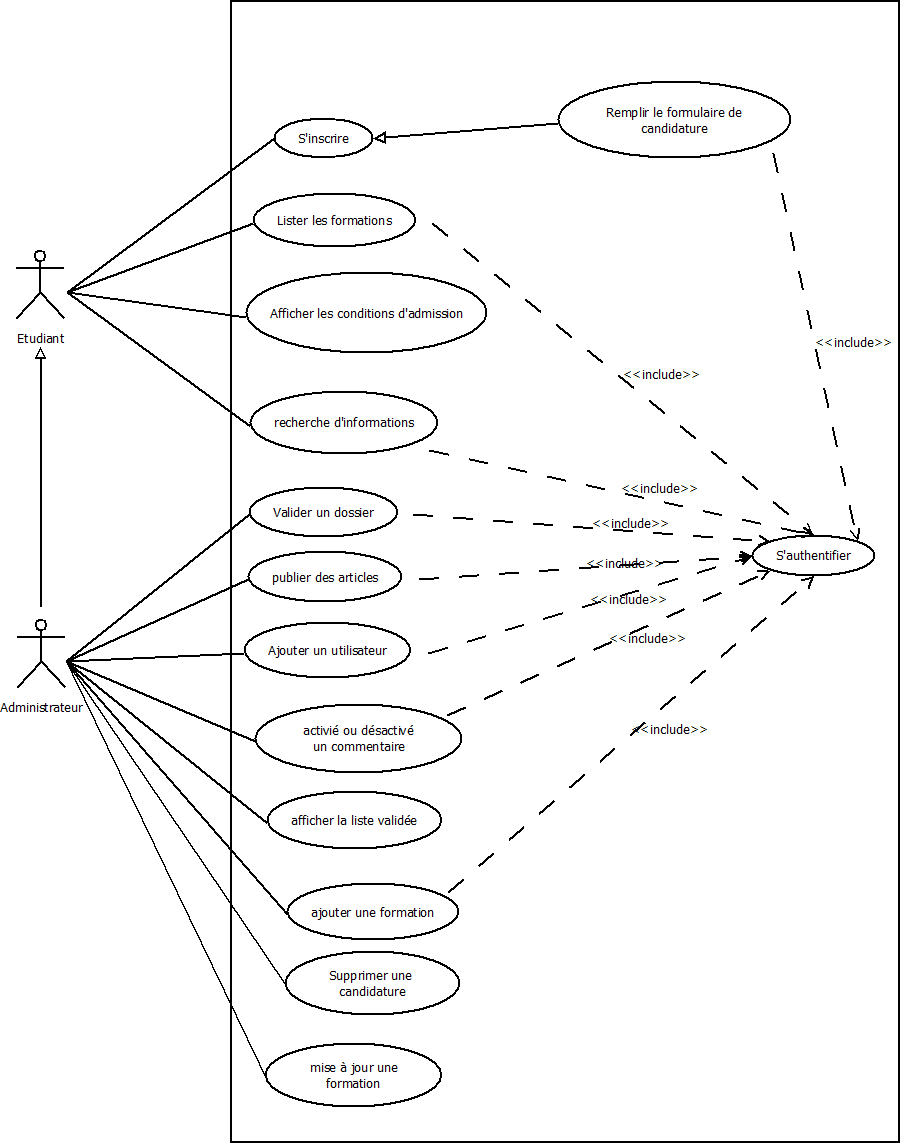


Figure 10 Diagramme de cas d’utilisation du site Web

Chapitre 3 : Développement (codage)

# Introduction

Dans ce chapitre, consacré à la réalisation et la mise en œuvre de notre application de gestion de Scolarité et de site web de ZIP, nous allons présenter les outils de développement adoptés; soit le SGBDR Wamp Server, le langage de manipulation de bases de données HQL ainsi que l’environnement utilisé qui est éclipse, la plateforme Java 2EE et enfin nous montrerons les principales interfaces et fenêtres de l’application.

# Outils de développements

Pour développer notre application nous avons utilisé plusieurs outils et nous allons faire une petite présentation de ces derniers.

## Eclipse

Eclipse est une plate-forme universelle pour des environnements de développement intégrés. Elle est fondée sur une architecture ouverte et extensible. C'est un logiciel gratuit portable, car écrit en Java, indépendant de tout langage et convivial.

## Joomla

Joomla est un [système de gestion de contenu](https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_gestion_de_contenu) (en [anglais](https://fr.wikipedia.org/wiki/Anglais) CMS pour Content Management System) [libre](https://fr.wikipedia.org/wiki/Logiciel_libre), [open source](https://fr.wikipedia.org/wiki/Open_source) et gratuit. Il est écrit en PHP et utilise une [base de données](https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es) MySQL. Joomla inclut des fonctionnalités telles que des flux RSS, des news, une version imprimable des pages, des [blogs](https://fr.wikipedia.org/wiki/Blogs), des sondages, des recherches. Joomla est sous licence GNU GPL.

## SGBDR Wamp Server

Wamp Server est un système de gestion de bases de données relationnel robuste et puissant, aux fonctionnalités riches et avancées, capable de manipuler en toute fiabilité de gros volumes de données, mêmes dans des situations critiques.

Un SGBDR est un ensemble de logiciels chargé les fonctions minimales suivantes :

* Le maintien de la cohérence des données entres elles ;
* Le contrôle d’intégrités des données entre elles ;
* Les opérations classiques sur les données (consultation, insertion, modification, suppression) :
* Les autorisations d’accès aux données.

## La plateforme Java EE

Les développements Java EE reposent sur un découpage en couches ou tiers, nous parlons alors d'applications multi-tiers. Trois grands tiers sont représentés : La couche présentation (tiers Web), la couche métier (tiers Métier ou tiers Business), la couche stockage des informations (tiers Enterprise Information System).

### La couche Web : JavaServer Faces (JSF)

La technologie JavaServer Faces est construite à partir de Servlets et fournit un Framework de développement pour accélérer la création d'applications Web.

Cette Framework est basée sur la notion de composants. Une page JSF est une page xhtml (ou jsp) liée aux Managed Bean via le language EL. A cette technologie nous avons ajouté d’autres outils comme :

* Primefaces : bibliothèque de composants graphiques qu’on peut utiliser avec JSF pour gagner en productivité et efficacité.
* Springs Security : Framework d’authentification et de contrôle d’accès. Il gère **L’authentification qui** consiste à garantir que la personne connectée est bien celle qu’elle prétend être et **les autorisations qui** consistent à vérifier que la personne connectée a bien les permissions d’effectuer une action donnée ou d’accéder à une ressource.

### Couche métier : Entreprise JavaBeans (EJB)

Un EJB est un composant  Java EE exécuté par un conteneur  spécifique dans une machine virtuelle java. Ce composant est une archive JAR (Java Archive), contenant un ou plusieurs objets java particuliers appelés **Beans**, possédant et  implémentant un certain nombre d'interfaces. Ces **Beans**  permettent au programmeur de travailler sur une base de données sans se soucier d'un quelconque SGBDR car un EJB Possède un langage de requêtes qui lui est propre appelé EJB-QL. Il possède un ou plusieurs fichiers de configuration XML (persistence.xml) et s’intègre dans une architecture trois tiers.

Un **Bean** est un objet Java particulier qui hérite d'une interface définissant un ensemble de méthodes gérant son cycle de vie au sein du conteneur. La spécification EJB propose différents types de **Beans** ayant chacun une tâche qui lui est propre.

Les Beans utilisés :

* Les Entity Beans qui sont la représentation objet des données présentes en base de données. Pour être simple un **Entity** correspond à  une table en base de données. La modification de cet objet engendre des modifications sur les données correspondantes en base de données.
* Les Sessions Beans qui accomplissent des tâches spécifiques pour un client distant ou local. Et en fonction des besoins l’état d’un Bean peut être conservé ou non d’où l’existence de deux types de Bean Session : **Statefull** (avec état) et **Stateless** (sans état).

### Serveur : JBoss as 7.1.1. Final

Les serveurs **Java EE** proposent plusieurs types de conteneurs (containers en anglais). Chaque conteneur a un rôle bien défini et offre un ensemble de services.

Une application **Java EE** de type Web nécessite un conteneur Web pour son exécution alors qu'une application utilisant les EJB nécessite un conteneur EJB.

Nous utilisons dans notre cas un serveur dans lequel les deux types d’application (WEB et EJB) peuvent s’exécuter en même temps : **Jboss as.**

**JBoss Application Server** ou **JBoss AS**, est un serveur d'applications **Java EE** Libre, écrit en **Java**, publié sous licence **GNU LGPL**. Étant écrit en Java, il peut être utilisé sur tout système d'exploitation fournissant une machine virtuelle Java (**JVM**).

# Codage de la couche métier EJB

Pour la partie métier nous allons créer un projet EJB. Comme tout projet EJB, nous aurons à créer les interfaces (**Locale** et **Remote**), **implémenter** ces deux **interfaces** et créer les **Entités** (***classes persistantes***).

## Structure de la couche métier

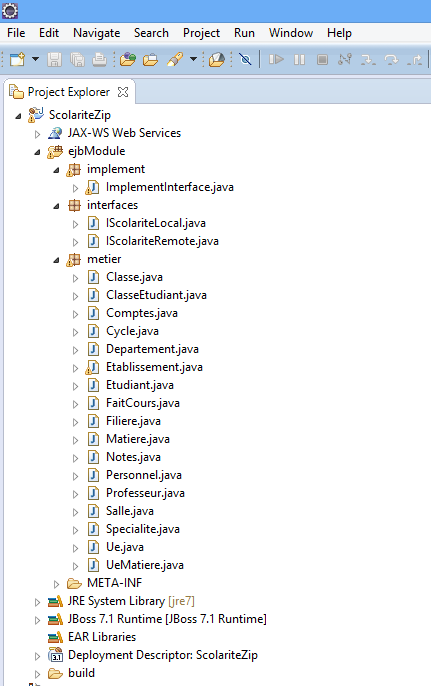


Figure 11 Structure de la couche métier

## Codage des Entités

Avec **JPA** (**Java Persistence API**) les programmeurs n’ont plus besoin d’être des spécialistes en base de données ou d’un SGBD quelconque. **JPA** leur permet de programmer les classes (entités) qui seront transformées en tables dans une base de données. JPA permet aux programmeurs d’utiliser des notions avancées en base de données tout en restant programmeurs.

* Exemples de quelques entités :

**@Entity** signifie que cette classe est une table. JPA créera une table, de même nom que la classe, dans notre base de données.

**@Id** signifie que l’attribut qui est en dessous (idClasse) est la clé primaire de la table classe.

**@GeneratedValue** signifie que la clé primaire est automatiquement générée par le **SGBD**.

**@OneToMany** signifie que cette classe (table) est liée à plusieurs Utilisateurs (1 à plusieurs).

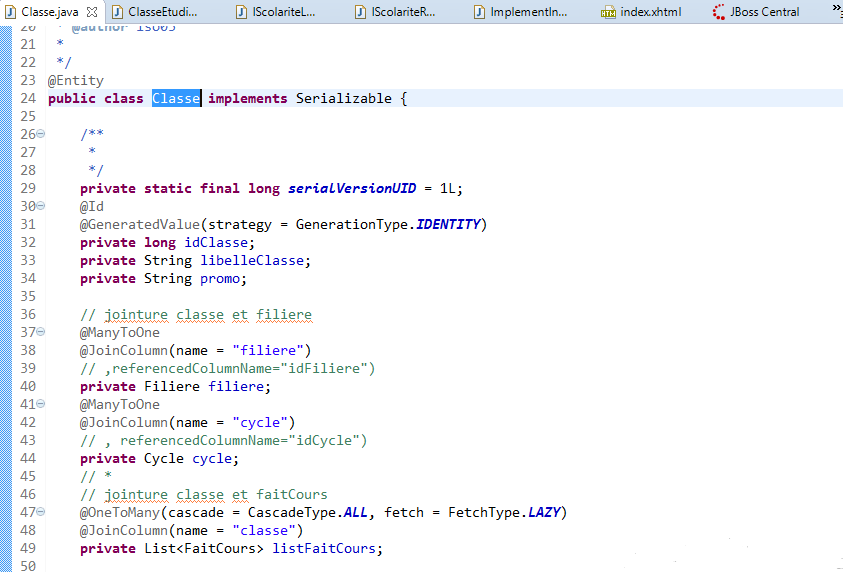


Figure 12 Entité Classe

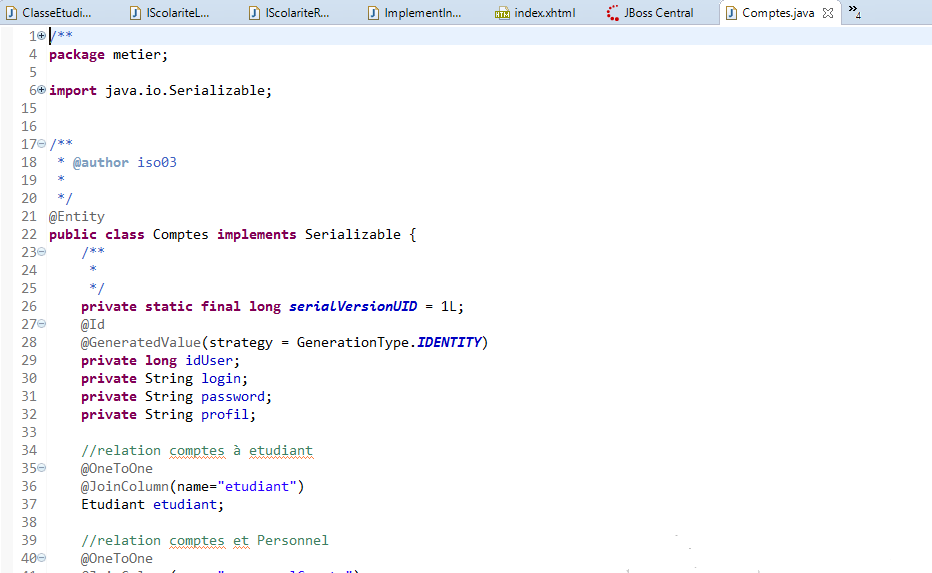


Figure 13 Entité Comptes

## Codage des interfaces

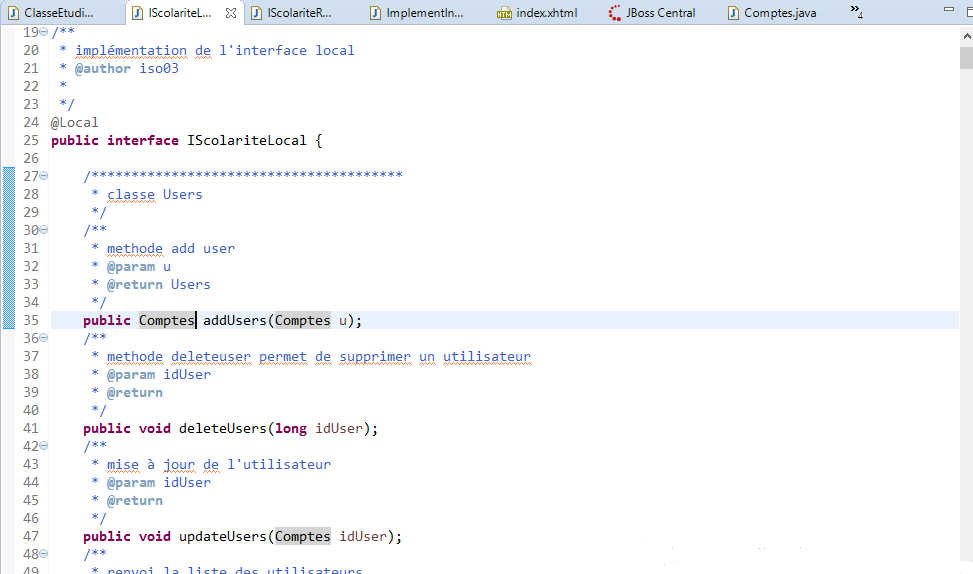
* L’interface locale

Figure 14 Interfaces Local : IScolariteLocal

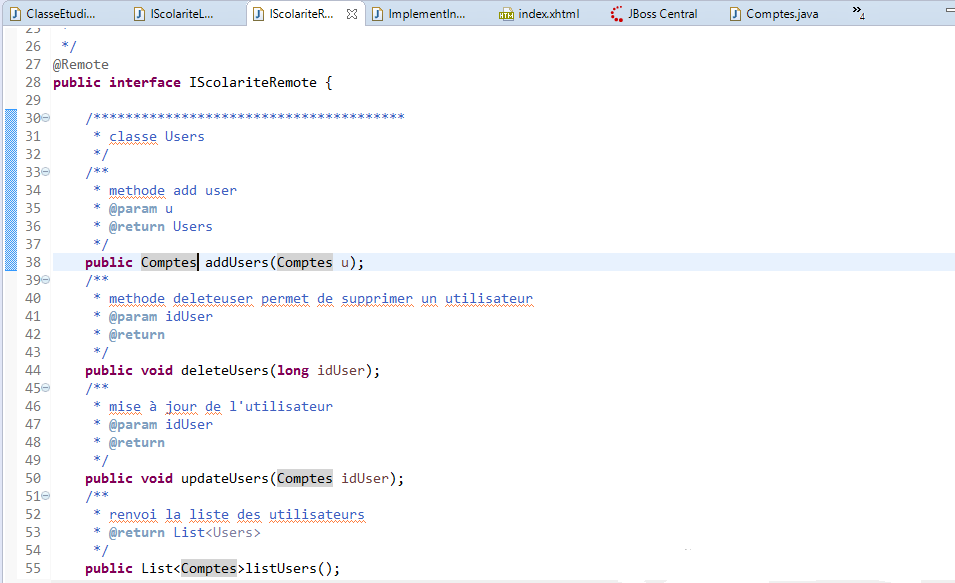
* L’interface distante Remote

Figure 15 Interface Remote : IScolariteRemote

## Implémentation des interfaces

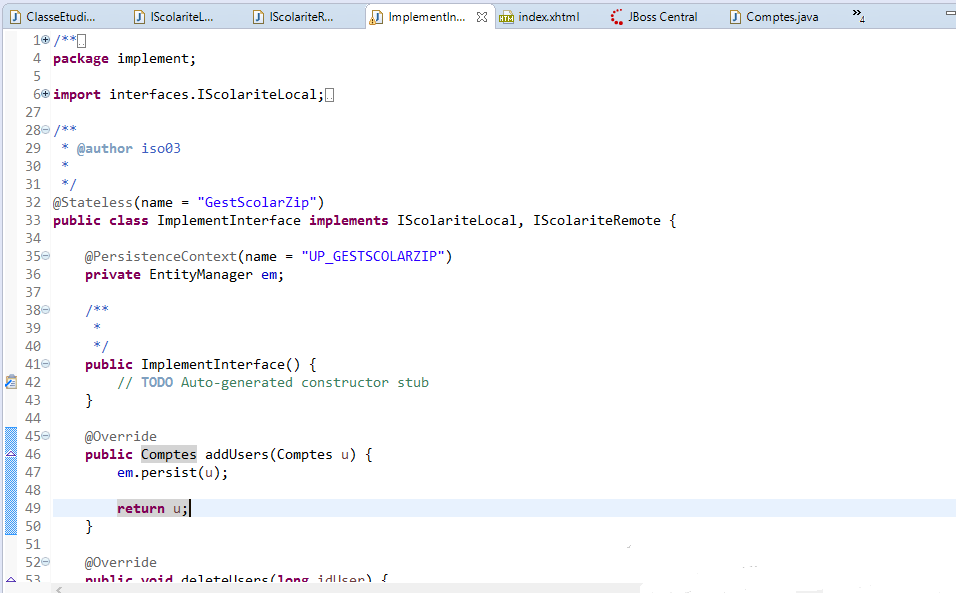


Figure 16Implémentation des interfaces IScolariteLocal et IScolariteRemote

Chapitre 4 : Intégration Continue

# Introduction

L'intégration continue est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée. Le concept a pour la première fois été mentionné par ***Grady Booch1*** et se réfère généralement à la pratique de ***l'eXtreme-Programming***. Le principal but de cette pratique est de détecter les problèmes d'intégration au plus tôt lors du développement. De plus, elle permet d'automatiser l'exécution des suites de tests et de voir l'évolution du développement du logiciel.

L'intégration continue est de plus en plus utilisée en entreprise afin d'améliorer la qualité du code et du produit final. Il y’a différents types de tests :

* **Test Fonctionnels** :
  + Est-ce que le logiciel est conforme à la spécification ?
  + Liés à la spécification, à la qualité, à la performance, l’interfaçage…
* **Test Non-fonctionnels**:
  + Est-ce que son usage est conforme ?
  + Liés à la configuration, à la documentation, à la compatibilité…
* **Test structurels** :
  + Est-ce le codage est correct ?

# Les différentes étapes du test des logiciels

Le test commence de suite ! Ici le cycle en V (***Figure 2 cycle de vie en V***)

Mais aussi approches incrémentales, agiles…

Les différents niveaux :

* **Tests de recette:** test de réception du logiciel chez le client final
* **Tests intégration système**: test de l’intégration du logiciel avec d’autres logiciels
* **Tests système**: test d’acception du logiciel avant livraison (nouvelle version par exemple)
* **Tests Intégration**: test de l’intégration des différents composants (avec ou sans hardware)
* **Tests Unitaires**: tests élémentaires des composants logiciels (une fonction, un module, ...)

# Test unitaire de la plateforme

## Les outils de test

Pour effectuer nos différents tests unitaires des codes sources on aura besoin d’une bibliothèque **JUnit** qui est un **Framework** open source pour le développement et l'exécution de tests unitaires automatisables. Le principal intérêt est de s'assurer que le code répond toujours aux besoins même après d'éventuelles modifications. Plus généralement, ce type de tests est appelé tests unitaires de non régression.

**JUnit** propose :

* Un **Framework** pour le développement des tests unitaires reposant sur des assertions qui testent les résultats attendus
* Des applications pour permettre l'exécution des tests et afficher les résultats

Le but est d'automatiser les tests. Ceux-ci sont exprimés dans des classes sous la forme de cas de tests avec leurs résultats attendus. JUnit exécute ces tests et les comparent avec ces résultats.

Cela permet de séparer le code de la classe, du code qui permet de la tester. Souvent pour tester une classe, il est facile de créer une méthode ***main()*** qui va contenir les traitements de tests. L'inconvénient est que ce code "superflu" est inclus dans la classe. De plus, son exécution doit se faire manuellement.

La rédaction de cas de tests peut avoir un effet immédiat pour détecter des bugs mais surtout elle a un effet à long terme qui facilite la détection d'effets de bords lors de modifications.

Les cas de tests sont regroupés dans des classes Java qui contiennent une ou plusieurs méthodes de tests. Les cas de tests peuvent être exécutés individuellement ou sous la forme de suites de tests.

**JUnit** permet le développement incrémental d'une suite de tests. Avec **JUnit**, l'unité de test est une classe dédiée qui regroupe des cas de tests. Ces cas de tests exécutent les tâches suivantes création d'une instance de la classe et de tout autre objet nécessaire aux tests appel de la méthode à tester avec les paramètres du cas de test comparaison du résultat attendu avec le résultat obtenu : en cas d'échec, une exception est levée.

**JUnit** est particulièrement adapté pour être utilisé avec la méthode ***eXtreme Programming*** puisque cette méthode préconise, entre autre, l'automatisation des tâches de tests unitaires qui ont été définies avant l'écriture du code. La version utilisée dans ce projet est la 4.

## Installation de JUnit

**JUnit** est intégré par défaut dans les environnements de développements intégré (**IDE**) **Java** tels que ***BlueJ, Eclipse et Netbeans***.

Première étape on fait un clic droit sur le projet, puis sur **Build Path** et ensuite ***Configuration Build Bath…***.

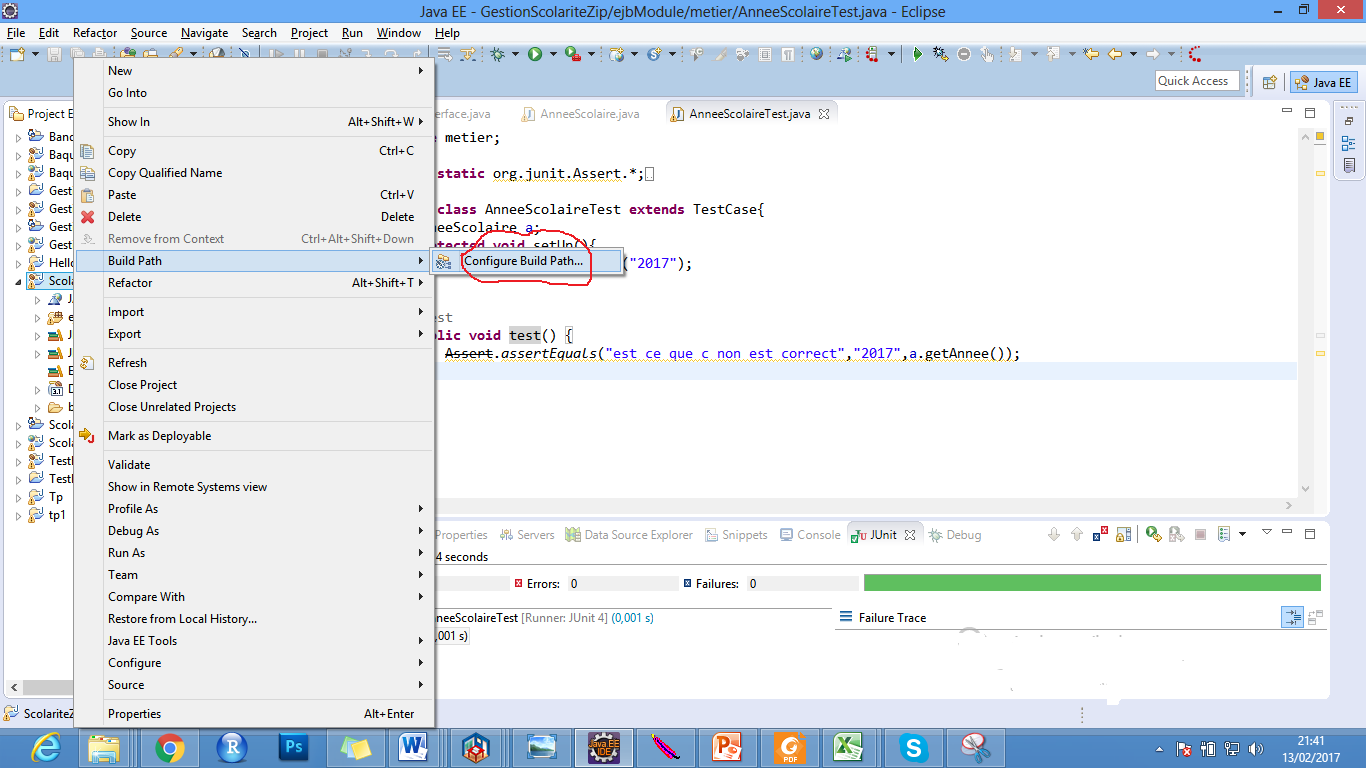


Figure 17 ajout de la librairie JUnit

Une fenêtre s’ouvre, on clique sur ***Add Library***, on choisisse ***JUnit*** puis on clique sur ***Next***

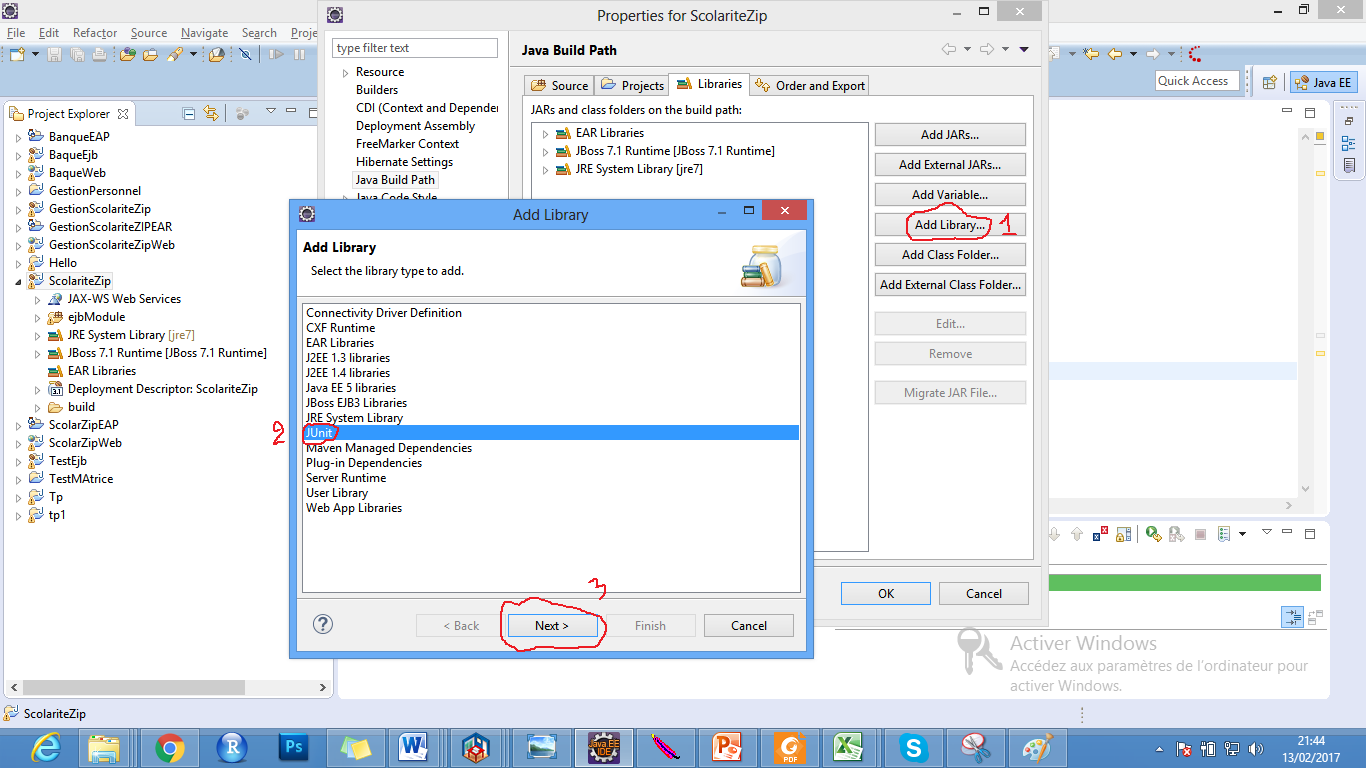


Figure 18 librairie JUnit 2

Et on choisit la dernière version de ***JUnit***. Ici on a utilisé la version 4.11. Puis on clique sur ***Finish***

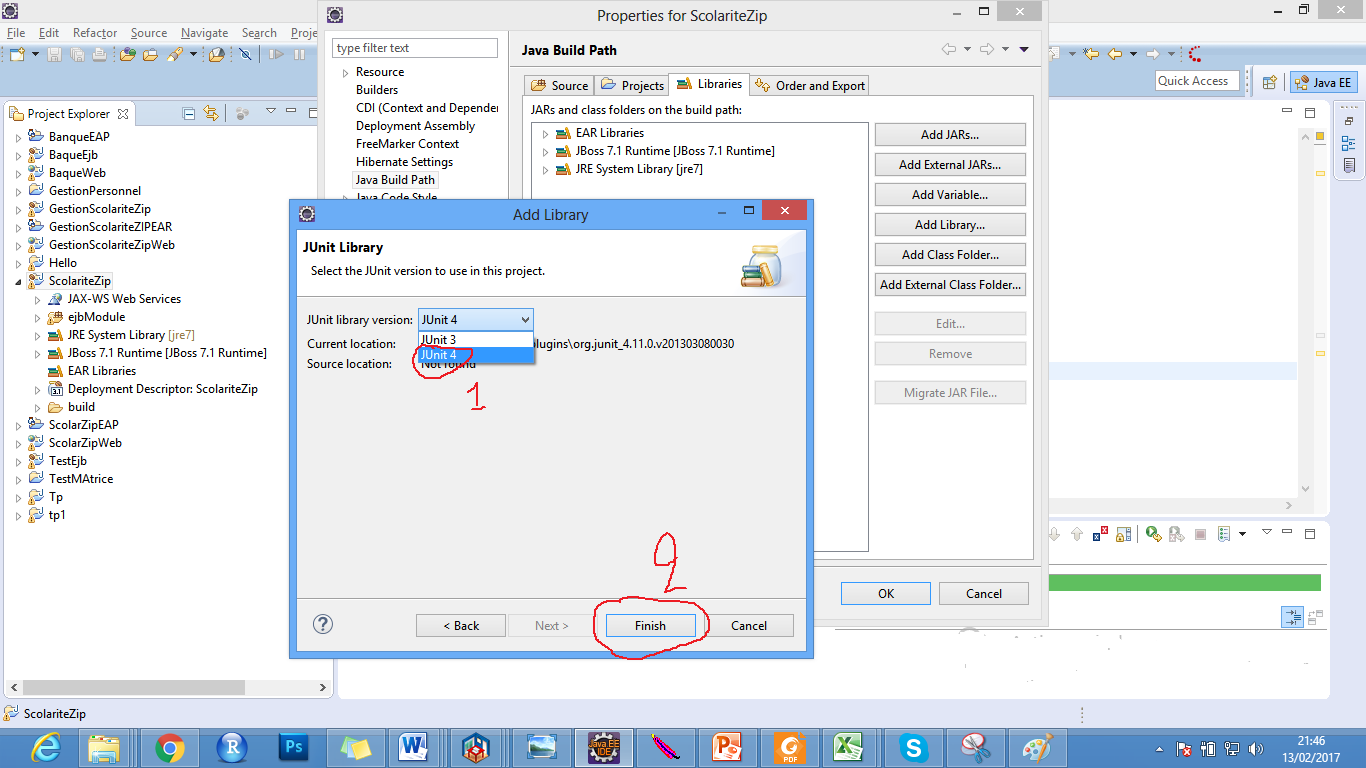


Figure 19 Librairi JUnit 3

On revient dans notre fenêtre principale on s’assure que la librairie est bien ajoutée puis on valide en cliquant sur ***Ok***

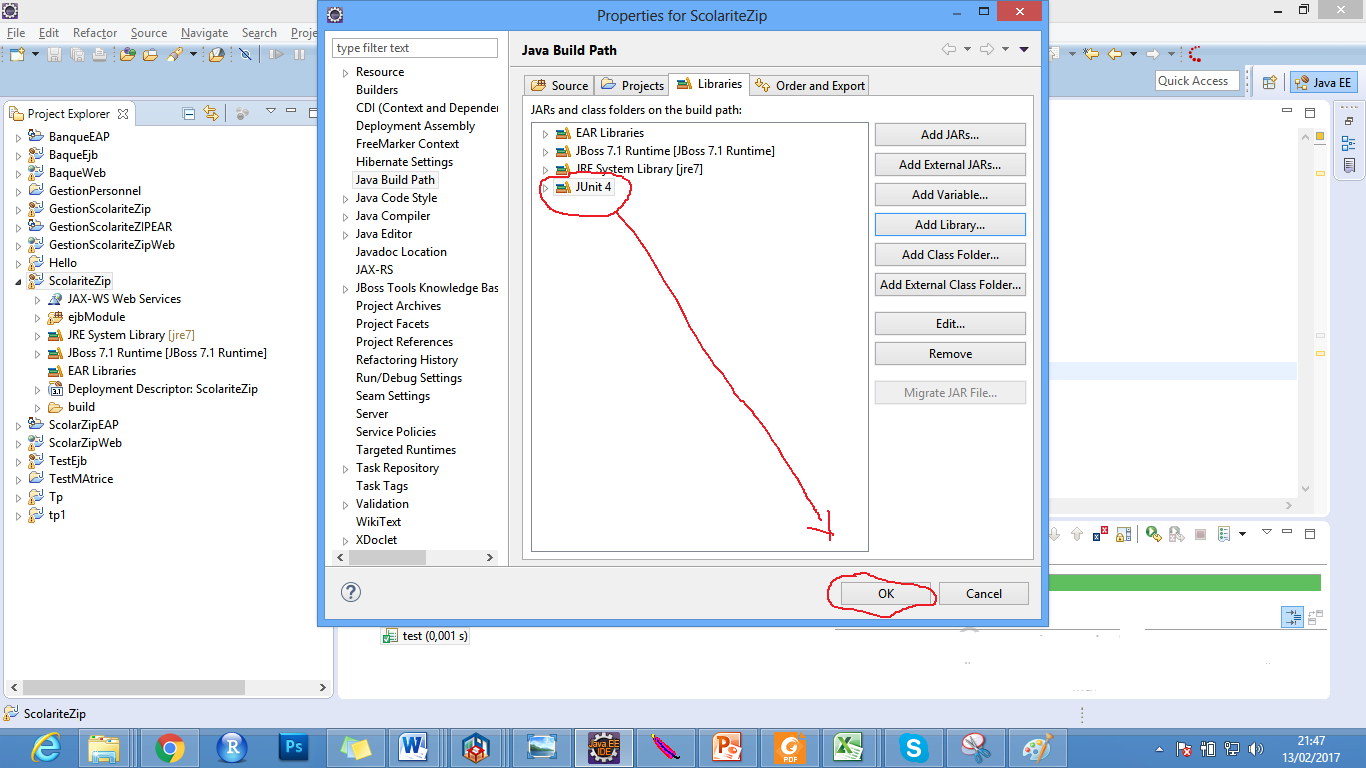


Figure 20 Librairie JUnit 4

## Codage des Classes de tests

**JUnit** définit deux types de fichiers de tests. Les **TestCase** sont des classes contenant un certain nombre de méthodes de tests. Un **TestCase** sert généralement à tester le bon fonctionnement d'une classe. Une **TestSuite** permet d'exécuter un certain nombre de **TestCase** déjà définis.

### Exemple de la classe TestCase

Une telle classe hérite de ***junit.framework.TestCase***. La méthode annotée par ***@Before*** est exécutée avant les méthodes de test, celle précédée par ***@After*** est appelée à la fin. De la même manière, la méthode annotée par ***@BeforeClass*** est appelé au lancement du testCase, celle précédée par ***@AfterClass*** est appelée juste avant la fin. Les tests sont des méthodes annotées par ***@Test***, elles font des traitements et vérifient le bon comportement des classes testées par des méthodes ***assert\*\*\*(),*** toute assertion non vérifiée est signalée comme défaillante.

### Test des classes métier

#### La classe Comptes

Pour créer une classe de test pour Comptes il faut faire clic droit sur la classe puis sur New et Other

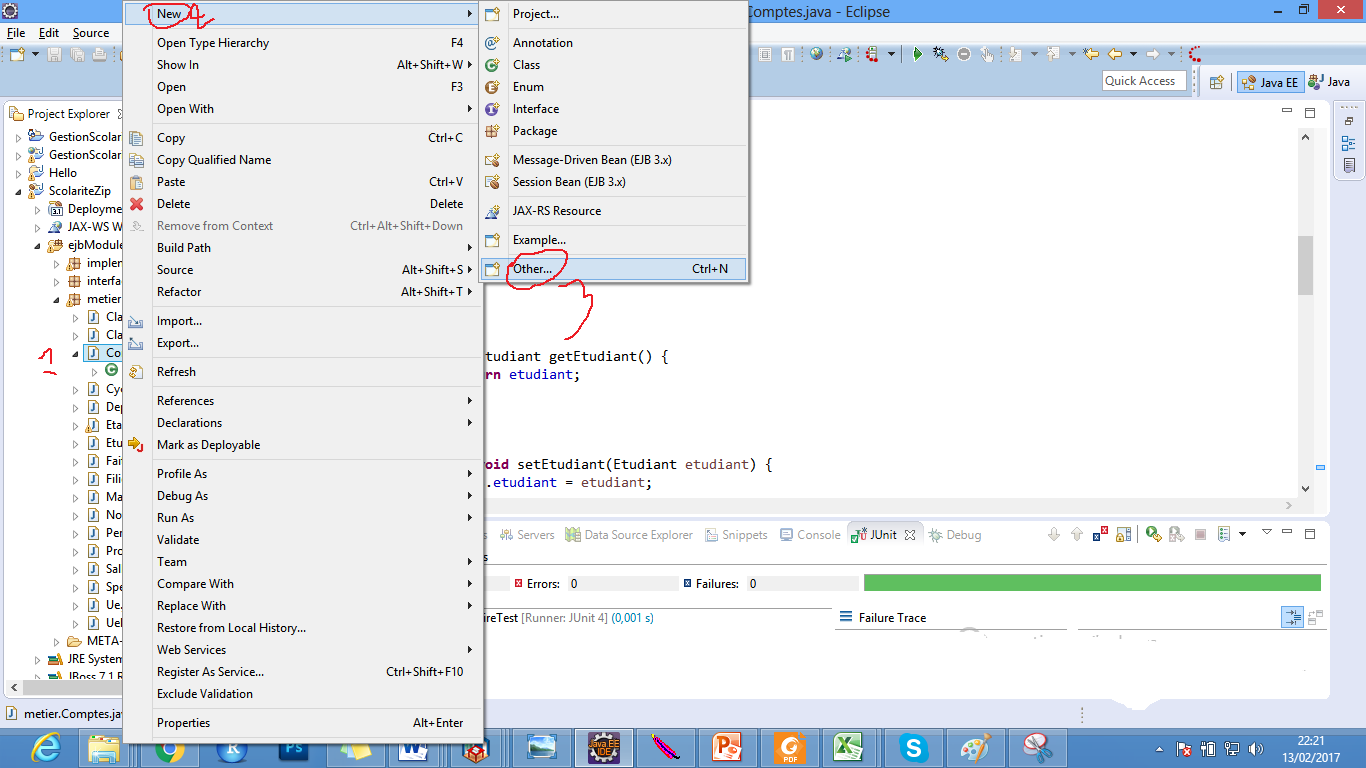


Figure 21 classe de test de la classe Comptes

Une fenêtre s’ouvre, on choisit JUnit TestCase

22

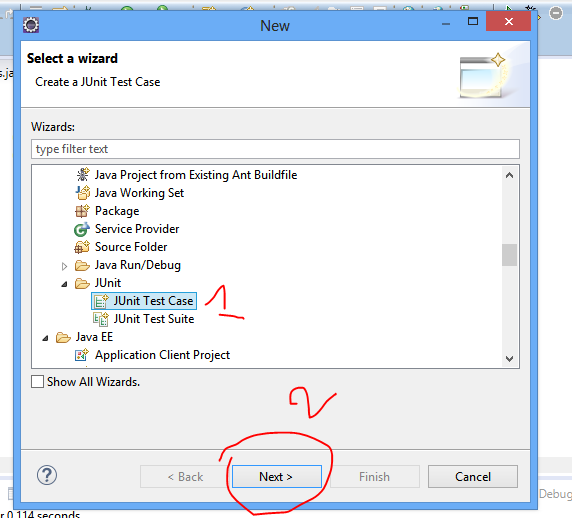


Figure 23 classe de test

Ensuite on écrit nos différentes méthodes de test

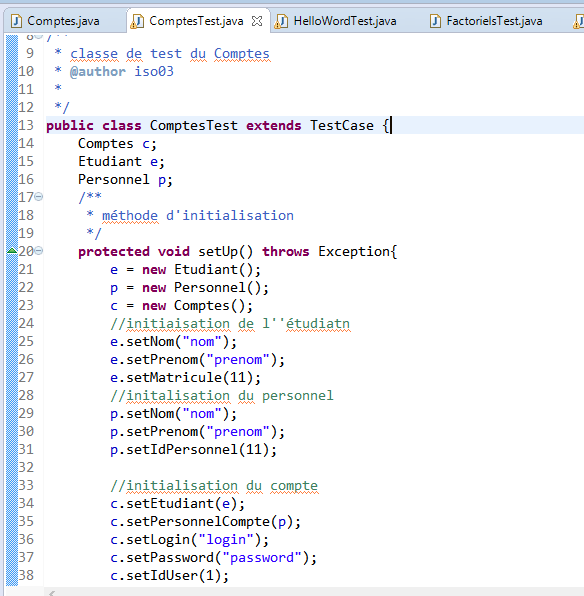


Figure 24 Définition des méthodes de test

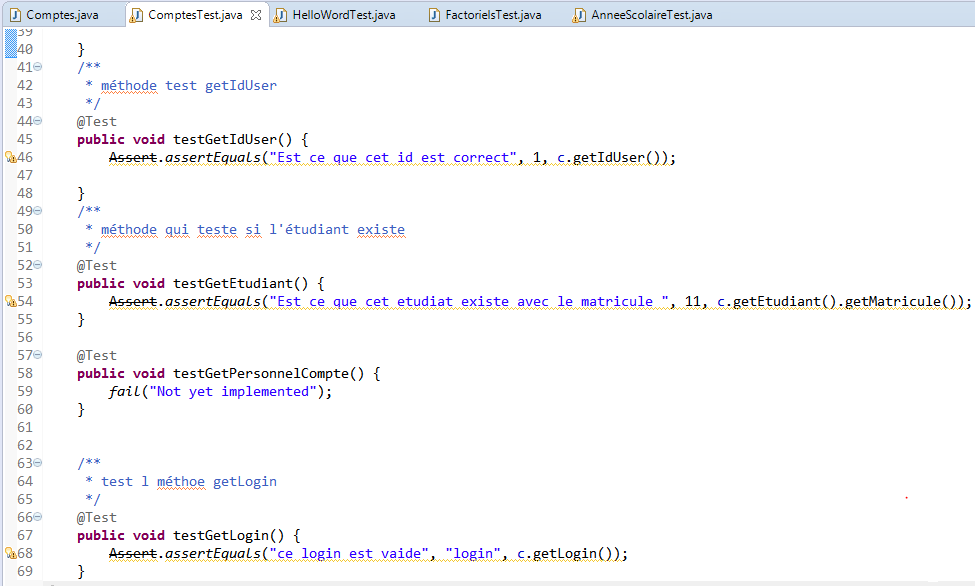


Figure 25 les méthodes de tests pour la classe Comptes

Une fois que toutes les méthodes sont définies, on doit lancer l’exécution du test. Pour ce faire on fait un clic droit la classe de test, puis sur ***Run As*** et on clique sur ***JUnit Test***

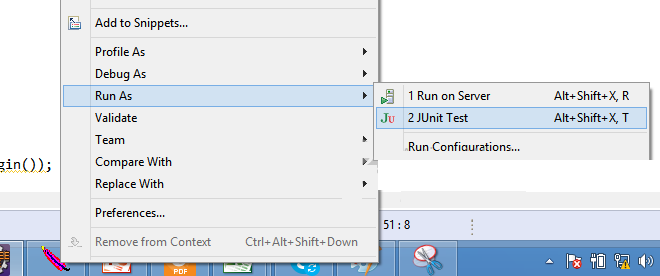


Figure 26 Exécution du test

Et si l’exécution se passe normalement on aura cette fenêtre avec ***0 Errors*** et ***0 Failures*** plus une bande verte indiquant que le code ne contient pas d’erreurs.

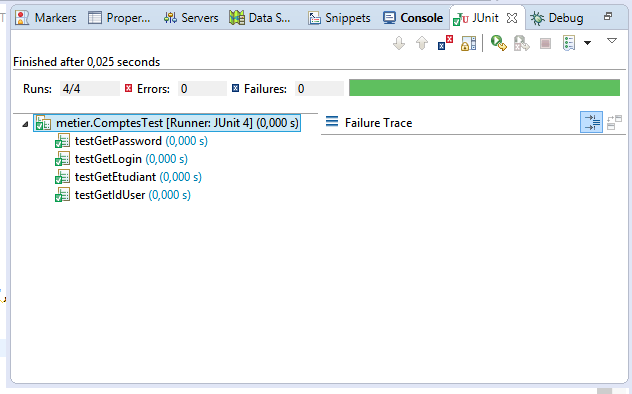


Figure 27 Résultat de l’exécution du test

Chapitre 5 : Interface Homme Machine (IHM)

# Présentation de l’application Gestion de Scolarité

L’application que nous avons conçue permet d’effectuer plusieurs tâches telles que :

* Ajouter un utilisateur
* Inscription d’un étudiant
* Enregistrement des informations relatives à l’établissement
* Ajouter une nouvelle filière
* Gérer le paiement mensuel des étudiants
* Etablir un emploi du temps…

## Menus de l’application

Notre application comporte plusieurs menus tels que:

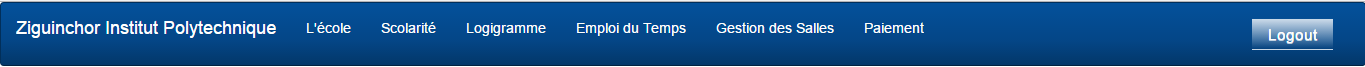


Figure 28 Menu principal du gestionnaire



Figure 29 menu Administrateur

## Les interfaces de l’application

Ci-dessous nous avons l’espace gestionnaire qui permet de gérer les professeurs (affichages de la liste des professeurs, ajouter un nouveau professeur, affecter des modules aux professeurs, modification) **Erreur ! Source du renvoi introuvable.**.

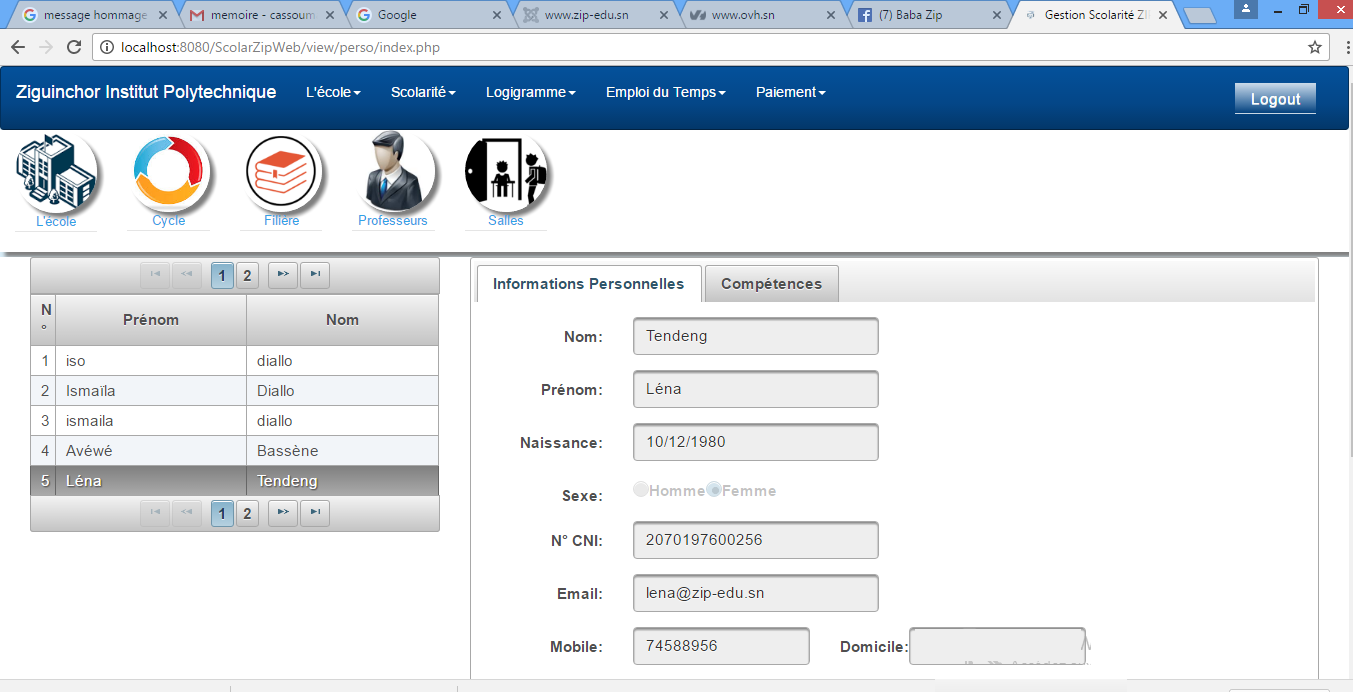


Figure 30 interface de gestion des professeurs

L’espace administrateur dans cette interface on gère les comptes des utilisateurs (compte étudiant, professeur …) **Erreur ! Source du renvoi introuvable.**.

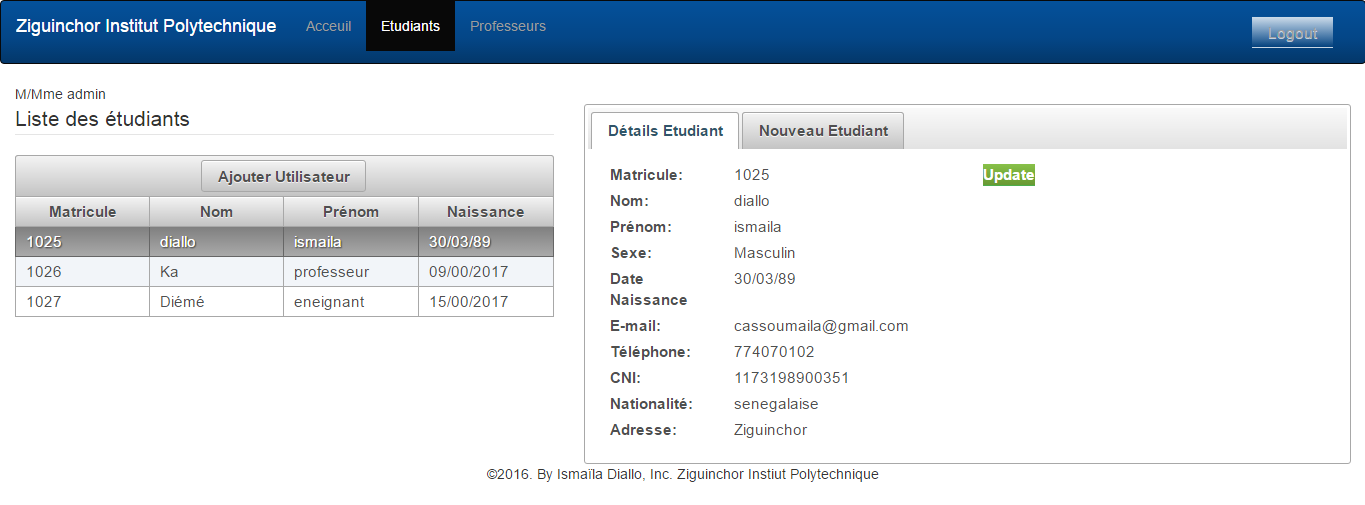


Figure 31 Gestions des utilisateurs

Espace réservé à la gestion de l’école (lis informations concernant l’établissement, la gestion des cycles, gestion des filières, gestion des professeurs et gestion des salles **Erreur ! Source du renvoi introuvable.**.

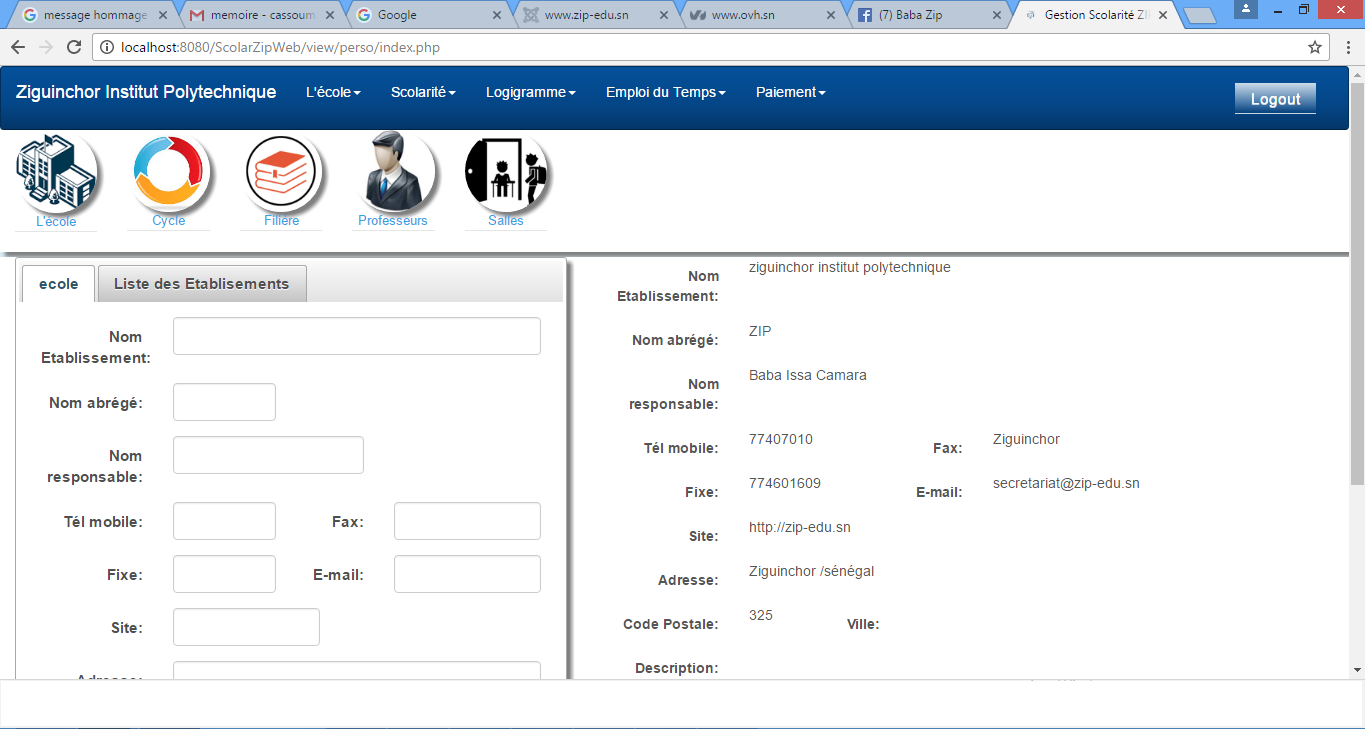


Figure 32 Gestion de l’école

Espace réservé à la gestion de la scolarité (gestion des classes, gestion des étudiants, gestion des inscriptions) **Erreur ! Source du renvoi introuvable.**.

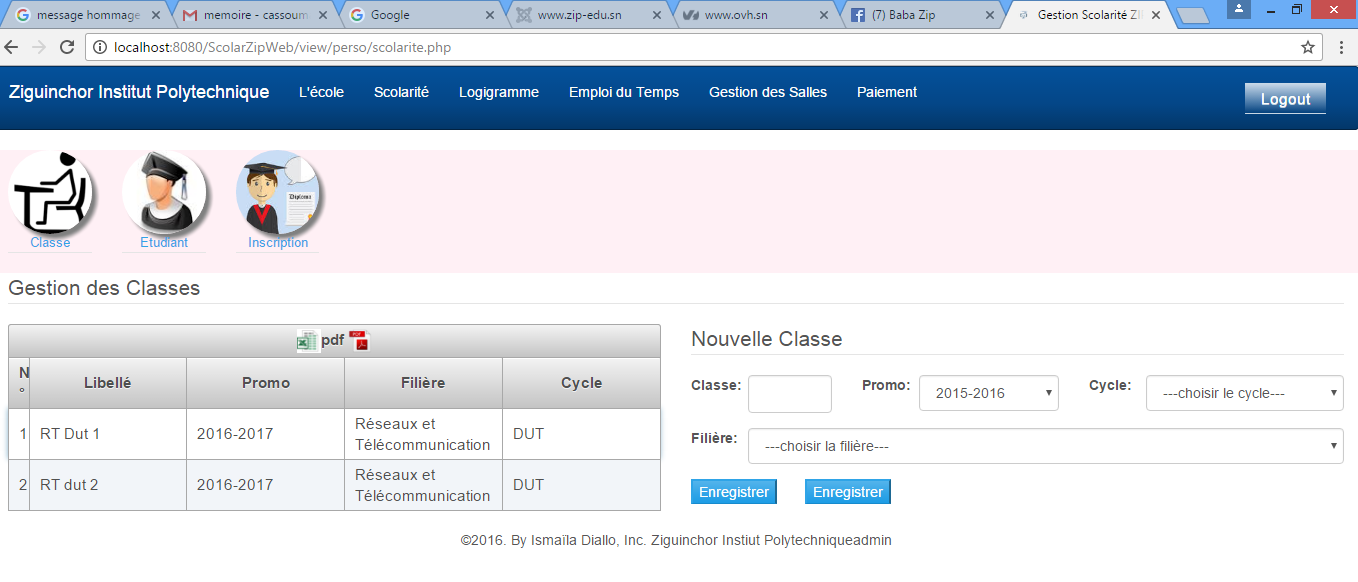


Figure 33 Gestion Scolarité

# Présentation du site Web <http://zip-edu.sn>

## Les menus

L’application ne dispose que deux menus l’un en « back-office » et l’autre en « font-office ». Il s’agit de :

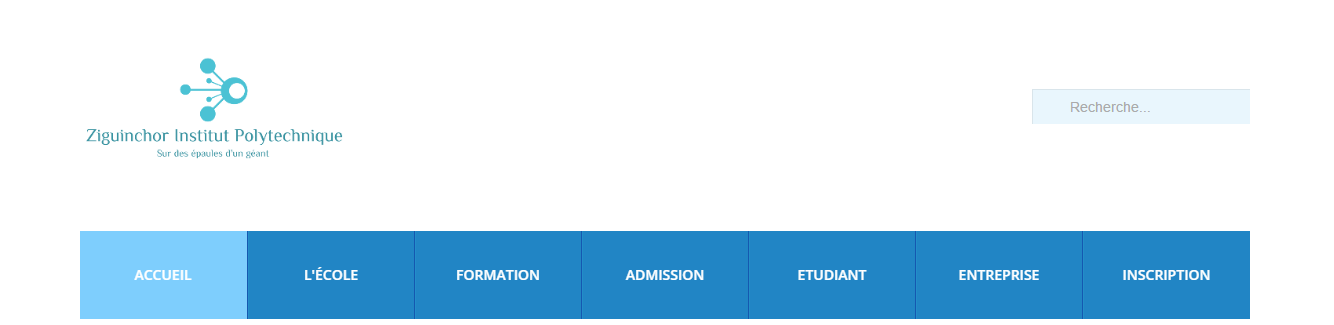


Figure 34 Menu « front-office » du site web

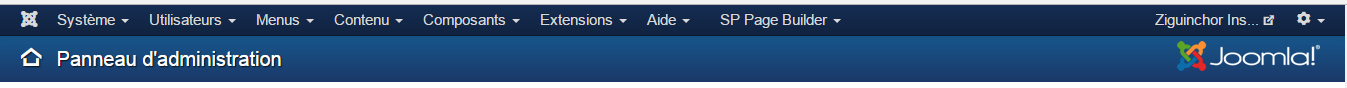


Figure 35 Menu « back-office » du site web

## Les interfaces

* Accueil

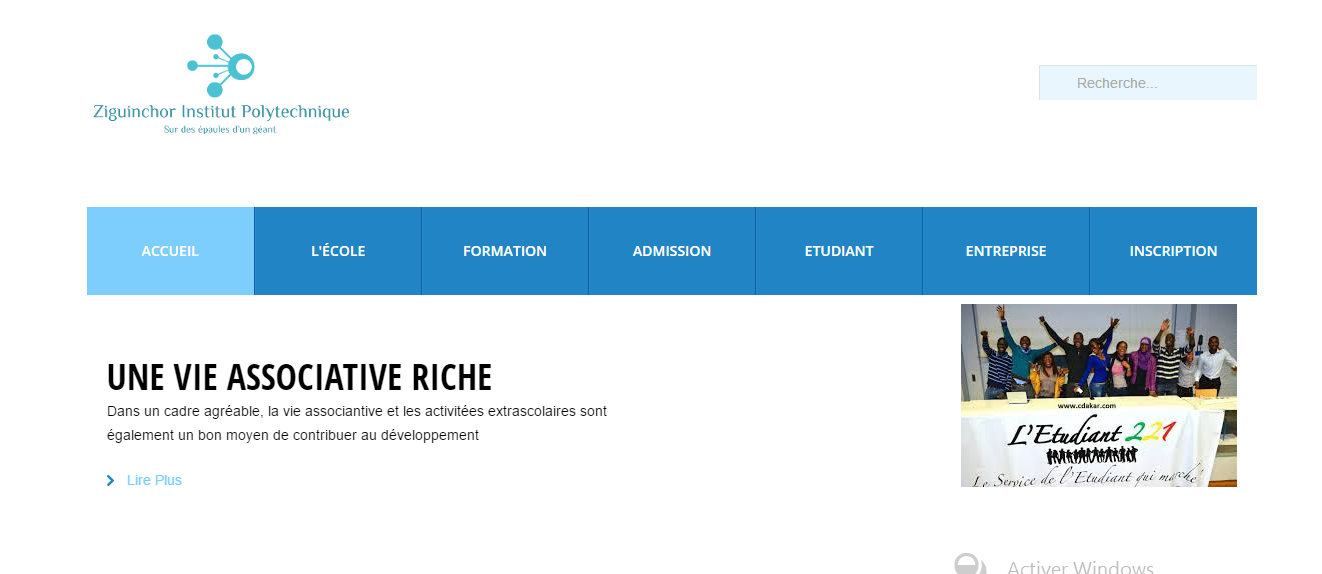


Figure 36 Interface Accueil

* Formation



Figure 37 Formation

* Inscription

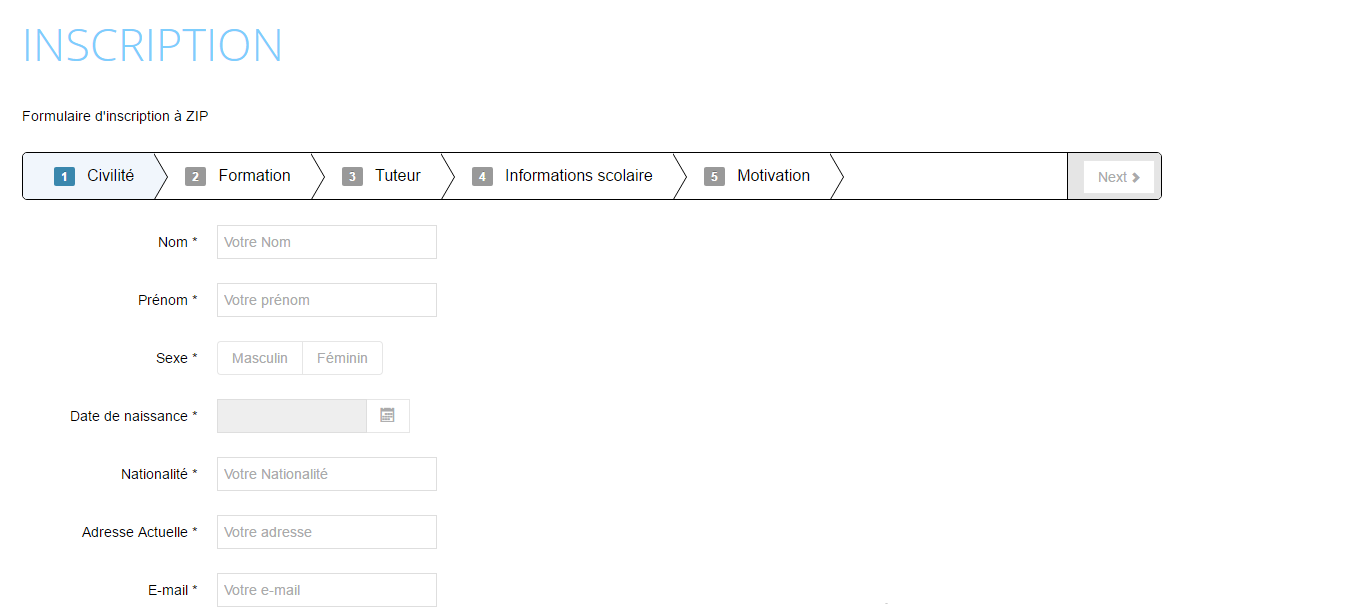


Figure 38 Inscription d’un étudiant

# Conclusion

Du 15 Février 2015 au 15 Décembre 2015, pour une durée de 9 mois, j’ai effectué un stage au sein de l’entreprise **ZIP** qui est un institut d’enseignement supérieur polytechnique (située au quartier de ***Diabir/Ziguinchor*** en face de l’université ***Assane Seck de Ziguinchor***). Au cours de ce stage au département Informatique de **ZIP**, j’ai pu m’intéresser essentiellement sur le développement et la mise en place de deux plates-formes (un site Web <http://zip-edu.sn> et d’une application de gestion de scolarité **ScolarZip**).

Plus largement, ce stage a été l’opportunité pour moi d’appréhender le fonctionnement général d’une école d’enseignement supérieur mais aussi de parfaire mes compétences (connaissances) acquises tout au long de mon cursus universitaire que ce soit sur le domaine de l’informatique, de l’électronique ou de mathématique.

Au-delà d’enrichir mes connaissances en Informatique (génie logiciel), ce stage m’a permis de comprendre dans quelle mesure il a influencé mon futur parcours professionnel. En effet j’ai été séduit par les enseignements qui y sont dispensés notamment dans le domaine de l’électronique et l’informatique combinés avec le **kit Arduino** qui permet de faire pas mal de choses (création de jeux de lumières, création de système d’alerte, des drones…) et à l’avenir je comptes faire ma thèse spécialement dans le domaine de l’électronique et l’informatique (programmation) soit dans le domaine de création de drones soit dans l’internet des objets.

Mon stage chez ZIP a été aussi très instructif. Au cours de ces 9 mois, j’ai ainsi pu observer le fonctionnement d’une école d’enseignement supérieur. Au-delà, de l’activité de chacun des services, j’ai pu apprendre comment s’articulent les différents départements d’une telle entreprise. Par ailleurs, les relations humaines entre les différents employés de la société, indépendamment de l’activité exercée par chacun d’eux, m’a appris sur le comportement à avoir en toute circonstance.

Cependant des perspectives d’améliorations de notre application restent envisageables. D’autant plus c’est une application intranet

# Bibliographiques

[1] Rie. Cours et Exercices.

[2] Les bases de l’informatique et de la programmation. 2005. Berti d’alger  
édition.

[3] Uml 2 pour les bases de données. 2007. Christian Soutou.

[4] Définition et caractéristique d’UML. 2008.

[5] *Pratique de MySQL et PHP*. Dunod, 2009. Conception et réalisation de site  
web dynamique.

[7] Les diagrammes UML. Février 2001. Les conséquences du passage à l’OO Annexe. Division Consultance - Section des Recherches.

[8] Merise et UML pour la modélisation des systèmes d’information. Mars 2004.  
Volume 5. Dunod édition,

[9] De mérise à UML. Octobre 2001. Dominique Mignet

# Index

@

***@After*** 35

***@AfterClass*** 35

***@BeforeClass*** 35

***@Test*** 35

A

acteurs **8**, **10**, **11**

***assert\*\*\*(),*** 35

B

besoins **8**, **10**, **21**, **26**

**Build Path** 32

C

**classes** **10**, **15**, **26**, **27**, **42**

**Contrôleurs** **18**

Couche métier **25**

D

Définition **8**, **10**, **45**

Dut en Gestion des Entreprises et des Administrations (DUT-GEA) **3**

Dut en Gestion Logistique et Transport (DUT-GLT) 3

Dut en Informatique et Statistique Décisionnelle (DUT-ISD) 3

Dut en Réseaux et Télécommunications (DUT-RT) 3

E

EJB **25**, **26**

**Entités** **26**, **27**

**Entity** **25**, **28**

étudiants **7**, **8**, **9**, **10**, **12**, **20**, **40**, **42**

***eXtreme Programming*** 32

F

Figure **11**

**Framework** 32

G

**Gestion** **3**, **9**, **42**

I

Institut Polytechnique 3, 20, 21

Introduction **3**, **20**, **24**

J

**Java** 24, 25, 26, 27, 32

Java EE **25**, **26**

**Java Persistence API** **27**

**Jboss as** **26**

**JPA** **27**, **28**

**JUnit** 32, 33, 34, 35, 38

L

Licences Professionnelles **3**

**Locale** **26**

M

***main()*** 32

**modèle** **15**, **18**

P

***persistantes*** **26**

R

**Remote** **26**, **30**

S

**ScolarZipWeb** **3**, **4**

**SGBD** **14**, **15**, **27**, **28**

T

**TestCase** 34, 35

U

**UML** **8**, **10**, **45**, **46**

V

**vue** **3**, **18**, **20**

Z

Ziguinchor **3**, **20**, **21**, **22**

Zip **3**, **4**