Université Assane Seck de Ziguinchor

UFR Sciences et Technologies

Département Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de master

Mention : Informatique ; Spécialité : Génie Logiciel

Sujet :

Conception et développement d'une application pour la gestion de la répartition des enseignements à l'UASZ

Présenté par :

M. Henry DIALLO

le 07/11/2018

Membres du jury :

- Pr Alassane DIEDHIOU (**Président**)
- Dr Marie NDIAYE (**Encadrante**)
- Dr Ibrahima DIOP (**Co-encadrant**)
- Dr Youssou FAYE (**Rapporteur**)
- Dr Khadim DRAME (**Rapporteur**)
- Mr Alousseynou FALL (**Maître de stage au CRI**)

Sous la direction de :

- Dr Marie NDIAYE
- Dr Ibrahima DIOP
- Mr Alousseynou FALL

Sous la supervision de :

- Pr Alassane DIEDHIOU

Année Universitaire 2017-2018

RESUME

L'Université Assane SECK de Ziguinchor (UASZ) utilise depuis très longtemps des fichiers Excel (Voir **Annexe 3**) pour la gestion de la répartition des enseignements. Ces fichiers ne répondent pas entièrement aux attentes des utilisateurs, car, non seulement, ils renferment beaucoup de manquements en terme de fonctionnalités (détection des conflits lors des choix, visualisation des répartitions pour chaque département, etc.), mais aussi des problèmes en terme d'usage (manque d'autonomie, de commodité et de rapidité dans le processus, difficulté de constituer le fichier final des répartitions).

Dans ce mémoire, nous proposons ainsi une application dans le but de répondre aux préoccupations de l'UASZ dans ce domaine. Pour ce faire, des études ont été menées ainsi que des séries d'entretiens avec les utilisateurs (Vice-recteur, Chefs de département et Enseignants). Nous avons aussi fait appel à quelques pratiques de la méthodologie Scrum pour le processus de développement de notre application. Une spécification des besoins nous a permis de formaliser les données avec UML (Unified Modeling Language). En ce qui concerne la structuration et le stockage des données, nous avons utilisé le Système de Gestion de Bases de Données Relationnelles (SGBDR) MySQL. L'implémentation en PHP nous a permis de mettre en place une application souple, sécurisée, facile à utiliser et permettant de gérer parfaitement la répartition des enseignements à l'UASZ.

DEDICACES

Je rends grâce à Allah de m'avoir donné la capacité de réfléchir,

la force d'y croire, et la patience d'aller jusqu'au bout du rêve.

Je dédie ce modeste travail :

A ma très chère maman en signe d'amour, de reconnaissance et de gratitude pour tous les soutiens et les sacrifices dont elle a fait preuve à mon égard ;

A mon père qui m'a toujours soutenu et encouragé ;

A mes sœurs et frères qui m'ont toujours soutenu et encouragé ;

A tous les membres de ma famille : tantes, oncles, cousins maternels et paternels ;

A ceux que j'aime, qui m'ont toujours soutenu et étaient toujours à mes cotés : mes amis, Adama SEYE, Ibrahima SENE, Boubacar BALDE ;

A tous ceux qui ont contribué de près ou de loin pour que ce projet soit réalisé.

REMERCIEMENTS

Nous tenons à remercier en premier lieu le bon Dieu de nous avoir donné la force et le courage d'accomplir ce travail.

Nous tenons profondément à remercier tous les enseignants qui ont participé à notre formation.

Notre profonde gratitude et sincères remerciements vont à nos professeurs Dr Marie NDIAYE et Dr Ibrahima DIOP pour nous avoir confié ce travail, pour leur suivi, leur disponibilité, leurs orientations et leurs conseils.

Nos remerciements vont aussi aux membres de jury qui ont accepté d'évaluer ce modeste travail.

Nous tenons fermement à mentionner le plaisir que nous avons eu à faire notre stage au sein du Centre des Ressources Informatiques de l'Université Assane SECK de Ziguinchor ; Merci à M. Alousseynou FALL pour son suivi, ses orientations et ses précieux conseils.

Nous n'oublions pas de remercier, vivement, tous ceux qui ont contribué, de prés ou de loin à réaliser ce mémoire de fin d'études.

SOMMAIRE

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABREVIATIONS

UASZ :	Université Assane Seck de Ziguinchor
CM :	Cours Magistral
TD :	Travail Dirigé
TP :	Travail Pratique
UML :	Unified Modeling Language
SGBD :	Systéme de Gestion de Bases de Données
SGBDR :	Systéme de Gestion de Bases de Données Relationnelles
UFR :	Unité de Formation et de Recherche
UE :	Unité d'Enseignement
EC :	Elément Constitutif
CRI :	Centre des Ressources Informatiques
Vice-rectorat EVU :	Vice-rectorat chargé des Etudes et de la vie de l'université
LMD :	Licence Master Doctorat
DERU :	Direction des Etudes et des Réformes Universitaires
DIVU :	Direction de la Vie de l'Université
PATS :	Personnel Administratif, Technique et de Service
PER :	Personnel Enseignant et de Recherche
PATSF :	Personnel Administratif, Technique et de Service Fonctionnaire
FOAD :	Service de Formation Ouverte et à Distance
BU :	Bibliothèque Universitaire
DSE :	Direction du service aux Etudiant
DAF :	Direction des Affaires Financières

DCS : Direction Centrale de la Scolarité

DGPM : Direction de la Gestion du Patrimoine et de Maintenance

DES : Direction de l'Environnement et de la Sécurité

DRH : Direction des Ressources Humaines

INTRODUCTION GENERALE

La répartition des enseignements universitaires consiste à attribuer, à chaque enseignant, des enseignement(s) pouvant être des CM (Cours Magistraux), des TD (Travaux Dirigés) et/ou des TP (Travaux Pratiques). La répartition des enseignements est un processus périodique et obligatoire permettant de connaître, pour chaque semestre donné, les enseignements qui doivent être dispensés par chaque enseignant ainsi que son nombre d'heures total d'enseignements.

À l'UASZ, le processus de répartition des enseignements se fait par département. On note que dans les départements, la répartition se fait en utilisant des fichiers Excel contenant les enseignements d'un semestre ou d'une année donnée. Ces fichiers sont envoyés à tous les enseignants pendant la période de répartition pour que chacun choisisse des enseignements.

L'utilisation de ces fichiers n'est pas très pratique et ne permet pas de gérer certaines tâches et présente beaucoup de manquements. En effet, ces fichiers dont les départements de l'UASZ se servent depuis très longtemps pour gérer les répartitions des enseignements est problématique, car ils requièrent beaucoup de tâches répétitives (l'envoi mutuel du fichier entre les acteurs du système) et manuelles (le réglage des conflits) qui doivent être automatiques.

Pour résoudre ce problème, l'UASZ, à travers le CRI, en collaboration avec les différents départements, a jugé nécessaire de mettre en place une application optimale et fiable pour la gestion automatique de la répartition des enseignements pour l'ensemble de ses départements. Cette application web permettra, d'abord, à tous les enseignants de pouvoir choisir, pour un semestre donné, des enseignements (CM, TD ou TP). Ensuite, de consulter les choix, de calculer la totalité des heures d'enseignements choisis par un enseignant. Enfin, de régler les éventuels conflits lors des choix d'enseignements, mais aussi de fournir assez de données pour la mise en place des emplois du temps.

Ainsi, afin de mettre en place une telle solution, nous avons fait un stage au CRI (Centre des Ressources Informatiques) de l'UASZ. L'objectif de ce stage était de concevoir et développer cette solution. Ainsi, une étude du système existant a été faite en organisant des entretiens avec des enseignants, le Vice recteur chargé des études et de la vie de l'université et des chefs de département de l'UASZ. Ces derniers nous ont fait part du mode de fonctionnement de la répartition au niveau des départements, du système existant ainsi que les problèmes rencontrés avec celui-ci. Face à cela, le Centre des Ressources Informatiques (CRI) a exprimé ses besoins

en précisant ses attentes sur la nouvelle application. Elle doit répondre à toutes les qualités de logiciel, permettre d'effectuer les répartitions des enseignements, de régler les conflits d'affectation, d'être accessible via l'internet et de communiquer par e-mail.

Pour identifier, analyser et concevoir les fonctionnalités de l'application, nous avons utilisé UML afin de représenter graphiquement les besoins exprimés ainsi que les données qui doivent être stockées dans un SGBD. Pour assurer l'interaction entre ces données, nous avons travaillé avec des outils ou technologies informatiques (PHP, HTML, CSS, Bootstrap et JQuery) adaptées nous permettant de bien faire le travail et d'atteindre nos objectifs.

Le travail qui a été réalisé est décrit dans la suite de ce document qui est organisé en cinq principaux chapitres :

- **Chapitre I:** ***Contexte justificatif du sujet*** expose les problèmes dans la répartition des enseignements à l'UASZ.
- **Chapitre II: P*rocessus de développement de l'application*** permet de faire connaissance au Framework Scrum adapté dans notre cas.
- **Chapitre III:** ***Spécification et l'analyse des besoins fonctionnels*** identifie les acteurs et fonctionnalités, analyse les besoins qui doivent être satisfaits par ce système
- **Chapitre IV:** ***Conception du système*** aborde la conception générale et la conception détaillée du système.
- **Chapitre V:** ***Implémentation et réalisation*** présente l'application ainsi que les outils utilisés pour la réaliser.

Pour bien finaliser ce travail, ce mémoire dispose d'une conclusion et des perspectives.

CHAPITRE 1 : CONTEXTE JUSTIFICATIF DU SUJET

Ce chapitre permet de mettre en évidence le contexte justificatif du sujet de notre mémoire. Il est composé de trois parties principales. D'abord, nous présenterons l'UASZ dans son ensemble. Ensuite, nous parlerons succinctement de la gestion des répartitions des enseignements à l'UASZ. Enfin, nous aborderons la problématique du sujet.

1.1 Présentation de l'UASZ

L'UASZ est créée par le décret 2008-537 du 22 mai 2008 comme troisième Université du Sénégal [1]. Elle fonctionne sous le système Licence Master Doctorat (LMD) et offre de nombreuses formations. Elle a démarré ses activités en février 2007 avec trois **UFR** (Unité de Formation et de Recherche), que sont :

- l'UFR des Sciences Economiques et Sociales ;
- l'UFR des Sciences et technologies ;
- l'UFR des Lettres, Arts et Sciences Humaines.

En 2011, une nouvelle UFR a été ajoutée : il s'agit de l'UFR des Sciences de la Santé.

Chacune des UFR dispose d'un **service chargé de la pédagogie** géré par un **chef de service pédagogique**, et de **départements** qui sont gérés par des **chefs de départements**. Ces départements proposent des **formations** qui sont gérées par des **responsables de formation**. Pour la bonne gestion de ces quatre entités (UFR, services pédagogiques, départements et formations) l'UASZ dispose :

- d'un Personnel Administratif, Technique et de Service (PATS),
- d'un Personnel Enseignant et de Recherche (PER),
- et d'un Personnel Administratif, Technique et de Service Fonctionnaire (PATSF).

De nombreux centres et services communs sont notés et ils ne cessent d'augmenter d'année en année. Ainsi, on distingue :

- le Service de Formation Ouverte et à Distance (FOAD);
- le Centre des Ressources Informatiques (CRI) ;

- la Bibliothèque Universitaire (BU);
- la Direction du service aux Etudiant (DSE);
- la Direction des Affaires Financières (DAF);
- la Direction Centrale de la Scolarité (DCS);
- la Direction de la Gestion du Patrimoine et de Maintenance (DGPM);
- la Direction de l'Environnement et de la Sécurité (DES);
- la Direction des Ressources Humaines (DRH).

Ainsi, depuis sa mise en place jusqu'à nos jours, l'UASZ évolue dans tous ses domaines d'activités. Cela a entraîné, d'année en année, une hausse du nombre d'étudiants orientés, du nombre de formations ouvertes, du nombre d'enseignants, ainsi que du nombre d'enseignements dispensés. D'après le mode de fonctionnement de l'Université, avant le début de chaque semestre, tous les enseignements doivent être répartis, c'est-à-dire être attribués aux enseignants.

Cette répartition des enseignements est supervisée par le **Vice-rectorat chargé des Etudes et de la Vie de l'Université (vice-rectorat EVU)**. En effet, il est chargé de visualiser et valider les répartitions des enseignements au niveau des départements, avant de les envoyer aux chefs de services pédagogique afin qu'ils élaborent les emplois du temps.

Présentation du vice-rectorat EVU

Il a en charge la coordination de la politique pédagogique, la formation initiale et continue non-doctorale de l'Université. Il veille à faciliter l'acte d'enseigner et une meilleure prise en charge des apprenants. Celle-ci passera aussi par l'animation de la vie étudiante. Il définit la politique de l'Université dans ses apports avec les collectivités locales et assure la participation à la vie de la cité.

Dans ses fonctions, le vice-rectorat EVU est composé de deux directions : la Direction des Etudes et des Réformes Universitaires (DERU) et la Direction de la Vie de l'Université (DIVU).

Dans notre contexte, la Direction qui nous intéresse est celle des Etudes et des Réformes Universitaires (DERU). C'est elle qui se charge de tout ce qui touche la pédagogie, plus particulièrement, la gestion des répartitions des enseignements, que nous allons aborder dans la section suivante.

1.2 La gestion des répartitions des enseignements à l'UASZ

La gestion de la répartition des enseignements requiert une bonne organisation et le respect à certaines règles. Selon le mode de fonctionnement de l'UASZ, pour chaque semestre, tout enseignant doit choisir des enseignements à dispenser. C'est ainsi que nous allons, d'abord, aborder les différents acteurs intervenant dans la gestion des répartitions des enseignements. Ensuite, nous parlerons des types d'enseignants et leur organisation. Ensuite, nous aborderons les services d'enseignements ainsi que la manière dont les groupes d'étudiants sont organisés. Enfin, nous décrivons la gestion des répartitions des enseignements.

1.2.1 Les acteurs

La répartition des enseignements fait appel à six principaux acteurs, que sont :

- Le **Vice-rectorat EVU (DERU)** : il définit les périodes des répartitions, consulte et valide les répartitions, puis les envoie aux chefs de service pédagogiques ;
- Les **UFR** : elles sont composées des entités suivantes :
    - Les **Départements**, ils renferment les entités suivantes :
        - Les **Chefs de départements** et les **Responsables de formations** : ils établissent la liste d'enseignements à choisir, assurent la définition des étapes des choix d'enseignements, règlent les conflits lors des choix et envoient les choix (répartitions) au Vice-rectorat EVU ;
        - Les **Enseignants** (permanents et vacataires) : ils choisissent des enseignements ;
    - Les **services pédagogiques** : ils élaborent les emplois du temps suite à la réception des répartitions envoyées par le Vice-rectorat EVU. Ils gèrent aussi les **classes** et les **étudiants**. Les effectifs des étudiants dans les classes impactent beaucoup sur le nombre d'enseignements.

1.2.2 Les enseignants

Les enseignants sont des acteurs du système éducatif. A l'UASZ, ils sont au nombre de 200 environs. Ils sont chargés de dispenser des enseignements. Dans le domaine universitaire, un enseignant a un grade. Les différents grades sont les suivants :

- Professeur Titulaire CAMES (PTC)

- Maître de Conférence CAMES (MCC)
- Maître-assistant CAMES (MAC)
- Assistant (A)

1.2.3 Les groupes d'étudiants

Chaque année académique, un ensemble d'étudiants s'inscrivent pour une formation bien précise ainsi qu'à un niveau. Par exemple, en 2018, il y a 100 étudiants inscrits au niveau 2 de la formation "Licence Management Informatisé des Organisations (MIO)".

A partir de cet ensemble d'étudiants, des groupes de CM, de TP et de TD sont formés pour chaque classe (L2 MIO, par exemple). Il est important de noter que, pour une classe donnée, le nombre de groupes de CM, de TD ou de TP créés dépend du nombre d'étudiants. Ce dernier a un impact sur la répartition, car agit sur le nombre d'heures d'enseignements à répartir.

1.2.4 Les enseignements

Le système LMD qui reconnaît trois diplômes et grades nationaux :

- Licence (L) : diplôme de niveau bac+3 ;
- Master (M) : diplôme de niveau bac+5 ;
- Doctorat (D) : diplôme de niveau bac+8.

Les parcours de formations du système LMD sont découpés en UE (Unité d'Enseignement). Une UE est composée d'EC (Elément Constitutif). Un EC est constitué de CM, de TD et/ou de TP. Pour une UE donnée, un nombre d'heures d'enseignement est fixé, pareil pour l'EC ainsi que les CM, TD et TP.

Dans la suite de ce document, un enseignement désignera un CM, un TD ou un TP.

1.2.5 Les services d'enseignement

Le nombre minimal d'heures d'enseignement annuel à effectuer par un enseignant permanent dépend de son grade (Professeur Titulaire, Maître de Conférence, Maître Assistant, Assistant) qui peut changer au courant d'un semestre. En effet, un nombre minimal d'heures d'enseignement semestriel est fixé pour chaque grade :

- Professeur Titulaire : 60 heures CM par semestre ;

- Professeur Assimilé : 60 heures CM par semestre ;
- Maître de Conférence Titulaire : 102 heures TD par semestre ;
- Maître Conférence Assimilé : 102 heures TD par semestre ;
- Assistant : 102 heures TD par semestre.

Il y a une équivalence entre les heures de CM, de TD et de TP : 5 heures CM correspondent à 8 heures 30 minutes TD, qui correspondent à 12 heures TP.

En d'autres termes :

- 1 heure CM correspond à 1,25 heures TD.
- 1 heure CM correspond à 2 heures TP.
- 1 heure TD correspond à 1,6 heures TP.

Ainsi, un Professeur ou un Maître de Conférence, peut effectuer des TD ou des TP puis utiliser les équivalences pour se ramener à ses heures statutaires en CM. De même un Maître Assistant ou un Assistant peut effectuer des CM ou des TP puis utiliser les équivalences pour se ramener à ses heures statutaires en TD.

Un enseignant permanent peut faire plus que son nombre d'heures de grade à condition que les autres enseignants permanents du département aient complété leurs heures statutaires du semestre.

L'enseignant permanent peut aussi effectuer des tâches administratives (Ex : la direction d'un département ou la direction du centre des ressources informatiques de l'Université) qui sont prises en compte sur le nombre d'heures d'enseignements choisis. Chaque tâche administrative correspond à un certain nombre d'heures de TD.

Contrairement aux enseignants permanents, les enseignants vacataires n'ont aucune charge statutaire. Autrement dit, il n'y pas un nombre minimal d'heures d'enseignement fixé pour eux.

1.2.6 Description de la gestion de la répartition des enseignements

La répartition des enseignements (ou le choix des enseignements) est semestrielle et s'effectue en huit principales activités, numérotées par ordre d'exécution :

Activité 1 : **ouverture de la répartition** par le Vice-recteur
Activité 2 : **ouverture de l'étape** par le chef de département ou le responsable de formation

Activité 3 : **choix d'enseignements** par l'enseignant
Activité 4 : **réglage des conflits éventuels** par le chef de département ou le responsable de formation
Activité 5 : **fermeture de l'étape** par le chef de département ou le responsable de formation
Activité 6 : **fermeture de la répartition** par le Vice-recteur
Activité 7 : **visualisation et validation des répartitions** par le Vice-recteur
Activité 8 : **visualisation des répartitions** pour élaboration des emplois du temps par le chef de service pédagogique

L'ensemble des activités ci-dessus représente la procédure de répartition de façon générale. Mais il est important de noter que les **choix d'enseignements** (*Activité 3*) s'effectuent, de façon idéale, en trois étapes :

Etape 1 : elle consiste pour l'enseignant permanent à choisir des enseignements conformément à ses obligations en heures de CM, TD ou TP par rapport à son grade. Les vacataires ne sont pas concernés par cette étape. Le chef de département ou le responsable de formation, après vérification et règlement des éventuels conflits ferme l'étape.

A cette étape, si on est au premier semestre, l'enseignant permanent n'est pas obligé de compléter ses heures de grade semestrielles. Par exemple, le Professeur DIATTA qui doit faire 60 heures semestre, soit 120 heures CM l'année, peut choisir d'en faire 40 au premier semestre et 80 au second semestre.

Etape 2 : elle consiste pour l'enseignant permanent à choisir des enseignements en plus s'il le souhaite dans le cas où il reste des enseignements. Cette étape ne concerne pas les vacataires non plus. Le chef de département ou le responsable de formation, après vérification et règlement des éventuels conflits ferme cette étape.

A cette étape, si on est au premier semestre, l'enseignant permanent peut prendre plus que ses heures de grade semestrielles. Par exemple, le Professeur DIATTA qui doit faire 60 heures CM par semestre, soit 120 heures CM l'année, peut en faire 110 heures au premier semestre et 10 au second semestre.

Etape 3 : elle est réservée aux vacataires qui peuvent à leur tour choisir des enseignements s'il en reste. Le chef de département après vérification et règlement des éventuels conflits ferme cette étape. Si tous les enseignements du département ne sont pas choisis, cette étape reste ouverte. Dans ce cas, n'importe quel enseignant peut à nouveau choisir des enseignements.

Quel que soit l'étape, le chef de département ou le responsable de formation a la possibilité d'attribuer des enseignements à des enseignants. Cela permet de pouvoir intégrer les choix de certains enseignants qui, pour une raison ou pour une autre, n'ont pas accès à l'application au moment des choix.

La figure ci-dessous représente le diagramme d'activités (avec un compartiment pour chaque acteur) de la procédure de répartition détaillée ci-dessus.

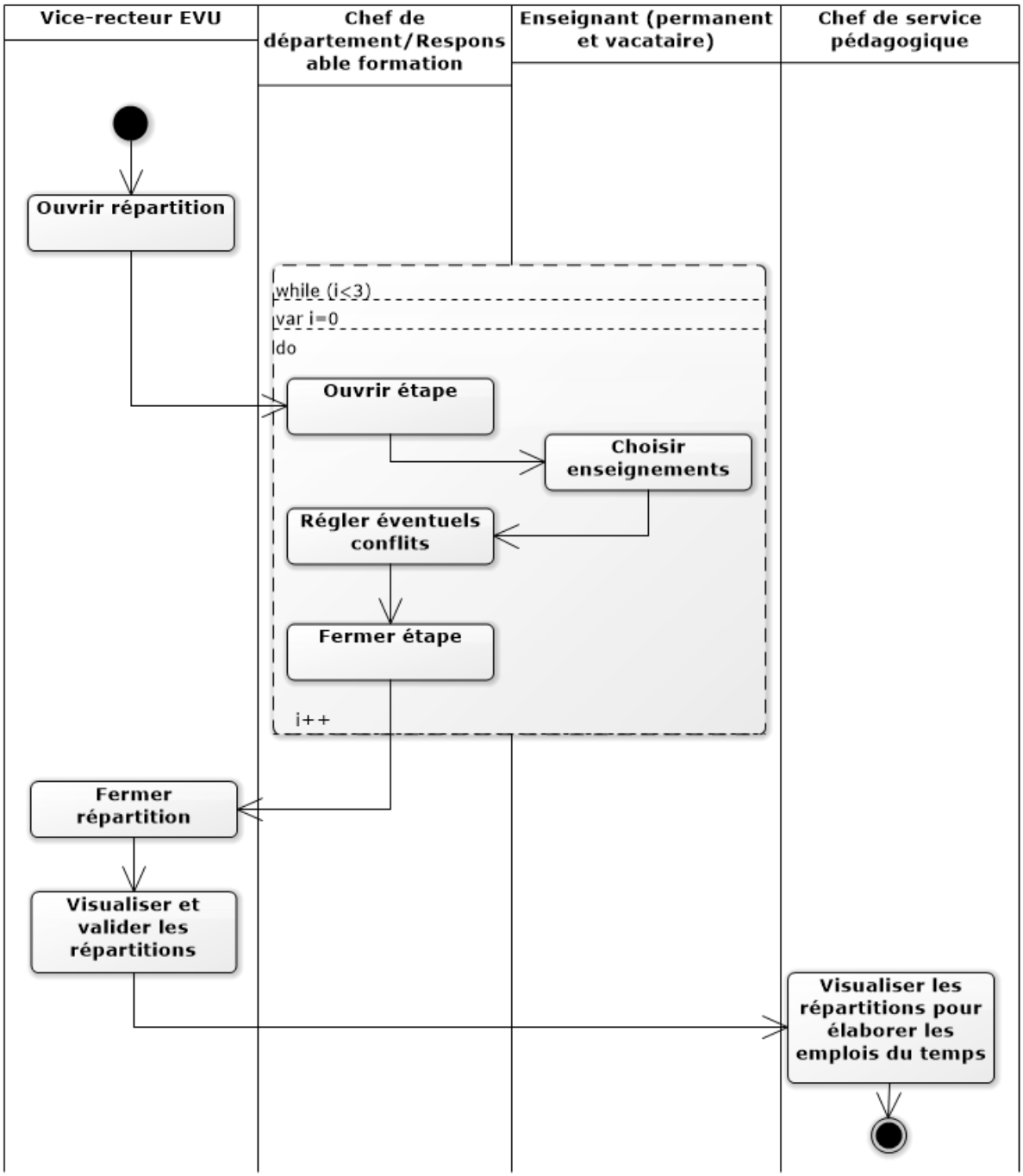


Figure 1 : Procédure de répartition des enseignements à l'UASZ

Cette procédure établie dans la figure ci-dessus représente le processus de répartition des enseignements de façon théorique, mais dans le cas pratique, de nombreuses difficultés sont rencontrées. Ainsi, dans la partie suivante, nous allons décrire les problèmes liés dans la gestion des répartitions des enseignements à l'UASZ.

1.2.7 Problèmes liés à la gestion des répartitions des enseignements à l'UASZ

A l'UASZ on note l'utilisation des fichiers Excel dans lesquels sont répertoriés tous les enseignements. Ces fichiers sont envoyés à tous les enseignants afin que ces derniers choisissent des enseignements. Les conséquences majeures de l'utilisation de ces fichiers sont l'absence d'une certaine rapidité et d'automatisation de certaines tâches dans le processus de répartition. Ce qui est la cause de beaucoup d'autres problèmes dans la répartition des enseignements, comme :

- Le **manque de commodité** : cela est causé par le seul fait que ces fichiers ne sont pas partagés en temps réel (inaccessibles à partir du web pour les enseignants). Autrement dit, le chef de département envoie ces fichiers aux enseignants. Et chaque enseignant choisit des enseignements et renvoie, à nouveau, le fichier au chef de département.
- La **difficulté de générer le récapitulatif des répartitions** : à la réception des fichiers venant de chaque enseignant contenant leurs choix, c'est au chef de département de réunir tout ces fichiers en un seul tout en prenant en compte les choix de chaque enseignant, dans le but d'établir le fichier récapitulatif des répartitions.
- La **non-transparence de la répartition des enseignements** et les **conflits** : étant donné l'absence de la visualisation de la situation des répartitions lors des choix et la non-transparence de la répartition des enseignements avec ce fichier Excel, il est fort probable que deux ou plusieurs enseignants choisissent le même enseignement. Cela entraîne un conflit. Pour régler ce dernier une réunion doit être tenue dans le but d'accepter le choix pour un enseignant et le refuser pour les autres. Parfois, les conflits ne sont pas détectés et le fichier de la répartition est envoyé dans cet état au chef de service pédagogique pour l'élaboration des emplois du temps, et c'est là que les problèmes apparaîtront et il sera difficile, voire trop tard, pour les résoudre.
- Le **dépassement des charges horaires lors de la première étape** : avec le système actuel, il est très difficile, voire impossible, d'éviter que les enseignants permanents choisissent plus que leurs charges horaires. Et comme conséquence, certains enseignants n'auront pas suffisamment d'heures pour atteindre leurs charges horaires, alors que d'autres en ont jusqu'en ajouter des heures supplémentaires.
- Le **manque d'autonomie** : un enseignant qui veut annuler son choix sur un enseignement doit le signaler au chef de département. C'est à ce dernier d'annuler le

choix de l'enseignant et d'envoyer un e-mail informant aux autres enseignants de la disponibilité d'un nouvel enseignement.

Ainsi, avec la hausse du nombre de formations ouvertes ainsi que le nombre d'étudiants orientés dans les départements, les enseignements deviennent de plus en plus nombreux. Pour s'adapter à cette croissance, l'UASZ souhaite mettre en place une solution répondant aux attentes pour la gestion des répartitions des enseignements.

Ainsi, dans la partie qui suit, nous aborderons la problématique de notre sujet en proposant une solution et ses objectifs.

1.3 Problématique du sujet

Tout au début de son fonctionnement (voire les trois premières années), l'UASZ ne ressentait pas les problèmes liés à la gestion des répartitions des enseignements avec l'utilisation des fichiers Excel. Cela parce que le système était géré avec de petits effectifs et un nombre réduit de formations. Mais avec son évolution, les problèmes commencent sérieusement à se ressentir avec le système actuel qui n'est pas pratique et ne permet pas l'automatisation de certaines tâches.

Dans cette partie, nous proposons une solution aux problèmes rencontrés dans la gestion des répartitions des enseignements, puis nous décrivons les objectifs spécifiques du sujet de notre mémoire.

1.3.1 Solution proposée

Face aux problèmes et difficultés rencontrés par l'UASZ pendant la répartition des enseignements, nous proposons une solution informatique (une application web) adaptée aux besoins spécifiques de la gestion des répartitions des enseignements. Pour ce faire, nous avons fait un stage au sein du CRI (Centre des Ressources Informatiques) de l'UASZ, dans le but de renforcer la supervision du développement de l'application. Elle permettra, entre autres, de :

Gérer les évènements : avant que les enseignants ne puissent choisir des enseignements, le Vice-recteur doit, d'abord, définir les dates d'ouverture et de fermeture des répartitions. Pendant la période de répartition, le Chef de département ou le Responsable de formation définit les trois étapes de la répartition en fixant les dates d'ouverture et de fermeture.

Choisir des enseignements : dans chaque étape d'une répartition donnée, tout enseignant doit choisir des enseignements (CM, TD ou TP) pour le semestre tout en respectant les règles définies à cet effet.

Régler les conflits : pendant le choix des enseignements, il peut arriver que deux ou plusieurs enseignants aient choisi le même enseignement. Dans ce cas, il y a un conflit et c'est au Chef de département ou le Responsable de formation de voir comment affecter l'enseignement à un seul enseignant qu'il choisira et éliminer les choix des autres enseignants sur cet enseignement.

Récapituler les heures des enseignants : pendant et après le choix des enseignements, l'affichage du total des heures pour chaque enseignant doit être fait en donnant les heures supplémentaires (au-delà du service dû) et en signalant les enseignants qui n'ont pas encore assez d'heures pour effectuer le service qu'ils doivent.

Afficher les répartitions pour un semestre donné : à la fin de chaque répartition d'un semestre, répertorier tous les choix des enseignants sur les enseignements avec toutes les informations nécessaires.

Faciliter l'établissement des emplois du temps : l'intérêt majeur de cette solution est, non seulement, de permettre la répartition des enseignements, mais aussi de pouvoir faciliter, pour chaque semestre, l'établissement des emplois du temps à partir des choix des enseignants sur les enseignements.

1.3.2 Objectifs spécifiques du sujet

Les principaux objectifs de notre système informatique pour la gestion des répartitions des enseignements sont les suivants :

- Pouvoir gérer les enseignements et les enseignants;
- Permettre à l'enseignant de choisir des CM, TD ou TP et lui donner la possibilité de savoir s'il en reste à prendre dans chacun des cours ;
- Permettre de signaler et de gérer les conflits entre les choix des enseignants ;
- Informer l'enseignant sur le total de ses heures d'enseignement (en équivalent CM ou équivalent TD selon son grade) ;
- Permettre la visualisation des services d'un enseignant ;
- Permettre la visualisation des répartitions pour un EC donné ;

- Afficher les CM, TD et TP qui n'ont pas encore été choisis par un enseignant ;
- Afficher le total d'heures pour chaque enseignant, en donnant les heures supplémentaires (au-delà du service dû) et en signalant les enseignants qui n'ont pas encore assez d'heures pour effectuer le service qu'ils doivent ;
- Ouvrir et fermer une période de répartition ;
- Ouvrir et fermer une étape dans une répartition ;
- Imprimer le détail de la répartition des enseignements pour chaque EC ;
- Imprimer le détail des services pour chaque enseignant ;

Afin de pouvoir atteindre les objectifs énoncés ci-dessous, nous avons besoin de travailler en collaboration avec le client (ici, le Vice-rectorat), c'est la raison pour laquelle nous avons opté pour une méthodologie de développement qui implique ce dernier. Pour ce faire, nous utilisons l'une des méthodes agiles à savoir le **Scrum** qui sera détaillé dans le chapitre qui suit.

CHAPITRE 2 : PROCESSUS DE DEVELOPPEMENT DE L'APPLICATION

Un processus de développement est un ensemble d'activités ou d'actions structurées dans le but d'atteindre l'objectif d'un projet de système informatique. La réalisation d'un projet informatique nécessite une parfaite collaboration. La planification, l'exécution, ainsi que la gestion d'un projet restent des tâches complexes. Cela est dû au fait qu'elles soient touchées par des contraintes de délai, de coût, de qualité, de ressources et de capacité organisationnelle. Il est donc d'une importance capitale d'opter et de pratiquer une méthodologie de gestion de projet adaptée : il s'agira, dans notre cas, des méthodes Agiles [8].

Une méthode Agile est une approche itérative et collaborative, capable de prendre en compte les besoins initiaux du client et ceux liés aux évolutions. Les méthodes pouvant être qualifiées d'agiles sont : Crystal clear, Scrum, Extreme programming (XP), etc.

Pour la suite de ce chapitre, nous nous intéresserons à la méthodologie Scrum qui est une méthodologie de gestion de projet appartenant aux méthodes agiles [8]. Ensuite, nous l'adapterons à notre contexte.

2.1 La méthodologie Scrum

Scrum est une méthode agile consacrée à la gestion de projets. C'est un cadre de travail permettant de répondre à des problèmes complexes et changeants, tout en livrant de manière productive et créative des produits de la plus grande valeur possible. Elle utilise une approche itérative et incrémentale pour optimiser la prédictibilité et pour contrôler les risques.

Dans cette section, nous allons décrire la méthodologie Scrum et justifier notre choix sur elle.

2.1.1 Description de la méthodologie Scrum

Le cadre du Scrum consiste en la définition des rôles du projet, des sprints, du product backlog, des user story, du sprint planning meeting, de la mêlée.

Les Rôles

Le Scrum Master : il s'assure au respect des principes et valeurs du Scrum, facilite la communication au sein de l'équipe et cherche à rendre meilleures la productivité et la compétence de son équipe.

L'équipe : elle est constituée d'architectes, de développeurs et de testeurs.

Le Product Owner : il est l'expert métier et joue le rôle du client. Il définit les spécifications fonctionnelles, établit les priorités des fonctionnalités à développer et valide les fonctionnalités développées.

Les sprints

Le cycle de vie Scrum est rythmé par des itérations de quelques semaines, les sprints. Un sprint est une période d'un mois au maximum, au bout de laquelle l'équipe délivre un incrément du produit, potentiellement livrable.

Le product backlog

Le product backlog est un document qui renferme les exigences initiales (fonctionnalités). Il est établi et organisé avec le client.

User Story

Une User Story représente le nom que portent les fonctionnalités décrites. Elle contient généralement les informations suivantes :

- **ID** – un identifiant unique
- **Nom** – un nom court, descriptif de la fonctionnalité attendue par le client
- **Importance** – un entier qui fixe la priorité des Story.
- **Estimation** – la quantité de travail nécessaire pour développer, tester et valider cette fonctionnalité
- **Demo** – un simple test
- **Note** – toute autre information

Le sprint planning meeting

Le sprint planning meeting est une réunion de planification organisée avant chaque sprint. L'objet de cette réunion est de sélectionner, dans le product backlog, les exigences

(fonctionnalités) les plus prioritaires pour le client. Elles seront développées, testées et livrées au client à la fin du sprint.

La mêlée

Pendant le déroulement du spint, il est organisé, chaque jour, une réunion d'avancement (environ 15 min) entre les membres de l'équipe dans le but de s'assurer que les objectifs du sprint seront respectés, c'est le **Scrum** ou **mêlée**. Après chaque réunion Scrum, le Scrum Master dresse un graphique appelé **sprint burndown chart**. Ce dernier donne une vision de ce qui a été fait et du rythme de travail de l'équipe.

La figure suivante représente le fonctionnement du Scrum.

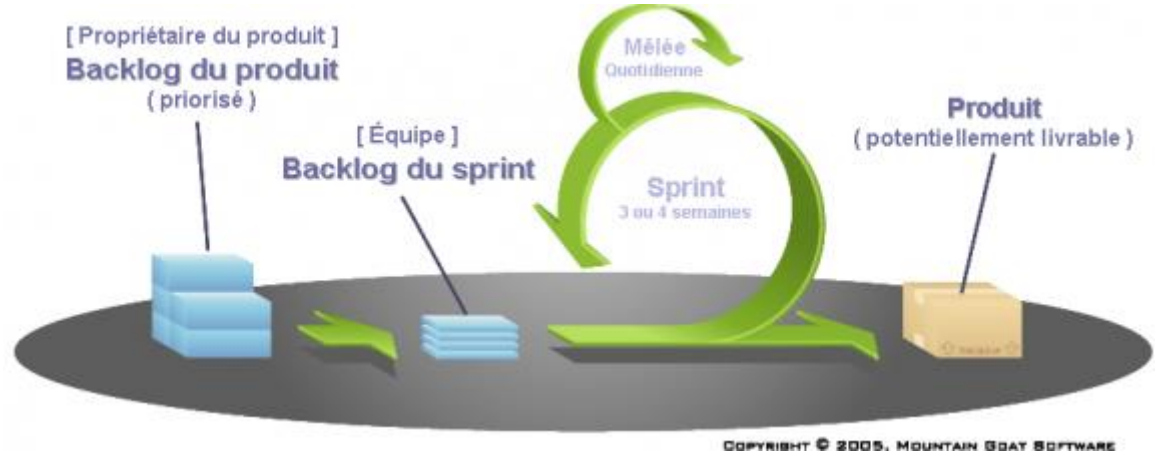


Figure 2 : Fonctionnement du Scrum

2.1.2 Justification du choix sur Scrum

Le choix que nous avons fait sur Scrum repose sur de nombreux avantages par rapport à notre contexte. En effet, la méthodologie Scrum, une méthode agile, exige l'implication du client dans le processus de développement du produit afin d'établir une communication entre l'équipe de développement et le client, et d'accueillir positivement les éventuelles demandes de changement du product backlog. Cela pour, non seulement, éviter qu'il ait des divergences entre les deux parties (le client et l'équipe de développement), mais aussi de livrer un produit potentiel correspondant aux attentes du client.

L'autre point important sur le choix de Scrum est le développement propre et bien organisé du produit. En effet, le Product Owner (le client) ayant dressé la liste des exigences ou fonctionnalité ainsi que leurs priorités, le développement se fait fonctionnalité par

fonctionnalité selon le niveau de priorité, jusqu'à la réalisation du produit final souhaité par le client.

Jusque-là nous avons étudié que la partie théorique de la méthodologie Scrum. Il serait important de l'adapter dans notre contexte. Cette adaptation sera l'objet de la section suivante.

2.2 Adaptation de la méthodologie Scrum à notre contexte

Dans cette partie, nous mettons en évidence la façon dont nous avons adapté la méthodologie Scrum à notre projet, en commençant par la manière à laquelle l'équipe s'est organisée, ensuite décrire le déroulement du développement.

Nous tenons à signaler que, dans ce projet, la méthodologie Scrum n'a pas été utilisée dans son intégralité, mais nous avons seulement fait appel à ses pratiques.

2.2.1 Organisation de l'équipe

Le tableau ci-dessous représente l'équipe chargée de la réalisation du projet.

Tableau 1 : Equipe du projet

Personnes	**Rôles**	**Fonctions**
Dr. Alassane DIEDHIOU	Product Owner	Vice-recteur EVU
Dr. Marie NDIAYE	Scrum Master	Enseignante-chercheuse au Département d'Informatique de l'UASZ
Dr. Ibrahima DIOP		Enseignant-chercheur au Département d'Informatique de l'UASZ
M. Alousseynou FALL		Chef de division Etudes et Développement au CRI
M. Henry DIALLO	Scrum Team (équipe)	Etudiant en Master 2 Génie loigiciel au Département d'Informatique de l'UASZ

2.2.2 Déroulement du développement

La méthodologie Scrum a été appliquée à notre guise vu que le Product Owner (vice-recteur) manque de disponibilité afin de bien visualiser le produit et faire ses feed-backs. Néanmoins, nous avons eu à faire des séances séance de travail avec lui, dans le but de bien expliquer le processus de la répartition des enseignements, de dégager l'ensemble des fonctionnalités (**product backlog**) et acteurs du système de répartition des enseignements, mais aussi de faire des feed-backs après lui avoir présenté le produit.

Cependant, le travail a été beaucoup plus intense avec les Scrum Masters. Ils supervisaient et contrôlaient parfaitement le développement du produit. En effet, des rencontres (**mêlées**) ont été organisées, dès que possible, dans leurs bureaux. Et pour chaque rencontre, on vérifie si les fonctionnalités (**sprint**) qu'on s'était fixées précédemment, ont été développées, sinon on cherche des solutions aux problèmes empêchant l'atteinte de ces objectifs, puis on se fixe de nouveaux objectifs. Chaque semaine, nous faisons un compte-rendu aux Scrum Masters pour évaluer l'état d'avancement avant de continuer le développement des fonctionnalités suivantes.

Pendant les sprints, nous avons dressé notre **sprint burndown chart** (graphique d'avancement) afin de donner un aperçu sur l'évolution de la quantité de travail restante par rapport au temps sur une période donnée.

La figure suivante représente notre burndown chart. Le travail restant se situe sur l'axe vertical, et le temps est sur l'axe horizontal.

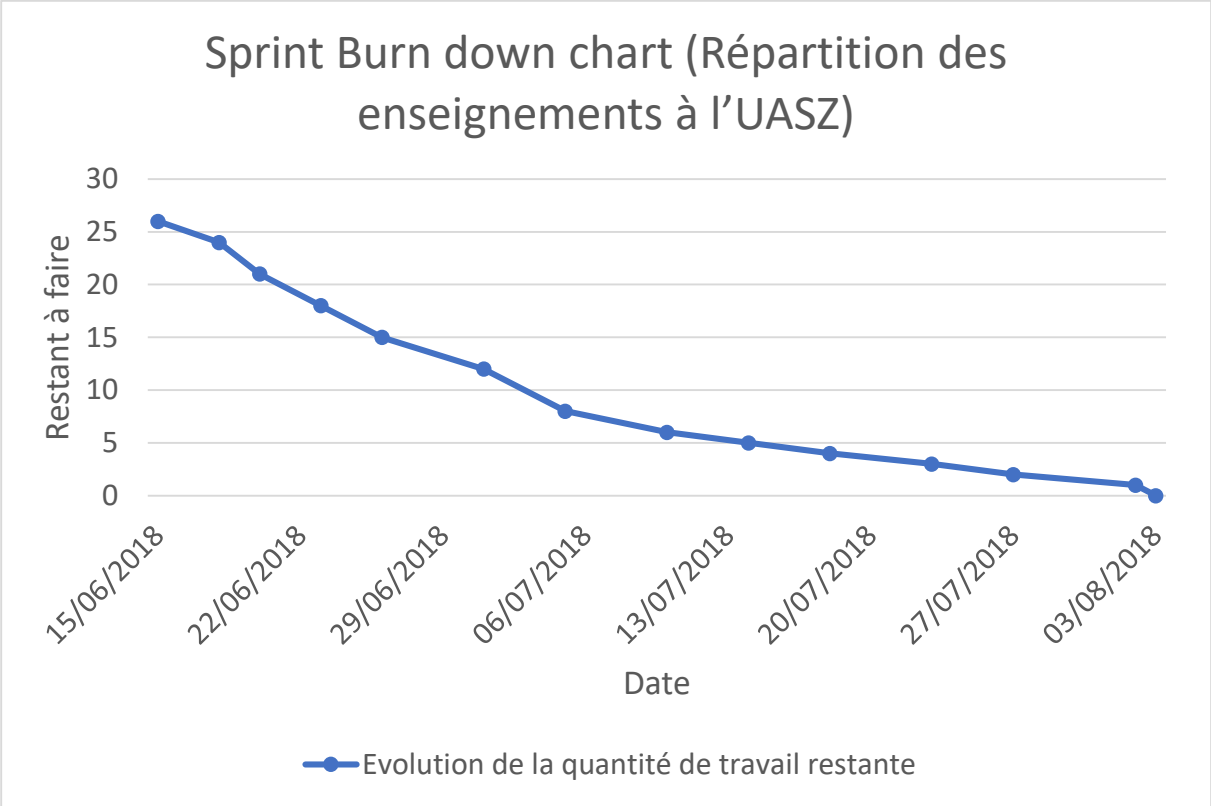


Figure 3 : Sprint burn down chart

Pour chaque date du burndown chart, des sprints sélectionnés depuis le product backlog sont implémentés.

Les méthodes agiles impliquent au maximum le demandeur (client) et permettent une grande réactivité à ses demandes. Scrum, étant une de ces méthodes, propose le développement par étapes courtes, d'inspecter l'efficacité des pratiques mises en œuvres et le produit qui en résulte et adapter les objectifs du produit.

Dans le chapitre qui suit, nous aborderons la spécification et l'analyse des besoins fonctionnels de notre système.

CHAPITRE 3 : SPECIFICATION ET ANALYSE DES BESOINS FONCTIONNELS

Suite à la définition du processus de développement notre application, nous entamons, dans ce chapitre, la spécification (identification des acteurs et fonctionnalités) et l'analyse (description des fonctionnalités) des besoins fonctionnels.
Nous y parlerons un peu de la modélisation objet en faisant appel à UML [2] afin de traduire, en diagrammes fonctionnels, les spécifications des besoins fonctionnels.

3.1 Spécification des besoins fonctionnels

Dans cette section, nous allons commencer par identifier les acteurs, ensuite les fonctionnalités et terminer par le diagramme de cas d'utilisation.

3.1.1 Identification des acteurs du système

Un acteur est une entité qui agit sur le système, comme une personne humaine ou un robot. Une même personne (ou robot) peut jouer le rôle du plusieurs acteurs dans un système, c'est pourquoi les acteurs doivent surtout être décrits par leur rôle. Ce rôle décrit les besoins et les capacités de l'acteur. L'activité du système a pour but de satisfaire les besoins de l'acteur. Ce dernier est représenté par un pictogramme humanoïde sous-titré par le nom de l'acteur [3].

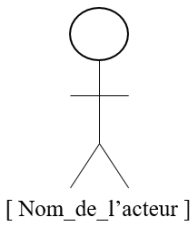


Figure 4 : Représentation d'un acteur

Pour connaître les acteurs de notre système, nous nous sommes basés sur le cahier des charges, obtenu à partir d'une étude préliminaire au Vice-rectorat et aux départements. Dans le tableau, ci-dessous, sont répertoriés tous les acteurs ainsi que leurs actions dans la gestion des répartitions des enseignements à l'UASZ.

Tableau 2 : Identification des acteurs

Acteurs	**Actions**

Vice Recteur EVU	Définir les périodes de répartition (date de début et date de fin), choisir des enseignements en tant que enseignant et aussi valider les répartitions des départements avant qu'elles ne soient traitées par les services pédagogiques pour l'élaboration des emplois du temps.
Chef de département	Définir les étapes de la répartition (date de début et date de fin), choisir des enseignements en tant que enseignant et régler les éventuels conflits.
Responsable de formation	
Enseignant	choisir des enseignements.
Responsable pédagogique	Consulter les répartitions afin d'élaborer les emplois du temps.
Administrateur	Administrer le système

Un acteur est toujours lié à une ou des action(s) du système. Ces actions sont souvent appelées des **fonctionnalités**. Nous allons, dans la section suivante, identifier l'ensemble des fonctionnalités de notre système.

3.1.2 Identification des fonctionnalités du système

Les besoins fonctionnels représentent les actions que doit exécuter un système : les fonctionnalités du système. Ce dernier sera opérationnel que lorsqu'il satisfait les besoins.

Dans le tableau, ci-dessous, sont répertoriées toutes les fonctionnalités identifiées, accompagnées de leurs acteurs.

Tableau 3 : Identification des fonctionnalités

Fonctionnalités	**Acteurs**
1. S'authentifier	Vice Recteur, Chef de département, Responsable de formation, Enseignant, Responsable pédagogique, Administrateur
2. Ouvrir une répartition	Vice Recteur EVU

3. fermer une répartition	
4. Ouvrir une étape dans la répartition	Chef de département, Responsable de formation
5. fermer une étape dans la répartition	
6. Choisir des enseignements	Enseignant
7. Régler les conflits	Chef de département, Responsable de formation
8. Visualiser les enseignements non choisis	Vice Recteur, Chef de département, Responsable de formation, Enseignant
9. Visualiser le récapitulatif horaire des enseignants	Vice Recteur, Chef de département, Responsable de formation
10. Attribuer des enseignements à des enseignants n'ayant pas encore assez d'heures	Chef de département, Responsable de formation
11. Visualiser la répartition pour un EC	Vice Recteur, Chef de département, Responsable de formation, Enseignant, Responsable pédagogique
12. Imprimer le détail des services pour un enseignant	Vice Recteur, Chef de département, Responsable de formation, Enseignant, Responsable pédagogique
13. Imprimer le détail de la répartition pour un EC	Vice Recteur, Chef de département, Responsable de formation, Enseignant, Responsable pédagogique
14. Gérer les enseignants (ajouter, modifier et supprimer un enseignant)	Responsable pédagogique, Administrateur
15. Gérer les enseignements (ajouter, modifier et supprimer un enseignement)	Responsable pédagogique, Administrateur
16. Gérer les EC (ajouter, modifier et supprimer un EC)	Responsable pédagogique, Administrateur

17. **Gérer les UE (ajouter, modifier et supprimer une UE)**	Responsable pédagogique, Administrateur
18. **Gérer les formations (ajouter, modifier et supprimer une formation)**	Responsable pédagogique, Administrateur
19. **Gérer les classes (ajouter, modifier et supprimer une classe)**	Responsable pédagogique, Administrateur
20. **Gérer les niveaux (ajouter, modifier et supprimer un niveau)**	Responsable pédagogique, Administrateur
21. **Gérer les étudiants (ajouter, modifier et supprimer un étudiant)**	Responsable pédagogique, Administrateur
22. **Gérer les promotions (ajouter, modifier et supprimer une promotion)**	Responsable pédagogique, Administrateur
23. **Gérer les semestres (ajouter, modifier et supprimer un semestre)**	Responsable pédagogique, Administrateur
24. **Gérer les départements (ajouter, modifier et supprimer un département)**	Responsable pédagogique, Administrateur
25. **Gérer les UFR (ajouter, modifier et supprimer une UFR)**	Responsable pédagogique, Administrateur
26. **Gérer les grades (ajouter, modifier et supprimer un grade)**	Responsable pédagogique, Administrateur, Vice-recteur
27. **Gérer les tâches administratives (ajouter, modifier et supprimer une tâche administrative)**	Responsable pédagogique, Administrateur, Vice-recteur

Les liaisons entre acteurs et fonctionnalités sont représentées, en UML, par des diagrammes de cas d'utilisation. Nous allons, dans la section qui suit, élaborer les diagrammes de cas d'utilisation de notre système.

3.1.3 Diagrammes de cas d'utilisation

Un diagramme de cas d'utilisation est un diagramme UML qui fournit une représentation graphique des exigences du système, il aide à identifier la façon dont les utilisateurs interagissent avec ce dernier.
Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur et un système. Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs, ils interagissent avec les cas d'utilisation [5].

Pour ce travail, nous présenterons les diagrammes de cas d'utilisation par groupe de fonctionnalités : la gestion des périodes de répartitions, la gestion des choix d'enseignements, les affichages et impressions et l'administration.

- **Le diagramme de cas d'utilisation de la gestion des périodes de répartition**

Pour gérer les périodes de répartition, c'est au Vice-recteur d'ouvrir et/ou de fermer les périodes de répartitions (en définissant les dates de début et de fin). Après l'ouverture d'une répartition, le chef de département ou le responsable de formation ouvre et/ou ferme les périodes d'étape de la répartition (en définissant les dates de début et de fin) afin que les choix d'enseignements puissent être effectués.

Il est important de noter que l'exécution de tout cas d'utilisation nécessite, au préalable, une authentification.

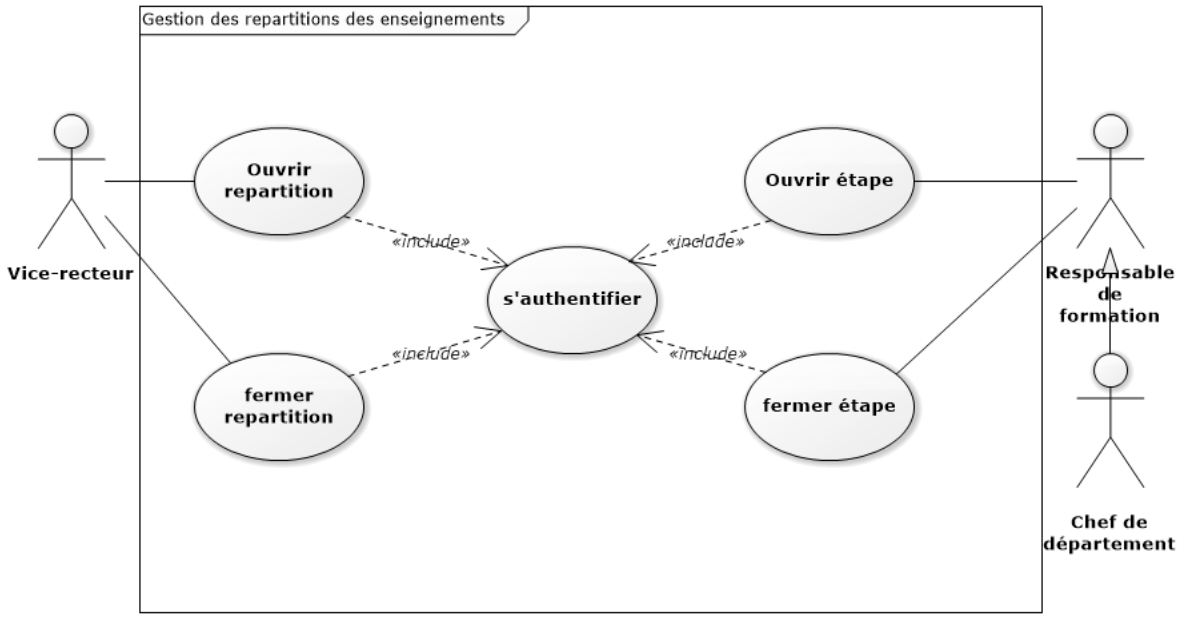


Figure 5 : Diagramme de cas d'utilisation de la gestion des périodes de répartition

- **Le diagramme de cas d'utilisation de la gestion des choix d'enseignements**

Les enseignements sont choisis par les enseignants (permanents et vacataires). Après les choix des enseignements, le chef de département et/ou le responsable de formation règle les éventuels conflits. Ils ont la possibilité de vérifier la prise totale des enseignements et, lorsqu'il en reste, d'en attribuer aux enseignants.

Il est important de noter que l'exécution de tout cas d'utilisation nécessite, au préalable, une authentification.

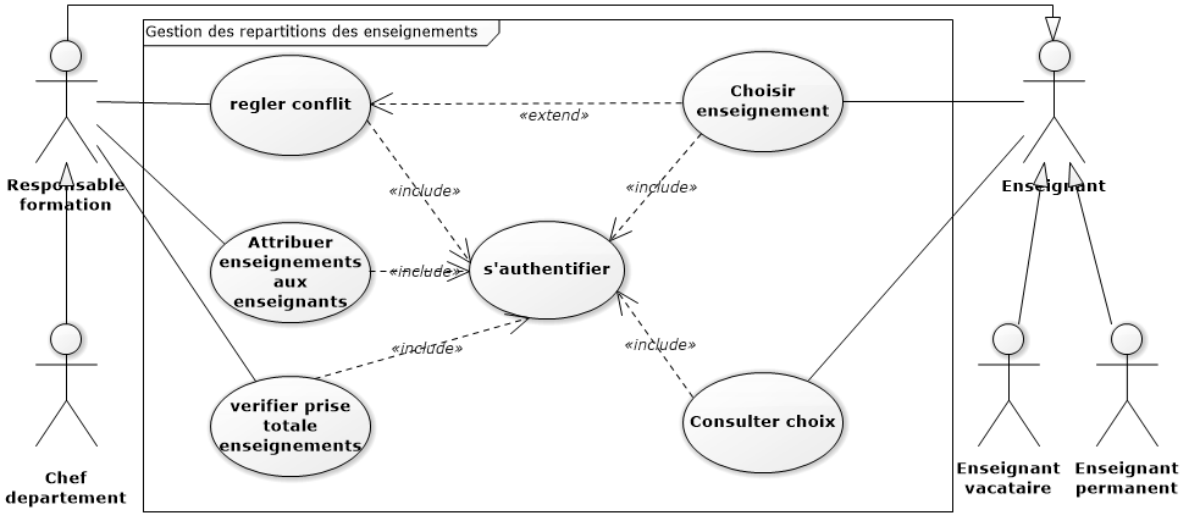


Figure 6 : Diagramme de cas d'utilisation de la gestion des choix d'enseignements

- **Le diagramme de cas d'utilisation pour l'affichage et l'impression de données**

Les enseignants peuvent consulter et imprimer leurs choix d'enseignements. Le vice-recteur et le chef de service pédagogique peuvent visualiser et imprimer les répartitions des départements.

Il est important de noter que l'exécution de tout cas d'utilisation nécessite, au préalable, une authentification.

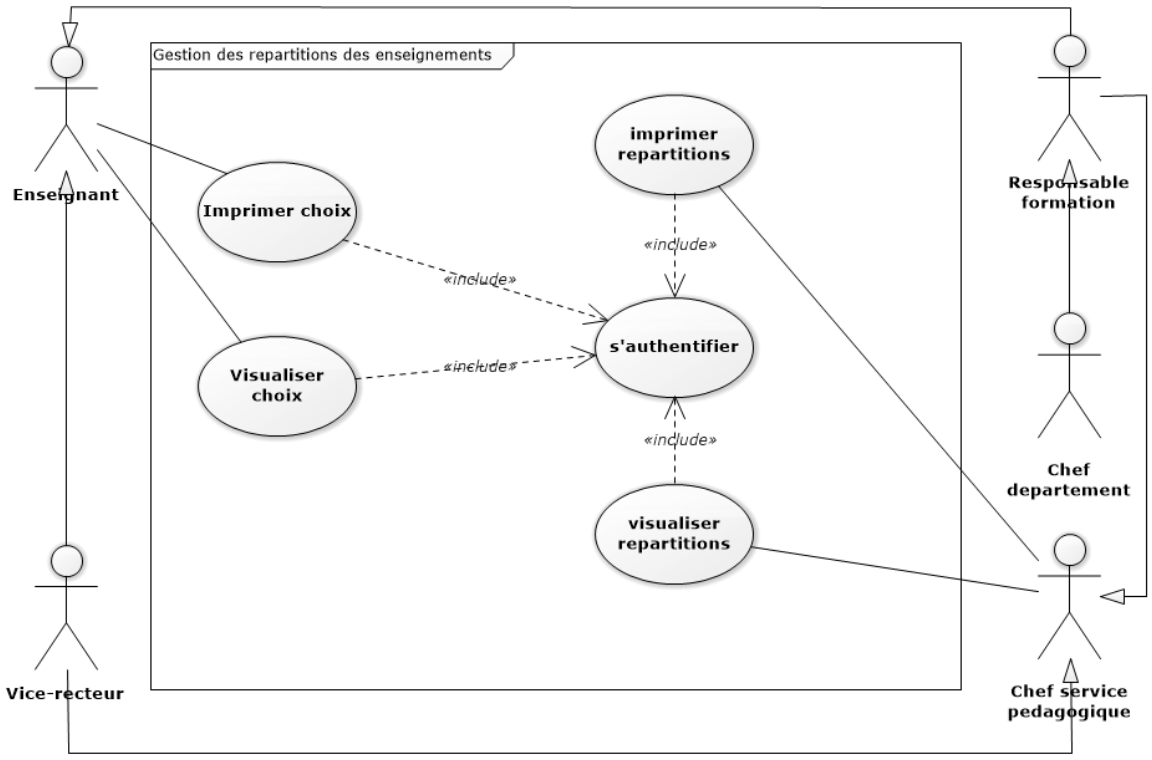


Figure 7 : Diagramme de cas d'utilisation pour l'affichage et l'impression de données

- **Le diagramme de cas d'utilisation pour l'administration du système**

Le chef de service pédagogique assure la gestion (ajout, modification et suppression) des maquettes (formations, classes, niveaux, UE, EC, enseignements, promotions, étudiants). L'administrateur , en plus des tâches du chef de service pédagogique, assure la gestion des UFR, des départements, des enseignants, des grades, des tâches administratives et des semestres.

Une authentification est nécessaire avant toute opération sur le système.

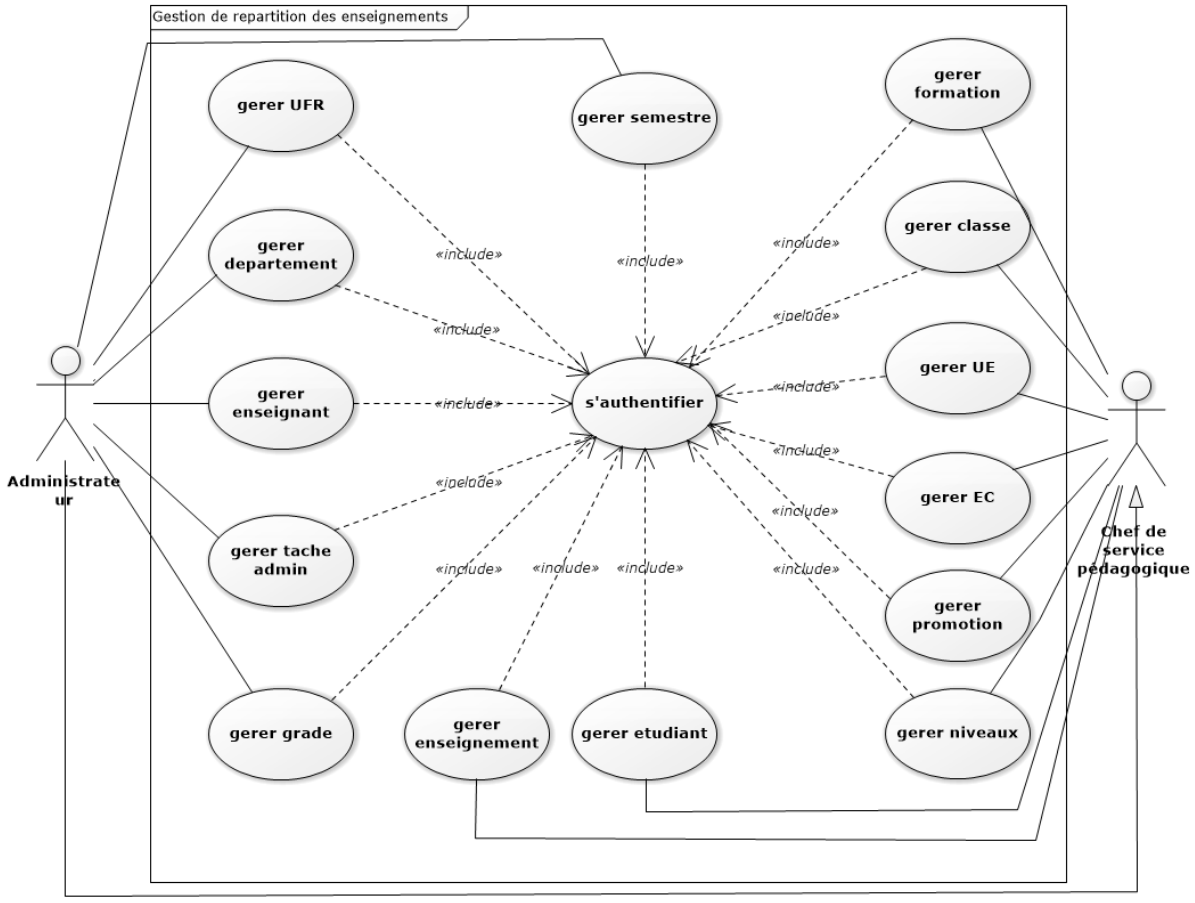


Figure 8 : Diagramme de cas d'utilisation pour l'administration

Les besoins spécifiés nécessitent souvent d'être examinés en leurs différentes parties. C'est pour cela, dans la section qui va suivre, nous analyserons quelques besoins fonctionnels de notre système.

3.2 Analyse des besoins fonctionnels du système

L'analyse des besoins fonctionnels est une démarche qui consiste à décrire les fonctionnalités du système. Dans cette partie, nous analysons les besoins fonctionnels qui semblent les plus complexes. Pour ce faire, nous commenerons par analyser le besoin d'authentification, ensuite, analyser celui de la gestion (ouverture et fermeture) des étapes et enfin analyser le besoin de choix d'enseignements.

3.2.1 Analyse de l'authentification

Dans cette section, nous décrivons d'abord le cas d'utilisation « s'authentifier », ensuite élaborer ses diagrammes d'activité et de séquence.

Description du cas d'utilisation « s'authentifier »

Le tableau ci-dessous permet de décrire le cas d'utilisation « s'authentifier ».

Tableau 4 : Description de cas d'utilisation « s'authentifier »

Description de cas d'utilisation « s'authentifier »	
Titre	S'authentifier
Résumé	Permet de vérifier l'accès au système
Acteurs	Vice-recteur, Chef de département, Responsable de formation, Enseignant permanent, Chef de service pédagogique, Administrateur
Pré condition	Avoir un compte d'utilisateur
Scénario nominal	• L'utilisateur saisit son pseudo et son mot de passe ; • Le système vérifie les informations saisies ; • Le système recupère le profil de l'utilisateur.
Post condition	Affichage de la page d'accueil correspondant au profil de l'utilisateur
Exception	Saisie du pseudo ou du mot de passe incorrect

Diagramme d'activités du cas d'utilisation « s'authentifier »

Un diagramme d'activité permet de modéliser un processus interactif, global ou partiel pour un système donné (logiciel, système d'information). Une activité est l'exécution d'une partie du cas d'utilisation. Elle est représentée par un rectangle aux bords arrondis[6].

Le cas d'utilisation « s'authentifier » regroupe plusieurs activités. Nous commençons par saisir un pseudo et un mot de passe. Après l'exécution de cette activité, le système passe à une vérification. Si les données saisies sont incorrectes, un message d'erreur s'affiche et l'activité

« saisir un pseudo et un mot de passe » doit être reprise. Par ailleurs, si les données saisies sont correctes, le sytème donne accès à la page d'accueil.

L'ensemble de ces enchaînements est décrit dans le diagramme d'activité ci-dessous.

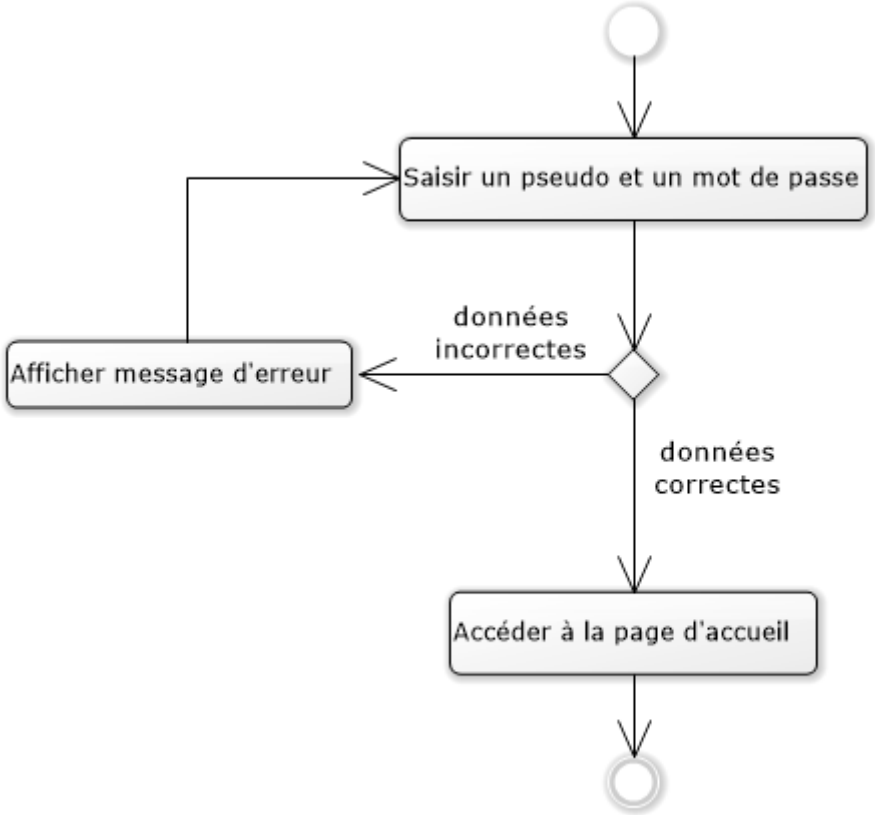


Figure 9 : Diagramme d'activité du cas d'utilisation "s'authentifier"

Diagramme de séquence du cas d'utilisation « s'authentifier »

Le diagramme de séquence est une représentation graphique d'interactions entre les acteurs et le système selon un ordre chronologique dans la formulation UML. Il permet de montrer les interactions d'objets dans le cadre d'un scénario d'un diagramme de cas d'utilisation [7].

Dans notre cas, nous représentons le scénario nominal. L'utilisateur saisit son pseudo et son mot de passe. Le système procède à une vérification. Après cela, le système affiche la page d'accueil.

La figure ci-dessous représente le diagramme de séquence illustrant le scénario nominal de l'authentification.

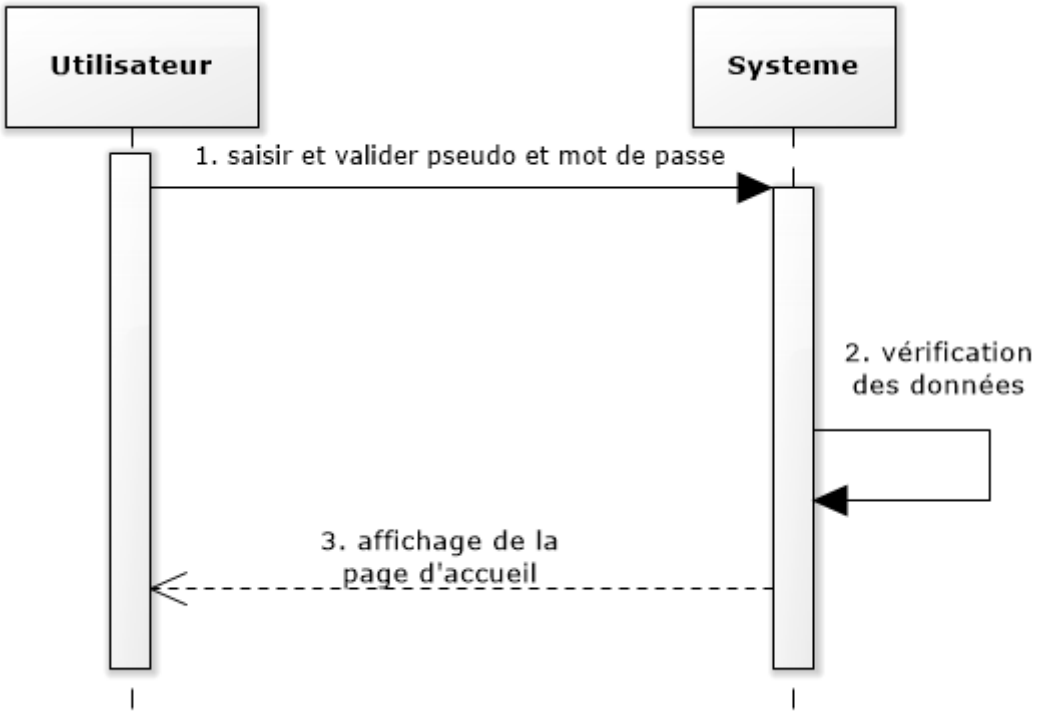


Figure 10 : Diagramme de sequence du cas d'utilisation "s'authentifier"

Suite à l'analyse de l'authentification, nous allons, dans la section qui suit, décrire le besoin fonctionnel de l'ouverture d'étape.

3.2.2 Analyse de la gestion des étapes

Dans cette partie, nous décrivons textuellement d'abord le cas d'utilisation « ouvrir des étapes », ensuite son diagramme d'activité et enfin le diagramme de séquence.

Description du cas d'utilisation « ouvrir étape »

Le tableau ci-dessous permet de décrire le cas d'utilisation « ouvrir étape ».

Tableau 5 : Description de cas d'utilisation « ouvrir étape »

Description de cas d'utilisation « ouvrir étape »	
Titre	ouvrir une étape
Résumé	Permet d'ouvrir des étapes dans lesquelles les choix d'enseignements se font
Acteurs	Chef de département et Responsable de formation

Pré condition	Authentification, ouverture d'une période de répartition
Scénario nominal	• Le système affiche la liste des étapes ; • L'utilisateur saisit et valide l'étape en donnant les dates de début et de fin ; • Le système enregistre ; • Message de confirmation de l'opération effectuée.
Post condition	L'étape est activée/ouverte
Exception	Les dates de début et de fin de l'étape ne sont incluses entre celles de la répartition (date de début et fin)

Diagramme d'activité du cas d'utilisation « ouvrir étape»

Le cas d'utilisation « ouvrir étape » regroupe plusieurs activités. Nous commençons par choisir l'option « Etapes », ensuite le système passe à l'affichage des étapes. Une fois, cela est fait, on passe l'activité « cliquer sur le bouton *nouvelle étape* ». Après l'exécution de ces activités, le système passe à une vérification. Si aucune répartition n'est ouverte, un message d'erreur s'affiche et l'activité « cliquer sur le bouton *nouvelle étape* » doit être reprise. Par ailleurs, si une répartition est ouverte, le sytème autorise l'activité « saisie des informations de l'étape ». le systéme passe à nouveau à une vérification. Si la période de l'étape n'est pas incluse dans celle de la répartition, un message d'erreur s'affiche et l'activité « saisie des informations sur l'étape » doit être reprise. Par ailleurs, si elle est incluse, le sytème enregistre l'opération. L'ensemble de ces enchaînements est décrit dans le diagramme d'activité ci-dessous.

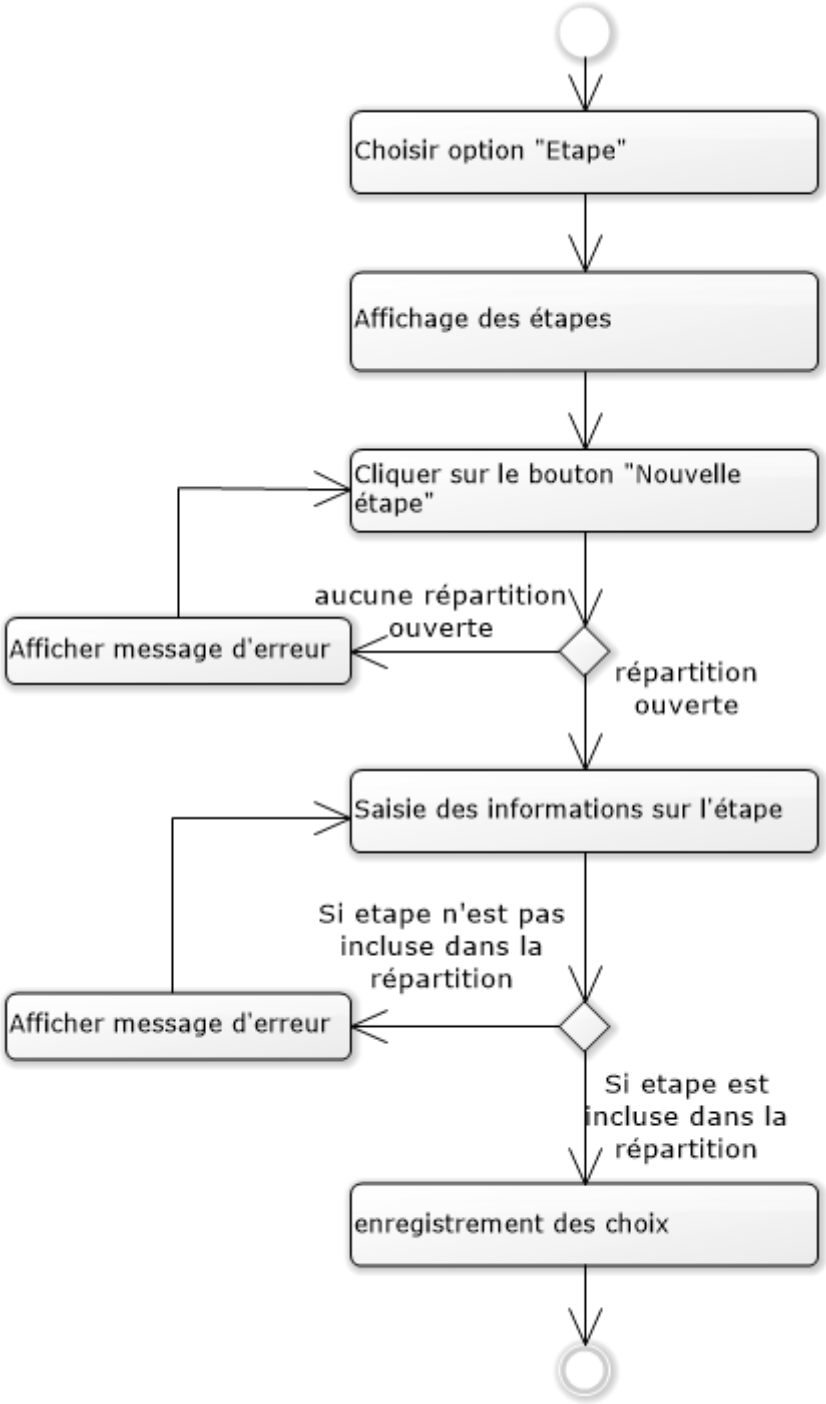


Figure 11 : Diagramme d'activité du cas "ouvrir étape"

Diagramme de séquence du cas d'utilisation « ouvrir étape »

Nous représentons, dans cette partie, le scénario nominal de l'ouverture d'étapes. L'utilisateur demande à ouvrir une étape. Le système procède à une vérification de répartition ouverte. Après cela, si tout se passe bien, le système autorise la saisie et la validation des informations de

l'étape. Le système procède à nouveau à une vérification de l'inclusion de l'étape dans la répartition. Après cela, si tout se passe bien, le système enregistre l'opération. La figure ci-dessous représente le diagramme de séquence illustrant le scénario nominal du cas d'utilisation « gérer étapes ».

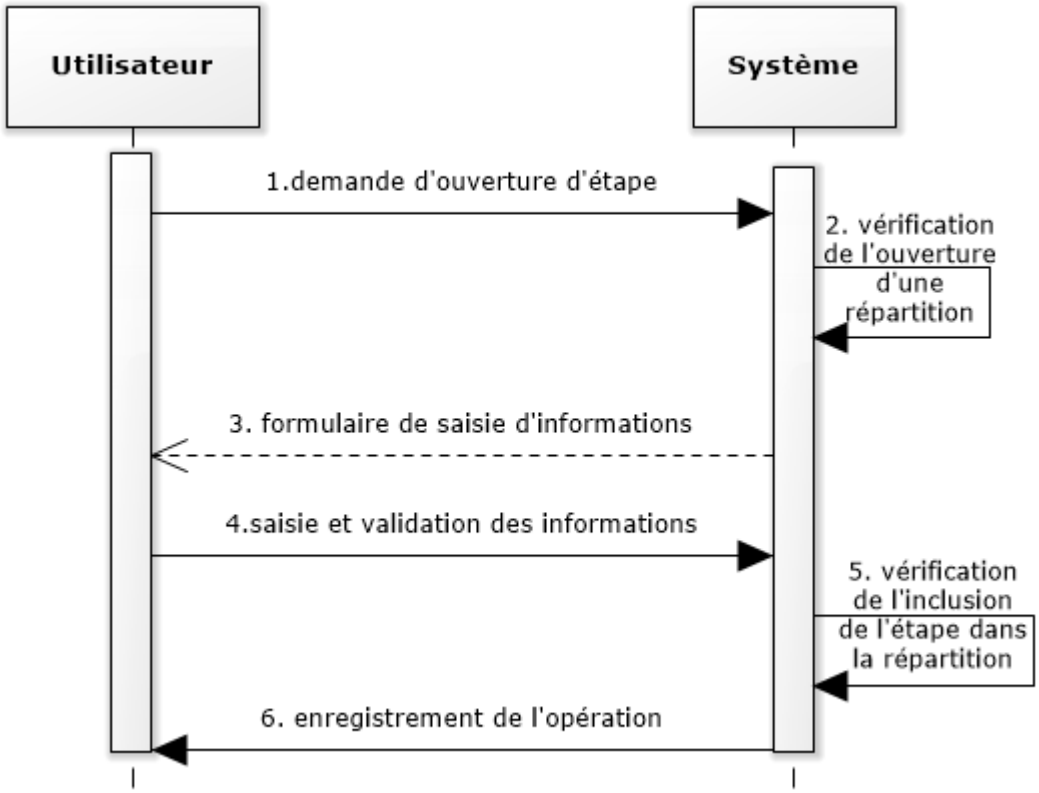


Figure 12 : Diagramme de sequence du cas d'utilisation "ouvrir étapes"

Après l'analyse de l'ouverture d'étape, nous allons, dans la section suivante, décrire le besoin fonctionnel des choix d'enseignements.

3.2.3 Analyse des choix d'enseignements

Dans cette partie, nous décrivons d'abord le cas d'utilisation « choisir enseignements », ensuite le diagramme d'activité des choix d'enseignements et enfin le diagramme de séquence.

Description du cas d'utilisation « choisir enseignements »

Le tableau ci-dessous permet de décrire le cas d'utilisation « s'authentifier ».

Tableau 6 : Description de cas d'utilisation « choisir enseignements »

Description de cas d'utilisation « choisir enseignements »	
Titre	choisir des enseignements
Résumé	Permet de choisir des enseignements ou des cours de CM, de TD ou de TP à dispenser dans le semestre ou dans l'année
Acteurs	Enseignants permanents et vacataires.
Pré condition	Authentification, consulter choix, ouverture d'une étape
Scénario nominal	• L'enseignant clique sur le bouton « choisir mes enseignements » • Le système affiche la liste des enseignements • L'enseignant choisit et valide ses choix d'enseignements • Le système enregistre les choix
Post condition	Message de confirmation des choix
Exception	Etape non encore ouverte

Diagramme d'activité du cas d'utilisation « choisir enseignements »

Le cas d'utilisation « choisir enseignement » regroupe plusieurs activités. Nous commençons par choisir l'option « consulter mes choix », ensuite le système passe à l'affichage des enseignements choisis par l'utilisateur. Une fois, cela est fait, on passe l'activité « cliquer sur le bouton choisir mes enseignements ». Après l'exécution de ces activités, le système passe à une vérification. Si aucune étape n'est ouverte, un message d'erreur s'affiche et l'activité « choisir mes enseignements » doit être reprise. Par ailleurs, si une étape est ouverte, le sytème affiche la liste des enseignements. Dans ce cas l'activité « choisir enseignements » s'exécute puis celle de l'enregistrement des choix. L'ensemble de ces enchaînements est décrit dans le diagramme d'activité ci-dessous.

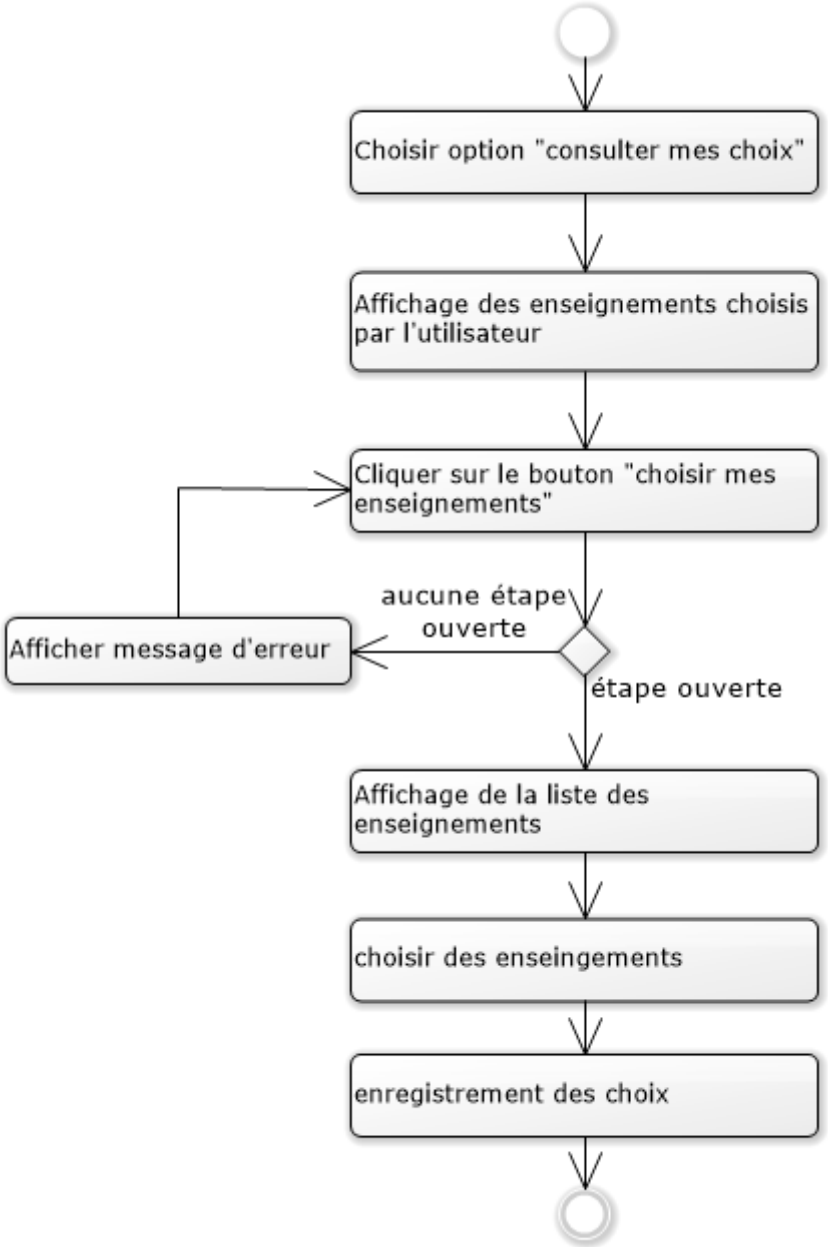


Figure 13 : Diagramme d'activité du cas "choisir enseignements"

Diagramme de séquence du cas d'utilisation « choisir enseignements »

Nous représentons, dans cette partie, le scénario nominal du cas d'utilisation « choisir enseignement ». L'utilisateur demande l'interface de choix des enseigements. Le système procède à une vérification d'étape ouverte. Après cela, si tout se passe bien, le système affiche la liste des enseignements à choisir. L'utilisateur choisit ses enseignements et le système

enregistre ses choix. La figure ci-dessous représente le diagramme de séquence illustrant le scénario nominal du cas d'utilisation « choisir enseignements ».

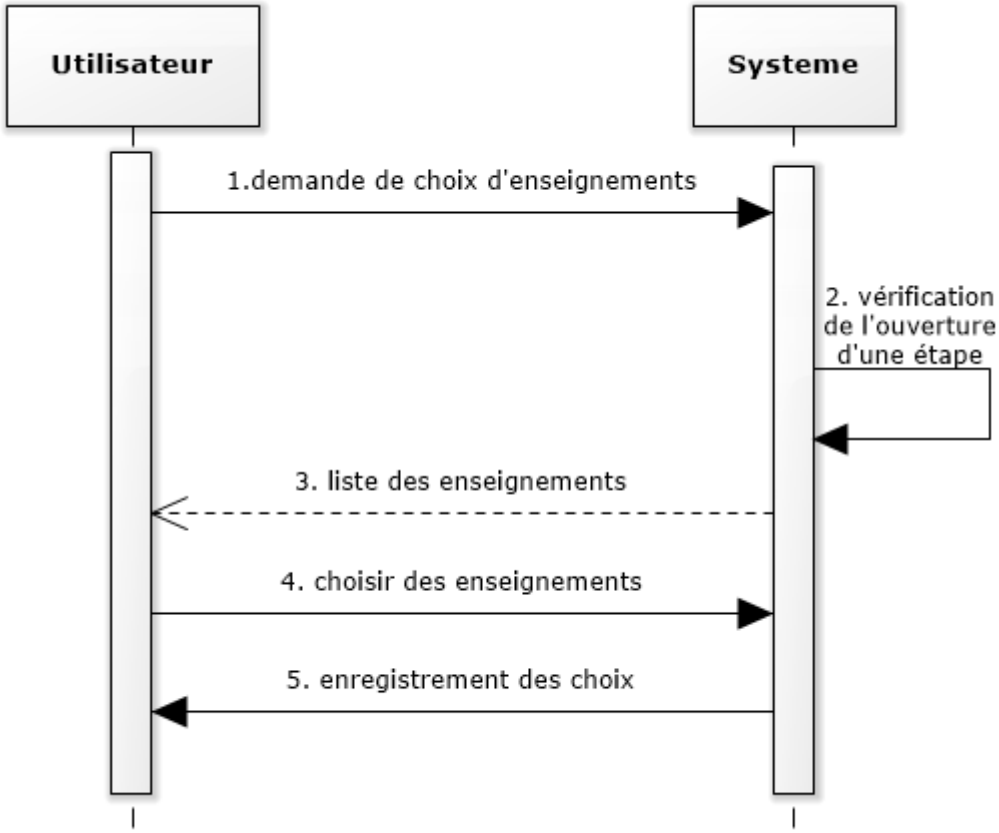


Figure 14: Diagramme de sequence du cas d'utilisation "choisir enseignements"

Dans ce chapitre, nous avons présenté la spécification et l'analyse des besoins fonctionnels accompagnée de la modélisation de notre système d'informations de par les diagrammes de cas d'utilisation, d'activités et de séquence dans le but d'aborder le chapitre suivant sur la conception de notre système.

CHAPITRE 4 : CONCEPTION DU SYSTEME

La conception d'un système est une phase préliminaire et essentielle pour la concrétisation ou la réalisation d'une application. Il s'agit de mettre en œuvre un ensemble d'activités qui, à partir d'une demande d'informatisation d'un processus (demande qui peut aller de la simple question orale jusqu'au cahier des charges complet), permettent la conception, l'écriture et la mise au point d'une application (et donc de programmes informatiques) jusqu'à sa livraison ou déploiement au client ou demandeur [12].

Dans ce chapitre, nous parlons d'abord de la conception générale de notre système en s'appuyant sur l'architecture, et les diagrammes de packages et de déploiement. Ensuite, nous abordons la conception détaillée en s'appuyant sur le diagramme de classes et l'établissement du dictionnaire de données.

4.1 Conception générale

Dans cette section, nous aborderons l'architecture ainsi que les diagrammes de composants, de packages et de déploiement de notre système.

4.1.1 Architecture de l'application

L'architecture d'une application décrit de façon symbolique et schématique les différents éléments d'un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions. Dans notre cas, nous travaillons avec l'architecture trois tiers. Egalement appelée architecture à trois (3) niveaux ou à trois couches, l'architecture trois tiers est une architecture client-serveur dans laquelle coexistent des modules indépendants permettant le rendu d'une interface utilisateur, les process logiques, fonctionnels et métiers ainsi que l'accès aux données. Ainsi, cette architecture est composée de trois niveaux ou couches que sont :

- **La couche de présentation** (premier niveau) : c'est la première couche qui compose l'infrastructure trois tiers. Elle correspond à la partie visible et interactive de l'application pour les utilisateurs. On parle d'interface homme-machine. Elle peut être représentée en HTML, CSS, JavaScript, etc., pour être exploitée par un navigateur web.

- **La couche métier ou fonctionnelle** (deuxième niveau) : c'est la seconde couche qui compose l'infrastructure trois tiers. Elle correspond à un ensemble de composants métiers qui permettent de traiter un ensemble d'actions sur un serveur, et de faire éventuellement appel à des services externes pour envoyer des réponses aux requêtes envoyées par le client via l'interface graphique (la couche présentation). Les langages serveurs les plus utilisés sont : le PHP, le Ruby, le C/C++, etc.
- **La couche d'accès aux données** (troisième niveau) : c'est la troisième couche qui compose l'infrastructure trois tiers. Elle correspond au serveur de la base de données. Il s'agit aussi de la couche d'accès aux données. Sur ce troisième tiers, un SGBD est installé, comme par exemple MySQL ou PostgreSQL.

La figure ci-dessous représente l'architecture de notre application.

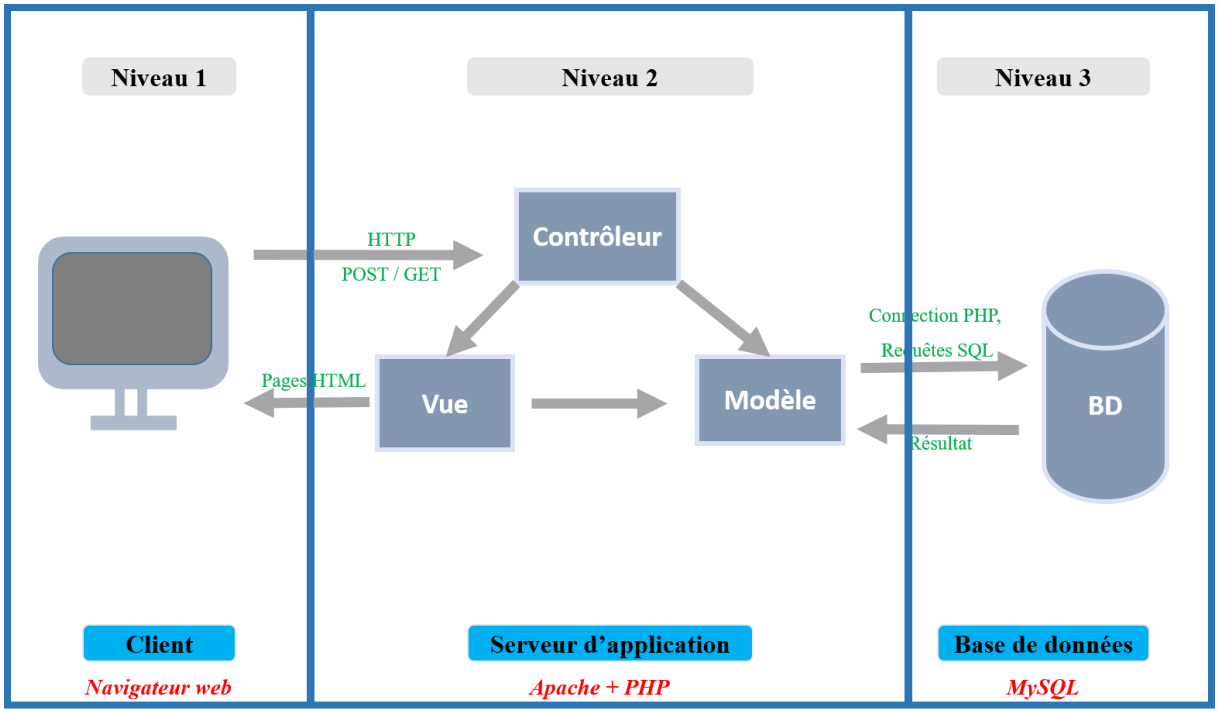


Figure 15 : Architecture de l'application

Dans la partie *serveur d'application* de l'architecture précédente (figure 15), une structure des fichiers a été faite afin de faciliter le développement et la maintenance de l'application : c'est la structure **MVC**.

Le MVC ou Modèle-vue-contrôleur est un motif d'architecture logicielle destiné aux interfaces graphiques et très populaire pour les applications web. Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs.

- Un **modèle** (Model) contient les données à afficher.
- Une **vues** (View) contient la présentation de l'interface graphique.
- Un **contrôleur** (Controller) contient la logique concernant les actions effectuées par l'utilisateur.[23]

4.1.2 Diagramme de composants

Un composant représente une entité logicielle d'un système. Autrement dit, un composant est une entité autonome représentée par un classeur structuré, stéréotypé « component », comportant une ou plusieurs interfaces requises ou offertes [24]. Le diagramme de composants décrit l'organisation du système du point de vue des éléments logiciels comme les modules (paquetages, fichiers sources, bibliothèques, exécutables), des données (fichiers, bases des données) ou encore d'éléments de configuration (paramètres, scripts, fichiers de commandes). Ce diagramme permet de mettre en évidence les dépendances entre les composants (qui utilise quoi).

La figure ci-dessous représente le diagramme de composants de notre système.

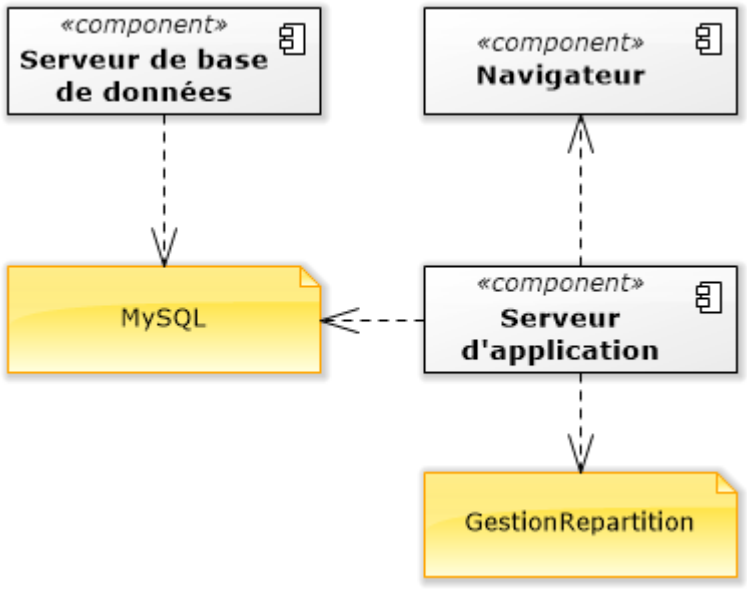


Figure 16 : Diagramme de composants

4.1.3 Diagramme de packages

Un package en UML (ou paquetage en français) sert à regrouper des éléments en un ensemble cohérent, et à fournir un espace de noms pour ces éléments. Un package peut contenir la plus

part des éléments UML : classes, objets, cas d'utilisation, etc. Il peut également contenir d'autres packages, selon une organisation hiérarchique. L'intérêt des packages est de permettre de structurer les diagrammes et de donner une vision globale plus claire [13]. Ainsi, le diagramme de packages permet d'identifier les liens de généralisation et de dépendance entre les packages. La représentation ci-dessous est le diagramme de packages de notre système.

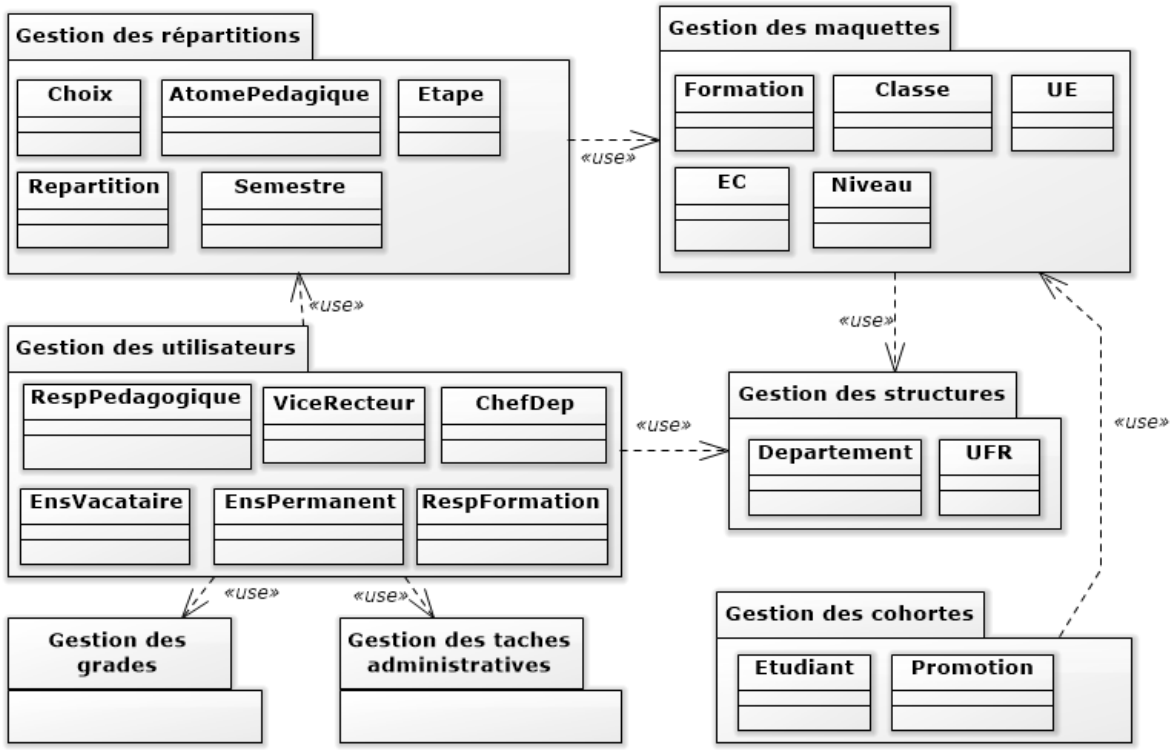


Figure 17 : Diagramme de packages

Notre diagramme de packages (en Figure 11) est composé de sept packages à savoir :

- **Gestion des structures** : dans ce package, nous assurons la gestion (ajout, modification et suppression) des départements et des UFR.
- **Gestion des maquettes** : dans ce package, nous assurons la gestion (ajout, modification et suppression) des formations, des classes, des UE, des EC et des niveaux. Ce package utilise le package « Gestion des structures ».
- **Gestion des cohortes** : dans ce package, nous assurons la gestion (ajout, modification et suppression) des étudiants et des promotions. Ce package utilise le package « Gestion des maquettes ».
- **Gestion des répartitions** : dans ce package, nous assurons la gestion (ajout, modification et suppression) des choix d'enseignements, des enseignements ou atomes

pédagogiques, des semestres et de l'ouverture et de fermeture des étapes et des répartitions. Ce package utilise le package « Gestion des maquettes ».

- **Gestion des grades** : dans ce package, nous assurons la gestion (ajout, modification et suppression) des grades des enseignants.
- **Gestion des tâches administratives** : dans ce package, nous assurons la gestion (ajout, modification et suppression) des tâches administratives attribuées aux enseignants.
- **Gestion des utilisateurs** : dans ce package, nous assurons la gestion (ajout, modification et suppression) des vice-recteurs, des chefs de département, des responsables de formation, des enseignants vacataires et permanent et des responsables pédagogiques. Ce package utilise les packages « Gestion des tâches administratives », « Gestion des grades », « Gestion des structures » et « Gestion des répartitions ».

4.1.4 Diagramme de déploiement

En UML, un diagramme de déploiement est une vue statique qui sert à représenter l'utilisation de l'infrastructure physique par le système et la manière dont les composants du système sont répartis ainsi que leurs relations entre eux. Les éléments utilisés par un diagramme de déploiement sont principalement les nœuds, les composants, les associations et les artefacts. Les caractéristiques des ressources matérielles physiques et des supports de communication serveurs et routeurs peuvent être précisées par stéréotype [14]. La représentation ci-dessous illustre le diagramme de déploiement de notre système.

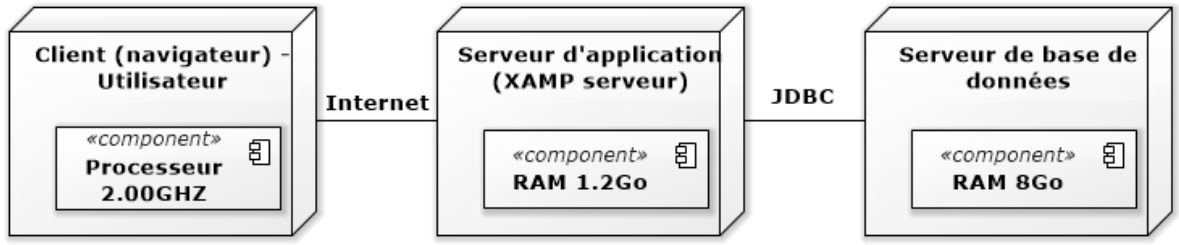


Figure 18 : Diagramme de déploiement

Comme le montre la figure ci-dessus, notre application sera déployée sur des machines de 2 Go de RAM et 2.00GHZ de vitesse processeur pour la plupart. Les serveurs sont contenus dans des machines de 8 Go de RAM. Ils empruntent le réseau internet.

Dans la section qui suit, nous aborderons la conception détaillée qui entrera dans les détails de la conception générale.

4.2 Conception détaillée

La conception détaillée est une phase ultime de la conception de notre système d'information. Elle consiste à construire les classes qui interviennent dans le codage de l'application. Pour cela, nous abordons le diagramme de classes ainsi que le dictionnaire de données de notre système.

4.2.1 Diagramme de classes

Une classe est une description d'un groupe d'objets partageant un ensemble commun de propriétés (les attributs), de comportements (les opérations ou méthodes) et de relations avec d'autres objets (les associations et les agrégations). Une classe de conception est composée par des attributs (données de la classe) et des méthodes (comportement de la classe). Le diagramme de classes est un schéma utilisé pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci.

Dans cette section, nous allons décrire le diagramme de classes de notre système. Pour cela, nous représenterons uniquement les classes participantes aux principales fonctionnalités à savoir la gestion de l'authentification et des utilisateurs, la gestion des répartitions et la gestion des maquettes. Le diagramme de classes complet est mis en *Annexe 1*.

4.2.1.1 Diagramme de classes participantes aux fonctionnalités de la gestion l'authentification et des utilisateurs

Dans le diagramme, ci-dessous, Utilisateur, RespPédagogique, ChefDépartement, EnsVacataire, Enseignant, EnsPermanent, ViceRecteur et RespFormation sont les classes participantes. Il y a six utilisateurs que sont : le vice-recteur, le chef de département, le responsable de formation, l'enseignant permanent, l'enseignant vacataire et le responsable de pédagogique. Ce diagramme permet de gérer les utilisateurs, leur authentification ainsi que leur espace de travail numérique.

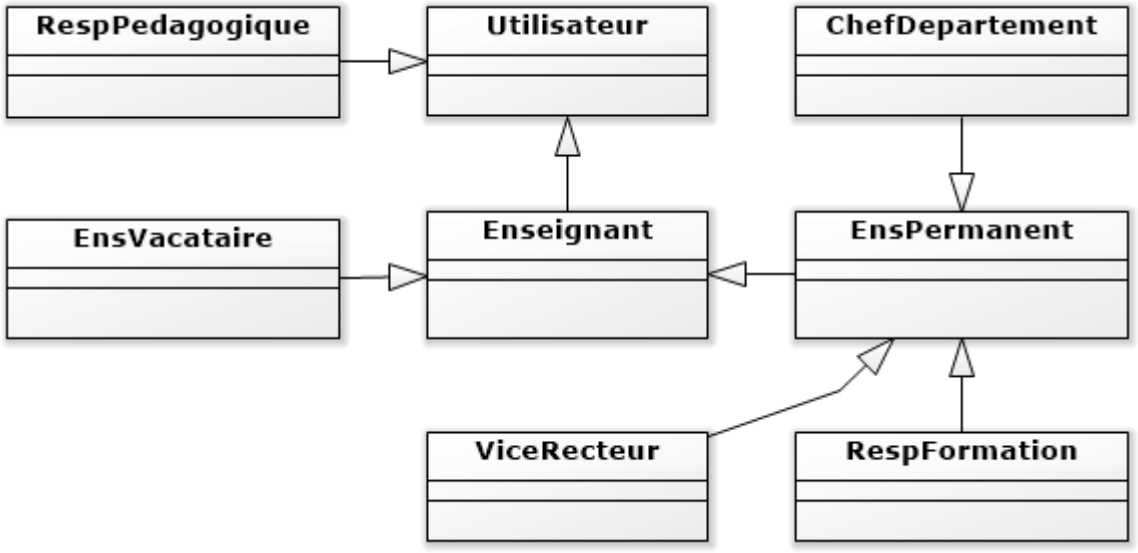


Figure 19 : Diagramme de classes participantes aux fonctionnalités de la gestion l'authentification et des utilisateurs

4.2.1.2 Diagramme de classes participantes aux fonctionnalités de la gestion des maquettes

Les classes intervenant dans le diagramme ci-dessous sont : Formation, Classe, Niveau, UE, et EC. Ce diagramme permet de gérer les maquettes de par les formations, les classe et niveaux, les unités d'enseignement (UE) et les élèments constitutifs (EC).

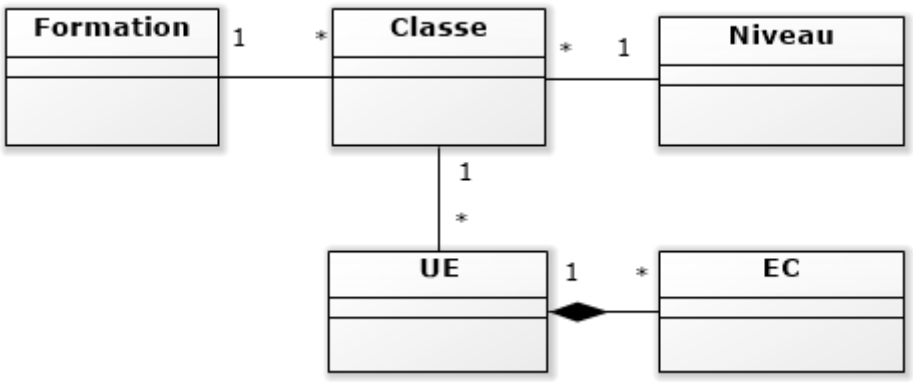


Figure 20 : Diagramme de classes participantes aux fonctionnalités de la gestion des maquettes

4.2.1.3 Diagramme de classes participantes aux fonctionnalités de la gestion des répartitions

Les classes intervenant dans le diagramme ci-dessous sont : Semestre, Répartition, Etape, Enseignement, EC, UE, Enseignant et Choix. Ce diagramme permet, pour chaque semestre, d'ouvrir ou de fermer une répartition ainsi que les étapes de celle-ci. Dans chaque étape, la liste des enseignements (groupes ou atomes pédagogiques) est établie afin que les enseignants choisissent des enseignements à dispenser dans le semestre.

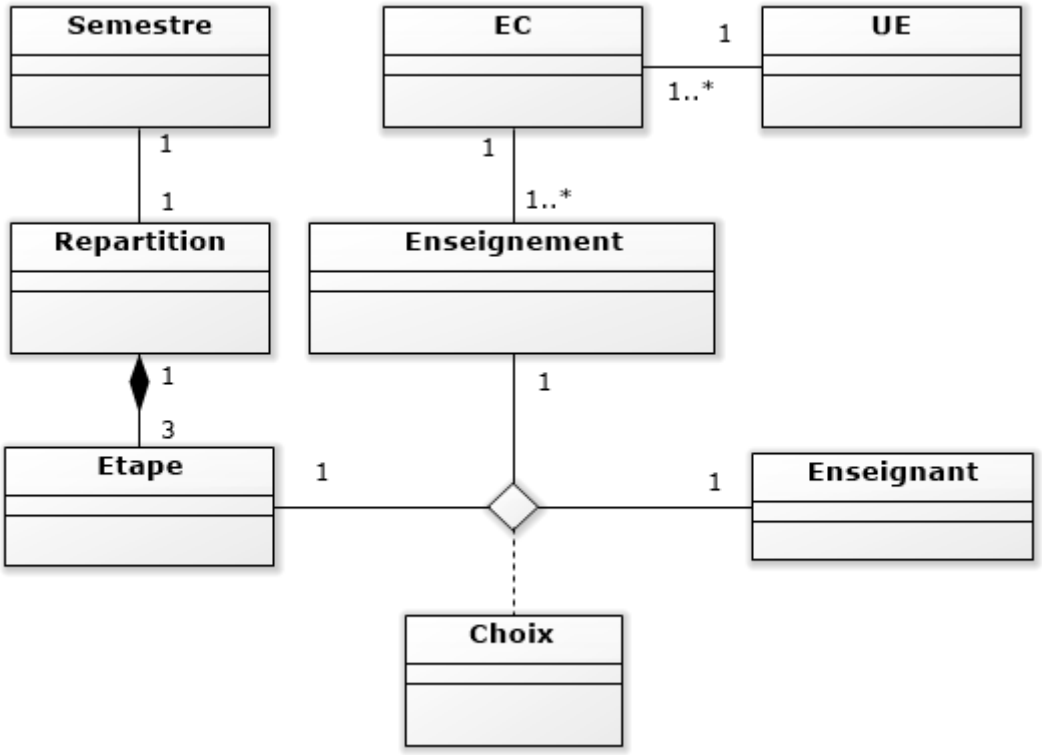


Figure 21 : Diagramme de classes participantes aux fonctionnalités de la gestion des répartitions

4.2.2 Dictionnaire de données

Un dictionnaire des données est une collection de métadonnées ou de données de référence nécessaire à la conception d'une base de données relationnelles. Il revêt une importance particulière, car il est le vocabulaire commun de la conception, mais aussi décrit des données importantes. Il est représenté, dans notre cas, par un tableau à quatre colonnes contenant le noms des tables, les champs (les données), le type de données ainsi que les commentaires.
Le dictionnaire de données de notre système est représenté en *Annexe 2*.

Dans ce chapitre, nous avons évoqué la conception générale dans laquelle nous avons parlé de l'architecture du système, les diagrammes de package et de déploiement. Nous avons terminé par la conception détaillée où nous avons présenté les diagrammes de classes ainsi que le dictionnaire de données. Tout cela va dans le sens de préparer l'implémentation et la réalisation de l'application qui feront l'objet du chapitre suivant.

CHAPITRE 5 : IMPLEMENTATION ET PRESENTATION DE L'APPLICATION

Ce chapitre est consacré à l'implémentation (ou codage) et à la présentation de l'application en tant que telle. Pour ce faire, nous allons d'abord présenter les outils et technologies (soient des outils de conception, de codage, de SGBDR et de langage de programmation) utilisés pour le codage. Ensuite, nous présenterons les principales interfaces graphiques de l'application.

5.1 Outils et technologies utilisés

Le développement d'une application telle que la nôtre (système de gestion de la répartition des enseignements à l'UASZ) nécessite l'utilisation de certains outils et technologies, que ce soit lors de la conception, du codage, de la gestion de base de données ou de l'implémentation.

5.1.1 Outil de conception : Software Ideas Modeler

Un outil de conception est un logiciel qui permet de concevoir des diagrammes, comme les diagrammes de cas d'utilisation, de classes, de séquences, etc. **Software Ideas Modeler** est un des logiciels de conception parmi tant d'autres (ArgoUML, Papyrus, UML Designer, etc.). C'est un outil d'UML et de génie logiciel assisté par ordinateur. Le modeleur prend en charge tous les 14 types de diagramme spécifiés dans UML 2.4. Il supporte également entre autres les diagrammes et standards, comme les organigrammes, les diagrammes de flux de données, etc.

5.1.2 Environnement de développement : Visual Studio Code

En programmation informatique, un environnement de développement est un ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui développent des logiciels. Il comporte un éditeur de texte destiné à la programmation, des fonctions qui permettent, par

pression sur un bouton, de démarrer le compilateur ou l'éditeur de liens ainsi qu'un débogueur en ligne, qui permet d'exécuter ligne par ligne le programme en cours de construction [15]. **Visual Studio Code** est un environnement de développement parmi tant d'autres (Eclipse, NetBeans, QtCreator, etc.). C'est un éditeur de code extensible développé par Microsoft pour Windows, Linux et OS X.

5.1.3 Langage de programmation : PHP

Un langage de programmation est une notation conventionnelle destinée à formuler des algorithmes et produire des programmes informatiques qui les appliquent [16]. **PHP** (officiellement, ce sigle est un acronyme récursif pour PHP Hypertext Preprocessor) est un langage de programmation parmi tant d'autres (Java, C , C++, etc.). Autrement dit, c'est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web. Il peut être intégré facilement au HTML.

5.1.4 Langage de requête : SQL

Dans Wikipédia, **SQL** (Structured Query Language, en français langage de requête structurée) est un langage informatique normalisé servant à exploiter des bases de données relationnelles.

5.1.5 Langage de balisage : HTML

Les langages de balisage représentent une classe de langages spécialisés dans l'enrichissement d'information textuelle. Ils utilisent des balises, unités syntaxiques délimitant une séquence de caractères ou marquant une position précise à l'intérieur d'un flux de caractères (par exemple un fichier texte) [17]. L'HyperText Markup Language, généralement abrégé HTML est un langage de balisage parmi tant d'autres (DocBook, TEI, WML, etc.). Autrement dit, c'est le langage de balisage conçu pour représenter les pages web. C'est un langage permettant d'écrire de l'hypertexte, d'où son nom.

5.1.6 Frameworks : Bootstrap et jQuery

En programmation informatique, un framework désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel [18].

L'utilité d'un framework est d'éviter de passer du temps à développement ce qui a déjà été fait par d'autres, souvent plus compétents, et qui a en plus utilisé et validé par de nombreux utilisateurs. **Bootstrap** et **jQuery** sont des frameworks parmi tant d'autres (Symfony, Zend Framework, Laravel, etc.).

- **Bootstrap** est une collection d'outils utile à la création du design de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option [19].
- **jQuery** est une bibliothèque JavaScript libre et multi-plateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web [20].

5.1.7 Serveur web : XAMPP

Un serveur web est spécifiquement un serveur multi-service utilisé pour publier des sites web sur Internet ou un intranet. **XAMPP** (X (cross) Apache MariaDB Perl PHP) est un serveur web parmi tant d'autres (EasyPHP, WAMP, etc.). C'est un ensemble de logiciels permettant de mettre en place facilement un serveur Web local, un serveur FTP et un serveur de messagerie électronique. Il offre une bonne souplesse d'utilisation, réputée pour son installation simple et rapide [21].

5.1.8 SGBDR : MySQL

Un Système de Gestion de Bases de Données Relationnelles (abr. SGBDR) est un logiciel système servant à stocker, à manipuler ou à gérer et à partager des informations dans une base de données, en garantissant la qualité, la pérennité et la confidentialité des informations, tout en cachant la complexité des opérations [22]. **MySQL** est un SGDBR parmi tant d'autres (Oracle Database, PostgreSQL, SQLite, etc.). Il fait partie des logiciels de gestion de bases de données les plus utilisés au monde.

5.1.9 Logiciel de gestion et d'administration de base de données : MySQL Workbench

MySQL Workbench est un logiciel de gestion et d'administration de base de données MySQL. Via une interface graphique intuitive, il permet, entre autres, de créer, modifier ou supprimer

des tables, des comptes utilisateurs, et d'effectuer toutes les opérations inhérentes à la gestion d'une base de données. Pour ce faire, il doit être connecté à un serveur MySQL [24].

Le choix des outils et technologies à utiliser pour la réalisation de ce projet nous mène directement à la phase de l'implémentation de l'application. Cette dernière sera l'objet de la section suivante.

5.2 Implémentation

La phase de l'implémentation consiste au codage (ou à la programmation) du système. Pour ce faire, nous allons d'abord présenter l'implémentation de la base de données, ensuite celle de l'application.

5.2.1 Implémentation de la base de données

L'implémentation de la base de données consiste d'abord à mettre en place le modèle logique de données pour ensuite passer à l'implémentation de ce dernier.

5.2.1.1 Modèle logique de données (MLD)

Le modèle logique des données (MLD) consiste à implanter une base de données dans un SGBDR, c'est-à-dire le traduire dans un langage de définition de données. Le langage utilisé pour ce type d'opération est le SQL.

La migration des clés primaires dépend des cardinalités des tables. Les clés primaires migrent vers les tables où les cardinalités sont minimes. Dans notre cas, les clés primaires sont précédées par une clé jaune tandis que les clés étrangères sont précédées par une clé rouge.

La figure ci-dessous représente notre MLD.

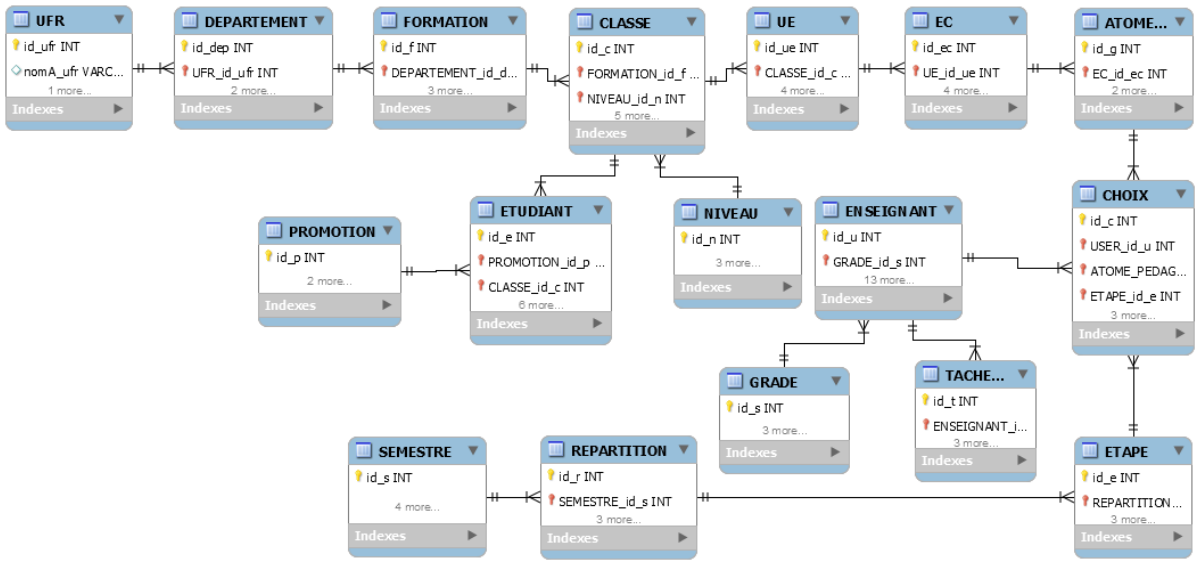


Figure 22 : Modèle Logique de Données (MLD)

5.2.1.2 Etablissement de connexion à la base de données

La connexion à la base de données nécessite le démarrage du serveur XAMPP. A ce niveau nous pourrons nous connecter à la base de données nommée « re-uasz ». Ce nom sera mentionné lors de la configuration du fichier DBFactory (Figure 19) pour établir la connexion.

```php
<?php
class DBFactory{
    public static function getMysqlConnexionWithPDO(){
        $db = new PDO('mysql:host=localhost;dbname=re-uasz', 'root', '');
        $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $db;
    }
}
?>
```

Figure 23 : DBFactory

Dans la section qui suit, nous parlons du codage de l'application.

5.2.2 Implémentation de l'application

L'implémentation est une phase du projet qui consiste à coder l'application en tant telle.

Ainsi, dans cette partie, nous allons présenter quelques captures d'écran de codes du module « Enseignement » en l'adaptant au modèle MVC.

5.2.2.1 La classe « Enseignement » : Modèle

La figure ci-dessous représente la classe Enseignement permettant d'afficher l'ensemble des données pour un enseignement.

```php
<?php
class Enseignement{
    protected $erreurs = array(),$id,$categorie,$type,$ec;
    public function hydrate($donnees){
        foreach ($donnees as $attribut => $valeur){
            $methode = 'set'.ucfirst($attribut);
            if (is_callable(array($this, $methode))){
                $this->$methode($valeur);
            }
        }
    }
    public function __construct($valeurs = array()){
        if (!empty($valeurs)) {
            $this->hydrate($valeurs);
        }
    }
    public function erreurs(){return $this->erreurs;}
    public function getId(){return $this->id;}
    public function getCategorie(){return $this->categorie;}
    public function getType(){return $this->type;}
    public function getEc(){return $this->ec;}
    public function setId($id){$this->id = $id;}
    public function setCategorie($categorie){$this->categorie = $categorie;}
    public function setType($type){$this->type = $type;}
    public function setEc($ec){$this->ec = $ec;}
}
?>
```

Figure 24 : Classe « Enseignement » : Modèle

5.2.2.2 La classe « ManagerEnseignement » : Contrôleur

La figure ci-dessous représente la classe permettant de Contrôler les traitements sur Enseignement. Elle permet d'orienter toutes les actions (ajout, suppression, modification, …) effectuées par l'utilisateur vers leurs méthodes ou pages concernant.

```php
<?php
namespace Applications\Backend\Modules\Enseignement;
class EnseignementController extends \Library\BackController {
    // ...
    public function executeDeleteComment(\Library\HTTPRequest $request) {
        $this->managers->getManagerOf('Comments')->delete($request->getData('id'));
        $this->app->user()->setFlash('L\'Enseignement a bien été supprimé !');
        $this->app->httpResponse()->redirect('Enseignement.class.php');
    }
}
```

Figure 25 : Classe «EnseignementController » : Contrôleur

5.2.2.3 Le fichier « Enseignement » : Vue

L'aspect visuel de l'application est représenté par la figure ci-dessous. Elle représente le design et l'interface graphique afin de faciliter l'utilisation du système.

```php
<?php require 'inc/autoload.php';?>
<!DOCTYPE html>
<html lang="en">
<head>···
</head>
<body>
    <?php include ('inc/navbar1.php'); ?>
    <?php include ('inc/navbar2.php'); ?>
    <div class="container" id="main">
        <div class="row">
            <div class="col-md-3 col-sm-5">
                <div class="box box-primary">
                    <form role="form" action="enseignement.php" method="post" enctype="multipart/form-data">···
                    </form>
                </div>
            </div>
            <div class="col-md-45 col-sm-9">
                <div class='titreTab'><h4><b>LISTE DES ENSEIGNEMENTS</b></h4></div>
                <table class="table table-bordered table-striped" id="example1">···
                </table>
            </div>
        </div>
    </div>
    <?php include ('inc/ref.php'); ?>
</body>
</html>
```

Figure 26 : Fichier « Enseignement » : Vue

5.2.2.4 Le fichier « ManagerEnseignement » : métier

La figure ci-dessous représente la classe Manager Enseignement permettant de gérer toutes les actions (ajout, suppression, modification, …) effectuées par l'utilisateur.

```php
<?php
class EnseignementManager{
    protected $db;
    public function __construct(PDO $db){
        $this->db = $db;
    }
    public function add(Enseignement $enseignement){···
    }
    public function count(){···
    }
    public function delete($id){···
    }
    public function exists($info){···
    }
    public function getUnique($id){···
    }
    public function getList($debut = -1, $limite = -1){···
    }
    public function update(Enseignement $enseignement){···
    }
    public function save(Enseignement $enseignement){···
    }
    public function setDb($db){···
    }
}
?>
```

Figure 27 : La classe « ManagerEnseignement » : méthier

De façon générale, les logiciels et applications sont exposés à des risques d'attaques (cybercriminalité). Il faudra donc, avant, pendant et après le développement, mettre en œuvre des techniques de sécurité afin de protéger les applications des éventuelles attaques. Ainsi, dans la section qui suit, nous parlerons de la sécurisation de notre application.

5.3 Sécurisation de l'application

Notre application est développée avec la technologie PHP. Nous avons pris le temps de la protéger contre les éventuelles attaques en passant notamment par :

L'application de certaines règles de configuration

Afin de sécuriser notre application, nous avons appliqué certaines règles lors de la configuration de l'environnement d'exécution :

- **limiter l'accès aux répertoires**. Pour cela, nous avons utilisé le *safe mode* qui est un mode de sécurité de PHP. Nous l'avons activé afin qu'il empêche un programme d'accéder à des fichiers situés en dehors du dossier où se trouve notre application.
- **Empêcher l'interprétation des requêtes malintentionnées**. La fonction *addlashes()* permet d'ajouter le caractère "\" devant les apostrophes, les guillemets et le caractère nul. Elle empêche le système d'interpréter les requêtes qu'un utilisateur malintentionné aurait saisi dans un formulaire.

L'authentification avant utilisation de l'application

Afin de limiter l'accès à l'application, nous avons mis en œuvre un système d'authentification (saisie de *pseudo* et *de mot de passe*) pour les utilisateurs. Cela nous permettra de répondre aux questions suivantes : qui s'est connecté ? Qu'est-ce qu'il a le droit de faire ? Qu'est-ce qu'il a fait ?.

La mise en œuvre d'une stratégie de programmation défensive

La programmation défensive est une méthode qui consiste à écrire son code de façon à s'attendre à des attaques. En effet, l'utilisateur peut insérer des fautes non détectées. Pour éviter que cela arrive, nous avons prévu des traitements pour les fautes :

- **L'injection de données**. Afin d'éviter que des données non voulues soient interprétées dans le code (requêtes SQL, par exemple), nous les avons bien vérifiées avant de les passer en paramètres à des fonctions de notre application. En effet, nous avons testé si les types des variables sont bien les types attendus. En plus, les mots de passe des utilisateurs sont cryptés par la fonction *MD5()*.
- **Journaliser toutes les erreurs**. Cela pour capter les erreurs afin de repérer les attaques**.** Par exemple, les erreurs envoyées à un utilisateur après trois ou quatre fois de tentatives de connexion (saisie de pseudo et de mot de passe) sans succès sont susceptibles de laisser croire que c'est un cybercriminel.

La prise en garde de notre environnement réseau et d'hébergement

Sur le plan d'hébergement, nous avons prévu de réaliser certaines actions pour se munir des attaques :

- **Des outils comme les reverse proxy**. Ils nous permettront d'écarter certaines requêtes inattendues.

- **Utiliser un pare-feu**. Dans le but d'éviter l'inclusion de fichier PHP distant (PHP include), nous bloquerons les connexions sortantes depuis le serveur web**.**

Maintenant, nous allons, dans la section suivante, présenter l'application que nous avons développée.

5.4 Présentation de quelques interfaces graphiques de l'Application

Les interfaces graphiques concernent une partie importante dans la réalisation d'une application (que ce soit web, mobile ou desktop) car permettant à l'utilisateur d'y naviguer (utiliser) facilement.

Dans cette partie, afin de bien présenter les principales interfaces graphiques, nous allons faire un petit scénario. Ce dernier consistera à simuler, de façon pratique et générale, la répartition des enseignements à l'UASZ. Mais avant cela, nous présentons une vue globale de l'application sur la figure suivante.

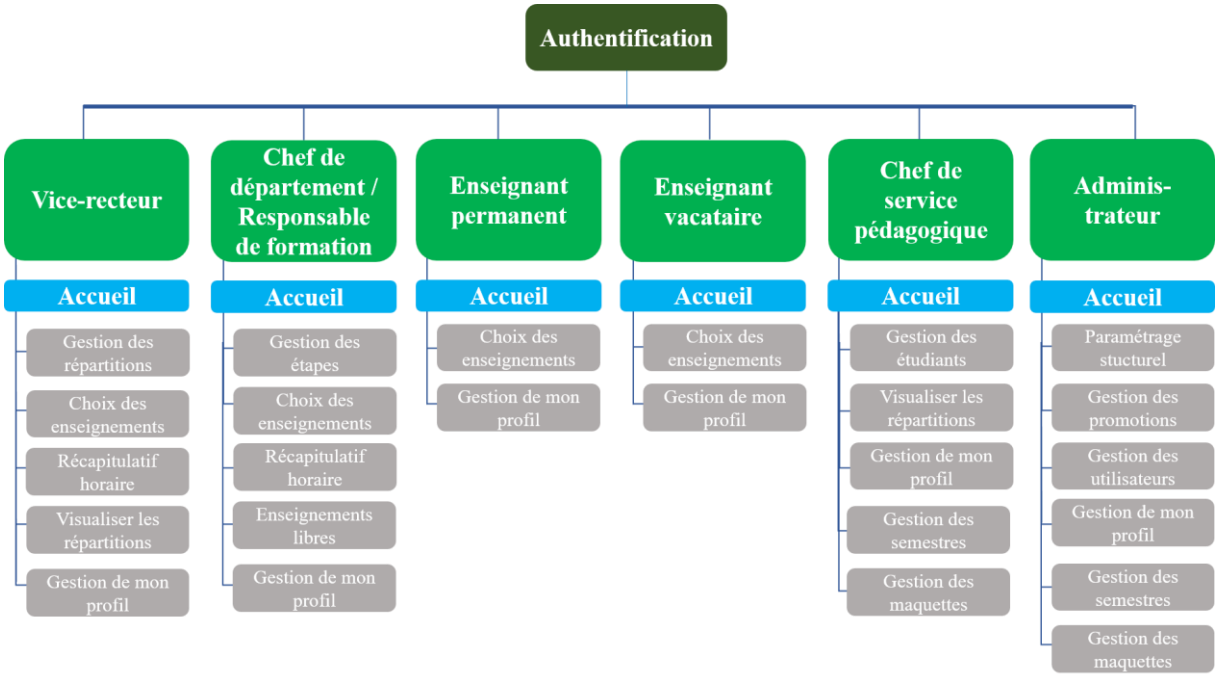


Figure 28 : Vue d'ensemble de l'application

5.4.1 Authentification

Le processus de la répartition des enseignements à l'UASZ commence par l'ouverture de la période de répartition elle-même, qui sera assurée par le **Vice-recteur EVU**, et uniquement lui.

Mais avant cela, ce dernier doit s'identifier via l'interface suivante en saisissant son **speudo** et son **mot de passe**.

Figure 29 : Interface d'authentification

5.4.2 Ouverture de la période de répartition

Une fois son authentification réussie, le **Vice-recteur EVU** pourra accéder à l'interface (voir la figure suivante) dédiée à l'ouverture de la période de répartition dans laquelle il fournit les dates de début et de fin de celle-ci. C'est cette ouverture de répartition qui permettra, par la suite, d'ouvrir des étapes.

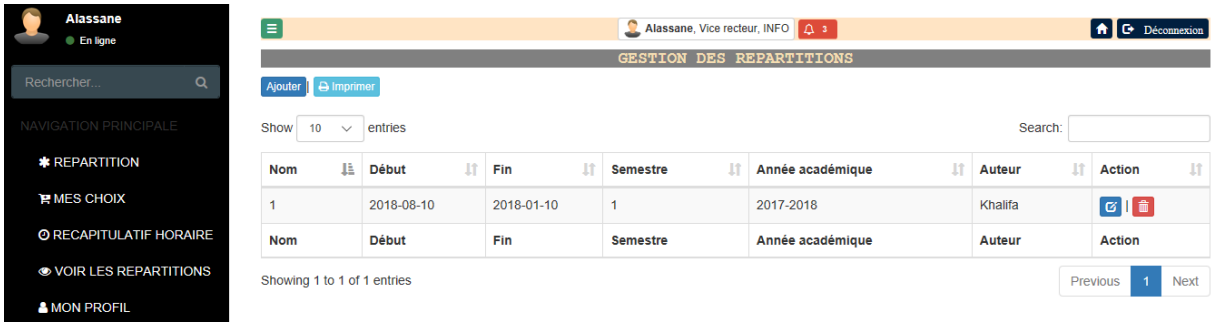


Figure 30 : Interface d'ouverture de répartition

5.4.3 Ouverture d'étapes

Après l'ouverture de la répartition, le **chef de département** (ou le **responsable de formation**) aura la possibilité, via l'interface ci-dessous, d'ouvrir des étapes (en donnant les dates de début et de fin) afin que des choix d'enseignements puissent être faits.

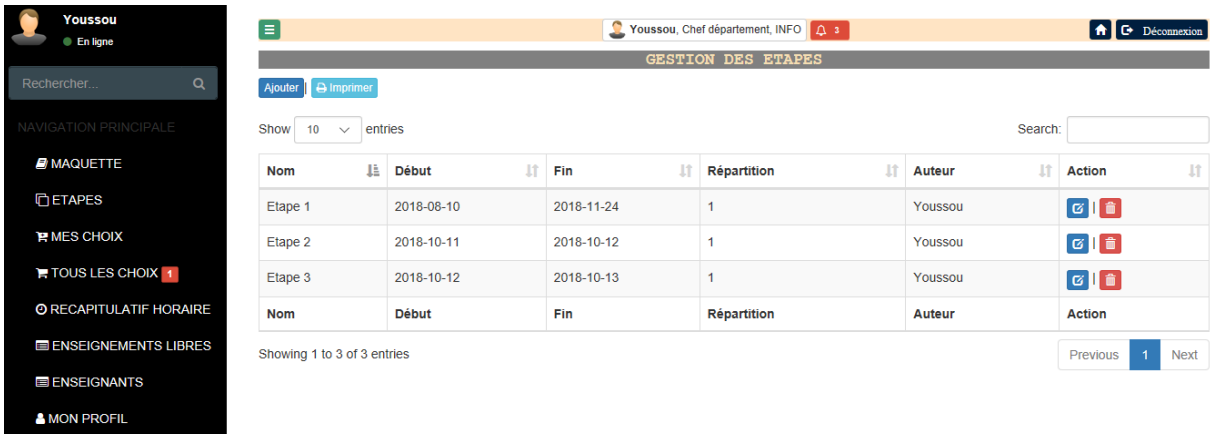


Figure 31 : Interface d'ouverture d'étapes

5.4.4 Choix d'enseignements

Une fois l'étape ouverte, les **enseignants permanents et vacataires** pourront choisir des enseignements via l'interface ci-dessous permettant de les lister.

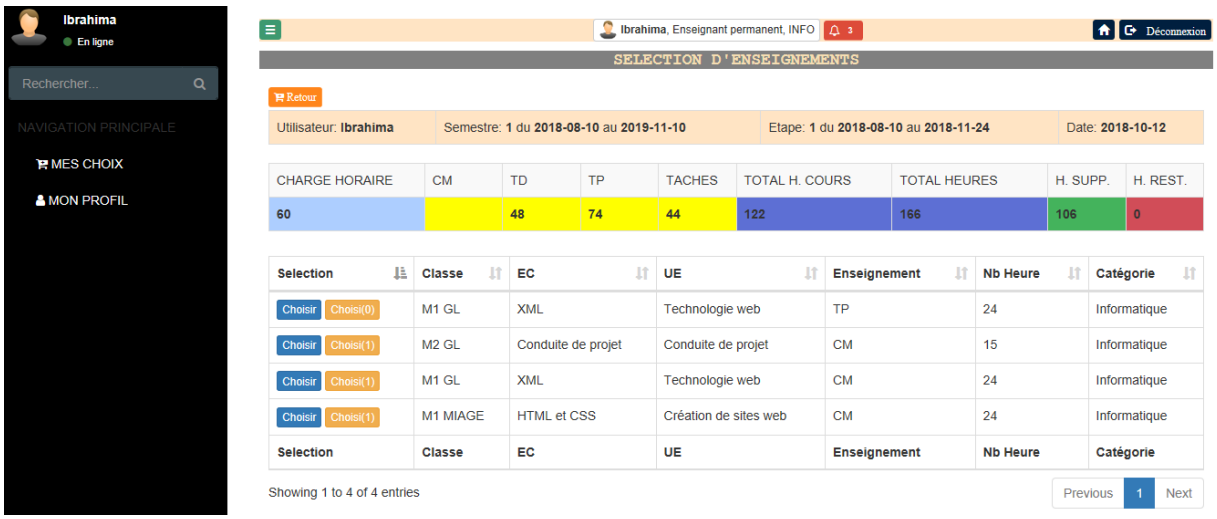


Figure 32 : Interface des enseignements à choisir

Après avoir choisi ses enseignements, l'enseignant pourra les visualiser via l'interface ci-dessous représentant le panier d'enseignements choisis. De là, l'enseignant saura automatiquement ses choix qui sont en conflit et les annuler s'il le souhaite, sinon il laisse le chef de département régler le conflit.

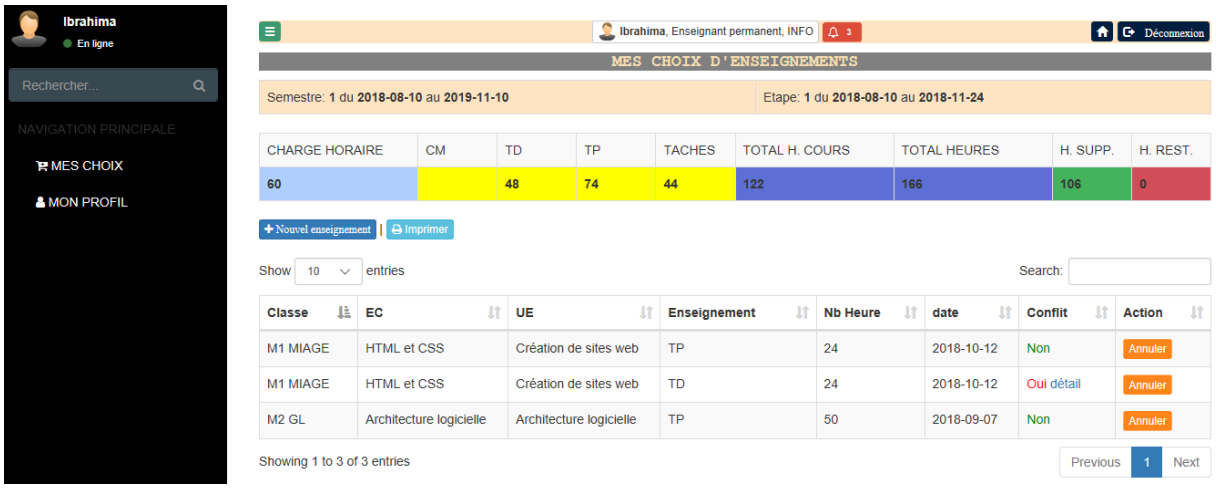


Figure 33 : Interface de panier d'enseignements choisis

5.4.5 Réglage de conflits

Suite aux conflits (un enseignement choisi par plusieurs enseignants), le **chef de département** (ou le **responsable de formation**) accède à l'interface ci-dessous de visualisation et de réglage de conflits

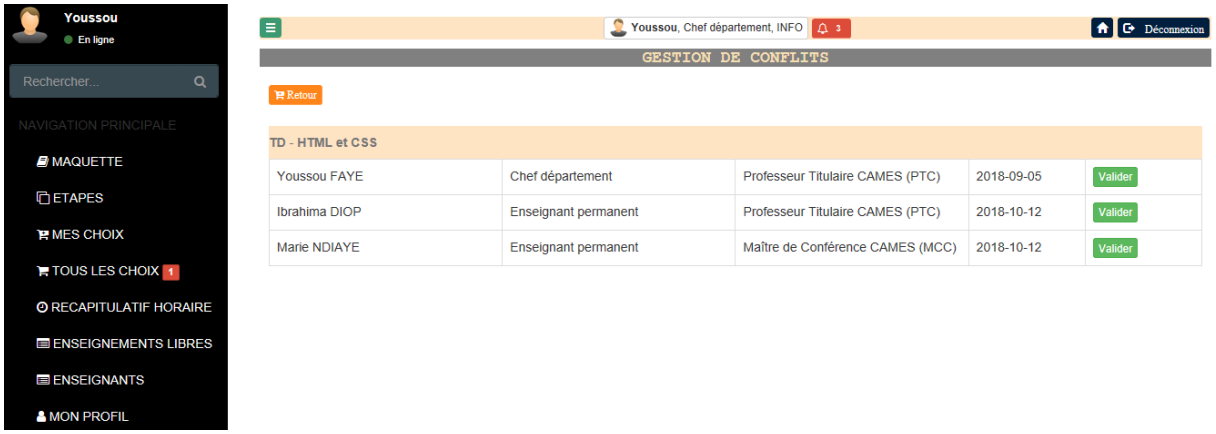


Figure 34 : Interface de visualisation et de réglage de conflits

5.4.6 Visualisation et validation des répartitions

L'objectif de la répartition des enseignements n'est pas seulement l'attribution des enseignements aux enseignants, mais aussi de permettre, après choix des enseignements et réglage des éventuels conflits, l'élaboration des emplois du temps. Pour cela, c'est au Vice-recteur EVU de visualiser et valider les répartitions dans chaque département (voir l'interface suivante) avant que les chefs de service pédagogique s'en servent pour élaborer les emplois du temps.

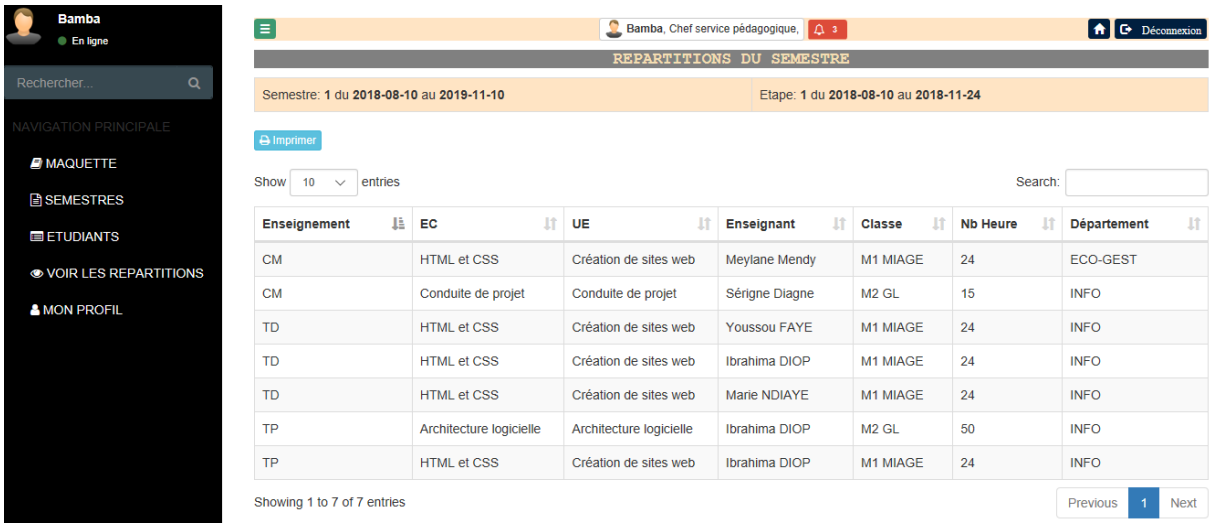


Figure 35 : Interface de visualisation des répartitions dans chaque département

Dans ce dernier chapitre, nous avons enfin montré le résultat tant attendu, l'application en tant que telle. En effet, les technologies et outils utilisés pour implémenter l'application ont été montrés ainsi que quelques interfaces graphiques de l'application.

CONCLUSION GENERALE ET PERSPECTIVES

Ce travail avait comme objectif de concevoir et d'implémenter un logiciel pour la gestion de la répartition des enseignements à l'UASZ.

Pour cela, l'objectif est atteint, car le logiciel est conçu et implémenté. Il répond parfaitement aux attentes décrites sur le cahier des charges. En effet, ce logiciel permet, entre autres, à chaque département, de pouvoir ouvrir et fermer des répartitions et des étapes, de mettre en place la liste des enseignements à choisir, de permettre à chaque enseignant de faire ses propres choix sur ces enseignements, de régler les conflits lors des choix et de générer la répartition de chaque département afin de faciliter l'élaboration des emplois du temps.

A cet effet, afin de bien présenter les objectifs et les résultats de notre travail, ce mémoire a été scindé en cinq principaux chapitres. Le premier chapitre a fait l'objet de la présentation du contexte justificatif du sujet ainsi que les solutions aux problèmes rencontrés lors de la répartition des enseignements. Le deuxième chapitre a parlé du processus de développement dans lequel nous avons présenté et adapté de la méthodologie Scrum à notre projet. Le troisième chapitre s'est intéressé à la spécification et l'analyse des besoins fonctionnels du système. Le quatrième chapitre a abordé la conception de l'application. Enfin, le cinquième chapitre a présenté le résultat, à savoir les interfaces de l'application créée.

Cependant, l'application que nous avons présentée dans ce document présente quelques limites, à savoir :

- **L'inclusion des groupes de classes lors des choix d'enseignements**. Cela permettra à certains enseignants de choisir des enseignements pour un ou deux groupes de la classe ;
- **La prise en rigueur des équivalences entre les heures de CM, de TD et de TP**. Cette fonctionnalité permettra de bien calculer et comparer les heures d'enseignements choisis et la charge horaire pour un enseignant donné.

Dans les perspectives, nous envisageons d'améliorer l'application de par les fonctionnalités suivantes :

- **L'envoi de mails automatiques après réglage des conflits**. Cela consistera à informer les enseignants qui ont perdu l'enseignement suite au conflit, afin qu'ils en choisissent d'autres ;
- **La validation des répartitions**. Permettre au Vice-recteur EVU de valider les répartitions avant que les chefs de service pédagogiques ne puissent les visualiser et élaborer les emplois du temps.

WEBOGRAPHIE

[1] : Site de l'UASZ www.univ-zig.sn. Consulté le 05/07/2018

[2] : Site de Iris http://www.lsis.org/dea/M6optionD/Exp-GL-UML. Consulté le 05/07/2018

[3] : Site du LIPN de l'Université de Paris 13 https://lipn.univ-paris13.fr/~gerard/uml-s2/uml-cours04.html. Consulté le 05/07/2018

[5] : Site du LIPN de l'Université de Paris 13 https://lipn.univ-paris13.fr/~gerard/uml-s2/uml-cours04.html. Consulté le 05/07/2018

[6] : Site de développez.com https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-activites. Consulté le 06/07/2018

[7] : Site du LIPN de l'Université de Paris 13 https://lipn.univ-paris13.fr/~gerard/uml-s2/uml-cours05.html. Consulté le 06/07/2018

[8] : Site de l'Agiliste https://agiliste.fr/introduction-methodes-agiles/. Consulté le 09/07/2018

[10] : Site de l'Agiliste https://agiliste.fr/introduction-methodes-agiles/. Consulté le 10/07/2018

[12] : Site de EMSE https://www.emse.fr/~boissier/enseignement/aco/pdf/UML.analyseconceptionUML.4pp.pdf. Consulté le 12/07/2018

[13] : Site de Sparx Systems https://www.sparxsystems.fr/resources/uml2_tutorial/uml2_packagediagram.html. Consulté le 27/07/2018

[14] : Site de développez.com https://laurent-audibert.developpez.com/Cours-UML/?page=diagrammes-composants-deploiement. Consulté le 27/07/2018

[15] : Site de Wikipédia https://fr.wikipedia.org/wiki/Environnement_de_développement. Consulté le 07/08/2018

[16] : Site de Culture informatique https://www.culture-informatique.net/cest-quoi-langage-de-programmation/. Consulté le 07/08/2018

[17] : Site de Wikipédia https://fr.m.wikipedia.org/wiki/Langage_de_balisage. Consulté le 07/08/2018

[18] : Site du journal du net http://www.journaldunet.com/developpeur/algo-methodes/analyse/le-framework-des-composants-a-assembler.shtml. Consulté le 07/08/2018

[19] : Site de Algocool https://algocool.fr/bootstrap/. Consulté le 07/08/2018

[20] : Site de Edutechwiki http://edutechwiki.unige.ch/fr/JQuery. Consulté le 07/08/2018

[21] : Site du Standard du web http://www.standard-du-web.com/xampp.php. Consulté le 08/08/2018

[22] : Site de Piloter https://www.piloter.org/techno/support/base-de-donnees-relationnelle-definition.htm.Consulté le 08/08/2018

[23] : Site de Irif https://www.irif.fr/~carton/Enseignement/InterfacesGraphiques/MasterInfo/Cours/Swing/mvc.html. Consulté le 20/09/2018

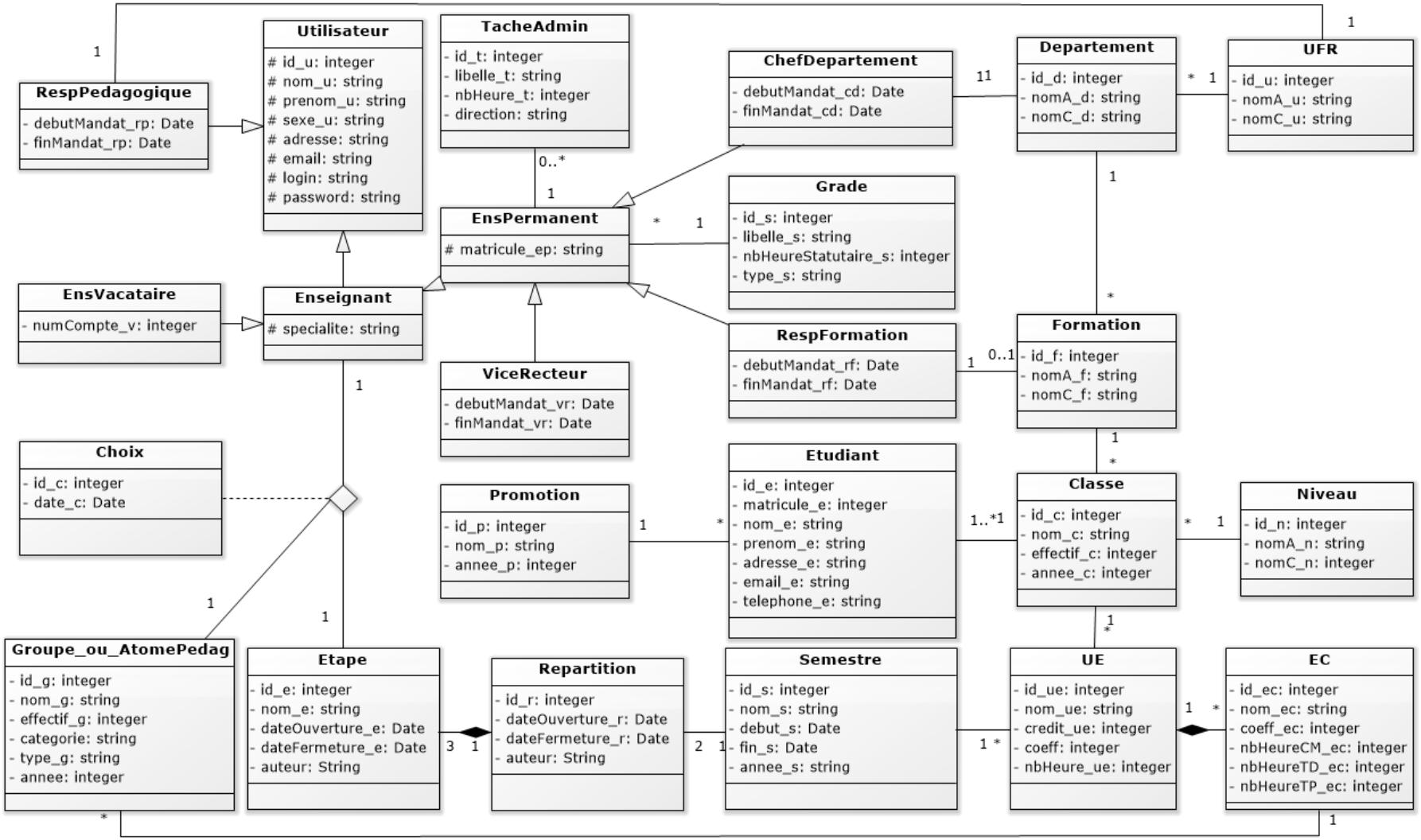
[24] : Site de Sparx Systems https://www.sparxsystems.fr/resources/uml2_tutorial/uml2_componentdiagram.html. Consulté le 12/10/2018

ANNEXE

Annexe 1 : Diagramme de classes

Sur le diagramme de classe, ci-dessous, un **Utilisateur** peut être un **RespPedagogique** (Responsable pédagogique) qui opère dans une **UFR** ou un **Enseignant**. L'Enseignant peut être un **EnsVacataire** (Enseignant vacataire) ou un **EnsPermanent** (Enseignant permanent) qui a un **Grade** et qui peut exécuter des **TacheAdmin** (Tâches administratives). L'Enseignant permanent peut être un **ChefDepartement** (Chef de département) qui gère un **Département** appartenant à une UFR, il peut être un **RespFormation** (Responsable de formation) qui gère une **Formation** appartenant à un département, il peut enfin être le **ViceRecteur** (Vice-recteur). L'enseignant fait des **Choix** sur des **Groupe_ou_AtomePedag** (groupe, atome pédagogique ou enseignement) lors d'une **Etape** appartenant à une **Répartition** qui, à son tour, appartient à un **Semestre**. Un enseignement est contenu dans un **EC** qui est contenu dans une **UE**. Chaque UE appartient à une **Classe** qui appartient à une formation. Chaque classe a un **Niveau** et contient des **Etudiants** qui appartiennent à une **Promotion**.

La figure suivante représente notre diagramme de classes.


Utilisateur
id_u: integer
nom_u: string
prenom_u: string
sexe_u: string
adresse: string
email: string
login: string
password: string
TacheAdmin
- id_t: integer
- libelle_t: string
- nbHeure_t: integer
- direction: string
RespPedagogique
- debutMandat_rp: Date
- finMandat_rp: Date
ChefDepartement
- debutMandat_cd: Date
- finMandat_cd: Date
Departement
- id_d: integer
- nomA_d: string
- nomC_d: string
UFR
- id_u: integer
- nomA_u: string
- nomC_u: string
Grade
- id_s: integer
- libelle_s: string
- nbHeureStatutaire_s: integer
- type_s: string
EnsPermanent
matricule_ep: string
EnsVacataire
- numCompte_v: integer
Enseignant
specialite: string
RespFormation
- debutMandat_rf: Date
- finMandat_rf: Date
Formation
- id_f: integer
- nomA_f: string
- nomC_f: string
ViceRecteur
- debutMandat_vr: Date
- finMandat_vr: Date
Choix
- id_c: integer
- date_c: Date
Promotion
- id_p: integer
- nom_p: string
- annee_p: integer
Etudiant
- id_e: integer
- matricule_e: integer
- nom_e: string
- prenom_e: string
- adresse_e: string
- email_e: string
- telephone_e: string
Classe
- id_c: integer
- nom_c: string
- effectif_c: integer
- annee_c: integer
Niveau
- id_n: integer
- nomA_n: string
- nomC_n: integer
Groupe_ou_AtomePedag
- id_g: integer
- nom_g: string
- effectif_g: integer
- categorie: string
- type_g: string
- annee: integer
Etape
- id_e: integer
- nom_e: string
- dateOuverture_e: Date
- dateFermeture_e: Date
- auteur: String
Repartition
- id_r: integer
- dateOuverture_r: Date
- dateFermeture_r: Date
- auteur: String
Semestre
- id_s: integer
- nom_s: string
- debut_s: Date
- fin_s: Date
- annee_s: string
UE
- id_ue: integer
- nom_ue: string
- credit_ue: integer
- coeff: integer
- nbHeure_ue: integer
EC
- id_ec: integer
- nom_ec: string
- coeff_ec: integer
- nbHeureCM_ec: integer
- nbHeureTD_ec: integer
- nbHeureTP_ec: integer

Annexe 2: Dictionnaire de données

Tables	Champs	Types	Légendes
Utilisateur	Id_u	Entier	Identifiant de l'utilisateur
	Nom_u	Chaine de caractères	Nom de l'utilisateur
	Prenom_u	Chaine de caractères	Prénom de l'utilisateur
	Sexe_u	Chaine de caractères	Sexe de l'utilisateur
	Adresse_u	Chaine de caractères	Adresse de l'utilisateur
	Email_u	Chaine de caractères	Email de l'utilisateur
	Login_u	Chaine de caractères	Pseudo de l'utilisateur
	Password_u	Chaine de caractères	Mot de passe de l'utilisateur
Esneignant	Spécialité_u	Chaine de caractères	Spécialité de l'enseignant
Enseignant permanent	Matricule_ep	Chaine de caractères	Matricule de l'enseignant
Enseignant vacataire	numCompte_v	Entier	Numéro de compte bancaire de l'enseignant
Chef de département	debutMandat_cd	Date	Date de début de mandat du chef de département
	finMandat_cd	Date	Date de fin de mandat du chef de département
Responsable de formation	debutMandat_rf	Date	Date de début de mandat du

			responsable de formation
	finMandat_rf	Date	Date de fin de mandat du responsable de formation
Vice-recteur	debutMandat_vr	Date	Date de début de mandat du vice-recteur
	finMandat_vr	Date	Date de fin de mandat du vice-recteur
Responsable pédagogique	debutMandat_rp	Date	Date de début de mandat du responsable pédagogique
	finMandat_rp	Date	Date de fin de mandat du responsable pédagogique
Semestre	Id_s	Entier	Identifiant du semestre
	Nom_s	Chaine de caratères	Libellé du semestre
	Debut_s	Date	Date de début du semestre
	Fin_s	Date	Date de fin du semestre
	Annee_s	Entier	Année du semestre
Répartition	Id_r	Entier	Identifiant de la répartition
	Debut_r	Date	Date de début de la répartition

	Fin_r	Date	Date de fin de la répartition
	Auteur_r	Chaine de caratères	Auteur de l'ouverture et de fermeture de la répartition
Etape	Id_e	Entier	Identifiant de de l'étape
	Debut_e	Date	Date de début de l'étape
	Fin_e	Date	Date de fin de de l'étape
	Auteur_e	Chaine de caratères	Auteur de l'ouverture et de fermeture de de l'étape
Enseignement	Id_e	Entier	Identifiant de l'enseignement
	Libellé_e	Chaine de caractères	Libellé de l'enseignement
	Effectif_e	Entier	Nombre d'étudiant dans ce groupe de CM, de TD ou TP
	Catégorie	Chaine de caractères	Catégorie de l'enseignement (informatique, par ex)
	Type_e	Chaine de caractères	Type de l'enseignement (CM, TD, ou TP)
Choix	Id_c	Entier	Identifiant du choix effectué par l'enseignant

	Date_c	Date	Date du choix effectué par l'enseignant
UFR	Id_u	Entier	Identifiant de l'UFR
	Nom_u	Chaine de caractères	Libellé de l'UFR
Département	Id_d	Entier	Identifiant du département
	Nom_d	Chaine de caractères	Libellé du département
Formation	Id_f	Entier	Identifiant de la formation
	Nom_f	Chaine de caractères	Libellé de la formation
Classe	Id_c	Entier	Identifiant de la classe
	Nom_c	Chaine de caractères	Libellé de la classe
	Effectif_c	Entier	Nombre d'étudiants de la classe
Niveau	Id_n	Entier	Identifiant du niveau
	Nom_n	Chaine de caractères	Libellé du niveau
UE	Id_ue	Entier	Identifiant de l'UE
	Nom_ue	Chaine de caractères	Nom de l'UE
	Credit_ue	Entier	Nombre de crédits de l'UE
	Coeff_ue	Entier	Coefficient de l'UE
	nbHeure_ue	Entier	Nombre de heure de l'UE
EC	Id_ec	Entier	Identifiant de l'EC
	Nom_ec	Chaine de caractères	Libellé de l'EC
	Coeff_ec	Entier	Coefficient de l'EC
	nbHeureCM_ec	Entier	Nombre de heure de CM de l'EC

	nbHeureTD_ec	Entier	Nombre de heure de TD de l'EC
	nbHeureTP_ec	Entier	Nombre de heure de TP de l'EC
Tâche administratives	Id_t	Entier	Identifiant de la tâche
	Libelle_t	Chaine de caractères	Nom de la tâche
	nbHeure_t	Entier	Nombre d'heure correspondant aux TD de la tâche
	Direction_t	Chaine de caractères	Lieu où est exécutée la tâche
Grade	Id_g	Entier	Identifiant du grade
	Libelle_g	Chaine de caractères	Libellé du grade
	nbHeureGrade	Entier	Nombe d'heure du grade
Etudiant	Id_e	Entier	Identifiant de l'étudiant
	Matricule_e	Entier	Matricule de l'étudiant
	Nom_e	Chaine de caractères	Nom de l'étudiant
	Prenom_e	Chaine de caractères	Prénom de l'étudiant
	Adresse_e	Chaine de caractères	Adresse de l'étudiant
	Email_e	Chaine de caractères	Email de l'étudiant
	Telephone_e	Chaine de caractères	Téléphone de l'étudiant
Promotion	Id_p	Entier	Identifiant de la promotion
	Nom_p	Chaine de caractères	Libellé de la promotion
	Annee_p	Chaine de caractères	Année de la promotion

Annexe 3 : Fichier Excel utilisé pour la gestion de la répartition des enseignements à l'UASZ.

Répartition des Unités d'Enseignement en Informatique

Classe	Effectif	Nbre de groupes	Semestre	Unité d'Enseignement	Crédit	Durée Cours	Enseignant	CM	Responsables TD	Responsables TP	Travaux Dirigés	Travaux Pratiques
L1GE (Gestion)	90	3	1	Initiation à l'informatique	3	36		0				36
			1									36
			1									36
			1									36
L2GE (Gestion)	60	2	3	Informatique (Excel avancée)	2	24		0				24
			3									24
L3GE (Economie Gestion)	60	2	6	Informatique 3 (Access)	2	24		0				24
L1AF (Agroforesterie)	90	3	1	Initiation à l'informatique	2	24		4				20
			1									20
L2AF (Agroforesterie)	60	2	3	Informatique 2 (Excel avancée)	3	36		12				24
			3									24
L3AF (Agroforesterie)	60	2	5	Informatique (BD Access)	2	24		0				24
												24
L1TO (Tourisme)	60	2	1	Initiation à l'informatique	2	24		0				24
			1									24
L2TO (Tourisme)	60	2	1	Informatique 2	2	24		0				24
			1									24
L3TO (Tourisme)	60	2	5	Informatique 3	2	24		0				24
Master 1 Tourisme	30	1	7	Création et administration de site web	3	36		12				24
L2GEO (Géographie)	60	2	3	Base de Données (Access)	2	24		12			0	12
			3								0	12
			3								0	12
			3								0	12