République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la
Recherche Scientifique

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Université Abderrahmane MIRA Béjaïa
Faculté Des Sciences Exactes
Département D'informatique

Mémoire de fin de cycle

En Vue de l'Obtention du Diplôme de Master Professionnel en Informatique

Option Génie Logiciel

Thème

Conception et Réalisation d'une Application Web pour la Gestion des Emplois Du Temps. Cas d'étude : département Informatique

Réalisé par :

Mr. ADJAOUT Sohaib
Mr. BENAMOKRANE Amine

Encadrante :

Mme. GASMI Badrina

Devant un jury composé de :

Dr BOUADEM Nassima
Dr SADI Mustapha

Année Universitaire 2019/2020

Remerciements

Nos premiers remerciements s'adressent à Dieu le tout puissant qui par sa bonté et sa miséricorde nous a permis d'avoir le courage, la foi et la Volonté de mener ce travail.

Nous tenons aussi à remercier Madame GASMI Badrina notre encadrante qui a été présenté à tout moment de la réalisation de ce projet et surtout sans lequel ce modeste travail n'aurait jamais vu le jour. Ainsi que les membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs critiques.

Nous remercions également tous les enseignants qui ont contribués de près ou de loin à notre formation universitaire, sans oublier toute personne qui nous a aidés à mener à terme notre projet.

Dédicace

A nos familles.

A nos amis.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

C.L.R.S. ..... *Course, Lecture, Room, Slot*

C.S.P ..... *Constraint Satisfaction Problem*

CJSP ..... *Cyclic Job-Shop Scheduling Problem*

CSS ..... *Cascading Style Sheets*

Dis C.S.P. ..... *C.S.P distribués*

EDT ..... *Emploi Du Temps*

F.C ..... *ForwardCheking*

G.C.P ..... *Graph Coloring Problem*

GBCSP ..... *General Basic Cyclic Scheduling Problem*

HTML ..... *Hyper Text Markup Language*

L.R.S ..... *Lecture, Room, Slot*

L.R.S.R.S ..... *Lecture, Room, Slot, Room, Slot*

L.R.S.R.S.F ..... *Lecture, Room, Slot, Room, Slot, Fonction*

O.E.P ..... *Optimisation par Essaims Particulaires*

P.L.N.E ..... *Programmation Linéaire en Nombres Entiers*

P.L.N.R ..... *Programmation Linéaire en Nombres Réels*

PHP ..... *Hypertext Preprocessor*

PPC ..... *Programmation Par Contraintes*

RWD ..... *Responsive Web Design*

SMA ..... *Systèmes Multi Agents*

SQL ..... *Structured Query Language*

TD ..... *Travaux Dirigés*

TP ..... *Travaux Pratiques*

UML ..... *Unified Modeling Language*

UP ..... *Unified Process*

W3C ..... *World Wide Web Consortium*

Introduction Générale

L'humanité a traversé trois révolutions industrielles jusqu'au nouveau millénaire, et nous vivons actuellement la quatrième révolution qui est l'air du numérique modifiant complètement le quotidien des gens sachant que toutes les dix minutes, il y'a de nouvelles innovations dans tous les domaines grâce à l'informatique.

En effet, l'informatique est devenue l'outil indispensable de gestion, puisqu'on parle de maisons et villes intelligentes mais aussi d'administrations, d'usines, d'hôpitaux et de transports, l'informatique a permis une meilleure optimisation de temps et de rendement en traitant les informations et les données grâce à des logiciels.

Au cours de la vie professionnelle, on se trouve confrontés au problème de la planification d´horaire de travail, l'élaboration d'un planning ou Emploi du temps, revient à résoudre un problème complexe, chaque institution (entreprises de production, hôpitaux, universités, etc.) ayant ses propres règles et critères.

La planification du temps consiste à affecter des ressources données (humaines, matérielles, etc.) à des intervalles de temps, pour satisfaire au mieux un ensemble d´objectifs tels que l´amélioration de la qualité de service et les conditions de travail, tout en respectant les contraintes imposées par les entités citées (disponibilité des ressources humaines, matérielles, etc.). Parmi les domaines d'application de la planification d´horaire, on trouve celui des établissements éducatifs, notamment dans les universités qui exploitent de nombreuses ressources humaines.

Chaque année, les responsables pédagogiques des universités ont pour mission d'organiser les emplois du temps des différentes formations ou filières en essayant, au mieux, de satisfaire les contraintes « humaines » des enseignants et des étudiants, les contraintes pédagogiques imposées par la progression des enseignements et en tenant compte des contraintes « physiques » liées aux ressources matérielles (les salles, les équipements, etc.).

La complexité de cette tâche requiert l'utilisation de l'outil informatique qui va permettre la réalisation d'un system qui devra apporter une solution optimale grâce au respect des contraintes, mais aussi un système qui sera bénéfique à toutes les parties impliquées.

A travers ce mémoire nous allons suivre une démarche qui va nous permettre la création d'une application semi-automatique qui permettra de générer des emplois du temps à partir de données préalablement renseignées (toutes les données concernant les séances, leurs horaires, les enseignants concernés, les modules, les groupes …etc.), l'application aura pour but l'automatisation des tâches manuelle. Nous avons pour cas d'étude le département

informatique. Pour ce faire, nous avons entreprit le processus d'analyse et de conception UP (Unified Process) avec le formalisme graphique UML, et pour le développement nous avons utilisé les outils et langages suivants : PHP (Hypertext Preprocessor), MySQL avec Wampserver, HTML (Hyper Text Markup Language) et CSS (Cascading Style Sheets) et JavaScript.

Pour une bonne élaboration nous avons structuré notre mémoire en trois chapitres :

Le premier chapitre commence par quelques notions fondamentales sur le problème d'ordonnancement pour ensuite évoquer le problème de l'emploi du temps universitaire et les approches liées à sa résolution.

Dans le deuxième chapitre sera une étude de l'existant, tout ce qui concerne la présentation de l'organisme d'accueil, la critique de l'existant et la solution que nous proposons.

Le troisième chapitre traite de la phase de conception de notre système.

Dans le quatrième chapitre on parle de la réalisation, en commençant par la définition de tous les langages et logiciels utilisés, et enfin l'illustration de quelques interfaces essentielles de l'application.

Enfin, une conclusion synthétise notre travail et présente les perspectives envisagées.

Chapitre 1 Etat de l’art

1.1 Introduction

Dans une Institution ou un établissement une importance particulière doit être accordée à l'emploi du temps car ce dernier permettra une bonne gestion du temps et des priorités. Cependant il peut aussi poser problème car sa réalisation à la main est une tâche draconienne qui peut mobiliser plusieurs personnes plusieurs jours de travail. Sans oublier, que toute modification des données du problème peut complètement remettre en cause la solution trouvée.

Dans ce chapitre, nous allons présenter quelques concepts de l'ordonnancement, parler du problème d'emplois du temps universitaire, et citer quelques travaux liés à la résolution de ce problème.

1.2 L'ordonnancement

Dans les systèmes organisés, en particulier ceux qui disposent d'un grand nombre de ressources, la planification des tâches est un mécanisme lourd, difficile et complexe; le processus de génération doit couvrir le traitement de plusieurs phases afin de satisfaire différentes contraintes liées à ces ressources. C'est pourquoi nous devons avoir recours à l'ordonnancement.

1.2.1 Définition de l'ordonnancement

Définition 1

Ordonnancer c'est programmer l'exécution d'une tâche en lui attribuant des ressources et en fixant leurs dates d'exécution. Les problèmes d'ordonnancement apparaissent dans tous les domaines de l'économie : l'informatique (tâches : jobs; ressources : processeurs ou mémoire...), la construction (suivi de projet), l'industrie (problèmes d'ateliers, gestion de production), l'administration (emplois du temps). La théorie de l'ordonnancement traite de modèles mathématiques mais aussi de la modélisation de situations réelles fort complexes; aussi le développement de modèles utiles ne peut être que le fruit de contacts entre la théorie et la pratique. **[14]**

Définition 2

Un problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelles (délais, contraintes d'enchaînement) et de contraintes portant sur la disponibilité des ressources requises. **[W1]**

En production (manufacturière, de biens, de service), on peut le présenter comme un problème où il faut réaliser le déclenchement et le contrôle de l'avancement d'un ensemble de commandes à travers les différents centres composant le système. **[W1]**

Un ordonnancement constitue une solution au problème d'ordonnancement. Il est défini par le planning d'exécution des tâches (« ordre » et « calendrier ») et d'allocation des ressources et vise à satisfaire un ou plusieurs objectifs. Un ordonnancement est très souvent représenté par un diagramme de Gantt**. [W1]**

Selon ces dernières définitions, on peut estimer que l'ordonnancement permet de planifier des tâches dans une unité de temps et un ordre précis en fonction de la disponibilité des ressources nécessaires à ce fait tout en respectant les contraintes liées aux tâches et aux ressources. Nous allons maintenant projeter ces notions un peu plus en détail.

1.2.2 Terminologie

1.2.2.1 Les ressources et les tâches [1]

Lorsqu'il y a un problème d'ordonnancement, deux concepts de base entrent en jeu: les tâches et les ressources. Une ressource peut être technique ou humaine, sa disponibilité limitée ou pas est connue à l'avance. Une tâche est un travail élémentaire dont la mise en œuvre nécessite un certain nombre d'unités de temps et de chaque ressource.

Pour résoudre le problème de planification, deux objectifs doivent être conciliés. Les aspects statiques incluent la génération de plans pour effectuer des travaux sur la base de données prévisionnelle. L'aspect dynamique inclut la prise de décisions en temps réel tout en tenant compte de l'état des ressources et de l'avancement des différentes tâches au fil du temps.

Les données d'un problème d'ordonnancement sont les tâches et leurs caractéristiques, les contraintes potentielles, les ressources, et la fonction économique. On note en général $I = \{1, 2, \ldots, n\}$ l'ensemble des tâches et P_i la durée de la tâche i si cette durée ne dépend pas des ressources qui lui sont allouées. Souvent une tâche i ne peut commencer son exécution avant une *date de disponibilité* notée r_i et doit être achevée avant une *date échue* d_i. Parfois, on a une durée de latence q_i qui minore la durée entre la fin de la tâche i et la fin de l'ordonnancement. Les dates réelles de début et de fin d'exécution de la tâche i sont notées respectivement t_i et C_i. Les tâches sont souvent liées entre elles par des *relations d'antériorité.* Si ce n'est pas le cas on dit qu'elles sont *indépendantes.* La contrainte d'antériorité la plus générale entre deux tâches i et j, appelée *contrainte potentielle*, s'écrit sous la forme $t_j - t_i >= a_{ij}$, elle permet d'exprimer la succession simple $(a_{ij} = p_i)$ et de nombreuses variantes**.**

1.2.2.2 Les contraintes [W1]

Les contraintes expriment des restrictions sur les valeurs que peuvent prendre simultanément les variables de décision. On distingue :

1. Des contraintes temporelles :

- Les contraintes de temps alloué, issues généralement d'impératifs de gestion et relatives aux dates limites des tâches (délais de livraison, disponibilité des approvisionnements) ou à la durée totale d'un projet ;
- Les contraintes de cohérence technologique, ou contraintes de gammes, qui décrivent des relations d'ordre entre les différentes tâches ;

2. Des contraintes de ressources :

- Les contraintes d'utilisation de ressources qui expriment la nature et la quantité des moyens utilisés par les tâches, ainsi que les caractéristiques d'utilisation de ces moyens;

- Les contraintes de disponibilité des ressources qui précisent la nature et la quantité des moyens disponibles au cours du temps. Toutes ces contraintes peuvent être formalisées sur la base des distances entre débuts de tâches ou potentiels**.**

1.2.3 Approches de résolution

Les méthodes de résolution des problèmes d'ordonnancement garantissent généralement l'optimalité de la solution fournie, ces dernières puisent dans toutes les techniques de l'optimisation combinatoire (programmation mathématique, programmation dynamique, procédures par séparation et évaluation, théorie des graphes...).

Les problèmes de grande taille ne peuvent pas être résolus par des algorithmes dont la complexité n'est pas polynomiale, souvent NP-difficiles, ces problèmes nécessitent la construction de méthodes de résolution approchées.

L'objectif ici est de présenter une typologie générale de construction de méthodes de résolution qui regroupe à la fois les méthodes exactes et les méthodes approchées.

1.2.3.1 Méthodes par construction progressive

Les méthodes de construction progressive sont des méthodes itératives dans lesquelles une solution partielle est complétée à chaque itération.

En ordonnancement, on peut construire de nombreuses méthodes itératives. Par exemple celles où chaque itération place successivement toutes les opérations d'un travail, celles où au contraire chaque itération place une opération au plus sur une des machines. Ces méthodes peuvent suivre le déroulement du temps ou non (on suit le déroulement du temps si la suite des instants de début des opérations que l'on place est non décroissante). Elles peuvent également construire des ordonnancements quelconques, semi-actifs, actifs ou sans-délai. Si le choix de l'opération à ajouter à l'ordonnancement est défini par l'application d'un théorème de dominance, la méthode peut fournir une solution optimale; c'est par exemple le cas pour le problème de base à une machine et la minimisation de la somme des temps de présence ou du plus grand retard (règles de Smith ou de Jackson **[1]**).

1.2.3.2 Méthodes par voisinage

Les méthodes par voisinage travaillent sur des solutions complètes, ce qui n'est pas le cas pour les méthodes par construction qui travaillent sur des solutions partielles. Chaque itération de la méthode par voisinage a pour objectif (pas toujours atteint) de passer d'une solution complète à une autre, meilleure par rapport au critère considéré.

Afin de pouvoir construire une méthode basée sur le voisinage, il est nécessaire d'avoir une solution initiale complète, une fonction f (x), qui fournit la valeur du critère de solution x, et une fonction de voisinage v (x), qui sélectionne une solution proche de x. Pour définir v (x), on peut par exemple inverser deux opérations en cas de problème de séquencement ou en cas de problème avec plusieurs machines, attribuer l'opération à une autre machine, éventuellement en sélectionnant au hasard l'opération et / ou la machine**[1]**.

1.2.3.3 Méthodes par décomposition

II existe de nombreuses façons de construire des méthodes de résolution par décomposition **[1] :**

- **La décomposition « hiérarchique »** consiste à décomposer les problèmes en plusieurs niveaux.
- **La décomposition « structurelle »** utilise la modélisation du problème et ses grandes familles d'inconnues, considère tout d'abord que les moyens sont illimités et résout le problème temporel. Les dates étant fixées, on cherche la meilleure affectation possible des moyens, puis on revient au problème temporel en ajoutant de nouvelles contraintes liées à l'utilisation des moyens...
- **La décomposition de « l'ensemble des solutions du problème »** conduit, si l'on veut obtenir la solution optimale, à des procédures par séparation et évaluation. Toutefois, ces méthodes exactes peuvent être transformées en heuristiques de différentes façons.
- **La décomposition « temporelle »**, dans le cas des ordonnancements dynamiques où les travaux arrivent à des dates différentes et dispersées dans le temps résout lors de la première itération un problème réduit en ignorant les travaux qui arrivent au-delà d'une date choisie, puis retient définitivement cet ordonnancement partiel jusqu'à une autre date déterminée. A l'itération suivante, on décalera ces deux dates de manière à construire la portion suivante de l'ordonnancement...
- **La décomposition « spatiale »** décompose l'atelier en sous-ateliers avec le moins possible de déplacements de produits entre les sous-ateliers grâce à des outils de technologie de groupe et construit une méthode itérative dans laquelle chaque itération ordonnance un sous-atelier**.**

1.2.3.4 Méthodes par modification des contraintes

Le modèle des problèmes que l'on a à résoudre est changé avec ces méthodes. Ce peut être, par exemple, de transformer un flow-shop normal en un flow-shop de permutation de manière à avoir moins de solutions à explorer, mais on trouve surtout ici toutes les méthodes dites de « relaxation » : relaxation de contraintes d'intégrité, relaxation lagrangienne ....

1.2.3.5 Méthodes liées à l'intelligence artificielle

II s'agit de méthodes qui utilisent des techniques de représentation des connaissances et de résolution de problèmes issues de l'intelligence artificielle.

De nombreux travaux ont abordé les problèmes d'ordonnancement à l'aide de ces outils (pour une synthèse, on peut consulter les travaux de A. KUSIAK et M. CHEN **[13]**). Citons par exemple la propagation par contrainte qui alliée à des règles de production permet de résoudre des problèmes d'ordonnancement, les systèmes experts et les systèmes à base de connaissances utilisant de l'apprentissage, comme par exemple la mémoire artificielle**.**

Parmi les méthodes qui cherchent à s'inspirer de phénomènes concrets, on peut également citer les algorithmes génétiques. Comme le recuit simulé, ce sont des algorithmes aléatoires, mais au lieu de chercher à améliorer progressivement une seule solution, ils font évoluer une

population de solutions qui s'améliore globalement par des techniques de reproduction, de sélection et de mutation. Ces méthodes commencent à être utilisées en ordonnancement.

Il est bien sûr possible de combiner ces techniques et de les croiser avec les autres méthodes présentées **[1]**.

1.2.4 Technique d'ordonnancement

La réalisation d'un projet nécessite souvent une succession de tâches auxquelles s'attachent certaines contraintes :

- **De temps** : délais à respecter pour l'exécution des tâches ;
- **D'antériorité** : certaines tâches doivent s'exécuter avant d'autres ;
- **De production** : temps d'occupation du matériel ou des hommes qui l'utilisent.

Les techniques d'ordonnancement dans le cadre de la gestion d'un projet ont pour objectif de répondre au mieux aux besoins exprimés par un client, au meilleur coût et dans les meilleurs délais, en tenant compte des différentes contraintes.

L'ordonnancement se déroule en trois étapes :

- **La planification** : qui vise à déterminer les différentes opérations à réaliser, les dates correspondantes, et les moyens matériels et humains à y affecter.
- **L'exécution :** qui consiste à la mise en œuvre des différentes opérations définies dans la phase de planification.
- **Le contrôle :** qui consiste à effectuer une comparaison entre planification et exécution, soit au niveau des coûts, soit au niveau des dates de réalisation.

1.2.5 La classe des problèmes d'ordonnancement

La diversité des problèmes d'Ordonnancement ne réside pas dans leur contexte : une même théorie peut étudier l'ordonnancement du fonctionnement d'un atelier ou l'établissement d'un emploi du temps scolaire, elle se trouve plutôt dans la nature des contraintes présentes sur le problème.

1.2.5.1 Problèmes centraux

Cela inclut l'ordonnancement des taches soumises à des contraintes de succession et de localisation temporelle et ce en une durée minimale. La représentation des contraintes est résumée par un graphe ***potentiels-tâches G= (X, U)*** où ***X*** est l'ensemble des tâches complété par une tâche fictive début notée ***0*** et une tâche fictive fin notée *******, et ***U*** est formé des arcs associés aux contraintes potentielles, l'arc ***(i,j)***value par c_{ij} correspondant à la contrainte potentielle $t_j - t_i \geq a_{ij}$. Un ordonnancement est un ensemble $T = \{t_i / i \in X\}$de dates de début des tâches telles que la date de début de la tâche fictive ***0*** soit zéro et que, pour tout arc ***(i,j)***, $t_j - t_i \geq a_{ij}$. Un tel ordonnancement existe si et seulement si ***G*** ne contient pas de circuit de valeur strictement positive**.**

1.2.5.2 Problèmes à ressource unique [1]

Le problème de base consiste à déterminer sur la machine un séquencement optimal de n tâches disponibles à l'instant 0. Pour minimiser les encours, c'est-à-dire $\sum w_i C_i$, il suffit de ranger les tâches dans l'ordre des P_i/w_i croissants (règle de Smith) si les tâches sont indépendantes. Si les tâches sont dépendantes, le problème est ***NP-difficile*** dans le cas général et polynomial pour une arborescence. Pour minimiser le plus grand retard, il suffit de ranger les tâches dans l'ordre des d_i croissants (règle de Jackson) et, si les tâches sont dépendantes de les ranger dans l'ordre des d'_i croissants où $d'_i = min\{d_j / j \text{ descendant de } i\}$. L'algorithme polynomial de Hodgson permet de minimiser le nombre de tâches en retard, il consiste à placer les tâches, une par une, dans l'ordre des dates échues croissantes, en rejetant la tâche déjà placée de plus grande durée à la fin de l'ordonnancement tant qu'un retard apparaît dans l'ensemble des tâches non rejetées. Le problème est ***NP-difficile*** pour la somme pondérée des retards, y compris lorsque tous les poids sont égaux**.**

1.2.5.3 Problèmes à ressources multiples

Les problèmes d'ordonnancement à contraintes de ressources renouvelables sont souvent appelés problèmes d'ateliers car on les rencontre dès qu'il faut gérer la production. Toutefois il s'agit de problèmes très généraux car ils apparaissent dans d'autres contextes, en particulier en Informatique.

Dans le cas des problèmes d'atelier, une tâche est une opération, une ressource est une machine et chaque opération nécessite pour sa réalisation une machine. Dans le modèle de base de l'ordonnancement d'atelier, l'atelier est constitué de m machines, n travaux (jobs) disponibles à la date 0 doivent être réalisés, un travail i est constitué de n_i opérations, l'opération j du travail i est notée (i, j) où $(i, 1) < (i, 2) < \ldots < (i, n_i)$ si le travail i possède une gamme ($A < B$ signifie A précède B). Une opération (i,j) utilise la machine $m_{i,j}$ pendant toute sa durée $p_{i,j}$ et ne peut être interrompue**[1].**

1.2.5.4 Problèmes d'ordonnancement cycliques

Les problèmes d'ordonnancement cycliques ont de nombreuses applications tant informatiques (calculs cycliques sur architectures parallèles) que dans le domaine de la production (job-shops cycliques) ou de la planification (affectation périodique de ressources). Jusqu'à maintenant, la plupart des travaux qui leur sont consacrés sont très proches de leurs applications respectives et concernent surtout des heuristiques permettant une résolution approchée rapide. Cependant, de plus en plus d'auteurs s'intéressent à l'étude théorique des problèmes sous-jacents, avec toutefois un partage regrettable des chercheurs selon le domaine d'application d'origine de leur problème.

1.2.5.4.1 Problème d'ordonnancement cyclique de base

Le problème d'ordonnancement cyclique de base (***GBCSP*** : General Basic Cyclic Scheduling Problem) est caractérisé par un ensemble de tâches élémentaires (ou génériques) $T = \{1, ..., n\}$ qui seront répétées une infinité de fois. Chaque tâche i a une durée p_i, on note $<i, k>$ la k^{eme} occurrence de i telle que $k \in \mathbb{N}$. Dans un ***GBCSP***, les tâches sont liées par des

contraintes de précédence dites ***uniformes***. Résoudre un ***GBCSP*** revient à trouver un ordonnancement cyclique σqui détermine pour chaque occurrence <***i, k***>, sa date de début ***t(i, k)*** tout en minimisant le temps de cycle asymptotique α **[2].**

1.2.5.4.2 Ordonnancement *K*-cyclique

Ce paragraphe est dédié à la distinction entre les ordonnancements 1-cycliques (ou 1-périodiques, ou encore strictement périodiques) et les ordonnancements *K*-cycliques (ou *K*-périodiques) avec $K > 1$ et $K \in \mathbb{N}$. Dans un ordonnancement 1-cyclique, tel que présenté dans la section précédente, la différence entre les dates de début de deux occurrences différentes d'une même tâche est égale au temps de cycle α, ainsi, dans un intervalle de temps de longueur α chaque tâche est traitée exactement une fois. En revanche, dans un ordonnancement *K*-cyclique la différence entre la date de début de l'occurrence <***i, n***>et la date de début de l'occurrence < ***i, (n + K)***>est égale au temps de cycle αK. Par conséquent, dans un intervalle de temps de longueur αKchaque tâche est traitée exactement ***K***fois. Autrement dit, dans un ordonnancement *K*-cyclique, l'ordonnancement de ***K*** occurrences successives de chaque travail est fixe et se répète à un intervalle régulier de longueur αK **[2].**

1.2.5.4.3 Problème d'ordonnancement cyclique à machines parallèles

Dans un problème de machines parallèles un ensemble des opérations (tâches) $\{O_i\}_{1 \le i \le n}$ est exécuté sur ***m*** processeurs identiques. Pour chaque exécution d'une opération ***i***, il est nécessaire d'utiliser un des ***m*** processeurs pendant p_i unités de temps. La préemption n'est pas autorisée. Les contraintes de ressources sont liées au nombre limité de processeurs et au nombre d'opérations qui sont exécutées à une période donnée. Un ensemble de contraintes de précédence est également à considérer. Un ordonnancement réalisable est un choix de dates de début $\{t_i\}_{1 \le i \le n}$, tel que les contraintes de ressources et de précédence soient respectées **[2].**

1.2.5.4.4 Problème Job-Shop Cyclique

Le problème de Job-Shop cyclique (***CJSP*** : Cyclic Job-Shop Scheduling Problem) est aussi un problème NP-difficile. Le nombre de machines dans un CJSP est inférieur au nombre de tâches génériques, mais ces machines ne sont pas identiques et il n'y a pas de problème d'affectation : toute tâche est pré affectée à une des machines, qu'elle doit donc partager avec toutes les autres tâches également affectées à cette machine. Par conséquent, les machines jouent un rôle important et les contraintes qu'elles génèrent sont ajoutées au CJSP. L'objectif de la résolution d'un CJSP est de trouver un ordonnancement périodique ayant un temps de cycle asymptotique minimal et vérifiant toutes les contraintes de précédence et de ressources **[2].**

1.2.5.4.5 Problèmes d'ordonnancement cyclique sous contraintes de ressources

Au-delà des problèmes d'atelier, où les ressources sont des machines ou des robots de type "disjonctif", c'est à dire qu'une ressource ne peut exécuter qu'une seule tâche à un instant donné, des travaux récents considèrent des problèmes impliquant des ressources plus

complexes, appelées ressources cumulatives ou discrètes selon les auteurs. Ces problèmes apparaissent par exemple dans le contexte de l'ordonnancement d'instruction au sein des compilateurs pour les architectures parallèles. Une ressource cumulative est définie par une capacité maximale instantanée (par exemple une taille mémoire) et chaque tâche demande éventuellement tout au long de son exécution un nombre limité mais possiblement supérieur à 1 d'unités de cette ressource. Ainsi à tout moment le nombre de tâches utilisant cette ressource exécutée en parallèle est limité par le fait que la somme des demandes des tâches ne doit pas excéder la capacité de la ressource **[3].**

1.3 Problème d'emploi du temps universitaire

Le problème d'emploi du temps consiste à l'affectation d'un ensemble de tâches et /ou activités à un ensemble de ressources en se basant sur des périodes de temps bien précises, soumettant à un ensemble de contraintes à respecter.

Le problème d'emploi du temps universitaire est une instance des problèmes d'ordonnancement cyclique les plus connues dans la littérature, il s'agit d'ordonnancer les tâches (qui possède un caractère cyclique) d'un ensemble d'enseignants en leur allouant un ensemble de salles et en leur fixant leurs dates de début et de fin **[3]**.

De nos jours, la construction d'un calendrier qui satisfait tous les besoins d'un établissement universitaire est une tâche très importante, mais elle est extrêmement difficile. Avec le progrès réalisé dans les technologies matérielles et logicielles, la communauté scientifique continue à travailler sur ce problème, afin d'élaborer des procédures formelles et automatisées pour élaborer des calendriers efficaces et souhaitables.

1.3.1 Problème d'emploi du temps des cours universitaires

Dans notre problématique, nous choisissons de traiter le problème d'emploi du temps consacrés au cours (cours, TD (travaux dirigés) ou TP (travaux pratiques)), qui se définit comme un ensemble de cours universitaire qui se déroulent tout au long des périodes spécifiques pendant cinq ou six jours par semaine, dirigés par un nombre limité de professeurs et de salles de classe nécessitant une meilleure gestion pour bien contenir le nombre important d'étudiants inscrits.

Le processus de génération d'emploi du temps universitaire pourrait être formellement défini avec une matrice binaire ***Mesgpjl*** qui prend la valeur ***1*** si l'enseignant ***e*** peut enseigner la séance ***s*** avec le groupe ***g*** dans la période ***p*** du jour ***j*** dans le local ***l.***

Et afin d'avoir une meilleure solution à ce problème, il faut prendre en compte l'ensemble des contraintes du problème qui doivent être satisfaites. Ces contraintes sont souvent classées en deux catégories, la première regroupe les contraintes Hard (dures) et la seconde catégorie regroupe des contraintes appelées souvent les contraintes Soft (molles) :

Les contraintes Hard : cette classe représente le type de contraintes que nous ne sommes pas autorisés à casser.

Les contraintes Soft : c'est l'ensemble des contraintes de préférence. En les libérant, la sortie est toujours valide mais pas optimale.

1.3.2 Les méthodes de résolution

De nombreuses approches ont été proposées dans la littérature pour résoudre ce problème. Nous regroupons ces méthodes en deux classes: une première classe contenant les méthodes centralisées et une seconde classe contenant des méthodes décentralisées.

1.3.2.1 Approches centralisées

Cette famille de recherche approchée est dotée de mécanismes généraux lui permettant une bonne investigation de l'espace de recherche, mais généralement, elle est non-déterministe et ne donne aucune garantie d'optimalité. Ce qui a permis l'apparition d'autres types de méthodes, à savoir les techniques d'hybridations des premières méthodes avec les méta-heuristiques.

1.3.2.1.1 Approches basées sur la programmation linéaire et la théorie des graphes

La programmation linéaire est un outil très puissant de la recherche opérationnelle. C'est un outil générique qui peut résoudre un grand nombre de problèmes. Une fois le problème est modélisé sous la forme d'équations linéaires, des méthodes assurent sa résolution de manière exacte**. [4]**

On distingue dans la programmation linéaire, la Programmation Linéaire en Nombres Réels (P.L.N.R), pour laquelle les variables des équations sont dans $\mathbb{R}+$ et la Programmation Linéaire en Nombres Entiers (P.L.N.E), pour laquelle les variables sont dans $\mathbb{N}$. Bien entendu, il est possible d'avoir les deux en même temps. La théorie des graphes est un domaine très riche contenant des modèles et des applications permettant de résoudre plusieurs types de problèmes difficiles. En effet, le Graph Coloring Problem (G.C.P)[1] est considéré comme une de ces fameuses applications les plus connue dans la littérature.

S.Daskalaki, T.Birbas et E. Housos **[4]** ont utilisé la P.L.N.E pour présenter une formulation du problème d'emploi du temps universitaire. Ils ont ajouté plusieurs fonctionnalités telles que les variables multidimensionnelles (intégrant plus de détails du problème dans le modèle) et une fonction de coût (permettant l'introduction de certaines préférences concernant les jours, les salles et les périodes de temps) donnant plus de flexibilité au système**.**

[1] **Problème de coloration de graphe (G.C.P)** : il s'agit d'affecter une couleur à chacun de ses sommets de sorte que deux sommets adjacents ne soient pas de même couleur (on peut de la même façon définir la coloration des arêtes mais ce problème se ramène très facilement à un problème de coloration des sommets en considérant le graphe de lignes correspondant). Une coloration optimale d'un graphe est une coloration utilisant le nombre minimum de couleurs (appelé nombre chromatique et noté ***X(G)***). Colorier un graphe d'une façon optimale est un problème NP-complet.

1.3.2.1.2 Approches basées sur les problèmes de satisfaction de contraintes

Les C.S.P (Constraint Satisfaction Problem) sont des problèmes mathématiques où l'on cherche des états ou des objets satisfaisant un certain nombre de contraintes ou de critères. Les C.S.P font l'objet de recherches intenses à la fois en intelligence artificielle et en recherche opérationnelle.

De nombreux C.S.P nécessitent la combinaison d'heuristiques et de méthodes d'optimisation combinatoire pour être résolus en un temps raisonnable. En profitant de ce type de formalisme, plusieurs chercheurs ont choisi de formuler le problème d'emploi du temps comme un problème de satisfaction de contraintes (C.S.P) **[3]**.

Citons Safaai Deris, Sigeru Omatu et Hiroshi Ohta **[5]** qui ont présenté une formulation pour ce problème reposant sur un arbre de recherche, dont les niveaux de l'arbre représentent les cours à enseigner et chaque niveau i a m fils qui correspond à m affectations possibles sachant que $\boldsymbol{1 \leq i \leq n}$.

De plus, ils ont choisi d'utiliser les techniques de propagation de contraintes, tel que l'adaptation de la procédure de ForwardCheking (F.C)[2] pour la résolution de ce problème. Cette procédure est assez proche du backtrack[3], elle affecte des valeurs aux variables au fur et à mesure, mais avant de choisir une valeur pour une variable $\boldsymbol{X_n}$, vérifie que cette affectation correspond aux contraintes et vérifie que les autres variables $\boldsymbol{X_i\ (i>n)}$ pourront être affectées. Si l'affectation de $\boldsymbol{X_i}$ne peut être faite, l'algorithme choisit une autre valeur pour $\boldsymbol{X_n}$ et si jamais il n'y a aucune solution, la procédure fera un retour en arrière sur $\boldsymbol{X_n-1}$ pour changer sa valeur.

1.3.2.1.3 Approches basées sur les méta-heuristiques

Les méta-heuristiques forment une famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficiles souvent issus des domaines de la recherche opérationnelle, de l'ingénierie ou de l'intelligence artificielle pour lesquels on ne connaît pas de méthodes exactes plus efficaces.

Une méta-heuristique est formellement défini comme un processus de génération itératif, qui permet de guider une heuristique en combinant intelligemment différents types de concepts afin d'avoir une meilleure exploration et exploitation de l'espace de recherche **[3]**.

---

[2] **ForwardChecking :** Après chaque instanciation, permet de réduire par filtrage les domaines des variables directement liées par une contrainte à la variable qui vient d'être instancié. Avant d'affecter une variable $\boldsymbol{X}$ à une valeur $\boldsymbol{V}$, on vérifie que toutes les autres variables non affectées et liées par au moins une contrainte à $\boldsymbol{X}$ possède au moins une valeur de leur domaine compatible avec $\boldsymbol{V}$.

[3] **Backtrack :** Il est un des principaux algorithmes les plus connus pour la résolution des problèmes de satisfaction de contraintes, il appartient à la catégorie des méthodes de recherches complètes. Cet algorithme permet un retour en arrière à chaque fois qu'il ne trouve pas une solution partielle. En effet, à chaque étape l'instanciation partielle courante est étendue en instanciant une nouvelle variable par une valeur compatible avec l'instanciation courante : $\boldsymbol{X_1, X_2, \ldots, X_{k-1}, X_k \rightarrow d_1, d_2, \ldots, d_{k-1}, d_k}$

Si aucune valeur dk∈Dk n'est compatible, alors il y a un retour en arrière (chronologique) sur la variable précédente $\boldsymbol{X_{k-1}}$ pour essayer une autre valeur $\boldsymbol{d'_{k-1} \in D_{k-1}}$.

Les méta-heuristiques sont souvent inspirées par des systèmes naturels, qu'ils soient pris en physique (cas du recuit simulé), en biologie de l'évolution (cas des algorithmes génétiques) ou encore en éthologie (cas des algorithmes de colonies de fourmis ou de l'optimisation par essaims particulaires).

En effet, cette famille d'algorithmes a été utilisée dans plusieurs travaux de recherche permettant de trouver des méthodes génériques pouvant optimiser une large gamme de problèmes difficiles, tel que celui de l'emploi du temps universitaire.

1.3.2.1.4. Approches basées sur l'hybridation des méthodes

Outre l'utilisation classique des méthodes complètes ou incomplètes, certains auteurs ont proposé d'autres techniques de résolution telles que l'hybridation et la combinaison entre ces deux familles de méthodes.

Citons les travaux de Wijaya et Ruli Manurung **[6]** qui ont essayé de combiner les algorithmes génétiques avec les méthodes de résolution des C.S.P pour la résolution du problème d'emploi du temps universitaire. Ils se sont basés sur la création de quatre types de phénotypes (R1, R2, R3, R4) dont :

- R1 est un phénotype de type C.L.R.S (Course, Lecture, Room, Slot),
- R2 est un génotype de type L.R.S (Lecture, Room, Slot),
- R3 est un génotype de type L.R.S.R.S.F (Lecture, Room, Slot, Room, Slot, Fonction),
- R4 est un génotype de type L.R.S.R.S (Lecture, Room, Slot, Room, Slot).

Ajoutant aussi, une fonction qui permet de mesurer le degré de satisfaction des contraintes et de pénaliser leurs violations.

Dans un deuxième cas, et dans la même année que Wijaya et Ruli Manurung, Ho Sheau Fen Irene, Safaai Deris et Siti Zaiton Mohd Hashim **[7]** ont présenté un autre modèle d'hybridation des algorithmes d'Optimisation par Essaims Particulaires (O.E.P) avec celui des C.S.P. En effet, ils ont essayé d'améliorer l'efficacité de son arbre de recherche, en introduisant quelques modifications inspirées du phénomène des O.E.P[4] à l'algorithme du backtrack.

Ajoutant une fonction de préférence qui permet de prendre en compte l'ensemble des salles et périodes les plus préférées pour chaque type de cours.

1.3.2.2 Approches décentralisées

Outre la première génération de chercheurs qui ont beaucoup travaillé sur des approches centralisées, nous avons eu l'apparition d'un autre comité de chercheurs, qui cherchent à réduire tout type d'exécution séquentielle, afin d'atteindre de nouvelles approches parallèles et fortement réparties.

[4] **L'optimisation par essaims particulaires (O.E.P ou P.S.O en anglais)** : une méta-heuristique d'optimisation qui s'inspire à l'origine du monde du vivant. Cette méthode d'optimisation se base sur la collaboration des individus entre eux. Elle a d'ailleurs des similarités avec les algorithmes de colonies de fourmis, qui s'appuient eux aussi sur le concept d'auto-organisation. Cette idée veut qu'un groupe d'individus peu intelligents peut posséder une organisation globale complexe.

1.3.2.2.1 Approches basées sur la satisfaction distribuée des contraintes appliquées dans les systèmes multi-agents

Un réseau de contraintes distribuées est un quintuplé ***(X, D, C, A, Ψ)*** où **[8]**:

- ***X, D*** et ***C*** sont définis comme précédemment dans un C.S.P.
- ***A= (A1, . . ., A_p)*** est un ensemble de p agents.
- ***Ψ : X→*** A est la fonction qui associe chaque variable de ***X*** à un agent de ***A***.

Les Dis C.S.P (C.S.P distribués) sont résolus par l'action collective et coordonnée des agents de **A**, chacun exécutant un processus de satisfaction de contraintes. Les agents communiquent par envoi de messages, avec le postulat suivant: "Un agent ne peut envoyer un message que s'il connaît l'adresse du destinataire" **[8]**.

Une solution d'un Dis C.S.P est une affectation de valeurs aux variables qui ne viole aucune contrainte (bien que la littérature des Dis C.S.P se concentre sur la satisfaction des contraintes inter-agents). Trouver une assignation de valeurs aux variables inter-agents peut être vu comme réaliser la cohérence ou la consistance d'un système multi-agents.

En profitant de ce type de formalisme, certains chercheurs ont choisi de formuler le problème d'emploi du temps universitaire comme un Dis C.S.P**.**

1.3.2.2.2 Approches basées sur les systèmes multi-agents

Le thème des Systèmes Multi Agents (SMA) est actuellement un champ de recherche très actif. Cette discipline est à la connexion de plusieurs domaines en particulier l'intelligence artificielle, les systèmes informatiques distribués et le génie logiciel. C'est une discipline qui s'intéresse aux comportements collectifs produits par les interactions de plusieurs entités autonomes et flexibles appelées agents **[9].**

1.3.2.2.2.1 Concept d'agent

Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et son environnement, qui, dans un univers multi-agents, peut communiquer avec d'autre agents, et dont le comportement est la conséquence de ses observations, de se connaissances et de ses interactions avec d'autre agents **[9]**.

Ses caractéristiques sont :

- ***Situé :*** l'agent est capable d'agir sur son environnement à partir des entrées qu'il reçoit de celui-ci (systèmes de contrôle de processus, systèmes embarqués, etc.);
- ***Autonome :*** l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne;
- ***Flexibles :*** l'agent est :

1. capable de répondre à temps : pour cela l'agent doit être capable de percevoir son environnement et d'élaborer une réponse requise dans le temps.

2. proactif : l'agent doit présenter un comportement opportuniste tout en étant capable de prendre l'initiative au bon moment.

3. social : l'agent doit être capable d'interagir avec les autres agents (logiciels et humains) quand la situation l'exige afin d'accomplir ses propres tâches ou aider les autres agents à accomplir leurs tâches.

Si un agent est capable de réaliser des actions flexibles et autonomes pour atteindre les objectifs qui lui ont été fixé on parle alors d'agents **"intelligents".**

1.3.2.2.2.2 Système multi agent

Il existe plusieurs définitions pour les (SMA). Une définition possible est la suivante : un système multi agents est un système distribué composé d'un ensemble organisé d'agents.

D'une manière générale on peut définir un SMA comme tout système composé d'un environnement, d'un ensemble d'objets pouvant associer une position dans l'environnement, d'un ensemble d'agents, de relations ou de contraintes qui unit ces agents et/ou objets entre eux et des opérateurs représentant la façon dont sont effectuées les actions et effets sur l'environnement. Les principales caractéristiques des SMA sont **[9]** :

- Chaque agent ne dispose que d'informations incomplètes et à un champ d'action limité ;
- Le contrôle du système est réparti (il n'y a aucun contrôle global du système) ;
- Les données manipulées sont décentralisées ;
- Les traitements sont asynchrones.

Un système multi-agent peut être homogène ou hétérogène **[9]** :

- **Homogène** : Au niveau d'un système multi agent homogènes tous les agents ont la même structure interne, ceci incluant les objectifs, les connaissances des domaines, et les actions possibles. Ces agents choisissent leurs prochaines actions en utilisant la même procédure.
- **Hétérogène** : les agents peuvent être hétérogènes en ayant des objectifs différents, des modèles de domaines et des actions différentes.

1.4 Programmation par contraintes

La Programmation Par Contraintes (PPC, ou CP pour Constraint Programming en anglais) est un paradigme de programmation apparu dans les années 1980 permettant de résoudre des problèmes combinatoires de grande taille tels que les problèmes de planification et d'ordonnancement. En programmation par contraintes, on sépare la partie modélisation à l'aide de problèmes de satisfaction de contraintes (ou CSP pour Constraint Satisfaction Problem), de la partie résolution dont la particularité réside dans l'utilisation active des

contraintes du problème pour réduire la taille de l'espace des solutions à parcourir (On parle de propagation de contraintes) **[W3].**

1.4.1 Problème de satisfaction des contraintes

Un problème de satisfaction de contraintes (CSP) est défini formellement par un triple **(X, D, C)** Représentant :

- Un ensemble fini de variables **X**.
- Une fonction **D** affectant à chaque variable **x ∈ X** son domaine **D(x).**
- Un ensemble fini de contraintes **C**.

Le domaine initial d'une variable $D_0(x)$ représente l'ensemble de valeurs auxquelles la variable **x** peut être instanciée avant le début de la résolution. Chaque contrainte $c \in C$ est une relation multidirectionnelle portant sur un sous-ensemble de variables noté $var(c) \subseteq X$ restreignant les valeurs que ces variables peuvent prendre simultanément. Les problèmes traités dans ce mémoire sont des CSP discrets **($D(x) \in Z$)** et statiques, c'est-à-dire l'ensemble des contraintes ne change pas au cours de la résolution.
Donc un problème de satisfaction de contraintes (CSP) est un triplet $(X, D,)$ avec :

- X un ensemble $\{X1,\ldots\ldots, Xk\}$ de variables ;
- D un ensemble $\{D1,\ldots\ldots, Dk\}$ de domaines : $Xi \in Di$, i=1…,;
- C un ensemble $\{C1,\ldots\ldots, Ct\}$ de contraintes où chaque Ci définit un sous-ensemble du produit cartésien des domaines des variables sur lesquelles elle porte :

  $(Xi1,\ldots\ldots,) \subseteq Di1 \times \ldots \times Dik$

La notion de domaine désigne l'ensemble de valeurs que peut potentiellement prendre une variable. Cette notion générique fait référence à des ensembles de natures potentiellement très différentes :

- Un ensemble de valeurs symboliques : on peut vouloir représenter des couleurs, e.g. **D= {rouge, bleu, vert}**, ou encore des jours de la semaine, par exemple $D = \{lundi, mercredi, samedi\}$;
- Un ensemble d'entiers non contigus : il est possible d'utiliser un ensemble d'entiers quelconques. $D = \{5,8,12\}$;
- Un intervalle d'entiers : on peut également définir un ensemble d'entiers contigus, en compréhension, ne spécifiant que les bornes inférieure et supérieure de l'intervalle, par exemple $D = \{1,...,10\}$;
- Un intervalle de réels : de la même façon, on peut définir en compréhension un ensemble de réels, en ne déclarant que ses bornes inférieure et supérieure. $D = [-\pi, \pi]$.

Chaque variable possède ainsi un domaine initial qui contraint la valeur qu'on peut lui affecter. Si sa valeur est à chercher dans N, dans Z ou dans tout ensemble discret de valeurs,

on parlera de variable discrète. Si sa valeur est à trouver dans R ou dans tout sous-ensemble continu de valeurs, on parlera de variable continue. En fonction du type des variables du problème, on pourra donc distinguer trois grandes catégories de problèmes de satisfaction de contraintes : selon qu'il ne contient que des variables discrètes, que des variables continues, ou bien à la fois des variables discrètes et continues, on dira d'un CSP qu'il est discret, continu ou mixte.

1.4 Conclusion

Au niveau de ce premier chapitre, Nous avons vu le problème d'ordonnancement, sa définition, ses approches de résolution, quelques techniques d'ordonnancement ainsi quelques différentes classes de problèmes d'ordonnancement. Nous avons fait une description du problème d'emploi du temps universitaire. Nous avons également, détaillé les diverses méthodes de résolution d'emplois du temps tel que la satisfaction des contraintes pour résoudre le problème de l'horaire universitaire. Le progrès des solutions CSP ne s'est pas arrêté, Il existe de plus en plus d'outils qui prennent en charge cette résolution, ils fournissent de multiples stratégies et réduisent considérablement les efforts de programmation.

Le problème de planification des emplois du temps est considéré comme étant le problème académique le plus contraignant dans le système de gestion universitaire, Dans ce chapitre nous avons essayé de déterminer quelques contraintes, et enfin de citer une modélisation du problème des emplois du temps, basées sur la notion de programmation par contrainte.

Chapitre 2 Etude de l’existant

2.1 Introduction

La critique de l'existant constitue une étape utile, ayant pour but de porter un jugement objectif afin de déclarer les insuffisances éventuelles rencontrées au cours de l'étude de l'existant en vue de proposer un système plus fiable que l'ancien système.

Dans ce chapitre nous présenterons l'organisme d'accueil et sa gestion quotidienne, en plus de l'étude et critique de l'existant en ce qui concerne la gestion des emplois du temps, en exposant la problématique et les objectifs de notre projet.

2.2 Présentation de l'organisme d'accueil : "département d'Informatique"

Le département d'informatique de l'UAMB avait assuré, depuis sa création en fin de l'année 2002, trois types de formation, à savoir la formation du système classique des deux cycles de la graduation, le cycle court (2003-2006) en vue de l'obtention du diplôme de DEUA en informatique et le cycle long (2003-2012) en vue de l'obtention du diplôme d'ingénieur d'état en informatique. A partir de l'année 2003 le département d'informatique a commencé à prendre en charge la formation du système L.M.D (Licence, Master et Doctorat) jusqu'à l'heure actuelle et qui est le seul système fonctionnel actuellement.

2.2.3 Missions du département

Le département assure un suivi pédagogique des cycles de graduation et de post-graduation ; gérer la scolarité des étudiants (inscription, évaluation, présence aux enseignements, absences et sanctions) et des enseignants (matières enseignées, affectation des modules, volume horaire, emplois du temps, planning des examens, gestion des soutenances, absences, saisie des notes, délibérations . . .).

Les niveaux d'étude ouverts au sein du département sont :

- Licence Académique Système informatiques;
- Master Professionnalisant Administration et Sécurité des Réseaux;
- Master Professionnalisant Génie Logiciel;
- Master Recherche Intelligence Artificielle
- Master Recherche Réseaux et Système distribué
- Doctorat Intelligence Artificielle et Génie Logiciel;
- Doctorat Réseaux et Système Distribué.

2.3 Problématique

La problématique de notre département est la planification des emplois du temps, pour chaque niveau d'étude, les étudiants sont rassemblés en formations et chaque formation en sections et chaque section en groupes. Ces derniers ont un planning hebdomadaire, ou chaque module est assuré sous forme de séance de cours, TP ou TD dans des locaux répartis au début de chaque semestre.

2.3.1 Description du problème à résoudre

Dans un établissement éducatif, un ensemble d'étudiants groupés sous une structure hiérarchique (filières, promotions, sections, groupes, . . .) sont censés avoir un ensemble d'enseignements qui se répètent périodiquement, chacun de ces enseignements s'étend sur une durée de temps, dont l'unité élémentaire est la période.

Résoudre le problème de l'emploi du temps, revient à affecter à chacun de ces enseignements un nombre de périodes consécutives égal à la durée exigée, un local dont le type et la capacité sont convenables, et un enseignant apte à assurer le module concerné par l'enseignement de façon à prévenir les conflits sur les enseignants, les étudiants et sur les locaux.

Dans notre cas, le problème de l'emploi du temps étudié est celui du département d'informatique où les responsables pédagogiques ont besoin chaque année d'établir une nouvelle planification des différentes promotions en essayant au mieux de satisfaire les contraintes " humaines " des enseignants et des étudiants, les contraintes pédagogiques imposées par la progression des enseignements et en tenant compte des contraintes " physiques " liées aux ressources matérielles (les locaux).

Le département d'informatique regroupe différentes formations qui ont une durée qui varie entre trois ans (licence) et cinq ans (master).

Le programme pédagogique d'emplois du temps de chaque formation est connu à priori, ce programme précise les modules à suivre, leurs volumes horaires et quelques informations pédagogiques (répartition des cours, travaux dirigés, travaux pratiques etc.…).

Selon les besoins pédagogiques et les conditions physiques des ressources, chaque formation est structurée en promotions, en sections, et en groupes. En résumé, les données du problème à résoudre sont constituées par :

- Un ensemble de créneaux horaires étalés sur une semaine de cinq jours, du samedi au mercredi ou du dimanche au jeudi avec un nombre six périodes. La durée d'une période est d'une heure trente minutes pour les emplois du temps.
- Un ensemble de groupes d'étudiants.
- Un ensemble de cours, TD ou TP à programmer dans la semaine.
- Un ensemble de locaux (salles, amphis).

2.3.2 Objectifs

Nous avons décidé de concevoir une application Web adaptative qui va gérer les activités du département informatique et qui va permettre par la suite de minimiser le support papier et d'améliorer la rapidité de l'accès à l'information. Et pour cela nous avons assigné à notre étude les objectifs suivants :

- Automatisation des tâches manuelles (générer les emplois du temps; gérer les formations, gérer les groupes), ...

- Sécuriser l'accès aux informations par une authentification.
- Faciliter et rendre plus rapide la recherche et l'accès aux informations ainsi que leur mise à jour.
- Proposer des interfaces simples et faciles à utiliser.
- Implémentation d'une base de données complète pour gérer les utilisateurs, la liste des modules, ...

2.4 Etude de l'existant

L'étude de l'existant est une phase importante pour bien comprendre le système actuel et définir ses objectifs. Pour chaque module, il sera question d'effectuer une description précise de l'existant en énumérant les principaux acteurs impliqués, les principales activités effectuées et les moyens de traitement utilisés.

2.4.1 Les activités

Avant d'aborder cette étape importante, il s'avère nécessaire de définir le périmètre de notre étude.

Dans le cadre de ce projet, les principaux modules étudiés sont :

- Gérer les étudiants;
- Gérer les enseignants;
- Gérer les formations;
- Gérer les semestres;
- Gérer les sections;
- Gérer les groupes;
- Gérer les modules;
- Gérer les salles;
- Gérer les séances;
- Gérer l'affichage de l'emploi du temps.

2.4.2 Etude des postes de travail

Le gestionnaire: c'est le seul poste de travail, le gestionnaire peut ajouter, modifier, et supprimer un utilisateur de l'application, et peut accéder à tous les gestions de l'application.

2.5 Critique de l'existant

Les logiciels EXCEL et WORD de Microsoft Corporation, bien qu'ils soient puissants ne permettent pas de satisfaire tous les besoins du gestionnaire et des secrétaires. En effet, la gestion des formations, des semestres, des sections, des salles, des modules, des groupes, des enseignants, des étudiants et l'affectation des modules aux enseignants ne se fait pas de manière automatique et rapide. La création des emplois du temps est une tâche très difficile car le gestionnaire fait face à beaucoup de problèmes tel que la prise en charge du créneau des

enseignants, du temps libre des étudiants ainsi que le chevauchement des séances (salles, enseignants …), il nécessite une intervention manuelle qui est la plupart du temps fastidieuse.

L'objectif visé est de minimiser le travail du gestionnaire en réduisant la charge due aux différents traitements habituellement effectués. Le but est de concevoir une application avec le plus de fonctionnalités possibles mais aussi une interface ergonomique afin d'offrir du confort et de l'efficacité aux futurs utilisateurs.

2.6 Proposition de la solution sur le champ d'étude exercée

Pour résoudre ces problèmes, nous allons développer une application Web adaptative pour améliorer la gestion du département et la création des emplois du temps, cette application permettra une meilleure génération des emplois du temps sans aucun chevauchement ainsi elle permettra de faciliter le travail et de gagner plus de temps.

L'application doit garantir :

- Une meilleure gestion des formations; semestres; groupes, enseignants ….
- Une meilleure génération des emplois du temps à partir des données renseignées préalablement en respectant les contraintes physiques et pédagogiques.

2.6.1 Les contraintes

Pour la bonne réalisation de notre système nous nous sommes basés sur les contraintes fondamentales pour la création d'un emploi du temps.

2.6.1.1 Les contraintes physiques

Ces contraintes ne doivent pas être violées sinon cela conduirait à des situations conflictuelles, par exemple :

- Une ressource ne peut pas être occupée en même temps par plus d'un groupe ou d'une section.
- La capacité maximale d'un local doit être supérieure ou égale au nombre d'étudiant dans un groupe pour les TD et TP ou dans une section pour les cours.
- Le volume horaire total des séances d'un enseignement ne peut pas dépasser le volume prévu.

2.6.1.2 Les contraintes pédagogiques

Des contraintes pédagogiques ont pu être identifiées, typiquement ces contraintes sont utilisées pour exprimer ce que doit être un " bon " planning. Voici quelques exemples de ces contraintes :

- Un enseignant ne peut pas assurer deux enseignements en même temps.
- Pendant un créneau donné, un local ne peut pas être utilisé que par un seul enseignement.
- Si un TD ou un TP est affecté à un groupe donné, on ne peut pas affecter un cours à

la section à laquelle appartient ce groupe et vice versa.

- Un TP n'aura pas lieu dans un amphithéâtre ou dans une salle TD.
- Un TD n'aura pas lieu dans un amphithéâtre ou dans une salle TP.
- Un cours n'aura pas lieu dans une salle TP (Les cours en général doivent avoir lieux dans des amphithéâtres).

2.7 Conclusion

Durant l'analyse de l'existant nous avons pu recenser toutes les informations nécessaires et indispensables pour l'accomplissement de notre projet. Ces informations tirées entre autre à partir de l'étude du système pédagogique nous aident énormément à entamer notre travail concernant le chapitre suivant sur l'analyse et la conception de notre système en utilisant le langage UML (Unified Modeling Language).

Chapitre 3 Analyse et conception

3.1 Introduction

Avant de développer un système d'information, il est nécessaire de choisir une bonne méthode bien définie afin de bien organiser le travail à effectuer en citant les différentes tâches et les différentes étapes. Nous avons opté pour le processus UP qui est centré sur l'architecture UML. Dans ce chapitre nous allons identifier les différents acteurs et leurs tâches, puis élaborer les diagrammes de cas utilisation pour spécifier le besoin de notre système, en suite les diagrammes d'activités et les diagrammes de séquence pour l'analyse ; et enfin le diagramme de classes pour la conception.

Cette démarche peut être représentée graphiquement comme suit :

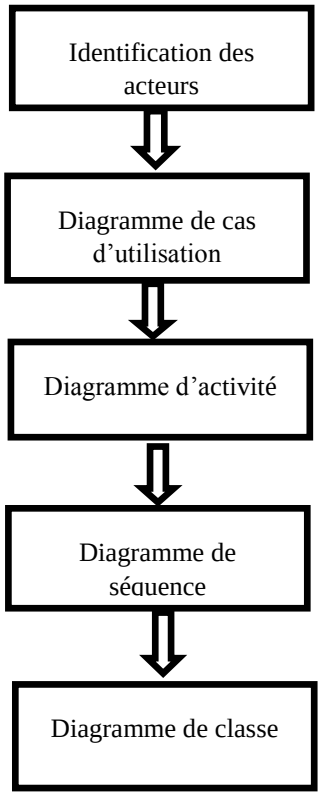

FIGURE 1 LA DEMARCHE DE MODELISATION DE L'APPLICATION.

Processus Unifié (UP)

Pour la réalisation de notre application, notre choix a été porté sur le Processus Unifié (UP). En effet, le processus unifié est une solution de développement logiciel adapté à tout type de projet, cette méthode est caractérisée par les trois notions suivantes :

- Centre sur l'architecture : l'architecture peut être considérée comme l'ensemble de vues du système qui vont provenir des besoins de l'entreprise et des différents intervenants.
- Pilote par les cas d'utilisations : le modèle des cas d'utilisations guide le processus unifié et d'écrit les fonctionnalités du système.
- Itératif et incrémental : les itérations se succèdent dans un ordre logique permettant de donner lieu à un incrément et donc d'établir un développement plus optimisé et efficace.

- Piloté par les risques : les causes majeures d'échec d'un projet logiciel doivent être écartées en priorité. **[11]**

Présentation d'UML

UML *(Unified Modeling Language)* se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue**. [10]**

3.2 Expression des besoins

L'analyse de ce projet nous a permis l'identification des différents besoins auxquels doit répondre notre application. Ces besoins dégagés sont classés en deux catégories, à savoir les besoins fonctionnels et les besoins non fonctionnels.

3.2.1 Besoins fonctionnels

Dans cette partie nous allons énumérer les différents besoins que doit assurer le système à travers les tâches suivantes :

- Gérer les utilisateurs de l'application : le système doit assurer la gestion des utilisateurs par l'ajoute, la suppression et la modification, et il doit permettre aux utilisateurs de se connecter avec un mot de passe et login.
  -Lors de l'ajoute, le système doit associer un login et un mot de passe à l'utilisateur ajouté.
  -Lors de la suppression d'un utilisateur, la suppression de celui-ci et son mot de passe et login doit s'effectuer automatiquement dans la base de données.
  -Lors d'une modification, une mise à jouer sera effectué automatiquement sur la base de données.
- Gérer les données : Le système doit ajouter modifier ou supprimer une donnée, les données de notre système sont : les formations, les semestres, les sections, les groupes, les modules, les salles et les séances
  -Il doit aussi affecter les modules aux enseignants.
- Générer les emplois du temps : elle concerne la création des emplois du temps relatifs aux groupes, aux enseignants et aux sections sans chevauchement.
  -le gestionnaire peut modifier l'emploi du temps, toujours le respect des contraintes sera automatique.

3.2.2 Besoins non fonctionnels

Une fois les besoins fonctionnels sont bien définis, les besoins non fonctionnels doivent être pris en compte tout au long du processus de développement de l'application à savoir :

- Authentification : chaque utilisateur doit s'authentifier par un mot de passe et un login unique pour accéder à l'application.
- La rapidité de traitement : l'application doit optimiser les traitements pour avoir un temps de génération raisonnable et que la durée d'exécution des traitements s'approche le plus possible du temps réel.

- La sécurité et la confidentialité : nous devons garantir une sécurité optimale. Ainsi, les droits d'accès au système doivent être bien attribués, afin d'assurer la sécurité des données.
- La performance : un logiciel doit être avant tout performant c'est à-dire à travers ses fonctionnalités, il doit répondre à toutes les exigences des usagers d'une manière optimale.
- La convivialité : le futur logiciel doit être facile à utiliser. En effet, les interfaces utilisateurs doivent être conviviales c'est-à-dire simples, ergonomiques et adaptées à l'utilisateur.

3.3 Analyse des besoins

Une fois ce premier recueil des besoins effectué, la description du contexte du système peut commencer. Elle consiste les activités suivantes : identification des acteurs, et identification des cas d'utilisations.

3.3.1 Identification des acteurs

Les acteurs sont les utilisateurs directs de l'application. Un acteur représente une entité externe (une personne, un dispositif matériel ou un autre système) qui interagit directement avec le système étudié, il peut consulter et/ou modifier l'état du système en émettant ou en recevant des messages susceptibles d'être porteurs de données **[10]**. Pour notre système, les différents acteurs sont :

- Gestionnaire *:* c'est l'administrateur du système représentant l'acteur essentiel ayant tous les privilèges au niveau de la gestion et de l'administration dans l'application.
- Enseignant : En plus de pouvoir consulter et imprimer son EDT il a la possibilité de consulter sa disponibilité ;
- Etudiant : c'est la personne qui consomme le fruit de cette application, il consulte et imprime son emploi du temps (EDT).

La figure suivante présente le tableau des différente **cas d'utilisation** de notre application:

Acteur(s)	**Cas d'utilisation**
Utilisateur	• S'authentifier ; • Consulter EDT ; • Imprimer EDT ; • Modifier LOGIN.
Enseignant / Gestionnaire	• Consulter disponibilité de l'enseignant.

Gestionnaire	• Gérer formation ; • Gérer semestre ; • Gérer section ; • Gérer groupe ; • Gérer salle ; • Gérer module ; • Gérer enseignant ; • Affecter module / enseignant ; • Gérer étudiant ; • Gérer séance .

TABLEAU 1 IDENTIFICATION DES ACTEURS AVEC LES FONCTIONNALITES.

3.4 Diagramme des cas d'utilisations

Le diagramme des cas d'utilisations identifie les fonctionnalités fournies par le système, les utilisateurs qui interagissent avec le système (acteurs), et les interactions entre ces derniers **[11],** donc il permet de décrire ce que le système devra faire, sans spécifier comment le faire.

Un cas d'utilisation correspond à un certain nombre d'actions que le système devra exécuter en réponse à un besoin d'un acteur **[12].**

- ***La relation <<include>>*** : Une relation inclusion d'un cas d'utilisation A par rapport à un cas d'utilisation B, signifie qu'une instance de A contient le comportement décrit dans B, le cas d'utilisation A ne peut pas être utilisé seul.

- ***Relation <<extend>>*** : Une relation d'extension d'un cas d'utilisation A par rapport à un cas d'utilisation B, signifie qu'une instance de A peut être étendue par le comportement décrit dans B.

La figure suivant présente le digramme global des cas d'utilisation de notre application :

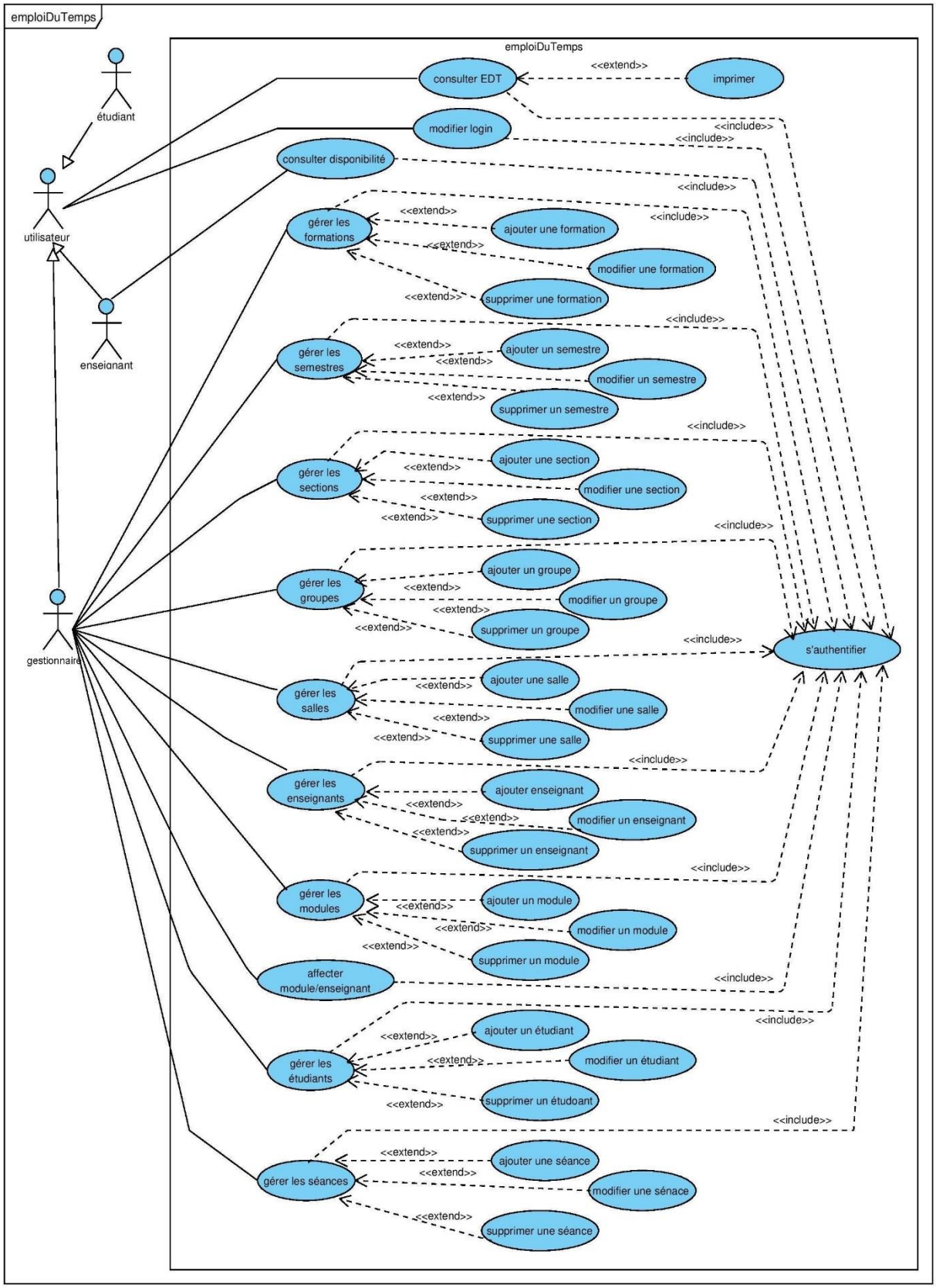

FIGURE 2 DIAGRAMME DE CAS D'UTILISATION « GLOBALE ».

A noté que, nous avons en l'occurrence 5 principaux cas d'utilisation représentant les activitées pouvant être effectuées par les utilisateurs : Ajouter, Supprimer, Consulter, Modifier et Imprimer un emplois du temps.

Le diagramme ci-dessus nous montre une vue globale de l'application mais pour voir réellement la succession des actions des acteurs il nous faut des descriptions textuelles des cas d'utilisation et puis un autre modèle qui nous détaillent le séquencement des opérations ils s'agissent du diagramme de séquences.

3.4.1 Cas d'utilisation « Authentification »

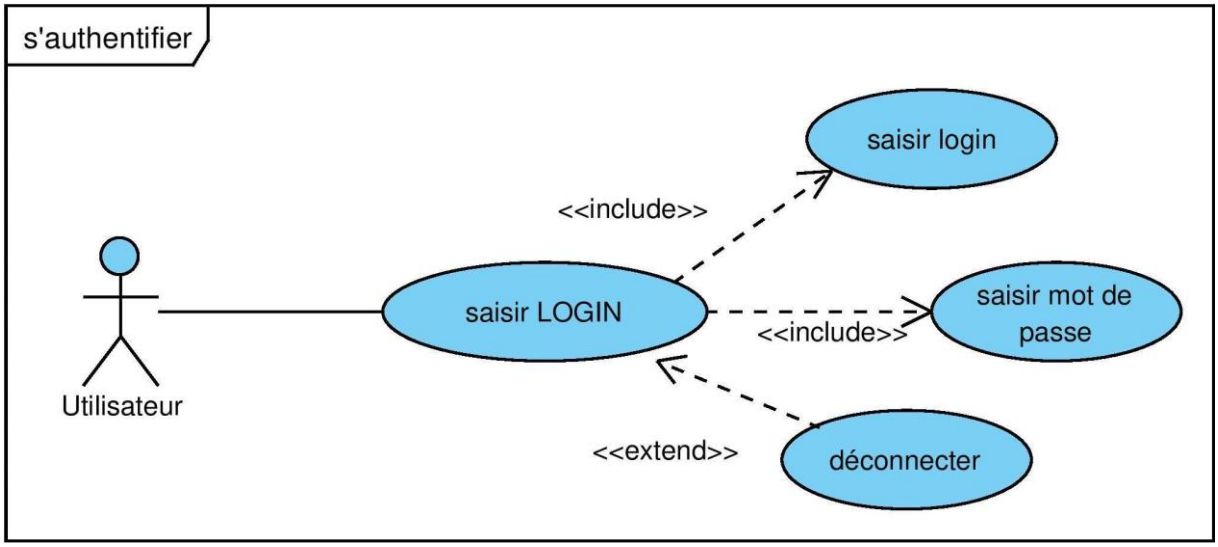

FIGURE 3 DIAGRAMME DE CAS D'UTILISATION « S'AUTHENTIFIER».

Acteurs	Utilisateur
Objectifs	Permet à l'utilisateur de se connecter au système, et s'assurer que l'utilisateur est bien celui qui prêtant être.
Scénario nominal	-Le système affiche la fenêtre d'authentification. -L'utilisateur introduit son login et son mot de passe. -Le système vérifie si les données saisies sont valides [A1]. -Le système affiche l'espace approprié pour chaque utilisateur.
Scénario alternatif	Alternative [A1] : si les informations introduites sont incorrectes le système

	affiche un message d'erreur et réaffiche la fenêtre d'authentification.

TABLEAU 2 DESCRIPTION DE CAS D'UTILISATION « AUTHENTIFICATION ».

3.4.2 Cas d'utilisation « Gérer les enseignants »

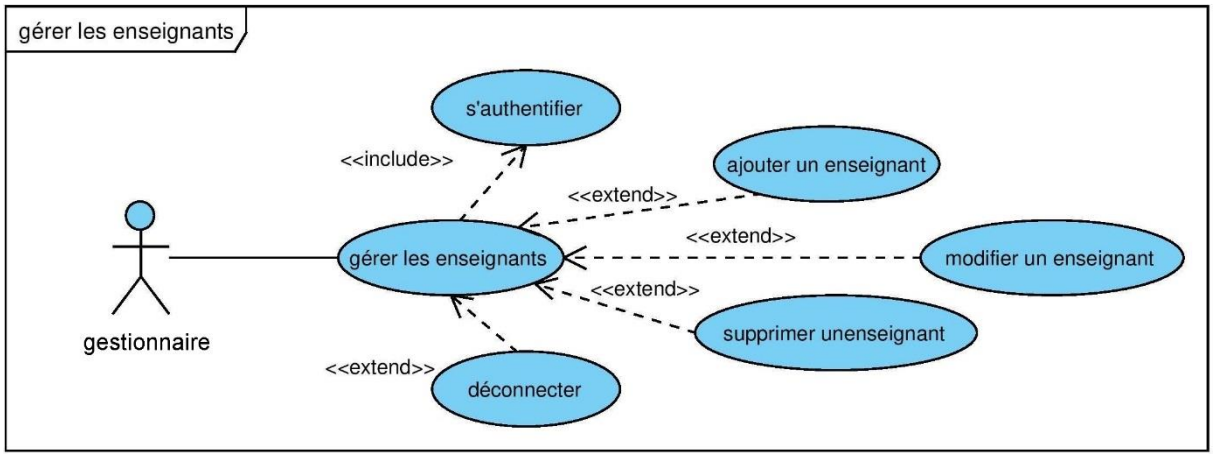

FIGURE 4 DIAGRAMME DE CAS D'UTILISATION « GERER LES ENSEIGNANTS».

Acteurs	Gestionnaire
Objectifs	Pouvoir ajouter, modifier, supprimer un enseignant.
Scénario nominal	-Le gestionnaire demande d'accéder à la liste des enseignants. -Le système affiche la fenêtre contenant la liste des enseignants. **Cas 1 : Ajouter un enseignant** -le gestionnaire choisit d'ajouter un enseignant. -Le système affiche le formulaire à remplir. - le gestionnaire rempli et valide le formulaire [A1]. -Le système ajoute les informations dans la base de données. -Le système actualise la liste des enseignants et l'affiche.

	Cas 2 : Modifier un enseignant - le gestionnaire choisit l'enseignant à modifier. -Le système affiche le formulaire de l'enseignant choisi. - le gestionnaire modifie les champs voulus [A2]. -Le système met à jour les informations dans la base de données. -Le système actualise la liste des enseignants et l'affiche. **Cas 3 : Supprimer un enseignant** - le gestionnaire choisit l'enseignant à supprimer. -Le système demande une confirmation. - le gestionnaire confirme ou annule la suppression. -Le système supprime l'enseignant de la base de données. -Le système actualise la liste des enseignants et l'affiche.
Scénario alternatif	Alternative [A1] : si les informations introduites sont incorrectes le système affiche un message d'erreur et réaffiche la fenêtre d'authentification. Alternative [A2] : modification avec des champs vides, champs non conforme aux types : un message d'erreur sera affiché.

TABLEAU 3 DESCRIPTION DE CAS D'UTILISATION « GERER LES ENSEIGNANTS ».

3.4.3 Cas d'utilisation « Gérer les étudiants »

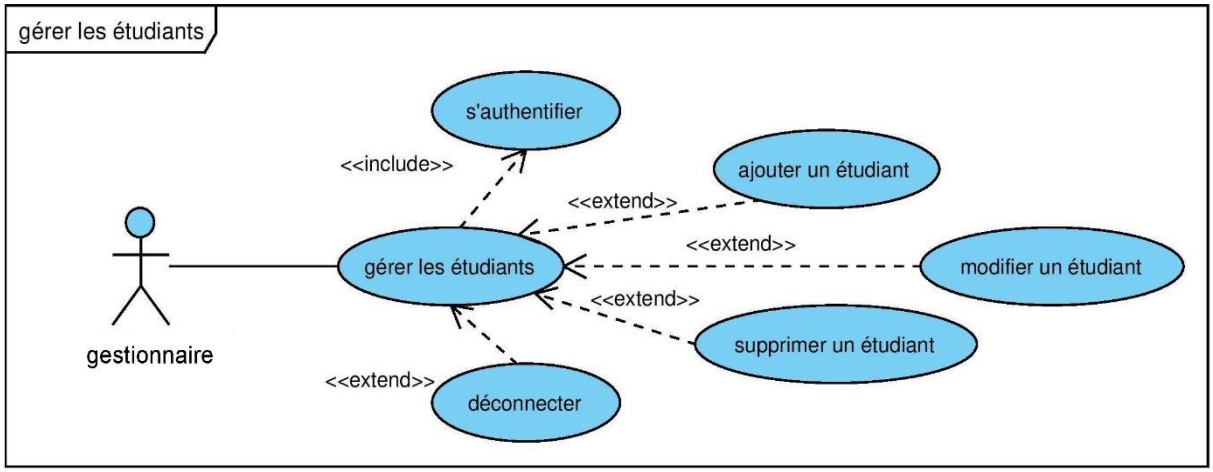

FIGURE 5 DIAGRAMME DE CAS D'UTILISATION « GERER LES ETUDIANTS».

Acteurs	Gestionnaire
Objectifs	Pouvoir ajouter, modifier, supprimer un étudiant.
Scénario nominal	-Le gestionnaire demande d'accéder à la liste des étudiants. -Le système affiche la fenêtre contenant la liste des étudiants. **Cas 1 : Ajouter un étudiant** -le gestionnaire choisit d'ajouter un étudiant. -Le système affiche le formulaire à remplir. - le gestionnaire rempli et valide le formulaire [A1]. -Le système ajoute les informations dans la base de données. -Le système actualise la liste des étudiants et l'affiche. **Cas 2 : Modifier un étudiant** - le gestionnaire choisit l'étudiant à modifier. -Le système affiche le formulaire de l'étudiant choisi.

	- le gestionnaire modifie les champs voulus [A2]. -Le système met à jour les informations dans la base de données. -Le système actualise la liste des étudiants et l'affiche. **Cas 3 : Supprimer un étudiant** - le gestionnaire choisit l'étudiant à supprimer. -Le système demande une confirmation. - le gestionnaire confirme ou annule la suppression. -Le système supprime l'étudiant de la base de données. -Le système actualise la liste des étudiants et l'affiche.
Scénario alternatif	Alternative [A1] : si les informations introduites sont incorrectes le système affiche un message d'erreur et réaffiche la fenêtre d'authentification. Alternative [A2] : modification avec des champs vides, champs non conforme aux types : un message d'erreur sera affiché.

TABLEAU 4 DESCRIPTION DE CAS D'UTILISATION « GERER LES ETUDIANTS ».

3.4.4 Cas d'utilisation « Gérer les formations »

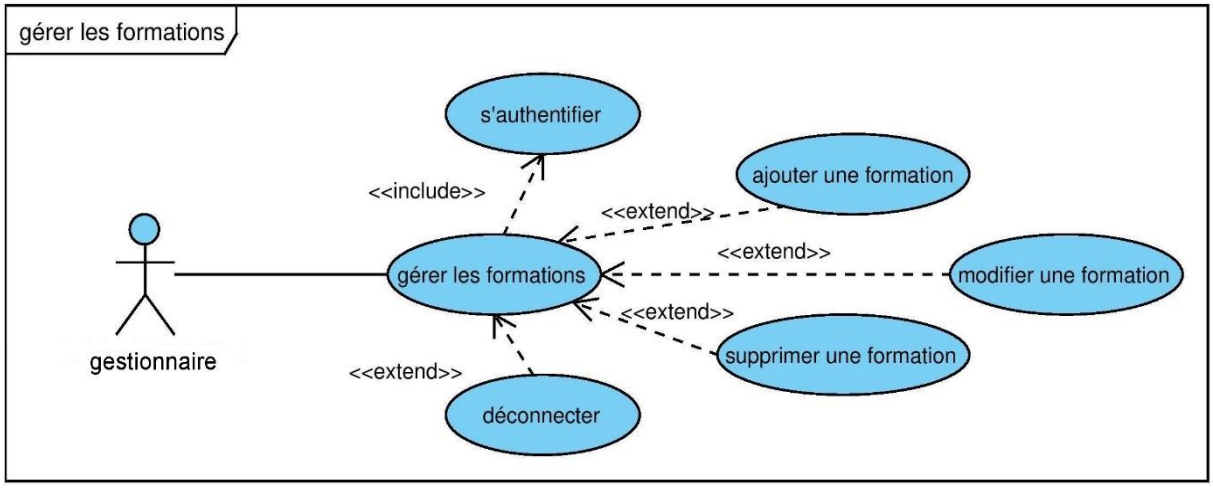

FIGURE 6 DIAGRAMME DE CAS D'UTILISATION « GERER LES FORMATIONS».

Acteurs	Gestionnaire
Objectifs	Pouvoir ajouter, modifier, supprimer une formation.
Scénario nominal	-Le gestionnaire accède à la page d'accueil après authentification [A1]. - Le gestionnaire demande d'accéder à la fenêtre gestion des formations. -Le système affiche la fenêtre correspondante. **Cas 1 : Créer une formation** - Le gestionnaire choisit d'ajouter une formation. -Le système affiche le formulaire à remplir. - Le gestionnaire ajout et valide [A2]. -Le système ajoute les informations dans la base de données. -Le système actualise la liste des formations et l'affiche. **Cas 2 : Modifier une formation** - Le gestionnaire choisit la formation à modifier. -Le système affiche le formulaire de modification. - Le gestionnaire modifie les champs voulus [A3]. -Le système met à jour les informations dans la base de données. -Le système actualise la liste des formations et l'affiche. **Cas 3 : Supprimer un groupe** - Le gestionnaire choisit la formation à supprimer. -Le système demande une confirmation de suppression. - Le gestionnaire confirme ou annule la suppression. -Le système supprime la formation dans la base de données.

	-Le système actualise la liste des formations et l'affiche.
Scénario alternatif	Alternative [A1] : le système n'arrive pas à identifier le gestionnaire et affiche un message d'erreur. Alternative [A2] : formation existe déjà ou champs non conforme aux types, formulaire vide : un message d'erreur sera affiché. Alternative [A3] : modification avec des champs vides, champs non conforme aux types : un message d'erreur sera affiché.

TABLEAU 5 DESCRIPTION DE CAS D'UTILISATION « GERER LES FORMATIONS ».

3.4.5 Cas d'utilisation « Gérer les groupes »

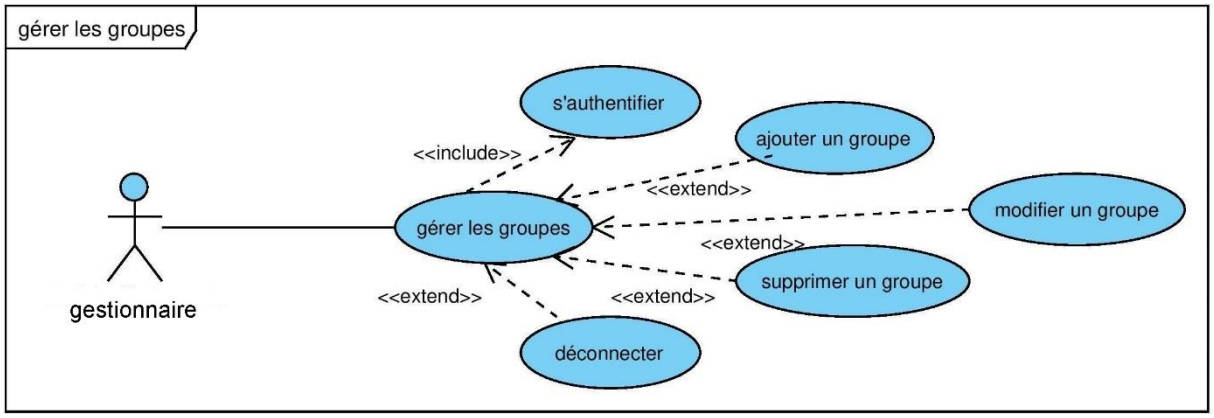

FIGURE 7 DIAGRAMME DE CAS D'UTILISATION « GERER LES GROUPES».

Acteurs	Gestionnaire
Objectifs	Pouvoir ajouter, modifier, supprimer un groupe.
Scénario nominal	-Le gestionnaire accède à la page d'accueil après authentification [A1]. - Le gestionnaire demande d'accéder à la fenêtre gestion des groupes. -Le système affiche la fenêtre correspondante.

	Cas 1 : Créer un groupe - Le gestionnaire choisit d'ajouter un groupe. -Le système affiche le formulaire à remplir. - Le gestionnaire ajout et valide [A2]. -Le système ajoute les informations dans la base de données. -Le système actualise la liste des groupes et l'affiche. **Cas 2 : Modifier un groupe** - Le gestionnaire choisit le groupe à modifier. -Le système affiche le formulaire de modification. - Le gestionnaire modifie les champs voulus [A3]. -Le système met à jour les informations dans la base de données. -Le système actualise la liste des groupes et l'affiche. **Cas 3 : Supprimer un groupe** - Le gestionnaire choisit le groupe à supprimer. -Le système demande une confirmation de suppression. - Le gestionnaire confirme ou annule la suppression. -Le système supprime le groupe dans la base de données. -Le système actualise la liste des groupes et l'affiche.
Scénario alternatif	Alternative [A1] : le système n'arrive pas à identifier le gestionnaire et affiche un message d'erreur. Alternative [A2] : groupe existe déjà ou champs non conforme aux types, formulaire vide : un message d'erreur sera affiché. Alternative [A3] : modification avec des

	champs vides, champs non conforme aux types : un message d'erreur sera affiché.

TABLEAU 6 DESCRIPTION DE CAS D'UTILISATION « GERER LES GROUPES ».

3.4.6 Cas d'utilisation « Gérer les modules »

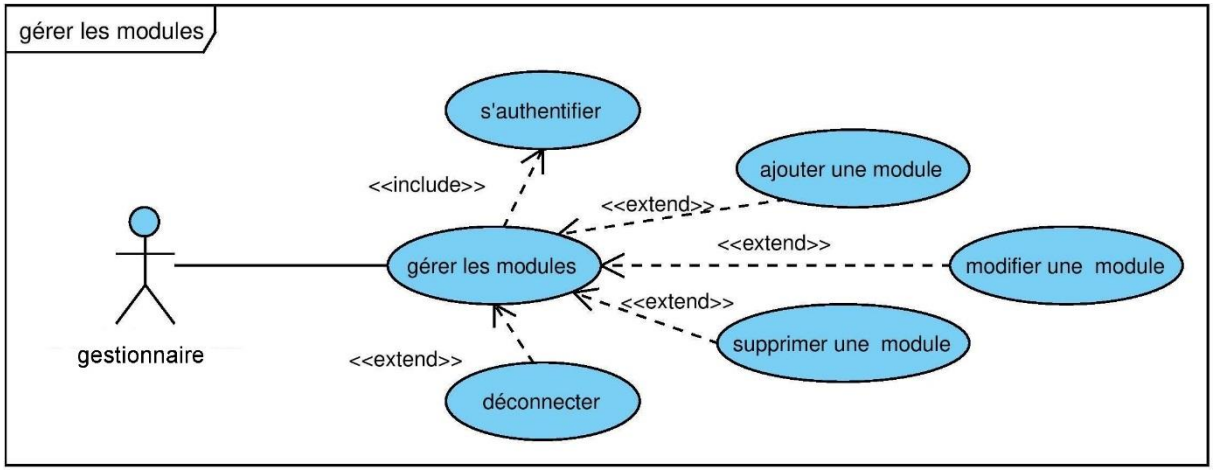

FIGURE 8 DIAGRAMME DE CAS D'UTILISATION « GERER LES MODULES».

Acteurs	Gestionnaire
Objectifs	Pouvoir ajouter, modifier, supprimer un module.
Scénario nominal	-Le gestionnaire accède à la page d'accueil après authentification [A1]. - Le gestionnaire demande d'accéder à la fenêtre gestion des modules. -Le système affiche la fenêtre correspondante. **Cas 1 : Créer un module** - Le gestionnaire choisit d'ajouter un module. -Le système affiche le formulaire à remplir. - Le gestionnaire ajout et valide [A2]. -Le système ajoute les informations dans la base de données. -Le système actualise la liste des modules et l'affiche.

	Cas 2 : Modifier un module - Le gestionnaire choisit le module à modifier. -Le système affiche le formulaire de modification. - Le gestionnaire modifie les champs voulus [A3]. -Le système met à jour les informations dans la base de données. -Le système actualise la liste des modules et l'affiche. **Cas 3 : Supprimer un module** - Le gestionnaire choisit le module à supprimer. -Le système demande une confirmation de suppression. - Le gestionnaire confirme ou annule la suppression. -Le système supprime le module dans la base de données. -Le système actualise la liste des modules et l'affiche.
Scénario alternatif	Alternative [A1] : le système n'arrive pas à identifier le gestionnaire et affiche un message d'erreur. Alternative [A2] : module existe déjà ou champs non conforme aux types, formulaire vide : un message d'erreur sera affiché. Alternative [A3] : modification avec des champs vides, champs non conforme aux types : un message d'erreur sera affiché.

TABLEAU 7 DESCRIPTION DE CAS D'UTILISATION « GERER LES MODULES ».

3.4.7 Cas d'utilisation « Gérer les salles »

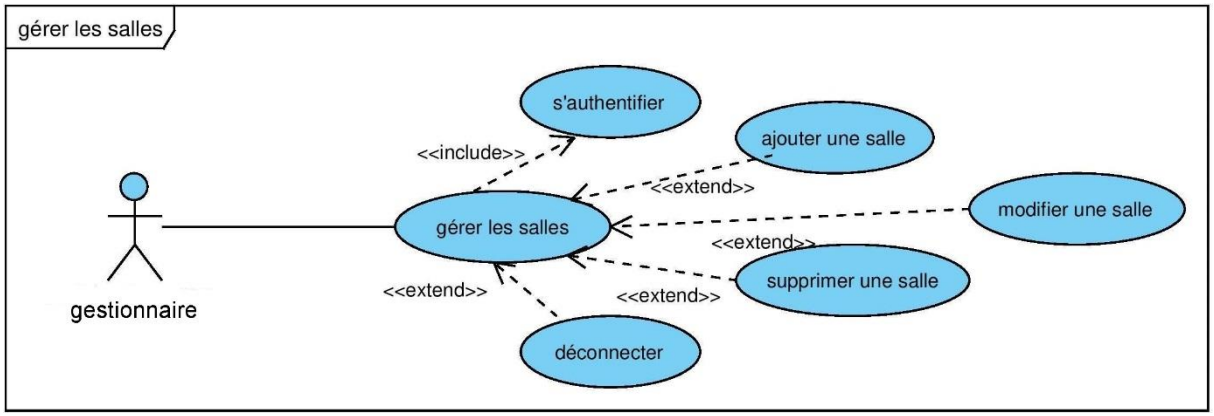

FIGURE 9 DIAGRAMME DE CAS D'UTILISATION « GERER LES SALLES».

Acteurs	Gestionnaire
Objectifs	Pouvoir ajouter, modifier, supprimer une salle.
Scénario nominal	-Le gestionnaire accède à la page d'accueil après authentification [A1]. - Le gestionnaire demande d'accéder à la fenêtre gestion des salles. -Le système affiche la fenêtre correspondante. **Cas 1 : Créer une salle** - Le gestionnaire choisit d'ajouter une salle. -Le système affiche le formulaire à remplir. - Le gestionnaire ajout et valide [A2]. -Le système ajoute les informations dans la base de données. -Le système actualise la liste des salles et l'affiche. **Cas 2 : Modifier une salle** - Le gestionnaire choisit la salle à modifier. -Le système affiche le formulaire de modification. - Le gestionnaire modifie les champs voulus [A3].

	-Le système met à jour les informations dans la base de données. -Le système actualise la liste des salles et l'affiche. **Cas 3 : Supprimer une salle** - Le gestionnaire choisit la salle à supprimer. -Le système demande une confirmation de suppression. - Le gestionnaire confirme ou annule la suppression. -Le système supprime la salle dans la base de données. -Le système actualise la liste des salles et l'affiche.
Scénario alternatif	Alternative [A1] : le système n'arrive pas à identifier le gestionnaire et affiche un message d'erreur. Alternative [A2] : salle existe déjà ou champs non conforme aux types, formulaire vide : un message d'erreur sera affiché. Alternative [A3] : modification avec des champs vides, champs non conforme aux types : un message d'erreur sera affiché.

TABLEAU 8 DESCRIPTION DE CAS D'UTILISATION « GERER LES SALLES ».

Note : la description des cas d'utilisation 'gérer section' et 'gérer semestre' fonctionnent de la même manière que 'gérer salle'

3.4.8 Cas d'utilisation « Affecter module / enseignant »

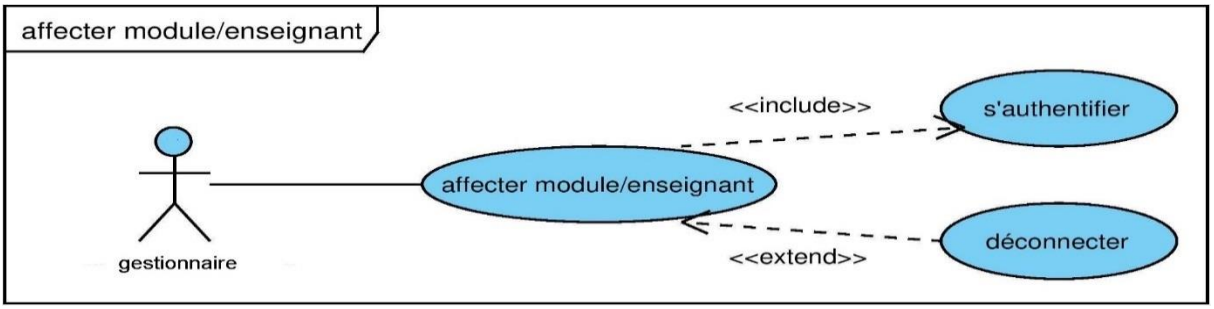

FIGURE 10 DIAGRAMME DE CAS D'UTILISATION « AFFECTER MODULE / ENSEIGNANT ».

Acteurs	Gestionnaire
Objectifs	Pouvoir affecter des modules aux enseignants
Scénario nominal	-Le gestionnaire accède à la page d'accueil après authentification [A1]. - Le gestionnaire demande d'accéder à la fenêtre d'affectation modules aux enseignants. -Le système affiche la fenêtre correspondante. - Le gestionnaire sélectionne la formation et le semestre -Le système affiche le formulaire à remplir. - Le gestionnaire sélectionne l'enseignant ; puis sélectionne le module à affecter et valide -Le système envoie la requête à la base de données. -La base de données renvoie la réponse au système. [A2] -Le système actualise la liste d'affectation des modules aux enseignants et l'affiche.
Scénario alternatif	Alternative [A1] : le système n'arrive pas à identifier le gestionnaire et affiche un message d'erreur. Alternative [A2] : affectation existe déjà: un message d'erreur sera affiché.

TABLEAU 9 DESCRIPTION DE CAS D'UTILISATION « AFFECTER MODULE / ENSEIGNANTS ».

3.4.9 Cas d'utilisation « gérer séances »

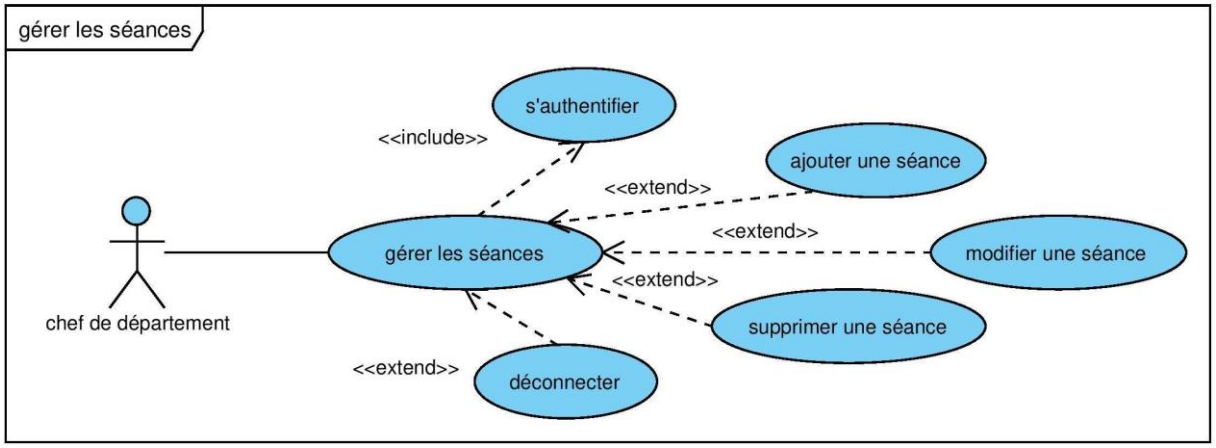

FIGURE 11 DIAGRAMME DE CAS D'UTILISATION « GERER LES SEANCES ».

Acteurs	Gestionnaire
Objectifs	Pouvoir ajouter, modifier, supprimer une séance.
Scénario nominal	-Le gestionnaire accède à la page d'accueil après authentification [A1]. - Le gestionnaire demande d'accéder à la fenêtre gestion des emplois du temps. -Le système affiche la fenêtre correspondante. **Cas 1 : Créer une séance** - Le gestionnaire choisit d'ajouter une séance dans un emploi du temps. -Le système affiche le formulaire des options (liste modules), (liste enseignants), (liste type séance) et (liste salles) à sélectionner : à noter que ses listes sont dynamiques. - Le gestionnaire sélectionne un module. -Le système génère la liste des enseignants correspondants. - Le gestionnaire sélectionne un enseignant ; puis sélectionne le type de séance. -Le système génère la liste de la salle inoccupée. - Le gestionnaire valide. [A2]

	-Le système actualise les séances dans l'emploi du temps. **Cas 2 : Modifier une salle** - Le gestionnaire choisit la séance à modifier. -Le système affiche le formulaire de modification. - Le gestionnaire modifie les champs voulus [A3]. -Le système met à jour les informations dans la base de données. -Le système actualise les séances dans l'emploi du temps et l'affiche. **Cas 3 : Supprimer une salle** - Le gestionnaire choisit la séance à supprimer. -Le système demande une confirmation de suppression. - Le gestionnaire confirme ou annule la suppression. -Le système supprime la séance dans la base de données. -Le système actualise la liste des salles et l'affiche.
Scénario alternatif	Alternative [A1] : le système n'arrive pas à identifier le gestionnaire et affiche un message d'erreur. Alternative [A2] : enseignant occupé : un message d'erreur sera affiché. Alternative [A3] : violation des contraintes (physiques ou pédagogiques) citées avant : un message d'erreur sera affiché.

TABLEAU 10 DESCRIPTION DE CAS D'UTILISATION « GERER LES SEANCES ».

Note *:* l'activité de suppression est transitive, alors :

- La suppression d'une formation implique la destruction de tous les semestres, les sections et les groupes qui l'appartiennent. Alors la destruction de la séance qu'elle appartient soit automatique ;
- La suppression d'un enseignant ou le module enseigné par ce dernier implique la destruction de la séance dans l'emploi du temps ;
- Et enfin, la suppression de la formation ou les semestres, les sections et les groupes qu'ils appartiennent, ou la suppression de la salle implique la destruction de la séance qu'ils appartiennent.

Après avoir consolidé les enchaînements des actions de nos cas d'utilisations, on passe maintenant à une autre illustration facilement compris par les utilisateurs, c'est les diagrammes de séquences.

3.5 Diagramme de séquence

Le diagramme de séquence est un diagramme d'interaction mettant l'accent sur **la chronologie** de l'envoi des messages. Il permet de :

- Montrer explicitement les interactions pouvant intervenir entre des objets ;
- Représenter les interactions en favorisant une vision temporelle de celles-ci ;
- Préciser la chronologie des interactions en précisant les contraintes temporelles.

Le diagramme de séquence permet de cacher les interactions d'objets dans le cadre d'un scénario d'un diagramme des cas d'utilisation. Dans un souci de simplification, on représente l'acteur principal à gauche du diagramme, et les acteurs secondaires éventuels à droite du système. Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets.

La dimension verticale du diagramme représente le temps, permettant de visualiser l'enchaînement des actions dans le temps, et de spécifier la naissance et la mort d'objets. Les périodes d'activité des objets sont symbolisées par des rectangles, et ces objets dialoguent par le biais de messages.

Les opérateurs d'interaction que nous avant utilisés dans les diagrammes de séquences sont :

- ***Référence (ref)*** : cet opérateur désigne que le fragment fait référence à un cas vue précédemment.
- ***Alternative (Alt) :*** cet opérateur ne désigne que le fragment composé représente un choix de comportement. Un opérande d'interaction au maximum sera choisi **[W4]**.

Puisque les quatre actions CRUD sont similaires et répétitives, on représente par le diagramme de séquence une seule action pour chaque cas d'utilisation.

CRUD : (Create, Read, Update, Delete) désigne les quatre opérations de base pour la persistance des données, en particulier le stockage des données.

3.5.1 Diagramme de séquence de cas d'utilisation « s'authentifier ».

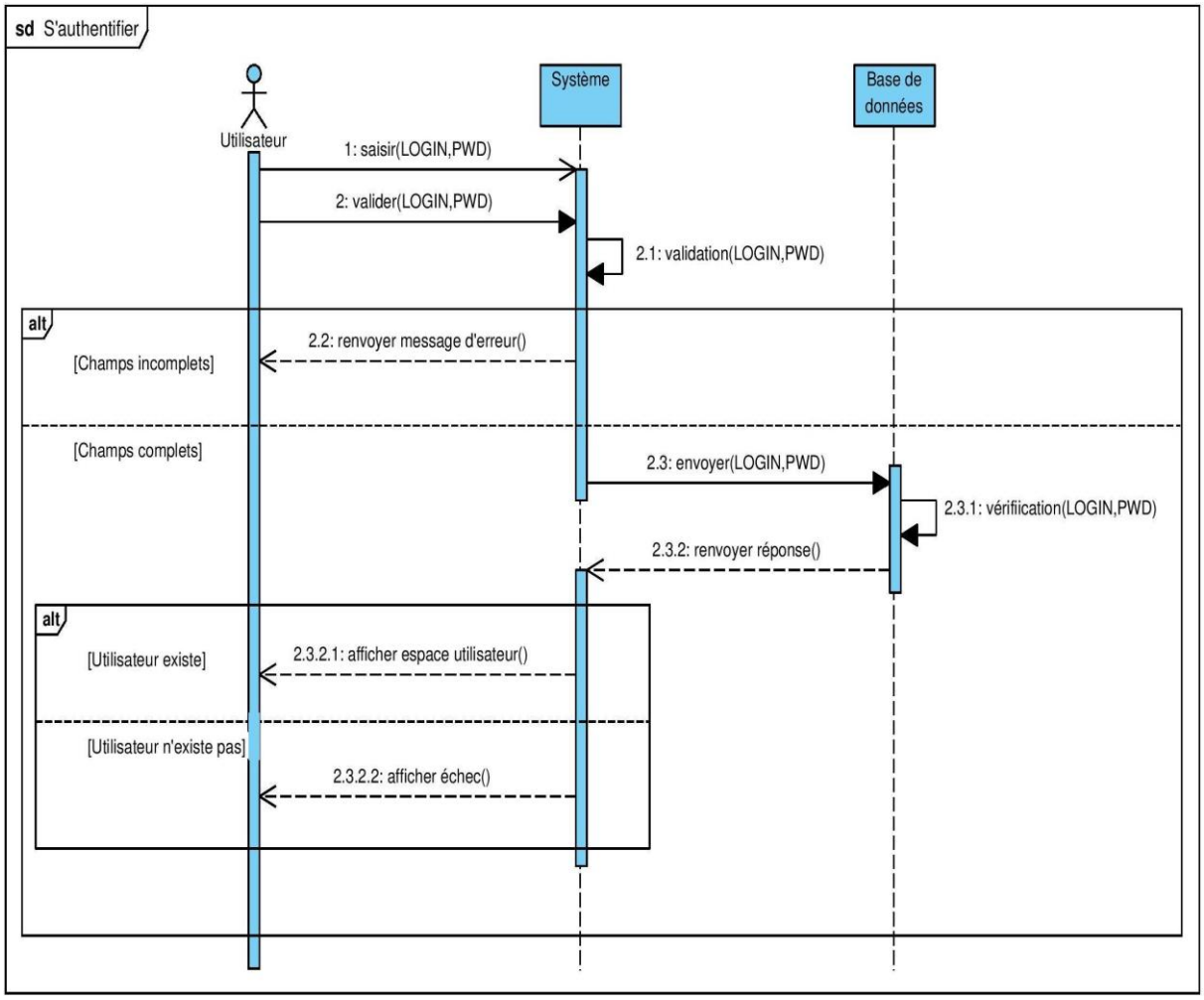

FIGURE 12 DIAGRAMME DE SEQUENCE « S'AUTHENTIFIER ».

3.5.2 Diagramme de séquence de cas d'utilisation « ajouter salle ».

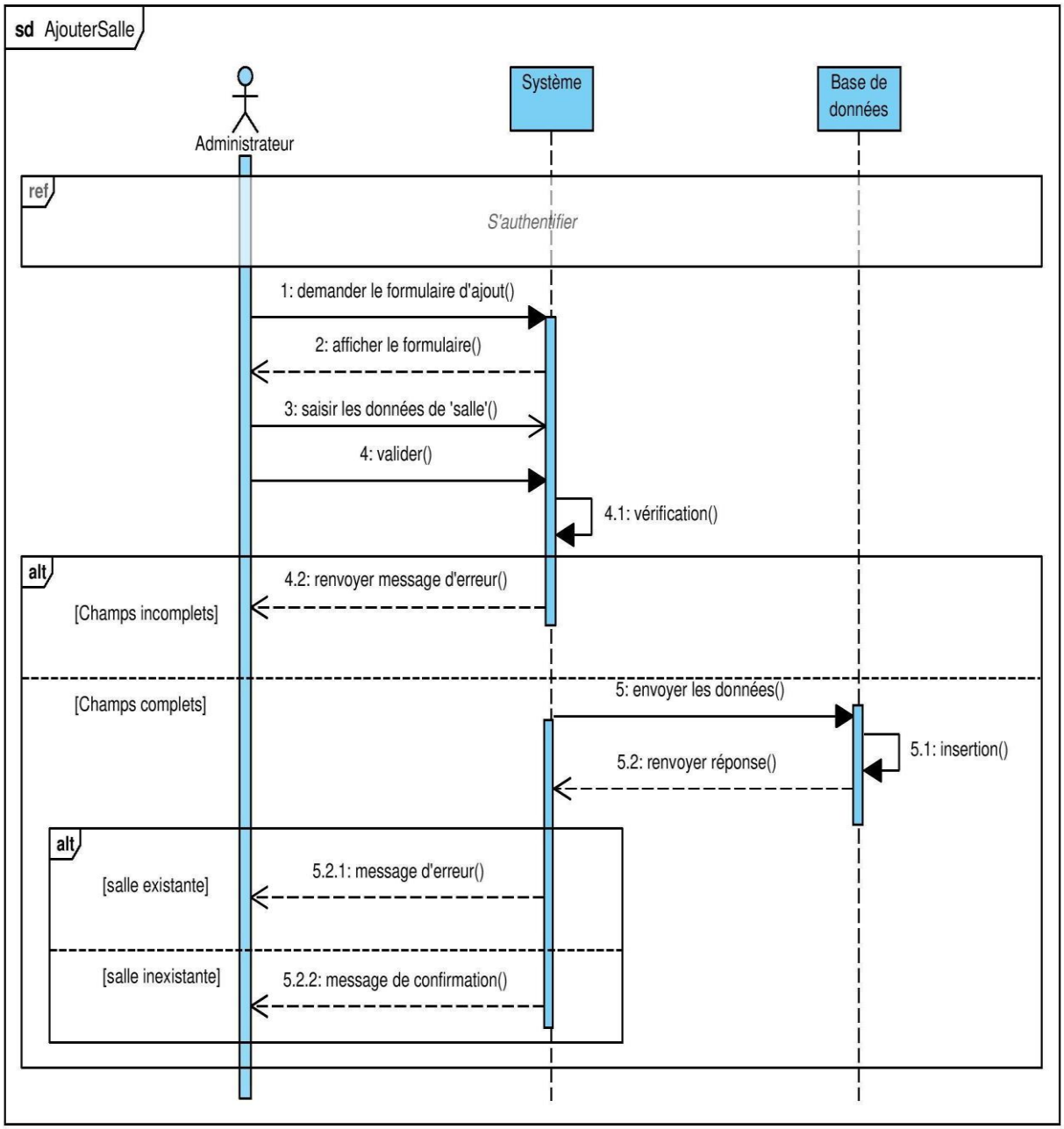

FIGURE 13 DIAGRAMME DE SEQUENCE « AJOUTER SALLE ».

3.5.3 Diagramme de séquence de cas d'utilisation « modifier module ».

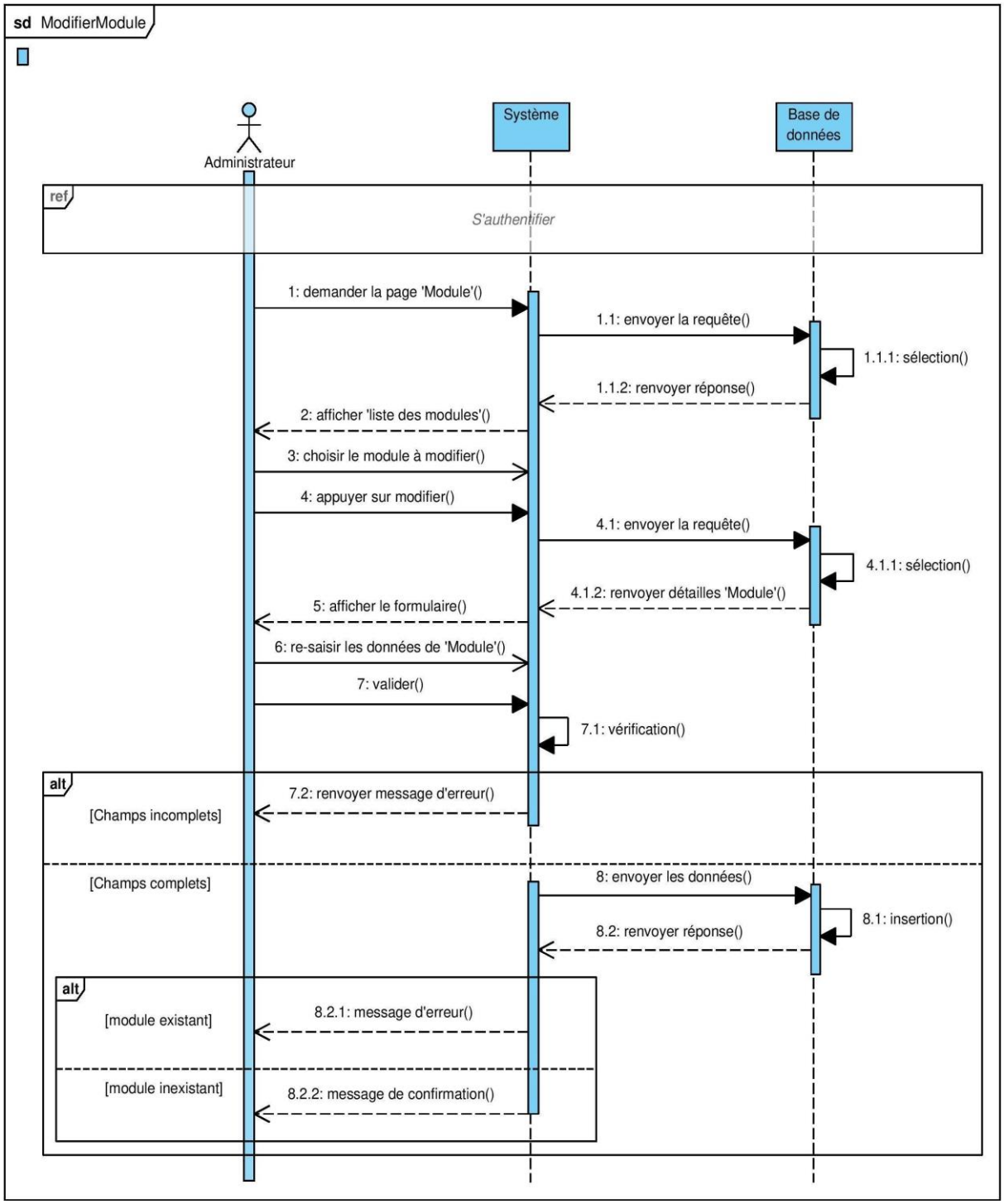

FIGURE 14 DIAGRAMME DE SEQUENCE « MODIFIER MODULE ».

3.5.4 Diagramme de séquence de cas d'utilisation « supprimer enseignant ».

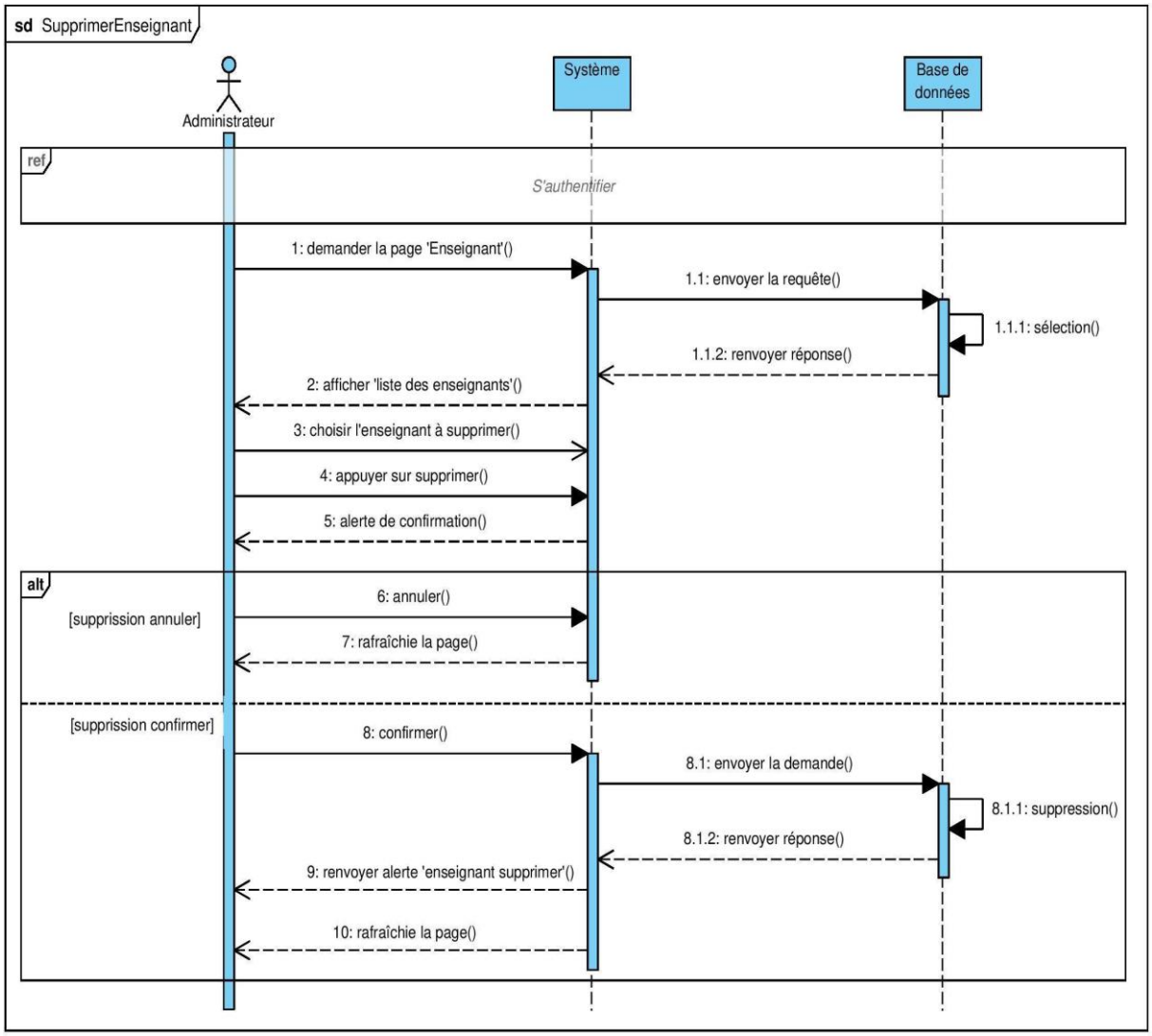

FIGURE 15 DIAGRAMME DE SEQUENCE « SUPPRIMER ENSEIGNANT ».

3.5.5 Diagramme de séquence de cas d'utilisation « affecter module à l'enseignant ».

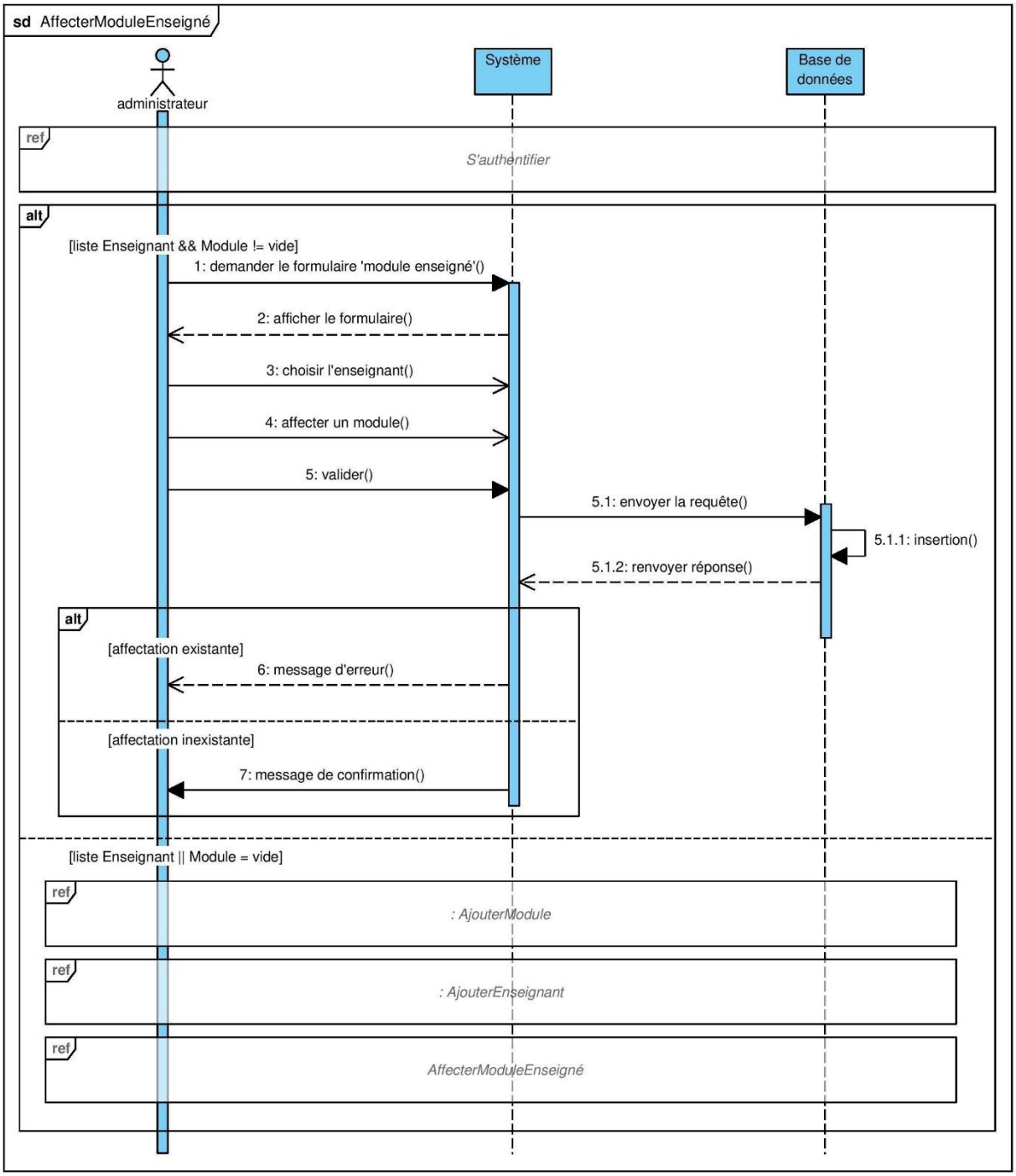

FIGURE 16 DIAGRAMME DE SEQUENCE « AFFECTER MODULE A L'ENSEIGNANT ».

3.5.6 Diagramme de séquence de cas d'utilisation « imprimer emploi du temps »

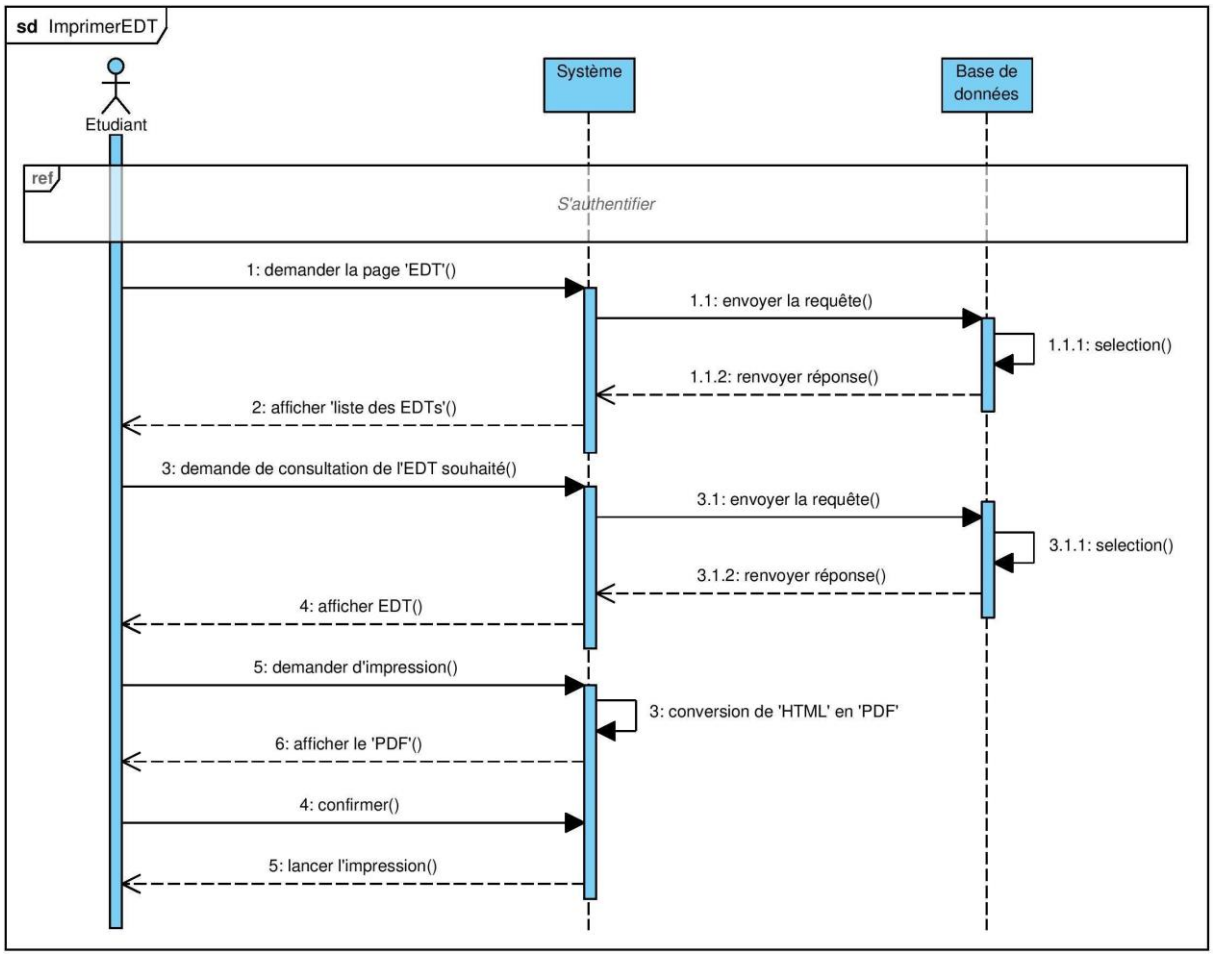

FIGURE 17 DIAGRAMME DE SEQUENCE « IMPRIMER EMPLOI DU TEMPS ».

À noter que, toute action d'impression exige la consultation.

Après avoir élaboré les descriptions textuelles puis les diagrammes de séquences des cas d'utilisation de notre système, Nous allons entamer une partie cruciale du développement Web et qui constitue un pont entre la spécification et la réalisation. Elle comporte la conception par le diagramme de classe ainsi que la conception de la base de données.

3.6 Diagramme de classe

Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne.

Le diagramme de classes est une représentation statique des éléments qui composent un système et de leurs relations. Chaque application qui va mettre en œuvre le système sera une instance des différentes classes qui le compose. Une classe permet de décrire un ensemble d'objet, tandis qu'une relation ou association permet de faire apparaitre des liens entre ces objets. Ce diagramme fait partie de la partie statique d'UML car il fait abstraction des aspects

temporels et dynamiques. Le diagramme de classes met en œuvre des classes contenant des attributs et des opérations, reliés par des associations ou des généralisations **[W5].**

Ce diagramme va nous permette de présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci.

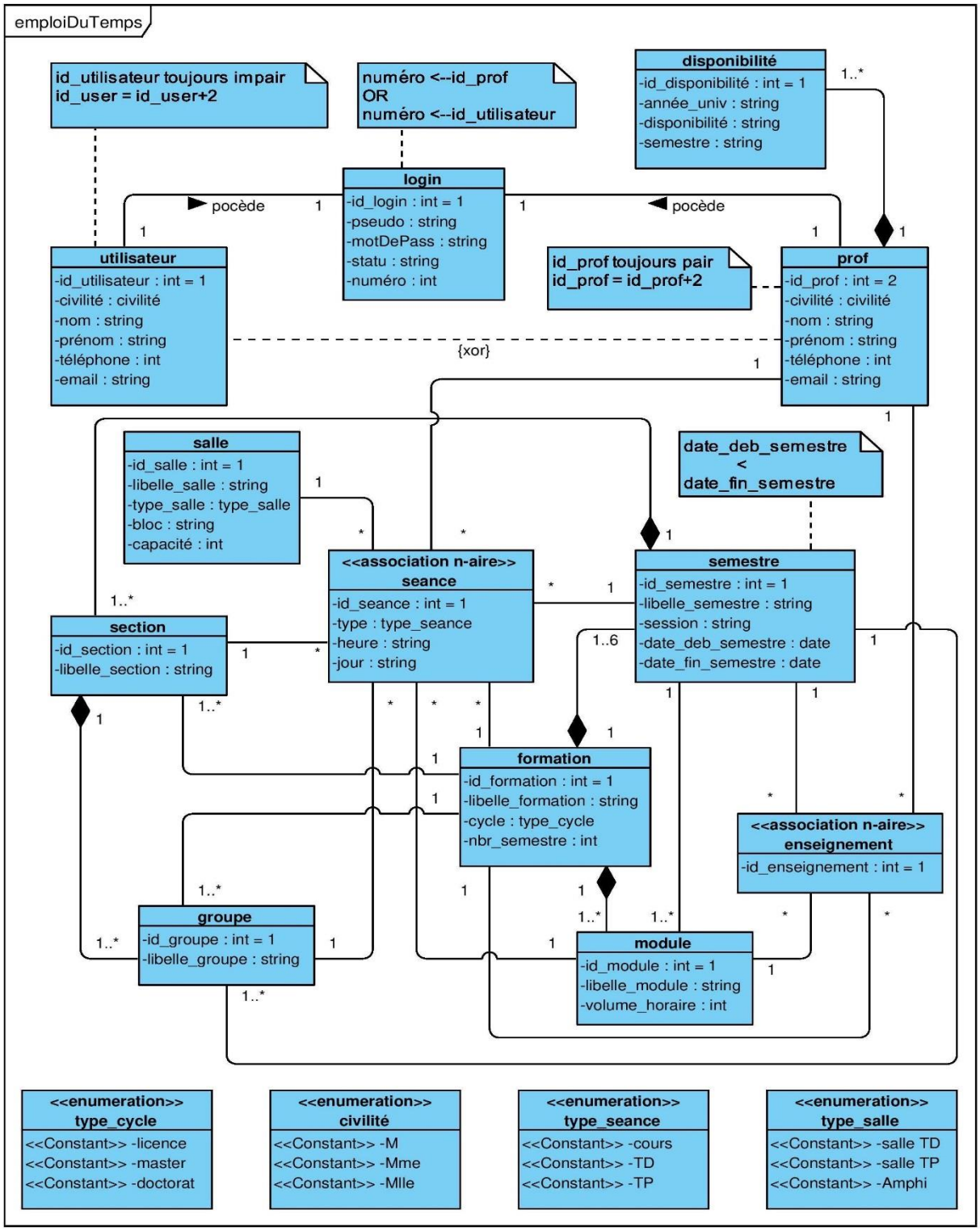

FIGURE 18 DIAGRAMME DE CLASSE DE NOTRE APPLICATION.

À noter que la classe séance et la classe enseignement sont exprimées au moyen d'un stéréotype qui précise que ces deux classes réalisent des associations n-aires, alors les multiples branches de ces associations s'instancient toutes simultanément, en un même lien.

La collection de l'ensemble des données primaires appeler attributs ou champs, constitue le dictionnaire des données. Chaque attribut du dictionnaire de données est caractérisé par : le nom de l'attribut, sa description et son type.

Classe	Attribut	Type
Disponibilité	• id_disponibilité • disponibilité • année_univ • semestre	• INT • VARCHAR(30) • VARCHAR(9) • VARCHAR(20)
Formation	• id_formation • libellé_formation • cycle • nbr_semestre	• INT • VARCHAR(50) • VARCHAR(10) • INT
Prof	• id_prof • civilité • nom • prénom • téléphone • email	• INT • VARCHAR(5) • VARCHAR(50) • VARCHAR(50) • INT • VARCHAR(50)
Utilisateur	• id_utilisateur • civilité • nom • prénom • téléphone • email	• INT • VARCHAR(5) • VARCHAR(50) • VARCHAR(50) • INT • VARCHAR(50)
Salle	• id_salle • libellé_salle • type_salle • bloc • capacité	• INT • VARCHAR(15) • VARCHAR(15) • VARCHAR(15) • INT

Semestre	• id_semestre • libellé_semestre • session • date_debu_semestre • date_fin_semestre	• INT • VARCHAR(30) • VARCHAR(20) • DATE • DATE
Login	• id_login • pseudo • motDePasse • statu • numéro	• INT • VARCHAR(30) • VARCHAR(30) • VARCHAR(5) • INT
Section	• id_section • libellé_section	• INT • VARCHAR(20)
Module	• id_module • libellé_module • volume_horaire	• INT • VARCHAR(30) • INT
Groupe	• id_groupe • libellé_groupe	• INT • VARCHAR(30)
Enseignement	• id_enseignement	• INT
Séance	• id_séance • type • heure • jour	• INT • VARCHAR(5) • VARCHAR(15) • VARCHAR(10)

TABLEAU 11 DICTIONNAIRE DES DONNEES

A ce niveau, il ne reste que la traduction de diagramme de classe de façon à optimiser l'implémentation de la base de données.

3.7 Passage au modèle relationnel

A partir de la description conceptuelle que nous avons effectuée, nous pouvons réaliser le modèle relationnel qui traduit le modèle entité/association en formalisme compréhensible par la machine **[W6].**

Le passage du diagramme de classes au modèle relationnel ne se fait pas au hasard. Il existe un certain nombre de règles qui nous permettent de réaliser cette opération.

Les éléments présents dans le diagramme des classes (classe, attribut, association, classe association) doivent se retrouver dans le modèle relationnel.

Règle 1. Transformation des classes : chaque classe du diagramme UML devient une relation, il faut choisir un attribut de la classe pouvant jouer le rôle de clé.

Règle 2. Transformation des associations : Nous distinguons trois familles d'associations.

- **2.1.** Association (1...*) : Il faut ajouter un attribut de type clé étrangère dans la relation fils de l'association. L'attribut porte le nom de la clé primaire de la relation père de l'association.
- **2.2.** Association (*...*) : La classe-association devient une relation. La clé primaire de cette relation est la concaténation des identifiants des classes connectées à l'association.

Règle 3. Présence d'une généralisation

Trois décompositions sont possibles pour traduire une association d'héritage en fonction des contraintes existantes :

- ***3.1.*** Méthode 1 (push-up)
  i. Créer une relation avec tous les attributs des classes.
  ii. Ajouter un attribut pour distinguer les types des objets.
- ***3.2.*** Méthode 2 (push-down)
  i. Créer une relation pour chaque sous type.
  ii. Chaque relation se compose des attributs génériques et des attributs.
- ***3.3.*** Méthode 3 (distinction)
  i. Transformer chaque sous-classe en une relation.
  ii. La clé primaire de la surclasse, migre dans la (les) relation(s) issue(s) de la (des) sous classe(s) et devient à la fois clé primaire et clé étrangère.

Note : Nous avons utilisé le caractère (**#**) pour désigner les clés étrangères, et le soulignement pour désigner les clés primaires.

En pratiquant les règles de gestion sises ci–dessus nous avons généré le modèle relationnel suivant :

Formation (id_formation, libellé_formation, cycle, nbr_semestre)

Disponibilité (id_disponibilité, disponibilité, année_univ, semestre, #id_prof)

Prof (id_prof, civilité, nom, prénom, téléphone, email)

Utilisateur (id_utilisateur, civilité, nom, prénom, téléphone, email**)**

Salle (id_salle, libellé_salle, type_salle, bloc, capacité)

Semestre (id_semestre, libellé_semestre, session, date_debu_semestre, date_fin_semestre, #id_formation)

Login (id_login, pseudo, motDePasse, statu, numéro)

Section (id_section**,** libellé_section, #id_formation, #id_semestre**)**

Module (id_module**,** libellé_module, volume_horaire, #id_formation, #id_semestre**)**

Groupe (id_groupe, libellé_groupe, #id_formation, #id_semestre, #id_section**)**

Enseignement (id_enseignement, #id_formation, #id_semestre, #id_prof, #id_module)

Séance (id_séance**,** type, heure, jour, #id_formation, #id_semestre, #id_section, #id_groupe, #id_prof, #id_module, #id_salle**)**

3.8 Conclusion

A l'issue de ce chapitre, nous avons pu concevoir notre application pour la génération des emplois du temps universitaire en se basant sur les diagrammes du langage UML à savoir le diagramme de cas d'utilisation, le diagramme d'activité, le diagramme de séquence et le diagramme de classe, et enfin, la conception de la base de données.

A ce stade de développement on est assez armé pour mettre sur pied l'application. Le chapitre suivant sera consacré à l'environnement technique de développement et à la réalisation de notre application

Chapitre 4 Réalisation et tests

4.1 Introduction

La partie réalisation est l'étape qui nous permet de concrétiser les solutions et suggestions proposées lors de l'analyse et conception que nous avons mis en place.

Dans ce chapitre nous allons présenter l'environnement et les outils de développement de notre application, expliquer son fonctionnement et présenter quelques interfaces illustratives.

4.2 Environnement et outils de développement

Pour la réalisation de notre application, nous avons utilisé un ordinateur portable ayant les caractéristiques suivantes :

- Intel(R) Core(TM)2 CPU T7400 @2.16GHz 2.17GHZ.
- 4 Go de mémoire vive.
- Windows 7 - 32 bits.

Pour réaliser notre application nous avons utilisé les logiciels et les langages suivants:

4.2.1 Les langages utilisés

4.2.1.1 HTML

HTML: HyperText Markup Language, est d'un langage de description de contenu qui va nous permettre de décrire l'aspect d'un document, d'y inclure des informations variées (textes, images, sons, animations etc.) et d'établir des relations cohérentes entre ces informations grâce aux liens hypertextes.

4.2.1.2 CSS

Les feuilles de style en cascade, forment un langage informatique qui décrit la présentation des documents HTML et XML (la mise en page, les couleurs et les polices), les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C).

4.2.1.3 JavaScript

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une partie essentielle des applications web. Avec les technologies HTML et CSS

JavaScript est un langage coté client, mais aussi employé pour les serveurs avec l'utilisation (par exemple) de Node.js ou de Deno **[W7].**

4.2.1.4 PHP

PHP: Hypertext Preprocessor, est un "langage de programmation" libre, coté serveur,

principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale **[W8].**

4.2.1.5 SQL

Le SQL signifie "Structured Query Language" : le langage standard pour les traitements de bases de données **[W9].** Sa syntaxe est proche de la langue anglaise.

4.2.2 La conception Web réactive (RWD)

La conception Web réactive (RWD : Responsive Web Design) est une approche de conception Web qui permet aux pages Web de bien s'afficher sur une variété d'appareils, de tailles de fenêtres et d'écran.

La conception Web réactive devient de plus en plus importante en raison du volume de trafic mobile qui représente désormais plus de la moitié de tout le trafic Internet.

4.2.2.1 Bootstrap

Bootstrap est un framework CSS gratuit et open-source destiné au développement Web frontal réactif et mobile. Il contient des modèles de conception CSS et (éventuellement) JavaScript pour la typographie, les formulaires, les boutons, la navigation et d'autres composants d'interface **[10]**.

4.2.2.2 JQuery

JQuery est une bibliothèque (c'est-à-dire un ensemble de codes prêts à l'emploi) conçue pour simplifier l'écriture de codes JavaScript et AJAX. Créée en 2006 par John Resig, cette bibliothèque est la plus célèbre et la plus utilisée à ce jour.

4.2.2.3 JQuery UI

JQuery UI est un ensemble organisé d'interactions, d'effets, de widgets et de thèmes d'interface utilisateur créés au-dessus de la bibliothèque JavaScript jQuery, conçu pour créer des applications Web hautement interactives.

Dans notre application, on a utilisé le widgets *datepicker* pour contrôler les dates (date de fin et date de début) **[W11]**.

4.2.3 WampServer

WampServer est une plate-forme de développement web sous Windows pour des applications web dynamiques à l'aide du serveur Apache, du langage de scripts PHP et d'une base de données MySQL. Il possède également PHPMyAdmin pour gérer les bases de données **[W12].**

4.2.4 MySQL

MySQL est un serveur de bases de données relationnelles Open Source. Un serveur de

bases de données stocke les données dans des tables séparées plutôt que de tout rassembler dans une seule table.

4.2.5 HeidiSQL

HeidiSQL est une interface graphique pour la gestion des serveurs MariaDB ou MySQL, des bases de données Microsoft SQL, PostgreSQL ou SQLite. "Heidi" vous permet de parcourir et d'éditer des données facilement, de créer et d'éditer des tables, ... En outre, nous pouvons exporter la structure et les données, soit vers un fichier SQL, un presse-papiers ou vers d'autres serveurs **[W13]**.

Cet outil est très léger, libre d'utilisation, portable et nous permet d'écrire des requêtes avec la coloration syntaxique et la complétion de code personnalisables.

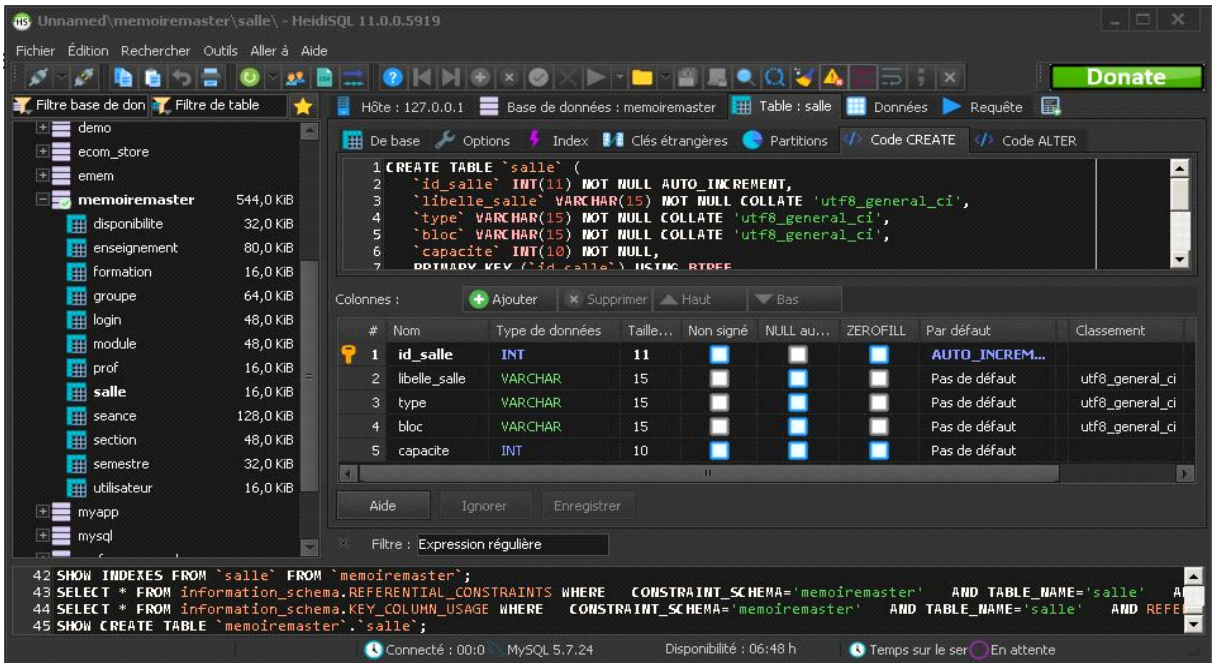

FIGURE 19 INTERFACE DE HEIDISQL VERSION 11.0.0.5919

4.2.6 Sublime Text

Sublime Text est un éditeur de texte générique multiplateforme. Le logiciel prend en charge 44 langages de programmation majeurs, tandis que des plugins sont souvent disponibles pour les langages plus rares, riche en fonctionnalités, parmi ses avantages, la coloration syntaxique et la complétion de code personnalisables, la minimap (mini carte), la sauvegarde automatique et la recherche et remplacement par expressions régulières **[W14]**.

4.2.7 Chrome DevTools

Chrome DevTools est un ensemble d'outils de développement Web intégrés directement dans le navigateur Google Chrome. DevTools peut nous aider à faire le débogage, à modifier des pages à la volée et à diagnostiquer rapidement les problèmes, ce qui vous aide finalement à créer de meilleurs sites Web, plus rapidement **[W15]**.

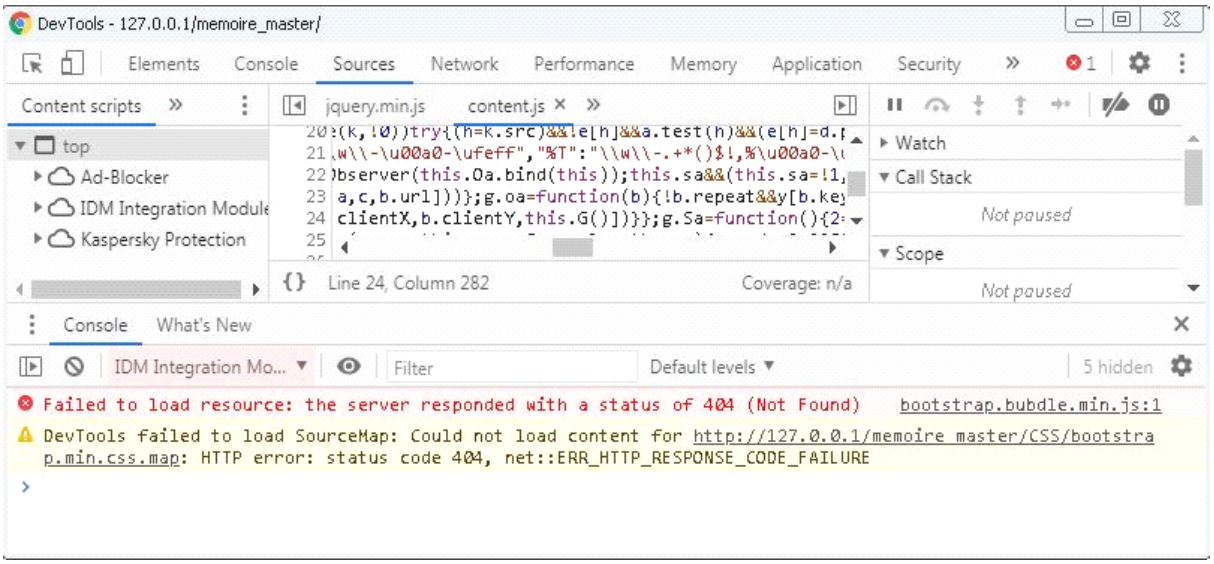

FIGURE 20 INTERFACE DE DEVTOOLS DE GOOGLE CHROME VERSION 86.0.4240.111

4.3 Présentation de quelques interfaces

La conception des interfaces de l'application est une étape très importante puisque toutes les interactions avec le cœur de l'application passent à travers ces interfaces, on doit alors guider l'utilisateur avec des messages d'erreurs et des notifications si besoin et présenter ainsi un système complet.

Notre application contient trois espaces : espace Administrateur du système (Ggestionnaire) espae Enseignant et espace Etudiant.

4.3.1 Page d'authentification

Pour qu'un utilisateur puisse accéder à son espace, il doit d'abord passer par la page d'authentification et saisir ses propres coordonnées (Login et mot de passe) puis cliquer sur le bouton « login ».

A noter que l'enseignant et l'étudiant saisissent leurs coordonnées directement dans le formulaire, et le type de reconnaissance de l'utilisateur est automatique, sauf pour l'administrateur, qui doit d'abord cliquer sur le bouton « admin » en haut à gauche, comme le montre la figure suivante.

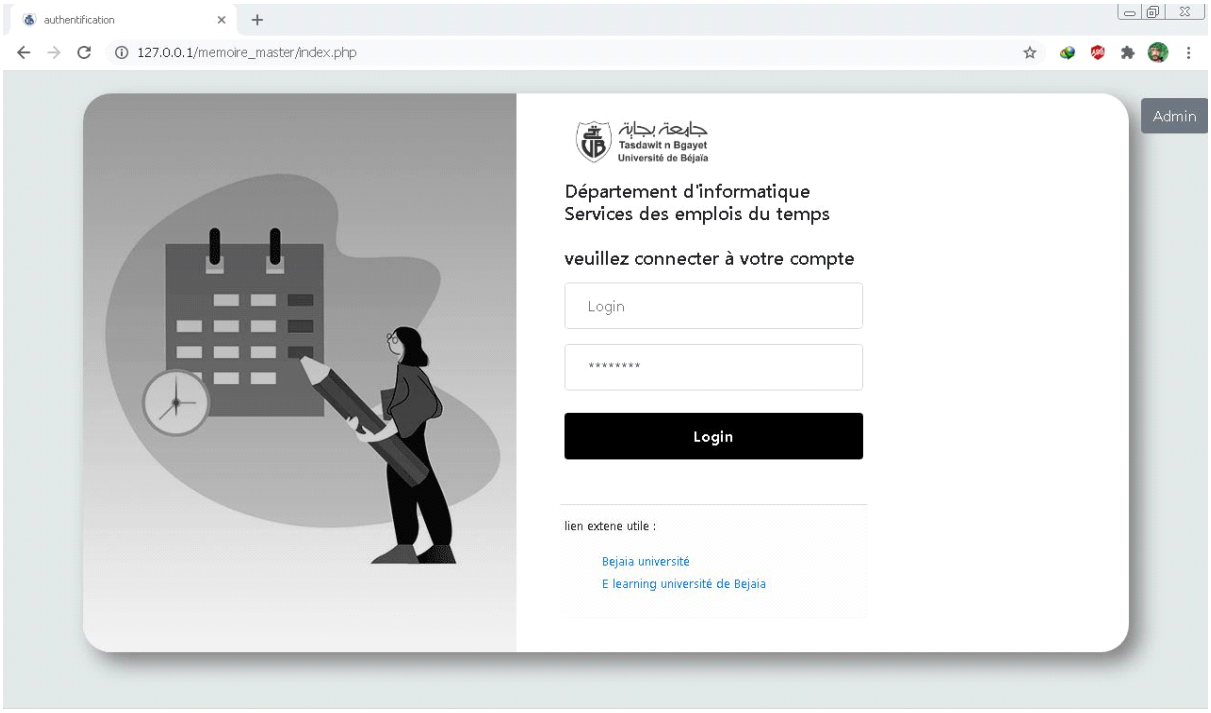

FIGURE 21 INTERFACE « AUTHENTIFICATION ».

Les privilèges de l'étudiant sont:

- La consultation et l'impression de l'emplois du temps de son groupe.

Les privilèges de l'enseignant sont:

- La consultation et l'impression de son propre emploi du temps;
- La consultation de sa disponibilité.

Les privilèges de l'administrateur sont:

- La gestion des ressources humaines, des ressources matérielles et des emplois du temps, il a donc des privilèges propres à lui ainsi que ceux de l'étudiant et de l'enseignant.

Dans ce qui suit, nous présentons les interfaces de l'administrateur.

4.3.2 Page du menu principal

Après avoir saisi un mot de passe valide, la fenêtre ci-dessous s'affiche, elle comporte le menu principal où l'utilisateur pourra sélectionner la tâche à effectuer.

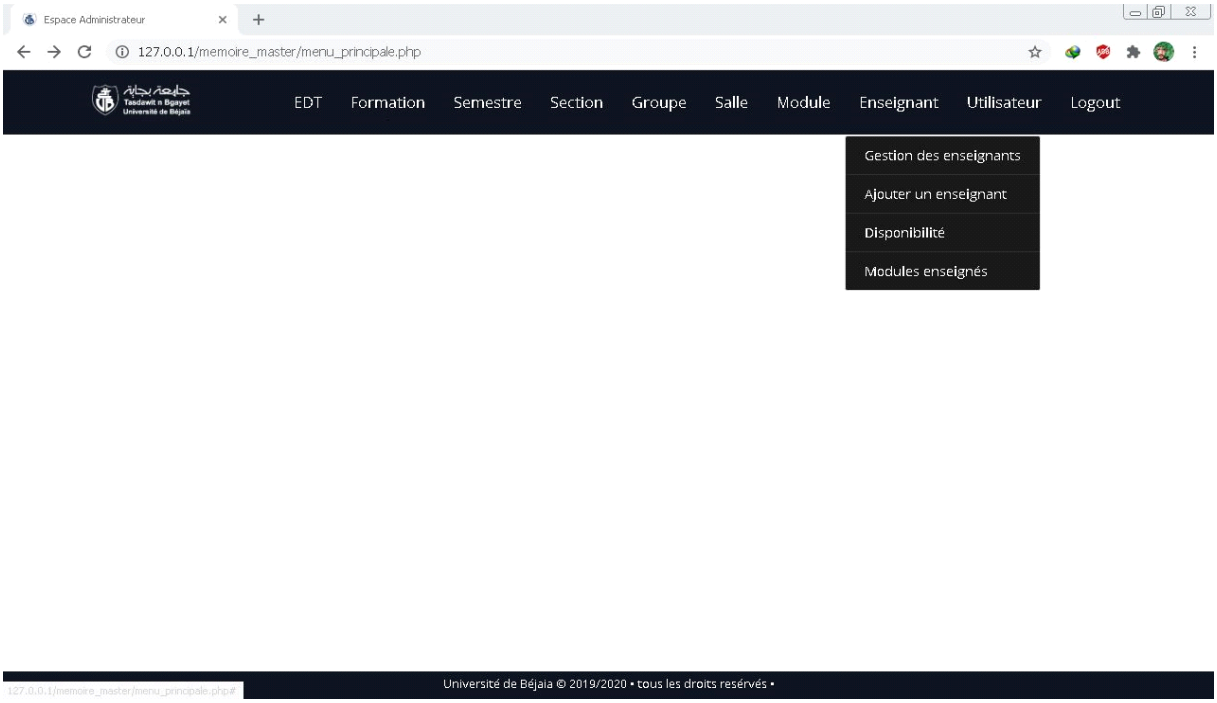

FIGURE 22 INTERFACE « MENU PRINCIPALE DE L'ADMINISTRATEUR, VERSION DESKTOP ».

Dans le cas précédent, le menu est horizontal et la page a été affichée dans un navigateur d'ordinateur doté d'un grand écran (plus de 1200 pixels).

L'application est réactive, elle s'adapte à des écrans plus petits tels que les tablettes et les Smartphone, la figure suivante montre la même page dans le navigateur d'un Smartphone (menu vertical).

FIGURE 23 INTERFACE « MENU PRINCIPALE DE L'ADMINISTRATEUR, VERSION MOBILE ».

4.3.3 Page ajouter formation

En cliquant sur formation -> (ajouter formation) dans le menu principale, le formulaire (ajouter formation) s'affiche.

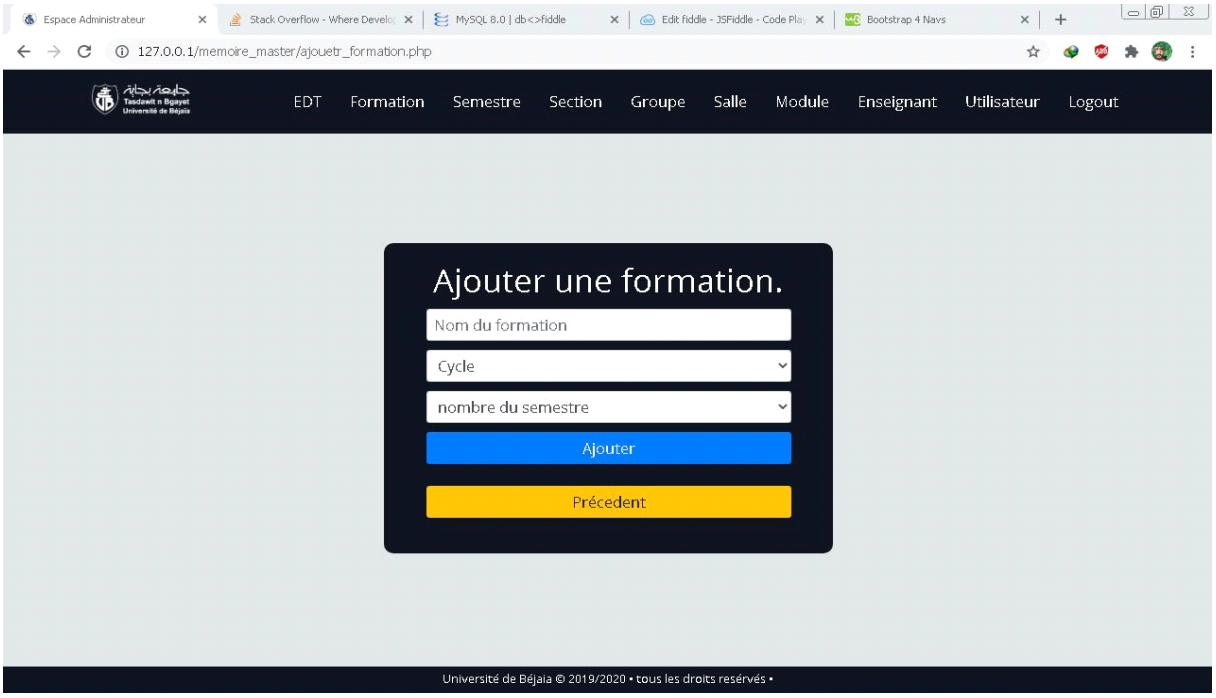

FIGURE 24 INTERFACE « AJOUTER FORMATION ».

Pour que l'administrateur puisse ajouter une formation, il doit remplir le formulaire avec les informations correspondantes, en commençant par entrer le libellé de la formation, choisir le cycle d'étude ainsi que le nombre de semestres pour cette formation, et enfin cliquer sur le bouton ajouter, sinon cliquer sur le bouton précédent pour quitter cette page.

4.3.4 Page ajouter semestre

En cliquant sur semestre -> (ajouter semestre) dans le menu principale, le formulaire (ajouter un semestre) s'affiche.

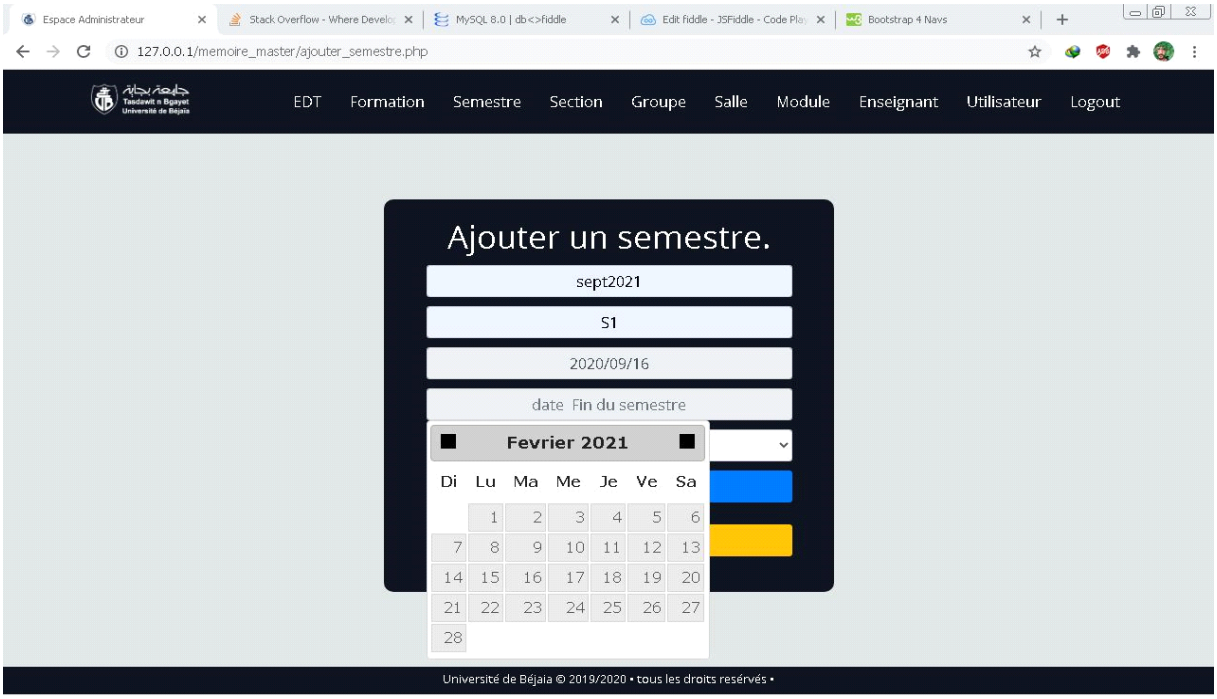

FIGURE 25 INTERFACE « AJOUTER SEMESTRE ».

Pour que l'administrateur puisse ajouter un semestre, il doit remplir le formulaire avec les informations correspondantes, en commençant par entrer le libellé de la session, choisir le semestre, la date de début et de fin du semestre, la formation, et enfin appuyer sur le bouton ajouter, sinon cliquer sur le bouton précédent pour quitter cette page.

A noter que le reste des procédures d'ajouts (ajouter section, groupe, salle, module, enseignant et utilisateur) fonctionnent de la même façon.

4.3.5 Page d'affectation module/enseignant

En cliquant sur enseignant -> (module enseignée) -> ajouter, dans le menu principale, le formulaire ajouter enseignement s’affiche.

L'ajout des modules pour les enseignants se fait en deux phases, choisir la formation et un semestre, puis l'affectation des modules, comme le montrent les figures suivantes:

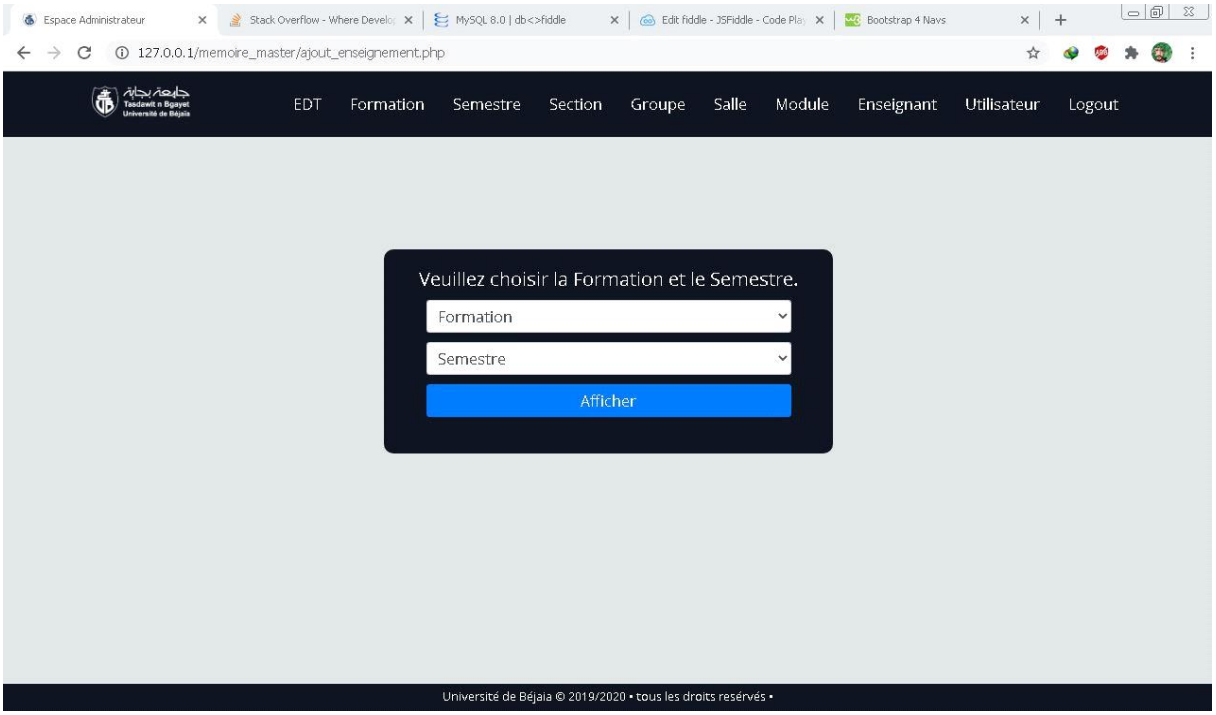

FIGURE 26 INTERFACE 1 « AFFECTATION MODULE/ENSEIGNANT ».

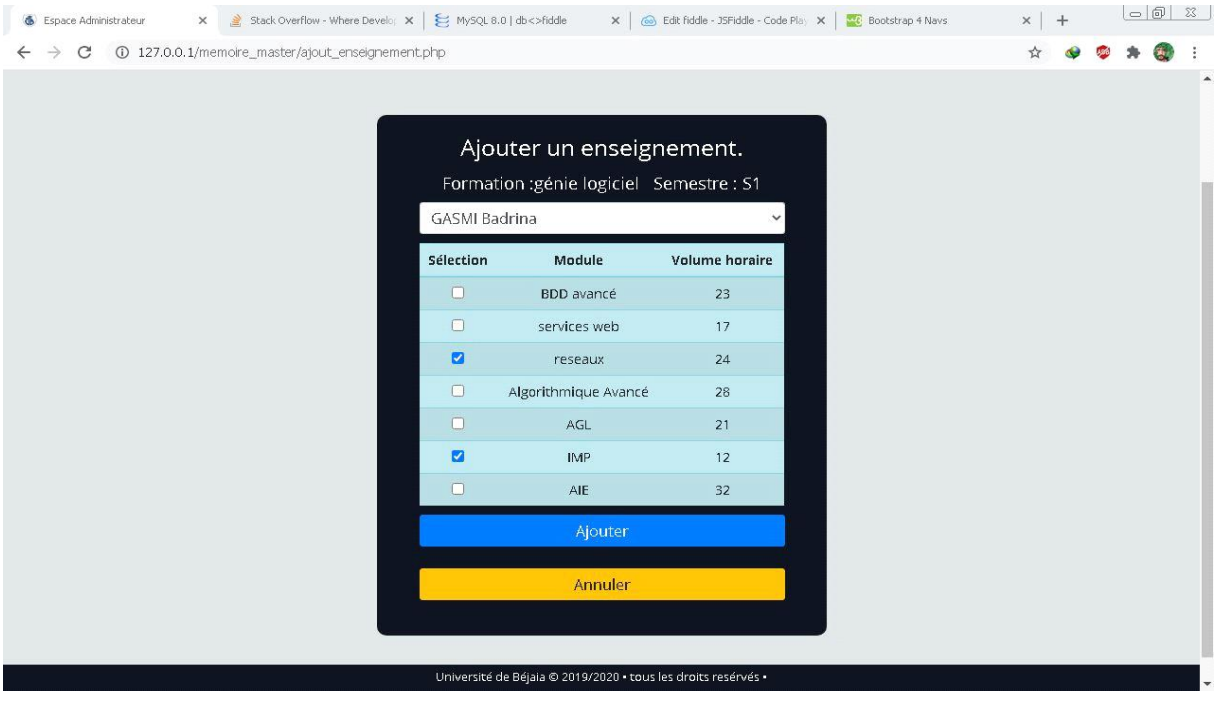

FIGURE 27 INTERFACE 2 « AFFECTATION MODULE/ENSEIGNANT ».

Pour que l'administrateur puisse affecter des modules pour les enseignants, il doit choisir l'enseignants dans la liste déroulante puis sélectionner les modules enseignés dans le tableau des modules et enfin appuyer sur le bouton ajouter, sinon cliquer sur le bouton annuler pour quitter cette page.

4.3.6 Page disponibilité des enseignants

En cliquant sur enseignant -> disponibilité -> ajouter, dans le menu principale, le formulaire ajouter disponibilité s'affiche.

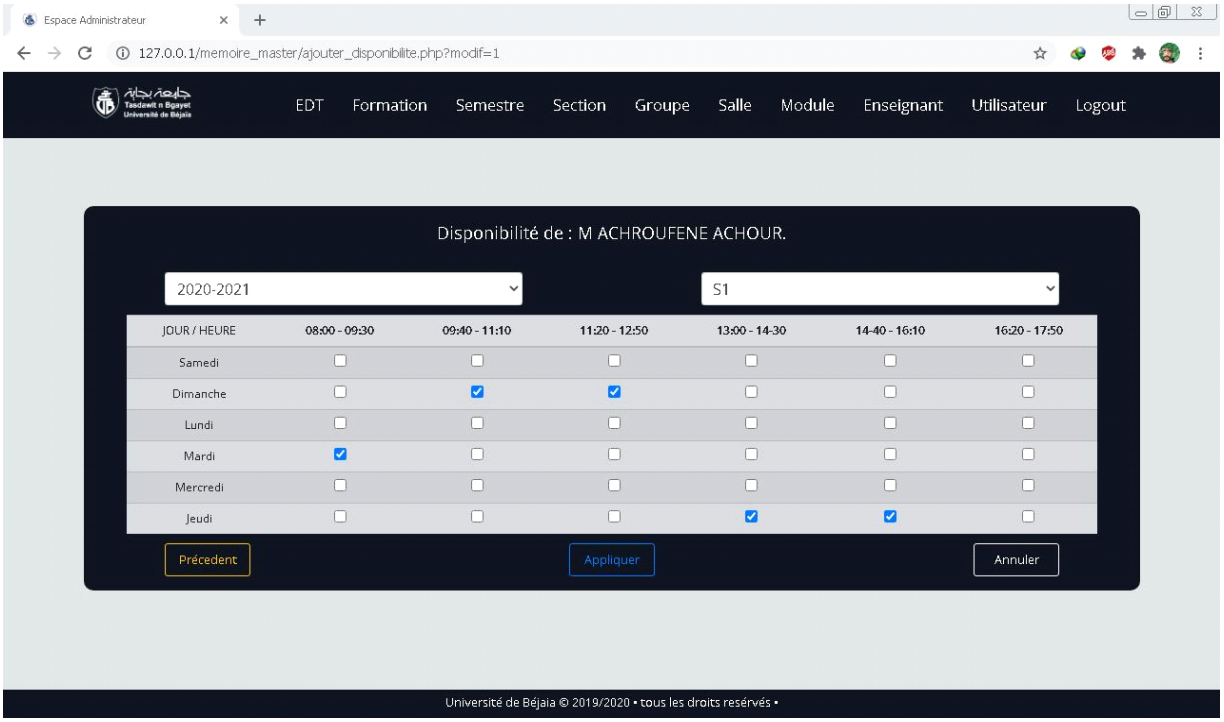

FIGURE 28 INTERFACE « AJOUTER DISPONIBILITE ».

Cette interface est vraiment intuitive, le principe est simple, il suffit juste de choisir l'année universitaire et le semestre d'étude, puis de sélectionner les cases correspondantes et enfin, appuyer sur le bouton appliquer, sinon sur le bouton précédent pour quitter la page sans application.

La figure ci-dessous montre la page "consulter disponibilité", l'avatar vert signifie que l'enseignant est disponible et l'avatar rouge signifie qu'il est non disponible pendant cette période.

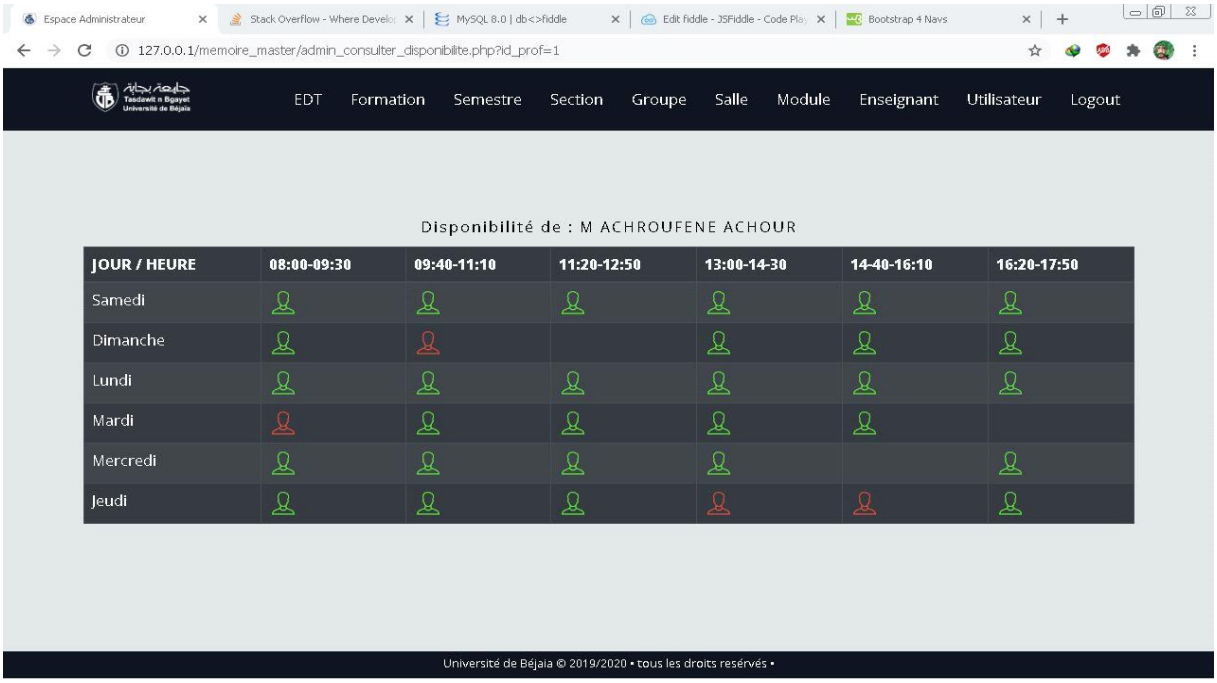

FIGURE 29 INTERFACE « CONSULTER DISPONIBILITE ».

4.3.7 Page gestion des emplois du temps

En cliquant sur l'intitulé de la gestion que l'administrateur veut effectuer, la page souhaitée s'affiche.

Prenons la page de gestion des emplois du temps comme exemple.

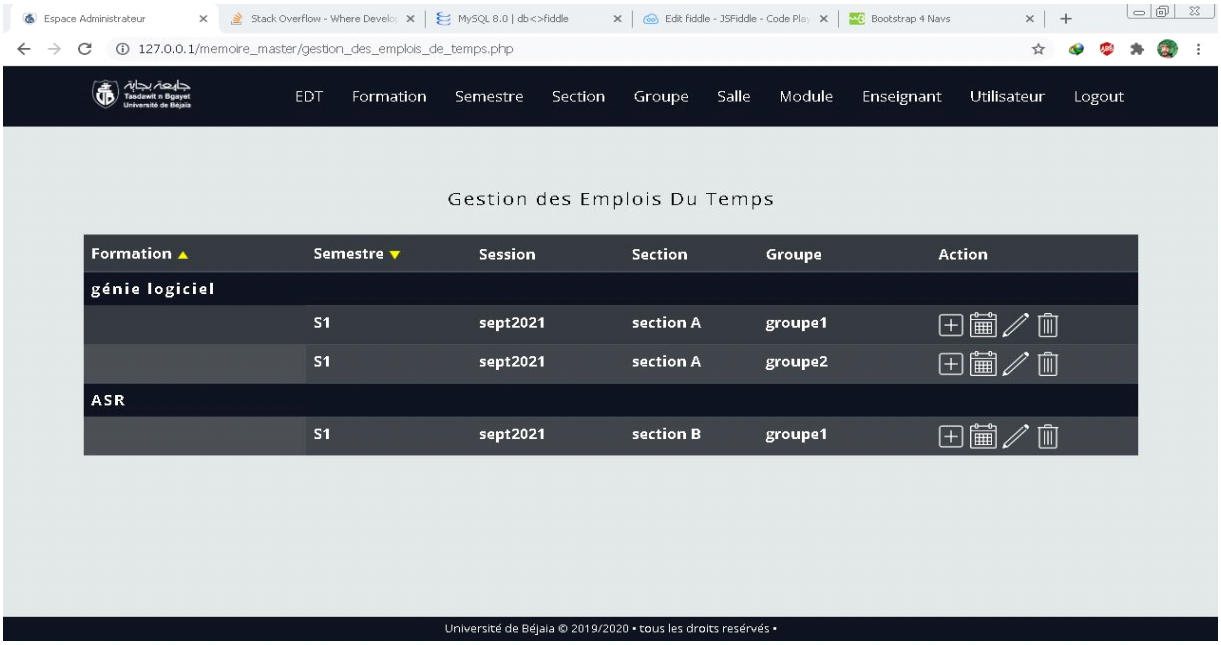

FIGURE 30 INTERFACE « GESTION DES EMPLOIS DU TEMPS ».

Dans cette exemple nous avons trois groupes, les deux premiers groupes appartiennent à la même section « section A » de la formation « génie logiciel », le dernier groupe appartient à la section « section B » de la formation ASR.

Quartes actions possibles pour chaque emploi du temps, l'ajout (favicon plus), la consultation (favicon calendrier), la modification (favicon crayon) et la suppression (favicon corbeille).

En cliquant sur le favicon 'ajouter' la page 'ajouter EDT' s'affiche comme suit.

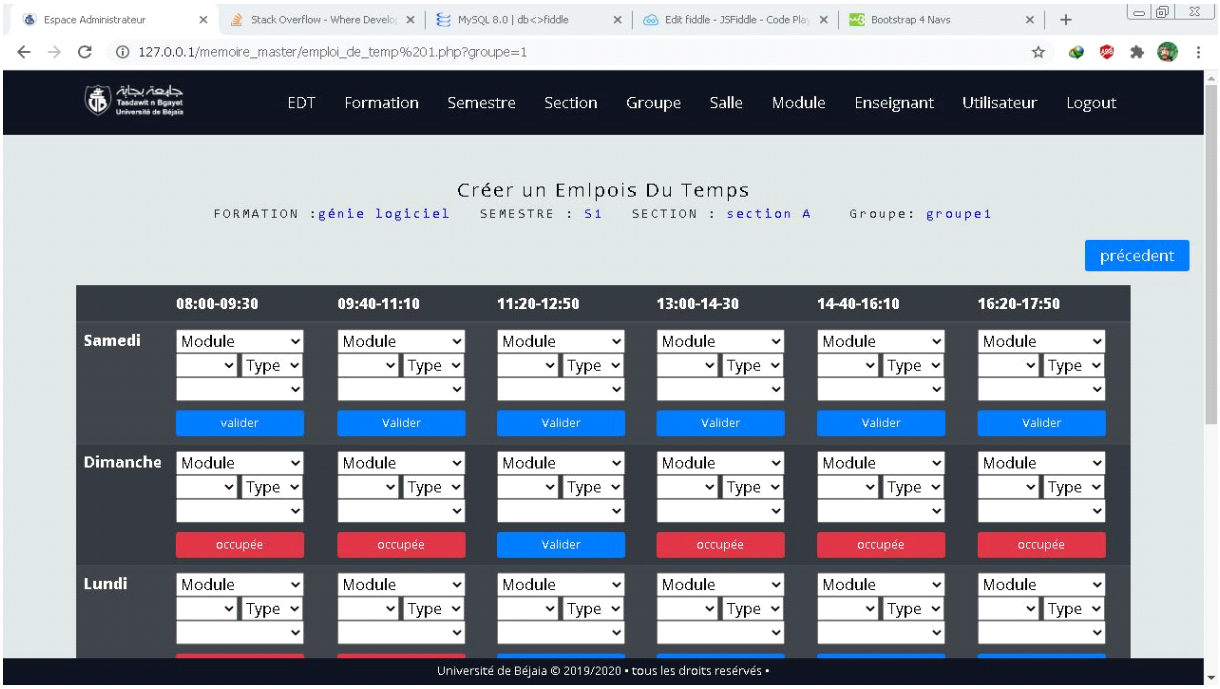

FIGURE 31 INTERFACE « CREER UN EMPLOI DU TEMPS ».

Pour que l'administrateur puisse créer un EDT, il doit choisir le module dans la liste déroulante, sélectionner un enseignant disponible, choisir le type de la séance, puis sélectionner une salle disponible, et enfin, appuyer sur le bouton 'valider'

A noter que la génération des listes des enseignants et des salles disponibles sera automatique, et si le créneau de séance est déjà occupé, un bouton inactif 'occupée' rouge sera affiché.

Si l'administrateur appuie sur le 'favicon modifier' pour modifier un EDT, l'interface suivante sera affichée.

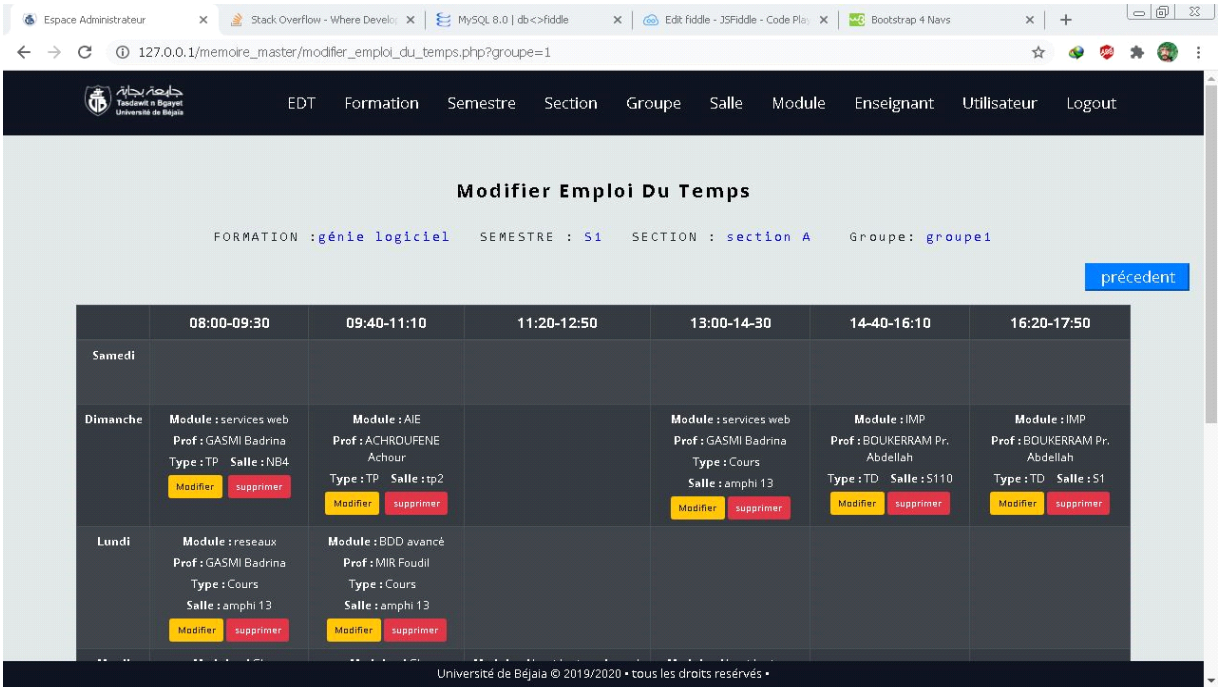

FIGURE 32 INTERFACE « MODIFIER UN EMPLOI DU TEMPS ».

Après avoir créé l'emploi du temps, quatre types seront créés, (EDT groupe, EDT salle, EDT enseignant et EDT section), voici dans la figure suivante un exemplaire de l'emploi du temps d'un groupe

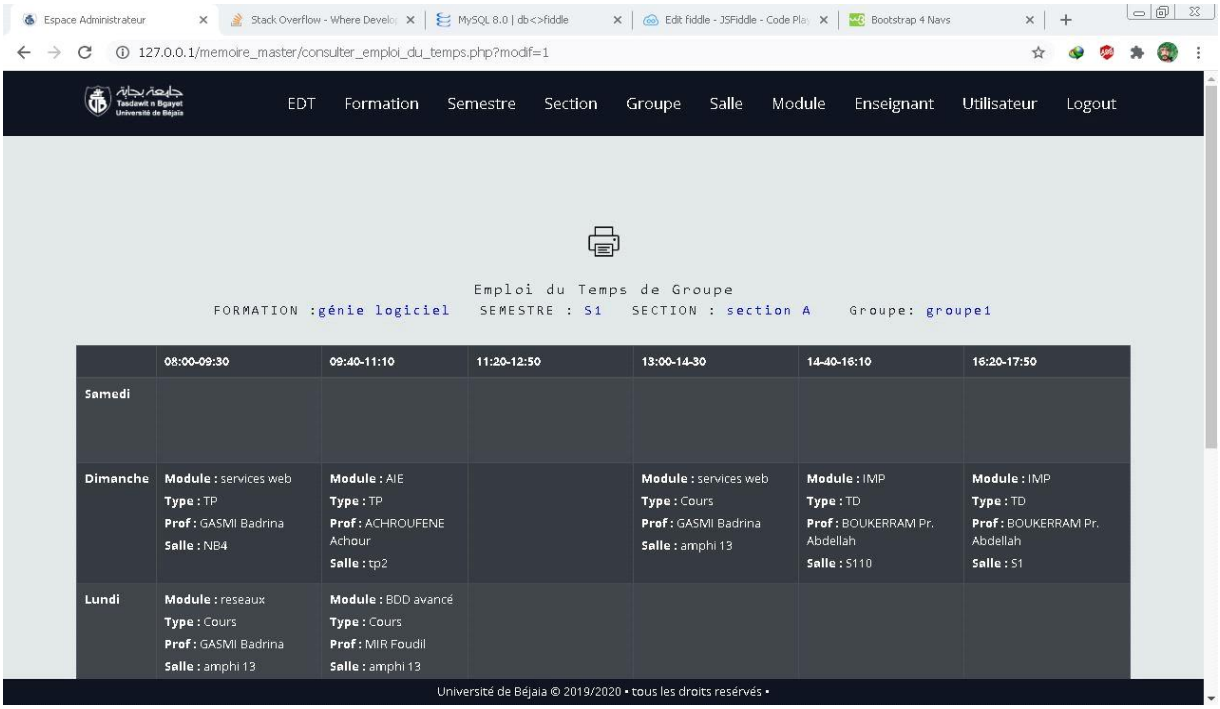

FIGURE 33 INTERFACE « CONSULTER EMPLOI DU TEMPS D'UN GROUPE ».

L'utilisateur peut imprimer cet emploi du temps en cliquant sur le (favicon imprimante),

le résultat après l'impression est dans la figure suivante.

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE LA RECHERCHE SCIENTIFIQUE ET DE L'ENSEIGNEMENT SUPÉRIEUR
université abderrahmane mira de Béjaïa
faculté des sciences exactes: départeement d'informatique

Emploi du Temps de Groupe

FORMATION :génie logiciel SEMESTRE : S1 SECTION : section A Groupe: groupe1

	08:00-09:30	09:40-11:10	11:20-12:50	13:00-14-30	14-40-16:10	16:20-17:50
Samedi						
Dimanche	**Module :** services web **Type :** TP **Prof :** GASMI Badrina **Salle :** NB4	**Module :** AIE **Type :** TP **Prof :** ACHROUFENE Achour **Salle :** tp2		**Module :** services web **Type :** Cours **Prof :** GASMI Badrina **Salle :** amphi 13	**Module :** IMP **Type :** TD **Prof :** BOUKERRAM Pr. Abdellah **Salle :** S110	**Module :** IMP **Type :** TD **Prof :** BOUKERRAM Pr. Abdellah **Salle :** S1
Lundi	**Module :** reseaux **Type :** Cours **Prof :** GASMI Badrina **Salle :** amphi 13	**Module :** BDD avancé **Type :** Cours **Prof :** MIR Foudil **Salle :** amphi 13				
Mardi	**Module :** AGL **Type :** Cours **Prof :** ACHROUFENE Achour **Salle :** amphi 25	**Module :** AGL **Type :** TD **Prof :** ALLEM Khaled **Salle :** S2	**Module :** Algorithmique Avancé **Type :** TD **Prof :** SLIMANI Hachem **Salle :** S110	**Module :** Algorithmique Avancé **Type :** TP **Prof :** SLIMANI Hachem **Salle :** tp2		
Mercredi	**Module :** Algorithmique Avancé **Type :** Cours **Prof :** SLIMANI Hachem **Salle :** amphi 13	**Module :** BDD avancé **Type :** Cours **Prof :** MIR Foudil **Salle :** amphi 13				
Jeudi		**Module :** reseaux **Type :** TP **Prof :** GASMI Badrina **Salle :** tp2		**Module :** AGL **Type :** Cours **Prof :** ACHROUFENE Achour **Salle :** amphi 25	**Module :** AGL **Type :** TD **Prof :** ACHROUFENE Achour **Salle :** S121	

FIGURE 34 IMAGE DE « EDT D'UN GROUPE APRES L'IMPRESSION ».

4.4 Conclusion

Dans ce chapitre, l'aspect pratique et fonctionnel de notre application a été présenté, en décrivant les outils et les logiciels avec lesquels nous l'avons implémenté.

Le résultat final est une application Web réactive qui fait la génération des emplois du temps universitaire en évitant les conflits en termes de ressources humaines et matérielles, visible à travers la présentation de quelques interfaces.

Conclusion générale

L'objectif initial étant de mettre en œuvre un système qui permet de minimiser le travail manuel effectué pour la création d'emploi du temps par le département informatique, nous avons réussi à créer une application semi-automatique avec des interfaces ergonomiques qui permet de faciliter l'accès et la mise à jour des informations pour l'administrateur et la consultation pour l'étudiant et l'enseignant.

Pour la conception de notre système nous avons étudié plusieurs méthodes pour finalement choisir le procédé UP (Unified Process) allant de la collecte d'informations et de l'étude du système existant, passant par l'analyse et la conception du nouveau système, jusqu'à la proposition et la mise en œuvre d'une solution technique. En effet le projet est le résultat d'un travail de quelques mois, ce qui nous a permis, d'une part, de mettre en pratique les connaissances acquises durant notre formation, d'autre part, d'acquérir de nouvelles pratiques techniques.

Nous souhaitons avoir bien résolu le problème de génération d'emploi du temps, néanmoins, il reste des besoins d'amélioration et d'enrichissement; à savoir :

- Trouver une solution qui satisfait les contraintes physiques : 'On ne peut pas mettre plus d'étudiants qu'il n'y a de places dans un local' et 'Le volume horaire total des séances d'un enseignement ne peut pas dépasser le volume prévu'.
- L'extension du system aux autres départements et facultés.
- Utiliser les méthodes d'optimisations pour résolutions les problèmes génération d'EDT comme AG, Tabou, RS…etc.
- Injecter un mécanisme qui permet le traitement automatique et intelligent pour la génération de l'emploi du temps.
- Ajouter quelques fonctions telles que la communication entre les utilisateurs (étudiants et enseignants) ;
- Améliorer l'interface du système.
- Héberger le système dans le serveur web de l'université

Bibliographie

[1] : GOTHA J. *Les problèmes d'ordonnancement. RAIRO - Operations Research - Recherche Opérationnelle*, Tome 27 (1993) no. 1, pp. 77-150. ISSN: Edition papier : 0399-0559, Edition électronique : 1290-3868. Période couverte dans Numdam : 1968-2014. **[En ligne]**. <http://www.numdam.org/item/RO_1993__27_1_77_0/>.

[2] : Touria Ben Rahhou. *Nouvelles méthodes pour les problèmes d'ordonnance-ment cyclique. Automatique / Robotique.* Université Paul Sabatier - Toulouse III, 2013. Français. Tel-00926977. **[En ligne]**. <https://tel.archives-ouvertes.fr/tel-00926977>.

[3] : Houssem Eddine NOURI et Olfa BELKAHLA, *Résolution multi agents du problème d'emplois du temps universitaire. Edition universitaire européennes*. ISBN :978-3-84-17-4241-4.**[Enligne]**.<https://www.researchgate.net/publication/273686717_Resolution_multiagents_du_probleme_d'emploi_du_temps_universitaire>.

[4] : S. Daskalaki, T. Birbas, E. Housos, *An integer programming formulation for a case study in university timetabling*, European Journal of Operational Research 153, p. 117-135. 2004.

[5]: SafaaiDeris, SigeruOmatu, Hiroshi Ohta*, Timetable planning using the constraint-based reasoning*Comp and Oper Res 27, p. 819-840. 2000.

[6]: Teddy Wijaya and RuliManurung, *Solving University Timetabling as a Constraint Satisfaction Problem with Genetic Algorithms (Proceeding) Penerbit*: ICACSIS 2009, Faculty of Computer Science Universitas Indonesia, ISSN: 2086-1796. 2009.

[7]: Ho Sheau Fen Irene, SafaaiDeris and SitiZaitonMohdHashim, *Incorporating Of Constraint-Based Reasoning Into Particle Swarm Optimization For University Timetabling Problem*, ISSR Journals, Vol1. 2009.

[8] : Jacques Ferber, *Les systèmes multi-agents vers une intelligence collective*. InterEditions. 1995.

[9] : ABBES faicel et OMRI abd el ouahebe et COULIBALY souleyman. *La mise au point d'un système de génération automatique de l'emploi du temps basé sur les Systèmes Multi-agents*. Université 08 mai 1945 guelma (2006).

[10] : P. Roques, F. Vallée. *UML en action - 4éme édition, de l'analyse des besoins à la conception*, EYROLLES, 2004.

[11] : J. Conallen. *Concevoir des applications web avec UML*. EYROLLES, 2000.

[12] : P. Roques, F. Vallée. *UML en action - 4éme édition, de l'analyse des besoins à la conception J2EE*.EYROLLES, 2007.

[13]: A. KUSIAK et M. CHEN, *Expert Systems for Planning and Scheduling Manufacturing Systems*, *EJOR,* 1988, *34,* p. 113-130

[14]: J. CARLIER P. CHRÉTIENNE. *Un domaine très ouvert : les problèmes d'ordonnancement, RAIRO. Recherche opérationnelle*, tome 16, no 3 (1982), p. 175-217. **[En ligne].** < http://www.numdam.org/item?id=RO_1982__16_3_175_0 >

Webographie

[W1] : Wikipédia. *Théorie de l'ordonnancement.* **[En ligne].** (Modifié le 27 septembre 2019) Disponible sur :< https://fr.wikipedia.org/wiki/Th%C3%A9orie_de_l%27ordonnancement>.

[W3] : Wikipédia. *Programmation par contraintes.* **[En ligne].** (Modifié le 09 aout 2020) Disponible sur :<https://fr.wikipedia.org/wiki/Programmation_par_contraintes>.

[W4] : UML diagrams. *UML Sequence Diagrams*. **[En ligne].** (Consulté le 21 aout 2020) Disponible sur : <https://www.uml-diagrams.org/sequence-diagrams.html>.

[W5] : UML en français. *Diagramme de classes.* [En ligne]. (Consulté le 15 aout 2020) Disponible sur : <http://uml.free.fr/cours/i-p14.html >.

[W6] : Mémoire Online, conception et réalisation d'une base de données pour la gestion de facturation à l'office congolais de contrôle direction provinciale du Kasaà_occidental, **[En line]**. (Consulté le 18 aout 2020) Disponible sur : <https://www.memoireonline.com/07/10/-3701/conception-et-realisation-dune-base-de-donnees-pour-la-gestion-de-facturation--loffice-con.html>.

[W7] : Wikipedia. *JavaScript.* **[En ligne].** (Modifié le 27 mars 2020) Disponible sur : <https://fr.wikipedia.org/wiki/JavaScript>.

[W8] : Wikipedia. *PHP.* **[En ligne].** (Modifié le 16 mars 2020) Disponible sur : <https://fr.wikipedia.org/wiki/PHP >.

[W9] : FUTURA. *MySQL.* **[En ligne].** (Consulté le 21 septembre 2020) Disponible sur : <https://www.futura-sciences.com/tech/definitions/internet-mysql-4640/ >.

[W10]: Wikipedia. *Bootstrap (front-end framework).* **[En ligne].** (Modifié le 04 juin 2020) Disponible sur : < https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)>.

[W11]: jQueryui, *jQuery user interface.* **[En ligne].** (Consulté le 26 aout 2020) Disponible sur : < https://-jqueryui.com/ >.

[W12] : WampServer. *Header: WampServer.* **[En ligne].** (Consulté le 26 aout 2020) Disponible sur : <https://www.wampserver.com/>.

[W13] : TablePlus. *HeidiSQL vs phpMyAdmin vs TablePlus - A quick comparison.* **[En ligne].** (Modifié le 17 octobre 2018) Disponible sur : <https://tableplus.com/blog/2018/10/heidisql-vs-phpmyad-min-vs-tableplus.html>.

[W14]: Wikipedia. *Sublime Text.* **[En ligne].** (Modifié le 20 mai 2020) Disponible sur : <https://fr.wiki-pedia.org/wiki/Sublime_Text>.

[W15] : Google. *Chrome DevTools.* **[En ligne].** (Modifié le 14 juillet 2020) Disponible sur : <https://-developers.google.com/web/tools/chrome-devtools>.

Abstract

The problem of the timetable is a recurring and complex problem for different institutions and establishments. Our work consists in the design of a semi-automatic application for creating time schedules, until now done manually by our department. The application will also facilitate access to data as well as consultation.

Keywords: Timetable, application, semi-automatic, management, planning.

Résumé

Le problème de l'emploi du temps est un problème récurrent et complexe pour différentes institutions et établissements. Notre travail consiste en la conception d'une application semi-automatique de création d'emplois du temps, jusqu'ici faits manuellement par notre département. L'application permettra aussi de faciliter l'accès aux données ainsi que la consultation.

Mots clés: Emploi du temps, application, semi-automatique, gestion, planification.

Agzul

Ugur n-umsqdec n-ukud d-ugur yessawen-en i-w-atas n-tisuddiwin. Leqdic-nneɣ d-ax^{w}eddim n-usnas id-i-slulayn amsqdec n-ukud aken ad-y-sifses lx^{w}ddma n-ymseqdcen n-ukud iw-gezdu n tasenselkimt, daɣen it ɛemmid aseddu akk d-usenfel, asnas neɣ yesifses asadf n-tmuca ak d-usfiqeḍ.

Awal ufrir: Amsqdec n-ukud, asnas, a-slal, asefrek, asnflul.