الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :……

Centre Universitaire
Abd elhafid boussouf Mila

Institut des sciences et de la technologie Département de Mathématiques et Informatique

Mémoire préparé En vue de l'obtention du diplôme de Licence
En: Filière informatique

Emploi du temps électronique d'utilisation des locaux pédagogique dans l'université

Préparé par : BENDAS Mohamed Amin
LAIB Moussa
CHERIAK Mouad

Encadrer par : BOUCHEKOUF Asma

Année universitaire :2014/2015

Remerciements

Après avoir terminé ce mémoire de fin d'étude, nous Réservons ces lignes

Pour exprimer nos remerciements les plus sincères à notre dieu

Tout puissant de nous avoir donné la capacité et la patience pour terminer

Ce travail Nous remercions tout d'abord

ALLAH

Nous tenons à exprimer nos remerciements

à notre encadreur Mlle BOUCHEKOUF Asma,

Ses conseils, Sa disponibilité et ses encouragements qui nous ont

Permis de réaliser ce travail dans les meilleures conditions.

Nous adressons aussi nos reconnaissances à tous les professeurs qui depuis

Quelques années leurs conseils et leurs connaissances qui nous ont bien servis.

Nous voudrions aussi exprimer notre gratitude envers tous ceux qui nous

ont accordé.

Leurs soutien, tant par leurs gentillesse que par leurs dévouement.

Nous ne pouvons nommer ici tout les personnes qui de près ou de loin qui nous

ont aidé.

Enfin nous tenons à dire combien le soutien quotidien de notre famille a été

Important tout au long de ces quelques années, nous leur dois

beaucoup.

Dédicace

A Ma Mère,

A Mon Père,

A mes chers frères et sœurs,

A Tous les Membres de Ma Famille,

A tous mes Amis et les Collègues de Promotion,

A M.Mamine

Je dédie ce modeste travail.

Résumé :

Malgré l'avancée technologique spectaculaire qu'a connu l'informatique, les instituts du centre universitaire -Abdalhafid Boussouf Mila- manipule les emplois du temps manuellement, l'objectif de notre projet est d'automatisé cette manipulation.

A travers ce projet nous avons réalisé une application de gestion emploi du temps qui facilite la création de ses tableaux concernant les conditions de répétition des enseignant, salle ou groupes dans la même séance et le volume horaire et les jour de travail des enseignants.

ملخص:

رغم التقدم التكنولوجي الملحوظ الذي عرفه الإعلام الآلي،لا تزال معاهد المركز الجامعي -عبد الحفيظ بوالصوف ميلة- تعالج التوقيت الأسبوعي بطريقة يدوية، الهدف من مشروعنا هو تألية هذه المعالجة.

بفضل هذا المشروع استطعنا تطوير برنامج يسهل عملية إنشاء التواقيت الأسبوعية مع الأخذ بعين الاعتبار شروط : عدم إمكانية تكرار الأستاذ، القاعة أو الفوج في نفس الحصة، كذلك الحجم الساعي و أيام العمل لكل أستاذ.

Summary:

Although the remarkable technological development of domain informatique, the institute of the University of Abdalhafid Bousouf processing time table still in manual way, the goal of our project is to make process automatic.

Due to this project we could create software to simplify the process of making the time tables we take in account the possibility of repetition of the professor, the class room or the group in the same session, also the amount of time and days for each professor.

Sommaire :

Chapitre 1 : Les étapes de développement des logiciels

Chapitre 2 : Langage de modélisation UML

Chapitre4 : Analyse et conception du projet :

Chapitre 5 : Implémentation

Liste des tableaux :

Liste des figures :

Introduction générale :

Avant l'invention de l'ordinateur, on enregistrait toutes les informations manuellement sur des supports en papier ce qui engendrait beaucoup de problèmes tels que la perte de temps considérable dans la recherche de ces informations .

Il ne fait désormais plus aucun doute que l'informatique est la révolution la plus importante qui a marqué la vie de l'humanité moderne. En effet, les logiciels informatiques proposent maintenant des solutions à tous les problèmes de la vie, aussi bien dans les domaines professionnels que pour les applications personnelles.

La finalité de notre projet est de développer un logiciel de système d'information pour la gestion d'un emploi du temps électronique d'utilisation des locaux pédagogique dans l'université. L'idée proposée donc consiste à faire l'automatisation de la création des emplois du temps au niveau de département, ces tableaux a été crée manuellement utilisant Excel ce qui provoque des erreurs d'interférence.

Nous nous sommes basés sur l'application de la méthode orientée objet utilisons le langage de programmation Java, aussi utilisons le standard des langages objet qui est la notation UML (Unified Modeling Language).

Notre travail est organisé en cinq chapitres principaux, qu'on peut résumer comme suit :

Dans le chapitre 1 on va présenter les nombreuses étapes de développement des logiciels.

Dans le chapitre 2 on va présenter le langage de modélisation UML d'une façon générale vu à son importance avant le développement des logiciels.

Dans le chapitre 3 on va présenter quelque langages de programmation les plus important et les plus utiliser (Java, Python, C ++…) et outils de travail que nous avons utilisés.

Dans le chapitre 4 on va présenter les scénarios et les diagrammes UML (cas d'utilisation, séquence, activité, et le diagramme de classe) qui peut nous aider pour programmer notre application.

Dans le chapitre 5 on va capturer et expliquer les interfaces du logiciel.

Chapitre 1 :

Les étapes de développement des logiciels

1.1. Introduction :

Dans ce chapitre nous allons présenter les nombreuses étapes de développement des logiciels. Au début on va parler à les bases de qualité de logiciel, puis le cycle de vie du logiciel avec ses phases, de la phase préliminaire jusqu'à la phase poste production.

Définition du logiciel (software) :

Un logiciel est un ensemble de programmes, qui permet à un ordinateur ou à un système informatique d'assurer une tâche ou une fonction en particulier (exemple: logiciels d'application, systèmes d'exploitation,….etc). Les logiciels, suivant leur taille, peuvent être développés par une personne seule, une petite équipe, ou un ensemble d'équipes coordonnées. [3]

1.2. Les bases de qualité de logiciel :

En génie logiciel, divers travaux ont mené à la définition de la qualité du logiciel en termes de facteurs, qui dépendent, entre autres, du domaine de l'application et des outils utilisés. Parmi ces derniers nous pouvons citer :

- Validité(Exactitude) : aptitude d'un produit logiciel à remplir exactement ses fonctions, définies par le cahier des charges et les spécifications.
- Fiabilité (robustesse) : aptitude d'un produit logiciel à fonctionner dans des conditions anormales (P.e. données erronées, format de données incorrect, problème de réseau, . . .).
- Extensibilité (maintenance) : facilité avec laquelle un logiciel se prête à sa maintenance, c'est-à-dire à une modification ou à une extension des fonctions qui lui sont demandées.
- Réutilisabilité : aptitude d'un logiciel à être réutilisé, en tout ou en partie, dans de nouvelles applications.
- Compatibilité : facilité avec laquelle des éléments logiciels peuvent être combinés avec d'autres.
- Efficacité : est la capacité d'un système logiciel d'utiliser le minimum de ressources matérielles que ce soit le temps machine, l'espace mémoire ou la bande passante de moyens de communication.
- Portabilité : facilité avec laquelle un logiciel peut être transféré sous différents environnements matériels et logiciels.
- Intégrité : aptitude d'un logiciel à protéger son code et ses données contre des accès non autorisés.

- Facilité d'emploi : facilité avec laquelle des personne présentant des compétences et des formations différentes peuvent apprendre à utiliser le produit logiciel. [3]

1.3. Cycle de vie du logiciel

1.3.1. Phase préliminaire :

Identification des besoins :

- **Type des besoins :**
  - **Les besoins fonctionnels** : Il s'agit des fonctionnalités du système. Ce sont les besoins spécifiant un comportement d'entrée / sortie du Système.
  - **Les besoins techniques** : Convivialité et simplicité d'utilisation de l'application:

L'application devra pouvoir être simplement utilisée. Elle doit avoir donc une interface graphique claire confortable.

- **Le temps de réponse** : Le temps de réponse doit être acceptable.

1.3.2. Phase d'études :

Analyse des besoins :

Lors de la phase d'analyse, on analyse les besoins de l'utilisateur et on définit ce que le logiciel devra faire. Le résultat de la phase d'analyse est consigné dans un document appelé cahier des charges du logiciel. Le cahier des charges décrit les fonctionnalités attendues du produit ainsi que les contraintes non fonctionnelles (temps de réponse, contraintes mémoire…). [3]

1.3.3. Phase de réalisation :

a. Lancement de projet :

Cette phase consiste notamment à un ajustement des prévisions de charge et de planning produites dans les étapes précédentes. Une planification précise peut maintenant être effectuée puisque toutes les données du problème sont supposées connues. Un accord entre l'utilisateur et le réalisateur du produit sur cette planification est nécessaire. Par ailleurs, une communication sur le projet

auprès des utilisateurs proches ou lointains du futur produit n'est pas à négliger.[3]

b. Développement :

C'est une phase dense pour le réalisateur du produit qui doit vérifier que la planification prévue est strictement suivie, mais qui doit aussi parer aux aléas inévitables : maladies, congés des développeurs, pannes de ressources, etc… ces aléas sont courant (bien plus que l'on croit) et ils doivent être prise en compte dans l'analyse des risques.

La phase de développement doit prévoir :

- Une relation permanente avec le futur utilisateur pour l'informer de l'avancement des travaux.
- Une mise en œuvre de la documentation relative au projet. [1]

c. La conception d'un système :

La conception d'un système précise comment le système va atteindre l'objectif. La conception consiste à concevoir des activités, les spécifications d'un système, qui répondent aux exigences fonctionnelles déterminées lors de l'analyse. Ces spécifications servent à l'élaboration des logiciels, à l'acquisition du matériel, à l'essai du système et à d'autres activités de la phase de mise en application.

Il existe trois principes de la conception sont:

- l'interface utilisateur.
- les données.
- les traitements. [2]

1. La conception de l'interface utilisateur:

A cette étape, on se concentre sur les interactions entre les utilisateurs et L'ordinateur. Les concepteurs travaillent surtout sur les méthodes d'entrée/sortie et sur La conversion des données et des informations entre des supports compréhensibles par l'être humains et des supports compréhensibles par la machine. La conception de l'interface utilisateur se fait souvent au moyen du prototypage. Dans le prototypage on conçoit des maquettes, ses méthodes d'interface que l'on modifie en fonction des commentaires des utilisateurs.

La Conception de l'interface utilisateurs permet de définir des spécifications détaillées sur les produits informatifs tels que les affichages sur écran, les dialogues interactifs entre l'ordinateur et l'utilisateur. [2]

2. La conception des données:

L'activité principale de la conception des données consistées à élaborer la structure des bases de données et des fichiers qu'utilisera le système. La conception des données donne souvent bien à un dictionnaire de données, (un catalogue des descriptions), détaillées sur :

- les attributs, ou les caractéristiques des entités (objets, personnes, lieux, événements) sur lesquels le système conservera des données.
- les relations existantes entre ces entités.
- les éléments d'information (bases de données, fichiers, enregistrements) que l'on devra conserver sur chaque entité suivie par le système d'information.
- les règles d'intégrité qui précisent la façon dont on précise et on utilise chaque élément d'information dans le système.

3. La conception du traitement:

La conception du traitement est centrée sur la conception des ressources logicielles elle donne lieu à l'ensemble des spécifications des programmes et des procédures requis pour répondre aux spécifications de l'interface utilisateur et des données. La conception du traitement doit aussi produire des spécifications qui répondent aux exigences du contrôle fonctionnel et du contrôle de la performance déterminées à l'étape d'analyse. [2]

d. Le codage (implémentation) :

Lors de cette phase, la conception est traduite dans un langage de programmation. Il faut également préparer les données nécessaires à l'exploitation du logiciel. La phase d'implémentation fournit donc les composants logiciels sur support informatique. [3]

e. Test du logiciel :

Le test est un processus qui vise à établir qu'un système vérifie les propriétés exigées par sa spécification, ou à détecter des différences entre les résultats engendrés par le système et ceux qui sont attendus par la spécification. Les tests ont pour but de mettre en évidence les erreurs. Ils peuvent prouver la présence d'erreurs mais ne peuvent pas prouver leur absence. Généralement 40% du budget global est consacrée à l'effort de test. [3]

Techniques de test :

Plusieurs techniques qui dépendent de l'objectif du test. Mais aucune technique ne sera jamais complète.

- **Les tests « boîte noire »**

On considère seulement l'aspect fonctionnel du programme (ce que doit faire), sans considérer sa structure interne. On s'appuie principalement sur les données et les résultats. Le programme est exécuté pour des données en entrée, les résultats obtenus sont comparés aux résultats prédits pour déterminer le succès du test.

La principale difficulté réside dans la sélection d'un ensemble adéquat de valeurs de test. L'espace des données en entrée du programme est généralement beaucoup trop grand pour être testé intégralement (souvent même infini). Une technique souvent utilisée consiste à partitionner cet espace en classes d'équivalences qui devraient avoir le même comportement, et on teste aux limites des classes. Elle se base sur l'observation que les erreurs arrivent souvent aux limites des domaines de définition des variables du programme. Il faudra prendre en compte aussi bien les données valides que non-valides. [3]

- **Les tests « boite blanche »**

En opposition, les tests « boite blanche » ouvrent la « boite » et examinent la structure interne du programme.

L'objectif est d'assurer que:

- Toutes les conditions d'arrêt de boucle ont été vérifiées.
- Toutes les branches d'une instruction conditionnelle ont été testées.
- Les structures de donne interne ont été testées. [3]

f. L'installation :

Après avoir intégré le logiciel, on peut l'installer dans son environnement d'exploitation, ou dans un environnement qui simule cet environnement d'exploitation, et le tester pour s'assurer qu'il se comporte comme requis dans la spécification élaborée lors de la phase d'analyse (validation). [3]

1.4. Phase poste production :

1.4.1. Mise en œuvre :

Appelée quelquefois phase de mise en production, la phase de mis en œuvre consiste en l'installation dans les locaux et sur les matériels du commanditaire du produit logiciel développé. La phase comprend également généralement des utilisateurs finaux. [1]

1.4.2. Maintenance :

Cette phase recouvre toutes les mesures à prendre pour que l'application installée continue à fonctionner et puisse évoluer. Cette phase peut faire l'objet d'un contrat de prestation complémentaire avec le réalisateur du projet. Par ailleurs, cette dernière se doit de conserver la documentation du projet : document fonctionnels et document techniques, non seulement en cas d'intervention ultérieure sur le produit mais aussi dans un but de capitalisation des connaissances (mémoire de l'entreprise). [1]

1.5. Conclusion :

Les étapes de développement des logiciels que nous avons présentées dans ce chapitre peut nous aidés pour ordonner notre procédure de développement du logiciel, faciliter ses étapes et son maintenance et peut réduire le temps de réalisation du projet.

Chapitre 2 : Langage de modélisation UML

2.1. Introduction au langage de modélisation UML :

Puisque nous avons besoin d'un langage de modélisation avant de développer un logiciel, nous ferons dans ce chapitre tout d'abord un survol sur quelques généralités du langage de modélisation unifié UML (Unified Modeling Language), par suite, nous allons présenter les diagrammes les plus importants et ses utilisation.

2.1.1. Définition :

UML (Unified Modeling Language) se définit comme un langage de modélisation graphique qui permet la spécification, la représentation et la construction des composantes d'un système informatique.

2.1.2. Historique :

- Les années 1980: Utilisation de méthodes adaptées à la programmation impérative (notamment Merise).
- Fin 80 / début 90 la programmation par objets prend de l'importance.
- Conséquence naturelle: mise en place de méthodes orientées objet. Plus de cinquante méthodes apparaissent entre 1990 et 1995:
  - OOD: Object Oriented Design Booch, 1991.
  - HOOD : Hierarchical Object Oriented Design Delatte et al., 1993
  - OOA : Object Oriented Analysis Schlaer, Mellor, 1992
  - OOA/OOD : Coad, Yourdon, 1991
  - OMT : Object Modeling Technique Rumbaugh, 1991
  - OOSE : Object Oriented Software Engineering Jacobson, 1992
  - OOM : Object Oriented Merise Bouzeghoub, Rochfeld, 1993
  - Fusion Coleman et al, 1994
- 1994 : le consensus se fait autour de trois méthodes :
  - OMT (Object Modeling Technique) de James Rumbaugh fournit une représentation graphique des aspects statique, dynamique et fonctionnel d'un système.
  - OOD (Object Oriented Design) de Grady Booch, définie pour le Département of Defense, introduit le concept de paquetage (package).
  - OOSE (Object Oriented Software Engineering) d'Ivar Jacobson fonde l'analyse sur la description des besoins des utilisateurs (cas d'utilisation, ou use cases). [4]

- 1995 : Fusion des 3 principales méthodes pour définir un langage de modélisation commun: UML (Unified Modeling Language).

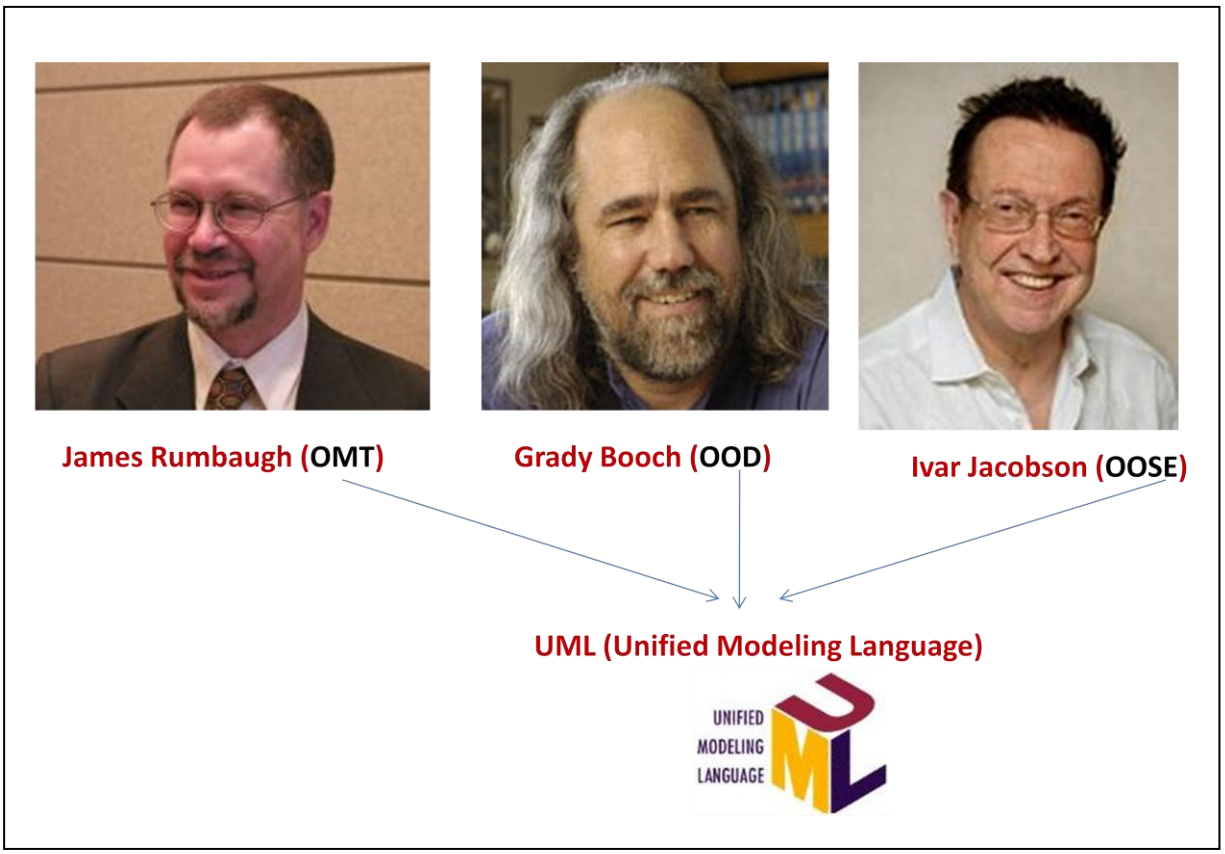

Figure 2.1. Les premiers réalisateurs d'UML

2.1.3. Définition des diagrammes :

Sont des éléments graphiques qui décrivent le contenu des vues.

1. Les différents diagrammes d'UML :

Il existe 13 diagrammes dans 2 types :

2. Diagrammes structurelle (statiques) :

- Diagramme de classes (Class diagram).
- Diagramme d'objets (Object diagram).
- Diagramme de composants (Component diagram).
- Diagramme de déploiement (Deployment diagram).
- Diagramme de paquetages (Package diagram).
- Diagramme de structures composites (Composite structure diagram).

3. Diagrammes comportementale (dynamiques) :

- Diagramme de cas d'utilisation (Use case diagram)
- Diagramme d'activités (Activity diagram)
- Diagramme d'états-transitions (State machine diagram)
- Diagramme de séquence (Sequence diagram)
- Diagramme de communication (Communication diagram)
- Diagramme global d'interaction (Interaction overview diagram)
- Diagramme de temps (Timing diagram)

Les diagrammes plus utiles sont :

- Le diagramme de cas d'utilisation.
- Le diagramme de séquences.
- Le diagramme d'activités.
- Le diagramme d'états-transitions.
- Le diagramme de classes.
- Le diagramme de composants.
- Le diagramme de déploiement. [4]

2.1.4. Les points forts d'UML:

- UML est un langage formel et normalisé.
- Gain de précision.
- Encourage l'utilisation d'outils.
- UML est un support de communication performant.
- Il facilite la compréhension des représentations abstraites complexes.
- Son caractère polyvalent et sa souplesse en font un langage universel. [5]

2.1.5. Les points faibles d'UML:

- La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation.
- UML n'est pas une méthode dans la mesure où elle ne présente aucune démarche.

2.2. Diagramme de cas d'utilisation :

Le diagramme de cas d'utilisation représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système.

2.2.1. Rôle du diagramme d'utilisation :

- Donne une vue du système dans son environnement extérieur.
- Définit la relation entre l'utilisateur et les éléments que le système met en œuvre.
- Est la base du modèle UML.

2.2.2. Les éléments d'un diagramme de cas d'utilisation :

1. Acteur :

Un acteur est l'archétype de l'utilisateur (personne, processus externe,...) qui interagit avec le système.

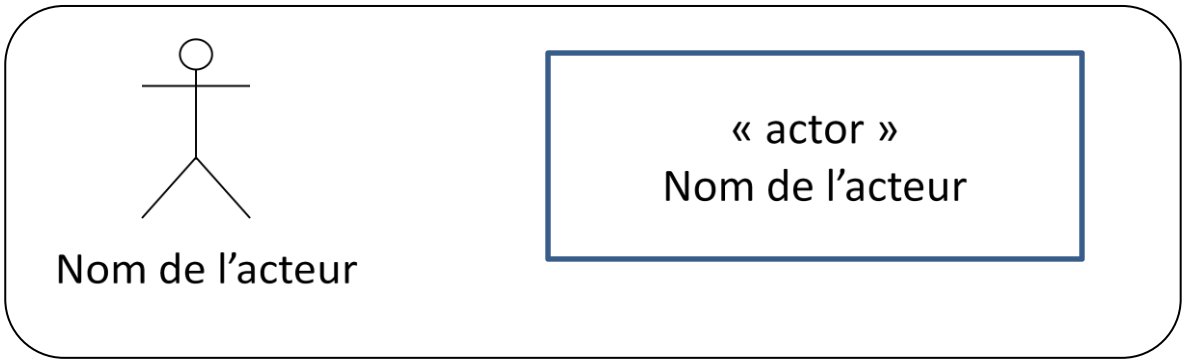

Figure 2.2. Représentation d'un acteur

2. Cas d'utilisation :

Un cas d'utilisation (« use case ») représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier.

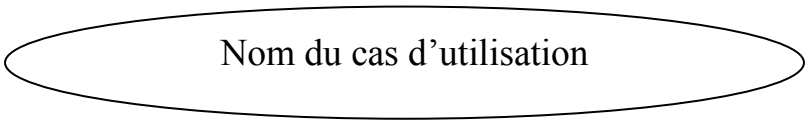

Figure 2.3. Représentation du cas d'utilisation

2.2.3. Les relations dans un diagramme de cas d'utilisation :

Un type de relation d'association est un lien de communication entre un acteur et un cas d'utilisation. [4]

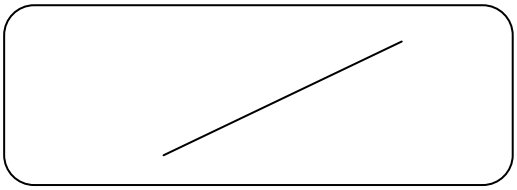

Figure 2.4. Relation d'association dans un diagramme de cas d'utilisation

2.2.4. Relations entre cas d'utilisation :

Pour affiner le diagramme de cas d'utilisation, UML définit trois types de relations standardisées entre cas d'utilisation :

1. **Relation d'inclusion :**

Une relation d'inclusion, formalisée par le mot-clé <<include>> : le cas d'utilisation de base en incorpore explicitement un autre, de façon obligatoire.

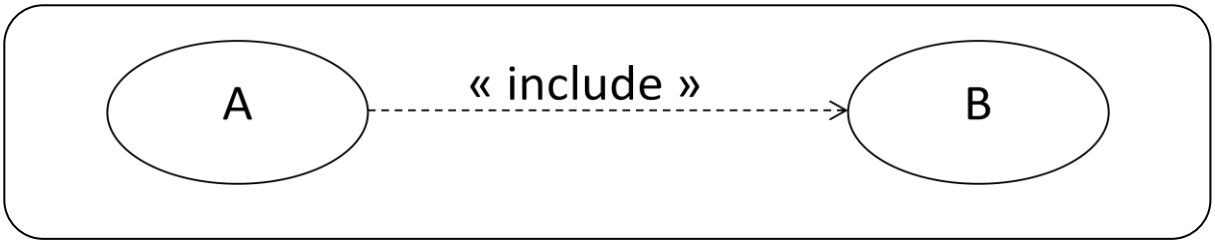

Figure 2.5. Relation d'inclusion dans un diagramme de cas d'utilisation

2. **Relation d'extension :**

Une relation d'extension, formalisée par le mot-clé <<extend>> : le cas d'utilisation de base en incorpore implicitement un autre, de façon optionnelle.

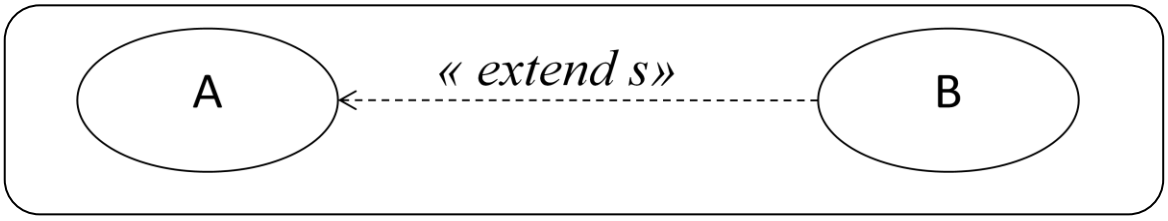

Figure 2.6. Relation d'extension dans un diagramme de cas d'utilisation

3. Relation de généralisation/spécialisation :

Les cas d'utilisation descendants héritent de la description de leur parent commun. Chacun d'entre eux peut néanmoins comprendre des interactions spécifiques supplémentaires.

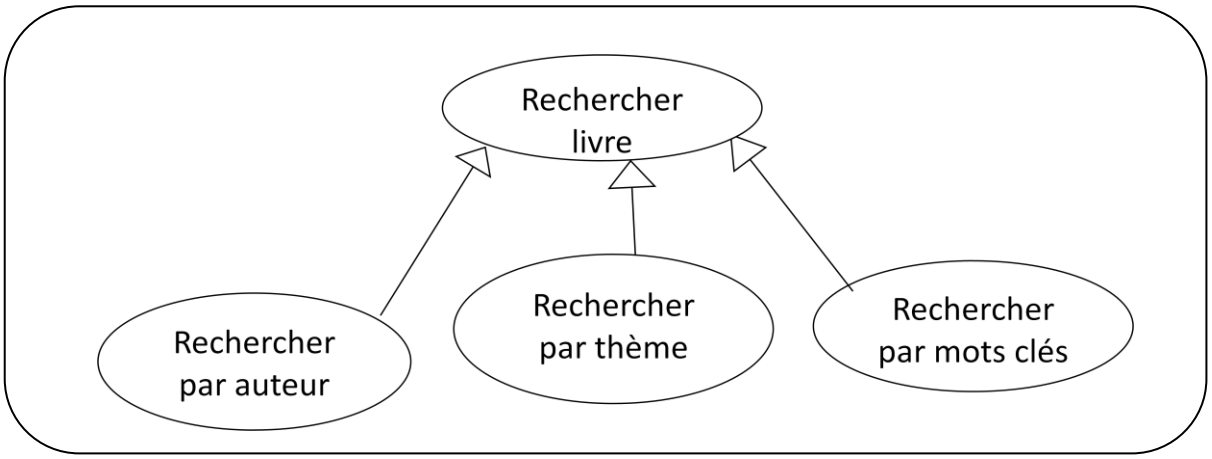

Figure 2.7. Relation de généralisation dans un diagramme de cas d'utilisation

2.2.5. Types de relations :

Notez les trois différents types de relations présentes sur :

- **l'association** (trait plein avec ou sans flèche, comme sur la figure 4) entre acteurs et cas d'utilisation
- **la dépendance** (flèche pointillée) entre cas d'utilisation, avec les mots-clés «extend» ou «include»
- **la relation de généralisation** (flèche fermée vide) entre cas d'utilisation. [3]

2.2.6. Méthodologie :

1. Identification des acteurs :

Les acteurs candidats sont systématiquement :

- Les utilisateurs humains directs
- Les systèmes informatiques externes au système qui interagissent avec lui.
- Les périphériques manipulés par le système (imprimantes,…).
- Ne pas confondre acteurs et utilisateurs
  - Plusieurs utilisateurs qui jouent le même rôle correspondent à un seul acteur. client d'une banque.
  - Un utilisateur qui joue des rôles différents vis-à-vis du système correspond à plusieurs acteurs.

- Les acteurs communiquent directement avec le système.
  - Une erreur fréquente consiste à répertorier en tant qu'acteur des entités externes qui n'interagissent pas directement avec le système, mais uniquement par le biais d'un des véritables acteurs.
- Éliminez les acteurs «physiques » au profit des «logiques».
  - Par exemple, il est peu judicieux de considérer comme acteurs: les terminaux informatiques (écrans, claviers, imprimantes, etc.).
- Pour faciliter la recherche des acteurs, on peut imaginer les frontières du système. Tout ce qui est à l'extérieur et qui interagit avec le système est un acteur ; tout ce qui est à l'intérieur est une fonctionnalité du système que le maître d'œuvre doit réaliser.
- Le nom de l'acteur doit refléter son rôle. Exemple : pour un logiciel de gestion de paie, le nom correct d'un acteur est Comptable plutôt que M. Kamel. [4]

2. Identification des cas d'utilisation:

- L'ensemble des cas d'utilisation doit décrire exhaustivement les exigences fonctionnelles du système. Chaque cas d'utilisation correspond donc à une fonction métier du système, selon le point de vue d'un de ses acteurs.
- Pour identifier les cas d'utilisation, il faut utiliser le point de vue de chaque acteur et déterminer comment et surtout pourquoi il se sert du système.
- Un cas d'utilisation a toujours au moins un acteur principal pour qui le système produit un résultat observable, et éventuellement d'autres acteurs ayant un rôle secondaire.
- Il faut éviter les redondances et limiter le nombre de cas en se situant au bon niveau d'abstraction (par exemple, ne pas réduire un cas à une action).
- L'étape de recueil des besoins est souvent longue et fastidieuse car les utilisateurs n'ont qu'une vague idée de ce que leur apportera un futur système; en outre, le cahier des charges contient des imprécisions, des oublis, voire des informations contradictoires difficiles à extraire. [4]

2.3. Diagramme de séquence :

2.3.1. Présentation :

- Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et les objets du système selon un ordre chronologique.
- Ils permettent de représenter un système pendant son exécution.
- Utilisés à tous les niveaux : Entre un acteur et le système ou Entre des objets.[4]

2.3.2. Eléments du diagramme de séquence :

1. Ligne de vie :

Représentation de l'existence d'un élément participant dans un diagramme de séquence. Cela peut être un acteur ou le système en modélisation d'exigences, des objets logiciels en conception préliminaire ou conception détaillée.

2. Message :

Élément de communication unidirectionnel entre objets qui déclenche une activité dans l'objet destinataire. La réception d'un message provoque un événement dans l'objet récepteur. La flèche pointillée représente un retour au sens UML. Cela signifie que le message en question est le résultat direct du message précédent. [6]

A. Types de messages :

- **Messages synchrones** : L'émetteur reste bloqué le temps que le récepteur traite le message envoyé et envoie la réponse.

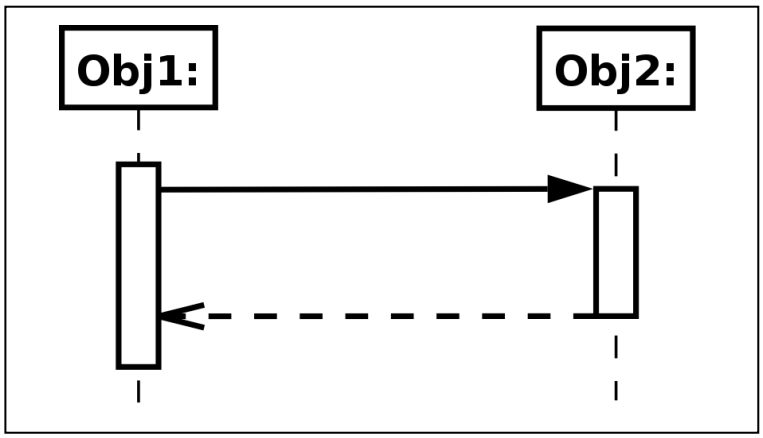

Figure 2.8. Messages synchrones dans un diagramme de séquence

- **Messages asynchrones** : L'émetteur n'est pas bloqué lorsque le récepteur traite le message envoyé.

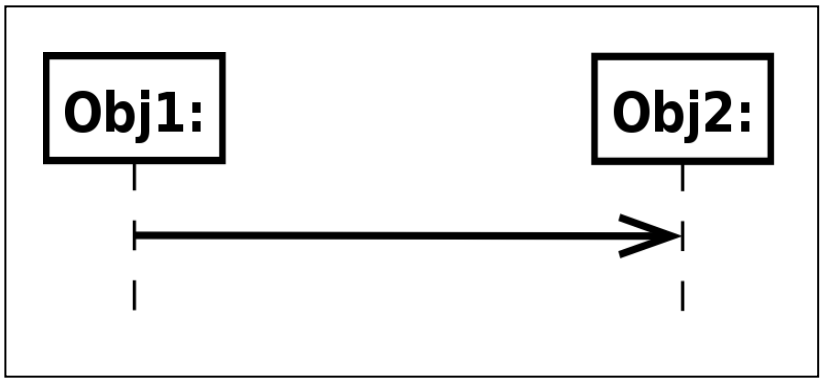

Figure 2.9. Messages asynchrones dans un diagramme de séquence

- **Messages de création et destruction d'instance** : La destruction d'un objet est matérialisée par une croix qui marque la fin de la ligne de vie de l'objet.

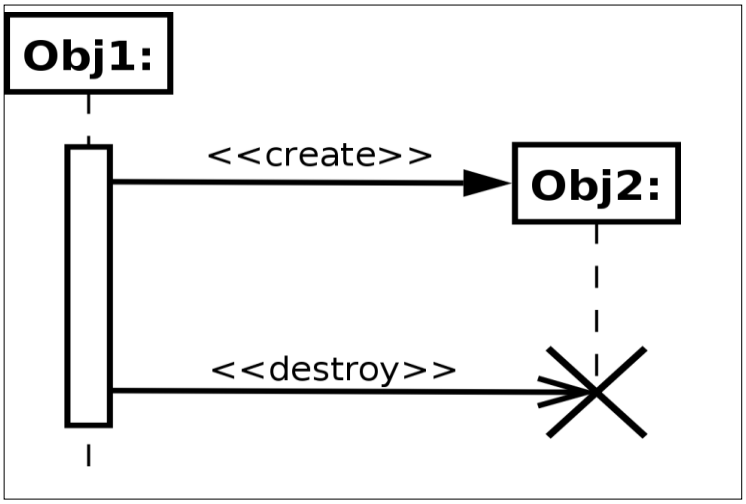

Figure 2.10. Messages de création dans un diagramme de séquence

B. Syntaxe d'un message:

Un message est habituellement spécifié sous la forme suivante : <msg>([<par>]).

- <msg> est le nom du message.
- <par> désigne les paramètres (optionnels) du message.

2.3.3. Les cadres d'interaction :

- Les cadres d'interaction permettent de décrire des diagrammes de séquence de manière compacte.
- Les cadres d'interaction permettent particulièrement d'indiquer qu'un groupe de messages est optionnel (mot-clé opt), répété (mot-clé loop) ou alternatif (mot-clé alt).
- Un cadre d'interaction est représenté dans un rectangle dont le coin supérieur gauche contient un pentagone.
- Un cadre peut être divisé en fragments. Les fragments d'un opérateur d'interaction sont séparés par des lignes pointillées.
- Les conditions de choix des opérandes sont données entre crochets ([ ]).

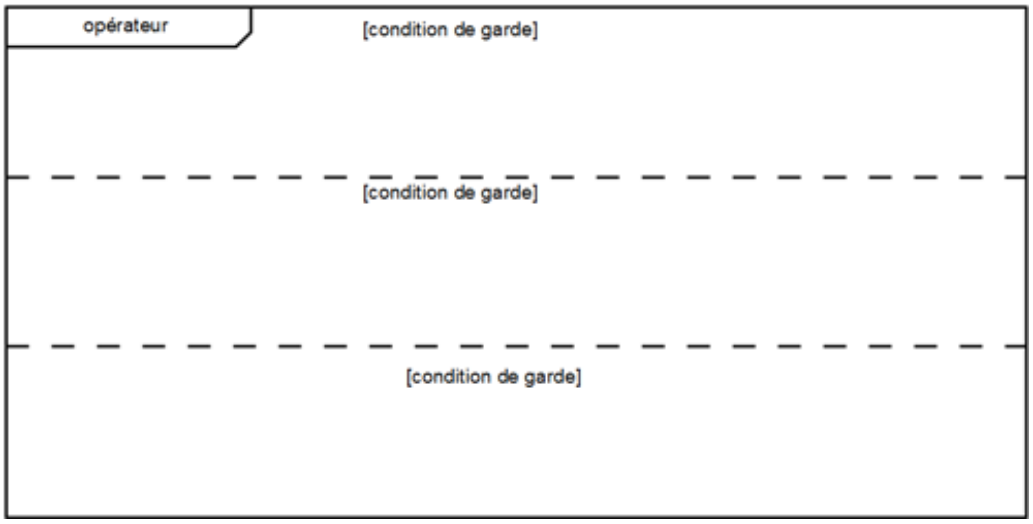

Figure 2.11. Cadres d'interaction

1. Les principaux opérateurs :

- Loop (boucle): Le fragment peut s'exécuter plusieurs fois tant que la condition de garde explicite l'itération est vraie.

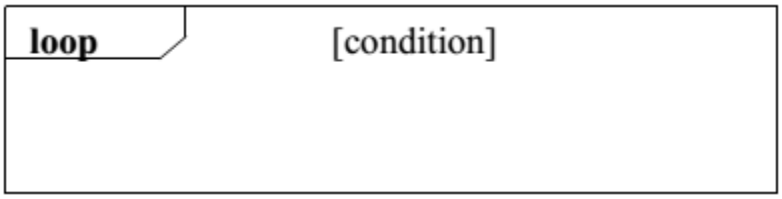

Figure 2.12. L'opérateur Loop

- Opt (optionnel) : Le fragment ne s'exécute que si la condition fournie est vraie.

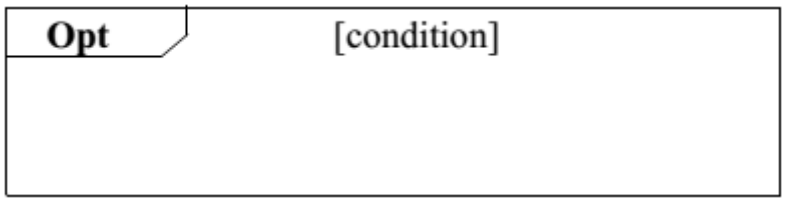

Figure 2.13. L'opérateur Opt

- Alt (fragments alternatifs): Seul le fragment possédant la condition vraie s'exécutera.

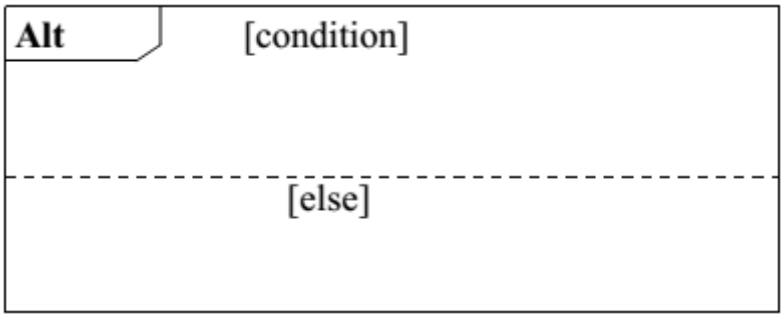

Figure 2.14. L'opérateur Alt

- Ref (Référence) : Utilisé pour référencer un autre diagramme de séquence (appel procédural).
- Par (Parallèle): montrent les activités conduites en parallèle.

2. Stéréotypes de Jacobson :

A. Les classes entités <<entity>> : Ne comportent que des attributs (elles ne possèdent pas des opérations).

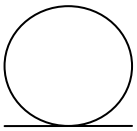

Figure 2.15. Représentation d'un Entité

B. Les classes de contrôles <<control>> : Ne comportent que des opérations (elles ne possèdent pas des attributs). Ils implémentent les fonctionnalités de l'application.

Figure 2.16. Représentation d'un control

C. Les classes dialogues (les fenêtres graphiques) :

Les dialogues comportent des attributs et des opérations.

- Règles sur les interactions possibles entre les instances des trois types de classes:
  - Les acteurs ne peuvent interagir (envoyer des messages) qu'avec les dialogues.
  - Les dialogues peuvent interagir avec les control ou exceptionnellement avec d'autres dialogues.
  - Les contrôles peuvent interagir avec les dialogues, les entité, ou d'autres contrôles.
  - Les entités ne peuvent interagir qu'entre elles. Requeste

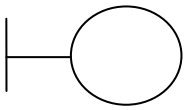

Figure 2.17. Représentation d'un Dialogues

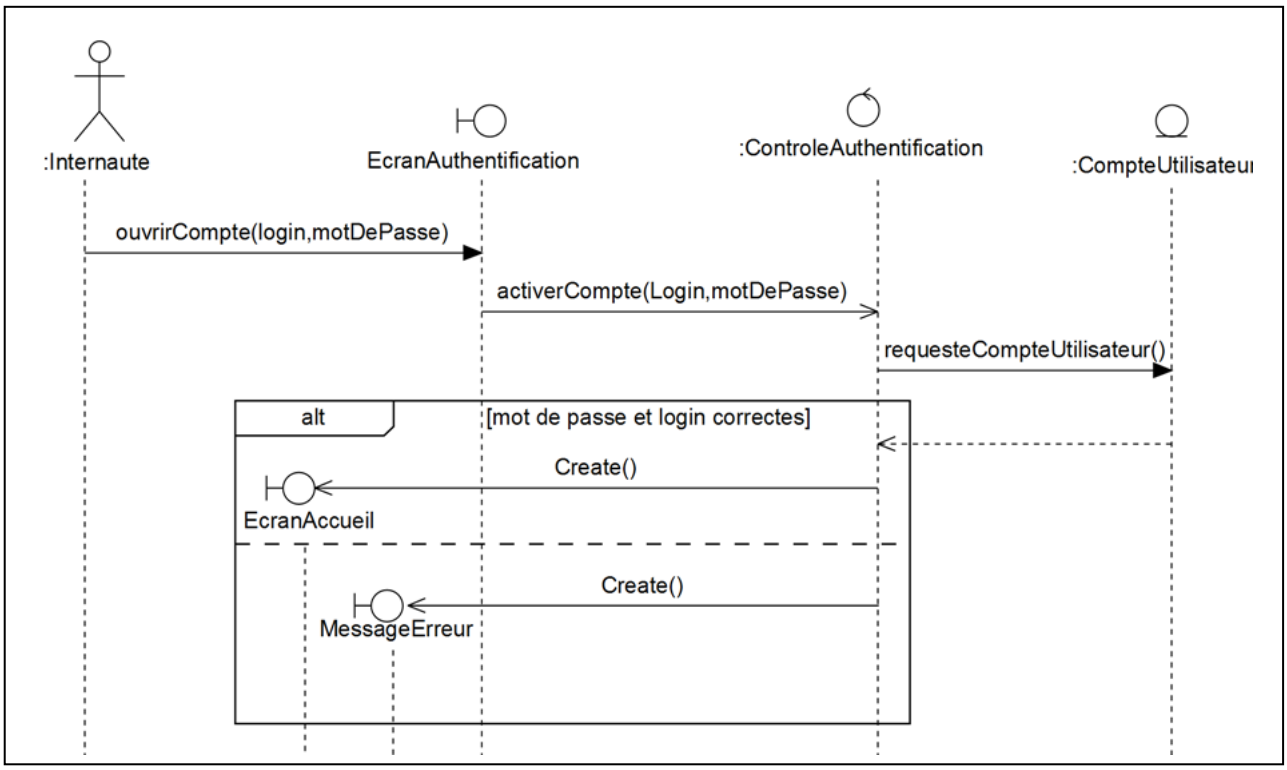

Figure 2.18. Exemple d'un diagramme de séquence

2.4. Diagramme de classe :

2.4.1. Introduction :

Le diagramme de classes est considéré comme le plus important et le plus utilisé de la modélisation orientée objet.

Le diagramme de classes présente la vue statique du système et ne tient pas compte des aspects dynamiques. [4]

2.4.2 Définition :

Le diagramme de classe représente l'architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens, que ceux-ci représentent un emboîtage conceptuel (héritage, marqué par une flèche terminée par un triangle) ou une relation organique (agrégation, marquée par une flèche terminée par un diamant). [5]

2.4.3. Classe :

1. Définition :

Description abstraite d'un ensemble d'objets qui partagent les mêmes propriétés (attributs et associations) et comportements (opérations et états).[6]

2. Représentation :

Une classe est représentée par un classeur divisé en trois compartiments : le premier indique le nom de la classe, le deuxième ses attributs et le troisième ses opérations. [4]

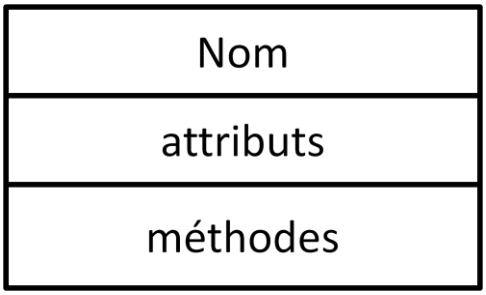

Figure 2.19. Représentation d'une classe

3. Nom d'une classe :

Le nom de la classe doit évoquer le concept décrit par la classe. Il commence par une majuscule.

[<Nom_du_paquetage_1>::...::<Nom_du_paquetage_N>]
<Nom_de_la_classe> [{ [abstract], [<auteur>], [<date>], ... }]

4. Les attributs :

La syntaxe d'un attribut est la suivante :

<visibilité> <nomAttribut> : <type> [= <valeur_par_défaut>]

[ ] **:** les crochets indiquent que ce qui est à l'intérieur est optionnel.

Visibilités :

Public ou + : tout élément qui peut voir le conteneur peut également voir l'élément indiqué.

Protected ou # : seul un élément situé dans le conteneur ou un de ses descendants peut voir l'élément indiqué.

Private ou - : seul un élément situé dans le conteneur peut voir l'élément.

Package ou ~ ou rien : seul un élément déclaré dans le même paquetage peut voir l'élément.

5. Les opérations :

La syntaxe d'une opération est la suivante :

<visibilité> <nom_opération> ([<*paramètre_1*>, ... , <*paramètre_N*>]) : [<type_renvoyé>]

2.4.4. Relations entre classes :

1. Association :

Une association est une relation entre deux classes qui indique qu'il peut y avoir des liens entre des instances des classes associées.

Une association est représentée par une ligne continue entre deux classes. [4]

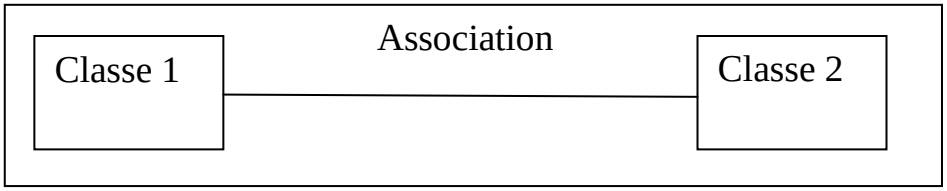

Figure 2.20. Représentation d'une relation d'association

2. Multiplicité *(cardinalité)* :

Aux deux extrémités d'une association, on doit faire figurer une indication de multiplicité.

La multiplicité spécifie le nombre d'objets qui peuvent participer à une relation avec un objet de l'autre classe dans le cadre d'une association.

Un et un seul	1 ou 1..1
Zéro ou un	0..1
plusieurs	* ou 0..*
De 1 à plusieurs	1..*
De M à N (entiers naturels)	M..N

Tableau 2.1. Les cardinalité

Figure 2.21. Représentation d'une relation de multiplicité

3. Les classes-association :

Les attributs d'une classe dépendent de l'identifiant de la classe. Parfois, un attribut dépend de 2 identifiants, appartenant à 2 classes différentes. On va donc placer l'attribut dans l'association entre les deux classes. Dans ce cas, l'association est dite « porteuse d'attributs ». Une association porteuse d'attributs est appelée classe-association. [4]

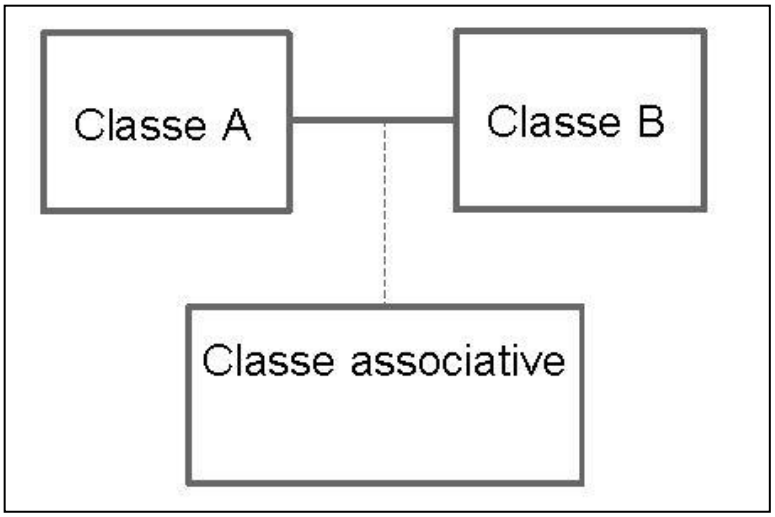

Figure 2.22. Classe d'association

4. Terminaison d'association :

La possession d'une terminaison d'association par la classe située à l'autre extrémité de l'association peut être spécifié graphiquement par l'adjonction d'un petit cercle plein (point).

La présence d'un point implique que la terminaison d'association appartient à la classe située à l'autre extrémité.

L'absence du point implique que la terminaison d'association appartient à l'association.

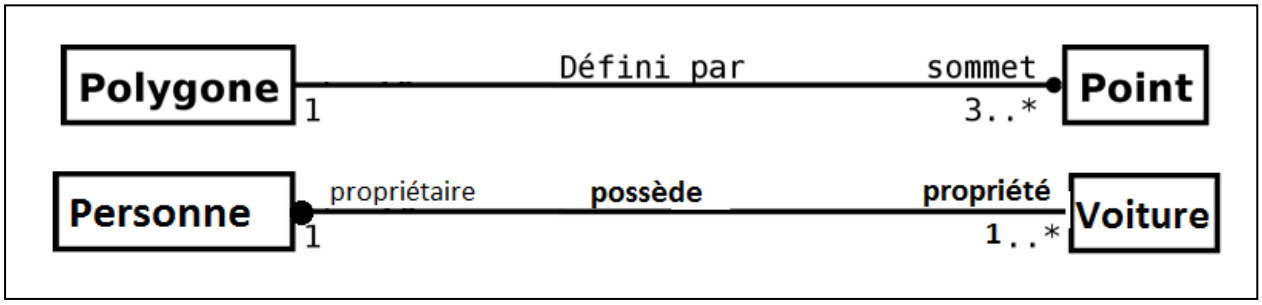

Figure 2.23. Terminaison d'association

5. Agrégation et composition

Une agrégation est une forme particulière d'association ou un tout (appelé classe agrégat) est relié à ses parties (appelé classe agrégées).

Graphiquement, on ajoute un losange vide du côté de l'agrégat.

Les agrégations n'ont pas besoin d'être nommées, elles signifient « contient » ou « est composé de ».

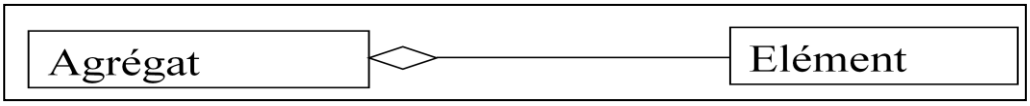

Figure 2.24. Agrégation et composition

6. La composition :

La composition est un cas particulier de l'agrégation implique que:

Un élément ne peut appartenir qu'à un seul agrégat composite (agrégation non partagée)

La destruction de l'agrégat composite entraîne la destruction de tous ses éléments (le composite est responsable du cycle de vie des parties). [4]

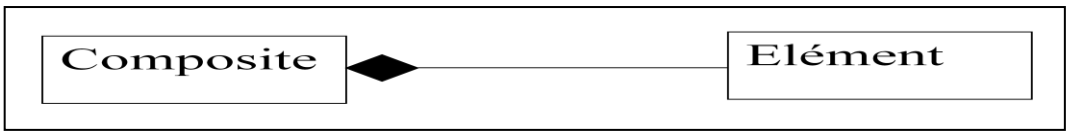

Figure 2.25. Représentation d'une composition

2.4.5. L'héritage :

L'héritage décrit une relation entre une classe générale (classe de base ou classe parent) et une classe spécialisée (sous-classe).

La classe spécialisée hérite les attributs et les méthodes de la classes générale et peuvent comporter des attributs et des méthodes spécifiques supplémentaires.

La relation d'héritage est représentée par une flèche avec un trait plein dont la pointe est un triangle fermé désignant le cas le plus général.

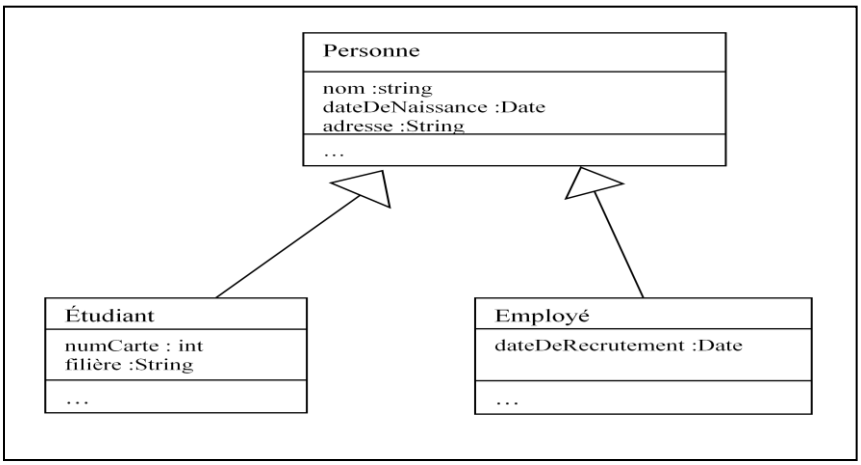

Figure 2.26. Représentation d'un héritage

2.4.6. Les interfaces :

Une interface est un ensemble de constantes et de déclarations de méthodes, une classe qui implémente une interface doit implémenter toutes ses méthodes l'interface est représenté par un classeur stéréotypé « interface », elle doit être réalisée par au moins une classe. [4]

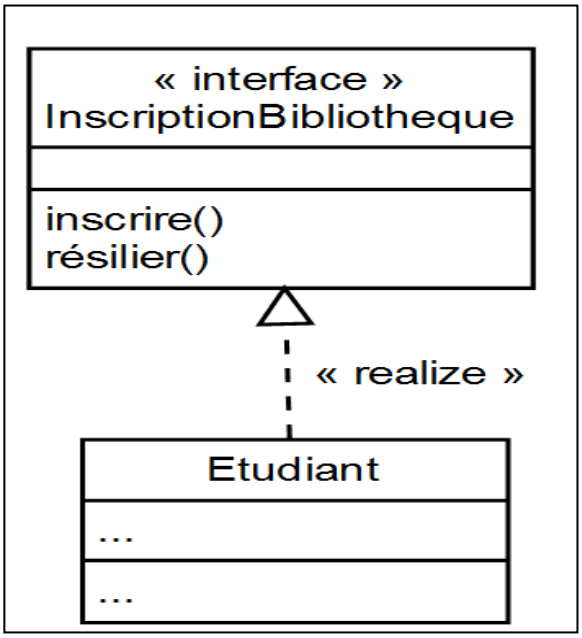

Figure 2.27. Représentation d'une interface.

2.4.7. Package :

Package (ou paquetage): mécanisme général de regroupement d'éléments tels que classes, interfaces, cas d'utilisation, etc.

Les packages peuvent être imbriqués dans d'autres packages. [4]

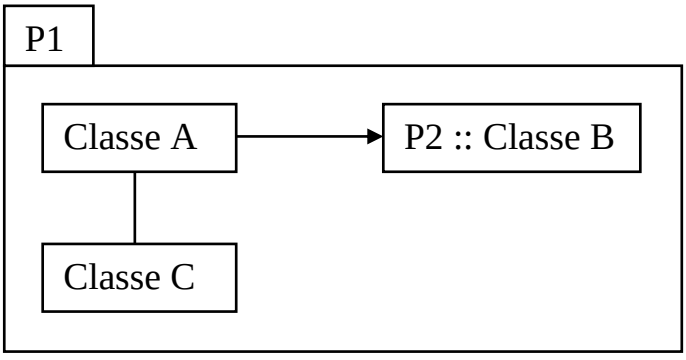

Figure 2.28. Représentation d'un package

2.4.8. Élaboration d'un diagramme de classes :

1. Trouver les classes du domaine étudié :

Cette étape empirique se fait généralement en collaboration avec un expert du domaine. Les classes correspondent généralement à des concepts du domaine.

2. Trouver les associations entre classes :

Les associations correspondent souvent à des verbes mettant en relation plusieurs classes, comme « est composé de », « pilote », « travaille pour ».

3. Trouver les attributs des classes :

Les attributs correspondent souvent à des noms, ou des groupes nominaux.

Exemple 1 :

- « la masse d'une voiture » : « masse » est attribut de la classe « Voiture »

Les adjectifs et les valeurs correspondent souvent à des valeurs d'attributs

Exemple 2 :

- « une voiture rouge »➔ « Couleur » est attribut de la classe « Voiture »

4. Organiser et simplifier le modèle :

En éliminant les classes redondantes et en utilisant l'héritage.

5. Itérer et raffiner le modèle :

Un modèle est rarement correct dès sa première construction. La modélisation objet est un processus non pas linéaire mais itératif. [4]

2.4.9. Passage du diagramme de classe au modèle relationnel :

1. Transformation d'une classe avec attributs

Chaque classe devient une relation.

Les attributs de la classe deviennent des attributs de la relation.

Si la classe possède un identifiant, il devient la clé primaire de la relation, sinon, il faut ajouter une clé primaire arbitraire. [4]

Exemple :

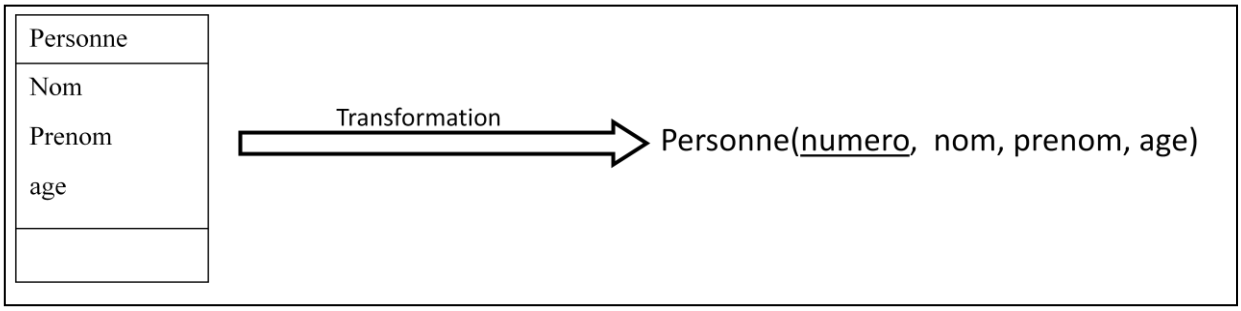

Figure 2.29. Exemple de Transformation d'une classe avec attributs

2. Association 1 vers 1 :

Pour représenter une association 1 vers 1 entre deux relations, la clé primaire de l'une des relations doit figurer comme clé étrangère dans l'autre relation.

Exemple :

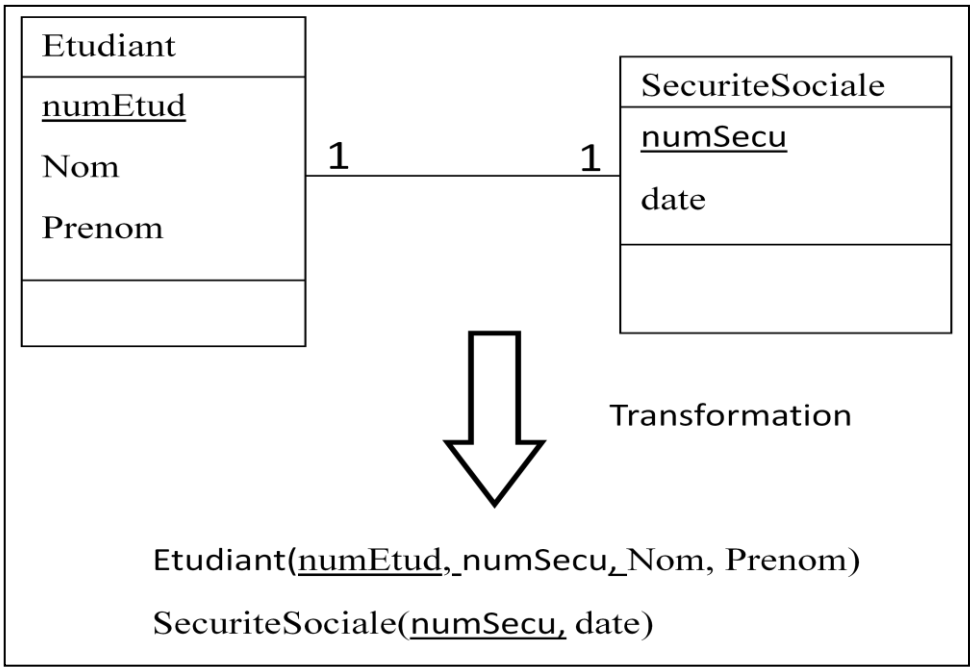

Figure 2.30. Exemple d'association 1 vers 1

3. Association 1 vers plusieurs :

Pour représenter une association 1 vers plusieurs, on procède comme pour une association 1 vers 1, excepté que c'est forcément la relation du côté plusieurs qui reçoit comme clé étrangère la clé primaire de la relation du côté 1.

Exemple :

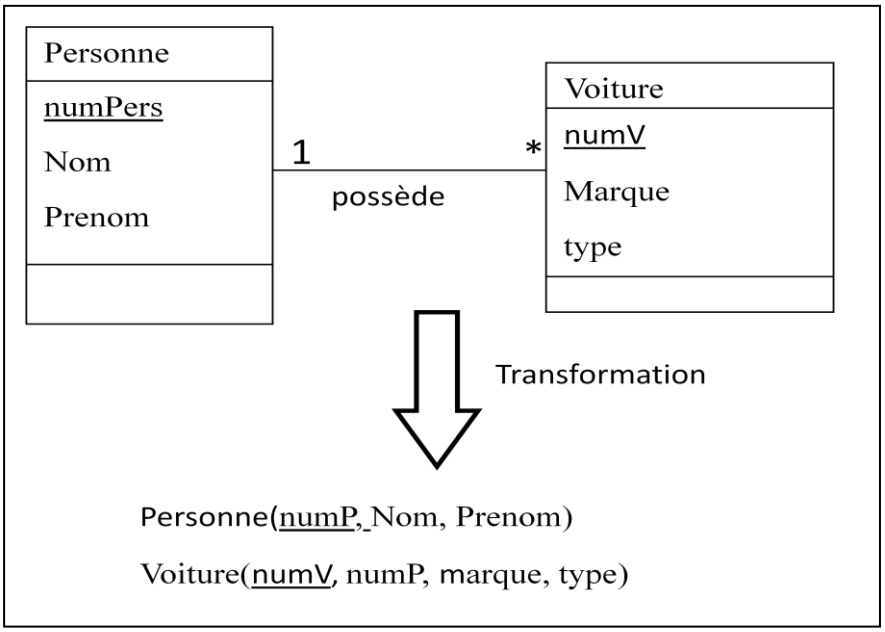

Figure 2.31. Exemple d'association 1 vers plusieurs

4. Association plusieurs vers plusieurs :

Pour représenter une association du type plusieurs vers plusieurs, il faut introduire une nouvelle relation dont les attributs sont les clés primaires des relations en association, et dont la clé primaire est la concaténation de ces deux attributs.

Si l'association possède des attributs, ils deviennent des attributs de la relation correspondante.

Exemple :

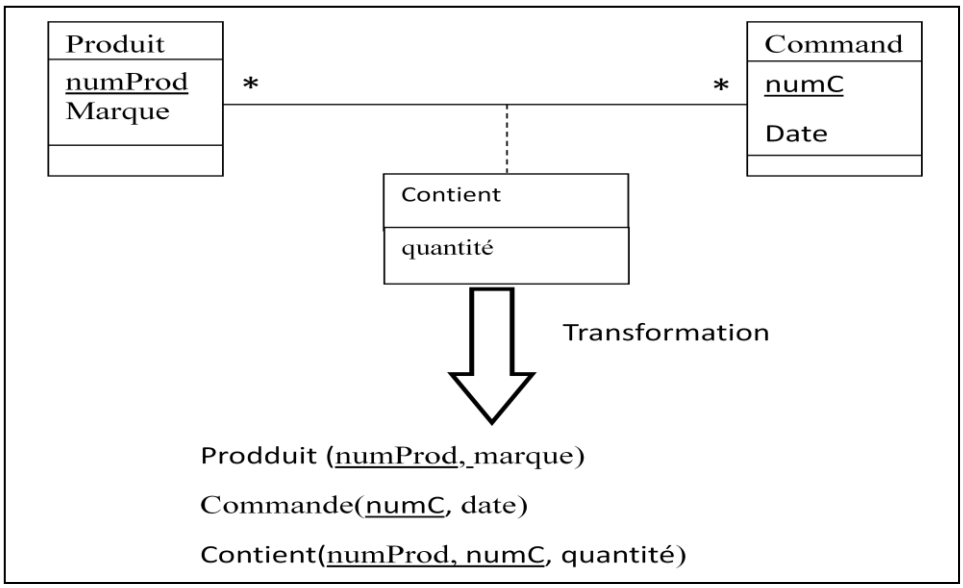

Figure 2.32. Exemple d'association plusieurs vers plusieurs

2.5. Digramme d'activité :

2.5.1. Définition :

Le diagramme d'activité est un diagramme états-transitions simplifié pour lequel les états se réduisent à de simples actions ou activités et dont les transitions se déclenchent automatiquement avec éventuellement des gardes.

Les diagrammes d'activité sont utilisés pour documenter le déroulement des opérations dans un système, du niveau commercial au niveau opérationnel (de haut en bas)

2.5.2. Nœud initial :

- Représente le point de départ du diagramme d'activités.
- Graphiquement, un état initial est représenté par un cercle plein.
- Un état initial possède un arc sortant et pas d'arc entrant. [4]

Figure 2.33. Nœud initial

2.5.3. Nœud final:

- Représente la fin du diagramme d'activités.
- Graphiquement, un état final est représenté par un cercle vide contenant un petit cercle plein.
- Un état final possède un ou plusieurs arcs entrants et aucun arc sortant. [4]

Figure 2.34. Nœud final

2.5.4. Action :

Correspond a un traitement qui modifié l'état de système. Le passage d'une action à une autre est matérialisé par une transition. Les transitions sont déclenchées par la fin d'une action et provoquent le début d'une autre

Figure 2.35. Action

2.5.5. Transition :

Une transition est le passage d'une action vers l'action suivante. Et représentées par des flèches en traits pleins qui connectent les activités entre elles.

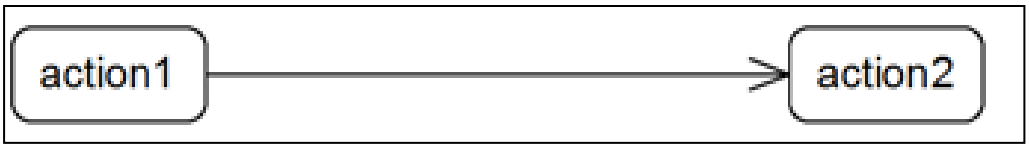

Figure 2.36. Transition

1. **Transition automatique :**

Déclenchées par la fin d'une activité et provoquent le début immédiate d'une autre

2. **Transition gardée :**

Le passage à l'activité suivante n'est possible que si la condition de la transition est vérifiée.

2.5.6. Nœud de décisions :

- Un nœud de décision permet de faire un choix entre plusieurs flots.
- Il possède un arc entrant et plusieurs arcs sortants.
- Les arcs sortants sont généralement accompagnés de conditions de garde pour conditionner le choix.
- Graphiquement, on représente un nœud de décision par un losange.

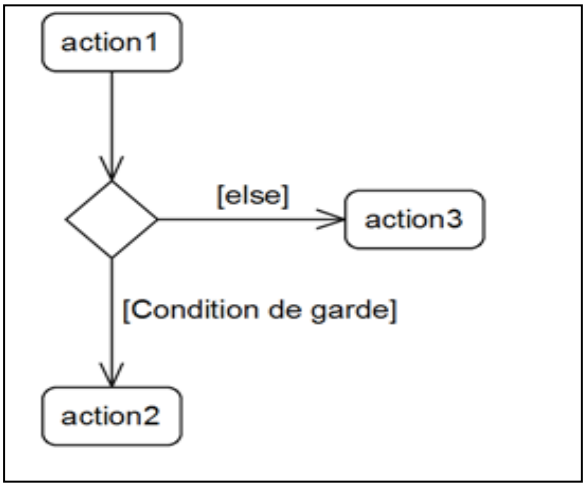

Figure 2.37. Nœud de décisions

2.5.7. Embranchement ou bifurcation (fork) :

- Un embranchement est la décomposition du flux de contrôle en deux ou plusieurs flux de contrôle.
- Un embranchement permet de spécifier des activités concurrentes.

2.5.8. Jonction (join) :

- Une jonction est la recomposition du flux de contrôle de deux ou plusieurs flux de contrôle en un seul.
- Un embranchement permet de spécifier la synchronisation d'activités concurrentes.
- Les flux qui sortent d'un embranchement doivent balancer les flux qui entrent dans la jonction correspondante. [7]

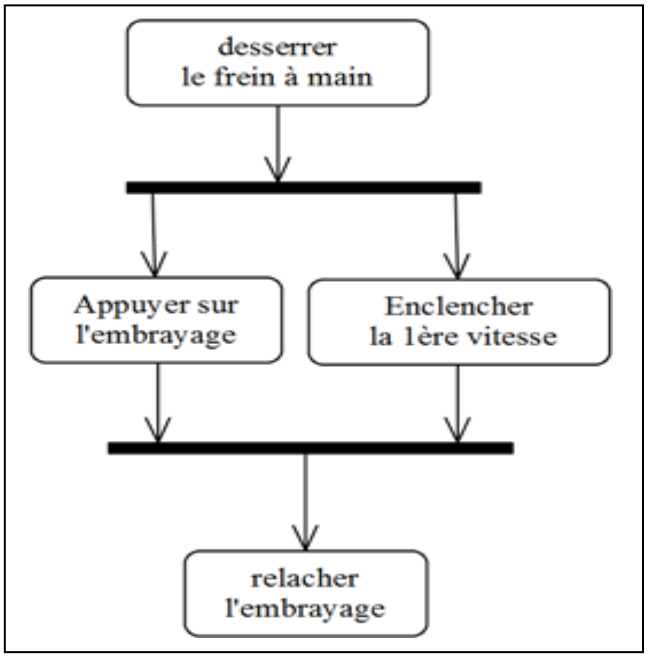

Figure 2.38. Exemple de fork et join

2.5.9. Couloirs d'activités (swimlane) :

Afin d'organiser un diagramme d'activités selon les différents responsables des actions représentées, il est possible de définir des couloirs d'activités.

Swimlanes sont un genre de paquet pour la responsabilité d'organisation des activités dans une classe. Elles correspondent souvent aux unités d'organisation dans un modèle d'affaires.

Un diagramme d'activité peut être divisé visuellement en chacun de swimlanes séparé des swimlanes voisins par les lignes solides verticales des deux côtés. Chaque swimlane représente la responsabilité d'une partie de l'activité globale, et peut par la suite être mis en application par un ou plusieurs objets. La commande relative des swimlanes n'a aucune signification sémantique mais pourrait indiquer une certaine affinité. Chaque action est assignée à un swimlane. Les transitions peuvent croiser des ruelles, il n'y a aucune signification au cheminement d'un chemin de transition. [7]

2.7. Conclusion :

Dans ce chapitre, nous avons fait un aperçu sur le langage de modélisation utilisé pour l'étude théorique de notre logiciel, et grâce aux diagrammes que nous avons parlés nous pourrons comprir la modélisation général de notre projet.

Chapitre 03 : Les langages de programmation et les outils de travail

3.1. Introduction :

Dans ce chapitre nous allons présenter les langages de programmation, qu'ils sont le lien entre le programmeur et la machine, avec ces types, niveaux, avantages et inconvénient, et avec lesquels on peut réaliser notre application et on va examiner les outils de travail qui peut nous aider.

Section 01 : Les langages de programmations :

3.2. Les programmes:

Les programmes sont à la base de l'informatique. Ce sont eux qui nous permettent d'exécuter des actions sur notre ordinateur.

3.3. Les langages de programmation :

L'ordinateur est une machine étonnante et complexe. À la base, il ne comprend qu'un langage très simple constitué de 0 et de 1. Ainsi, un message tel que celui-ci :
10100100101000110101010010101110101000110100 10
peut signifier quelque chose comme « Affiche une fenêtre à l'écran ».
Mais c'est très compliqué ! On va être obligé d'apprendre ce langage pour construire une application sur cet ordinateur ?
Heureusement non.

A cause de ca l'être humain a inventé plusieurs langages de programmation pour facilité la réalisation des applications.

On va présenter quelques Paradigmes :

3.4. Les types des langages :

3.4.1. Langages haut niveau :

En programmation informatique, un langage de haut niveau c'est le langage qui nous permet d'écrire des programmes en utilisant des mots des langages naturels (très souvent l'anglais) et des symboles mathématique familière, ce qui facilite l'écriture des programme. Ils sont généralement indépendants de la machine : le même programme pourra être utilisé tel que sur plusieurs types

d'ordinateurs, quoique les programmes puissent également être conçus pour un système d'exploitation en particulier.

En 2010, il existe plus de 200 langages de programmation de haut niveau. Les plus populaires sont Pascal, Java, Fortran, COBOL, OCaml et Python.

3.4.2. Langages bas niveau :

Un langage de programmation est dit de bas niveau lorsque le codage de celui-ci se rapproche du langage machine, et donc permet de programmer à un degré très bas. Les langages de bas niveau sont à opposer aux langages de haut niveau, qui permettent de créer un programme sans tenir compte de la façon dont fonctionne le matériel de l'ordinateur censé exécuter le programme.

Et parmi les langages de bas niveau on a : C, C++… [8]

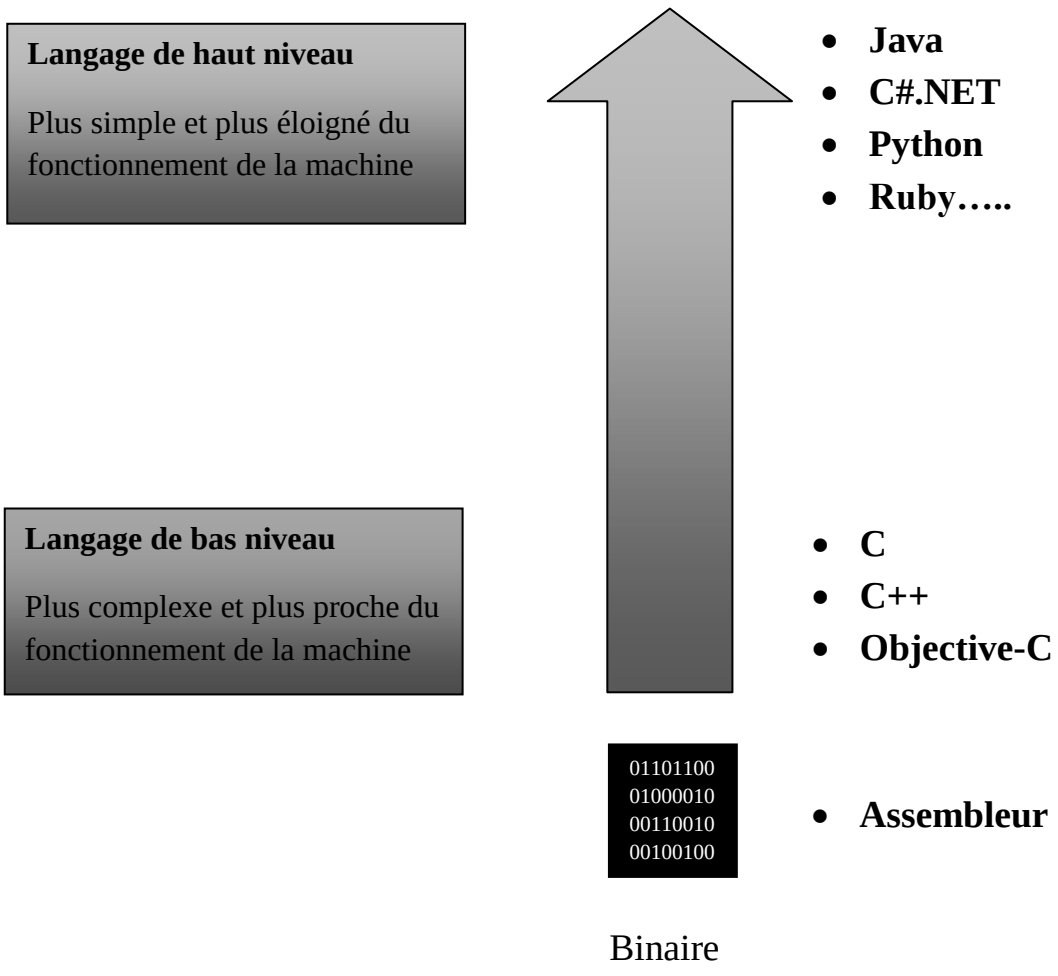

Figure 3.1. Les types des langages

3.5. Langage c :

Figure 3.2.logo du langage c

3.5.1. Présentation :

Créé par les concepteurs d'Unix dans le but, justement, de reprogrammer ce système d'exploitation (initialement écrit en Assembleur) dans un langage plus portable, c'est aujourd'hui l'un des langages les plus courants, et sûrement le plus utilisé dans le monde Unix. C'est l'un des langages les plus proches du processeur, mais qui reste très facilement transposable sur d'autres plates-formes pour peu que l'on s'en tienne à la syntaxe standard. Un code source C peut ainsi être compilé sur n'importe quelle machine, moyennant relativement peu d'efforts d'adaptation. Le fait qu'il soit composé d'un noyau très simple entouré d'un vaste choix de bibliothèques spécialisées lui permet également de fonctionner sur les systèmes les plus légers.[9]

3.5.2. Les avantages du langage c :

- Le langage C est un langage très portable sur n'importe quel système d'exploitation disposant d'un compilateur C : Windows, UNIX, VMS (système des VAX)
- C permet de développer des programmes concis et rapides.
- Il est inexact que le C est un langage difficile à apprendre ! Au contraire : le C dispose de peu d'instructions, les structures de données sont limitées
- C’est un langage universel et pas orienté vers un domaine d'applications spéciales.

3.5.3. Les inconvénients du langage c :

- Son approche de la modularité restée au niveau de ce qui se faisait au début des années 1970
- La gestion d'exceptions très sommaire
- Certaines erreurs ne peuvent être détectées automatiquement qu'à l'aide d'outils supplémentaires et non standardisés, comme lint et Valgrind.

3.6. Langage c++ :

Figure 3.3. Logo du langage c++

3.6.1. Présentation :

Le C++ est en quelque sorte un C amélioré. Ce langage qui permet pas mal de choses est considéré comme l'artillerie lourde de la programmation. Il combine les performances du C avec le paradigme de programmation objet, sans l'imposer pour autant. Il comporte pas mal de fonctionnalités dont certaines qui ne sont là que pour être vraiment complet (exemple : l'héritage privé). En contre-partie, il souffre d'une certaine complexité.[10]

3.6.2. Les avantages du langage c++ :

- Il est portable : un même code source peut théoriquement être transformé sans problème en exécutable sous Windows.
- l'exécution est un peu plus rapide
- Le gros avantage du C++ est la possibilité de pouvoir écrire des programmes très sécurisés.

3.6.3. Les inconvénients du langage c++ :

- L'utilisation de ces variables rend plus difficile la compréhension d'un programme.
- les variables globales empêchent la réentrance si elles sont mal utilisées.

3.7. Le langage python :

Figure 3.4. Logo du langage python

3.7.1. Présentation :

Python est un langage de programmation multiplateforme (portable), dynamique, gratuit, qui permet une approche modulaire et orientée objet de la programmation. Python est développé depuis 1989 par Guido van Rossum et de nombreux contributeurs bénévoles.

3.7.2. L'utilisation :

- Du calcul scientifique.
- Traitement de son.
- Traitement d'image.
- Des applications avec interface graphique.
- Des jeux vidéo 2D.
- Des applications web.
- Des applications réseaux. [11]

3.7.3. Les avantages du langage python :

- La simplicité en raison de son utilisation de la sémantique et la syntaxe régulière.
- La portabilité il est facile à porter sur d'autres systèmes d' exploitation tels que Microsoft Windows, Mac OS X d'Apple et toutes les distributions Linux /Unix.
- Python est un des langages de script les plus courantes et les plus populaires parce qu'il est open sources.
- L'exécution est nécessairement très rapide.
- Il est multi-plateforme et gratuit.

3.7.4. Les inconvénients du langage python :

- Détection des erreurs à l'exécution (pas à la compilation).
- 2 versions majeures non compatibles : Python 2 et Python 3. Python 3 est mieux, mais tous les modules ne sont pas compatibles.

3.8. Le langage VB :

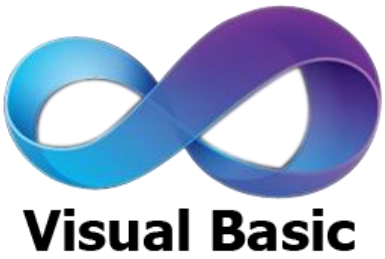

Figure 3.5. Logo du langage VB

3.8.1. Présentation :

Visual Basic est un langage de programmation haut niveau, typé orienté objets, et un langage de programmation événementielle de troisième génération ainsi qu'un environnement de développement intégré, créé par Microsoft au début des années 1990 . Visual Basic est directement dérivé du BASIC et permet le développement rapide d'applications, la création d'interfaces utilisateur graphiques, l'accès aux bases de données en utilisant les technologies DAO(Data Access Object), ADO(ActiveX Data Objet) et RDO, ainsi que la création de contrôles ou objets ActiveX .[12]

3.8.2. L'utilisation :

Visual Basic est un des langages les plus utilisés pour l'écriture d'applications commerciales. Il a également été très utilisé dans le monde de l'ingénierie et de la recherche appliquée en raison de sa capacité à permettre des développements très rapides et très efficients permettant ainsi aux scientifiques de se consacrer davantage à l'algorithmique et moins aux aspects formels du codage.[13]

3.8.3. Les avantages du langage VB :

- Développement rapide des applications.
- Facilité d'installation.
- Fonctionnalités infinies.
- Facilité de développement de contrôles personnalisés.

3.8.4. Les inconvénients du langage VB :

- Incompatibilité avec les programmes VB 6, et surtout, la taille du Framwork .NET.
- Pas très rapide.
- Dépend beaucoup des versions utilisées de Windows et MS Office.

3.9. Langage java :

Figure 3.6. Logo du langage java

3.9.1. Présentation :

C'est un langage très utilisé, notamment par un grand nombre de programmeurs professionnels, ce qui en fait un langage incontournable actuellement.

Java est un langage de programmation moderne développé par Sun Microsystems (aujourd'hui racheté par Oracle). Il ne faut surtout pas le confondre avec JavaScript (langage de scripts utilisé principalement sur les sites web), car Java n'a rien à voir. Une de ses plus grandes forces est son excellente portabilité : une fois votre programme créé, il fonctionnera automatiquement sous Windows, Mac, Linux, etc.

On peut faire de nombreuses sortes de programmes avec Java :

- Des applications, sous forme de fenêtre ou de console.
- Des applets, qui sont des programmes Java incorporés à des pages web.
- Des applications pour appareils mobiles, avec J2ME.
- Et bien d'autres ! J2EE, JMF, J3D pour la 3D… [13]

3.9.2. Quelques chiffres et faits à propos de Java en 2011 :

- 97% des machines d'entreprises ont une JVM installée.
- Java est téléchargé plus d'un milliards de fois chaque année.
- Il y a plus de 9 millions de développeurs Java dans le monde.
- Java est un des langages les plus utilisé dans le monde.
- Tous les lecteurs de Blue-Ray utilisent Java.
- Plus de 3 milliards d'appareils mobiles peuvent mettre en œuvre Java.

- Plus de 1,4 milliards de cartes à puce utilisant Java sont produites chaque année. [14]

3.9.3. Les caractéristiques :

Java possède un certain nombre de caractéristiques qui ont largement contribué à son énorme succès :

- Java est interprété : la source est compilé en pseudo code ou bytecode puis exécuté par un interpréteur Java (la Java Virtual Machine JVM). Ce concept est à la base du slogan de Sun pour Java : WORA (Write Once, Run Anywhere : écrire une fois, exécuter partout). En effet, le bytecode, s'il ne contient pas de code spécifique à une plate-forme particulière peut être exécuté et obtenir quasiment les mêmes résultats sur toutes les machines disposant d'une JVM.

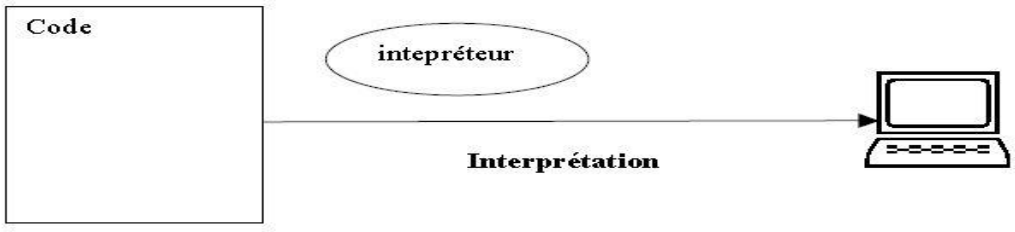

Figure 3.7. Opération de l'interprétation

- Java est portable : il est indépendant de toute plate-forme il n'y a pas de compilation spécifique pour chaque plate forme. Le code reste indépendant de la machine sur laquelle il s'exécute. Il est possible d'exécuter des programmes Java sur tous les environnements qui possèdent une Java Virtual Machine. Cette indépendance est assurée au niveau du code source grâce à Unicode et au niveau du bytecode.

- Java est orienté objet : comme la plupart des langages récents, Java est orienté objet. Chaque fichier source contient la définition d'une ou plusieurs classes qui sont utilisées les unes avec les autres pour former une application. Java n'est pas complètement objet car il définit des types primitifs (entier, caractère, flottant, booléen,...).

- Java est simple : le choix de ses auteurs a été d'abandonner des éléments mal compris ou mal exploités des autres langages tels que la notion de pointeurs (pour éviter les incidents en manipulant directement la mémoire), l'héritage multiple et la surcharge des opérateurs, …..

- Java est fortement typé : toutes les variables sont typées et il n'existe pas de conversion automatique qui risquerait une perte de données.
- Java assure la gestion de la mémoire : l'allocation de la mémoire pour un objet est automatique à sa création et Java récupère automatiquement la mémoire inutilisée grâce au garbage collector qui restitue les zones de mémoire laissées libres suite à la destruction des objets.
- Java est sûr : la sécurité fait partie intégrante du système d'exécution et du compilateur. Un programme Java planté ne menace pas le système d'exploitation. Il ne peut pas y avoir d'accès direct à la mémoire. L'accès au disque dur est réglementé dans une applet. [14]

3.9.4. Les différentes versions de java :

Au fur et à mesure des nouvelles versions de Java, le nombre de packages et de classes s'accroît

	Java 1.0	Java 1.1	Java 1.2	J2SE 1.3	J2SE 1.4	J2SE 5.0	Java SE 6	Java SE 7	Java SE 8
Nombre de packages	8	23	59	76	135	166	202	209	217
Nombre de classes	201	503	1520	1840	2990	3280	3780	4024	4240

Tableau 3.1. Différentes versions de Java. [14]

3.9.5. Les différences entre Java et JavaScript :

- Il ne faut pas confondre Java et JavaScript. JavaScript est un langage développé par Netscape Communications.[14]

	Java	JavaScript
Auteur	Développé par Sun Microsystems	Développé par Netscape Communications
Format	Compilé sous forme de bytecode	Interprété
Stockage	Applet téléchargé comme un élément de la page web	Source inséré dans la page web
Utilisation	Utilisable pour développer tous les types d'applications	Utilisable uniquement pour "dynamiser" les pages web
Exécution	Exécuté dans la JVM du navigateur	Exécuté par le navigateur
POO	Orienté objets	Manipule des objets mais ne permet pas d'en définir
Typage	Fortement typé	Pas de contrôle de type
Complexité du code	Code relativement complexe	Code simple

Tableau 3.2. Les différences entre Java et JavaScript. [14]

3.9.6. Les avantages du langage java :

- Facile à apprendre et à programmer (Un temps de développement réduit d'un tiers par rapport à C++ pour des développeurs expérimentés).
- Orienté objet, Interprété et portable, fortement et statiquement typé, à éditions de liens dynamique, Simplifié.
- Une documentation abondante sur le WEB.
- A ramasse-miette : la mémoire est gérée automatiquement par le langage.
- sécurisé : de nombreux garde-fous (encapsulation, exceptions gestionnaire de sécurité,. . .) contribuent à la robustesse des programmes tant à la compilation qu'à l'exécution.

- multithread : des processus légers peuvent s'exécuter en parallèle, ce qui est particulièrement utile dans le cas de la programmation de serveurs ou de la parallélisassions multiprocesseurs.
- surchargeable: les fonctions peuvent avoir le même nom et des arguments de type différent, ce qui permet de rapprocher la conception abstraite de l'implémentation. [14]

3.9.7. Les inconvénient du langage java :

- L'exécution est nécessairement moins rapide que si le code est compilé nativement.
- Difficultés d'accès aux ressources matérielles. [14]

Section 02 : Les outils de travail

3.10. Base de données:

BD un ensemble structuré de données enregistrées sur des supports accessibles par l'ordinateur, représentant des informations du monde réel et pouvant être interrogées et mises à jour par une communauté d'utilisateurs.

3.11. SQL: Structured Query Language

SQL est un langage complet de gestion de bases de données relationnelles. Il a été conçu par IBM dans les années 70. Il est devenu le langage standard des systèmes de gestion de bases de données (SGBD) relationnelles.

3.12. WampServer:

Figure 3.8. logo du programme wampserver

Est une plate-forme de développement Web sous Windows. Il vous permet de développer des applications Web dynamiques à l'aide du serveur Apache2,

du langage de scripts PHP et d'une base de données MySQL. Il possède également PHPMyAdmin pour gérer plus facilement vos bases de données.

3.12.1. PHPMyAdmin:

Est une interface conviviale faîte en PHP pour gérer une base de données MySQL. Utiliser pour créer des bases de données, les supprimer, créées des tables est exécuter toutes les requêtes permises par MySQL.

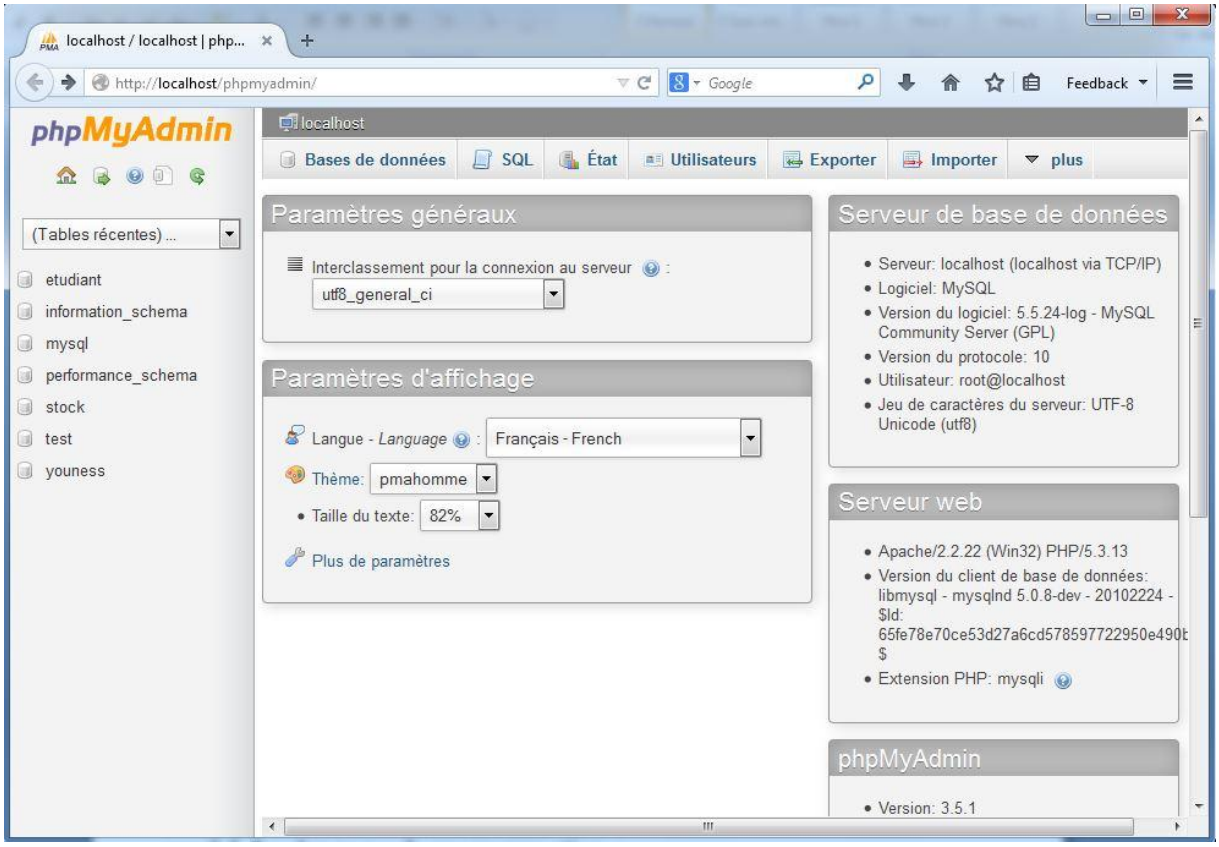

Figure 3.9. Interface du phpMyAdmin

3.12.2. Les avantages:

- WAMP est parmi les meilleurs pour développer les petits sites
- facile à installer, presque rien n'a configuré.

3.13. Adobe Photoshop :

Figure 3.10. Logo du programme Photoshop

Photoshop est un logiciel de retouche, de traitement et de dessin assisté par ordinateur. Édité par Adobe Systems, il fait partie d'une longue suite de logicielle touchant dans le domaine du multimédia qui est connue sous le nom d'Adobe Creative Suite (Photoshop, Illustrator, InDesign, Dreamweaver, Flash, entre-autres).

Il est surtout utilisé pour le traitement de photos numériques (ce que l'on appelle plus communément la retouche photographique), mais sert également à la création d'images.

Il travaille essentiellement sur images matricielles car les images sont constituées d'une grille de points appelés pixels.

Reconnu principalement par les infographistes professionnels à travers sa puissante galerie de filtres et d'outils graphiques performants, il est maintenant enseigné dans les plus grandes écoles et est utilisé par une grande majorité des studios et agences de créations.

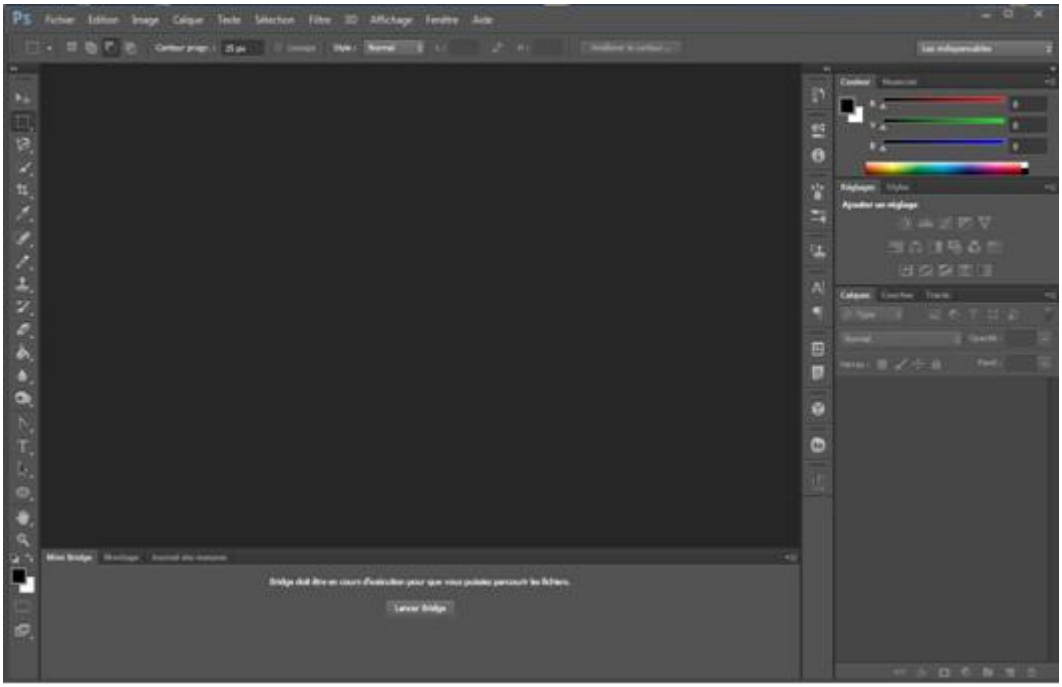

Figure 3.11. Interface du Photoshop

3.13.1. Que peut-on faire avec Photoshop ?

- **Peinture digitale**

La peinture digitale est l'art du dessin sur ordinateur et cela se fait dans presque tous les cas à l'aide d'une tablette graphique.

- **Montages photo**

Montages photos est assemblage des éléments venant de plusieurs photos différentes en une seule création.

- **La retouche photo**

La retouche photo est un art qui peut aller d'un simple réchauffement des couleurs à une transformation complète, en passant par la suppression d'éléments indésirables. Tous les photographes professionnels, sans exception, retouchent leurs photos, ne serait-ce que pour faire un renforcement de contraste ou de couleurs.

- **Des affiches, flyers, covers de CD/DVD, etc...**

Le Photoshop est utilisé aussi pour créer des affiches publicitaire ou des couver CD d'un album d'un groupe de musique ou des flyers…

3.14. Conclusion :

Après notre étude et recherche dans le domaine du langage de programmation on a trouvé plusieurs langage (c, c++, java….) chaque un a des avantages et des inconvénients et on a découvris que le java est le plus adapter pour notre projet et il est parmi les langages les plus utiliser dans le monde pour la réalisation des applications desktop et même les applications web et il a beaucoup d'avantages qu'on a déjà expliqué.

Chapitre4 : Analyse et conception du projet

4.1. Introduction :

Dans le cadre de ce chapitre, nous allons identifier les acteurs du système, pour pouvoir établir précisément les frontières fonctionnelles du système, puis nous identifions les cas d'utilisation du système, et les scénarios et quelque diagramme qui sont le diagramme d'activité, de séquence et le diagramme de classe.

4.2. Identification des acteurs :

Le chef département : la personne responsable du système de l'emploi du temps qui va utiliser le logiciel.

4.3. Le diagramme de cas d'utilisation :

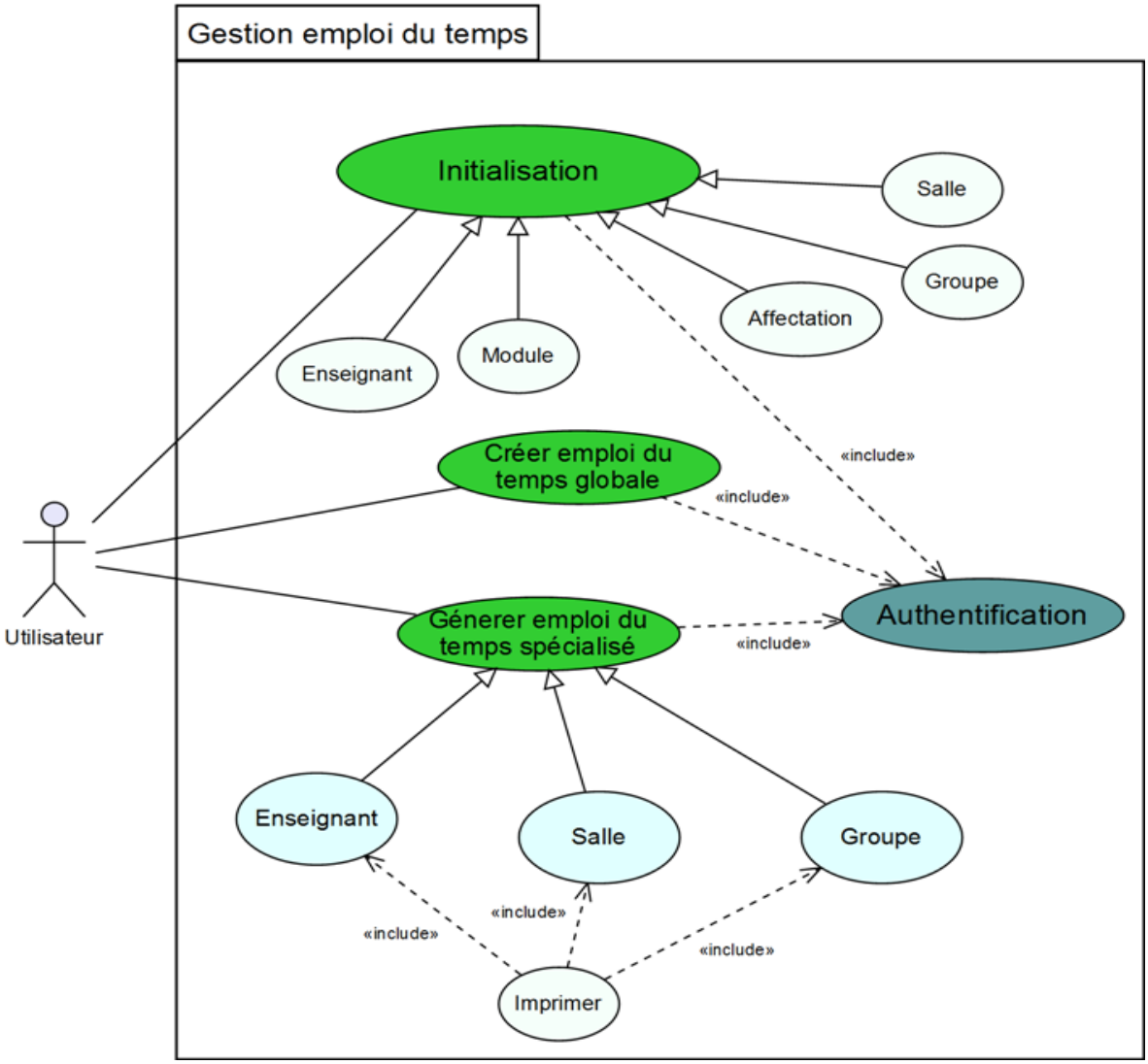

Figure 4.1. Le digramme de cas d'utilisation

4.4. Description textuelle du logiciel :

4.4.1. Description textuelle authentification :

Cas d'utilisation	Authentification
Objectif	Permettre à l'utilisateur de gérer des emplois du temps
Acteurs	Chef département
Responsable	Réalisateurs
Dates	03/02/2015
Version	1.0
Pré-conditions	Demande d'authentification
Post-conditions	L'utilisateur est authentifié par le système.
Scénario nominal	1. L'utilisateur demande entrer au système. 2. Le système demande le login le mot de passe. 3. L'utilisateur saisi le mot de passe. 4. Le système vérifie la validité du login et mot de passe et ouvre le système.
Scénario alternatif	**Le point 4 le mot de passe est erroné** 1. L e système affiche un message d'erreur. 2. Le système propose à l'utilisateur de renseigner une nouvelle fois le login et le mot de passe. 3. Reprise de l'enchainement du scénario nominal au point 2.
Scénario d'erreurs	**Le point 4 Login ou mot de passe est erroné pour la 3éme fois** Il sera déconnecté le système.

Tableau 4.1. La fiche descriptive d'authentification

4.4.2. Description textuelle d'initialisations :

Cas d'utilisation	Initialisation
Objectif	Permettre à l'utilisateur d'initialisé les ressources
Acteur	Chef département
Pré-conditions	L'utilisateur est entré au système
Post-conditions	Les ressources sont initialisées
Scénario nominal	1. l'utilisateur entré à la fenêtre d'initialisation. 2. initialisé la table des séances de cours disponible. 3. Initialisé domaines. 4. Initialisé enseignants. 5. Initialisé modules. 6. Sauvegardé domaine.
Scénarios alternatifs	Les points 4 ou 5 l'enseignant ou module est répété 1. Le système affiche un message de répétition. 2. Reprise de l'enchaînement du scénario nominal aux

	points 4 ou 5.
Scénarios d'erreur	/

Tableau 4.2. La fiche descriptive d'initialisations

4.4.3. Description textuelle d'affectation :

Cas d'utilisation	Affectation
Objectif	Permettre à l'utilisateur d'affecter les modules aux enseignants correspond
Acteur	Chef département
Pré-conditions	L'utilisateur est entré au système
Post-conditions	Les modules sont affectés
Scénario nominal	1. L'utilisateur entré à la fenêtre d'affectation. 2. Affecté les modules aux enseignants correspond
Scénarios alternatifs	Le point 2 le module déjà affecté à l'enseignant 1. Le système affiche un message d'erreur. 2. Reprise de l'enchaînement du scénario nominal au point 2.
Scénarios d'erreur	/

Tableau 4.3. La fiche descriptive d'affectation

4.4.4. Description textuelle d'emploi du temps global:

Cas d'utilisation	Remplir l'emploi du temps global
Objectif	Permettre à l'utilisateur d'remplir les séances devant les salles
Acteur	Chef département
Pré-conditions	L'utilisateur est entré à la fenêtre d'emploi du temps global
Post-conditions	Les séances sont remplies
Scénario nominal	1. L'utilisateur a rempli les champs d'une séance 2. La séance est remplie 3. L'utilisateur a supprimé une séance 4. La séance est supprimée 5. L'utilisateur a vidé l'emploi du temps global 6. L'emploi du temps est vidé
Scénarios alternatifs	Le point 1 l'enseignant ou le groupe est occupé 1. Le système affiche un message d'attention 2. Reprise le point 1 Le point 1 l'enseignant a dépassé le nombre des jours ou des heures maximales 1. Le système affiche un message d'attention 2. Reprise le point 1

Scénarios d'erreur	/

Tableau 4.4. La fiche descriptive d'emploi du temps global

4.4.5. Description textuelle d'emploi du temps salle, enseignant et filière:

Cas d'utilisation	Afficher les emplois du temps
Objectif	Permettre à l'utilisateur de voir les emplois du temps salle, enseignant et filière
Acteur	Chef département
Pré-conditions	L'utilisateur est entré à la fenêtre d'emploi du temps
Post-conditions	L'emploi du temps est imprimé
Scénario nominal	1. L'utilisateur entré a la fenêtre d'emploi du temps 2. L'utilisateur a choisi l'enseignant, la salle ou la filière 3. Le système affiche le choix de l'utilisateur 4. L'utilisateur imprime le tableau se choisi 5. Le système affiche la fenêtre d'impression
Scénarios alternatifs	/
Scénarios d'erreur	/

Tableau 4.5. La fiche descriptive d'emploi du temps salle, enseignant et filière

4.5. Diagramme de séquence du logiciel:

4.5.1. Diagramme de séquence système authentification :

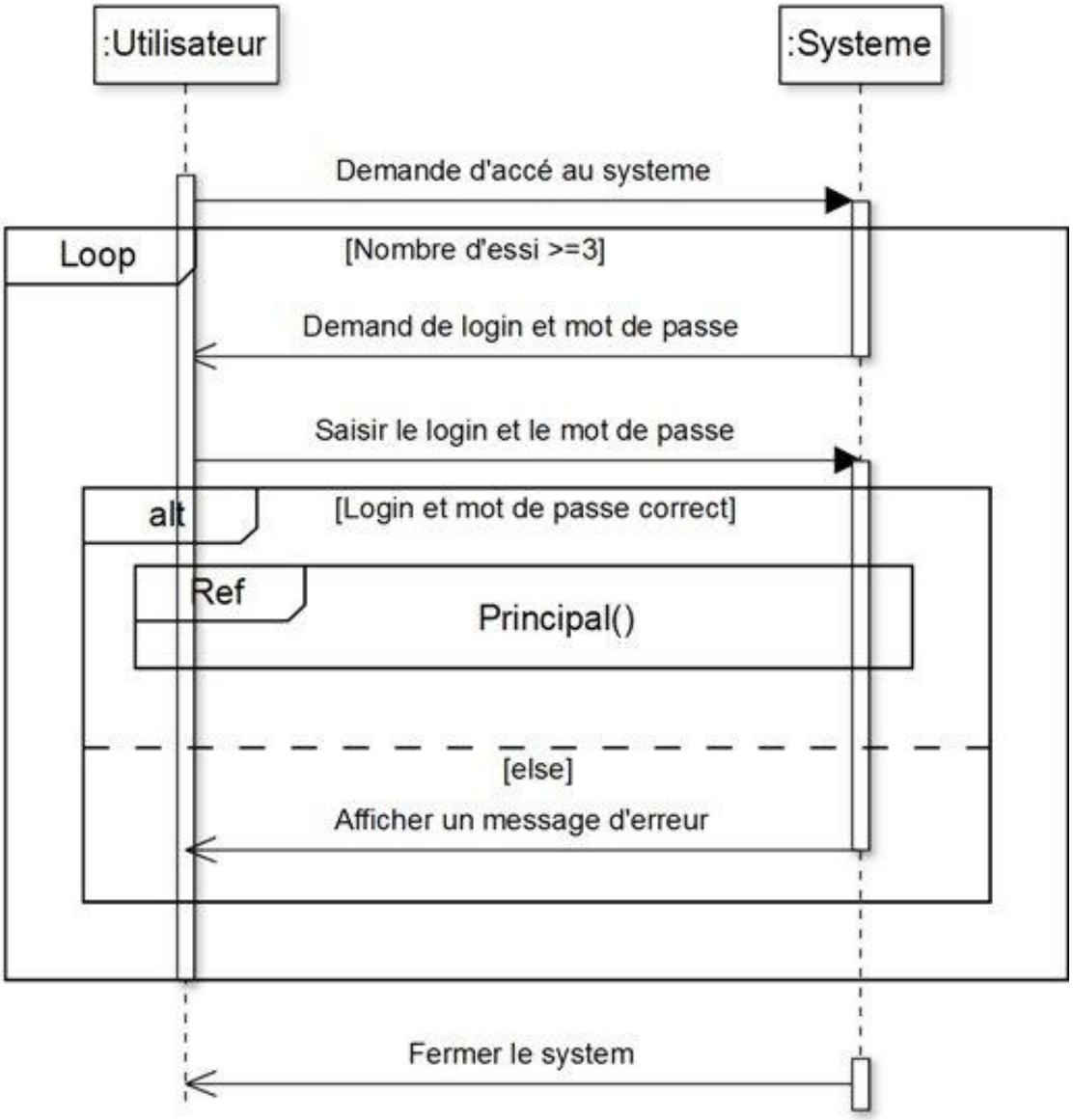

Figure 4.2. Diagramme de séquence système authentification

4.5.2. Le diagramme de séquence initialisation enseignant :

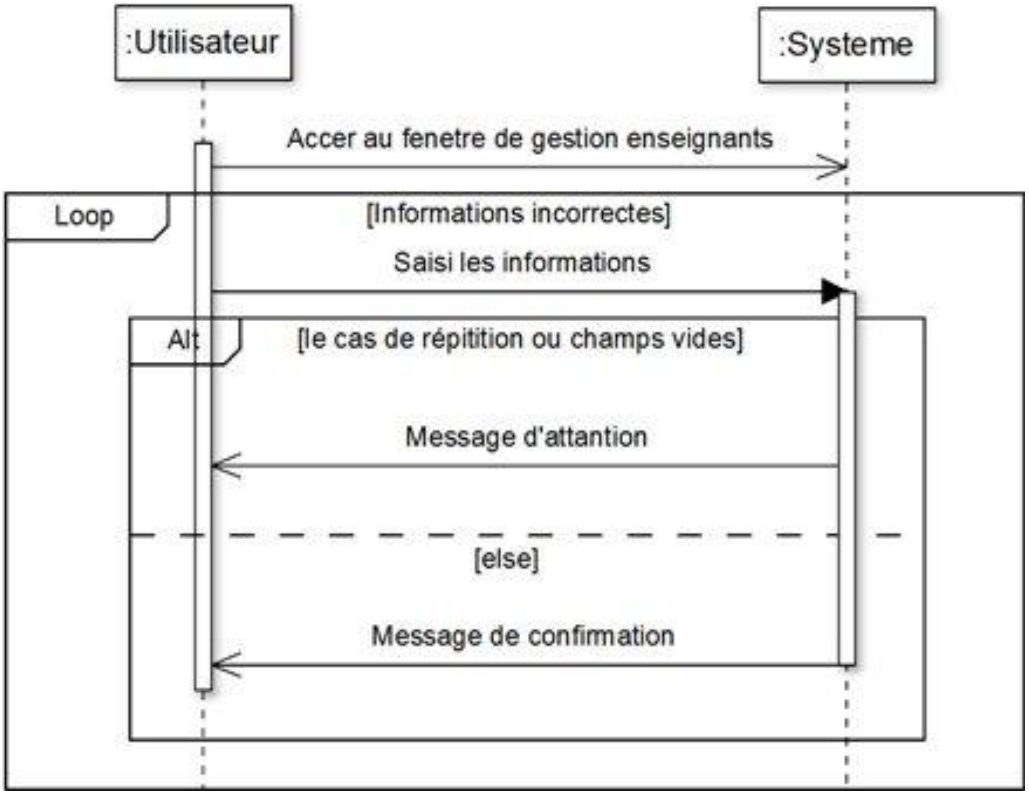

Figure 4.3. Le diagramme de séquence initialisation enseignant

4.5.3. Le diagramme de séquence initialisation salle :

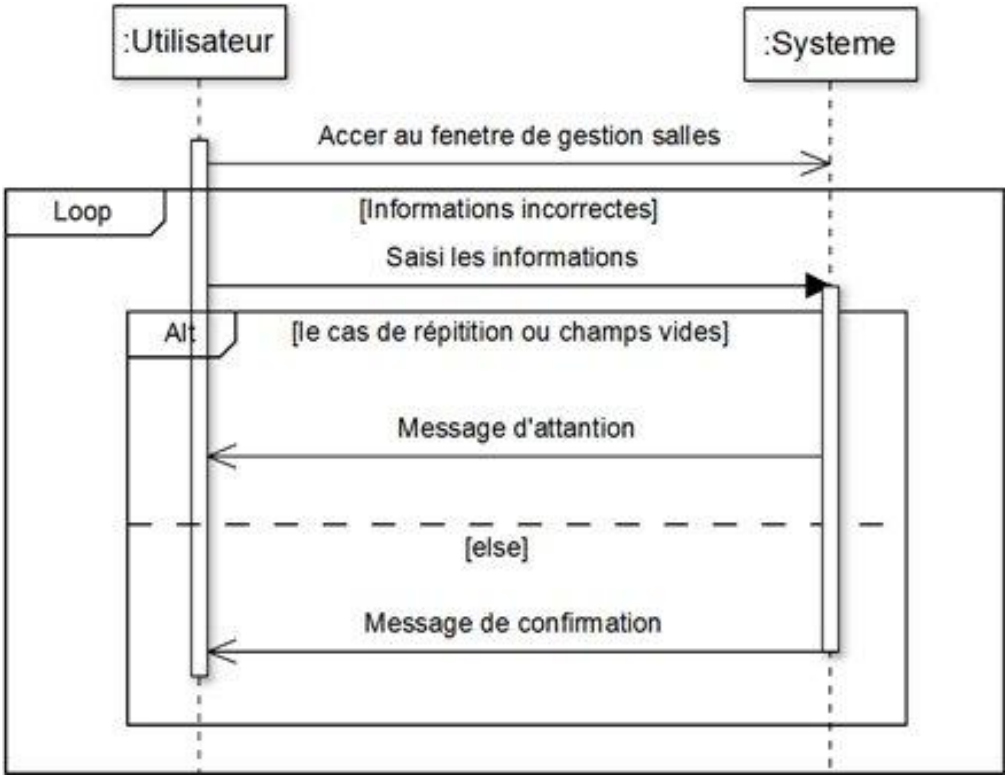

Figure 4.4. Le diagramme de séquence initialisation salle

4.5.4. Le diagramme de séquence initialisation module :

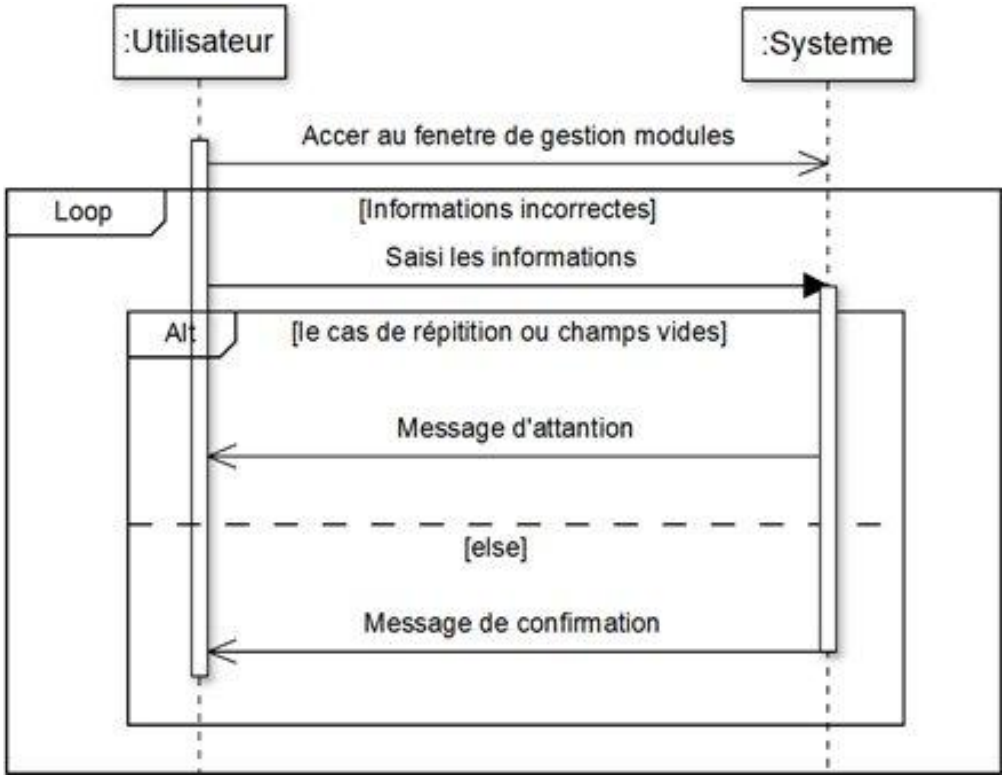

Figure 4.5. Le diagramme de séquence initialisation module

4.5.5. Le diagramme de séquence initialisation groupe :

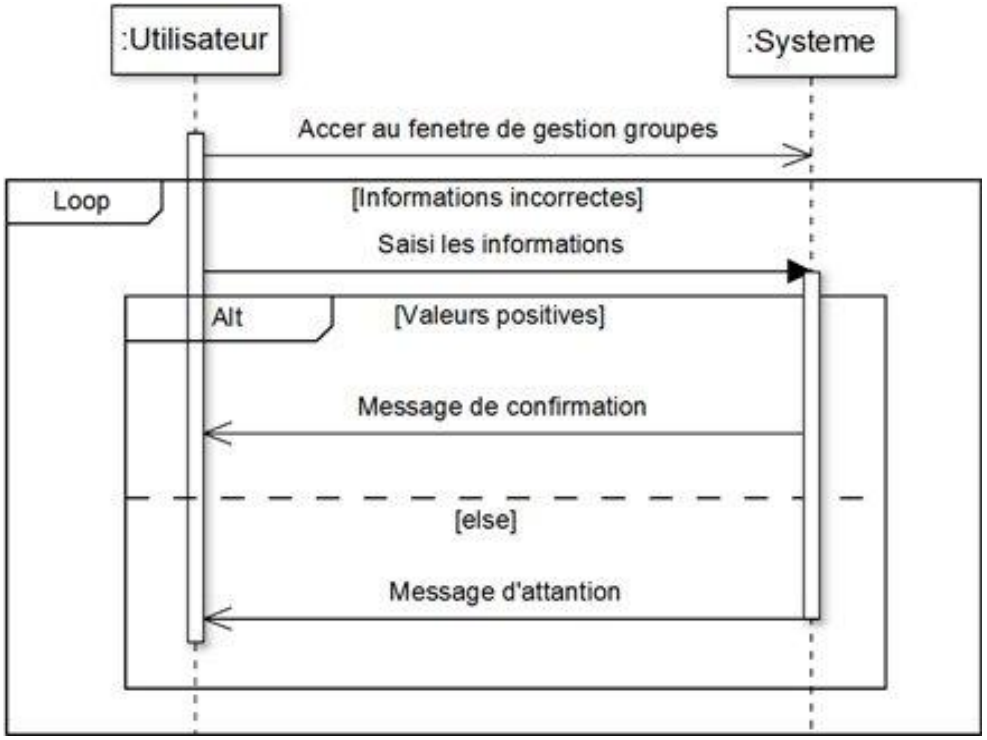

Figure 4.6. Le diagramme de séquence initialisation groupe

4.5.6. Le diagramme de séquence affectation :

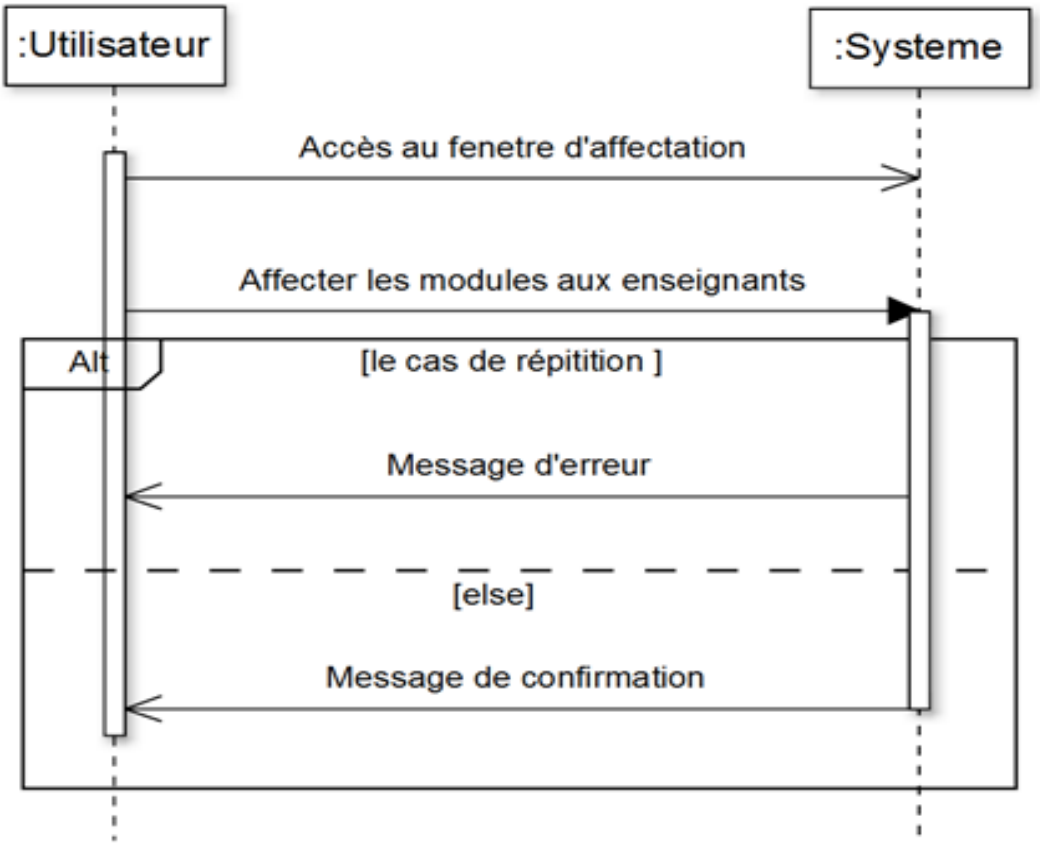

Figure 4.7. Diagramme de séquence affectation

4.5.7. Diagramme de séquence d'emploi du temps global :

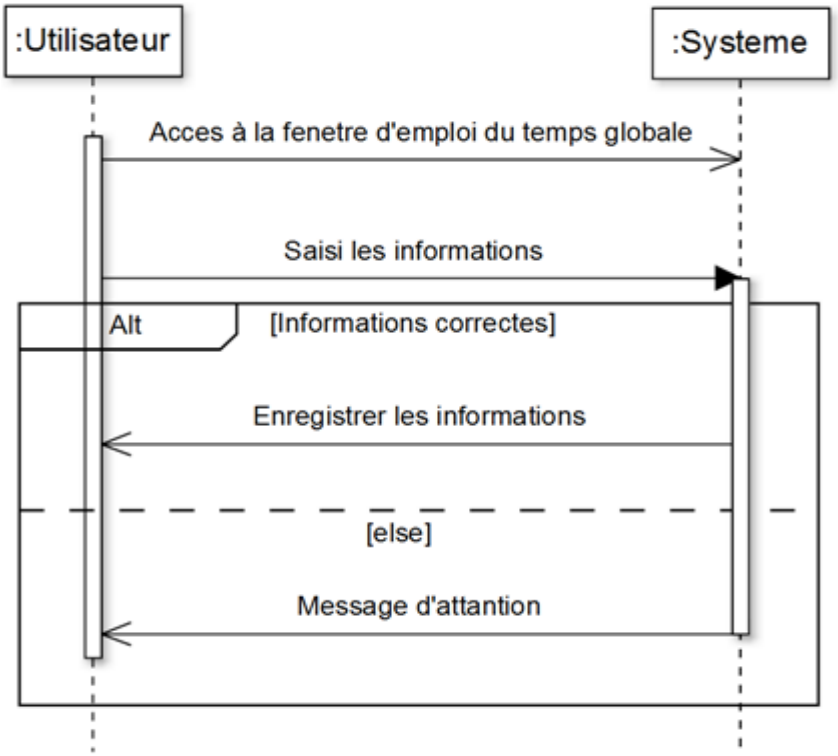

Figure 4.8. Diagramme de séquence d'emploi du temps global

4.5.8. Diagramme de séquence d'emploi du temps salle :

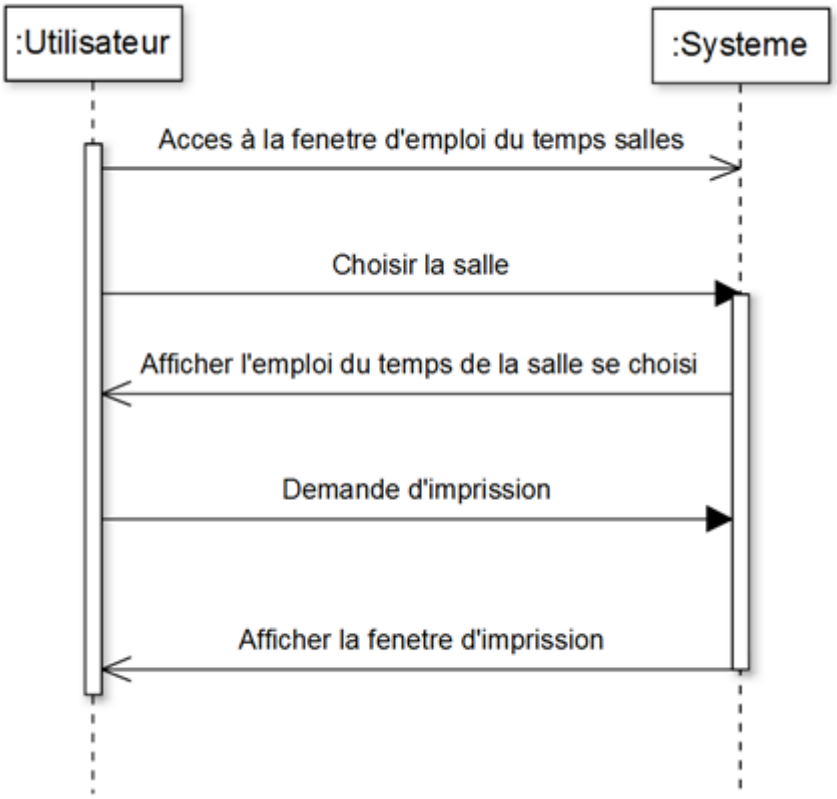

Figure 4.9. Diagramme de séquence d'emploi du temps salle

4.5.9. Diagramme de séquence d'emploi du temps enseignant :

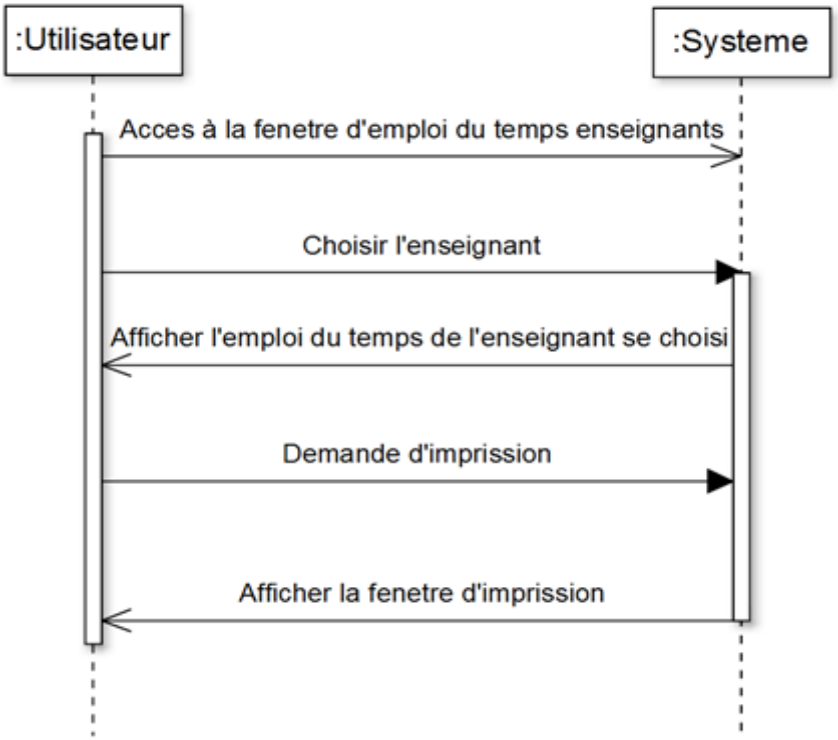

Figure 4.10. Diagramme de séquence d'emploi du temps enseignant

4.5.10. Diagramme de séquence d'emploi du temps filière :

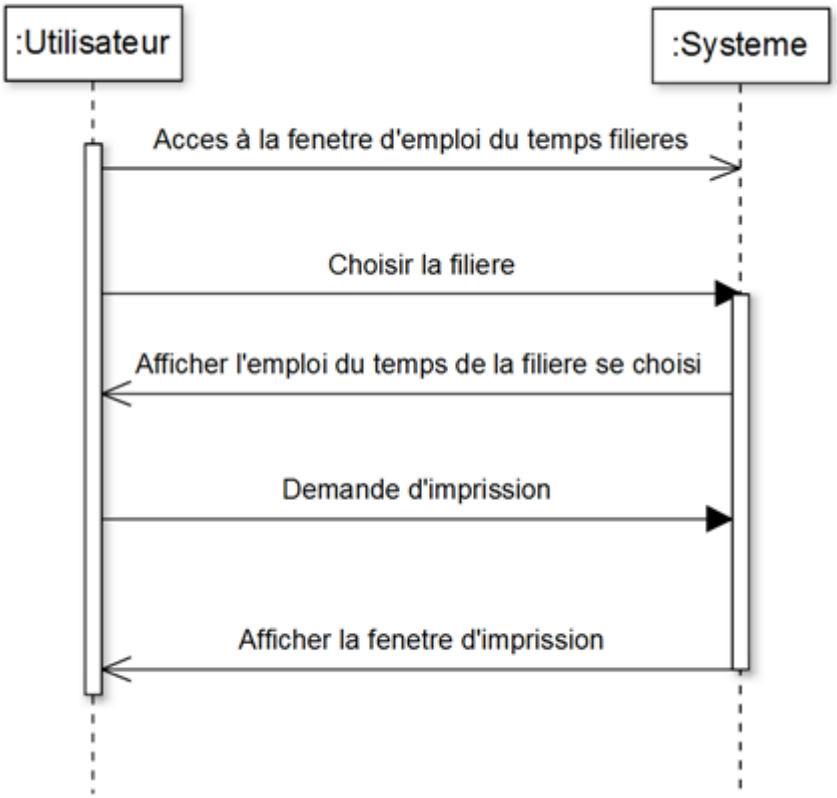

Figure 4.11. Diagramme de séquence d'emploi du temps filière

4.6. Diagramme d'activité du logiciel :

4.6.1. Diagramme d'activité authentification:

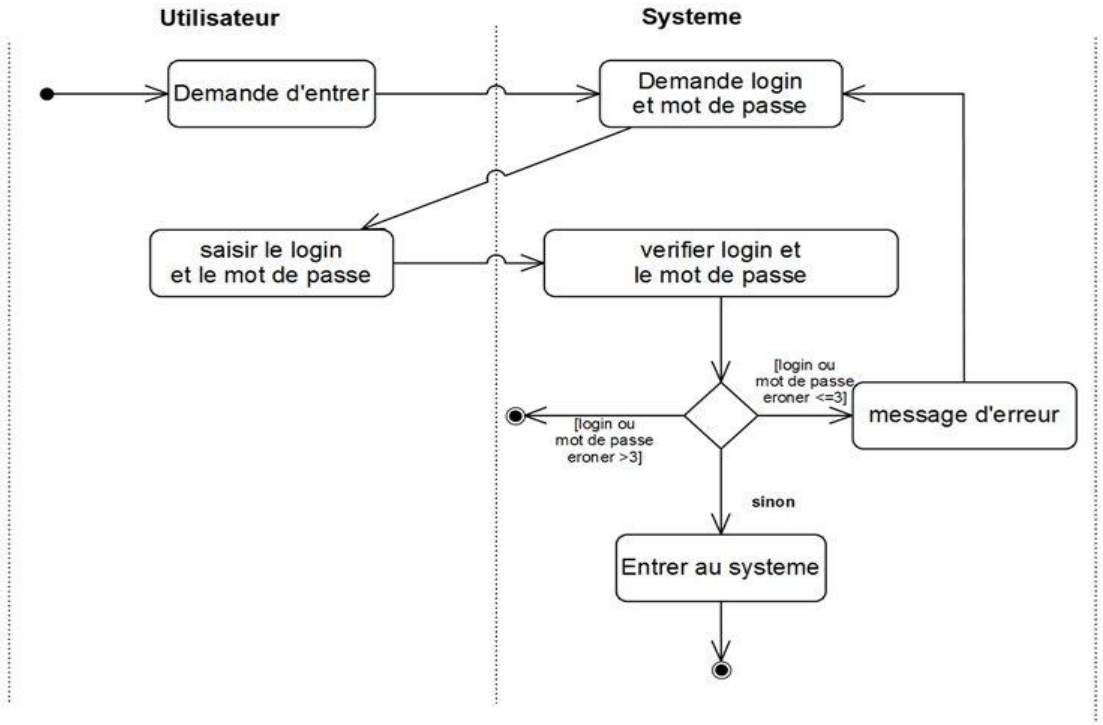

Figure 4.12. Diagramme d'activité authentification

4.6.2. Le diagramme d'activité initialisation enseignant :

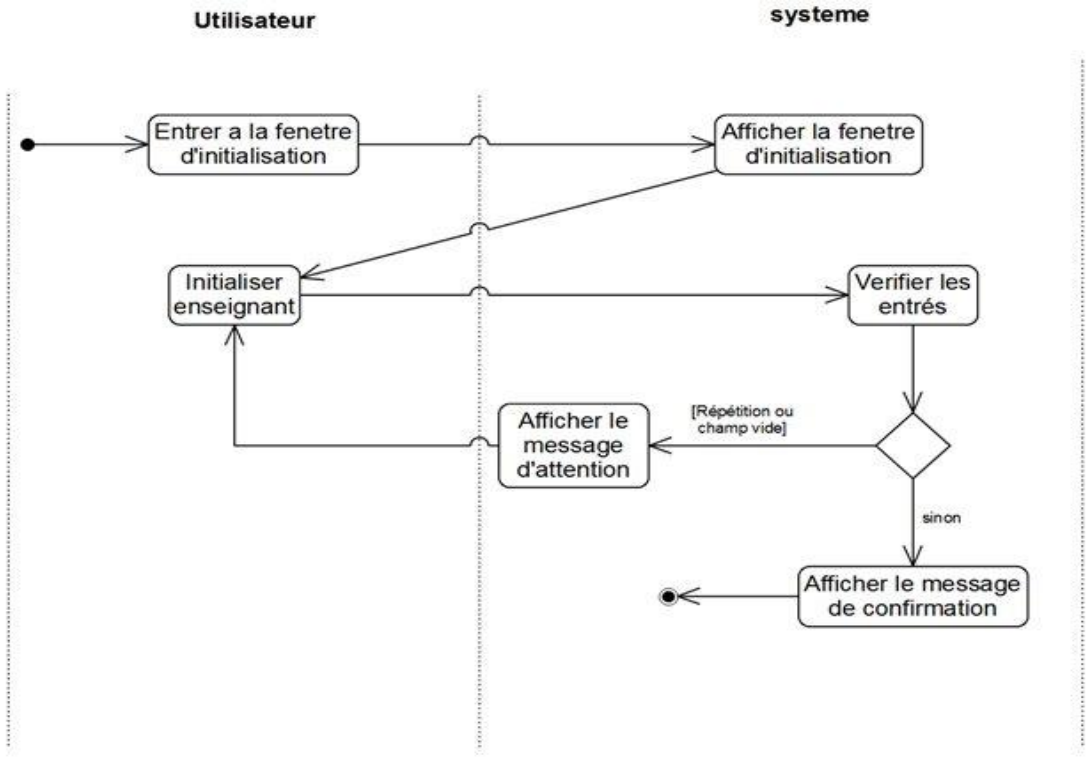

Figure 4.13. Diagramme d'activité initialisation enseignant

4.6.3. Le diagramme d'activité initialisation salle :

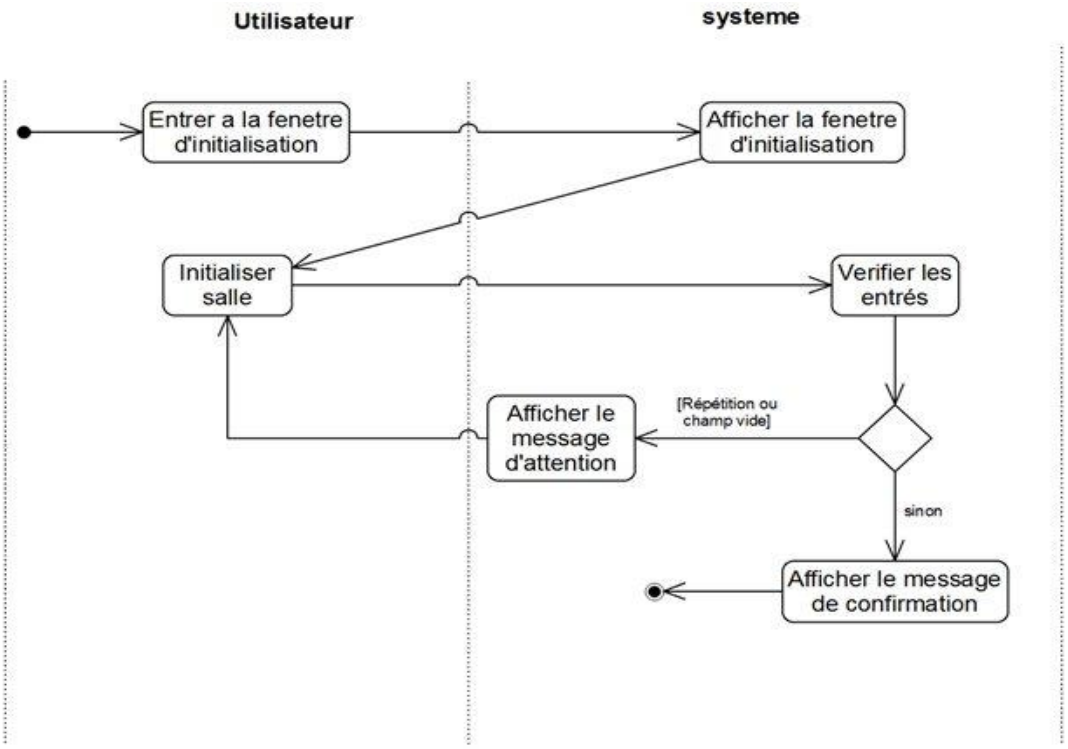

Figure 4.14. Diagramme d'activité initialisation salle

4.6.4. Le diagramme d'activité initialisation module :

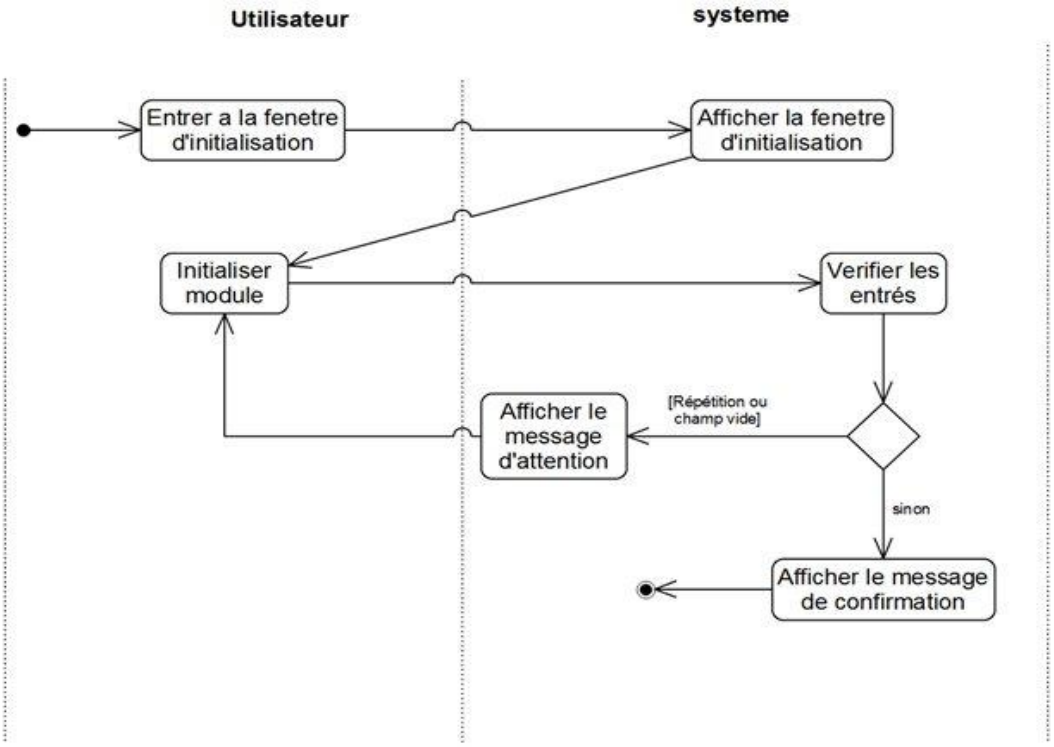

Figure 4.15. Diagramme d'activité initialisation module

4.6.5. Le diagramme d'activité initialisation groupe :

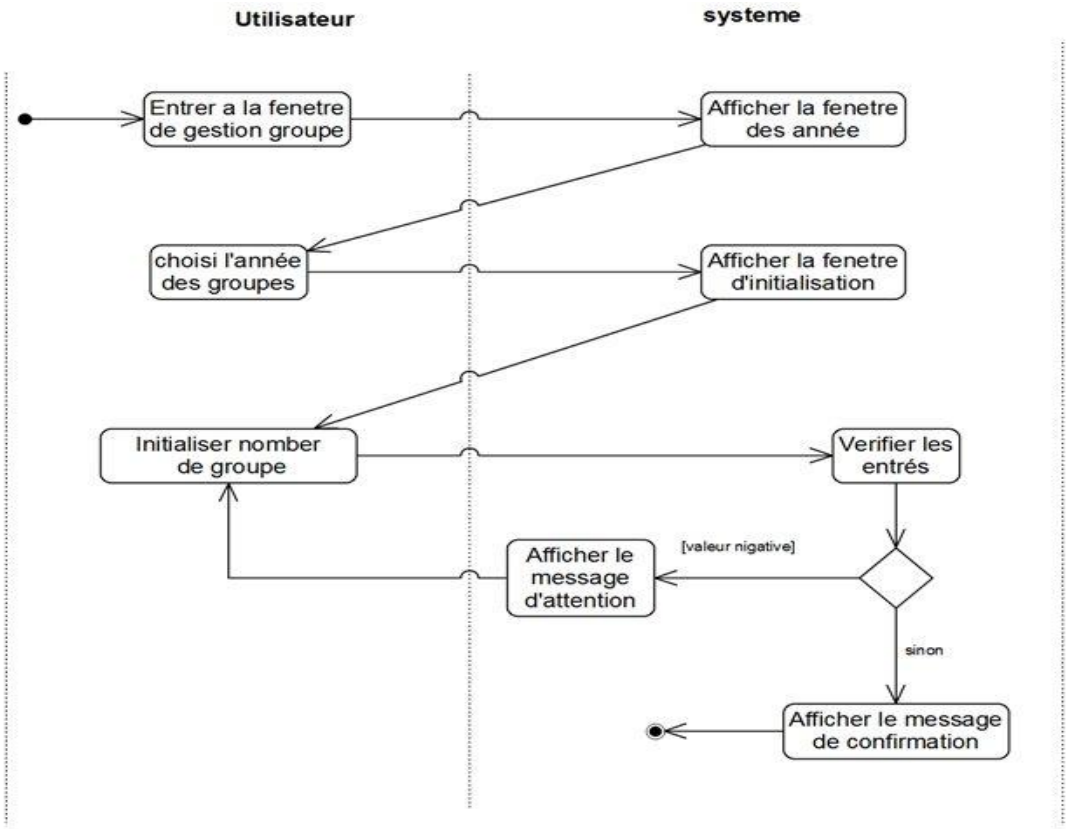

Figure 4.16. Diagramme d'activité initialisation groupe

4.6.6. Diagramme d'activité affectation :

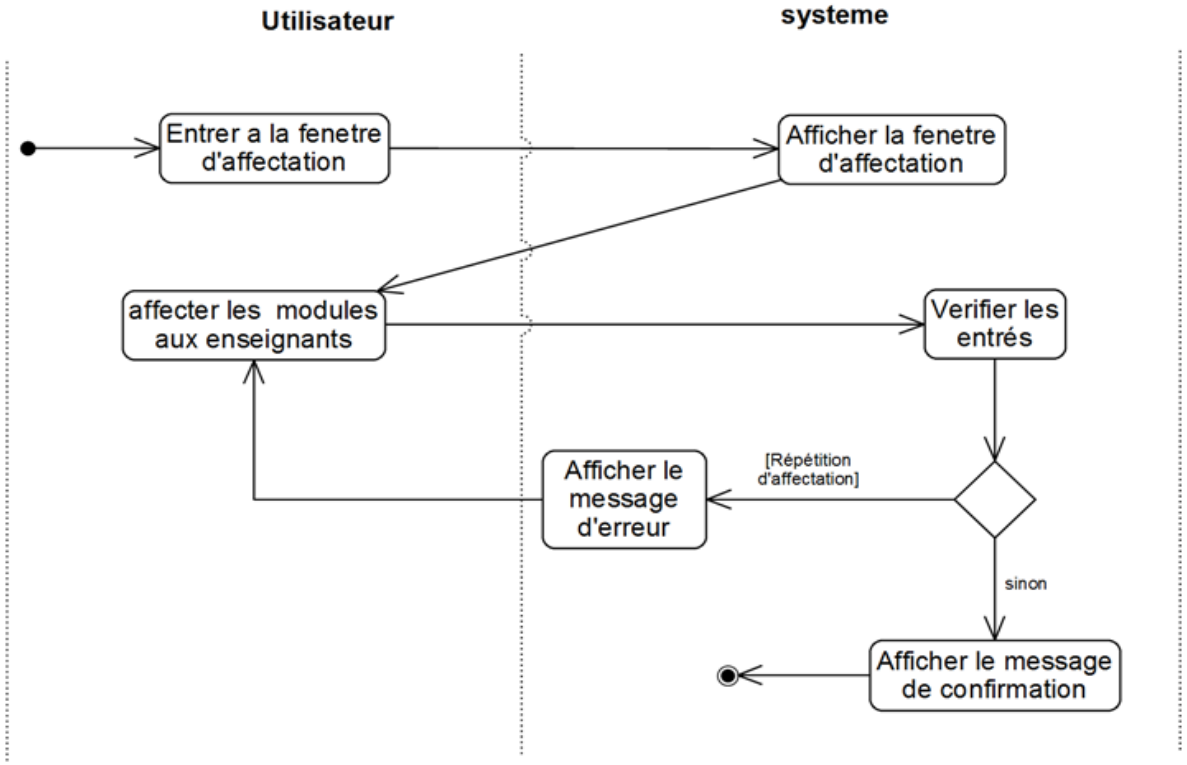

Figure 4.17. Diagramme d'activité affectation

4.6.7. Diagramme d'activité d'emploi du temps global :

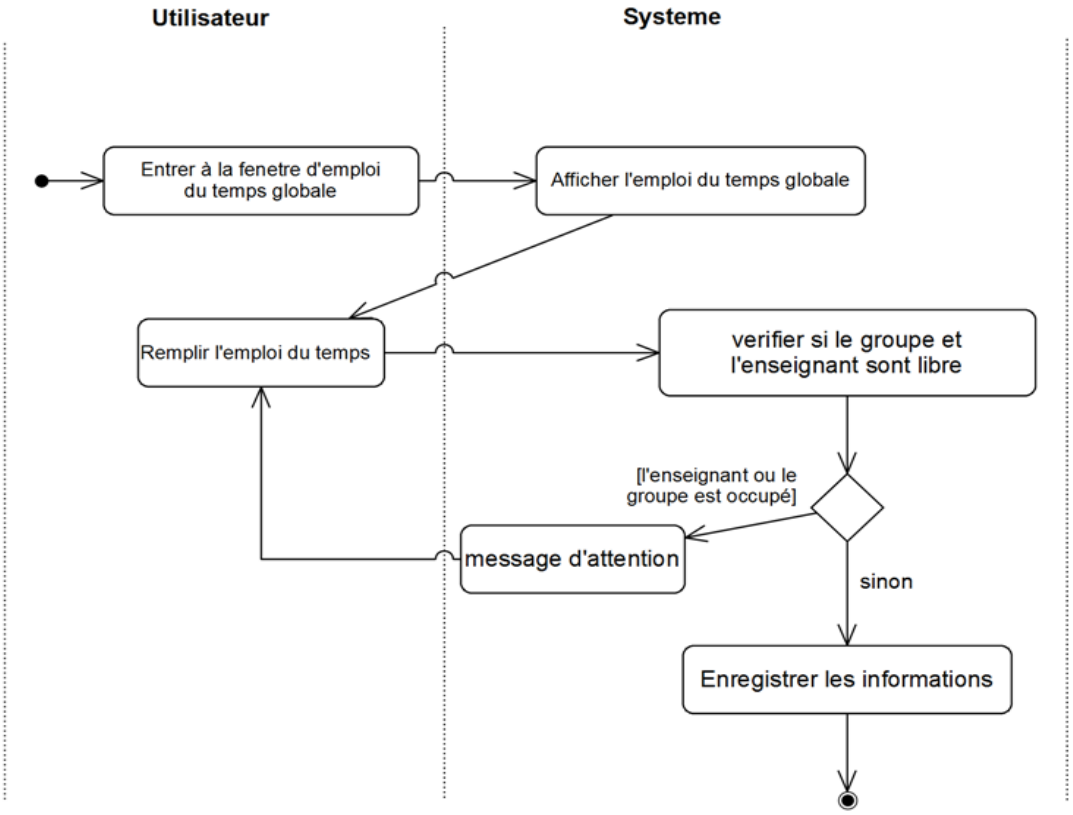

Figure 4.18. Diagramme d'activité d'emploi du temps

4.6.8. Diagramme d'activité d'emploi du temps salle :

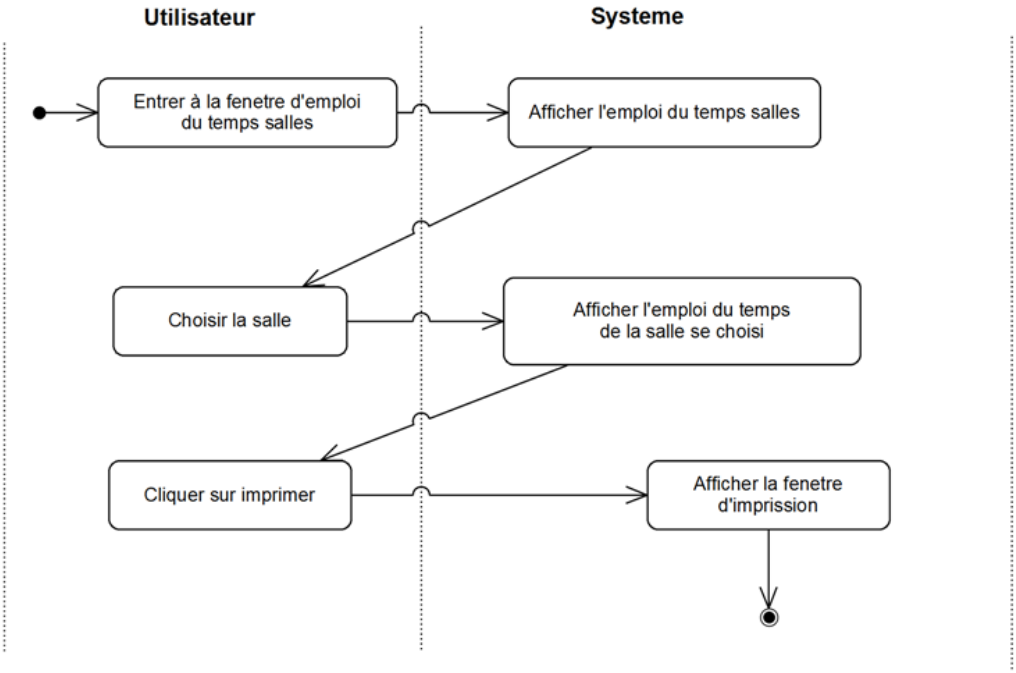

Figure 4.19. Diagramme d'activité d'emploi du temps salle

4.6.9. Diagramme d'activité d'emploi du temps enseignant :

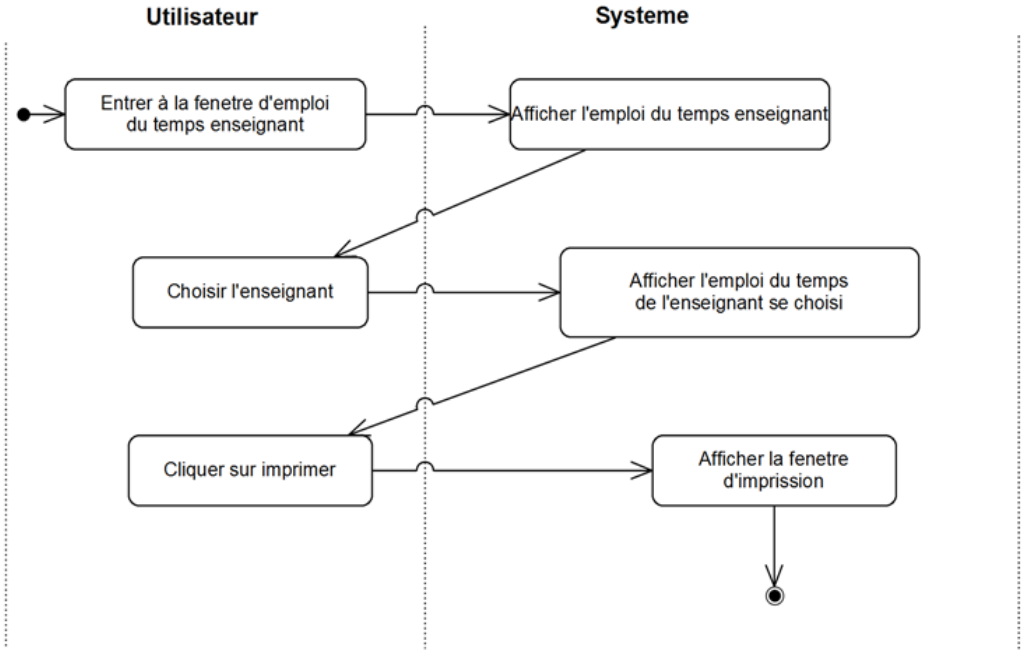

Figure 4.20. Diagramme d'activité d'emploi du temps enseignant

4.6.10. Diagramme d'activité d'emploi du temps filière :

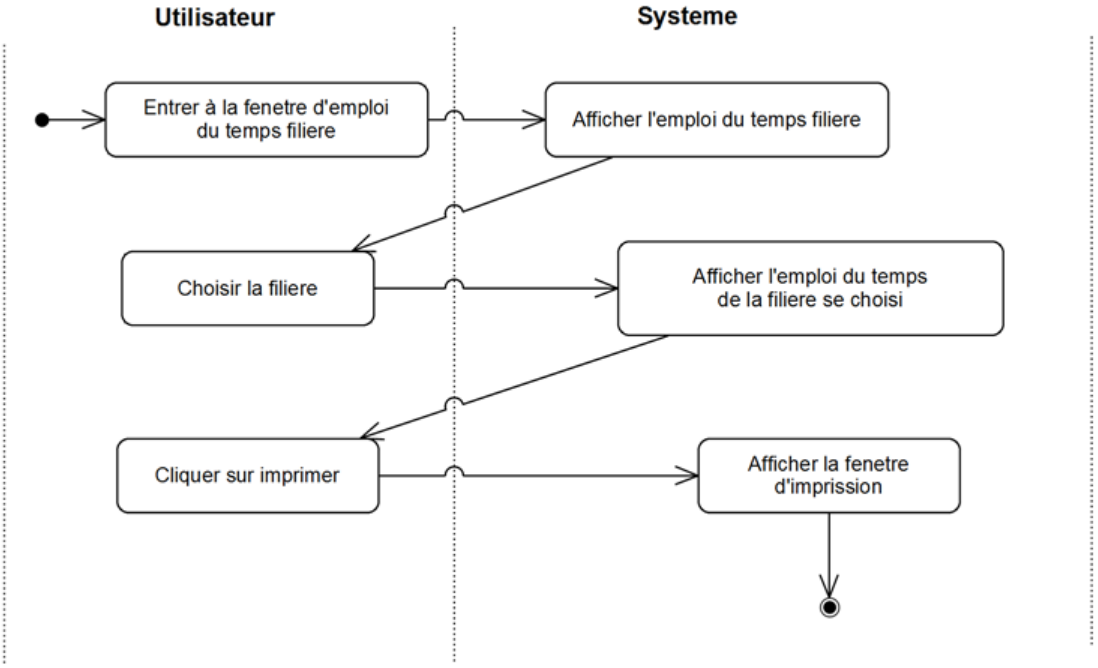

Figure 4.21. Diagramme d'activité d'emploi du temps filière

4.7. Diagramme de classe :

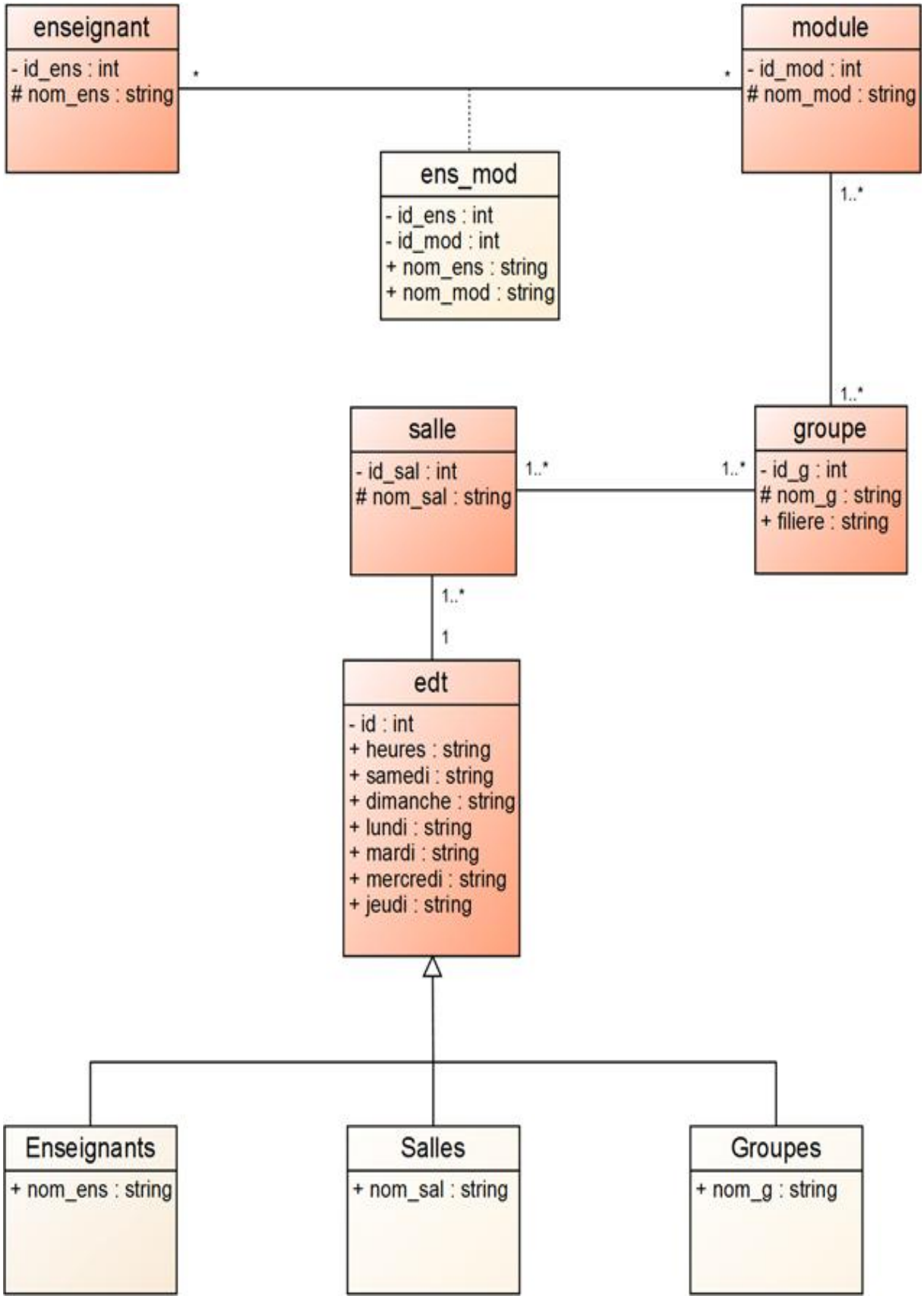

Figure 4.22. Diagramme de classe

4.8. Conclusion :

Dans ce chapitre, nous avons mené une étude détaillée de notre projet tout en élaborant les différents diagrammes de conception, en commençant par la définition des acteurs et leurs cas d'utilisation et en terminant par le diagramme de classe.

Chapitre 5 : Implémentation

5.1. Introduction :

Dans ce chapitre nous allons présenter les fenêtres de notre application, et on va expliquer tout les différents outils d'utilisation et leurs fonctions.

5.2. Fenêtre authentification : permet d'accéder au system avec un nom d'utilisateur et un mot de passe.

Figure 5.1. Fenêtre d'authentification

1. Champ pour saisir le nom d'utilisateur.
2. Champ pour saisir le mot de passe.
3. Accès à la fenêtre de modification de mot de passe.
4. Bouton pour quitte.
5. Bouton pour accès au système.

5.3. Fenêtre de modification du mot de passe : permet à l'utilisateur de modifier le mot de passe un ou plusieurs fois.

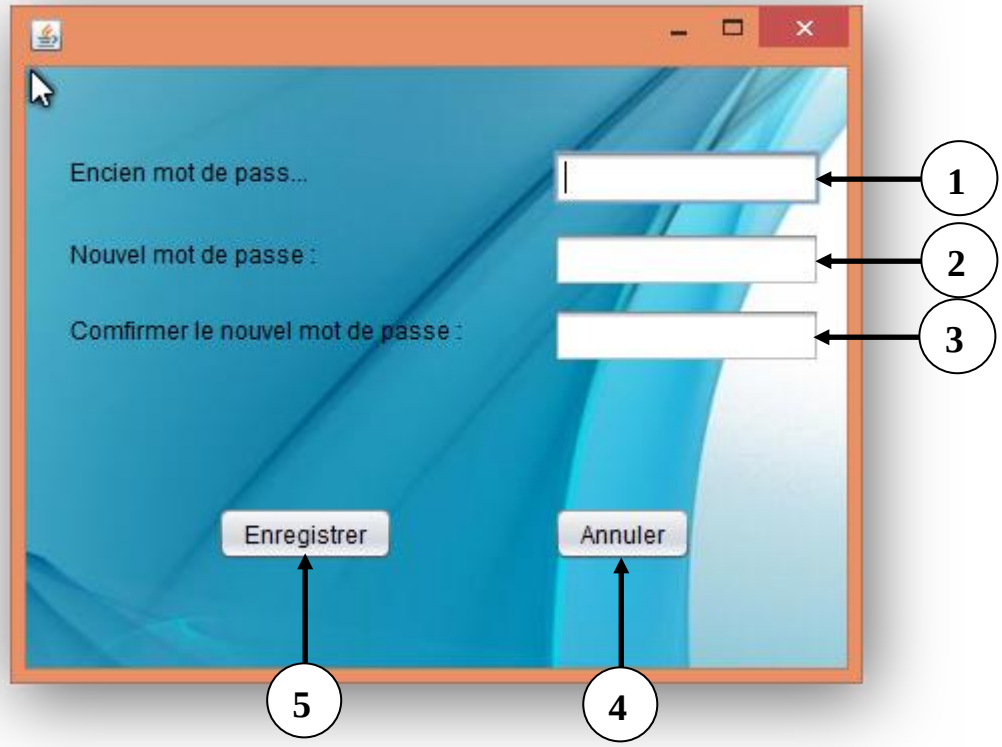

Figure 5.2. Fenêtre de modifier le mot de passe

1. Champ pour saisir l'ancien mot de passe.
2. Champ pour saisir le nouvel mot de passe.
3. Champ pour confirmer le nouvel mot de passe.
4. Bouton pour annuler la modification du mot de passe.
5. Bouton pour enregistrer le nouvel mot de passe.

5.4. Fenêtre du logiciel (page d'accueil): La première fenêtre qui affichée.

Figure 5.3. Fenêtre d'accueil

1. Bouton de menu pour accès à la fenêtre d'initialisation des ressources.
2. Bouton de menu pour l'aide.
3. Bouton pour afficher la fenêtre d'emploi du temps globale.
4. Bouton pour afficher la fenêtre d'emploi du temps des salles.
5. Bouton pour afficher la fenêtre d'emploi du temps des enseignants.
6. Bouton pour afficher la fenêtre d'emploi du temps des filières.

5.5. Fenêtre d'initialiser les enseignants : permet à l'utilisateur d'initialiser les informations des enseignants.

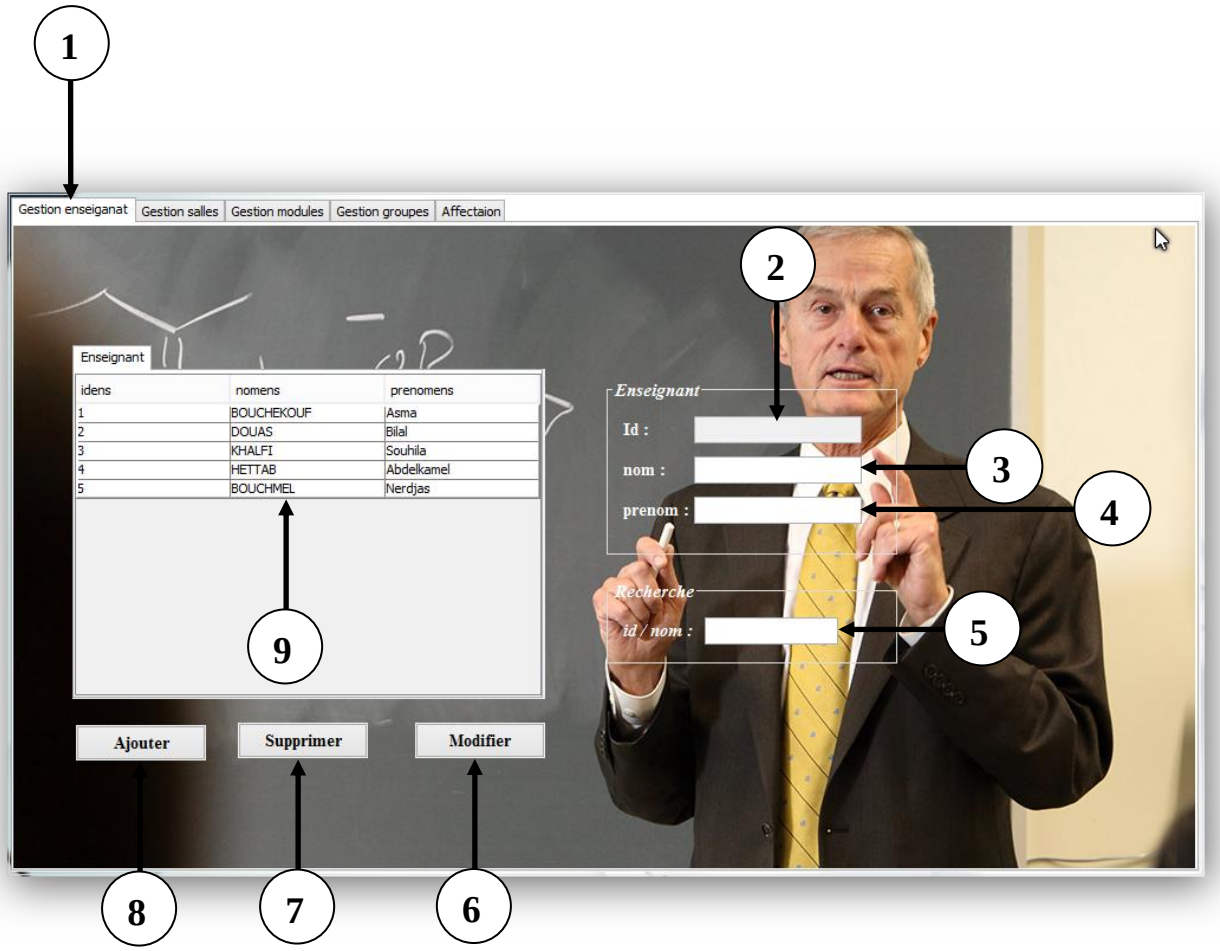

Figure 5.4. Fenêtre d'initialiser les enseignants

1. Bouton pour afficher la fenêtre d'initialiser les enseignants.
2. Champ pour afficher le numéro de l'enseignant.
3. Champ pour saisir le nom de l'enseignant.
4. Champ pour saisir le prénom de l'enseignant.
5. Champ pour rechercher des enseignants par id ou par nom.
6. Bouton pour modifier les informations de l'enseignant.
7. Bouton pour supprimer des enseignants.
8. Bouton pour enregistrer les enseignants.
9. Le tableau qui contient les informations des enseignants.

5.6. Fenêtre d'initialiser les salles : permet à l'utilisateur d'initialiser les salles.

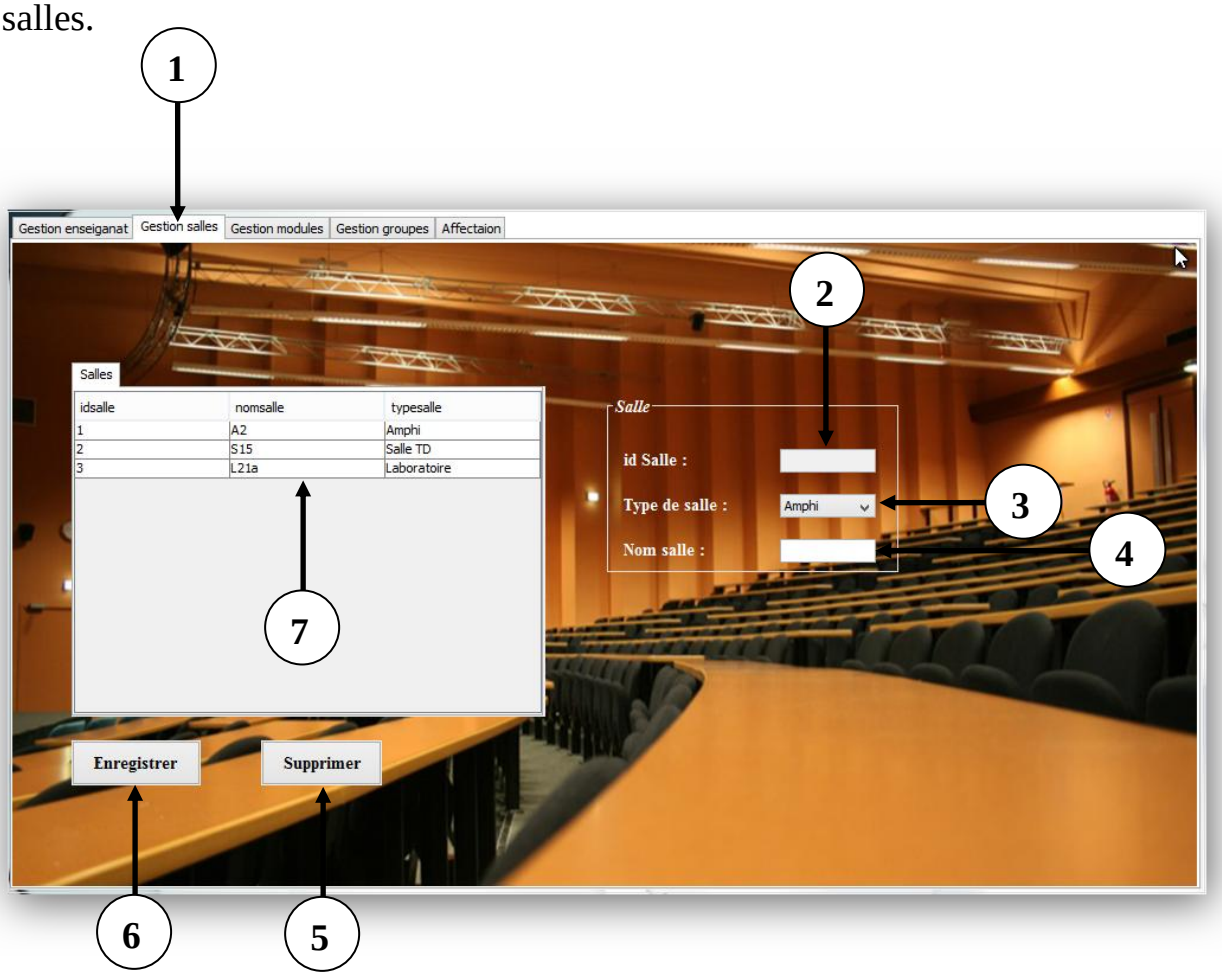

Figure 5.5. Fenêtre d'initialiser les salles

1. Bouton pour afficher la fenêtre d'initialiser les salles.
2. Champ pour afficher le id de la salle.
3. Bouton pour choisir le type de la salle.
4. Champ pour saisir le nom de la salle.
5. Bouton pour supprimer les salles.
6. Bouton pour enregistrer les salles.
7. Le tableau qui contient les informations des salles.

5.7. Fenêtre d'initialiser les modules : permet à l'utilisateur d'initialiser les modules.

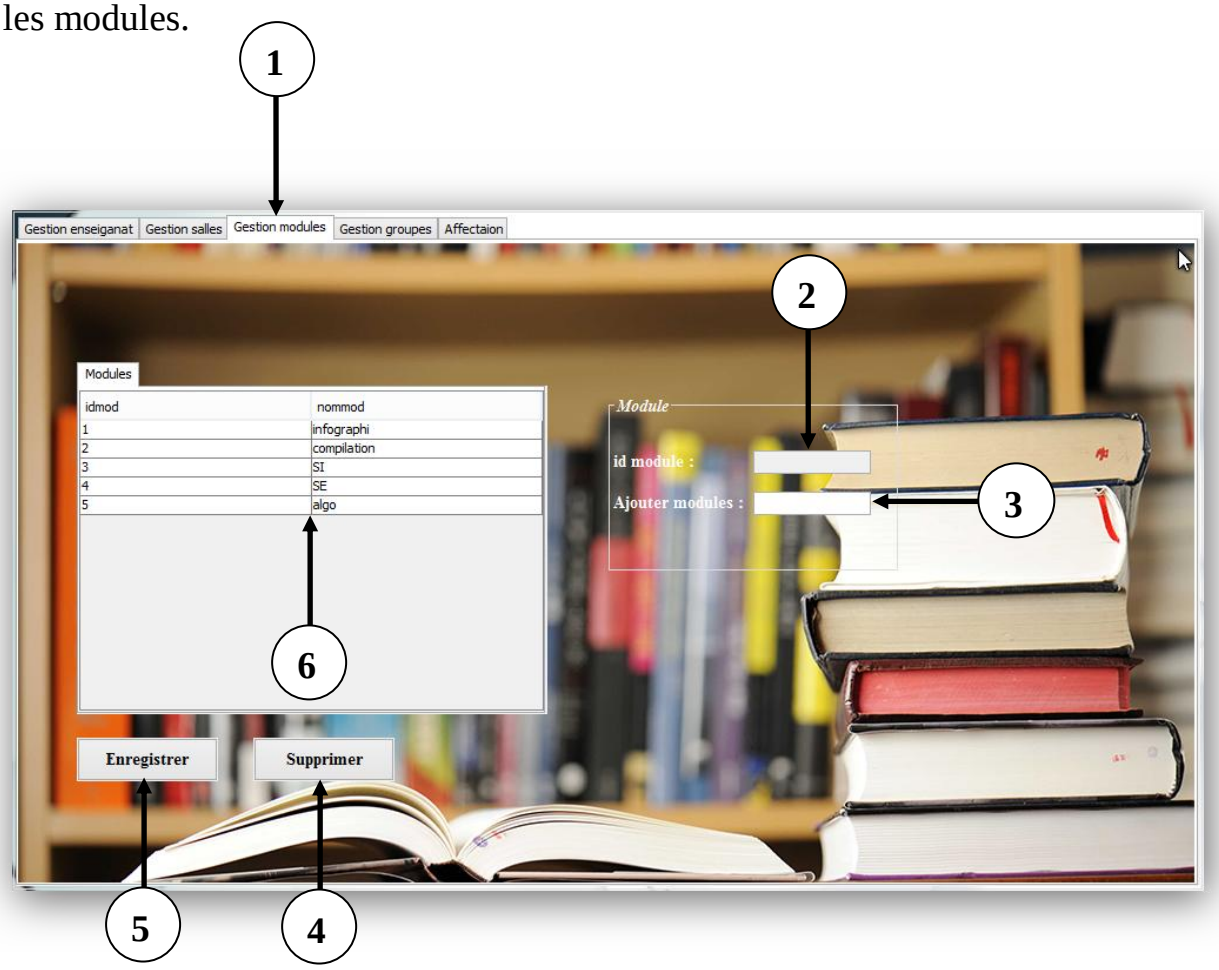

Figure 5.6. Fenêtre d'initialiser les modules

1. Bouton pour afficher la fenêtre d'initialiser les modules.
2. Champ pour afficher le id du module.
3. Champ pour saisir le nom du module.
4. Bouton pour supprimer les modules.
5. Bouton pour enregistrer les modules.
6. Le tableau qui contient les informations des modules.

5.8. Fenêtre d'initialiser les groupes : permet à l'utilisateur d'initialiser les groupes.

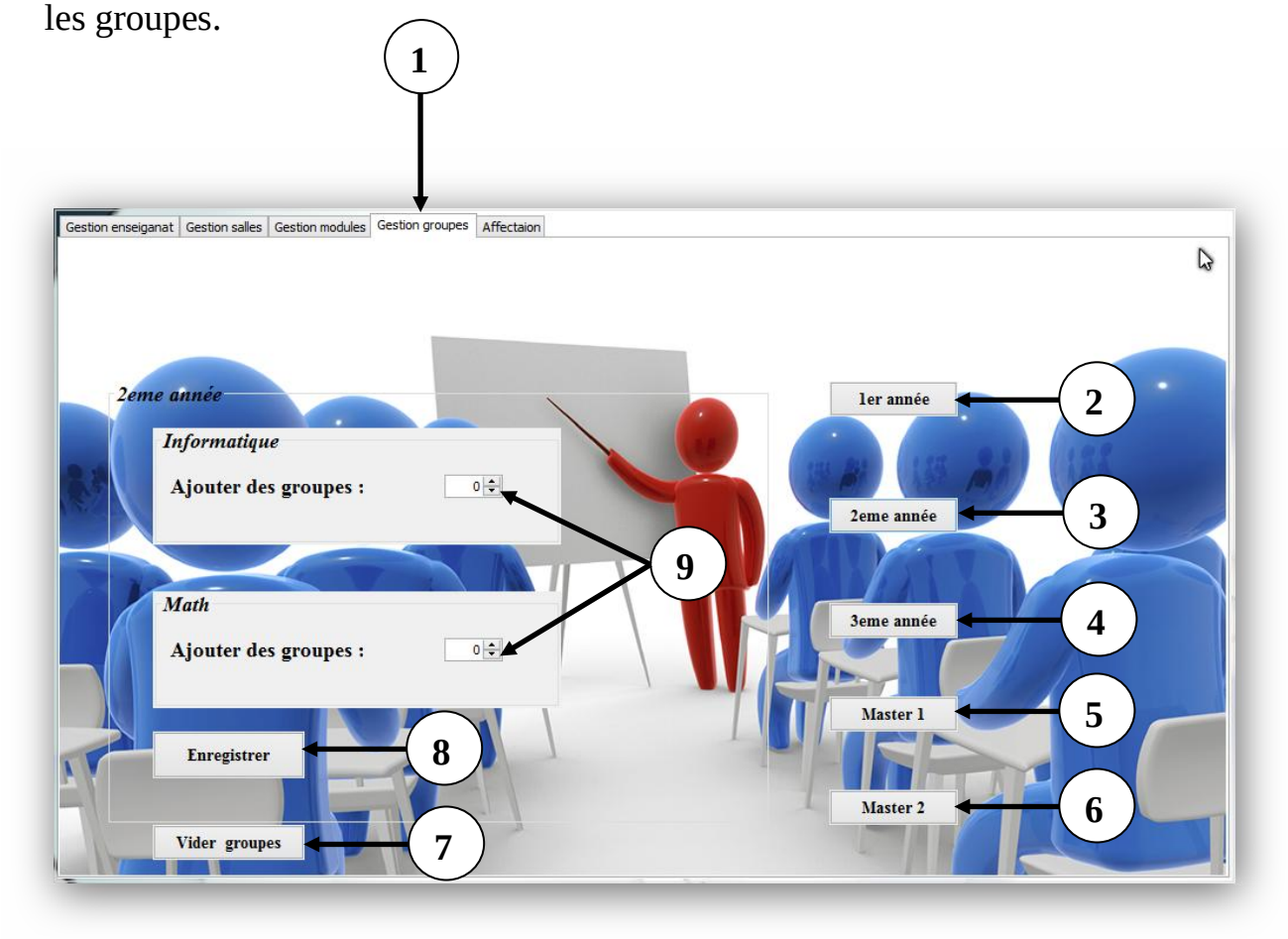

Figure 5.7. Fenêtre d'initialiser les groupes

1. Bouton pour afficher la fenêtre d'initialiser les groupes.
2. Bouton pour initialiser les groupes du 1ére année.
3. Bouton pour initialiser les groupes du 2éme année.
4. Bouton pour initialiser les groupes du 3éme année.
5. Bouton pour initialiser les groupes du master 1.
6. Bouton pour initialiser les groupes du master 2.
7. Bouton pour supprimer les groupes.
8. Bouton pour enregistrer les groupes.
9. Bouton pour choisir le nombre des groupes a ajouté.

5.9. Fenêtre d'affectation : permet à l'utilisateur affecter les modules aux enseignants correspondants.

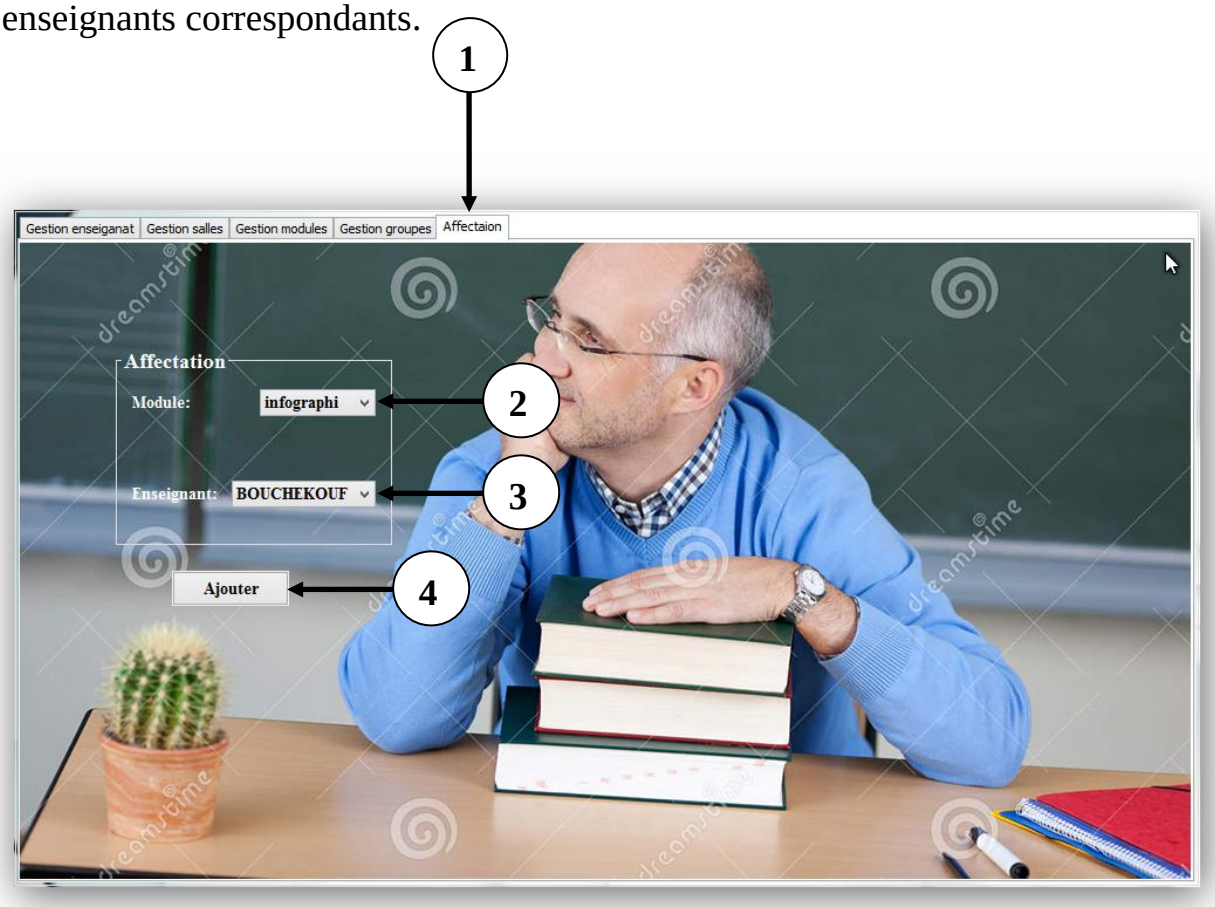

Figure 5.8. Fenêtre d'affecter les modules aux enseignants

1. Bouton pour afficher la fenêtre d'affectation.
2. Bouton pour choisir le module.
3. Bouton pour choisir l'enseignant.
4. Bouton pour enregistrer l'affectation.

5.10. Fenêtre d'emploi du temps global : permet à l'utilisateur de crée l'emploi du temps global.

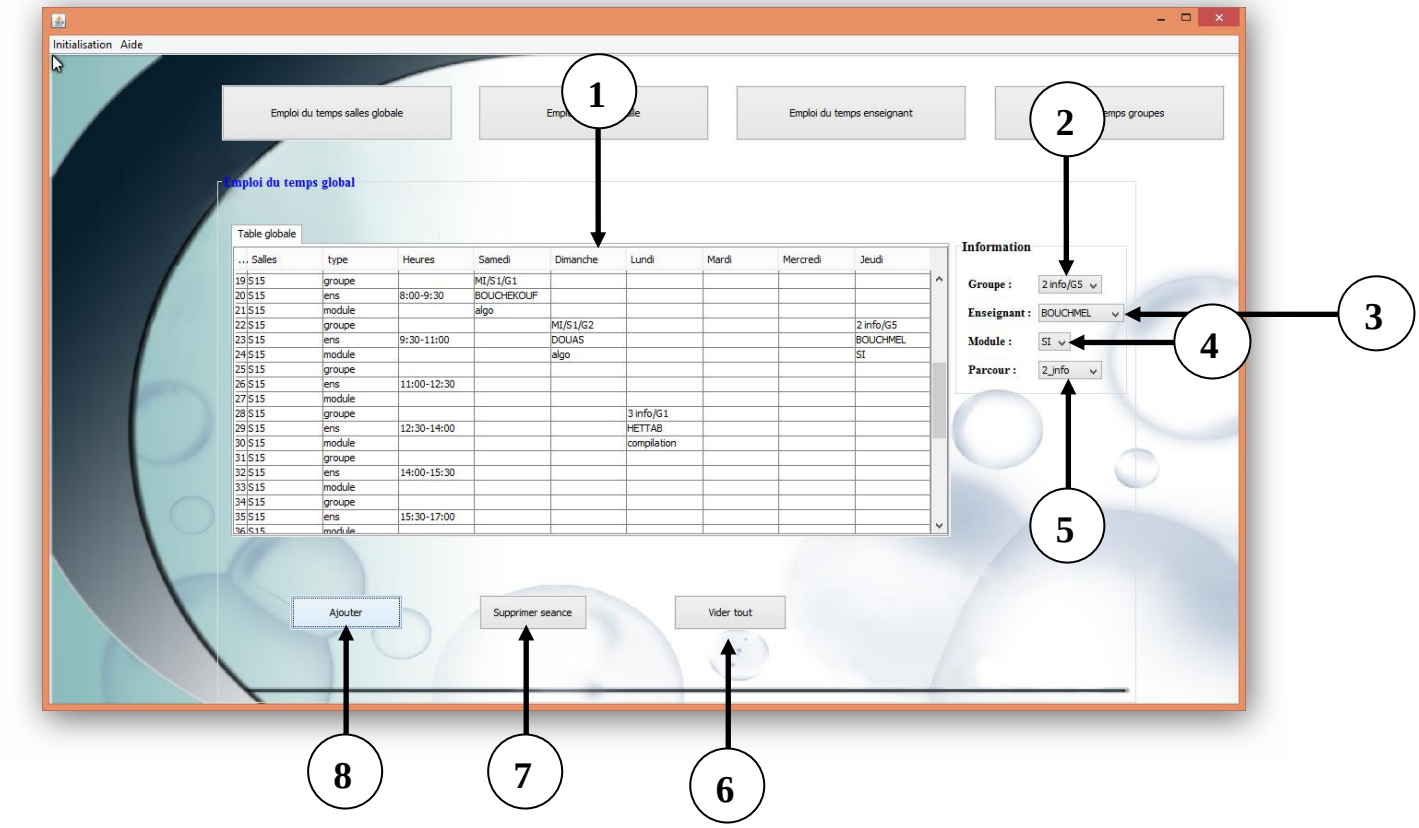

Figure 5.9. Fenêtre d'emploi du temps global

1. Le tableau qui contient l'emploi du temps global.
2. Bouton pour choisir le groupe.
3. Bouton pour choisir l'enseignant.
4. Bouton pour choisir le module.
5. Bouton pour choisir la filière.
6. Bouton pour supprimer tout les séances.
7. Bouton pour supprimer la séance sélectionner.
8. Bouton pour enregistrer les séances.

5.11. Fenêtre d'emploi du temps salles: permet à l'utilisateur d'afficher les emplois du temps des salles.

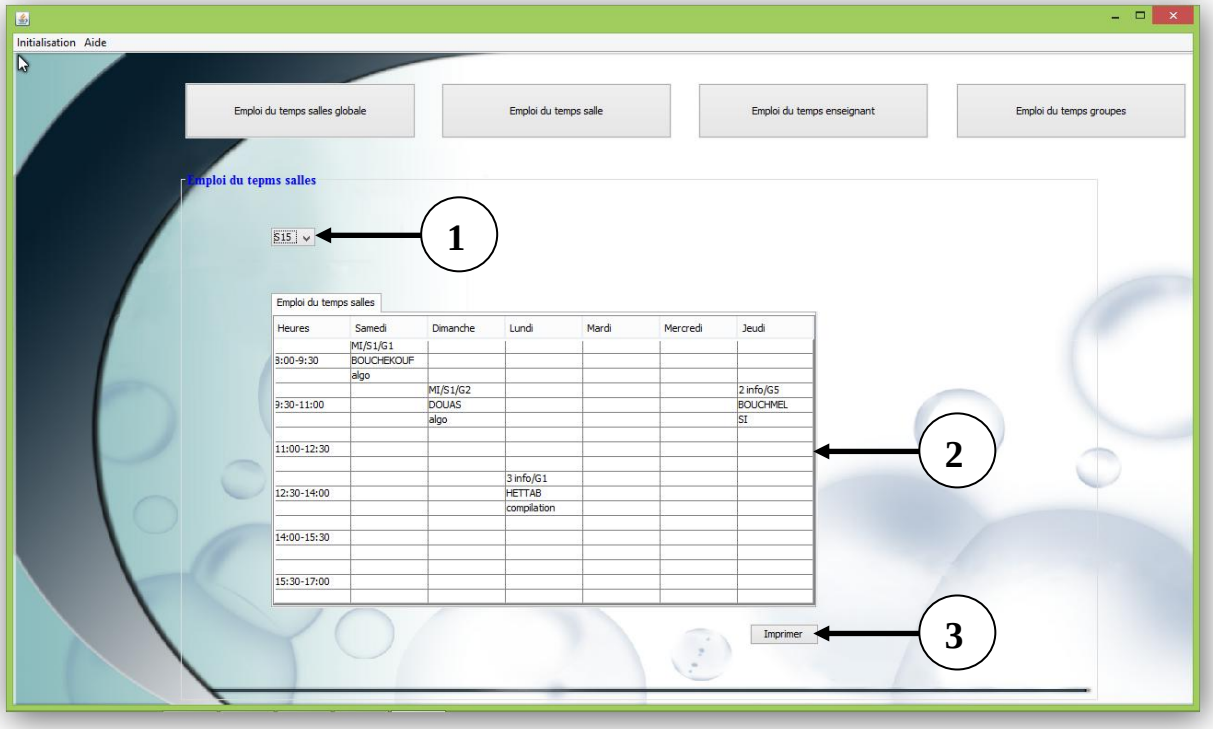

Figure 5.10. Fenêtre d'emploi du temps salle

1. Bouton pour choisir la salle.
2. Le tableau qui contient l'emploi du temps de la salle choisi.
3. Bouton pour imprimer et enregistrer l'emploi du temps de la salle.

5.12. Fenêtre d'emploi du temps enseignant : permet à l'utilisateur d'afficher l'emploi du temps d'un enseignant.

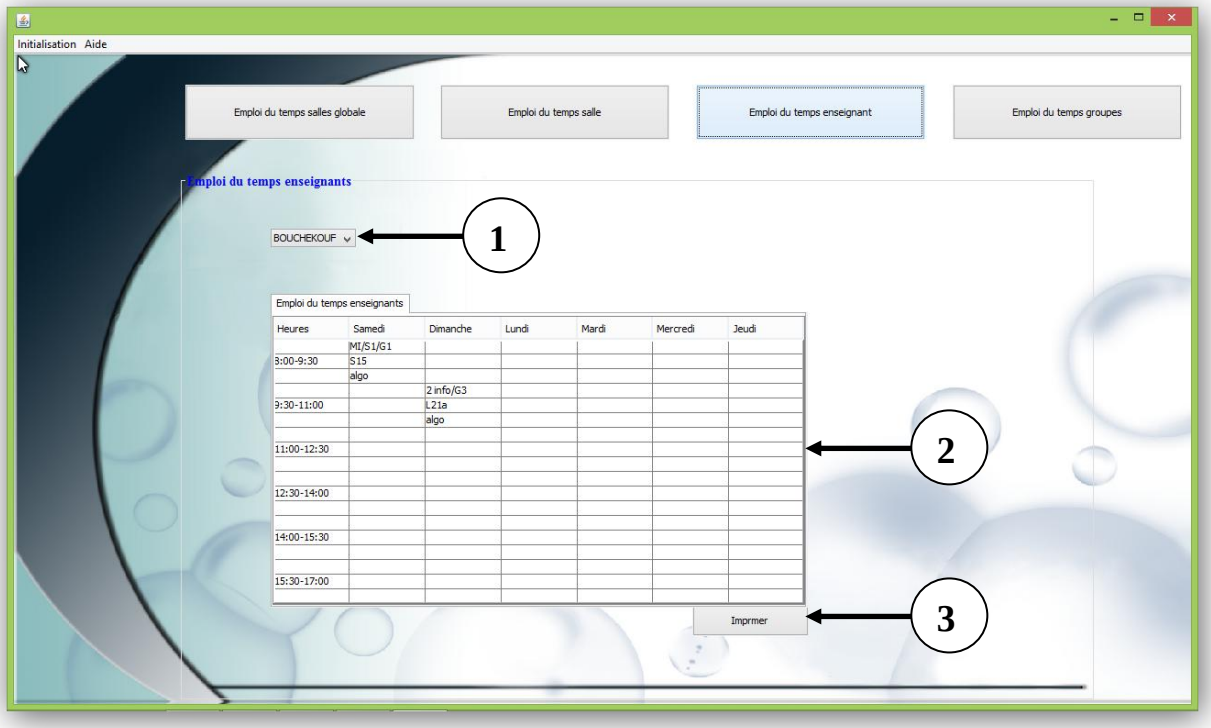

Figure 5.11. Fenêtre d'emploi du temps enseignant

1. Bouton pour choisir l'enseignant.
2. Le tableau qui contient l'emploi du temps de l'enseignant choisi.
3. Bouton pour imprimer et enregistrer l'emploi du temps de l'enseignant.

5.13. Fenêtre d'emploi du temps filière : permet à l'utilisateur d'afficher l'emploi du temps d'une filière.

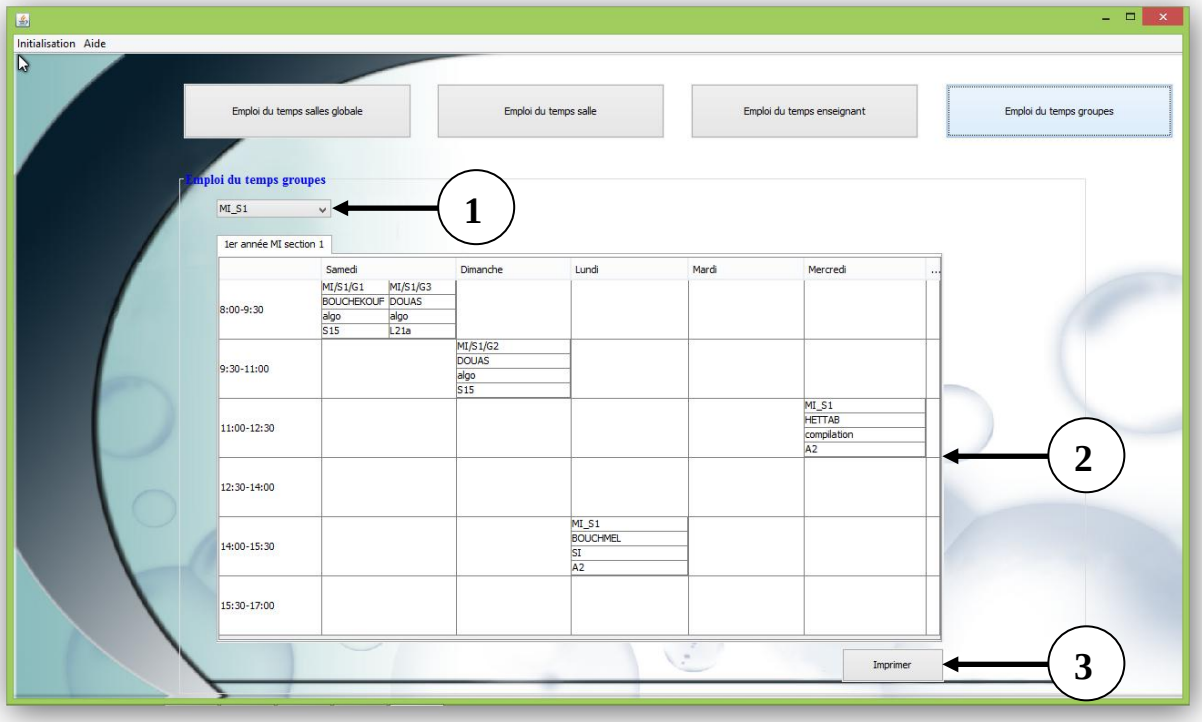

Figure 5.12. Fenêtre d'emploi du temps d'une filière

1. Bouton pour choisir la filière.
2. Le tableau qui contient l'emploi du temps de la filière choisi.
3. Bouton pour imprimer et enregistrer l'emploi du temps de la filière.

5.14. Conclusion :

Ce chapitre est la phase finale de notre travail, il comporte les interfaces qui déterminent de façon plus claire les activités réalisées dans le projet.

Conclusion générale :

Notre projet consiste à la réalisation d'une application graphique qui facilite la gestion des emplois du temps au niveau du département Math et informatique dans le centre universitaire Abdalhafid Boussouf Mila, notre interface est claire et facile à utiliser et tout les outils de l'application est bien fonctionner (initialisation des ressources, création, affichage, et impression des emplois du temps).

Nous avons pu passer du cahier de charge vers la réalisation de l'application Pendant cette période nous avons consolidé nos connaissance dans différents axes à savoir :

- La modélisation des applications de type système d'information via UML, la manipulation de base de données via MySQL (création, suppression, Requête …etc).
- L'utilisation d'un langage de programmation orienté objet (Java) doté d'une interface graphique.
- L'utilisation de quelques outils (Pacestar UML Diagrammer, Photoshop…).

Notre application peut être amélioré au future de département vers instituts ou université ainsi que la possibilité de la transféré à une application client-serveur avec plusieurs utilisateurs.

Les références :

[1] :M. Hettab Abdlkamel **«** La gestion d'un projet de système d'information », centre universitaire Abd Elhafid Boussouf.
[2] : Des classes 2ème année ingénieur « Systèmes d'information».

[3] : M.S. Benhammada « INTRODUCTION AU GENIE LOGICIEL », centre universitaire Abd Elhafid Boussouf, 2010.
[4] : M. Sadek Benhammada. « Diagramme de cas d'utilisation ». centre universitaire de abdelhafid boussouf. 2013.

[5] : « conception et Réalisation d'un site Web dynamique pour l'agence commerciale de Télécommunication» centre universitaire Mila.

[6] : Pascal Roques « UML 2 ». 4eme édition 2007.

[7] : http://saoudyihab.voila.net/cours_uml/Diagramme_d_activite.pdf

[8] : www.fr.wikipedia.org

[9] : http://www.journaldunet.com/developpeur/tutoriel/out/040728-comparaison-langages1b.shtml

[10] : http://www.lopr.net/informatique.php?n=8

[11] : http://fsincere.free.fr/isn/python/cours_python.php
[12] : M. Hankerspace « Visual Basic.NET »

[13] : M. cysboy « Apprenez à programmer en Java »
[14] :M. James M « Cours Java et Eclipse »