# Chapitre 1 : Contexte du projet

## Introduction :

L'objectif de ce chapitre est de décrire en premier le contexte du projet á travers la problématique, les objectifs attendus, les exigences fonctionnelles, les exigences non fonctionnelles et les contraintes techniques.

## Contexte & problématique du projet :

Dans un air où la société est en évolution perpétuelle, la technologie suit et s’adapte à cette évolution, elles sont lié d’une façon où chaque besoin manifesté par la société est tôt ou tard satisfait par une solution de plus en plus développée.

Au Maroc les nouvelles technologies apportent un ensemble de service sous forme de solutions pour de véritables besoins, pourtant ces solutions ne sont pas tout le temps efficace en terme de contenu et utilisation. Parmi les solutions que l’évolution des nouvelles technologies ont apportées au Maroc c’est le concept de la formation à distance ou ce qu’on peut appeler le E-LEARNING. Ce service de formation à attirer l’attention de différents éléments actifs dans la société, il est devenu á la demande, convoiter et considérer comme une solution à développer ici au Maroc.

A cet effet, nous constatons que les jeunes élèves et étudiants marocains, adoptent de plus en plus les solutions e-learning et privilégie la formation à distance que la formation professionnelle traditionnelle, cependant ces solutions de e-learning sont souvent des plateformes anglophones ou francophones, parfois imitées au Maroc, ce qui engendre une offre de service quasi similaire, présentant un contenu volumineux, qui peut manquer de vision et de créativité. Ceci nous mène à poser une réflexion majeure concernant à la fois l’adaptabilité, le fonctionnement et l’usage de ces plateformes tout en s’interrogeant si cette dernière répond à l’évolutions perpétuelle du besoin.

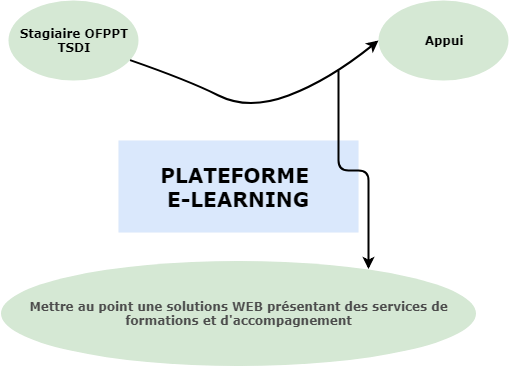
## Objectif du projet :

Pour répondre au besoin de notre cible, notre projet consiste à mettre au point une solution web **interactive** présentant des services de **formation et évaluation**, non seulement accessible à distance, mais aussi **novatrice et continuellement disponible**, à la fois **ludique et professionnelle et satisfaisant les besoins du stagiaire et exigences de l’Entreprise**.

Cette solution serra sous forme de plateforme de e-learning qui vise en premier plant les stagiaires en développement informatique au sein de l'OFPPT et qui mettra à leurs dispositions des cours, des exercices, des TP et des projets pour approfondir leurs acquis.

La figure 1 montre le diagramme bête a corne qui résume l’objectif du projet.

Figure 1 : Diagramme bête à corne



## Les acteurs :

Après la phase d’analyse, nous avons identifié trois catégories d’utilisateurs : Les apprenants, les formateurs et les administrateurs.

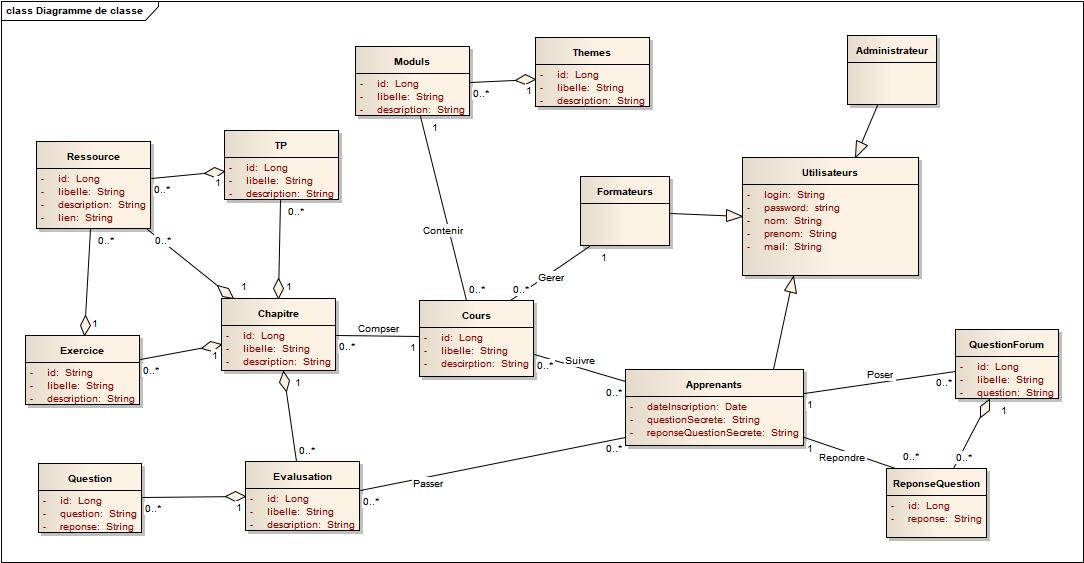
Tableau 1 : Les acteurs

|  |  |
| --- | --- |
| Acteurs | Droits |
| Les apprenants | S’inscrire et s’identifier à la plateforme |
| Consulter des cours et ajouter des cours à leurs listes de cours favoris |
| Passer des évaluations à la fin de chaque chapitre de cours pour valider la compréhension et passer au chapitre suivant. |
| Faire des exercices et TP pour approfondir leurs apprentissages |
| Faire des projets |
| Consulter le forum et poser des questions et répondre à ceux déjà poser par d’autre apprenants |
| Envoyer des messages aux Formateurs |
| Suivre des e-conférences |
| Les formateurs | S’identifier à la plateforme |
| Faire la gestion des ressources de cours, exercices, évaluations et TP. |
| Déployer des projets |
| Consulter le forum et répondre aux questions |
| Faire de e-conférences |
| Répondre aux messages des apprenants |
| Les administrateurs | Héritent de tous les droits des formateurs |
| Faire la gestion des formateurs et des apprenants |
| Modérer le forum |
| Faire la gestion des droits d’administrations |

## Les exigences fonctionnelles :

Les exigences fonctionnelles sont des exigences définissant des fonctionnalités du système à développer (ce que le système doit faire).

Dans notre cas la plateforme sera divisé en deux partie (Ecran d’utilisateur simple et écran admin) :

Tableau 2 : Exigences fonctionnelles de l’écran d’utilisateur simple

|  |  |  |
| --- | --- | --- |
| Fonction | | Description |
| S’inscrire | | Permettre aux visiteurs de s’inscrire sur la plateforme |
| S’identifier | | S’authentifier a la plateforme |
| Chercher un cours | | Permettre aux visiteurs ou/et aux inscris du site à chercher un cours selon des critères prédéfinies (nom du cours, thème, langage de programmation …) |
| Vois les cours | | Afficher tous les cours disponibles selon des prédéfinis |
| Gestion de la liste des cours suivis | **Ajouter** | Ajouter un cours à la listes des cours suivis |
| **Supprimer** | Supprimer un cours à la listes des cours suivis |
| Faire des exercices | | Faire des exercices pour valider la compréhension |
| Faire des évaluations | | Faire des évaluations à la fin de chaque chapitre |
| Faire TP | | Faire des TP pour approfondir l’apprentissage |
| Consulter le forum | | Permet de visité le forum |
| Poser des questions sur le forum | | Permet de poser des questions sur le forum |
| Répondre à des question sur le forum | | Permet de répondre aux questions déjà poser |
| Envoyer des messages | | Permet d’envoyer des messages aux formateurs |
| Suivre des e-conférences | | Suivre des conférences live proposer par la plateforme |

Tableau 3 : Exigences fonctionnelles de l’écran d’administration

|  |  |  |
| --- | --- | --- |
| Fonction | | Description |
| S’identifier | | S’authentifier a la plateforme |
| Gestion des cours | **Créer** | Créer un cours |
| **Modifier** | Modifier le contenu d’un cours |
| Gestion des évaluations | **Créer** | Créer une évaluation |
| **Modifier** | Modifier le contenu de l’évaluation |
| Gestion des Exercices | **Créer** | Créer un exercice |
| **Modifier** | Modifier le contenu d’un exercice |
| Gestion des TP | **Créer** | Créer un TP |
| **Modifier** | Modifier le contenu d’un TP |
| Gestion des projets | **Créer** | Créer un projet |
| **Modifier** | Modifier le contenu d’un projet |
| Faire des e-conférences | | Permet de diffuser en live des conférences |
| Gestion des e-conférences | | Gérer les e-conférences déjà déployer |
| Répondre aux questions | | Permet de répondre aux questions posées par les apprenants |
| Répondre aux messages | | Permet de répondre aux messages envoyer par les apprenants |
| Gestion des formateurs | **Ajouter** | Ajouter un nouveau formateur |
| **Modifier** | Modifier les données et droits d’un formateur |
| **Supprimer** | Supprimer un formateur |
| **Rechercher** | Rechercher un formateur |
| Modérer le forum | | Permet aux administrateurs de modérer les questions et réponses sur le forum, de valider ou supprimer une question sur le forum |

## Les exigences non fonctionnelles :

Les exigences non-fonctionnelles sont des exigences qui caractérisent des propriétés désirées du système contribuant à sa qualité. Cette qualité, selon le standard ISO-9000, est définie en étant « l’ensemble des caractéristiques intrinsèques qui lui confère l’aptitude à satisfaire les besoins et attentes exprimés ou implicites des clients et autres parties intéressées ». La qualité d’un système est donc concrètement un ensemble de caractéristiques de ce dernier qui contribuent à sa qualité appelées facteurs. Un facteur est généralement orienté à l’utilisateur du logiciel et est relié à un coût par l'intermédiaire des actions qu'il engendre.

Nous avons choisi les 11 facteurs du modèle de qualité logicielle McCall de 1977 (représentés dans le tableau 1). Pour chaque facteur, nous avons décrit les actions à prendre pour garantir son respect.

Tableau 1 : Actions concernant les facteurs de qualité

|  |  |  |
| --- | --- | --- |
| Facteur de qualité | description | Remarques / actions |
| Adaptabilité | La facilité d’ajout, de modification et de suppression de fonctionnalités | -Diviser le projet en petits modules a une fonctionnalité chacun. |
| Conformité | Le degré de conformité par rapport spécifications | -Définition d’un plan de test.  -différents cas de test |
| Efficacité | La minimisation de la consommation des ressources. | -Bon choix des outils de Dev  -Choix des bonne pratiques de codage qui limitent l’exploitation des ressources. |
| Maintenabilité | La facilité de localisation et correction des bugs résiduels | -A chaque version, l’ensemble de la documentation technique est conforme, exhaustive et cohérente avec la version de référence du système. |
| Maniabilité | L’utilisation agréable et facile du logiciel | -Simple à utiliser et maniable |
| Réutilisabilité | La possibilité de réutiliser des portions de code dans le cadre d’un autre projet | -Diviser le projet en petits modules.  -Utilisation des commentaires pour le code.  -Utilisation des fonctions générique pour les traitements répétitifs. |
| Confidentialité | La plateforme est protégé contre tout accès non autorisé. | -Bon choix des outils de développement (assurant un maximum de sécurité).  -Prévoir des tests de sécurité. |
| Robustesse | L’aptitude à conserver un comportement conforme aux besoins même dans des situations imprévues | -Révision et vérification des dossiers d’analyse et de conception.  -Prévoir des tests de performance. |
| Testabilité | La facilité de réalisation de tests | -Viser une grande couverture de code  -Définir le plan tests. |
| Interopérabilité | L’aptitude du logiciel à être  « couplé » à autre (échange de données, coopération) | -Bon choix d’outils de développement. |
| Portabilité | La facilité á changer d’environnement d’exécution (système d’exploitation, matériel,…) | -Bon choix d’outils de développement.  -Effectuer des tests sur divers plates-formes et environnements. |
|  | | |

# Chapitre 2 : Analyse & Conception

## Introduction :

Notre travail a nécessité une étude pour modéliser le système. Pour cela, nous avons établis des règles de gestion, puis nous avons adopté le langage UML pour symboliser nos idées sous forme de schémas.

## Règles de gestion :

Tableau 2 : Les règles de gestions

|  |  |
| --- | --- |
| Règle de gestion | Description |
| RG1 | Un thème rassemble un nombre de modules |
| RG2 | Un module contient plusieurs cours |
| RG3 | Un cours se compose de chapitres |
| RG4 | Un chapitre se compose de plusieurs exercices |
| RG5 | Un chapitre se compose de plusieurs TP |
| RG6 | Les chapitres, les exercices et les TP détiennent plusieurs ressource |
| RG7 | Un cour est créer par un formateur |
| RG8 | Un cours est suivi par un ou plusieurs apprenants |
| RG9 | Les apprenants s’inscrivent à la plateforme |
| RG10 | Les apprenants posent des questions dans le forum |
| RG11 | Les apprenants répondent a des questions dans le forum |
| RG12 | Une question dans le forum est poser par un apprenant |
| RG13 | Une question dans le forum peut avoir des réponses de plusieurs apprenants |
| RG14 | Une évaluation sert à évaluer le niveau d’apprentissage des apprenants suite à la fin de chaque chapitre. |
| RG15 | Une évaluation se compose de plusieurs questions. |
| RG16 | Une évaluation est créer par un formateur et passer par un apprenant. |
| RG17 | Les apprenants peuvent accéder aux ressources de cours. |

## Langage de modélisation :

UML (Unified Modeling Language, que l'on peut traduire par "langage de modélisation unifié") est un concept permettant de modéliser un problème de façon standard. Ce langage est né de la fusion de plusieurs méthodes existant auparavant, et est devenu désormais la référence en terme de modélisation objet, à un tel point que sa connaissance est souvent nécessaire pour obtenir un poste de développeur objet. UML se décompose en plusieurs sous-ensembles :

* **Les vues :** Les vues sont les éléments observables du système. Elles décrivent le système d'un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique, etc. En combinant toutes ces vues il est possible de définir (ou retrouver) le système complet.
* **Les diagrammes :** Les diagrammes sont des graphiques. Ceux-ci décrivent le contenu des vues, qui sont des notions abstraites. Les diagrammes peuvent faire partie de plusieurs vues.
* **Les modèles d'élément :** Les modèles d'élément sont les briques des diagrammes UML, ces modèles sont utilisés dans plusieurs types de diagramme. Exemple d’élément (cas d’utilisation, classe, association, etc.)

## Diagramme de classe :

Le **diagramme de classes** est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces d'un système ainsi que les différentes relations entre celles-ci. Ce diagramme appartient à la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques.

Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe.

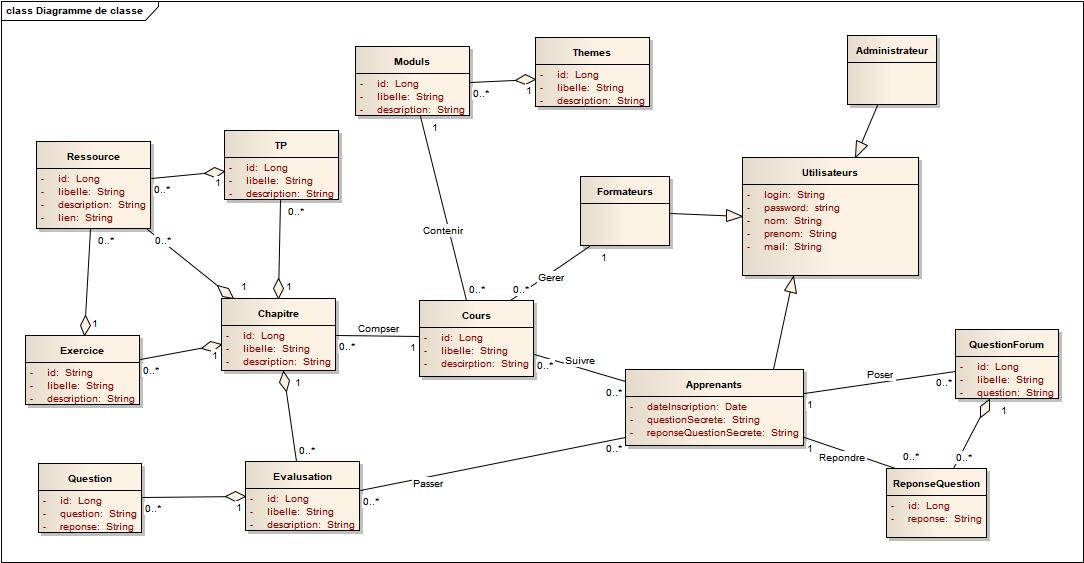
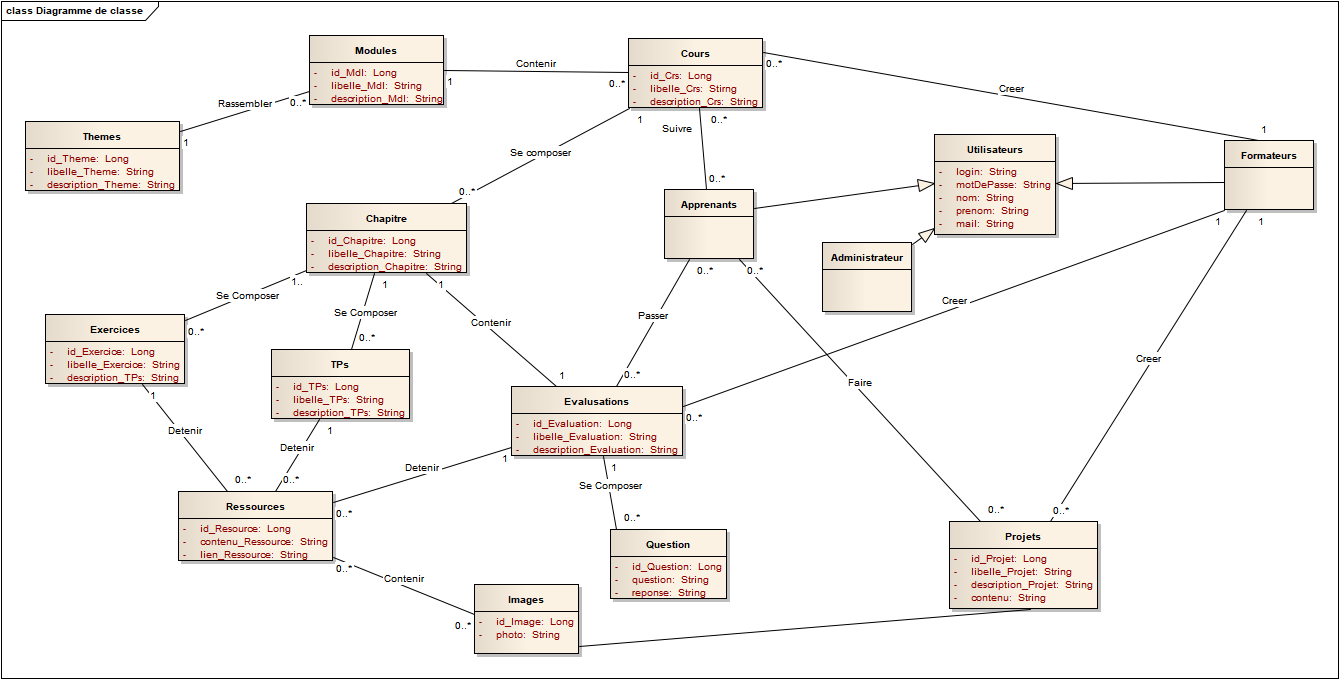
Une classe est un ensemble de fonctions et de données (attributs) qui sont liées ensembles par un champ sémantique. Les classes sont utilisées dans la programmation orientée objet. Elles permettent de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples.

Les classes peuvent être liées entre elles grâce au mécanisme d'héritage qui permet de mettre en évidence des relations de parenté. D'autres relations sont possibles entre des classes, chacune de ces relations est représentée par un arc spécifique dans le diagramme de classes.

Elles sont finalement instanciées pour créer des objets (une classe est un moule à objet : elle décrit les caractéristiques des objets, les objets contiennent leurs valeurs propres pour chacune de ces caractéristiques lorsqu'ils sont instanciés).

La figure2 montre le diagramme de classes que nous avons réalisés, en illustrant toutes les classes, relations entre ces classes et leurs cardinalités.

Figure 2 : Diagramme de classe



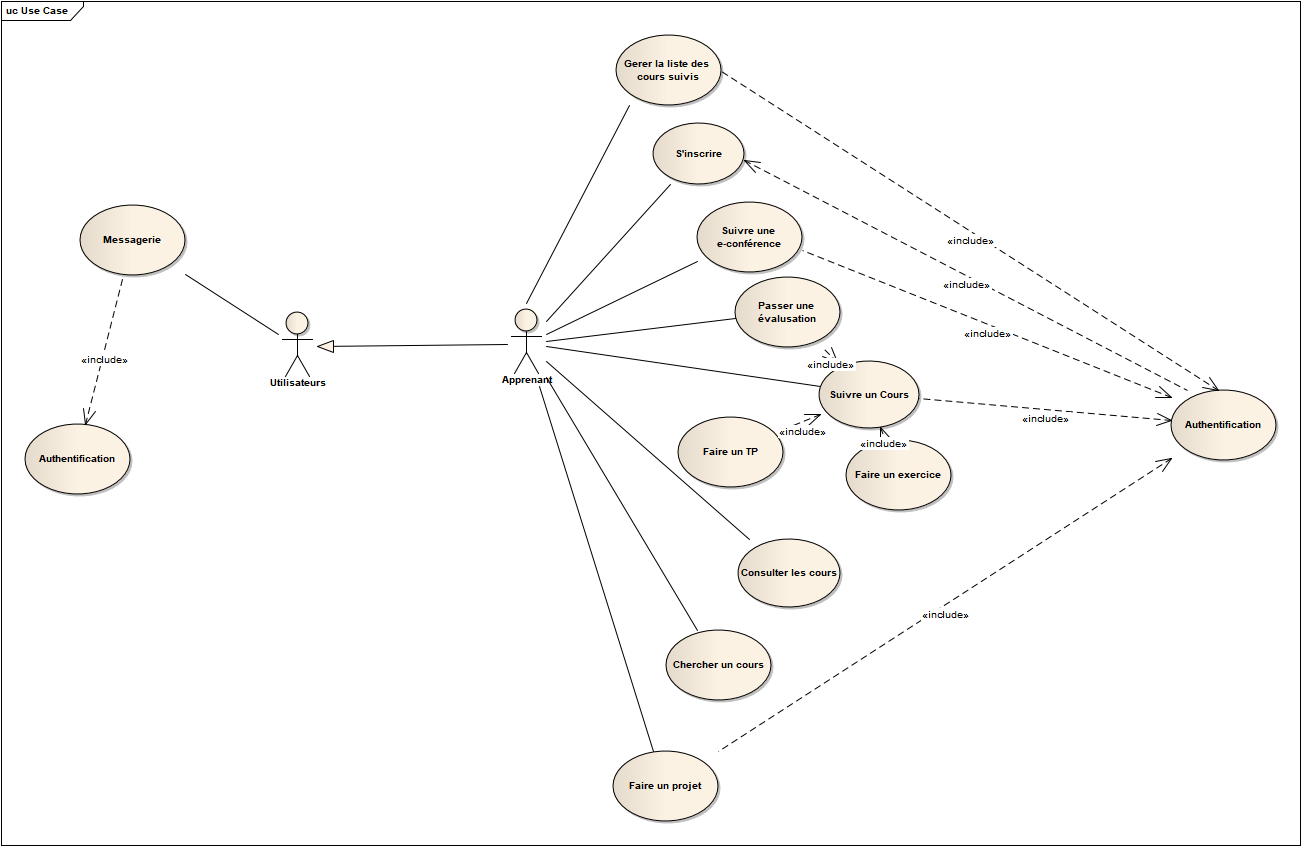
## Diagramme de cas d’utilisation :

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs (actors), ils interagissent avec les cas d'utilisation (use cases).

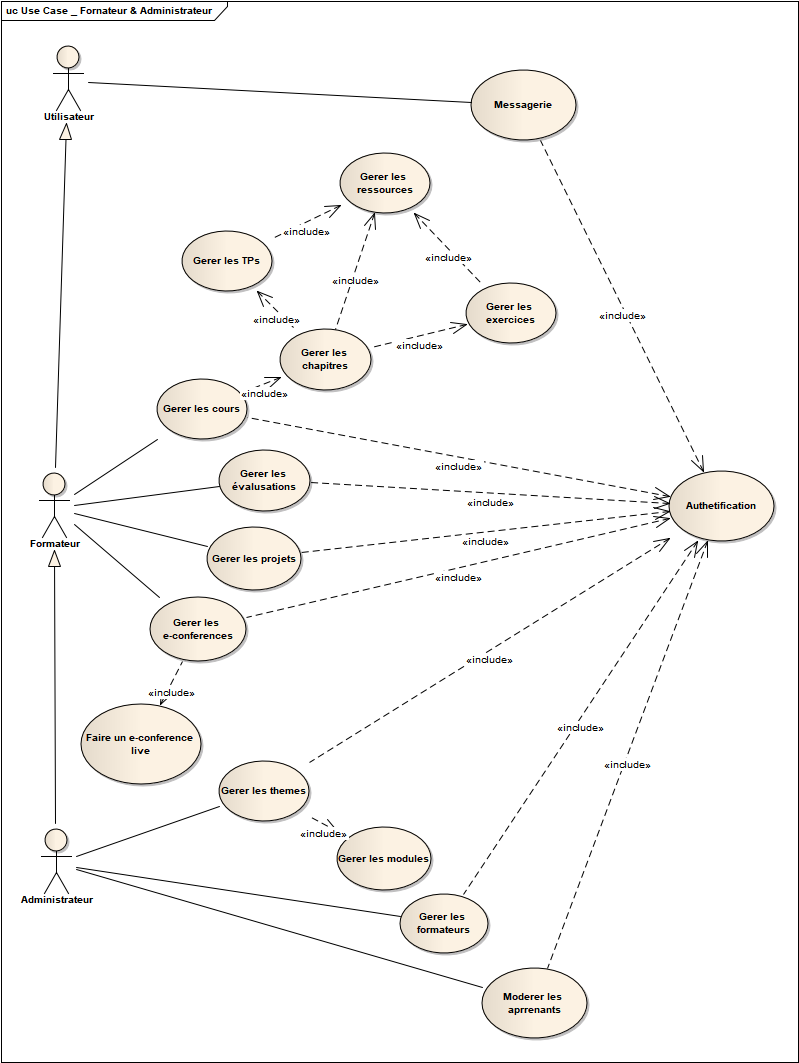
UML définit une notation graphique pour représenter les cas d'utilisation, cette notation est appelée diagramme de cas d'utilisation. UML ne définit pas de standard pour la forme écrite de ces cas d'utilisation, et en conséquence il est aisé de croire que cette notation graphique suffit à elle seule pour décrire la nature d'un cas d'utilisation. Dans les faits, une notation graphique ne peut donner qu'une vue générale simplifiée d'un cas ou d'un ensemble de cas d'utilisation. Les diagrammes de cas d'utilisation sont souvent confondus avec les cas d'utilisation. Bien que ces deux concepts soient reliés, les cas d'utilisation sont bien plus détaillés que les diagrammes de cas d'utilisation.

La figure3 décrit les différents cas d’utilisation de l’apprenant. Elle permet de distinguer les rôles.

Figure 3 : Diagramme de cas d’utilisations (Use cas) – Apprenant



La figure4 décrit les différents cas d’utilisation des formateurs et les administrateurs. Elle permet de distinguer les rôles.

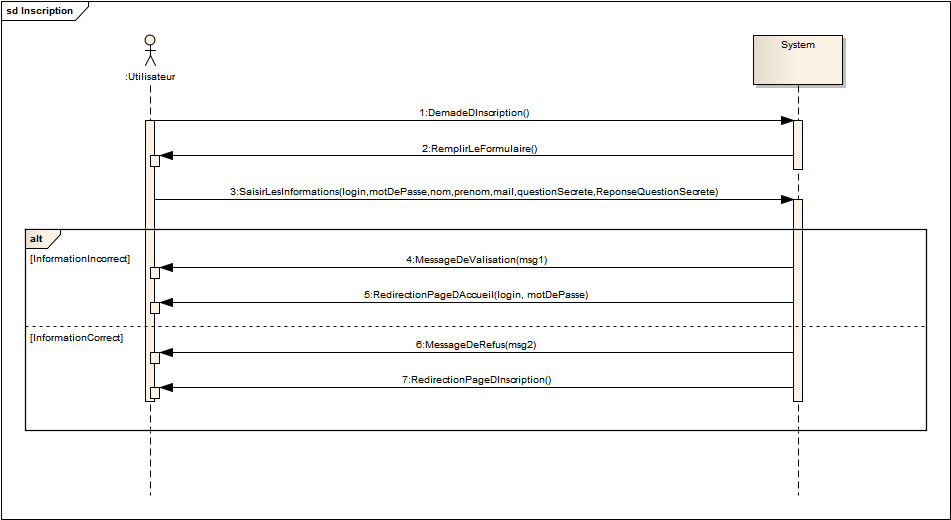
Figure 4 : Diagramme de cas d’utilisations (Use cas) – Formateur & Administrateur

## Diagramme de séquence :

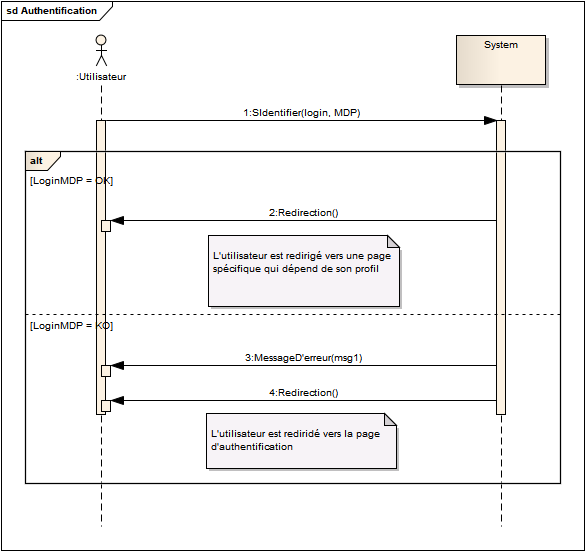
Les diagrammes de séquence sont couramment utilisés par nombre d'acteurs d'un projet. En effet, le diagramme de séquence est une représentation intuitive lorsque l'on souhaite concrétiser des interactions entre deux entités (deux sous-systèmes ou deux classes d'un futur logiciel). Ils permettent à l'architecte/designer de créer au fur et à mesure sa solution. Cette représentation intuitive est également un excellent vecteur de communication dans une équipe d'ingénierie pour discuter cette solution. Les diagrammes de séquence peuvent également servir à la problématique de test. Les traces d'exécution d'un test peuvent en effet être représentées sous cette forme et servir de comparaison avec les diagrammes de séquence réalisés lors des phases d'ingénierie. Les diagrammes de séquence tels que définis-en UML souffraient cependant d'un gros inconvénient. La quantité de diagrammes à réaliser pouvait atteindre un nombre élevé dès lors que l'on souhaitait décrire avec un peu de détail les différentes branches comportementales d'une fonctionnalité.

La figure 5 montre la séquence d’inscription, qui permet à un utilisateur de créer un compte au sein de la plateforme.

Figure 5 : Diagramme de séquence Inscription

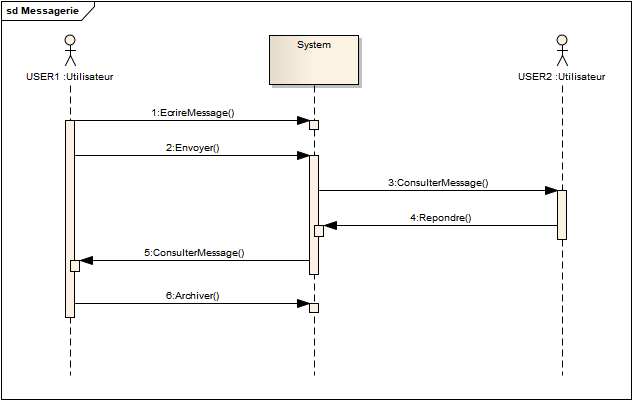


La figure 6 montre la séquence d'authentification. C'est la première séquence déclenchée dans ce projet. L'utilisateur en train de se connecter attend la réponse pour être redirigé vers la page d'accueil correspondante à son profil.

Figure 6 : Diagramme de séquence d’authentification

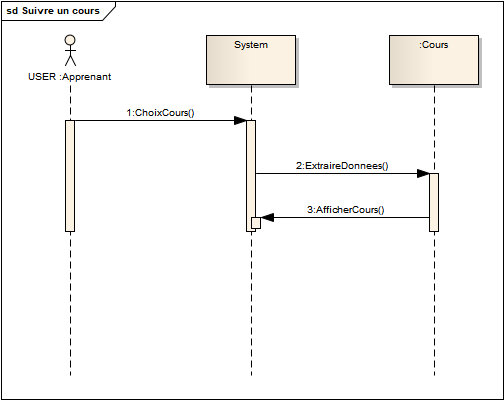
La figure 7 démontre l’interaction entre deux utilisateur échangeant des messages synchrones.

Figure 7 : Diagramme de séquence Messagerie

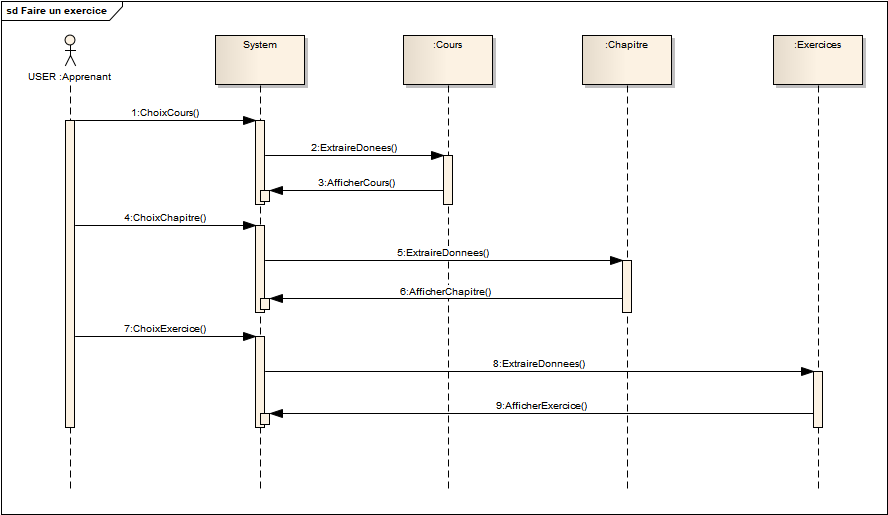


La figure 8 démontre le processus suivi par l’apprenant pour suivre un cours

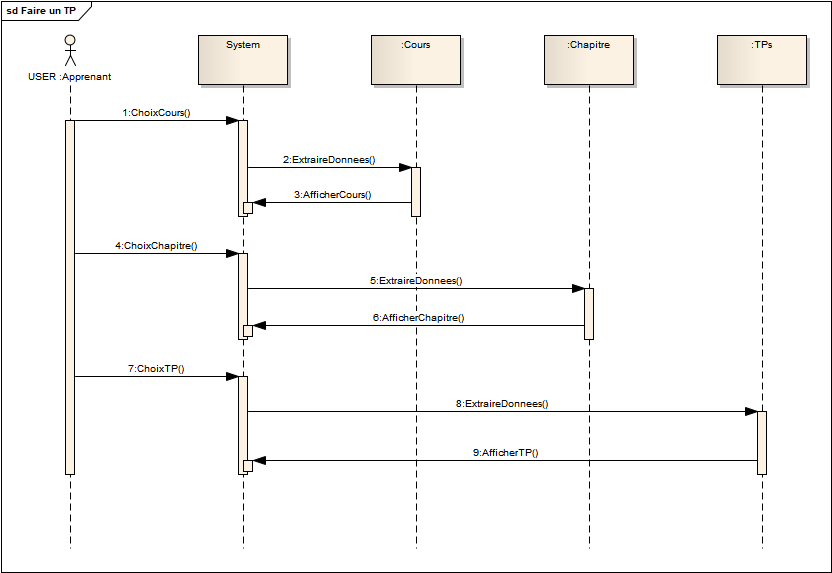
Figure 8 : Diagramme de séquence Suivre un cours



La figure 9 démontre le processus suivi par l'apprenant pour faire un exercice.

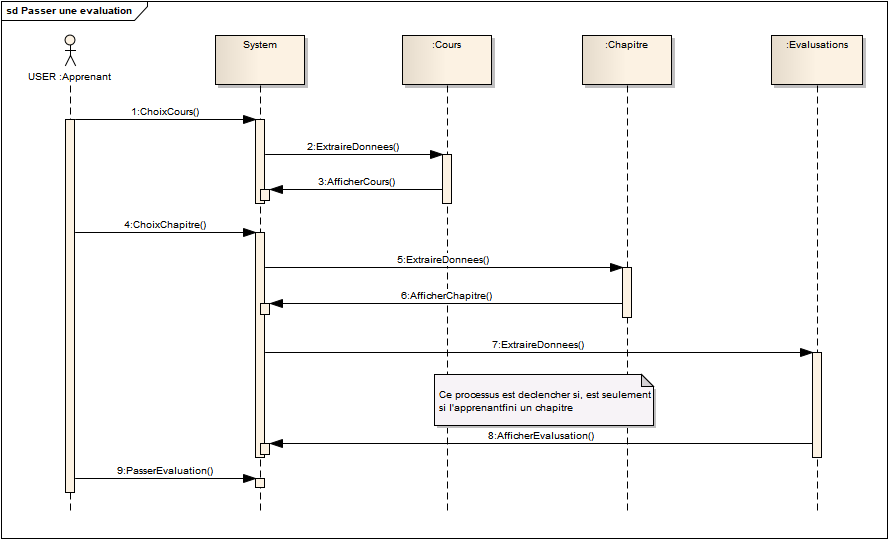
Figure 9 : Diagramme de séquence Faire un exercice

La figure 10 démontre le processus suivi par l'apprenant pour faire un exercice.

Figure 10 : Diagramme de séquence Faire un TP

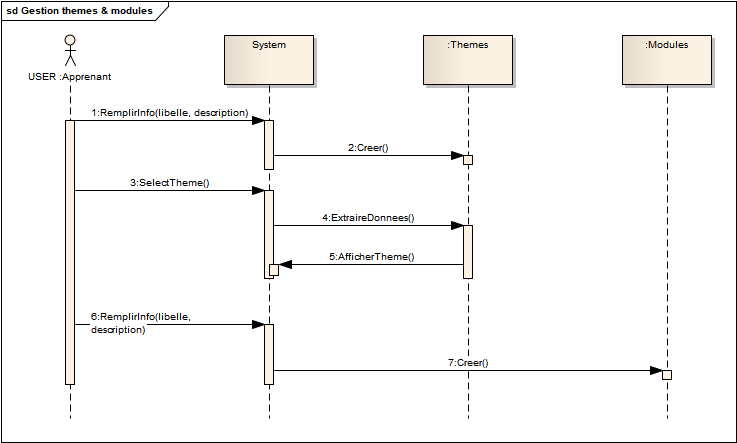
La figure 11 démontre le processus suivi par l'apprenant pour passer une évaluation.

Figure 11 : Diagramme de séquence Passer une évaluation



La figure 12 décrit la façon avec laquelle un administrateur crée des modules et des thèmes. Il doit spécifier le libelle et la description de chacun des deux.

Figure 12 : Diagramme de séquence Gestion thème module par administrateur



La figure 13 retrace le déroulement de la création des cours. Un cours contient plusieurs chapitres, un chapitre contient plusieurs exercices et TP. Ces trois deniers contiennent tous des ressources.

Figure 13 : Diagramme de séquence Faire un exercice

