Projet d’innovation informatique GI-3 :

Réalisation d’une application web de gestion d’un centre de formation à distance.

Rapport de conception

Groupe :
- ABOULBARAKET Abdelhay
- AIT ABDELMALEK Anouar

Encadré par :
- Mme ZRIKEM Maria

Table des matières

Partie I: Analyse fonctionnelle

Introduction :

Le présent chapitre présente une analyse qui joue un rôle important dans la démarche ergonomique du développement d'une application. Il a donc comme objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client qui sont divisés en besoins fonctionnels, besoins non fonctionnels et besoins techniques.

I.1. Etude préliminaire :

Le modèle de l'apprentissage dans l'histoire a toujours été assez simple. Jusqu'au début des années 2000, l'éducation se déroulait dans une classe d'élèves avec un enseignant qui dirigeait la classe. La présence physique n'était pas remise en question, et tout autre type d'apprentissage était, au mieux, discutable. Puis Internet est apparu, et le reste fait partie de l'histoire.

De nos jours, les apprenants sont habitués à l'utilisation des différents écrans et à l'utilisation d'Internet. Ainsi participer à un cours en ligne et l'animer est devenu simple.

Dans ce cadre, notre application a été conçue pour offrir aux utilisateurs un espace simple à manipuler qui va leur permettre de bénéficier d'une formation enrichissante depuis leurs machines tout en gardant la sécurité, la simplicité et la transparence.

I.2 Identification des acteurs :

Les acteurs qui interagissent avec le système sont :

- **L'administrateur :** l'administrateur est en charge du contrôle des formations et leurs intervenants.
- **L'intervenant:** Il est en charge d'ajouter du contenu aux formations qui lui sont associées.
- **L'étudiant :** Il a la possibilité de s'inscrire et suivre dans différentes formations.

I.3 Les exigences fonctionnelles :

Le système doit permettre :

- **A l'administrateur de :**

✓ Gérer les utilisateurs : Il peut gérer les comptes des autres administrateurs et des intervenants.

✓ Gérer les variables globales du système : Il peut gérer les catégories, les différentes formations offertes par l'application, leur prix et les différentes offres de réduction.

- **A l'intervenant de :**

✓ Gérer le contenu: Il peut consulter les formations qui lui sont associées, ajouter du contenu aux différents chapitres, supprimer ou modifier ce contenu.

✓ Interagir avec les étudiants : il peut répondre aux différentes remarques et questions posées par ses étudiants.

- **A l'étudiant de :**

✓ Enrôler des formations: Il peut naviguer dans l'application et choisir les formations qui l'intéressent pour qu'il puisse les poursuivre.

✓ Interagir avec le contenu: il peut poser différentes remarques et questions sur son instructeur à propos du contenu d'un chapitre de la formation.

I.4 Les diagrammes de cas d'utilisation :

- **Définition d'un cas d'utilisation :**

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). Ce diagramme permet d'identifier les possibilités d'interaction entre le système et les acteurs. Il représente toutes les fonctionnalités que le système doit fournir.

I.4.1 Diagramme des cas d'utilisation global :

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

- Une relation d'inclusion, formalisée par le mot-clé "include" : le cas d'utilisation de base en incorpore explicitement un autre, de façon obligatoire.
- Une relation d'extension, formalisée par le mot-clé "extend" : le cas d'utilisation de base en incorpore implicitement un autre, de façon optionnelle.
- Une 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.

Types de relations :

- L'association (trait plein avec ou sans flèche) 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.

La figure ci-dessous illustre le diagramme de cas d'utilisation globale, il nous permet d'obtenir une vision globale du comportement fonctionnel de notre l'application web.

Figure 1 : Diagramme de cas d'utilisation global

I.4.2 Les diagrammes des scénarios :

Les tableaux suivants représentent des exemples des cas d'utilisation, les acteurs impliqués et une description pour chaque cas :

- ✓ **Use case 1 - Authentification :**

Cas d'utilisation	S'authentifier.
Acteur	Administrateur, Intervenant ou étudiant.
Objectif	Permet à chaque utilisateur (acteur) d'accéder à son espace.
Pré conditions	- L'application est fonctionnelle. - L'utilisateur possède un compte.
Description	1- Le système invite l'acteur à saisir son login et mot de passe. 2- L'acteur saisit le login et le mot de passe. 3- Le système vérifie la validité des coordonnées. 4- Le système affiche l'espace correspondant à l'acteur.
Exception	Si un champ est manquant ou erroné, le système génère un message d'erreur.
Post condition	L'acteur accède à son espace.

Figure 2 : authentification

✓ **Use case 2 – Ajout d'une nouvelle formation:**

Cas d'utilisation	Créer une formation.
Acteur	Administrateur.
Objectif	Permet d'ajouter une nouvelle formation à l'application.
Pré conditions	- L'application est fonctionnelle. - L'administrateur possède un compte. - L'administrateur est authentifié.
Description	1- L'administrateur consulte la liste des formations. 2- L'administrateur choisit d'ajouter une nouvelle formation. 3- Le système affiche le formulaire. 4- L'administrateur remplit le formulaire. 5- Le système vérifie les informations. 6- Le système ajoute la formation à la BDD. 7- Le système affiche la liste des formations.
Exception	Si un champ est manquant, le système génère un message d'erreur.
Post condition	Une formation est ajoutée.

Figure 3 : Ajout d'une nouvelle formation

✓ **Use case 3 – Commenter un chapitre:**

Cas d'utilisation	Rédiger un commentaire.
Acteur	Intervenant ou étudiant.
Objectif	Permet à un acteur d'écrire un commentaire sur un chapitre d'une formation.
Pré conditions	- L'application est fonctionnelle. - L'acteur possède un compte. - L'acteur s'authentifie. - L'acteur est enrôlé dans la formation en question.
Description	1- L'acteur consulte sa liste de formation. 2- L'acteur choisit une formation. 3- Le système affiche la liste des chapitres. 4- L'acteur accède à un chapitre de la formation. 5- Le système affiche le contenu du chapitre et l'espace commentaires. 6- L'acteur saisit un commentaire et clique sur publier. 7- Le système ajoute le commentaire à la BDD. 8- Le système affiche le contenu du chapitre avec le nouveau commentaire dans l'espace commentaires.
Exception	Pas d'exceptions.
Post condition	Nouveau commentaire ajouté au chapitre.

Figure 4 : Commenter un chapitre

- ✓ **Use case 4 – Enrôler une formation:**

Cas d'utilisation	Enrôler une formation.
Acteur	Etudiant.
Objectif	Permet à un étudiant de s'inscrire dans une nouvelle formation.
Pré conditions	- L'application est fonctionnelle. - L'acteur possède un compte. - L'acteur s'authentifie.
Description	1- L'acteur consulte la liste des formations. 2- L'acteur choisit une formation. 3- Le système affiche le descriptif de la formation. 4- L'acteur clique sur le bouton Acheter. 5- Le système affiche le formulaire d'achat. 6- L'acteur remplit le formulaire avec ses coordonnées bancaires et un coupon s'il le possède. 7- Le système vérifie la validité des données. 8- Le système ajoute la formation à la liste des formations de l'étudiant. 9- Le système envoie un e-mail à l'étudiant. 10- Le système affiche la liste des chapitres de la formation.
Exception	Si un champ est manquant ou erroné, le système génère un message d'erreur.
Post condition	Nouvelle formation ajoutée à la liste de l'étudiant.

Figure 5 : Enrôler une formation

I.5 Exigences non fonctionnelles :

Ce sont des exigences qui ne concernent pas spécifiquement le comportement du système mais plutôt identifient des contraintes internes et externes du système. Les principaux besoins non fonctionnels de notre application se résument dans les points suivants :

- ✓ **Evolutivité :**

Le code doit être clair pour permettre des futures évolutions ou améliorations. Et d'être paramétrable le plus possible afin d'assurer une simplicité lors changements sans être obliger de modifier le code source.

- ✓ **L'ergonomie :**

L'application offre une interface conviviale et facile à utiliser.

- ✓ **Besoin de sécurité :**

L'application doit garantir à l'utilisateur connecté l'intégrité et la confidentialité de ses données. La sécurité du système est assurée par l'authentification des utilisateurs par un login et un mot de passe crypté.

L'application doit posséder une gestion de privilèges et de niveaux d'accès pour les différents types d'utilisateurs (Administrateur, intervenant, étudiant) selon leur profil.

- ✓ **Performance :**

L'application doit fournir tous les statuts et informations en temps réel et d'une manière optimale.

- ✓ **L'intégrité :**

Le système doit savoir comment traiter les échecs des interfaces, faire la capture des différentes erreurs d'entrées-sorties, effectuer le traitement des mauvaises données.

- ✓ **La traçabilité :**

Afin de pouvoir identifier un accès frauduleux ou une utilisation abusive de données personnelles, ou de déterminer l'origine d'un incident, il convient d'enregistrer certaines des actions effectuées sur les systèmes informatiques.

Conclusion :

Dans ce chapitre, Nous avons capturé les différents besoins de notre projet. Nous avons classé ces besoins en besoins fonctionnelles et non fonctionnelles. Après l'analyse des exigences et le choix des techniques à utiliser. Il est temps de passer à la conception de notre application dans le chapitre suivant.

Partie II : Conception

Introduction :

Dans cette partie, nous allons entamer la description des détails conceptuels relatifs à notre application. Nous allons identifier les diagrammes de classes et de séquences réalisés pour mettre en œuvre la solution proposée. La motivation fondamentale de la modélisation est de fournir une démarche antérieure afin de réduire la complexité du système étudié lors de la conception et d'organiser la réalisation du projet en définissant les modules et les étapes de la réalisation. Plusieurs démarches de modélisation sont utilisées. Nous adoptons dans notre travail une approche objet basée sur un outil de modélisation UML. En fait, UML (Unified Modleing Language) est un standard ouvert contrôlé par l'OMG, un consortium d'entreprises qui a été fondé pour construire des standards qui facilitent l'interopérabilité et plus spécifiquement, l'interopérabilité des systèmes orientés objet. UML est issu de l'unification de nombreux langages de modélisation graphique orientée objet. Il unifie à la fois les notations et les concepts orientés objets.

II.1 Diagramme de Classes :

II.1.1 Définition du Diagramme de classes :

Le diagramme des classes est un diagramme structurel (statique) qui permet de représenter :

- Les classes (attributs + méthodes)
- Les associations (relations) entre les classes.

Le diagramme de classes est le plus important des diagrammes UML, c'est le seul qui soit obligatoire lors de la modélisation objet d'un système. Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation.

Il s'agit d'une vue statique, car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application.

II.1.2 Notre Diagramme de classes :

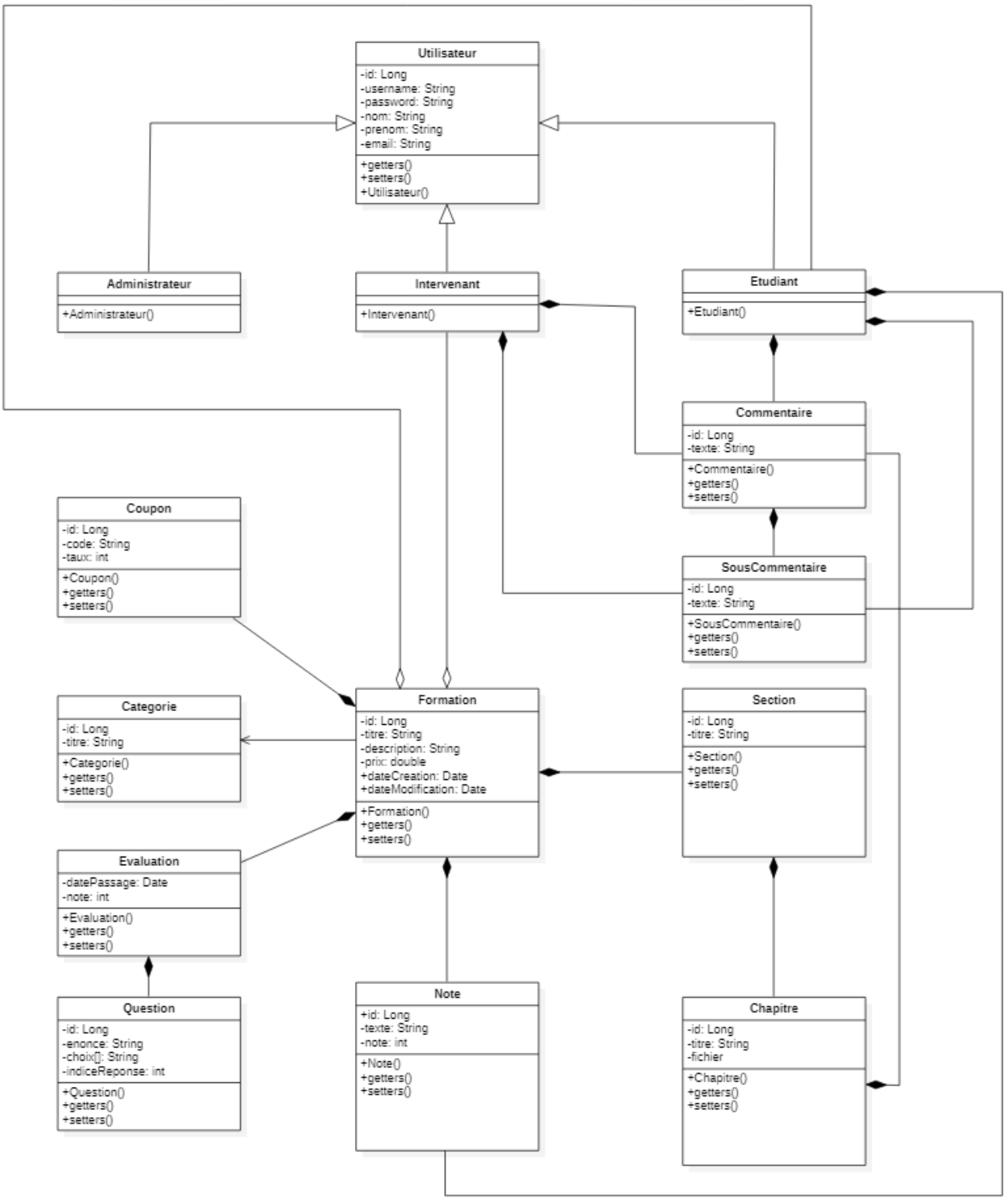

Figure 6 : Diagramme de classes

Le diagramme ci-dessus représente les différentes entités qui nous permettront de réaliser le projet en question. Commençant par les entités des acteurs (Administrateur, Intervenant, Etudiant) qui héritent toutes de la classe Utilisateur, cet héritage permet d'éviter les répétitions et simplifier la configuration de la partie sécurité et authentification.

À chaque étudiant nous avons associé une liste de formations qu'il peut noter à l'aide de la classe Note.

De même, chaque objet Intervenant est associé à une liste de formations dans laquelle il peut insérer et manipuler le contenu des chapitres.

Passant à notre objet principal : Formation. Chaque formation comme le montre le diagramme est composé d'un ensemble de d'objets Section, qui à leurs tours sont composées de plusieurs objets Chapitre.

La formation possède des Coupons que les étudiants peuvent utiliser lors de l'achat d'une formation.

Chaque formation appartient à une Catégorie spécifique.

L'objet Commentaire comme son nom l'indique permet aux étudiants et aux intervenants de laisser des commentaires sur les chapitres, et chaque Commentaire possède des Sous Commentaires pour permettre à ces acteurs d'y répondre et ne pas perdre le contexte.

À la fin de chaque formation l'étudiant passe une évaluation qui lui permettra de recevoir une certification, cette évaluation est sous forme de plusieurs objets Question, cet objet contenant une consigne, une liste de réponse possible et l'indice de la réponse correcte.

II.2 Diagramme de séquences :

Définition de diagramme de séquences :

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation UML.

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

Nous représentons donc quelques scenarios à l'aide des diagrammes de séquences.

- **Authentification** : Un utilisateur démarre l'application et le système lui renvoie le formulaire à remplir, cet acteur saisit ses identifiants. Le système vérifie les identifiants s'ils sont valides, puis il affiche la page d'accueil sinon il envoie un message d'erreur.

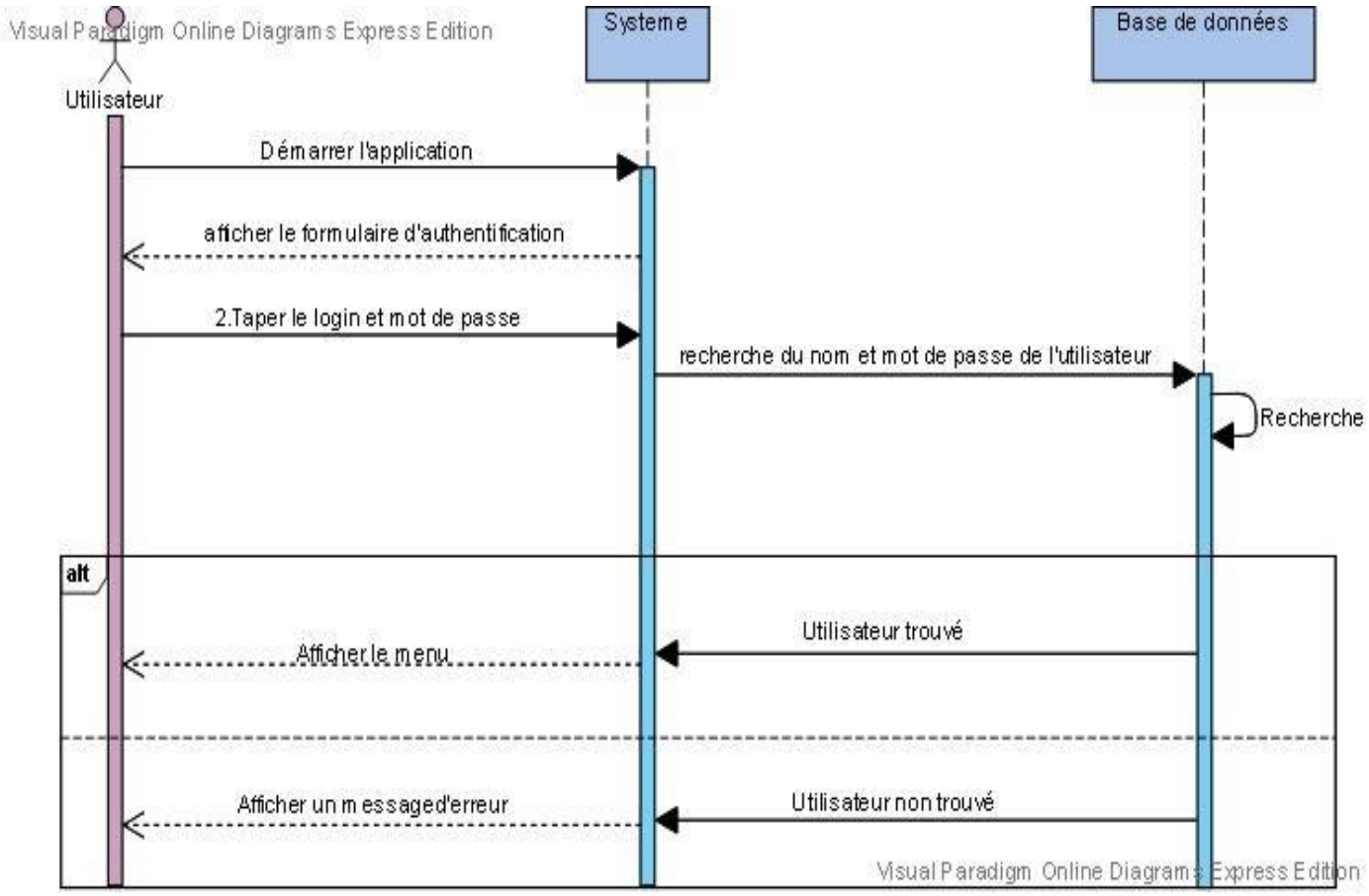

Figure 7 : Diag de séquences (Authentification)

- **Ajout de formation :** Après authentification, l'administrateur consulte la page des formations. Il demande d'ajouter une nouvelle formation et le système lui affiche un formulaire, il saisit les informations et valide la demande.
  Le système vérifie la validité des champs, s'ils sont corrects, la formation est créée, sinon affichage d'un message d'erreur.

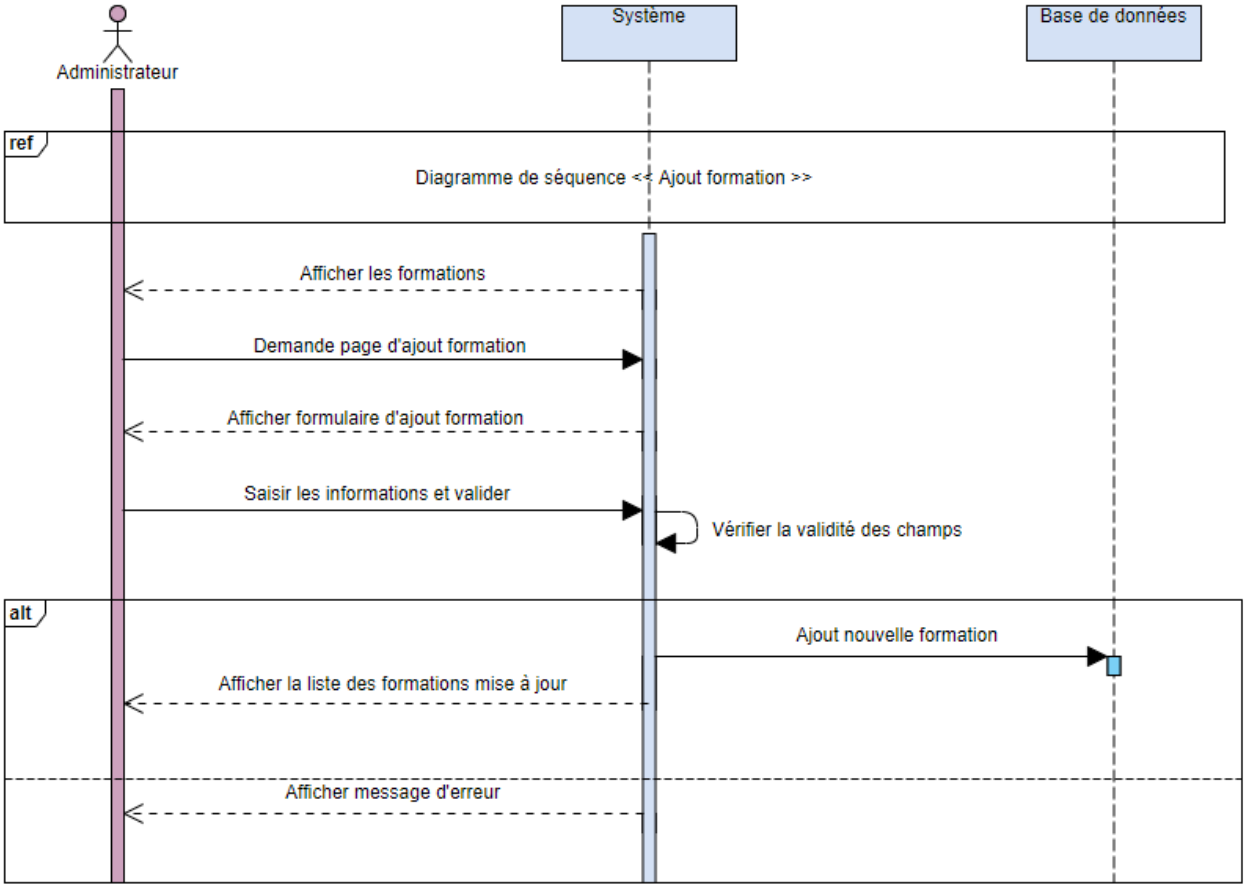

Figure 8 : Diag de séquence (Ajout formation)

- **Commenter un chapitre** : Après authentification, le système fournit à l'acteur (étudiant ou instructeur) sa liste de formations, ce dernier accède à l'une de ces formations. Après que le système lui affiche la liste des chapitres, l'utilisateur en choisit un, le système affiche le contenu du chapitre avec un espace commentaires en bas où il peut rédiger un commentaire. Une fois saisi et validé, le commentaire est ajouté par le système à la liste des commentaires du chapitre.

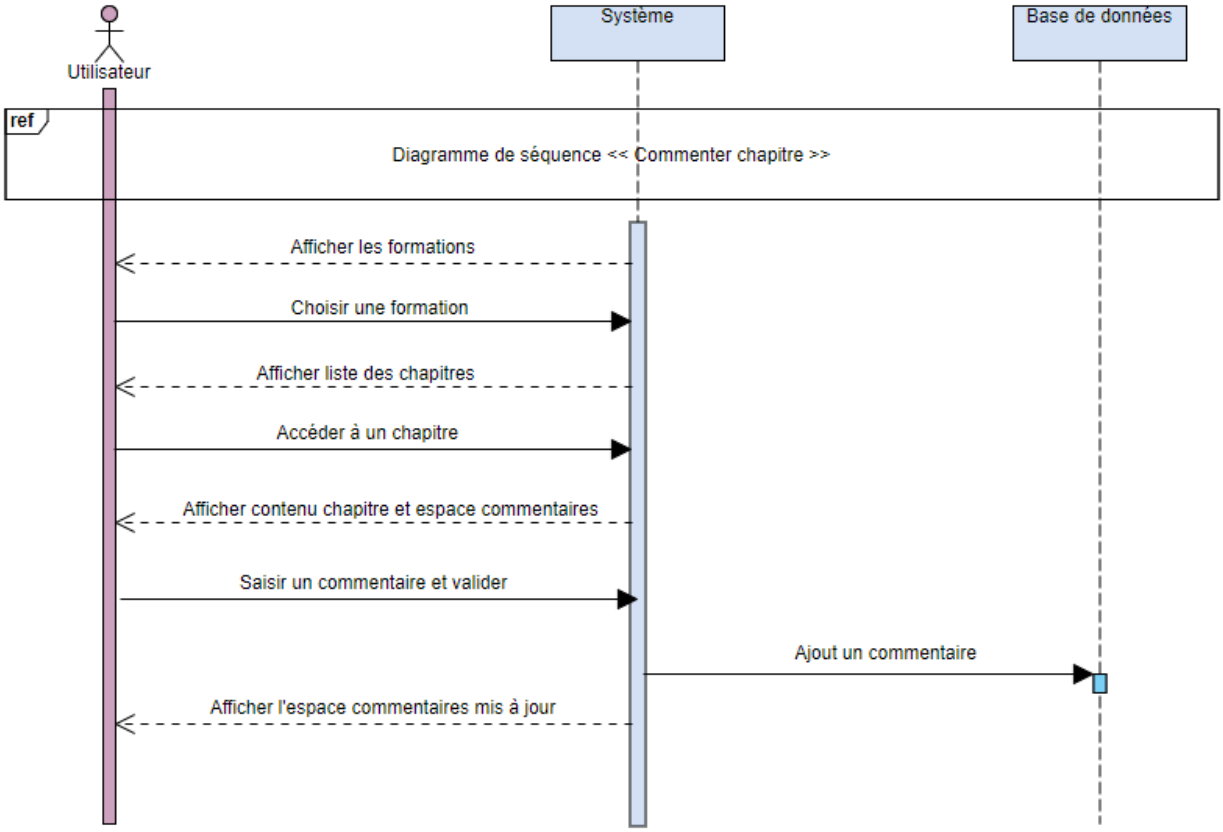

Figure 9 : Diag de séquences (Commenter un chapitre)

- **Commenter un chapitre** : Après authentification, l'étudiant accède à la liste des formations disponibles dans l'application et il choisit une formation. Le système affiche un descriptif de la formation. L'étudiant demande d'acheter la formation, le système lui affiche le formulaire pour remplir ces coordonnées bancaires. Après avoir rempli et validé, le système vérifie la validité des données, s'ils sont correctes, la formation est ajoutée à sa liste de formations et un e-mail de félicitations lui est envoyé, sinon affichage d'un message d'erreur.

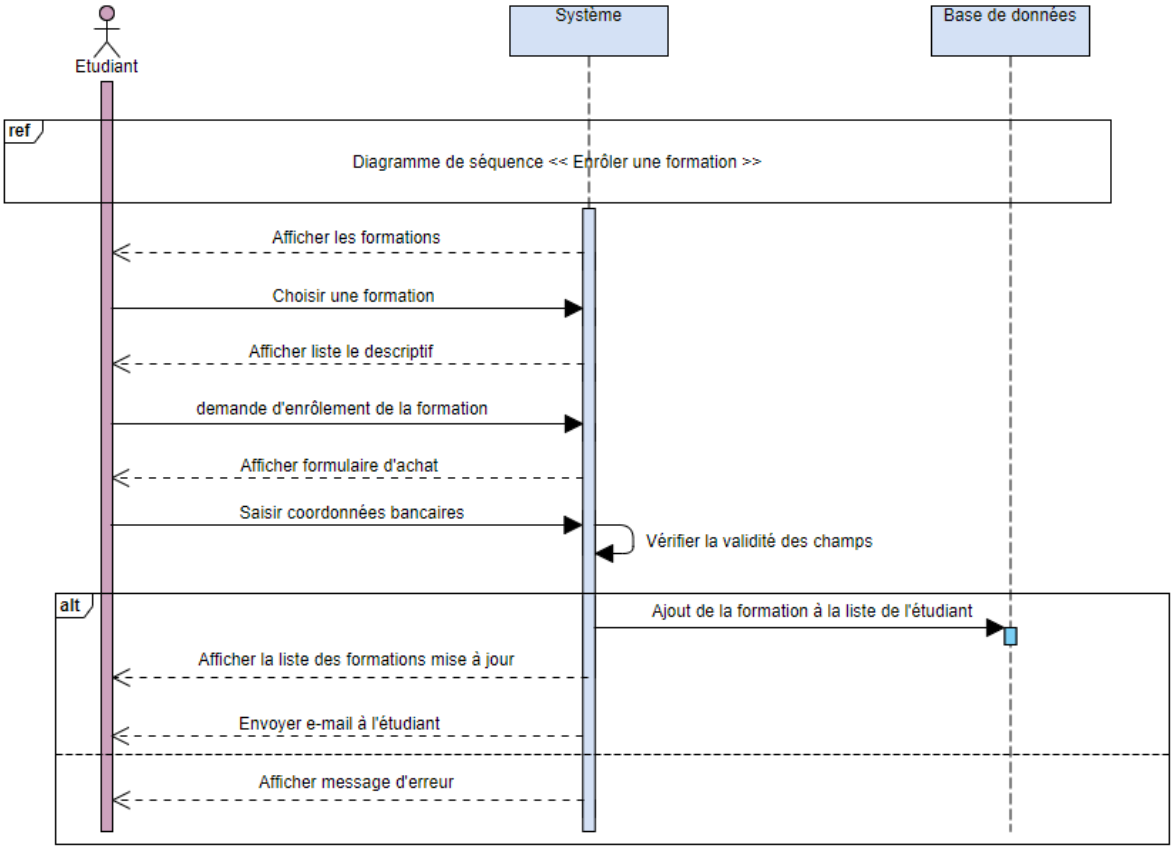

Figure 10 : Diag de séquences (Enrôler une formation)

Conclusion :

A travers ce chapitre, nous avons présenté notre conception de l'application. Nous avons fourni, dans un premier lieur, une conception globale de l'organisation de notre système. Ensuite, nous avons présenté la conception détaillée de l'application à travers les diagrammes UML. A présent, nous sommes capables d'entamer la partie réalisation.

Partie III : Etude technique

Introduction :

L'étude technique est une phase d'adaptation de conception à l'architecture technique. Elle a pour objectif de décrire au plan fonctionnel la solution à réaliser d'une manière détaillée ainsi que la description des traitements. Cette étude, qui suit l'étude détaillée, constitue le complément de spécification informatique nécessaire pour assurer la réalisation de l'application. Cette phase met aussi la lumière sur les outils de développement adoptés avec la justification de chaque utilisation d'un outil afin de mettre en œuvre ce projet.

III.1 Architecture physique et logique :

III.1.1 Architecture physique :

La conception de l'architecture physique élabore des solutions concrètes permettant d'exécuter l'architecture fonctionnelle du système.

L'architecture physique est une structure de constituants (sous-systèmes et/ou composants technologiques) et de liens physiques qui les connectent, ces éléments respectent les contraintes requises.

Dans notre cas on a une application web qui tourne dans un réseau local déployé sur un serveur http qui comporte aussi la partie donnée. Les multiples clients Web (de chacun des utilisateurs) communiquent via HTTP avec un serveur Web unique. Le navigateur Web qui fait le rendu des pages HTML participe à l'application (présentation). Les clients Web ne sont pas seulement les navigateurs des utilisateurs, mais peuvent aussi être des programmes se connectant à des APIs via HTTP. Les programmes écrits fonctionnent sur / derrière ce serveur Web. Chaque couche est plus ou moins indépendante des autres. On peut par exemple imaginer que d'autres clients que les clients Web peuvent accéder aux mêmes serveurs d'applications pour interagir avec les programmes (via des protocoles autres que HTTP). De même, les données stockées dans la base de données peuvent être manipulées par d'autres applications que ces applications Web, ou directement par des utilisateurs via les interfaces du SGBD.

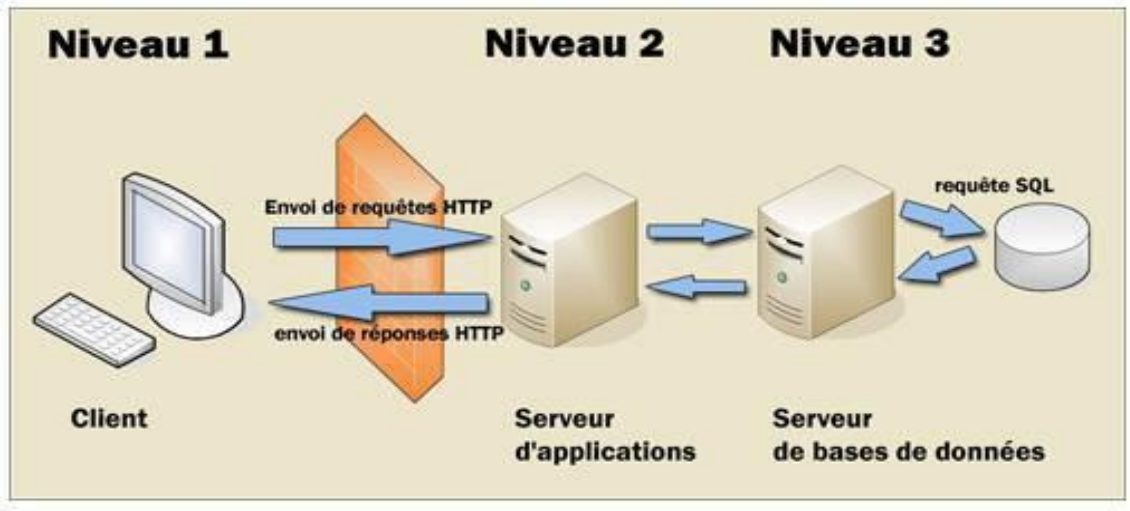

Figure 11 : Architecture physique

III.1.2 Architecture logique :

Dans la réalisation de notre Application, nous avons opté pour une architecture MVC.

Le pattern MVC permet de bien organiser son code source. Il va vous aider à savoir quels fichiers créer, mais surtout à définir leur rôle. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts, comme l'explique la description qui suit :

- **Modèle :** cette partie gère les données. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc les requêtes SQL.
- **Vue :** cette partie se concentre sur l'affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher.
- **Contrôleur :** cette partie gère la logique du code qui prend des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher la vue.

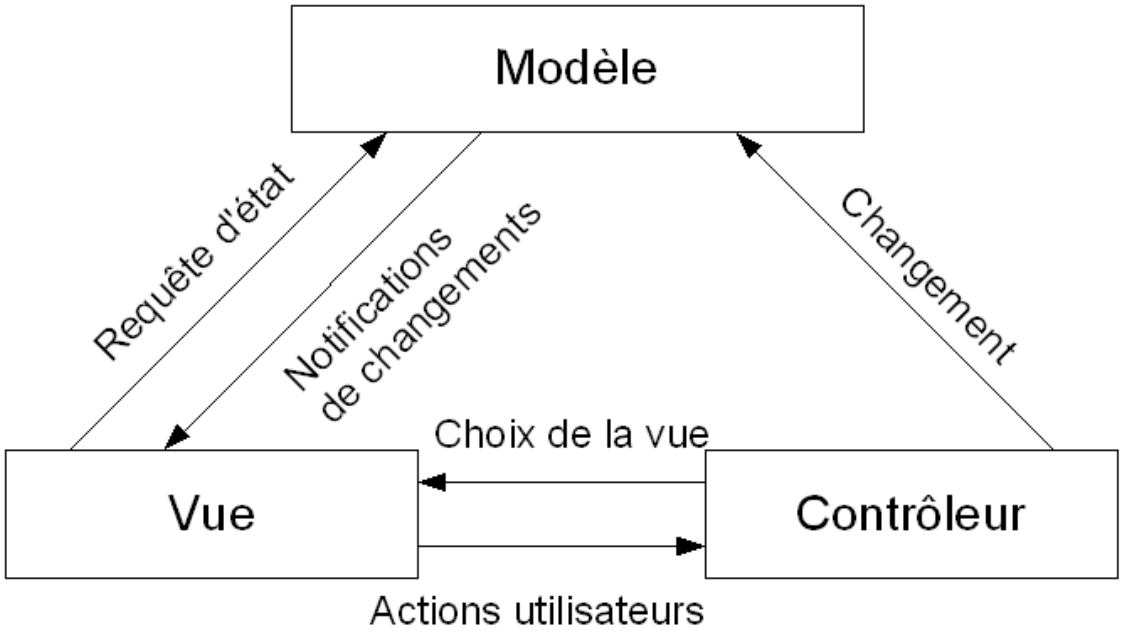

Figure 12 : MVC

III.2. Choix techniques :

Pour la réalisation de notre application, nous avons eu recours à plusieurs technologies :

A. Angular :

Figure 13 : Logo Angular

Développé par Google, Angular est un Framework open source écrit en JavaScript qui permet la création d'applications Web et plus particulièrement de ce qu'on appelle des « Single Page Applications » : des applications web accessibles via une page web unique qui permet de fluidifier l'expérience utilisateur et d'éviter les chargements de pages à chaque nouvelle action. Angular n'est pas une librairie mais bien un framework avec une approche "batteries included".

Pourquoi Angular ?

Angular fournit nativement tout le nécessaire pour produire une application entière avec une configuration standard :

- Configuration de build et d'optimisation complète.
- Module d'animations.
- ...

Les applications Angular sont donc homogènes et on tombera donc plus rarement sur des "cas particuliers".

Dans la plupart des cas, on évitera les problèmes de compatibilité de dépendances en laissant l'équipe Angular s'en charger.

B. Spring - Spring boot :

Figure 14 : Logo Spring-Spring boot

Pour la partie **backend** nous nous sommes basés sur un projet Spring Boot.

Spring boot permet de démarrer rapidement le développement d'applications et des services en fournissant les dépendances nécessaires et en auto-configurant celles-ci.

Nous avons notamment eu recours aux différents modules du framework Spring que ce soit :

- **Spring MVC :**
  Ayant une architecture de type MVC (Model-View-Controller) et ses composants servent pour développer des applications Web flexibles et faiblement couplées.

- **Spring data JPA :**
  Vise à améliorer la mise en œuvre de la couche d'accès aux données en réduisant considérablement l'effort d'écriture du code d'implémentation en particulier pour les méthodes CRUD et de recherche.

- **Spring Security :**
  Fournit une authentification et un support d'autorisation afin de sécuriser les applications Spring.

C. MySQL :

Figure 15 : Logo MySQL

MySQL est la base de données open source la plus populaire au monde. Bien qu'elle soit avant tout connue pour son utilisation par des sociétés Web, telles que Google, Facebook et Yahoo!, MySQL est également une base de données embarquée très populaire. MySQL opte pour une approche appelée base de données relationnelle.

Avec une base de données relationnelle, les données sont divisées en plusieurs zones de stockage séparées – appelées tables – plutôt que de tout regrouper dans une seule grande unité de stockage.

D. slf4j :

Figure 16 : Logo SLF4J

Une couche d'abstraction pour les API de journalisation Java, afin de garder trace des actions effectuées par chaque utilisateur.

Conclusion :

Sans une bonne architecture et de bons outils, une application est difficile à comprendre, prédire, gérer et optimiser. Cette étude technique a joué un rôle très important pour garantir un bon niveau de maintenance et de flexibilité et répondre aux caractéristiques tracées pour notre application.