Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Étude préalable

1.1 Introduction

Le but de ce premier chapitre est de mettre notre projet dans son contexte. Dans un premier lieu, nous présentons la société d'accueil. Ensuite,la problématique et la solution proposée puis nous décrivons l´étude de l'existant ainsi que la critique de l'existant et nous dressons la liste des objectifs à atteindre. Enfin, nous expliquons la méthode de développement suivie.

1.2 Présentation de l'organisme d'accueil

Olinky est une boite de développement qui conçoit et réalise des applications mobiles et web sur mesure. La société est établie en 2019 à Monastir, Tunisie.

FIGURE 1.1 – Entreprise

1.3 Contexte et proposition de projet

Ce travail est réalisé dans le cadre de la préparation du projet de fin d'études présenté à la Faculté de Sciences de Monastir en vue de l'obtention du diplôme de Master. Le Stage a été proposé et effectué au sein de la société Olinky dans le but d'atteindre un objectif final qui consiste à faire une application web et Mobile qui mise en relation entre les centres de formations, les formateurs et les apprenants.

Le sujet consiste à digitaliser les centres de formations dans laquelle les gérants de ces centres peuvent gérer leurs centres d'une manière facile et ergonomique.

Les clients peuvent s'inscrire, accéder aux formations, signer de contrat en ligne et évaluer les formations.

1.3.1 Problématique

Le monde a changé. Face à l'évolution et l'augmentation du nombre des centres des formations ainsi que la demande sur les formations elles même, la gestion et la mise en relation entre les centres et les apprenants devient de plus en plus compliquée.

Cette augmentation pose beaucoup de problèmes aux clients tels que le suivi des formations sur des différents sites.

De plus, il existe des centres de formation qui n'ont pas de sites pour gérer ces formations de manière pratique et simple et qui manquent de formateurs et de clients.

Parfois, les clients qui souhaitent une formation spécifique trouvent qu'il est difficile de rechercher et de comparer entre les formations des différents sites, ce qui mène a perdu assez de temps et des efforts.

Souvent les formateurs trouvent des difficultés de trouver un emploi convenable et ils postulent et recherchent sans aucun résultat.

Actuellement, pour organiser une formation, le centre doit assurer un nombre minimum des participants et chercher un enseignant pour assurer une telle ou telle formation Pour sortir de ce planning traditionnel, nous avons pensé d'une solution de digitalisation de ce processus afin de rendre la gestion, la mise en relation, le déroulement de la formation simples, accélères et surtout numérises.

1.3.2 Solution proposée

Le sujet consiste à digitaliser les centres de formations dans lequel les gérants de ces centres peuvent gérer leurs centres d'une manière facile et ergonomique.

Notre solution consiste à regrouper plusieurs centres de formations ce qui facilite l'accès à ces centres et ces formations

Ensuite, cette solution permettra aux différents centres de gagner du temps et de l'argent dépense dans la création de sites web en s'inscrivant sur notre plateforme.

Comme notre site regroupe beaucoup de centres de formation, ce qui donne la possibilité à d'autres centres qui sont débutants ou moins connus d'être connus et visités par nos clients.

Cette plateforme met à disposition des centres de formations des formateurs à travers une inscription dans le site puis un contrat de formation et aux formateurs des offres d'emploi.

les formations proposées par un centre peuvent être en ligne et/ou sur place.

1.4 Étude de l'existant

Malgré tous les efforts déployés par les centres de formations pour rendre l'éducation accessible à tous sauf que le développement et l'augmentation du nombre de sites de centres de formations ont posé un gros problème, car la majorité des sites ne gèrent que leurs propres centres. donc pour trouver un site qui gère vos centres et regroupe plusieurs autre centres de formations et qui offre d'emploi pour les formateurs et de formateurs pour les centres est difficile et peut prendre du temps. La bonne stratégie consiste donc à réunir les centres de formation, les formateurs et les clients cibles dans un même site.

parmi les sites de centres de formations on peut citer quelques exemples :

• acurschool logiciel gestion scolaire tunisie

• AdmiSco (acronyme de Administration Scolaire) est un logiciel de gestion scolaire disponible en trois langues : Arabe, Français et Anglais.

1.4.1 Critique de l'existant

Comme montre le tableau 1.1, les solutions existantes de plateforme web de centres de formations sur le marché proposent différents fonctionnalités de base. Dans notre travail nous ne cherchons pas à réaliser une solution mieux que les solutions existantes dues à la grande capacité du travail nécessaire pour faire cela en termes de connaissances et du temps. Lors de l'étude que nous avons faite dans la section précédente, nous avons relevé les problèmes suivants :

Système existant	Inconvénients

TABLE 1.1 – Comparaison des solutions existantes

1.4.2 Objectifs du projet

Cette plateforme aura les objectifs suivants :

- Créer des comptes centre de formations, formateur et candidat.

Le centre doit avoir la possibilité de :

-gérer les formations, les salles, les formateurs, les candidats, les contrats de formation, les diplômes et finalement son compte.

Le candidat doit être capable de :

-Gérer ses abonnements

-Gérer son compte

-Chercher un centre de formation (Ville, code postal, type de formation, autour de lui, etc ...)

-Évaluer une formation

-Signer un contrat de formation en ligne

Le formateur doit pouvoir :

- Gérer ses (formations (études effectuées), expériences)
- Gérer les notes des candidats
- Signer un contrat de formation en ligne

1.5 Méthodologie de travail

Pour assurer un bon rendement de développement en termes de qualité et de productivité le choix de la méthodologie en informatique est primordial. Compte tenu de la complexité des systèmes d'aujourd'hui, nous devrions essayer d'aborder cette complexité en présentant une approche à suivre avec des étapes bien précises. C'est le principe des méthodologies de travail. Une méthode d'analyse ou de conception est un procédé qui a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin ;de rendre ce travail plus fidèle au besoin du client. Pour ce faire nous partons d'un énoncé informel (le besoin tel qu'il est exprimé par le client complété par la recherche d'information auprès des experts du domaine fonctionnel, comme par exemple les futurs utilisateurs de ce logiciel), ainsi que l'analyse de l'existant éventuel (c'est à dire la manière dont les processus à traiter par le système se déroulent actuellement chez d'autre client).

La phase d'analyse permet de lister les résultats attendus, en termes de fonctionnalités, de robustesse, de maintenance de sécurité, d'extensibilités, etc. La phase d'analyse permet de décrire de manière ambiguë, le plus souvent en utilisant un langage de modélisation, le fonctionnement futur du système, afin d'en faciliter la réalisation. Aujourd'hui le standard

Industriel de modélisation objet est UML (Unified Modeling Langage). UML se définit comme un langage de modélisation graphique et textuelle. Notre choix s'est orienté vers le processus unifié (PU ou UP en anglais pour Unified Process) comme méthode de développement.

1.5.1 Comparaison

Le tableau ci-dessous contient un comparatif entre les principales méthodologies de développement que nous avons choisi vu la diversité de ces méthodes.

Méthodologies	Description	Points forts	Points faibles
Cascade	les phases sont déroulées d'une manière séquentielle	Distingue clairement les phases du projet	Non itérative, Pas de modèles pour les documents
RUP(Rational Unified Process)	Promu par Rational ; Le RUP est à la fois une méthodologie et un outil prêt à l'emploi ; Cible des projets de plus de 10 personnes	Itératif ; - Spécifie le dialogue entre les différents intervenants du projet ; - Propose des modèles de documents pour des projets types	- Assez flou dans sa mise en œuvre ; - Ne couvre pas les phases en amont et en aval au développement.
2TUP (Two Truck Unified Process)	- S'articule autour de l'architecture ; -Propose un cycle de développement en Y ; - Cible des projets de toutes tailles.	- Itératif ; - Laisse une large place à la technologie et à la gestion des risques ; - Définit les profils des intervenants, les plannings, les prototypes.	- Plutôt superficiel sur les phases situées en amont et en aval du développement ; - Ne propose pas de documents types.
XP (eXtreme Programming)	-Ensemble des bonnes pratiques de développement ; - Cible : Moins de 10 personnes.	- Itératif ; - Donne une importance aux aspects techniques ; - Innovant : programmation en duo.	- Assez flou dans sa mise en œuvre ; - Ne couvre pas les phases en amont et en aval au développement.
SCRUM	- Se base sur des itérations dites sprints de développement.	- Donne toute confiance aux développeurs et les laisser faire leur travail ; - Chaque itération a un objectif bien précis et fournit une Fonctionnalité testée.	- La mise en œuvre du développement n'est pas précisée ; - Le développement rapide et répétitif se traduit par une forte Pression sur l'ensemble des membres de l'équipe de développement.

TABLE 1.2 – Tableau comparative des différentes technologies de travail

L'étude comparative réalisé sur les trois principaux processus de développement Rup, Xp, 2TUP résumé dans le tableau nous permet de tirer les conclusions suivantes :

- Le processus RUP néglige les contraintes techniques qui sont indispensables dans notre projet, nous avons par conséquent choisie de l'écarter ; - Le processus XP néglige la phase de capture de besoins fonctionnels et techniques et la phase de conception et donne une grande importance à la phase de développement, il est par conséquent écarté ; - Le processus 2TUP permet en particulier de séparer les contraintes fonctionnelles des contraintes techniques érigées sous forme de deux branches permettant de les exploiter parallèlement ; - CASCADE est un processus séquentiel et non itératif ; - Scrum quant à lui subdivise les différentes phases

du projet en sprint qui en théorie ne dépasse pas 30 jours.

1.5.2 Choix de la méthodologie de travail

Suite à l'étude et à la comparaison des principaux processus de développement et afin de contrôler les risques et de mener à bon terme notre projet vu sa complexité, nous avons opté pour le processus 2TUP pour plusieurs raisons, **2TUP** (Two Tracks Unified Process) propose un cycle de development en Y et qui dissocie et parallélise la résolution des questions fonctionnelles et techniques. Le cycle de vie 2TUP s'apparente à un cycle en cascade, mais introduit une forme itérative interne à certaines taches.

1.5.2.1 Mise en Oeuvre du processus 2TUP

Après une étude qui concerne la méthodologie de développement 2TUP, nous pensons que cette méthodologie est la plus adaptable à notre projet vu que son approche est nouvelle et originale. En effet, notre projet est basé sur un processus de développement bien défini qui va de la détermination des besoins fonctionnels attendus du système jusqu'à la conception et le codage final.

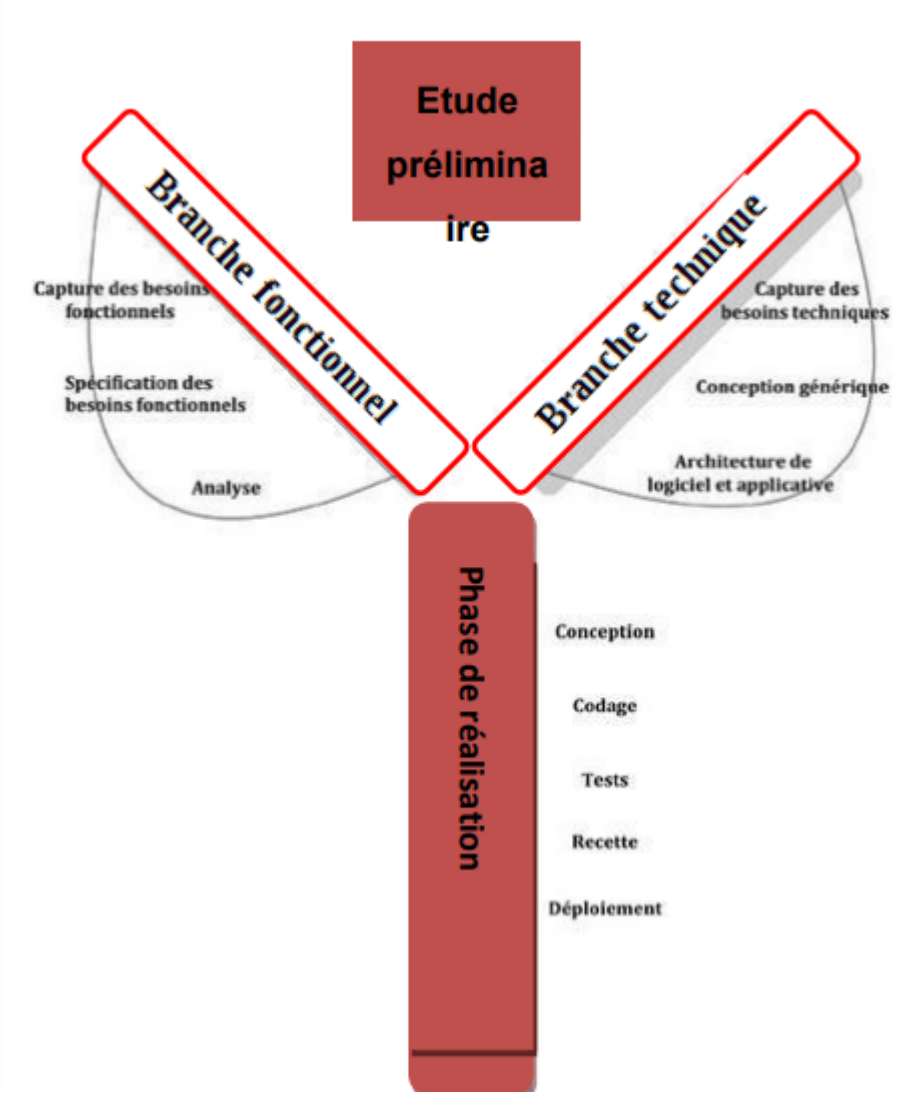

FIGURE 1.2 – La méthode 2TUP

Ce processus commence par une étude préliminaire qui consiste essentiellement à identifier les acteurs qui vont interagir avec le système à construite, les messages qu'échangent les acteurs et système, à produire le cahier des charges et à modéliser le contexte. Le processus s'articule ensuite autour de trois phases essentielles :

- **Branche fonctionnelle :** Les principales étapes de la branche fonctionnelle se présentent comme suit :
  - L'étape capture des besoins fonctionnels :Cette phase a pour objectif de définir :
    - La frontière fonctionnelle entre le système et son environnement.
    - Les activités attendues des différents utilisateurs par rapport au système.
  - L'étape d'analyse : consiste à étudier précisément les spécifications fonctionnelles de manière à obtenir une idée de ce que va réaliser le système en terme de métier.
- **Branche technique :** Les principales étapes de la branche technique se présentent comme suit :
  - L'étape capture des besoins techniques : Cette étape recense toutes les contraintes sur les choix de technologies pour la conception du système. Les outils et le matériel

sélectionnés ainsi que la prise en compte des contraintes d'intégration avec l'existant (pré requis d'architecture technique).

— L'étape conception générique : Définit les composants nécessaires à la construction de l'architecture technique. Cette conception est complètement indépendante des aspects fonctionnels. Elle permet de générer le modèle de conception technique qui définit les Framework.

- **Phase conception – réalisation :** Les principales étapes de cette branche se présentent comme suit :

— L'étape conception préliminaire : Cette étape permet de produire le modèle de conception système. Ce dernier organise le système en composants, délivrant les services techniques et fonctionnels, Ce qui induit le regroupement des informations des branches technique et fonctionnelle.

— L'étape de codage : permet d'effectuer la production des composants et les tests des unités de code au fur et à mesure de leur réalisation.

— L'étape de recette : consiste à valider les fonctionnalités du système développé.

1.6 Conclusion

Au niveau de ce chapitre, nous avons essayé de donner une présentation générale sur le cadre général de projet.

Dans le chapitre suivant, nous entamons l'activité d'expression des besoins. Et nous présentons le diagramme cas d'utilisation global et les principaux cas d'utilisation détaillés de l'application.

Chapitre 2

Analyse et spécification des besoins

2.1 Introduction

L'activité d'expression des besoins permet de définir les besoins qui doivent être satisfaits par le système pour ses utilisateurs. Dans ce chapitre, d'abord nous allons présenter les différents besoins fonctionnels, non fonctionnels et semi-formelles de l'application, Ensuite, nous identifions les différents acteurs impliqués dans notre application et leurs rôles. Et enfin, La présentation du diagramme de cas d'utilisation globale, diagramme de cas d'utilisation détaillé, selon le regroupement des cas sémantiques classés et les cas d'utilisation dans un tableau, selon leurs priorités.

2.2 Spécification des besoins

L'expression des besoins a pour objectif d'assurer une compréhension des besoins, des clients et de leurs exigences. Cela sera fait au niveau de l'application.

2.2.1 Spécification des besoins fonctionnels

Nous avons réalisé une application web et Mobile de centres de formations pour lequel trois acteurs agissent principalement : les centres de formation, les candidats et les formateurs. les services proposés par notre application peuvent être résumés en :

- Gestion d'un espace centre de formations
- Gestion d'un espace candidat
- Gestion d'un espace formateur
- Gestion d'un espace Administrateur
- Gestion de formations
- Gestion des candidats
- Gestion des abonnements
- Gestion des certificats
- Gestion des salles

— Gestion des formateurs
— Gestion d'évaluation de formation
— Gestion de contrat
— Gestion de notes des candidats

2.2.2 Spécifications des besoins non fonctionnels

La prise en compte des besoins non fonctionnels permet d'améliorer la qualité des services fournis. Parmi ces derniers on peut citer :

-La sécurité Notre application doit garantir l'intégrité des données.

-La confidentialité en assurant la validité de l'identité de l'utilisateur. Cela peut être fait, entre autres, via un mot de passe qui garantit le contrôle d'accès.

-Temps de réponse Notre application doit garantir un temps de réponse minimale.

-La convivialité les interfaces utilisateurs doivent être conviviales c'est-à-dire simple ergonomiques et adaptées à l'utilisateur.

-Fiabilité Le système doit être disponible à la demande par l'utilisateur, ainsi qu'assurer une gestion exhaustive des erreurs.

2.3 Spécifications semi-formelle

La spécification semi formelle permet de dégager les différents acteurs qui peuvent agir avec l'application et donne une représentation fonctionnelle des différents acteurs et la fonctionnalité du système sous la forme d'un diagramme de cas d'utilisation.

2.3.1 Identification des Acteurs

Un acteur représente un rôle joué par une personne qui interagit avec le système. Par définition, les acteurs sont à l'extérieur du système. Ils se recrutent parmi les utilisateurs du système et aussi parmi les responsables de sa configuration et de sa maintenance.

Dans le cas de notre projet nous présentons les acteurs suivants :

2.3.1.1 Administrateur

C'est l'administrateur de l'application comme son nom l'indique, Il avait tout l'accès, son rôle est de gérer toutes les fonctions de l'application et gérer la base de données. Il peut administrer la liste des centres ainsi que tous les comptes des utilisateurs.

2.3.1.2 Administrateur centre de formation

Représente l'utilisateur principale du système. Il inscrit puis gérer ce centre de formations.

2.3.1.3 Candidat

Il s'agit des utilisateurs chercheurs de formations qui doivent s'inscrire sur le portail.

2.3.1.4 Formateur

Ce sont les utilisateurs qui cherchent du travail et doivent s'inscrire sur le portail. Assure de cours soit en ligne ou sur place.

2.3.2 Formalisme de modélisation

Face à la diversité des formalismes utilisés par les méthodes d'analyse et de conception objet, UML (Unified Modified Language « langage de modélisation objet unifié ») représente un réel facteur de progrès par l'effort de normalisation. En effet, UML est issu de la fusion de trois méthodes qui ont le plus influencé la modélisation objet au milieu des années 90 : Booch Grady Booch, OMT (Object Modelling Technique) de James Rumbaugh et OOSE (Object Oriented Software Engineering) d'Ivar Jacobson. UML est à présent un standard défini par l'OMG (Object Management Group).
L'UML est un langage graphique de modélisation des données et des traitements, fondé sur des concepts orientés objets. La notation UML est un langage visuel constitué d'un ensemble de schémas, appelés des diagrammes, qui donnent chacun une vision différente du projet à traiter. UML nous fournit donc des diagrammes pour représenter le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc.
UML 2.0 propose de décrire un système à l'aide de 13 diagrammes repartis en deux grands groupes, on cite parmi ces diagrammes et notamment ceux que dans notre travail ont été amenés à effectuer :

- **Diagrammes structurels :**
  - Diagrammes de classes :
    Ils contiennent un ensemble de classes ainsi que leurs relations.
    L'intérêt majeur de ce diagramme est de représenter les entités du système d'information.
  - Diagrammes de paquetages :
    Ils sont utilisés pour divisé un modèle en terme de conteneurs logiques ou packages, ils permettent aussi de décrire les interactions entre packages.
- **Diagrammes comportementaux :**
  - Diagrammes de cas d'utilisation :
    Ils sont utilisés pour modéliser les interactions entre l'utilisateur et le système. Ils définissent le comportement et les besoins sollicités.
  - Diagrammes de séquence :

Ils montrent des interactions entre objet selon un point de vue temporel. Généralement ces diagrammes définissent les objets acteurs et le système.

— Diagramme d'activité :
Il permet de décrire sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants.

— Diagramme d'états-transitions :
Décrit le comportement des objets d'une classe au moyen d'un automate d'états associés à la classe.

2.4 Diagrammes de cas d'utilisation

Les diagrammes de cas d'utilisation (DCU) sont des diagrammes UML utilisés pour une représentation du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés.

2.4.1 Diagramme de cas d'utilisation global

Dans cette partie, on va préciser les fonctionnalités dans un diagramme global de cas d'utilisation.

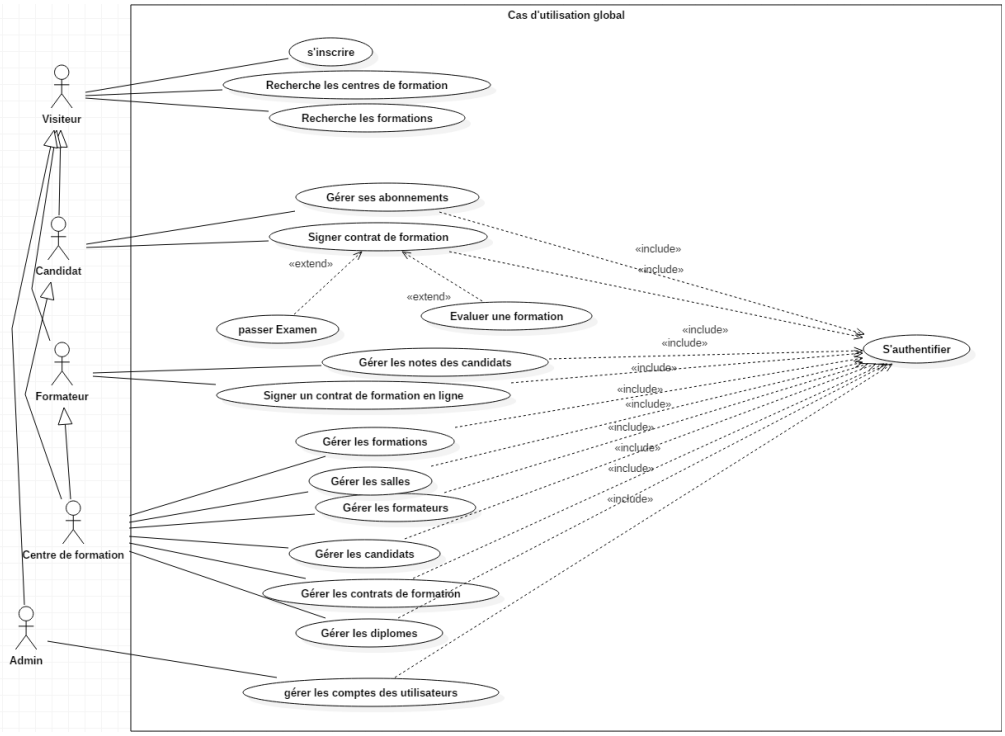

FIGURE 2.1 – Diagramme de cas d'utilisation global

Les acteurs sont : candidat,centre de formations (administrateur centre de formations),formateur et admin.
Les cas d'utilisation se changent d'un utilisateur à un autre suivant son rôle dans l'application.Dans ce qui suit, nous allons nous limiter à la description détaillée des cas d'utilisation les plus importants.

2.4.2 Diagrammes de cas d'utilisation détaillé

Dans cette partie, on va préciser les différentes fonctionnalités pour chaque acteur impliqué dans le système

2.4.2.1 Diagramme de cas d'utilisation authentification

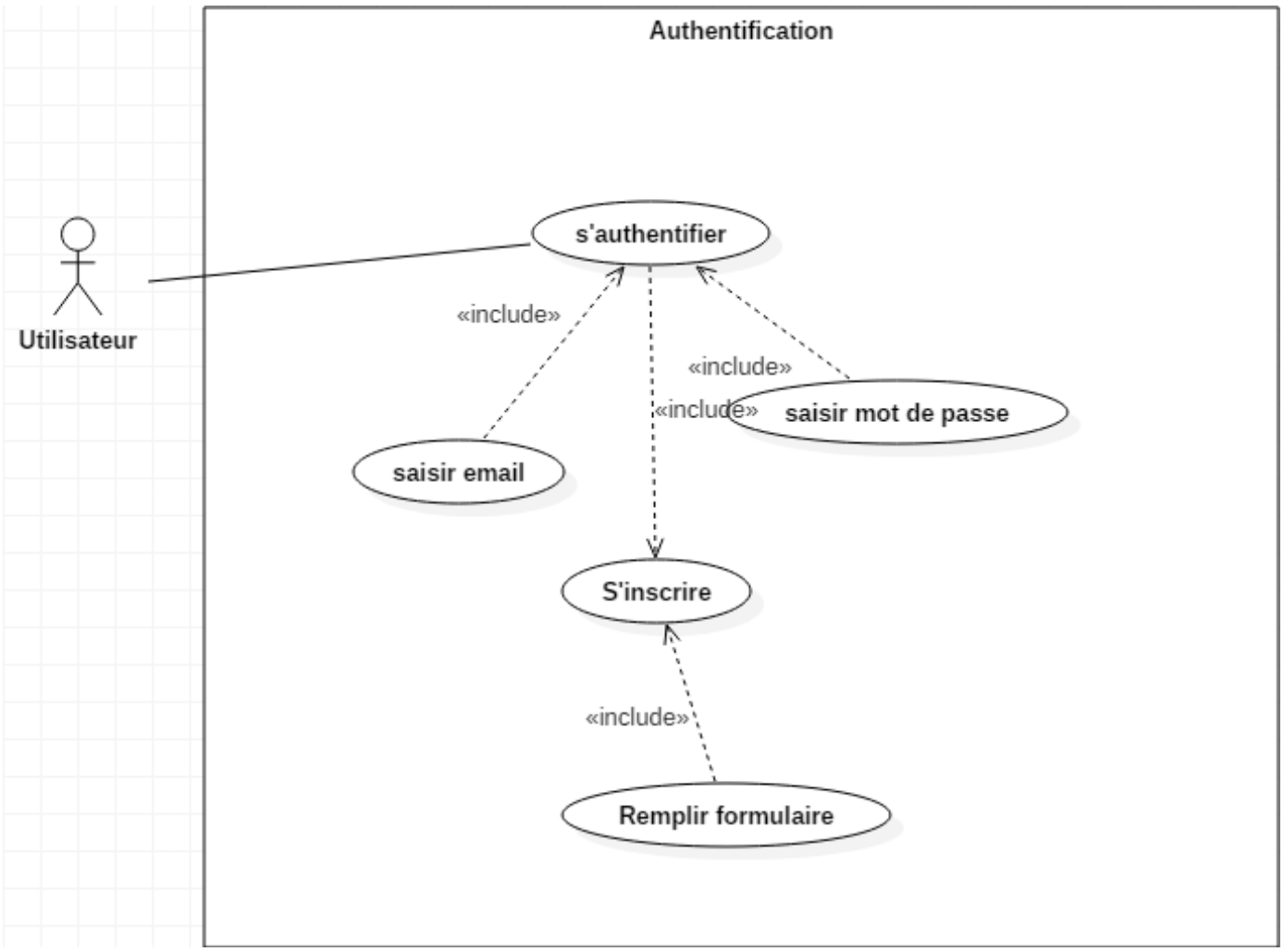

FIGURE 2.2 – Diagramme de cas d'utilisation authentification

Afin de s'authentifier l'utilisateur doit saisir son email et son mot de passe, s'il n'en a pas il doit remplir un formulaire d'inscription pour en avoir un.

Description textuelle du cas d'utilisation «s'authentifier»

Titre	s'authentifier
Résumé	Authentification d'utilisateur
Acteurs	candidat, Centre de formations,Formateur et Admin
Pré conditions	Les informations doit s'enregistrer dans le BD
Post condition	L'utilisateur peut consulter son espace.
Scenario nominal	L'utilisateur saisit login et mot de passe Si le login et le mot de passe sont corrects, l'utilisateur doit être dirigé vers sa page d'accueil.
Scenario alternatif	Le login et le mot de passe sont incorrects

TABLE 2.1 – Tableau cas d'utilisation Authentification

2.4.2.2 Diagramme de cas d'utilisation d'un visiteur

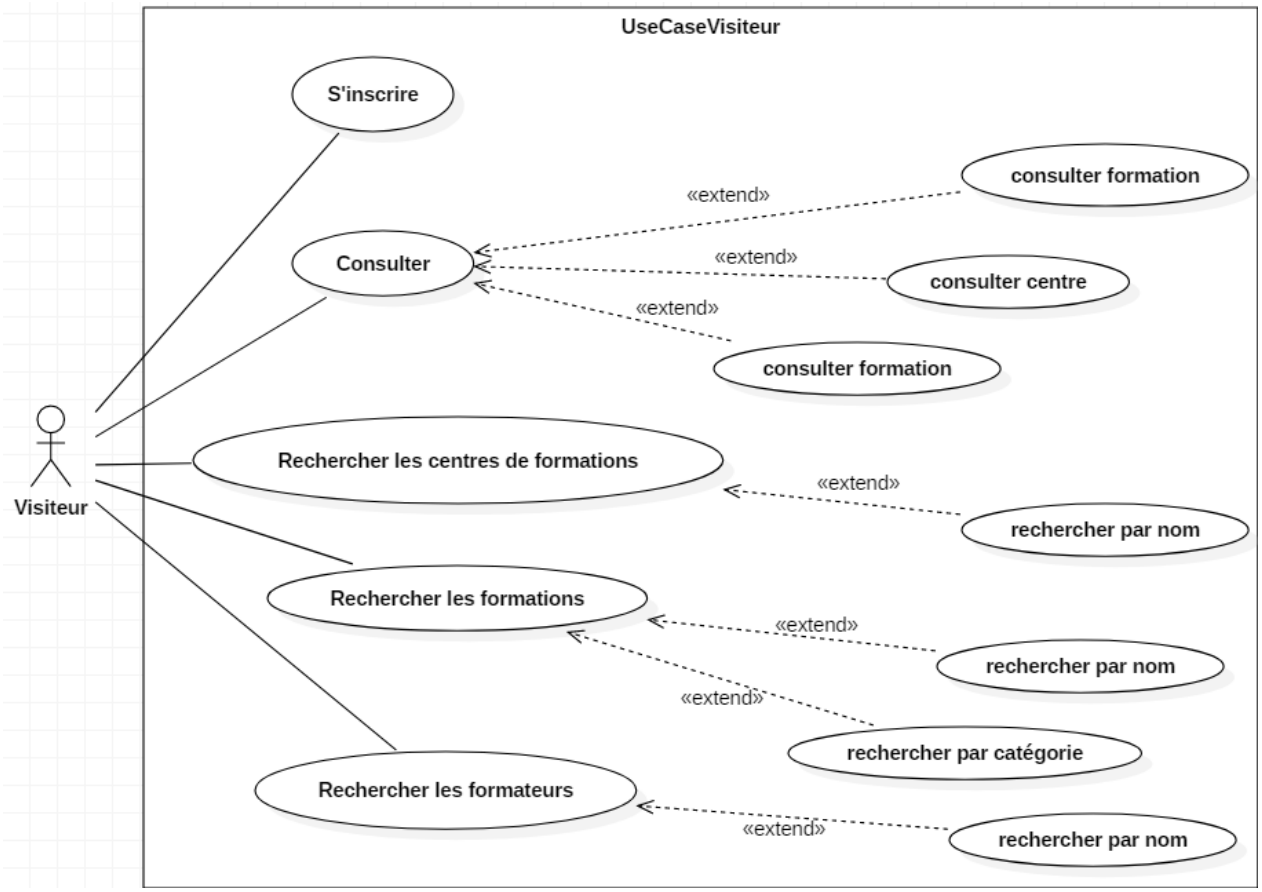

FIGURE 2.3 – Diagramme de cas d'utilisation Visiteur

La figure 2.3 présente le diagramme des cas d'utilisation d'un Visiteur où il peut :

-consulter l'accueil et chercher des centres de formations selon leurs noms
-consulter les formations, et chercher selon leurs noms et catégories
-consulter la page formateur et rechercher selon leurs noms ou des formations selon leurs catégories ou leurs noms.

Description textuelle du cas d'utilisation «Visiteur»

Titre	les différents gestions des Visiteur
Résumé	consultation de site
Acteurs	Visiteur
Pré conditions	Aucun
Post condition	peut consulter et rechercher dans les différentes pages
Scénario nominal	Le visiteur navigue et se dirige vers la page d'accueil et par suite il peut consulter les différentes pages.
Scénario alternatif	Aucun

TABLE 2.2 – Tableau cas d'utilisation de visiteur

2.4.2.3 Diagramme de cas d'utilisation de candidat

Dans ce diagramme, on présente de façon détaillée les fonctionnalités assurées par le candidat.

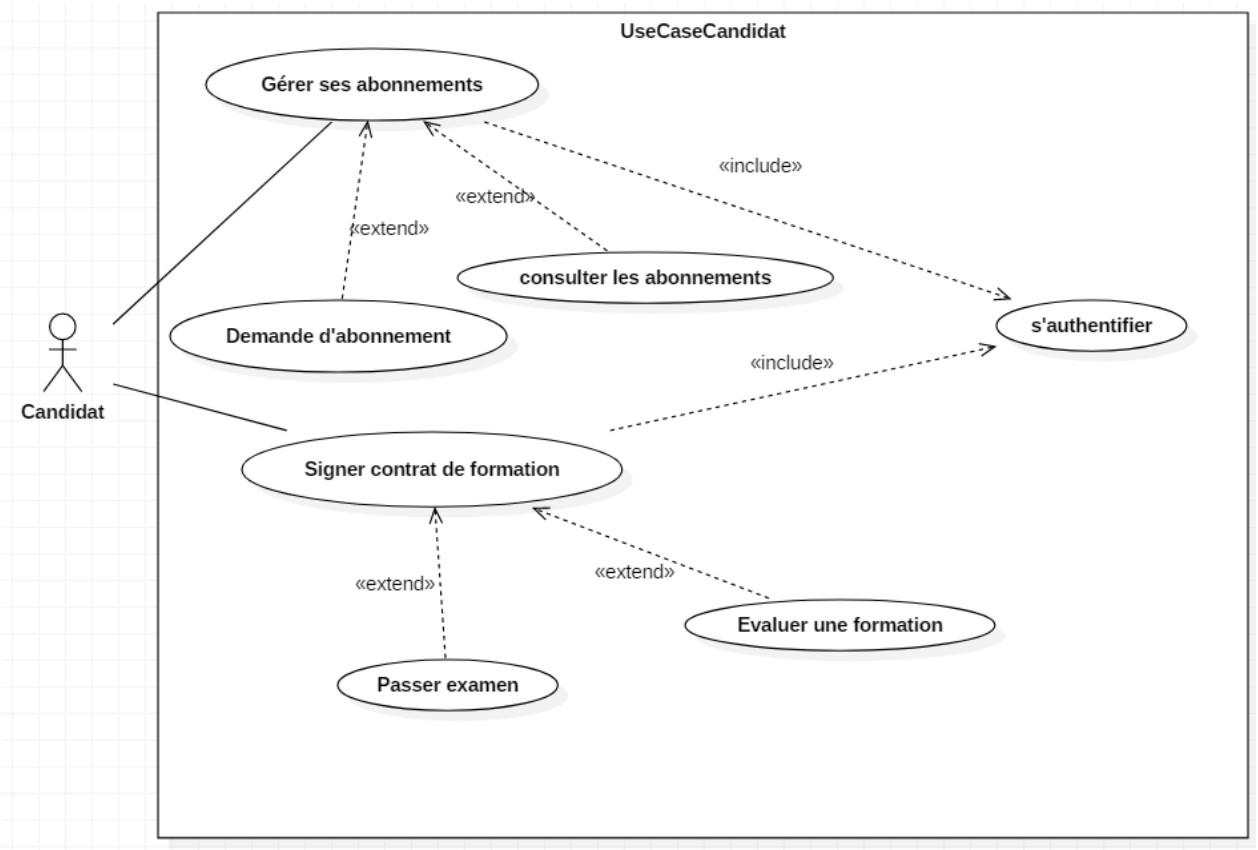

FIGURE 2.4 – Diagramme de cas d'utilisation Candidat

Lors de connexion de candidat, il va se rédiriger vers son dashboard.
Le candidat peut :
-gérer ses abonnements(demande d'abonnement, consulter d'abonnement).
-signer contrat de formation(passer examen, évaluer une formation) et gérer son profil.

Description textuelle du cas d'utilisation «candidat»

Titre	les différents gestions des candidat
Résumé	gestion des différents fonctionnalites de candidat
Acteurs	candidat
Pré conditions	Vérification de la validité de la session
Post condition	candidat gérée
Scénario nominal	Le candidat doit : -accéder à son profil, puis mettre à jour son profil et ses coordonnes. -accéder à son abonnements, puis le gérer. -consulter tous les formations puis il choisit ce qui lui convient et signe un contrat de formation
Scénario alternatif	N'est pas un candidat

TABLE 2.3 – Tableau cas d'utilisation de candidat

2.4.2.4 Diagramme de cas d'utilisation de formateur

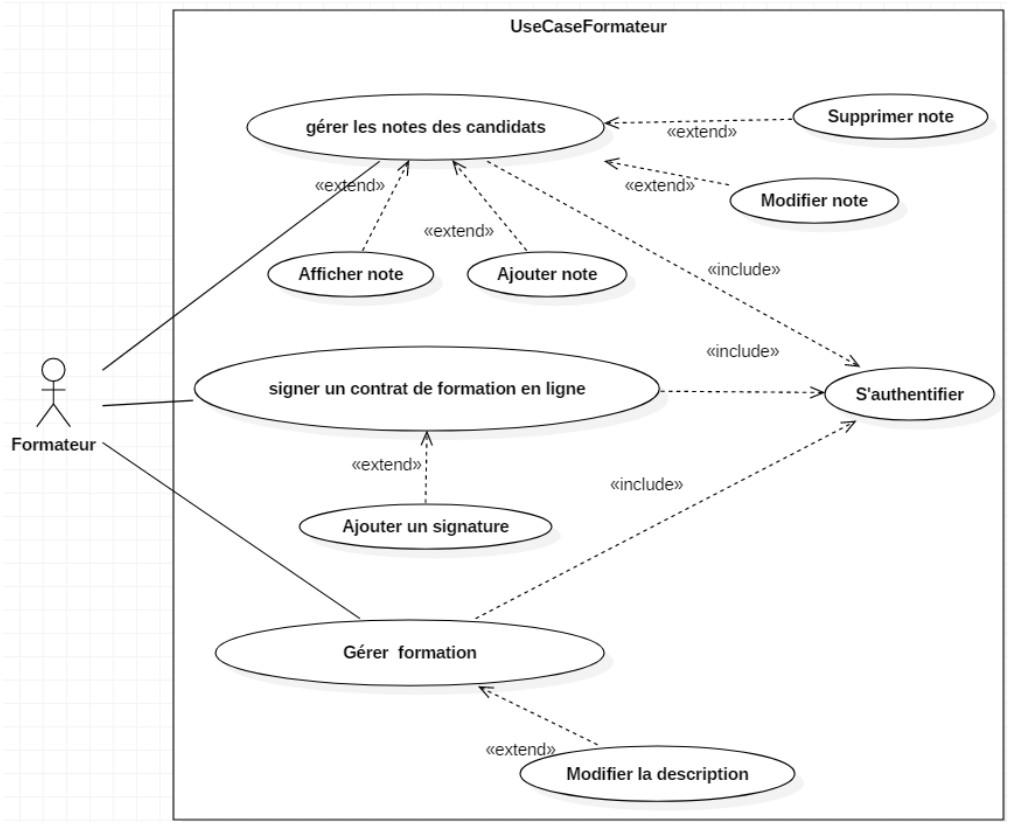

FIGURE 2.5 – Diagramme de cas d'utilisation formateur

Si le formateur connecte, il va se rédiriger vers son dashboard, il peut :
-gérer les notes des candidats(afficher une note, ajouter une note, modifier une note, supprimer une note).
-signer un contrat de formation en ligne.
-gérer une formation (modifier une description d'une formation)

Description textuelle de cas d'utilisation «formateur»

Titre	les différents gestions des formateur
Résumé	gestion des différents fonctionnalités de formateur
Acteurs	formateur
Pré conditions	Vérification de la validité de la session
Post condition	formateur gère son dashboard
Scénario nominal	Le formateur doit accéder : -à son profil, puis mettre à jour son profil et ses coordonnées. -à la page de note, puis gérer le note de candidat. -peut signer un contrat de formation en ligne après avoir une demande de formation envoyée par un centre de formation, puis modifie une description d'une formation.
Scénario alternatif	N'est pas un formateur

TABLE 2.4 – Tableau cas d'utilisation de formateur

2.4.2.5 Diagramme de cas d'utilisation de centre de formations

:

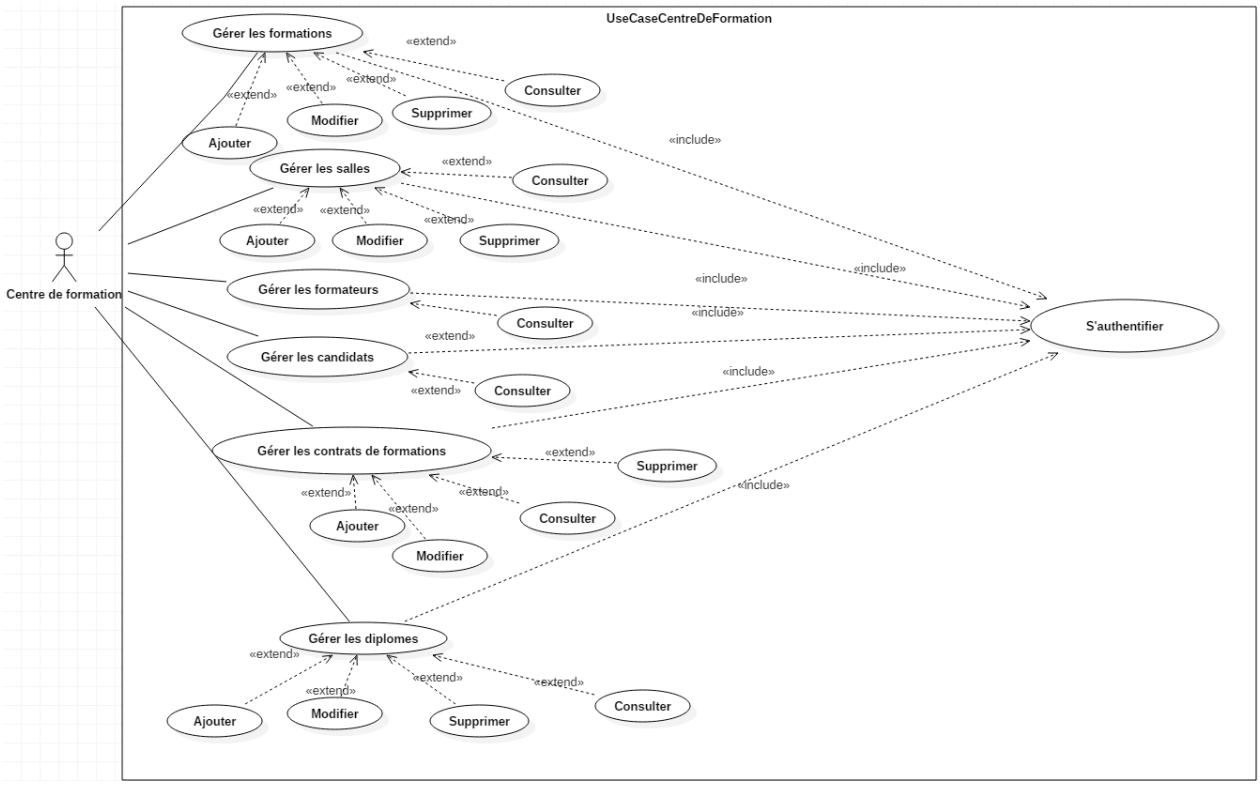

FIGURE 2.6 – Diagramme de cas d'utilisation de centre de formations

Titre	les différents gestions des centre de formations
Résumé	gestion des différentes fonctionnalites de centre de formations
Acteurs	centre de formations
Pré conditions	Vérification de la validité de la session
Post condition	centre de formations gère son dashboard
Scénario nominal	Le centre de formations doit accéder : -à son profil, puis mettre à jour son profil et ses coordonnées. -à la page de formations, puis gérer les formations de centre. -à la page de formateurs, puis gérer les formateurs de centre. -à la page de candidats, puis gérer les candidats de centre. -à la page de contrats de formations, puis gérer les contrats de formations de centre. -à la page de salles, puis gérer les salles de centre. -à la page de diplômes, puis gérer les diplômes de centre.
Scénario alternatif	N'est pas un centre de formations

TABLE 2.5 – Tableau cas d'utilisation de centre de formations

2.5 Conclusion

Dans ce chapitre on a détaillé les spécifications et les besoins du projet. Ensuite nous avons présenté les diagramme de cas d'utilisation global et les principaux cas d'utilisation

détaillés du système à développer.
L'objectif du chapitre suivant est de présenter la conception de l'application, qui c'est une phase importante dans le cycle de développement d'une application.

Chapitre 3

Conception

3.1 Introduction

L'étape de conception définit généralement les structures et les modèles à suivre lors de la phase d'implémentation de l'application. C'est la phase où nous préparons l'architecture du projet, et où nous définissons la structure de l'application.

3.2 Vue statique de l'application

3.2.1 Les modèles architecturaux

Nous proposons dans ce qui suit deux modèles qui nous avons pris en considération lors de la conception de notre plateforme de digitalisation des centres de formations. En premier lieu, nous avons défini l'architecture n-tiers traditionnelle (MERN Stack – ES6) qui représente l'architecture physique de notre système. Puis, nous présentons, en second lieu le patron de conception MVC.

3.2.1.1 L'architecture n-tiers

Le principe d'une architecture n-tiers est relativement simple : il consiste à séparer la réalisation des trois parties vues(stockage des données, logique applicative, présentation) représenté dans la figure ci-dessous qui représente une l'architecture 3 -tiers (MERN Stack – ES6)

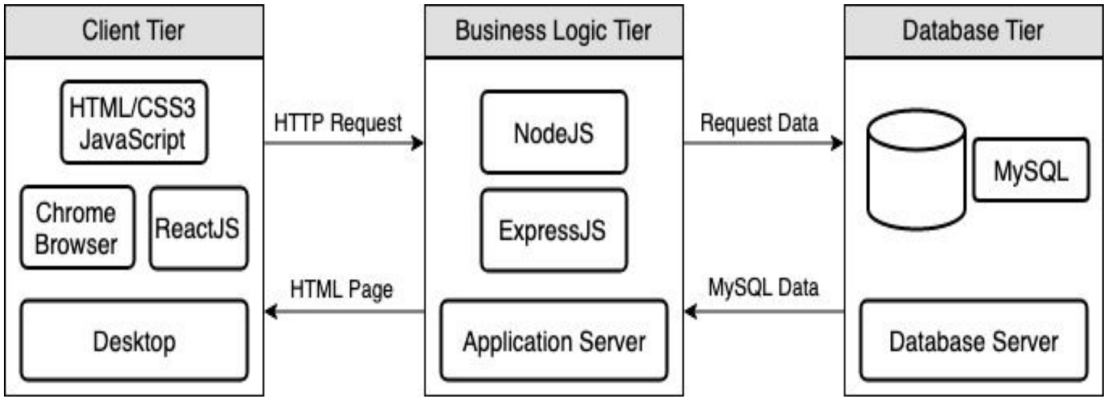

FIGURE 3.1 – Architecture 3-tiers (MERN Stack - ES6)

Comme vous voyez l'architecture de notre application est basée sur un modèle MVC typique. Notre niveau Client (View) sera écrit en Javascript, HTML et CSS, en utilisant ReactJS et Redux comme cadre. Ce niveau d'architecture est ce avec quoi l'utilisateur interagira pour accéder aux fonctionnalités de notre application. Le niveau de logique métier (contrôleur) sera écrit à l'aide de NodeJs et ExpressJS, et ce niveau représente le serveur d'application qui agira comme un pont de communication pour le niveau client et le niveau base de données. Ce niveau servira des pages HTML à l'appareil de l'utilisateur et acceptera les requêtes HTTP de l'utilisateur et suivra avec la réponse appropriée. Notre niveau de base de données (modèle) hébergera MongoDB. C'est là que nous stockerons toutes les données cruciales dont notre application a besoin pour fonctionner.

Implémenter une application avec une telle architecture donne la possibilité de séparer la conception de ces trois subdivisions, ici il s'agit de séparer leur implantation. Tout comme dans le client -serveur cette séparation signifie qu'il est possible de déployer chaque partie sur un serveur indépendant, toutefois cela n'est pas obligatoire. La mise en place de ce type d'architecture permet dans tous les cas une plus grande évolutivité du système. Il est ainsi possible de commencer par déployer les deux serveurs sur la même machine, puis de déplacer le serveur applicatif sur une autre machine lorsque la charge devient excessive.

Les éléments permettant la réalisation classique d'un système en architecture trois tiers sont les suivants :

- Système de base de données pour le stockage des données
- Serveur applicatif pour la logique applicative
- Navigateur web pour la présentation

3.2.1.2 L'architecture MVC

Le design pattern Modèle-Vue-Contrôleur (MVC) est un pattern architectural qui peut facilement sépare le code lié à la base de données (le modèle) de celui lié à l'interface homme-machine (la vue) avec une couche intermédiaire (le contrôleur), ce qui facilite la gestion et la mise à niveau.

La figure suivante décrit les interactions entre les différentes couches :

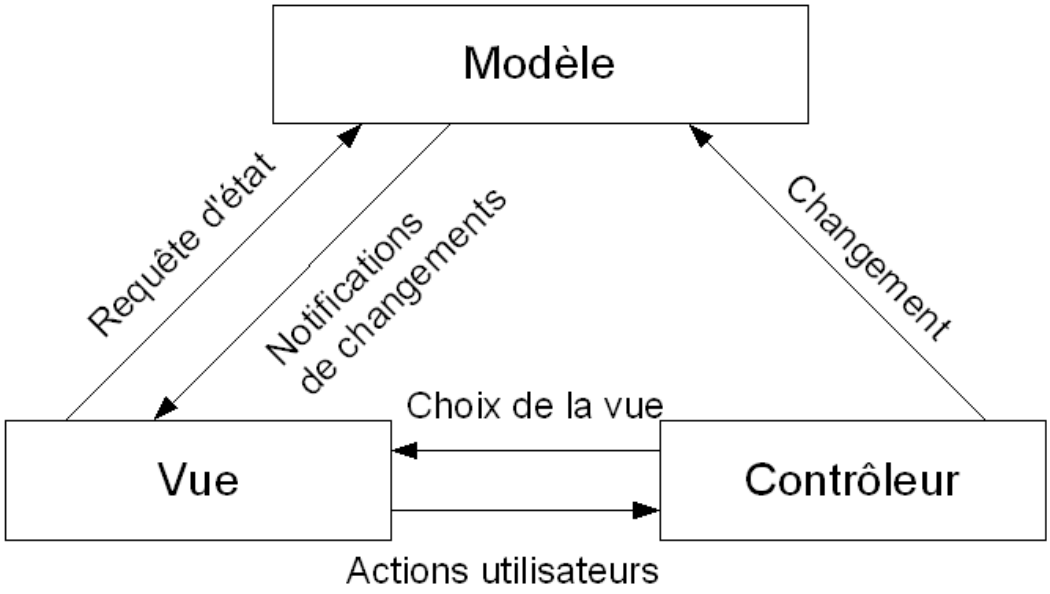

FIGURE 3.2 – Interactions entre les couches

Le modèle : La couche modèle représente les données de l'application, il définit aussi l'interaction avec la base de données et le traitement de ces données.

La vue :Elle représente l'interface utilisateur, ce avec quoi il interagit. Elle n'effectue aucun traitement, elle se contente simplement d'afficher les données que lui fournit le modèle.Il peut tout à fait y avoir plusieurs vues qui présentent les données d'un même modèle.

Le contrôleur :Il gère l'interface entre le modèle et le client. Il va interpréter la requête de ce dernierpour lui envoyer la vue correspondante. Il effectue la synchronisation entre le modèleet les vues.La figure suivante décrit les interactions entre les différentes couches

3.2.2 Conception de la Base de données

3.2.2.1 modèle conceptuel des données

Comme première étape de la conception statique, nous avons récolté toute donnée nécessaire dans notre système d'information pour construire un dictionnaire de données qui se présente dans l'annexe A.Le modèle conceptuel des données (MCD) a pour but de décrire d'une façon formelle les données qui seront utilisées par le système d'information. Il s'agit donc d'une représentation des données compréhensibles, permettant de décrire le système d'information à l'aide d'entités.Il faut noter que ce modèle a été définit après avoir identifié le dictionnaire de données présenté dans l'annexe A.La figure 3.3 représente le MCD de l'application composée des entités suivantes :

La figure 3.3 représente le MCD de l'application compose des entités suivantes :

— L'entité « User » représente le profil de l'utilisateur inscrit dans l'application , il doit posséder principalement un nom, un prénom, un email, password, statut (1 pour les

comptes actives, 0 pour les comptes inactives), rôle (l'utilisateur soit un administrateur soit un manager (gerant d'un centre de sport) soit un client), et password.

— L'entité « Administrateur» présente le profil d'un administrateur, elle contient les informations nécessaires pour leurs comptes. Administrateur est une classe héritée de la classe mère User, il possède tous les attributs de User.

— L'entité « candidat» présente le profil d'un client, elle contient les informations nécessaires pour leurs comptes. Candidat est une classe héritée de la classe mère User , elle doit possèder principalement une adresse, une ville, un code postale et une date de naissance.

— L'entité «Centre» représente l'administrateur de centre de formations , elle doit posséder principalement un nom, une photo, une ville, une adresse, un code postale.

— L'entité «Evaluation » modélise l'évaluation faite par le candidat , géré par l'administrateur elle se caractérise principalement par un libelle, et une note.

3.2.3 Conception logiciel

Après avoir élaboré la conception de la base des données bien détaillée, nous passons à la conception des fonctionnalités de notre solution en se basant sur le langage UML tout en respectant les contraintes du modèle MVC.

3.2.3.1 architecture générale de l'application

La figure ci-dessous présente l'architecture générale de notre application, nous avons adopté une architecture 3-tiers. C'est un modèle logique d'architecture applicative qui vise à

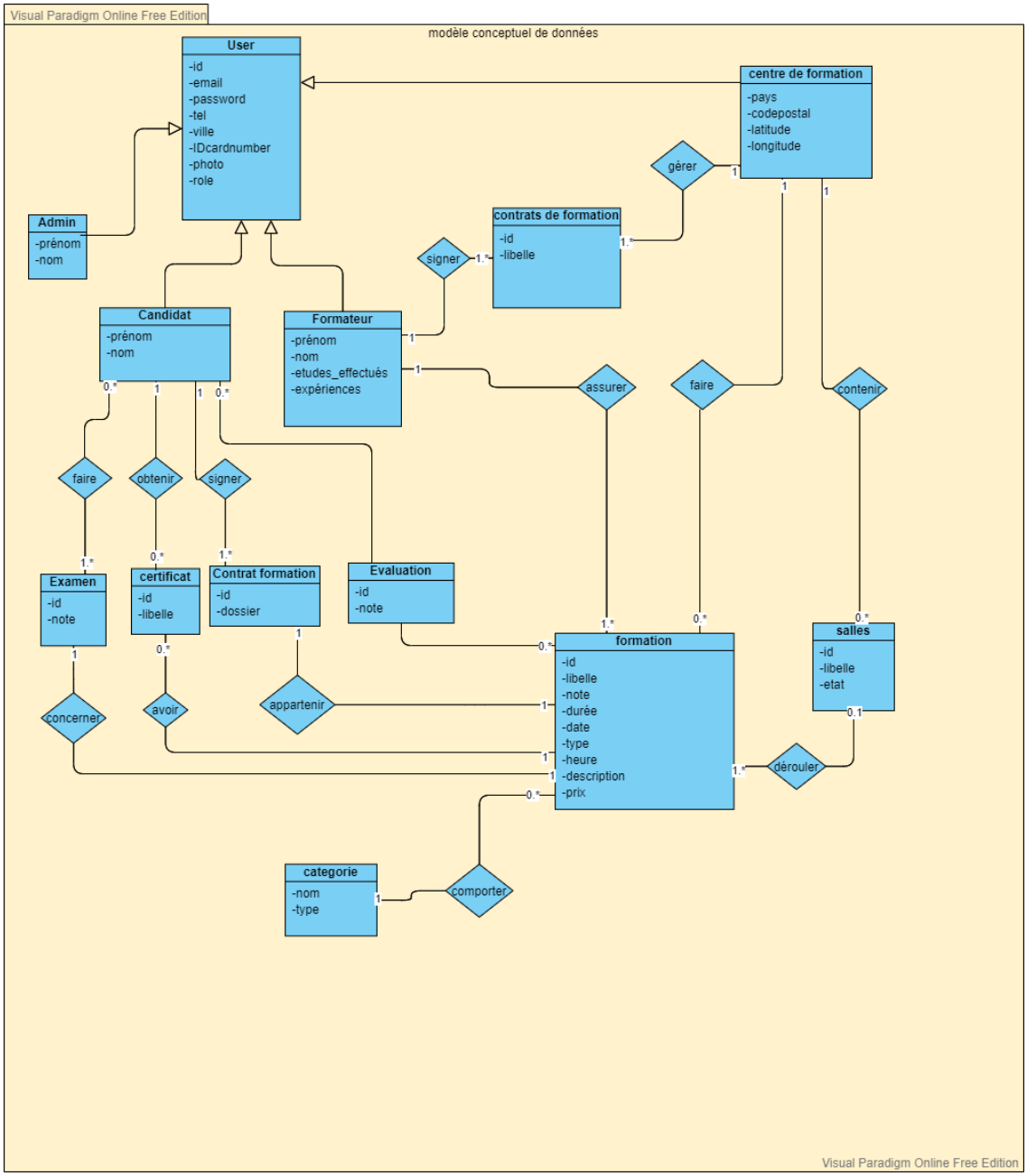

FIGURE 3.3 – modèle conceptuel de données

séparer nettement la couche de traitement et la couche de données au sein d'un même système. L'architecture suivante montre l'interaction entre les différentes technologies de notre projet, la communication entre le back-end (Node JS), front-end (React, Redux) et la base des données (Mongo DB). Cette interaction se réalise grâce aux services web.

La spécification de l'architecture du projet consiste à répartir les différentes composantes utilisées. Dans ce contexte, nous proposons le diagramme de déploiement de l'architecture illustré dans la figure 3.6 qui englobe deux partie : une partie back-office pour l'administration et une partie font-office pour l'utilisateur. La communication (l'échange) entre ces deux parties se fait à travers une API.

FIGURE 3.4 – architecture générale de l'application

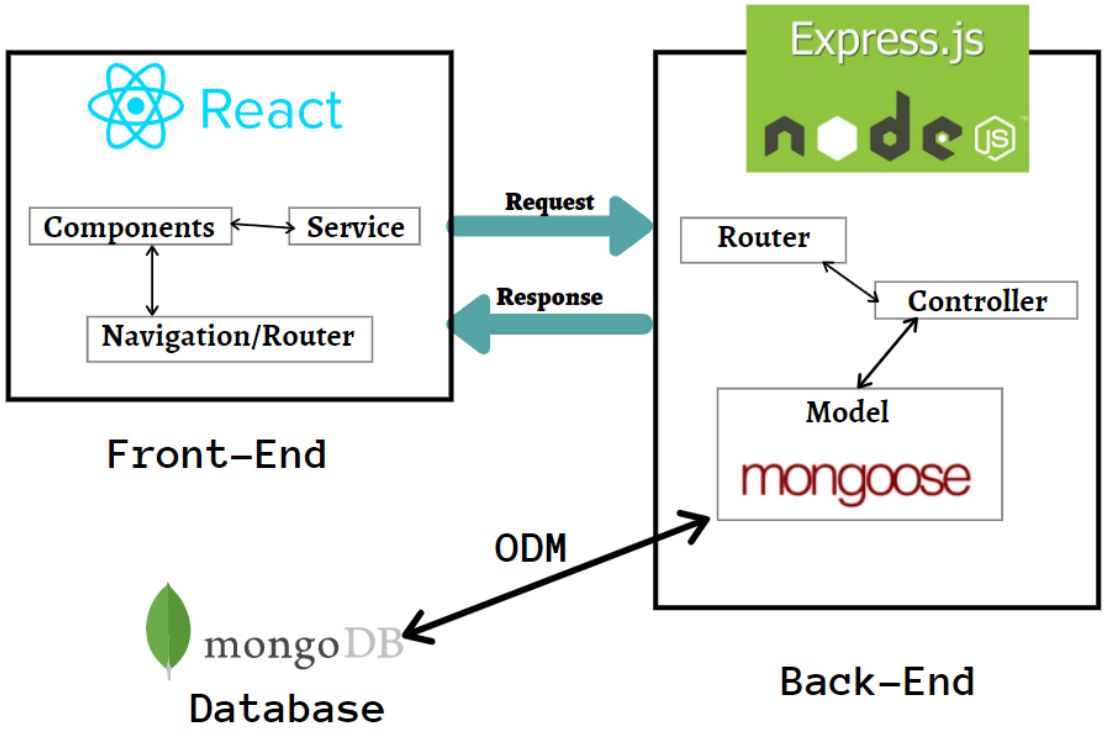

FIGURE 3.5 – diagramme de déploiement

3.2.4 Choix du modèle conceptuel

Le bon choix de l'architecture est une phase capitale dans le cycle de vie d'une application. Nous avons eu recourt au design pattern MVC pour la partie back-end, de l'architecture MVVM dédiée pour le front-end et SOA comme l'architecture globale de notre application.

3.2.4.1 Modèle MVC

On va commencer par la présentation du modèle MVC choisi pour la réalisation de notre application cotée back-end. MVC est un patron de conception qui permet de bien découper (ou séparer) le code en trois parties comme l'illustre la figure ci-dessous afin de mieux organiser le code de notre système :

— La vue :
Elle correspond à l'IHM (interface homme machine). Cette partie se concentre sur l'affichage pour cela on trouve essentiellement du code HTML.

— Le modèle :modèle de donnée qui représente la couche métier de l'application.
Il représente les données brutes et l'interaction avec la base de données.

— Le contrôleur :
Il orchestre la procédure entre la vue et le modèle. C'est une logique de contrôle qui permet de gérer les évènements et d'assurer la synchronisation.

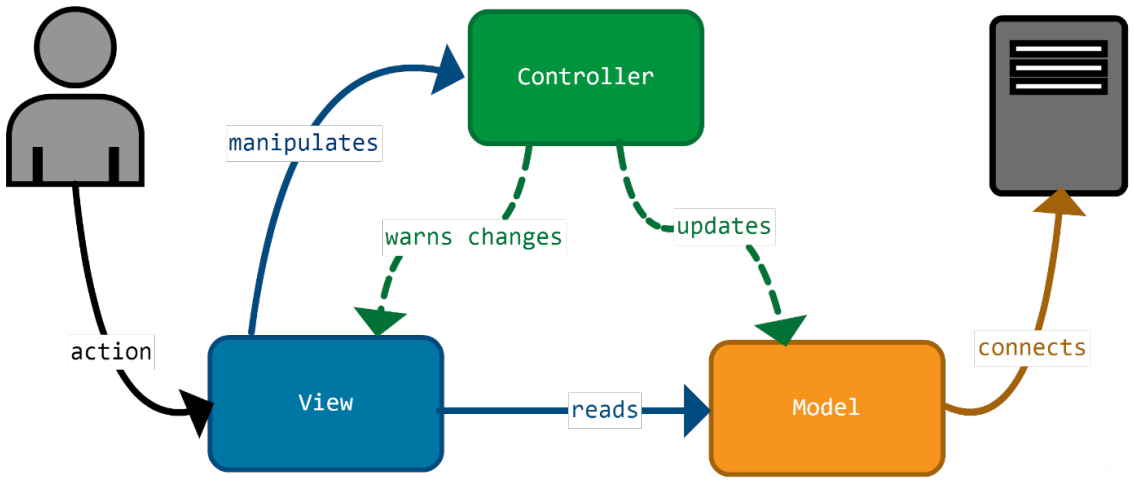

FIGURE 3.6 – architecture MVC

3.2.4.2 Architecture SOA

Notre plateforme est basée sur l'architecture orientée services (SOA) qui est une approche utilisée pour créer une architecture basée sur l'utilisation d'un ensemble des services (tels que les services web RESTful qui remplissent certaines fonctions comme la production de données et la fourniture de service par l'appel d'une url en http).
Un service web est un composant logiciel identifié par une URI, dont les interfaces publiques sont définies et appelées en XML. Ils sont transportés par les protocoles http mais qui peuvent s'appuyer sur d'autre protocole de transport.
La consommation d'un web service revient à appeler une simple url http (get, post, put, delete, update) .

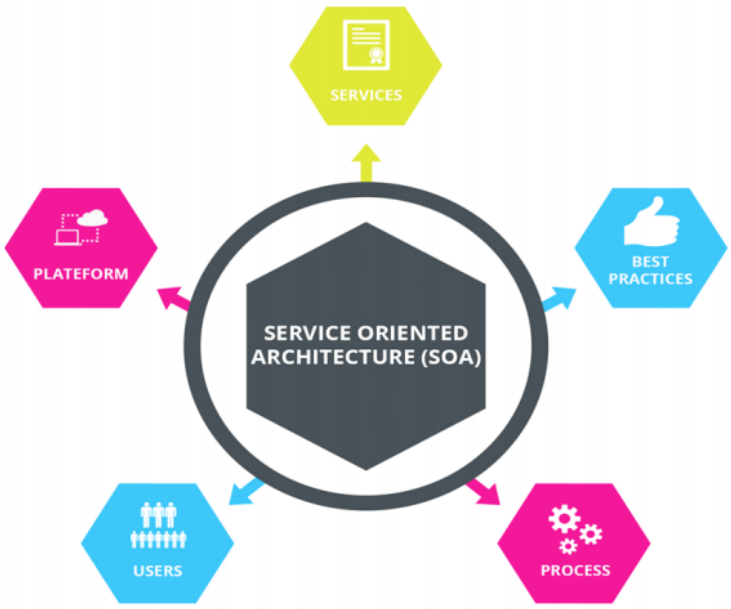

FIGURE 3.7 – architecture SOA

3.2.4.3 Modèle MVVM

Notre solution utilise l'architecture MVVM (modèle-vue-vue-modèle) pour la partie front-end. MVVM est similaire au modèle MVC en ce qui concerne le modèle et la vue, la seule différence réside entre le C (Controller) et la VM (View Model) . Il comprend trois composants :

— Le Modèle :
Il contient les données métier de l'application. Il permet de faire l'échange avec le serveur et notifie la Vue-Modèle de tout changement.

— La Vue :
Elle représente l'interface utilisateur de l'application. Elle reçoit les différentes actions de l'utilisateur.

— La Vue-Modèle : C'est un lien (intermédiaire) entre le modèle et la vue (elle permet de recevoir tout changement des données du modèle et le présenter à la vue).

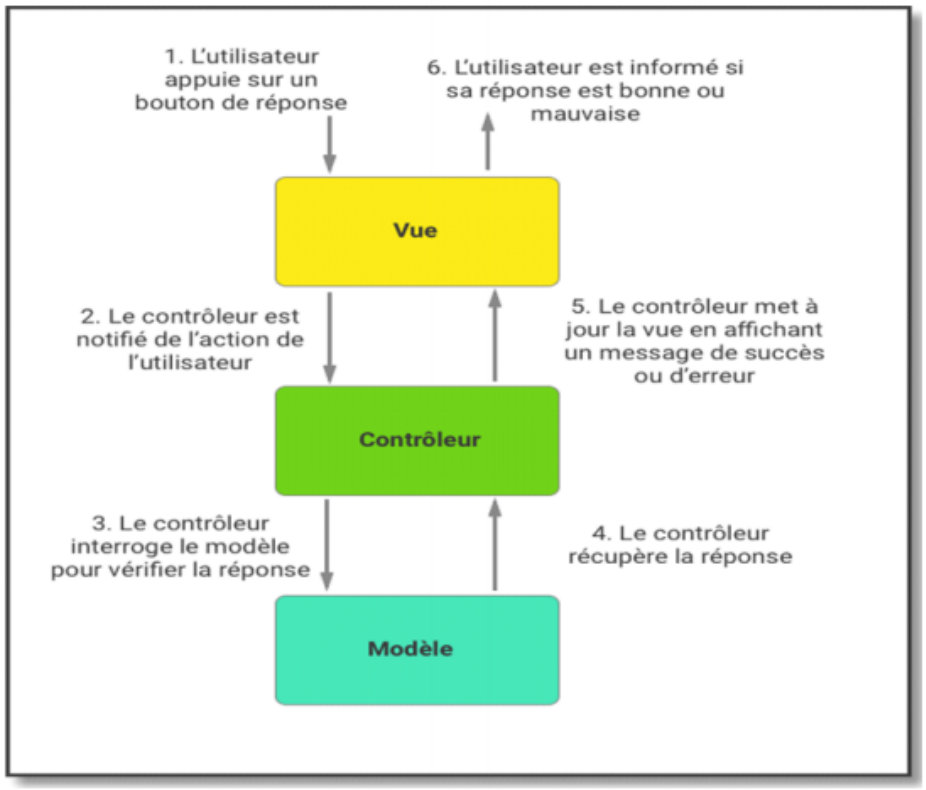

FIGURE 3.8 – Modèle MVC

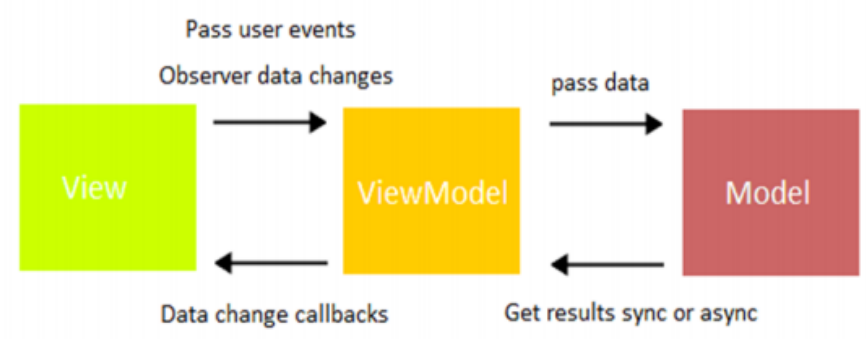

FIGURE 3.9 – Modèle MVVM

3.2.5 Diagrammes de classes

Le diagramme de classe est le point central de la modélisation du système pour exprimer sa structure statique.

3.3 Vue dynamique de l'application

3.3.1 Diagrammes de séquence

3.3.1.1 Diagrammes de séquences détaillés

Diagramme de séquence «authentification»

Diagramme de séquence «Inscription »

Diagramme de séquence «Ajouter formation»

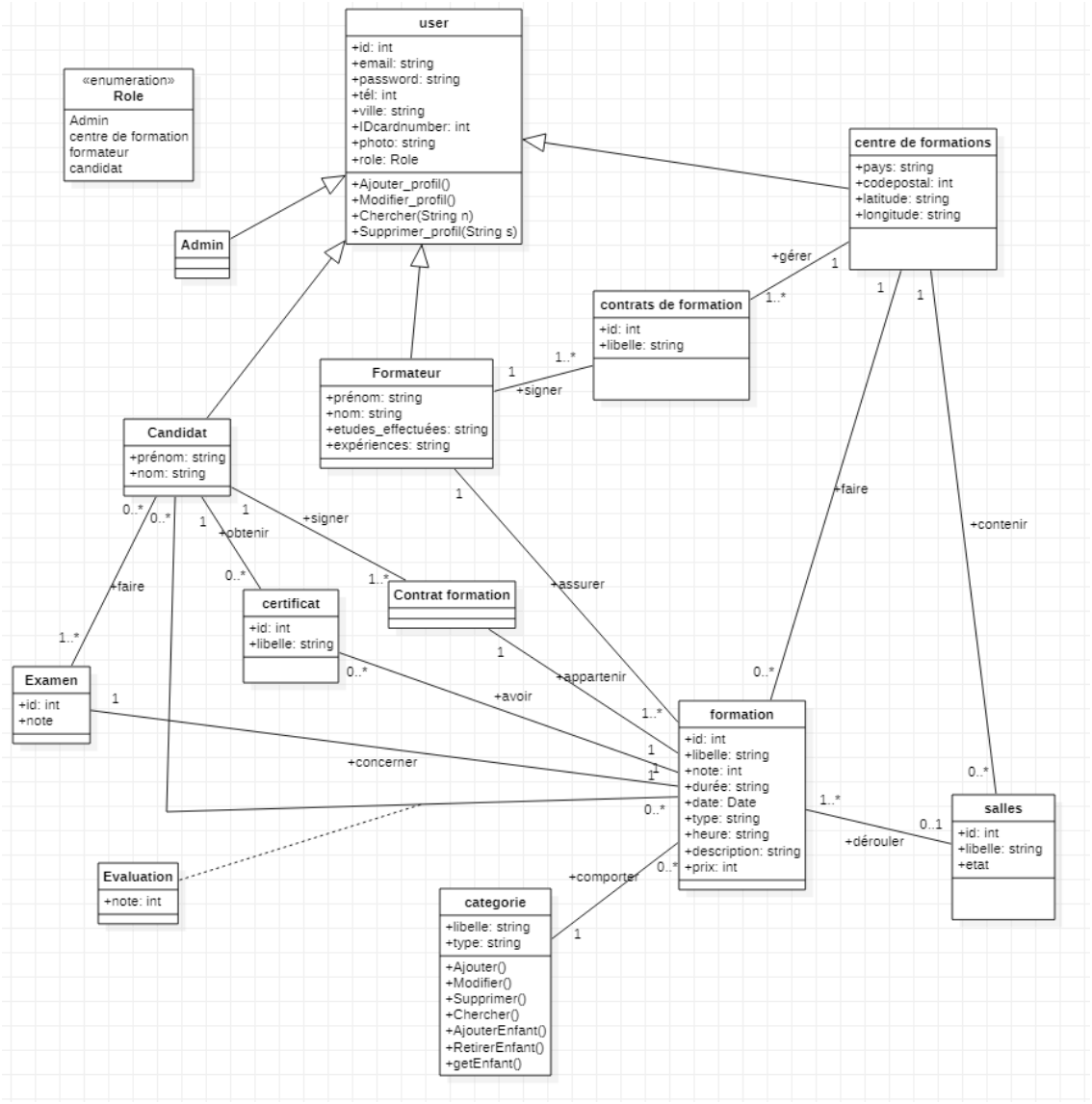

FIGURE 3.10 – Diagramme de classes

3.3.2 Diagrammes d'activités

C'est un Diagramme associé à un objet particulier ou à un ensemble d'objets, qui illustre les flux entre les activités et les actions. Il permet de représenter graphiquement le déroulement d'un cas d'utilisation.

3.3.2.1 Diagramme d'activité «Inscription»

La phase d'inscription est indispensable pour passer d'un simple visiteur du site qui n'a le droit que de consulter les produits et leurs prix à un client qui peut déposer des annonces ou des demandes de services.

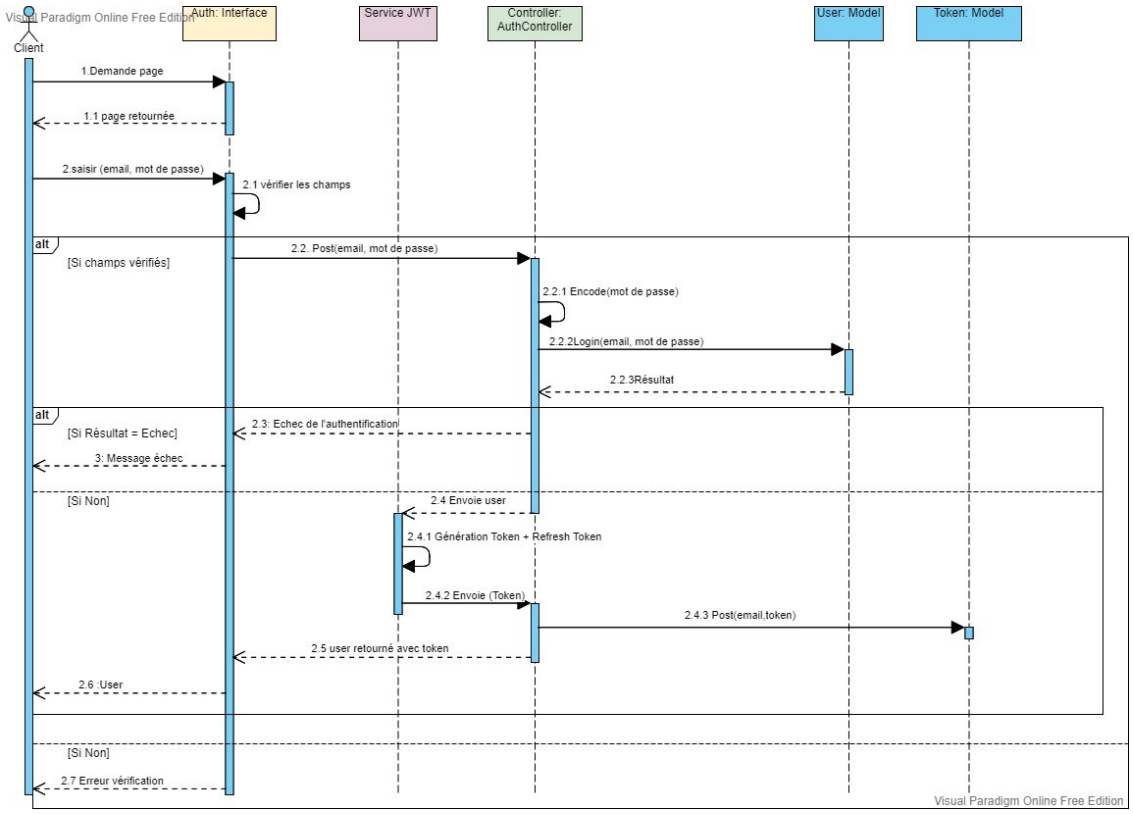

FIGURE 3.11 – Diagramme de séquence «authentification»

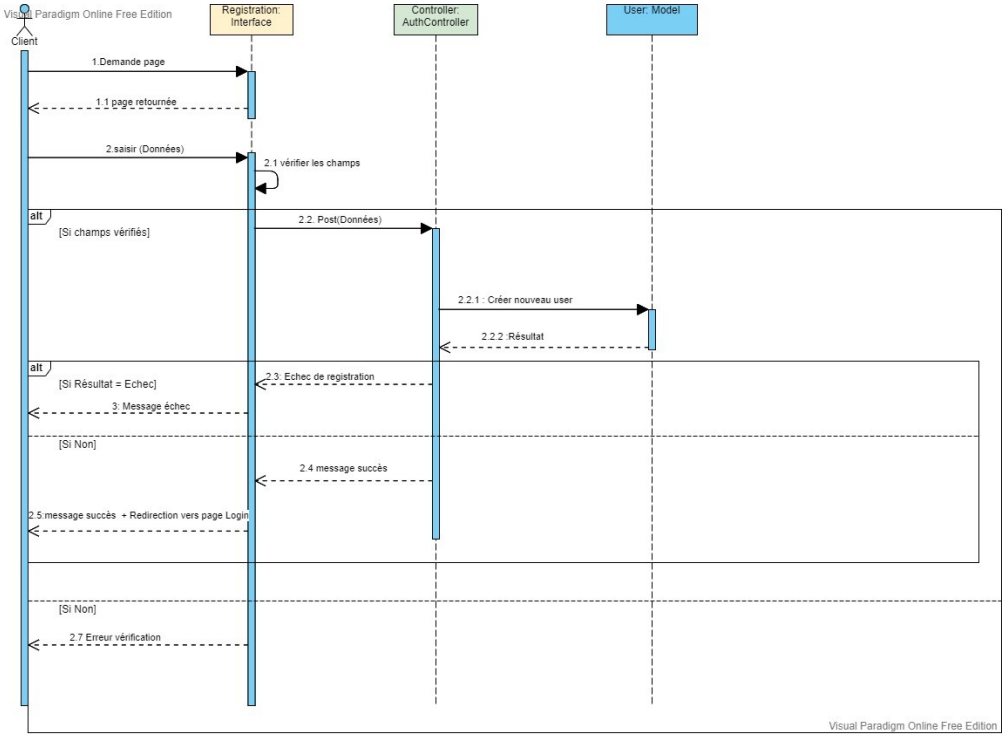

FIGURE 3.12 – Diagramme de séquence «Inscription»

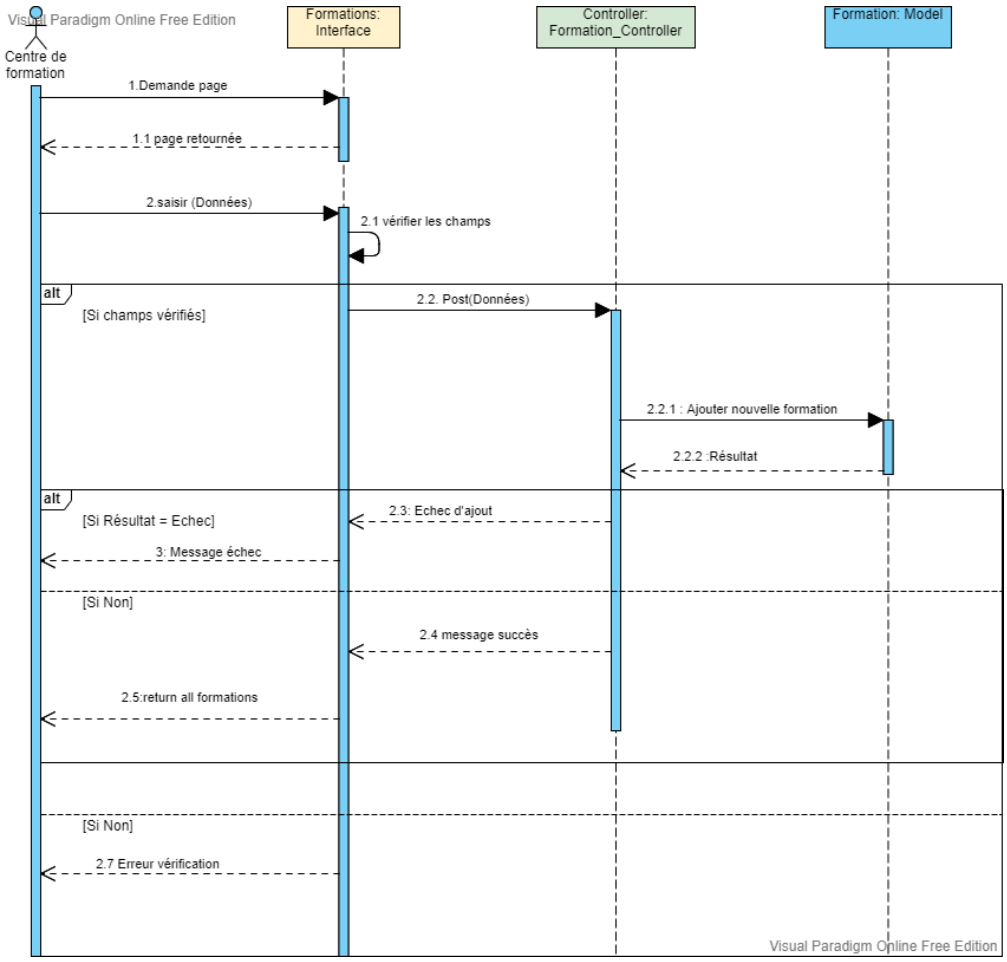

FIGURE 3.13 – Diagramme de séquence «Ajouter formation»

— Le visiteur demande l'inscription.

— Le formulaire d'inscription s'affiche sur l'écran.

— Le visiteur remplit les champs demandés dans le formulaire.

— Le système vérifie les données entrées.

— Si les données sont acceptées, le système les envoie à la base si non, il revient à l'étape précédente.

— Le serveur vérifie l'existence du client dans la base.

— Si le client(ou le manager) existe déjà, un message d'erreur s'affiche.

— Si le client(ou le manager) n'existe pas, l'inscription se termine avec succès.

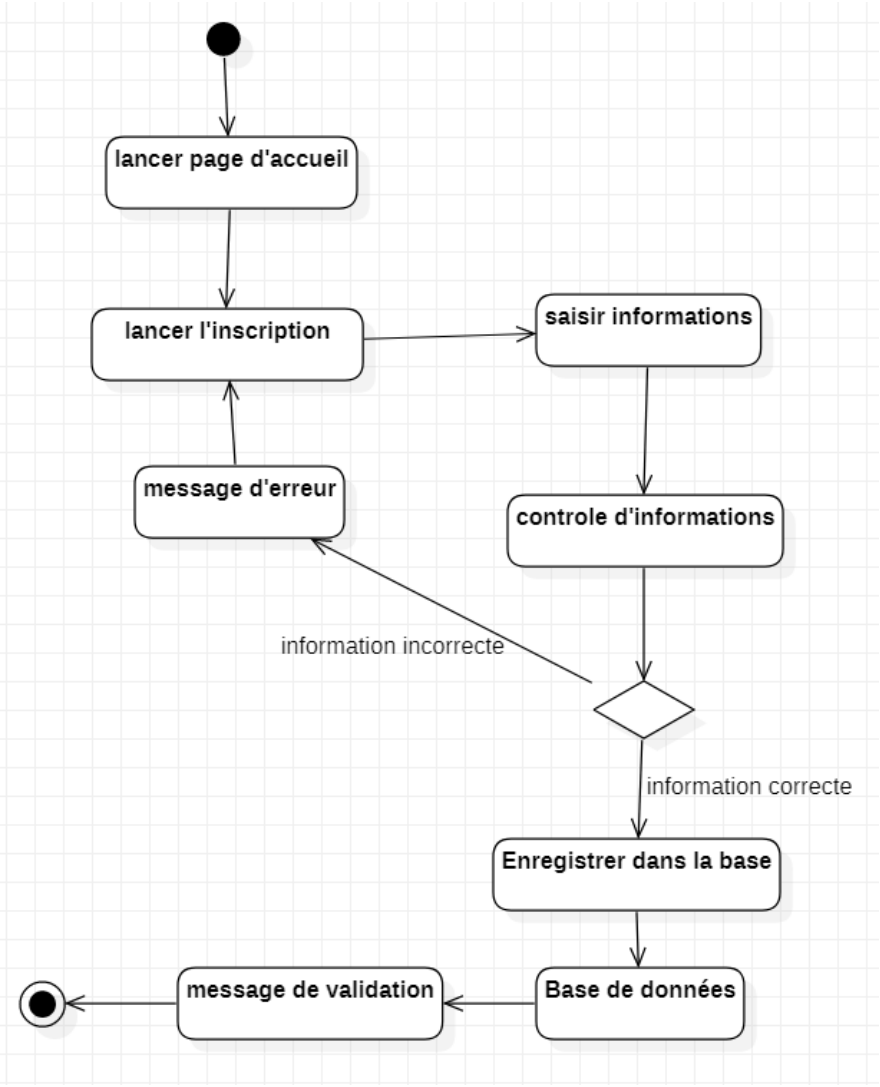

FIGURE 3.14 – Diagramme d'activité «Inscription»

3.3.2.2 Diagramme d'activité «Authentification»

— L'utilisateur (admin ou centre de formations ou formateur ou candidat) demande l'authentification en cliquant sur le bouton login.
— Le formulaire d'authentification s'affiche sur l'écran.
— L'utilisateur entre son email et son mot de passe.
— Le système vérifie les coordonnés du client sur la base.
— La conformation du succès ou échec est envoyée à l'écran de l'utilisateur.

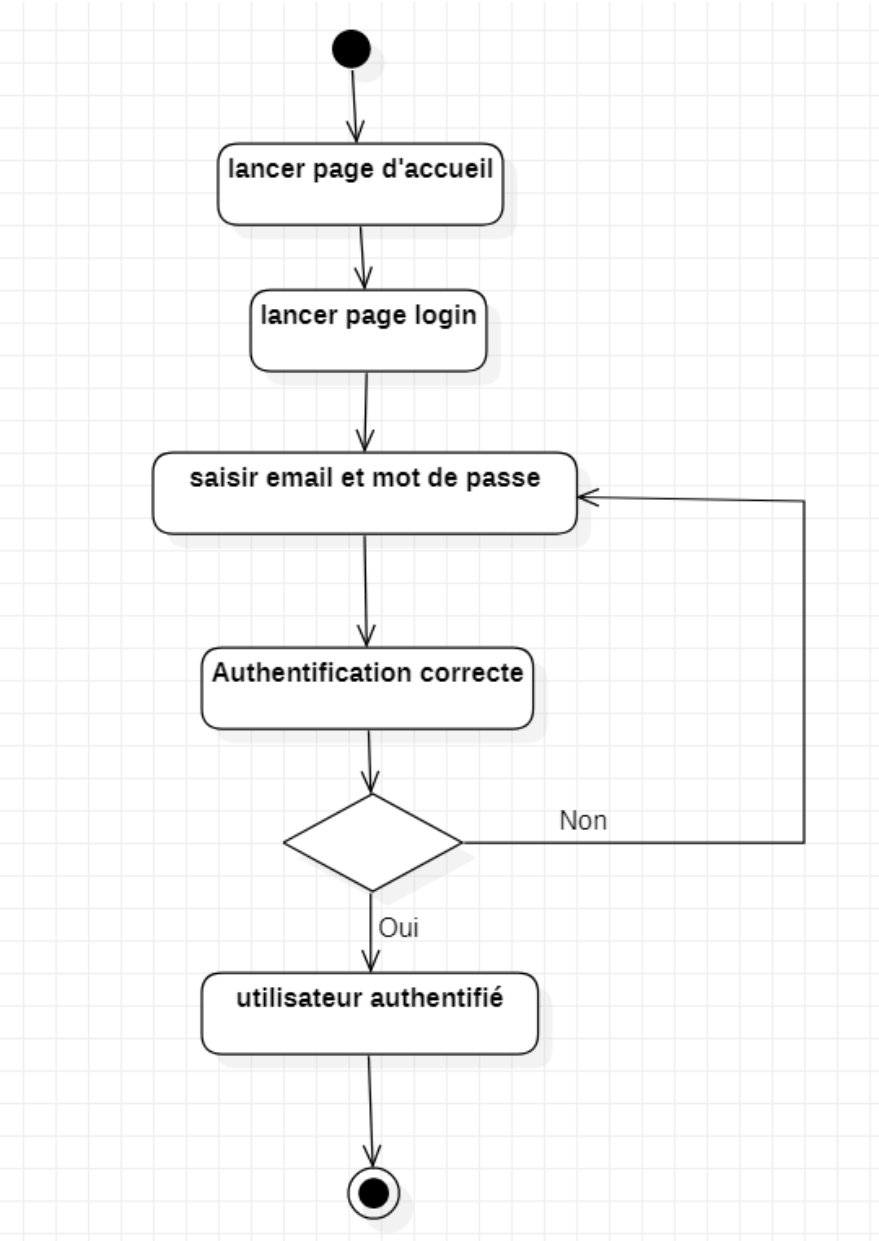

FIGURE 3.15 – Diagramme d'activité «Authentification»

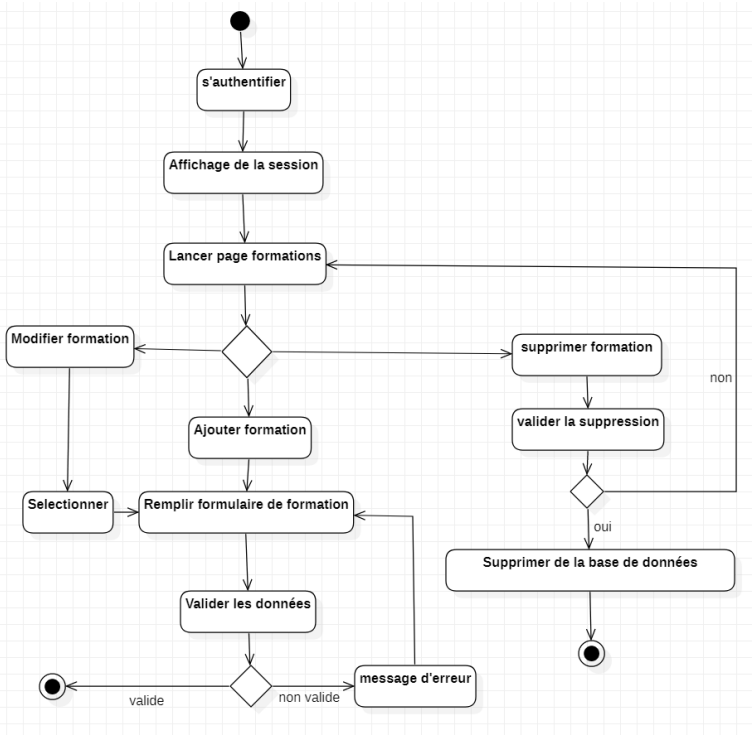

FIGURE 3.16 – Diagramme d'activité«gestion de formation»

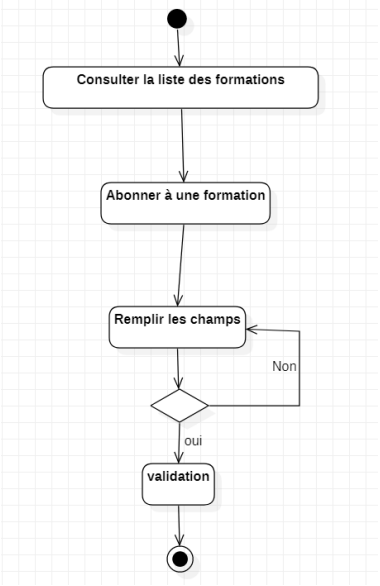

FIGURE 3.17 – Diagramme d'activité«abonner formation»

3.3.2.3 Diagramme d'activité «gestion de formation»

3.3.2.4 Diagramme d'activité «abonner formation»

3.3.3 Diagrammes d'états-transitions

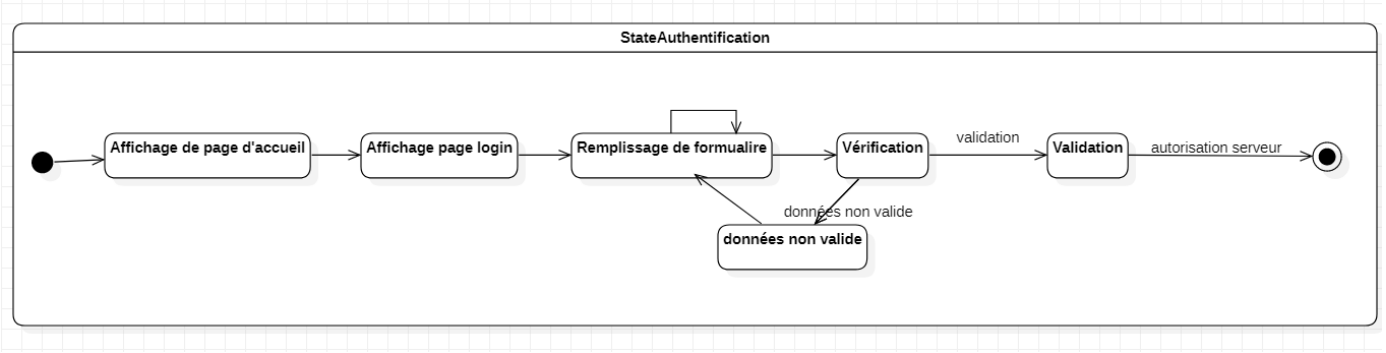

FIGURE 3.18 – Diagrammes d'états-transitions «Authentification»

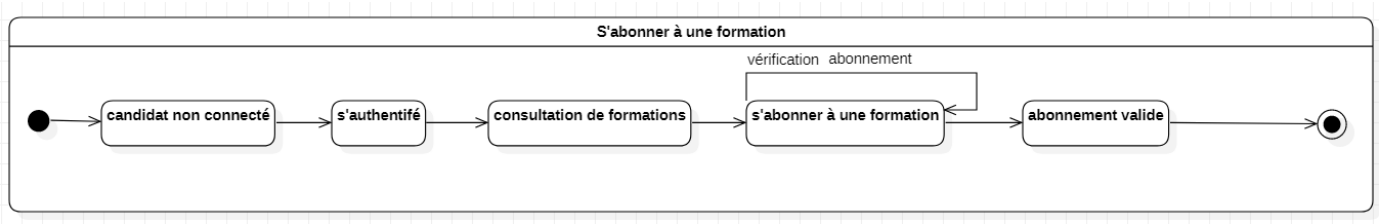

FIGURE 3.19 – Diagrammes d'états-transitions «s'abonner à une formation»

Cette étude adoptée facilite par la suite la réalisation de l'application, qui sera par la suite le contenu de notre quatrième chapitre.

Chapitre 4

Réalisation

4.1 Introduction

Après avoir élaboré la conception de notre application, nous avons abordé dans ce chapitre le dernier volet de ce rapport, qui a pour objectif d'exposer la phase de réalisation. La phase de réalisation est considérée comme étant la concrétisation finale de toute la méthode de conception. Nous menons tout d'abord une étude technique où nous décrivons les ressources logicielles utilisées dans le développement de notre projet. Nous présentons en premier lieu notre choix de l'environnement de travail, où nous spécifions l'environnement matériel et logiciel que nous avons utilisé pour réaliser notre application, aussi nous présentons quelques interfaces réalisées pour illustrer le fonctionnement de quelques activités du système.

4.2 Environnement et outils de travail

Dans cette section, nous présentons les environnements matériels et logiciels utilisés dans le cadre de notre projet.

4.2.1 Environnement matériel

Pendant les différentes phases de notre projet à savoir la documentation, la spécification des besoins, la conception et le développement, nous avons disposé d'un PC ayant les caractéristiques suivant :

Marque	Lenovo
Processeur	intel(R) Core(TM) i7-8565U CPU @ 1.80ghz 1.99 ghz
RAM	16.0 GO
Disque Dur	930 GO
Système d'exploitation	Microsoft Windows 10 Famille

Table 4.1 – Environnement matériel

4.2.2 Environnement logiciel

4.2.2.1 StarUML

StarUML est un logiciel de modélisation UML (Unified Modeling Language) open source qui peut remplacer dans bien des situations des logiciels commerciaux et coûteux comme Rational Rose1 ou Together2. Étant simple d'utilisation, nécessitant peu de ressources système, supportant UML 2, ce logiciel constitue une excellente option pour une familiarisation à la modélisation. Cependant, seule une version Windows est disponible.

FIGURE 4.1 – Logo de StarUml

4.2.2.2 Visual studio Code

C'est un éditeur de code source léger mais puissant qui s'exécute sur votre bureau. Il est disponible pour Windows, macOS et Linux. Il est livre avec un support integre pour JavaScript, TypeScript et NodeJs et dispose d'un riche écosystème d'extensions pour d'autres languages (tels que C++, Java, Python, PHP, Go ...) et des environnements d'exécution (tels que .NET et Unity) .Ainsi il fournit une structure complète du react js, et il contient un ensemble d'outils puissants auto-complètes du code.

FIGURE 4.2 – VSCode

4.2.2.3 Postman

C'est un outil multifonction pour un API Web, il permet de construire et d'executer des requettes HTTP, de les stocker dans un historique afin de pouvoir les rejouer, mais surtout de les organiser en Collections. Cette classification permet notamment de regrouper des requetes de facon fonctionnelle.Il fournit des services fiables pour tester les web services du REST API avant l'utiliser en ReactJs.

FIGURE 4.3 – Logo de Postman

4.2.2.4 Git

Lors de la réalisation de cette application, on a utilisé un logiciel de gestion des versions qui est un logiciel qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectué dessus. Il permet notamment de retrouver les différentes versions d'un lot de fichier connexes. Pour cela, on fait le choix sur Git qui est simplement un système de contrôle de version distribué gratuit et open source conçu pour tout gérer, des petits aux très grands projets, avec rapidité et efficacité.

FIGURE 4.4 – Logo git

Avec l'utilisation d'Git, on a fait deux étapes :

1. On a crée un dépôt local pour le projet, dans lequel Git stockera toutes les informations qui concerneront le suivis de l'état de vos fichiers.

2. Envoi de projet au GitHub pour mettre le projet à distant.

4.2.2.5 Github

GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git GitHub propose des comptes professionnels payants, ainsi que des comptes gratuits pour les projets de logiciels libres. Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités, la gestion de tâches et un wiki pour chaque projet. Github est l'outil que j'ai utilisé pour gérer les différents codes sources du développement de l'application. Ce logiciel m'a permis de garder à jour les différents codes sources sur la machine j'étais en train de travailler. De plus, cela me permettait de garder un backup de mes fichiers en cas de perte de données.

FIGURE 4.5 – Logo de github

4.2.2.6 overleaf

Overleaf est un éditeur LaTeX(pour éditer des documents) en ligne, collaboratif en temps réel.

en générale LaTeX est un langage de mise en page permettant de produire des documents d'une grande qualité typographique et rigoureusement homogènes dans leur présentation.

FIGURE 4.6 – Logo de overleaf

4.2.2.7 Logomakr

LogoMakr est un créateur de logo en ligne qui permet aux petites entreprises de créer des logos professionnels. Logomakr offre un accès gratuit à des outils de conception, à des centaines de types de polices et à plus d'un million de graphiques pouvant être utilisés pour créer des conceptions de logo.

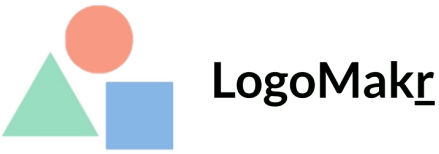

FIGURE 4.7 – Logo de logomakr

4.3 Environnement de développement

4.3.1 HTML

C'est un langage de balisage dont le rôle est de formaliser l'écriture d'un document avec des balises de formatage.

FIGURE 4.8 – Logo de HTML

4.3.2 CSS

C'est un langage informatique qui permet de mettre en forme le contenu des fichiers HTML ou XML.

FIGURE 4.9 – Logo de CSS

4.3.3 JavaScript

C'est un langage de programmation orienté objet principalement utilisé pour des pages HTML interactives.

FIGURE 4.10 – Logo de JavaScript

4.3.4 Les schémas NOSQL

MongoDB est une base de données NoSQL orientée document. Elle se distingue des bases de données relationnelles par sa flexibilité et ses performances.

4.3.5 MongoDBCompass

MongoDB Compass propose une interface graphique pour MongoDB, vous permettant d'envisager de nouvelles perspectives de gestion de vos bases de données NoSQL orientées documents. MongoDB Compass vous permet de garder la main sur vos données confidentielles ainsi que de vous connecter à un serveur MongoDB local.

4.3.6 Node JS

Node.js est une plate-forme construite sur l'environnement d'exécution JavaScript de Chrome pour créer facilement des applications réseau rapides et évolutives. Node.js utilise un modèle d'E/S non bloquant basé sur les événements qui le rend léger et efficace, parfait

FIGURE 4.11 – Logo de MongoDBCompass

pour les applications en temps réel gourmandes en données qui s'exécutent sur des appareils distribués.

FIGURE 4.12 – Logo de Node JS

4.3.7 ReactJS

React est une bibliothèque JavaScript utilisée pour construire des composants d'interface utilisateur réutilisables. Selon la documentation officielle de React, la définition est la suivante : - React est une bibliothèque permettant de construire des interfaces utilisateur composables. Elle encourage la création de composants d'interface utilisateur réutilisables, qui présentent des données qui évoluent dans le temps. Beaucoup de gens utilisent React comme le V de MVC (Modèle-Vue-Contrôleur). React fait abstraction du DOM, offrant un modèle de programmation plus simple et de meilleures performances. React peut également effectuer un rendu sur le serveur à l'aide de Node, et il peut alimenter des applications natives à l'aide de React Native. React met en œuvre un flux de données réactif unidirectionnel, ce qui réduit le nombre d'erreurs et facilite le raisonnement par rapport à la liaison de données traditionnelle.

4.3.8 Redux

C'est une bibliothèque open-source JavaScript de gestion d'état pour applications web.Elle est plus couramment utilisée avec des bibliothèques comme React ou Angular pour faire une gestion centralisée de notre application Web.

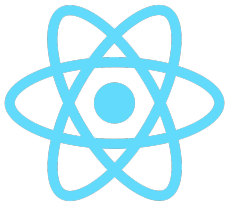

FIGURE 4.13 – Logo de ReactJS

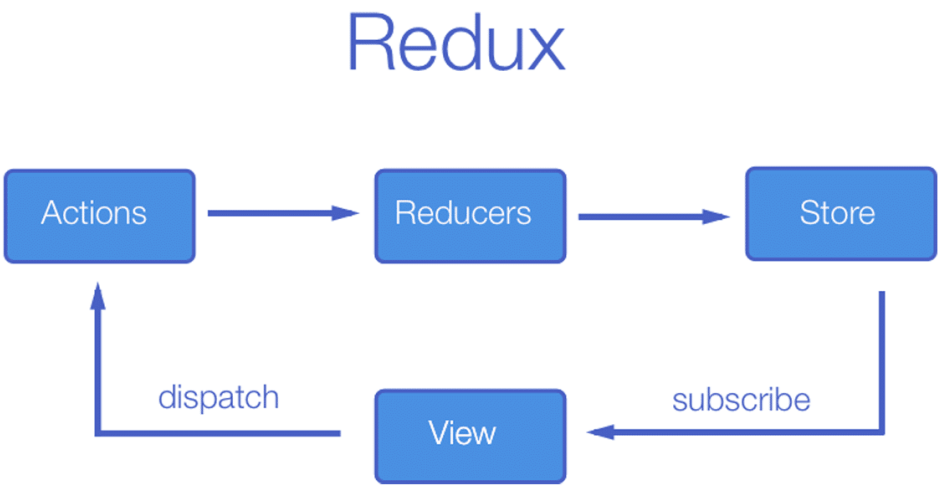

FIGURE 4.14 – Redux

FIGURE 4.15 – Logo de Redux

4.4 Réalisation du projet

4.5 Conclusion