# **Introduction générale**

De nos jours, l’internet a pris un grand élan dans notre vie. Avant elle était destinée qu’aux experts et les grandes entreprises. Aujourd’hui, elle est devenue indispensable et offre des services dans tous les domaines. Internet permet aux entreprises ou même les boutiques d’offrir ses services afin de répondre mieux, aux besoins de ses clients. Dans notre projet nous intéressons à la gestion de ces services.

En effet, dans notre vie quotidienne, lors de la recherche d’une personne professionnelle (ou compétente) dans un domaine nous rencontrons beaucoup de difficultés, par exemples : les informations, les contacts. Ce qui peut engendrer une perte de temps et d’argent. Pour pallier à ces problèmes, nous avons créés et développés une application web dynamique de la gestion d’offre des services. A travers cette application on regroupe plein de gens compétents qui peuvent partager leurs travaux et leurs contacts ainsi que toute personne pourra consulter ou bien acheter des services.

Notre choix de développer une application web, se justifie par le fait qu’elle est un moyen très utile et efficace pour exposer et offrir des services aux internautes via internet grâce à l’accessibilité qu’elle offre aux utilisateurs ou bien les clients indépendamment de leurs emplacements physiques, ce qui nous permettra de rapprocher les fournisseur et Clients.

Dans ce présent rapport, nous allons présenter les différentes étapes que nous avons suivies lors du développement de notre application web. Notamment, la démarche suivie, les diagrammes conçus, ainsi que les différents langages et outils utilisés. Et enfin nous clôturons ce travail avec une conclusion générale et quelques perspectives qui permettront d’améliorer notre solution.

# Chapitre 01

**Étude préalable**

* 1. **Introduction**

De nos jours les désires ne sont plus comme avant, aujourd'hui nous nous focalisons sur la réalisation des besoins efficacement en un temps minimal, c'est pour cela que nous optons pour les applications qui offrent des services en ligne sans se déplacer et sans perdre beaucoup de temps et d’argent.

Dans le cadre de notre travail, nous nous focalisons sur la gestion d’offre des services en ligne notamment sur la publication des services, l'objectif principale est de faciliter la tâche à l'internaute concernant ses information, ses contacts, ses achats, etc.

## **Problématique**

Lors de la recherche d’une personne professionnel (ou comptent) on confronte pas mal de difficultés, on cite quelques-unes

* la difficulté d avoir des informations et des contacts.
* 1. **Présentation de projet**

Le sujet principal de ce travail repose sur le développement d'une application web de gestion des services, qui regroupe un ensemble de services.

**Objectifs**

* Faciliter la publication et la recherche d'un service.
* Trouver son client ou son fournisseur plus proche possible.
* Offrir un large choix de services
* Gagner du temps et de d’argent.
* Etablir un contact entre le client et fournisseur.
* La recherche est simple et facile.

**1.3.2 Besoins fonctionnels**

Le système de gestion des services dont nous voulons se doter, doit être opérationnel, évolutif, convivial et offrant les options de gestion service nécessaires aux utilisateurs.

Pour cela, l'application à réaliser doit satisfaire les exigences de la totalité des utilisateurs. Nous présentons dans ce qui suit tout les besoins fonctionnels classés en espace.

Notre application est divisée en cinq grands espaces :

* Espace d'accueil : contient l'interface de démarrage de notre application, n'importe qui peut consulter cette page.
* Espace d'inscription : c'est l'interface qui permet à un utilisateur de créer son compte avec ses informations personnelles, un mot de passe et un identifiant. Le formulaire contient des champs qui concernent les informations que chaque utilisateur doit remplir.
* Espace de publication des services : après l'authentification, l'utilisateur peut consulter, acheter les déférent services en cliquant la page demande à l'utilisateur de POSTER UN BESOIN  par exemple : plombier et électricité, charpentier et autres.
* Espace de gestion du profil : le lien vers cette page ne s'affiche qu'après l'authentification de l'utilisateur, où il peut publier la liste de ses services, ou les informations concernant son compte.
* Espace d'administration : c'est uniquement l'administrateur de l'application qui peut accéder à cet espace, où il peut consulter et gérer la liste des utilisateurs, les fournisseurs et la liste des services publiées sur l'application.

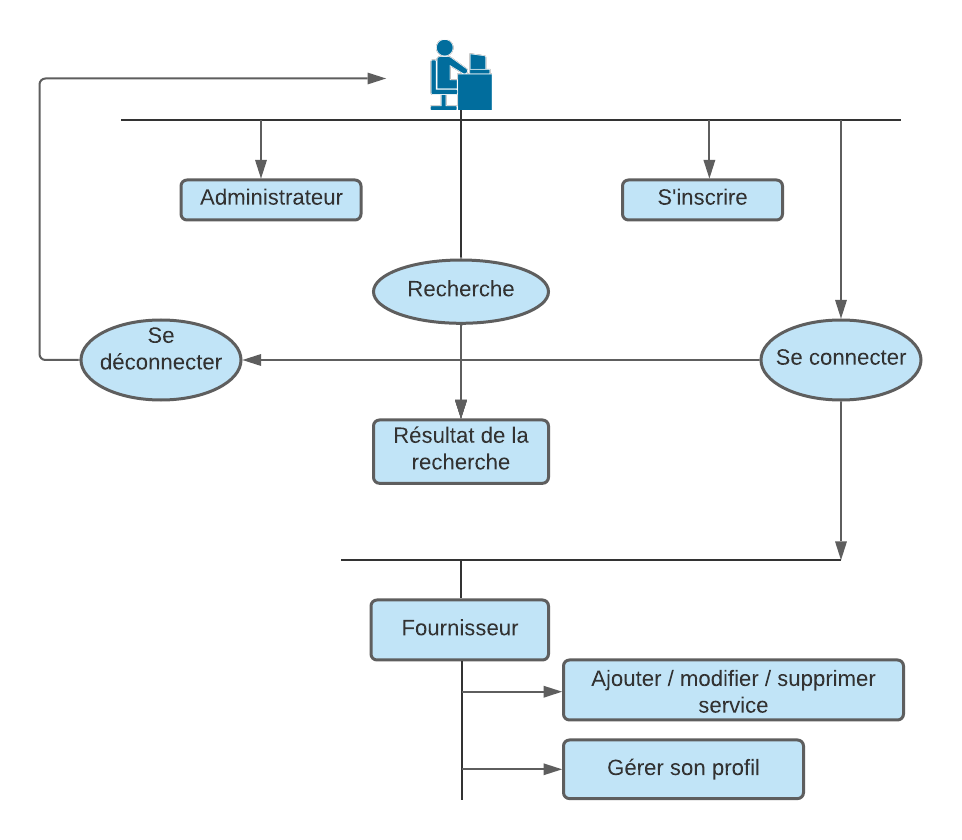
**1.2.3 Besoins non fonctionnels**

Disposer d'une application web fonctionnelle et attrayante contribue nettement à la réussite d'une grande majorité de la société d'accueil. Afin de se distinguer de la concurrence, l'application doit être esthétiquement attrayante et sa navigation doit être fluide et efficace.

* Le système doit avoir une interface conviviale permettant un dialogue simple entre l'utilisateur et l'application.
* Le système doit être fiable et sécurisé
* Le menu doit être bien organisé.
* Les informations figurant dans l'application doivent être claires et précises.
* L'application doit fonctionner sur différents systèmes d'exploitation.

**1.3.4 Vue globale de l'application à réaliser**

L’application à réaliser permettra au demandeur de service de confier leurs besoins à des freelances algériens qualifier.



**Figure 1.1 Vue globale de l'application à réaliser**

**1.3.5 Le logo**

**Figure 1.2 Logo de l'application**

**1.3.6 Analyse concurrentielle**

Toutes plateformes, destinées services, offrent la possibilité aux utilisateurs de déposer des services, de vendre ou d'acheter pratiquement tout ce qu'ils souhaitent dont chacune a ses propres avantages par rapport aux autres.

Prenant l'exemple du site algérien freehali :

Freehali est donc une plateforme de services de freelances qui permet d’un côté aux algériens ayant des compétences de pouvoir les monétiser, et de l’autre, permettre aux entrepreneurs algériens d’accéder en toute confiance au savoir-faire de ces compétence, sans se ruiner. Et cela en leur donnant le pouvoir de choisir et de négocier avec plusieurs freelances gratuitement, simultanément, et sans engagement. Plus de 1000 services sont postés par freehali. Nous avons présenté dans la figure ci-dessous un aperçu du site.

****

**Figure 1.3 Capture d'écran du site freehali**

Le tableau suivant résume les critères essentiels que nous avons pu cerner pendant notre analyse

|  |  |
| --- | --- |
| Nom du site | freehali.com |
| Adresse du site | https://www.freehali.com/ |
| Date de l'analyse | 25 mai 2021 |
| Technologie utilisé | PHP, Jquery. |
| Création | 2016 |
| Contenu | Abonnements professionnels, services |
| Points positifs | \_ La plus grande plate-forme des services en Algérie.  \_ Les pages web du site sont RESPONSIVE DESIGN c'est-à-dire qu'elles s'adaptent aux différents types d'équipements et périphériques.  \_ monétiser les compétences des gens.  \_ Permet d'acheter et de vendre des services dans des dizaines de catégories différentes.  \_  choisir et de négocier avec plusieurs freelances gratuitement  \_ S'engage à rester gratuit aux particuliers. |
| Points négatifs | * Ne contient pas des services professionnels. |

**Table 1.1 Résumé de l'analyse du site freehali**

**1.3.7 Discussion sur l'application par rapport à l'analyse concurrentielle**

L'analyse que nous avons effectuée nous a permis de mieux définir les besoins et d'avoir une meilleure vue et représentation de notre application.

Le nom sur le site sera ……….et son adresse internet sera https ://www………...com.

Notre application web s'adresse à un large public souhaitant publier des services.

Toute personne utilisant et possédant un Smartphone (ou ordinateur portable) et une connexion internet peut être un potentiel utilisateur du système.

**1.4 Conclusion**

À travers l'étude préalable, nous avons défini les besoins et les fonctionnalités qu'il va falloir implémenter à ces besoins, en effet la réalisation d'une application web nécessite une étude conceptuelle qui donnera une vision globale du futur système à implémenter et le chapitre suivant sera consacré à la conception.

**Chapitre 2**

**Étude conceptuelle**

**2.1 Introduction**

La phase de conception est la première étape dans la réalisation d'un projet, elle doit d'écrire de manière non ambiguë le fonctionnement futur du système, afin d'en faciliter la réalisation. Pour cela, différentes méthodes existent permettant de formaliser les étapes préliminaires du développement.

Ainsi dans ce chapitre, nous allons déterminer un premier découpage de notre application, et établir les différentes étapes de conception, nous définirons les relations qui peuvent exister

entre les cas d'utilisation, et nous entamerons l'élaboration de certains diagrammes définis dans le langage UML(Unified Modeling Language), comme le diagramme de cas d'utilisation qui nous aidera dans la spécification des besoins de notre système, le diagramme de séquence qui nous donnera le scénario des actions et interactions entre les acteurs et notre système, et le diagramme de classe qui définira l'architecture statique de notre système.

**2.2 Le choix de la méthode**

Il existe plusieurs méthodes de développement logiciel construites sur UML comme : UP, RUP, TTUP, UPagile, XP, 2TUP, etc.

Parmi ses méthodes notre choix est basé sur la méthode UP (Unified Process).

**2.2.1 Définition d'un processus**

Un processus définit une séquence d'étapes, en partie ordonnées, qui concourent à l'obtention d'un système logiciel ou à l'évolution d'un système existant.

L'objet d'un processus de développement est de produire des logiciels de qualité qui répondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles.

**2.2.1.1 Définition du processus unifié**

Le processus unifié est un processus de développement logiciel itératif, centré sur l'architecture, piloté par des cas d'utilisation et orienté vers la diminution des risques. C'est un patron de processus pouvant être adapté à une large classe de systèmes logiciels, à différents domaines d'application, à différents types d'entreprises, à différents niveaux de compétences et à différents tailles de l'entreprise.

L'objectif d'un processus unifié est de maitriser la complexité des projets informatiques en diminuant les risques.

UP répond aux préoccupations suivantes :

* QUI participe au projet ?
* Quoi, qu'est-ce qui est produit durant le projet ?
* COMMENT doit-il être réalisé ?
* QUAND est réalisé chaque livrable ?

**2.2.1.2 Les caractéristiques du processus unifié**

* UP est itératif et incrémental

Le projet est découpé en itérations ou étapes de courte durée qui permettent de mieux suivre l'avancement globale. À la fin de chaque itération une partie exécutable du système finale est produite, de façon incrémentale (par ajout).

La figure ci-dessous illustre l'interaction d'UP :



**Figure 2.1 Déroulement du processus unifié**

* UP est centré sur l'architecture

Tout système complexe doit être décomposé en partie modulaire afin d'en faciliter la maintenance et l'évolution. Cette architecture (fonctionnelle, logique, matérielle, etc) doit être modéliser en UML, et pas seulement documentée en texte.

* UP est guidé par les cas d'utilisation d'UML

Le but principal d'un système informatique est de satisfaire les besoins du client. Le processus

de développement sera donc accès sur l'utilisateur. Les cas d'utilisation permettent d'illustrer ces besoins. Ils détectent puis décrivent les besoins fonctionnels et leur ensemble constitue le modèle de cas d'utilisation qui dicte les fonctionnalités complètes du système.

* UP est piloté par les risques

Les risques majeurs du projet doivent être identifiés au plus tôt mais surtout levés le plus rapidement. Les mesures à prendre dans ce cadre déterminant l'ordre des interactions.

**2.2.1.3 Cycle de vie du processus unifié**

L'objectif d'un processus unifié est de maitriser la complexité des projets informatiques en diminuant les risques. UP est un ensemble de principes génériques adapté en fonctions des spécificités des projets :

* L'architecture bidirectionnelle : UP gère le processus de développement par deux axes

(Voir la figure 2.2).

* L'axe vertical : représente les principaux enchaînements d'activités, qui regroupent les activités selon leur nature. Cette dimension rend compte l'aspect statique du processus qui s'exprime en termes de composants, de processus, d'activités, d'enchainements, d'artefacts et de travailleurs.
* L'axe horizontal : représente le temps et montre le déroulement du cycle de vie du processus ; cette dimension rend compte de l'aspect dynamique du processus qui s'exprime en terme de cycles, de phases, d'interactions et de jalons.



**Figure 2.2 L'architecture bidirectionnelle**

Pour mener efficacement un tel cycle, les développeurs ont besoins de toutes les représentations du produit logiciel, c'est-à-dire :

* Un modèle de cas d'utilisation.
* Un modèle d'analyse : détaillant les cas d'utilisation.
* Un modèle de conception : finissant la structure statique du système sous forme de sous systèmes, de classes et d'interfaces.
* Un modèle d'implémentation : intégrant les composants.
* Un modèle de déploiement : définissant les nœuds physiques des ordinateurs.
* Un modèle de test : décrivant les cas de test vérifiant les cas d'utilisation.
* Une représentation de l'architecture

**2.2.1.4 Les activités**

* **Expression des besoins**

L'expression des besoins comme son nom l'indique, permet de définir les différents besoins :

- Inventorier les besoins principaux et fournir une liste de leurs fonctions.

- Appréhender les besoins non fonctionnels (techniques) et livrer une liste des exigences.

- Le modèle de cas d'utilisation présente le système du point de vue de l'utilisateur et représente sous forme de cas d'utilisation et d'acteur, les besoins du client.

* **Analyse**

L'objectif de l'analyse est d'accéder à une compréhension des besoins et des exigences du client. Il s'agit de livrer des spécifications pour permettre de choisir la conception de la solution.

Un modèle d'analyse livre une spécification complète des besoins issus des cas d'utilisation et les structure sous une forme qui facilite la compréhension (scénarios), la préparation (définition de l'architecture), la modification et la maintenance du futur système. Il s'écrit dans le langage des développeurs et peut être considéré comme une première ébauche du modèle de conception.

* **Conception**

La conception permet d'acquérir une compréhension approfondie des contraintes liées au langage de programmation, à l'utilisation des composants et au système d'exploitation. Elle détermine les principales interfaces et les transcrit à l'aide d'une notation commune.

Elle constitue un point de départ à l'implémentation :

- elle décompose le travail d'implémentation en sous-système.

- elle crée une abstraction transparente de l'implémentation.

* **Implémentation**

L'implémentation est le résultat de la conception pour implémenter le système sous formes de composants, c'est-à-dire, de code source, de scripts, de binaires, d'exécutable et d'autres éléments du même type.

Les objectifs principaux de l'implémentation sont de planifier les intégrations des composants pour chaque itération, et de produire les classes et les sous-systèmes sous formes de codes sources.

* **Test**

Les tests permettent de vérifier des résultats de l'implémentation en testant la construction. Pour mener à bien ces tests, il faut les planifier pour chaque itération, les implémenter en créant des cas de tests, effectuer ces tests et prendre en compte le résultat de chacun



**Figure 2.3 Illustration des différentes activités**

**2.2.1.5 Les phases**

* **Analyse des besoins**

L'analyse des besoins donne une vue du projet sous forme de produit fini. Cette phase porte essentiellement sur les besoins principaux (du point de vue de l'utilisateur), l'architecture générale du système, les risques majeurs, les délais et les coûts.

* **Elaboration**

L'élaboration reprend les éléments de la phase d'analyse des besoins et les précise pour arriver à une spécification détaillée de la solution à mettre en œuvre.

L'élaboration permet de préciser la plupart des cas d'utilisation, de concevoir l'architecture du système et surtout de déterminer l'architecture de référence.

* **Construction**

La construction est le moment où l'on construit le produit. L'architecture de référence se métamorphose en produit complet. Le produit contient tous les cas d'utilisation que les chefs de projet, en accord avec les utilisateurs ont décidé de mettre au point pour cette version.

* **Transition**

Le produit est en version bêta. Un groupe d'utilisateurs essaye le produit et détecte les anomalies et défauts. Cette phase suppose des activités comme la formation des utilisateurs clients, la mise en œuvre d'un service d'assistance et la correction des anomalies constatées.

Tout simplement la phase de transition permet de faire passer le système informatique des mains des développeurs à celles des utilisateurs finaux.



**Figure 2.4 Résumé du processus UP**

**2.3 Analyse conceptuelle**

UML se définit comme un langage de graphique destiné à la représentation d'un système donné selon un point de vue particulier.

**2.3.1 Le langage UML**

C'est un langage de modélisation graphique à base de pictogrammes, conçu pour représenter et spécifier les artefacts de systèmes logiciels, de plus il est destiné à comprendre et décrire des besoins spécifiés et documentés des systèmes, esquissé des architectures logicielles, concevoir des solutions et communiquer des points de vue, comme il peut être appliqué à toutes sortes de systèmes ne se limitant pas au domaine informatique.

UML résulte de l'unification de techniques ayant fait leurs preuves pour l'analyse et la conception de grands logiciels et de systèmes complexes.

* **UML est une norme**

Il est nécessaire qu'une méthode objet soit définie de manière rigoureuse et unique afin de lever les ambiguïtés. De nombreuses méthodes objet ont été définies, mais aucune n'a su s'imposer en raison du manque de standardisation. C'est pourquoi l'ensemble des acteurs du monde informatique a fondé en 1989 l'OMG (Object Management Group), une organisation à but non lucratif, dont le but est de mettre au point des standards garantissant la compatibilité entre des applications programmées à l'aide de langages objet et fonctionnant sur des réseaux hétérogènes (de différents types). Et depuis 1997, UML est devenue une norme de l'OMG, ce qui lui a permis de s'imposer entant que méthode de développement objet et être reconnue et utilisée par de nombreuses entreprises.

* **UML est un langage de modélisation objet**

UML comble une lacune importante des technologies objet, il permet d'exprimer, d'élaborer et de modéliser au sens de la théorie des langages, de ce fait il contient les éléments constitutifs de ces derniers : concepts, une syntaxe et une sémantique.

* **UML décrit un méta modèle**

La puissance et l'intérêt d'UML est qu'il normalise la sémantique des concepts qu'il véhicule, il repose sur un méta modèle pour permettre à n'importe qui de déchiffrer son intention de manière non équivoque, il est donc primordial de s'accorder sur la sémantique des éléments de modélisation, bien avant de s'intéresser à la manière de les présenter.

**2.3.1.1 Points forts d'UML**

Il permet ainsi :

* Un gain de précision.
* Un gage de stabilité.
* L'utilisation d'outils.
* Il cadre l'analyse et facilite la compréhension de représentations abstraites complexes.
* Son caractère polyvalent et sa souplesse en font un langage universel.

**2.3.1.2 Points faibles d'UML**

* La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation.
* L'intégration d'UML dans un processus n'est pas triviale, et améliorer un processus est une tâche complexe et longue

**2.3.1.3 Les diagrammes**

**A) Définition d'un diagramme**

Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle. Chaque type de diagramme UML possède une structure et véhicule une sémantique précise.

**B) Les différents types de diagrammes**

Pour notre projet, nous présenterons trois diagrammes d'UML : cas d'utilisation, de séquence et de classe.

* **Diagramme de classe (class diagram)**

Les diagrammes de classes expriment de manière générale la structure statique d'un système, en termes de classes et de relations entre ses classes. Outre les classes, ils représentent un ensemble d'interfaces et de paquetages, ainsi que leurs relations.

Les diagrammes de classes contiennent généralement les éléments suivant :

* **Les classes** : Une classe est la description d'un ensemble d'objet qui partage les mêmes attributs, les mêmes opérations, les mêmes relations et la même sémantique. Une classe est symbolisée par un rectangle.
* **Attribut** : Un attribut est une propriété nommée d'une classe qui décrit un ensemble de valeurs que les instances de cette propriété peuvent prendre. Une classe peut ne pas avoir, comme elle peut avoir un ou plusieurs attributs.
* **Opérations** : Une opération est une abstraction de ce que peut réaliser un objet et qui est réalisable par tous les objets de la classe. Une classe peut ne pas avoir comme elle peut avoir plusieurs opérations.
* **Les relations d'association, d'agrégation et de composition :**
* Une association représente une relation sémantique durable entre deux classes.
* Une agrégation est un cas particulier d'association non symétrique exprimant une relation de contenance.
* Une composition est une agrégation plus forte.
* **Diagramme de cas d'utilisation (use case diagram)**

Les diagrammes de cas d'utilisation représentent un ensemble de cas d'utilisation, d'acteurs et leurs relations. Ils représentent la vue statique des cas d'utilisation d'un système et sont particulièrement importants dans l'organisation et la modélisation des comportements d'un système.

* **Les cas d'utilisation** : Les cas d'utilisation décrivent, sous la forme d'actions et de réactions, le comportement, ou tout simplement ce que fait le système du point de vue de l'utilisateur, encore appelé acteur. On recense, de la sorte, l'ensemble des fonctionnalités d'un système en examinant les besoins fonctionnels de chaque acteur.
* **Les acteurs** : Un acteur représente un ensemble cohérent de rôles joués par les utilisateurs des cas d'utilisation en interaction avec ces cas d'utilisation. En règle générale, un acteur représente un rôle qu'un homme, une machine ou même un autre système joue avec le système. Il existe 4 grandes catégories d'acteurs :
* Les acteurs principaux : cette catégorie regroupe les personnes qui utilisent les fonctions principales du système.
* Les acteurs secondaires : cette catégorie regroupe les personnes qui effectuent des tâches administratives ou de maintenance.
* Le matériel externe : cette catégorie regroupe les dispositifs matériels autres que les ordinateurs comme les périphériques.
* Les autres systèmes : cette catégorie regroupe les systèmes avec lesquels le système interagit.
* **Les relations entre les cas d'utilisation** : UML définit trois types de relations standardisées entre cas d'utilisation, détaillées ci-après :
* La relation d'inclusion : formalisée par le mot clé «  include », le cas d'utilisation de base en incorpore explicitement un autre de façon obligatoire.
* La relation d'extension : formalisée par le mot clé «  extend », le cas d'utilisation de base en incorpore explicitement un autre, de façon optionnelle.
* La relation de généralisation ou 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.
* **Diagramme de séquence (sequence diagram)**

Un diagramme de séquence met en évidence le classement des messages par ordre chronologique. On forme un diagramme de séquence en plaçant d'abord les objets qui participent à l'interaction en haut du diagramme sur l'axe des abscisses. En générale, on place l'objet qui débute l'interaction à gauche, puis on continue en progressant vers la droite, les objets les plus subordonnés étant tout à fait à droite. On place ensuite les messages envoyés et reçus par ces objets le long de l'axe des ordonnées, par ordre chronologique, du haut vers le bas. Cela donne au lecteur une indication visuelle claire du flot de contrôle dans le temps.

En générale, les diagrammes de séquence contiennent :

* **L'objet** : est une manifestation concrète d'une abstraction à laquelle on peut appliquer un ensemble d'opérations et qui possède un état capable de mémoriser les effets de ces opérations. On représente un objet en soulignant son nom.
* **Le lien** : est une liaison sémantique entre objets, en générale, il s'agit d'une instance d'une association. Chaque fois qu'une classe est reliée à une autre par une association, il peut y avoir un lien entre les instances des deux classes, et chaque fois qu'un lien existe entre deux objets, le premier objet peut envoyer un message au deuxième.
* **Le message** : est la spécification d'une communication entre objets, qui transporte des informations et qui s'affiche dans le but de déclencher une activité. La réception d'une instance de message peut être considérée comme une instance d'un événement

**2.3.2 Identification des acteurs**

Après avoir recensé l'ensemble des besoins (cahier des charges), nous avons distingué les différents acteurs suivants :

|  |  |  |
| --- | --- | --- |
| **Acteur** | **Codification** | **Rôle** |
| Administrateur | Admin | Gérer les utilisateurs, ajouter ou supprimer des services. |
| Les inscrits  . | Client | consulter un service,  rechercher une annonce,  acheter un service.  . |
| Les inscrits | Fournisseur | Publier un service, consulter un service, rechercher un service, modifier son compte, supprimer ou modifier ses services. |
| visiteur anonyme | Internaute | Consulter le site, remplir le formulaire  de préinscription, se connecter et voir  des services. |

**Table 2.1 Identification des acteurs.**

**2.3.3 Diagrammes des cas d'utilisations**

Les cas d'utilisation sont des représentations fonctionnelles du système. Ils permettent de modéliser les attentes des utilisateurs dans le but d'effectuer une bonne délimitation du système et également la compréhension de son fonctionnement.

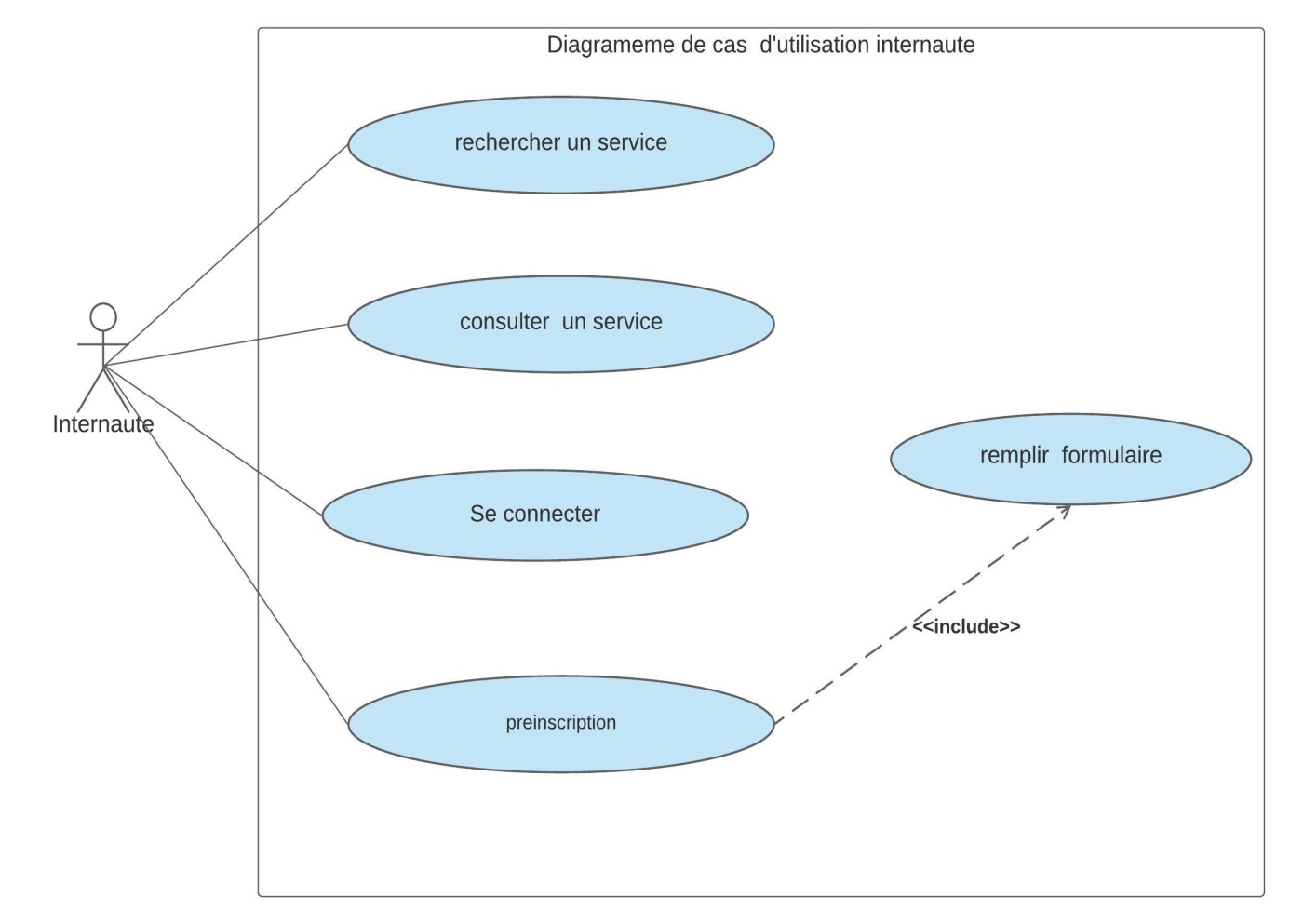
À partir des besoins exprimés, nous avons déduit les cas d'utilisation tel que résumés dans le tableau ci-dessous :

|  |  |  |
| --- | --- | --- |
| **N°** | **Cas d'utilisation** | **Acteur** |
| 1 | Préinscription | Internaute |
| 2 | Se connecter |
| 3 | Rechercher un service | Internaute et Client |
| 4 | Consulter des services |
| 5 | Publier un service | Fournisseur |
| 6 | Modifier son service |
| 7 | Supprimer son service |
| 8 | Modifier son compte |
| 9 | Ajouter des services ou utilisateurs | Administrateur |
| 10 | Supprimer des services ou utilisateurs |

**Table 2.2 Les cas d'utilisation pour chaque acteur.**

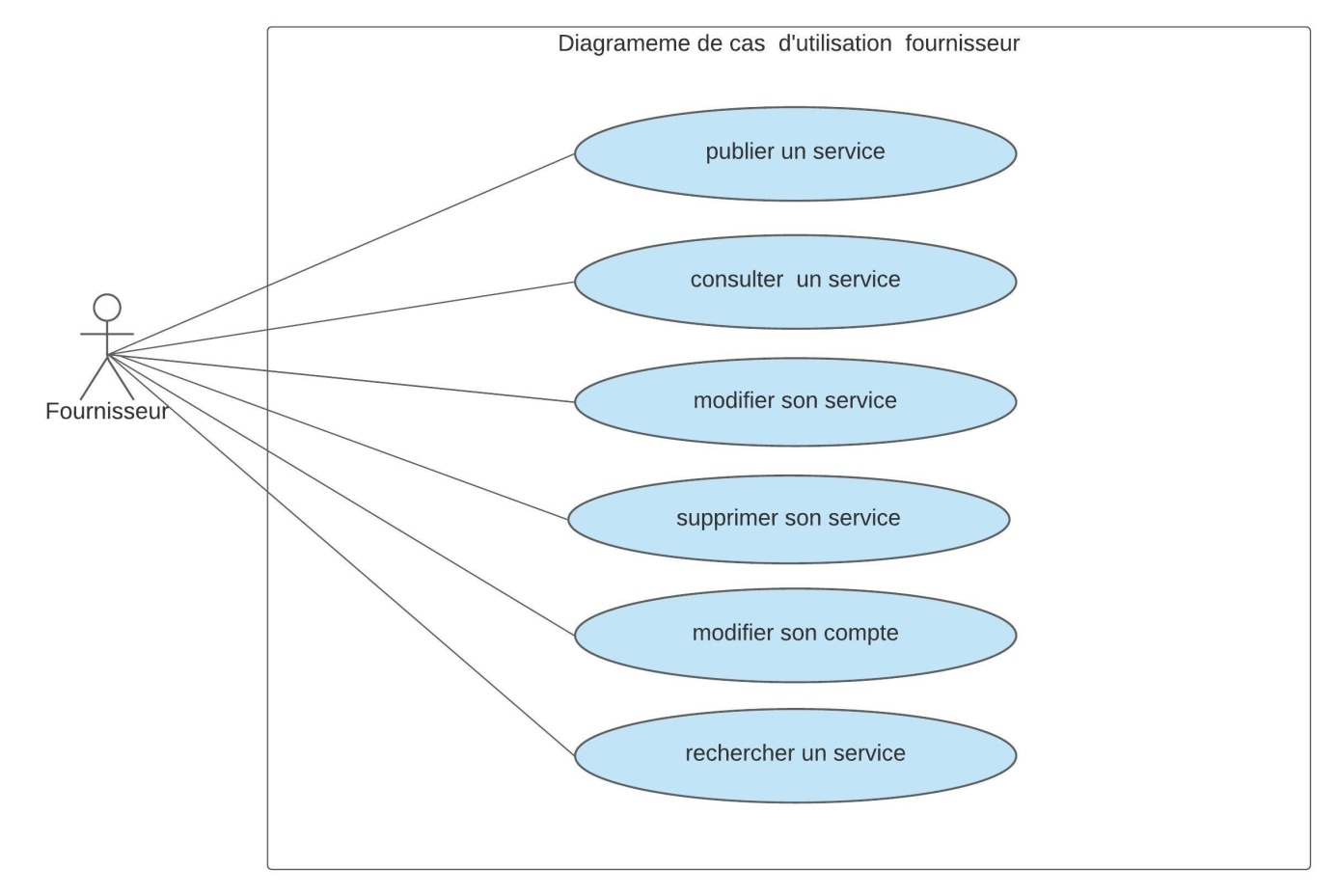
* **Spécification des cas d'utilisation**

Diagramme de cas d'utilisation « Internaute »

****

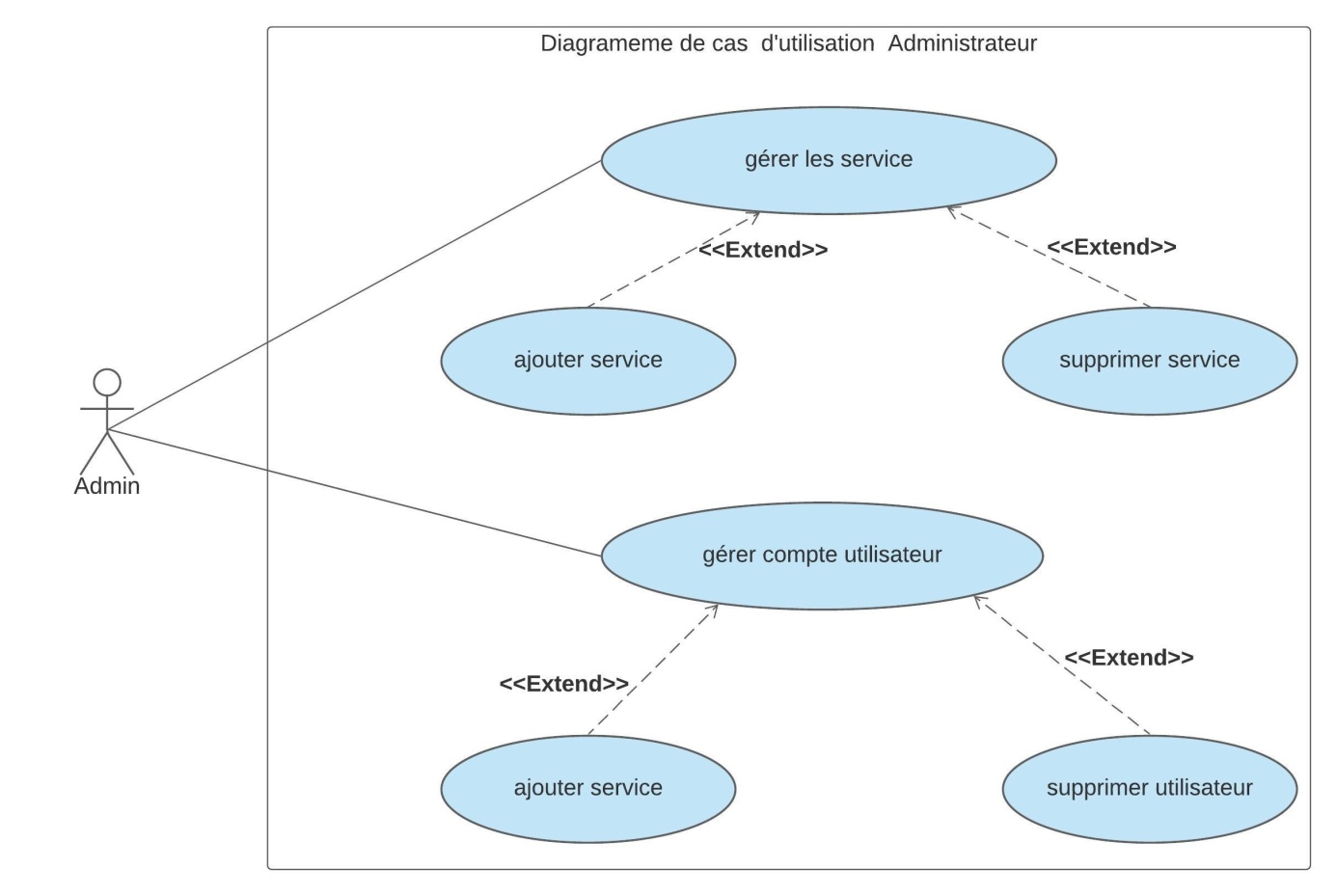
**Figure 2.5 Diagramme de cas d'utilisation «Internaute»**

Diagramme de cas d'utilisation « Fournisseur »



**Figure 2.6 Diagramme de cas d'utilisation « Fournisseur »**

Diagramme de cas d'utilisation « Administrateur »



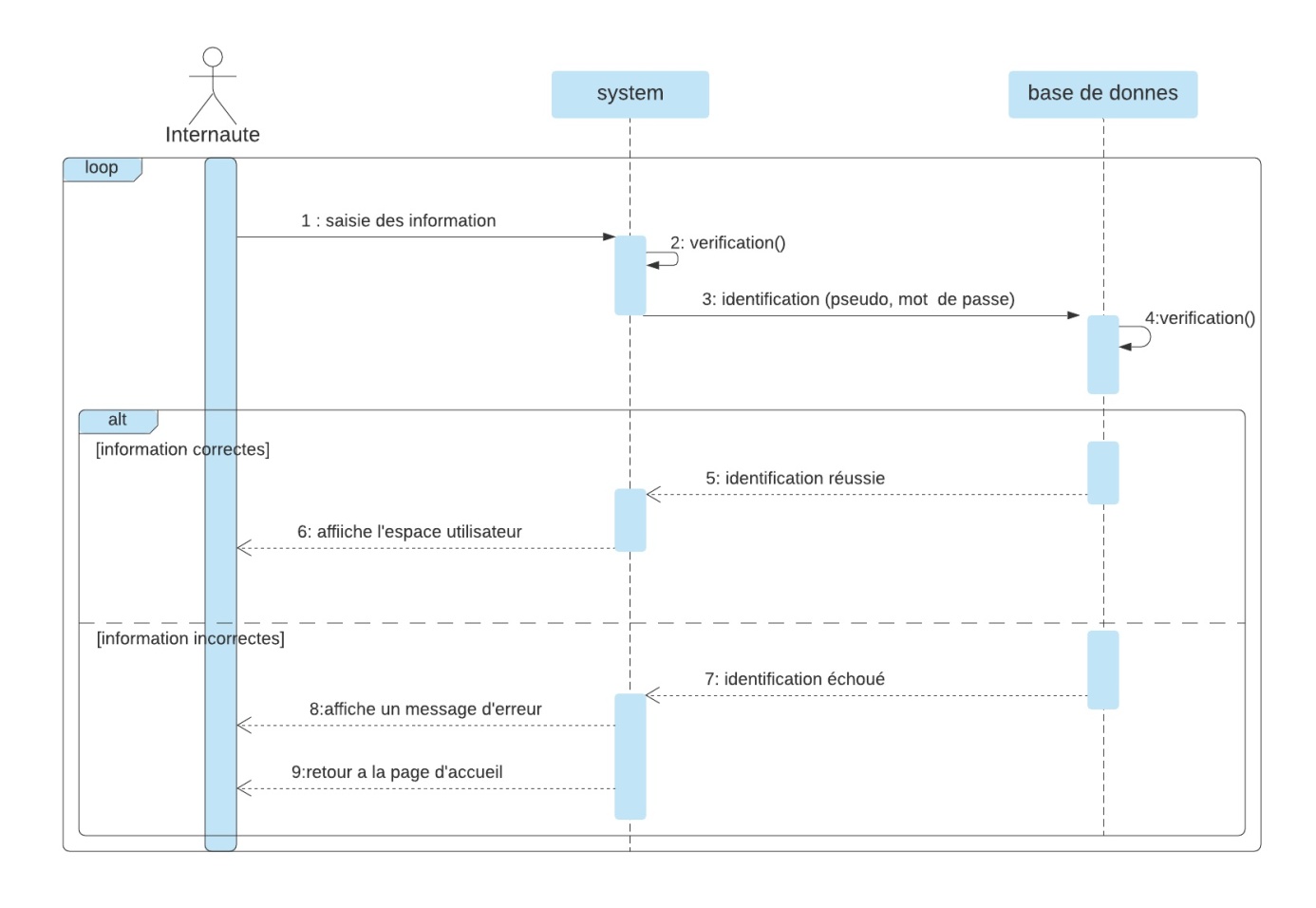
**Figure 2.7 Diagramme de cas d'utilisation « Administrateur »**

**2.3.4 Modélisation des diagrammes de séquence**

Un diagramme de séquences est un diagramme d'interaction qui expose en détail la façon dont les opérations sont effectuées, avec une représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation UML.

* **Se connecter**

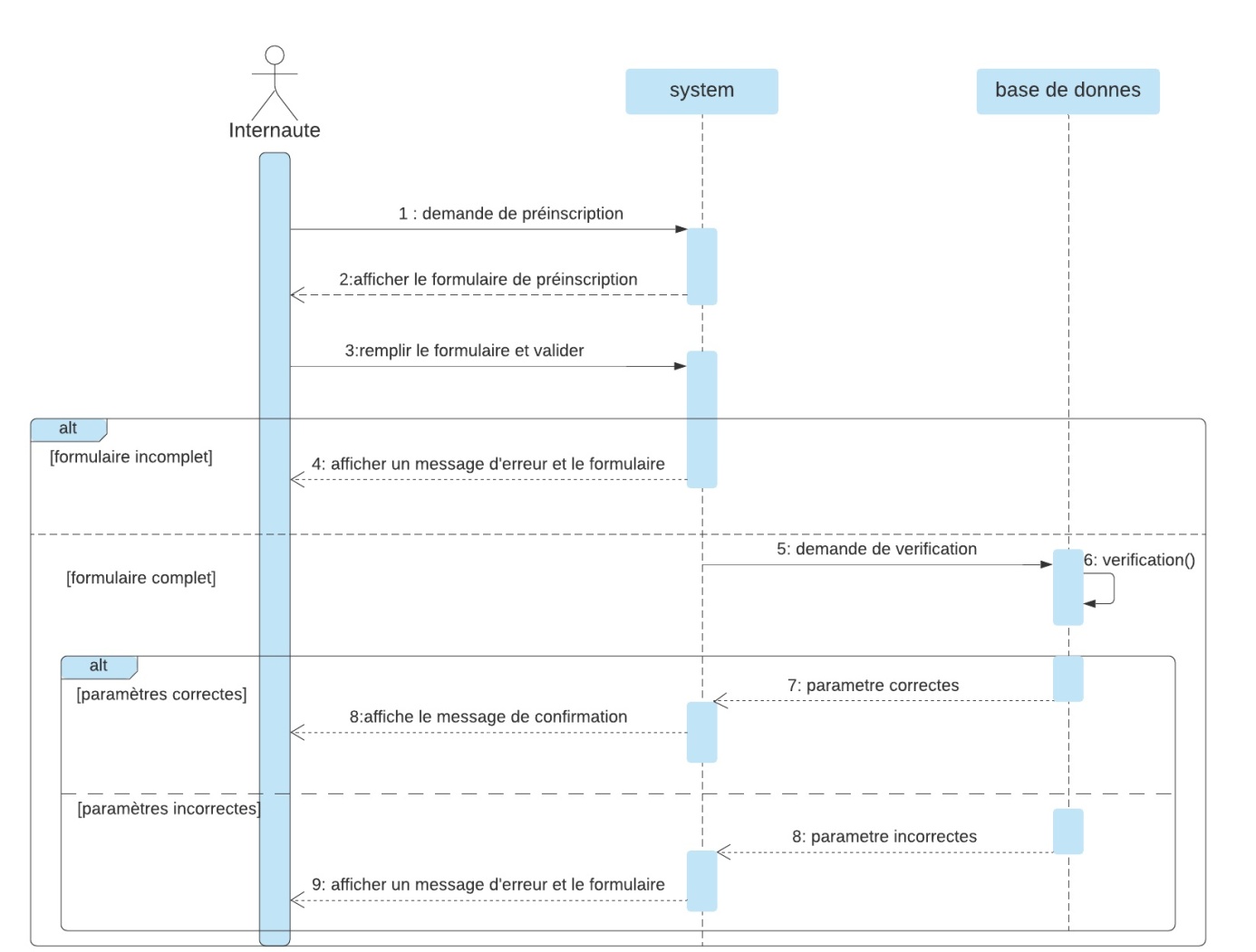
Le diagramme de séquence suivant illustre les interactions nécessaires pour la connexion d'un internaute. Après l'inscription utilisateur peut se connecter à l'application, pour cela il n'a qu'à cliquer sur le bouton de connexion en saisissant son identifiant et son mot de passe, le système vérifie ses champs auprès de la base de données, si ses informations sont correctes alors il aura accès à l'application pour gérer les services comme par exemple ajouter un service sinon un message d'erreur sera affiché et le système lui redemande à nouveau de saisir ses coordonnées.



**Figure 2.8 Diagramme de séquence du cas d'utilisation « Se connecter »**

* Préinscription

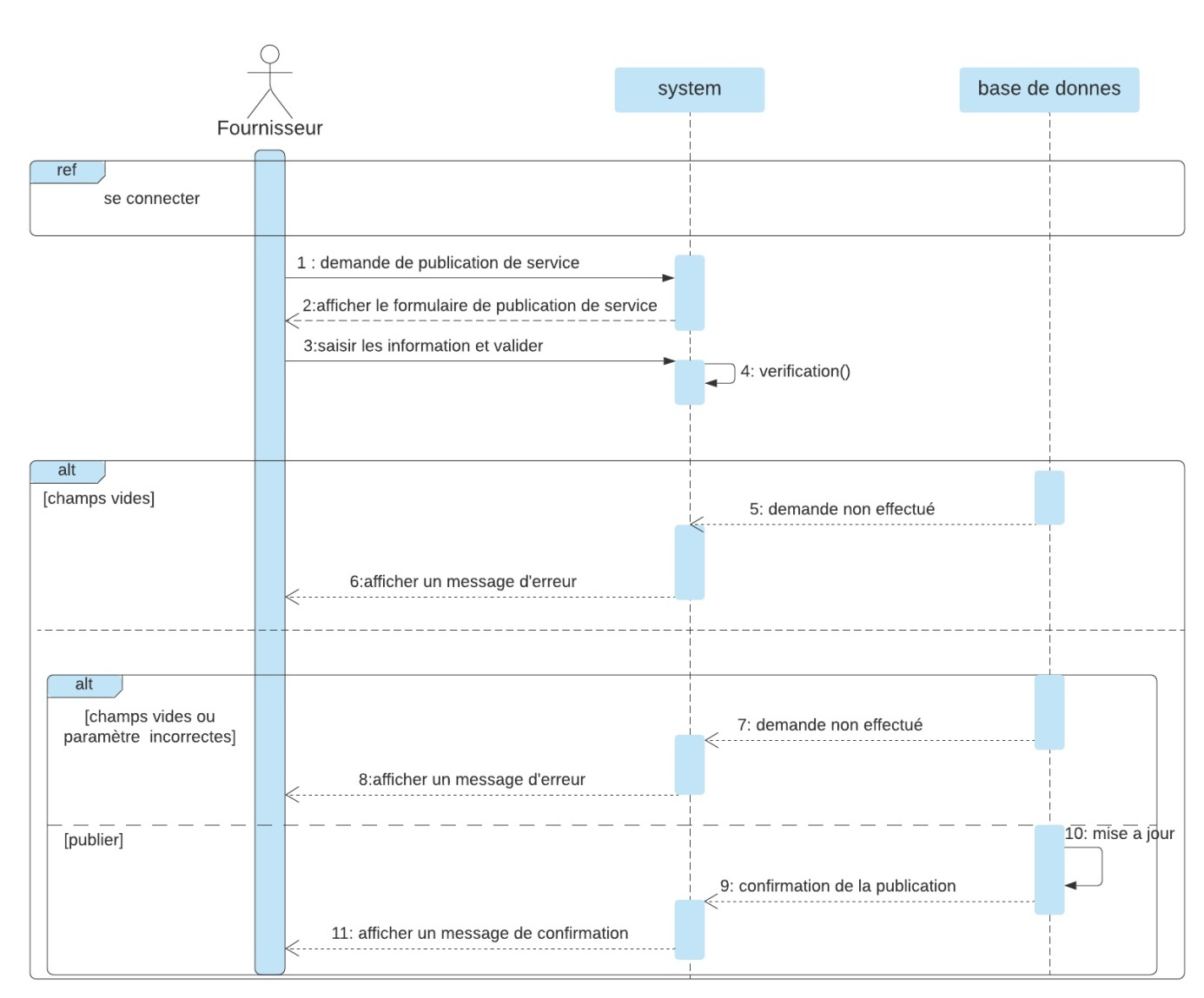
Le diagramme de séquence suivant illustre les interactions nécessaires pour l'inscription d'un utilisateur. En effet l'internaute va demander l'inscription auprès du système, ce dernier lui affiche un formulaire de préinscription, l'internaute remplit le formulaire de préinscription et le valide. Les informations saisies par l'utilisateur sont traitées, c'est-à-dire le système va vérifier si le formulaire est complet et si les données sont cohérentes, dans le cas contraire un message d'erreur sera affiché.



**Figure 2.9 Diagramme de séquence du cas d'utilisation «  préinscription»**

* **Publier un service**

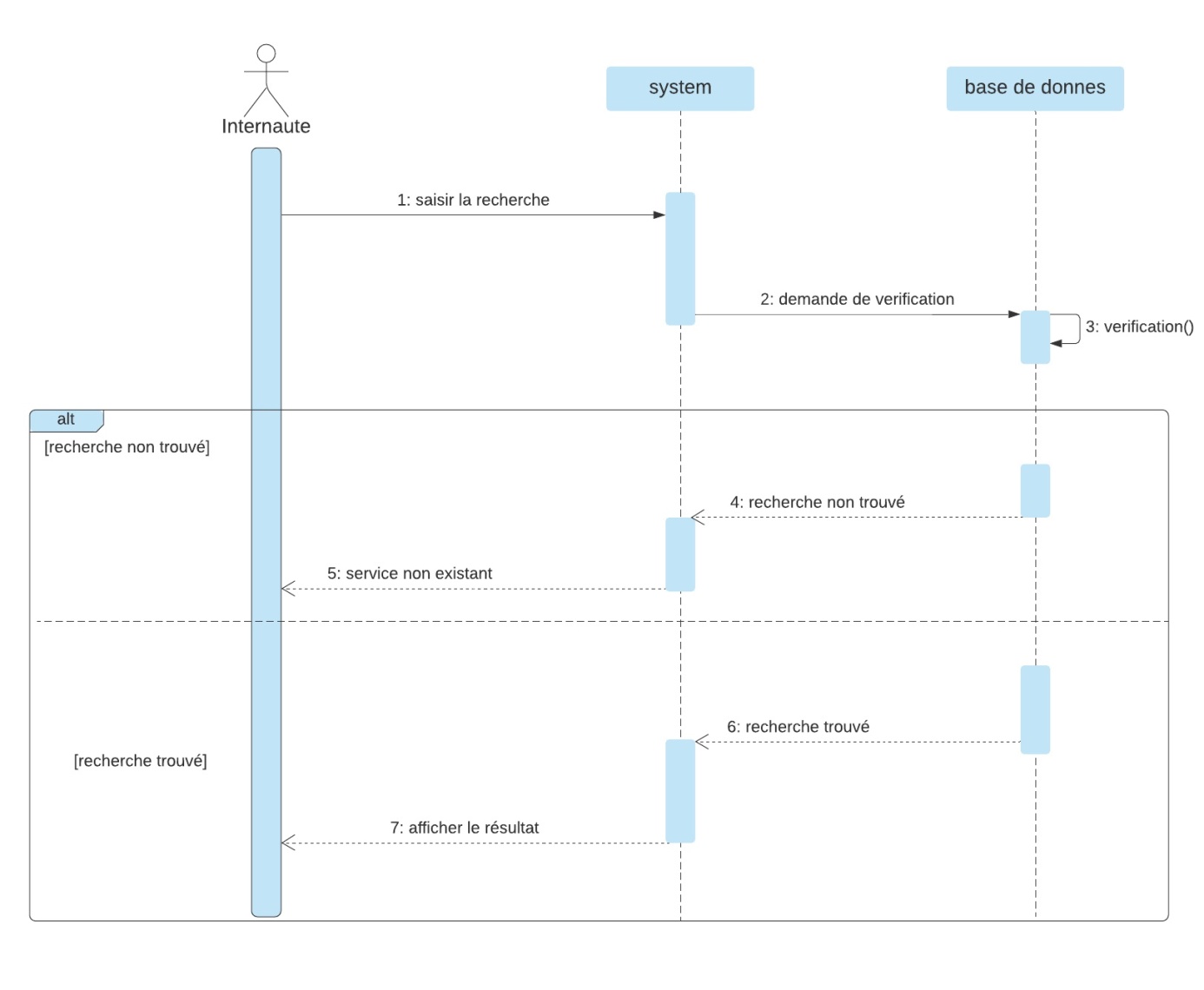
Le diagramme de séquence suivant illustre les interactions nécessaires pour la publication d'un service. Après l'authentification, le Fournisseur peut déposer un service tout en remplissant un formulaire de dépôt d'un service. Si le Fournisseur oublie de remplir l'un des champs ou remplit un champ dans une case inapproprié le service ne sera pas déposée.



**Figure 2.10 Diagramme de séquence du cas d'utilisation « publier un service»**

* **Chercher un service**

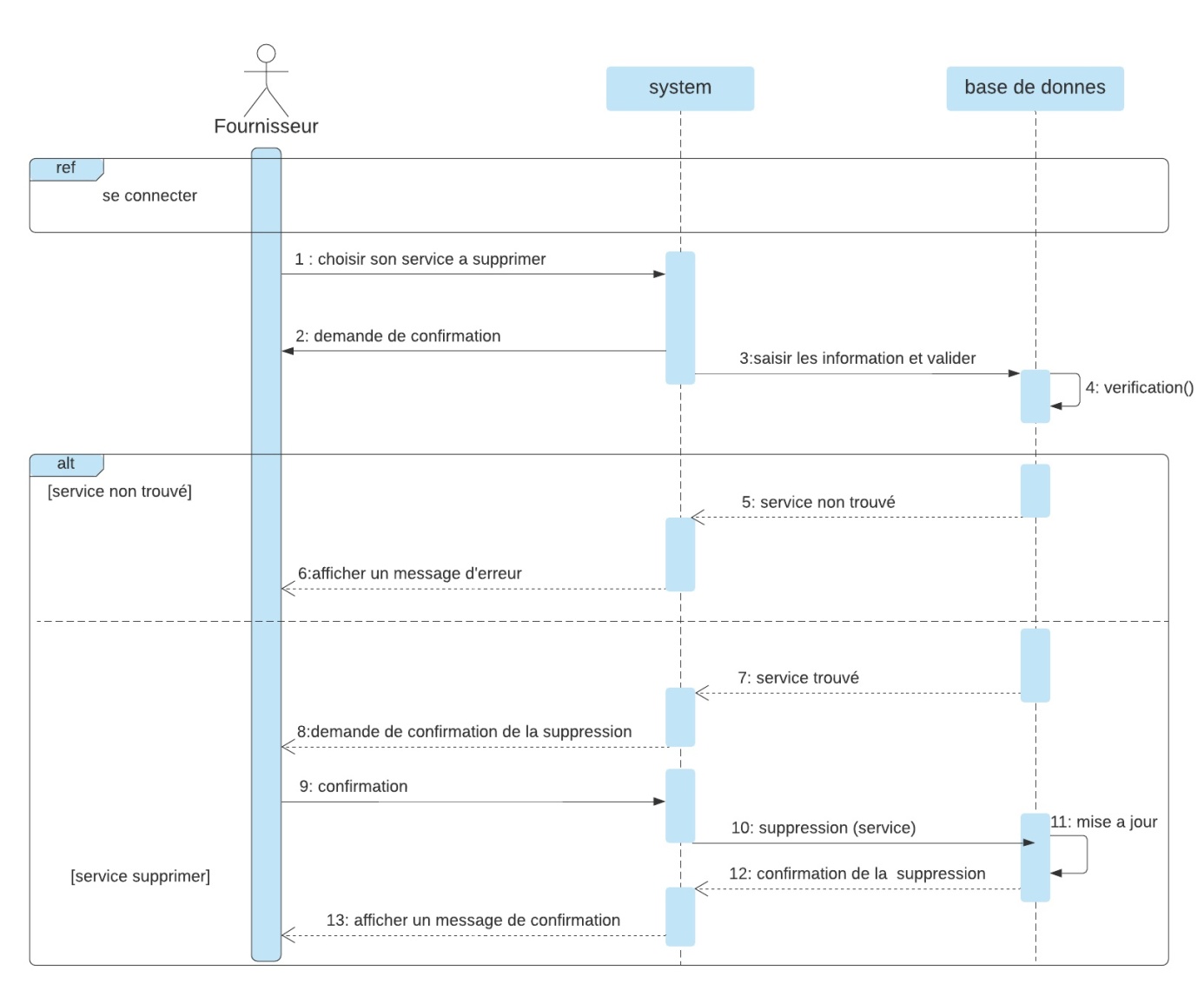
Le diagramme de séquence suivant illustre les interactions nécessaires pour la recherche d'un service. En effet, l'internaute saisit la recherche, le système va vérifier si le service cherché existe dans la base de données ou pas, et il lui affiche un résultat.



**Figure 2.11 Diagramme de séquence du cas d'utilisation « Chercher un service »**

* **Supprimer un service**

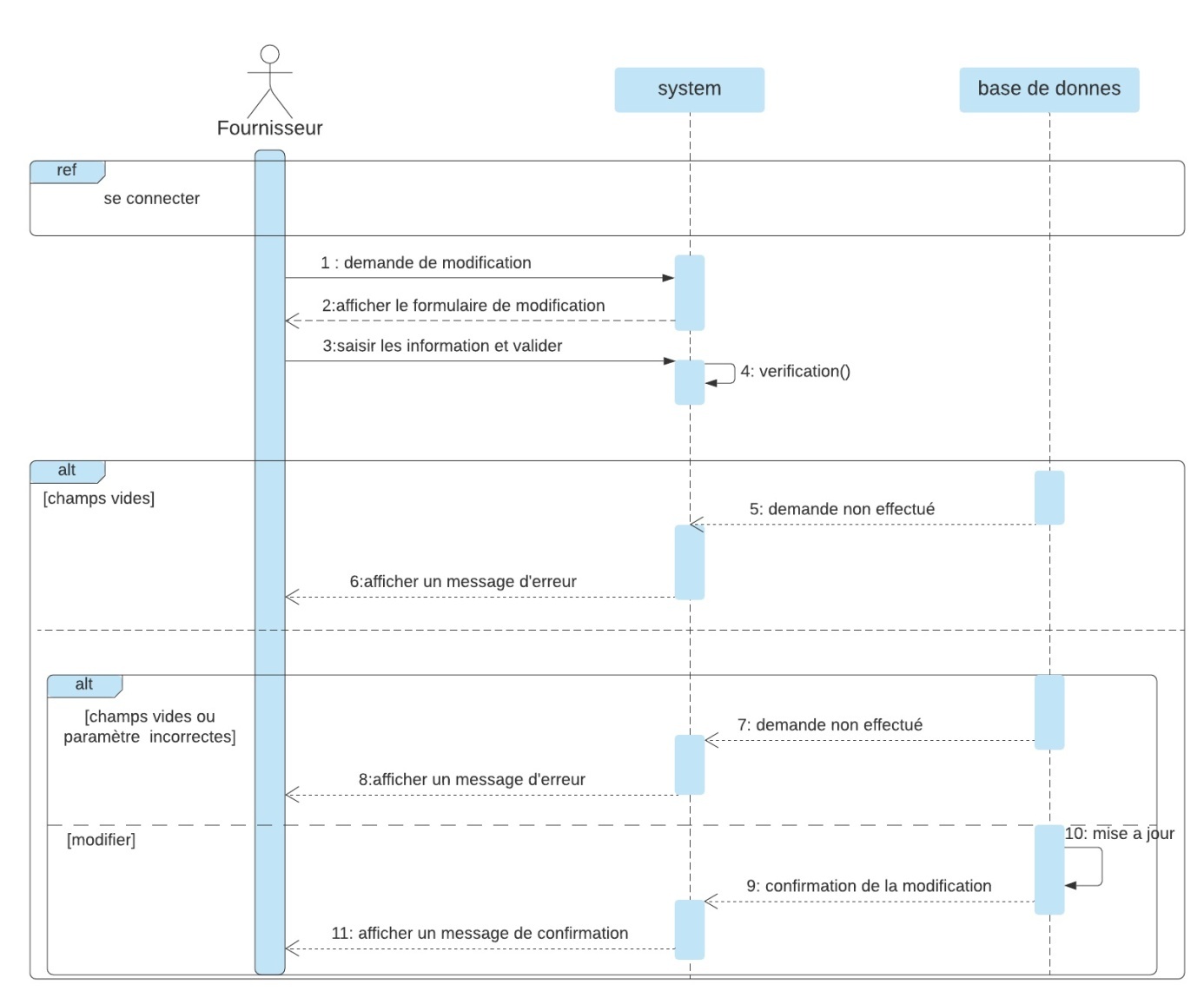
Le diagramme de séquence suivant illustre les interactions nécessaires pour la suppression d'un service. En effet le Fournisseur choisit un service, le système demande à le fournisseur une confirmation de suppression, si oui, le système vérifie l'existence de service dans la base de données, si le service existe la suppression va être appliquée sinon un message d'erreur va être affiché



**Figure 2.12 Diagramme de séquence du cas d'utilisation « Supprimer un service »**

* **Modifier son compte**

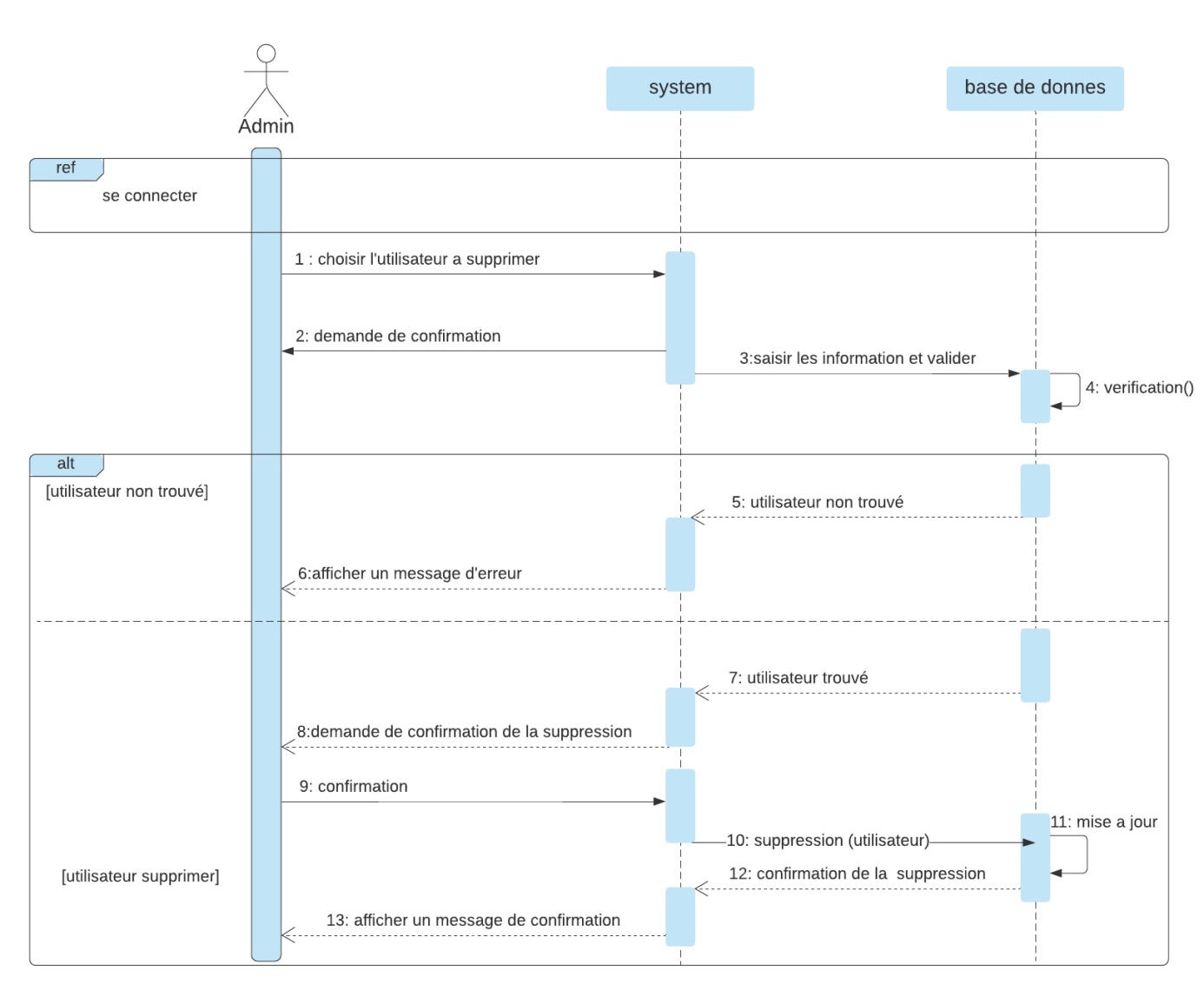
Le diagramme de séquence suivant illustre les interactions nécessaires pour qu'un fournisseur modifie les informations de son compte. En effet, le fournisseur demande au système la modification de son compte, ensuite le système affiche le formulaire de modification et le fournisseur modifie ses informations, le système vérifie à nouveau les informations, si elles sont correctes, la base de données sera mise à jour sinon un message d'erreur sera affiché.



**Figure 2.13 Diagramme de séquence du cas d'utilisation «  Modifier son compte»**

* **Supprimer un utilisateur**

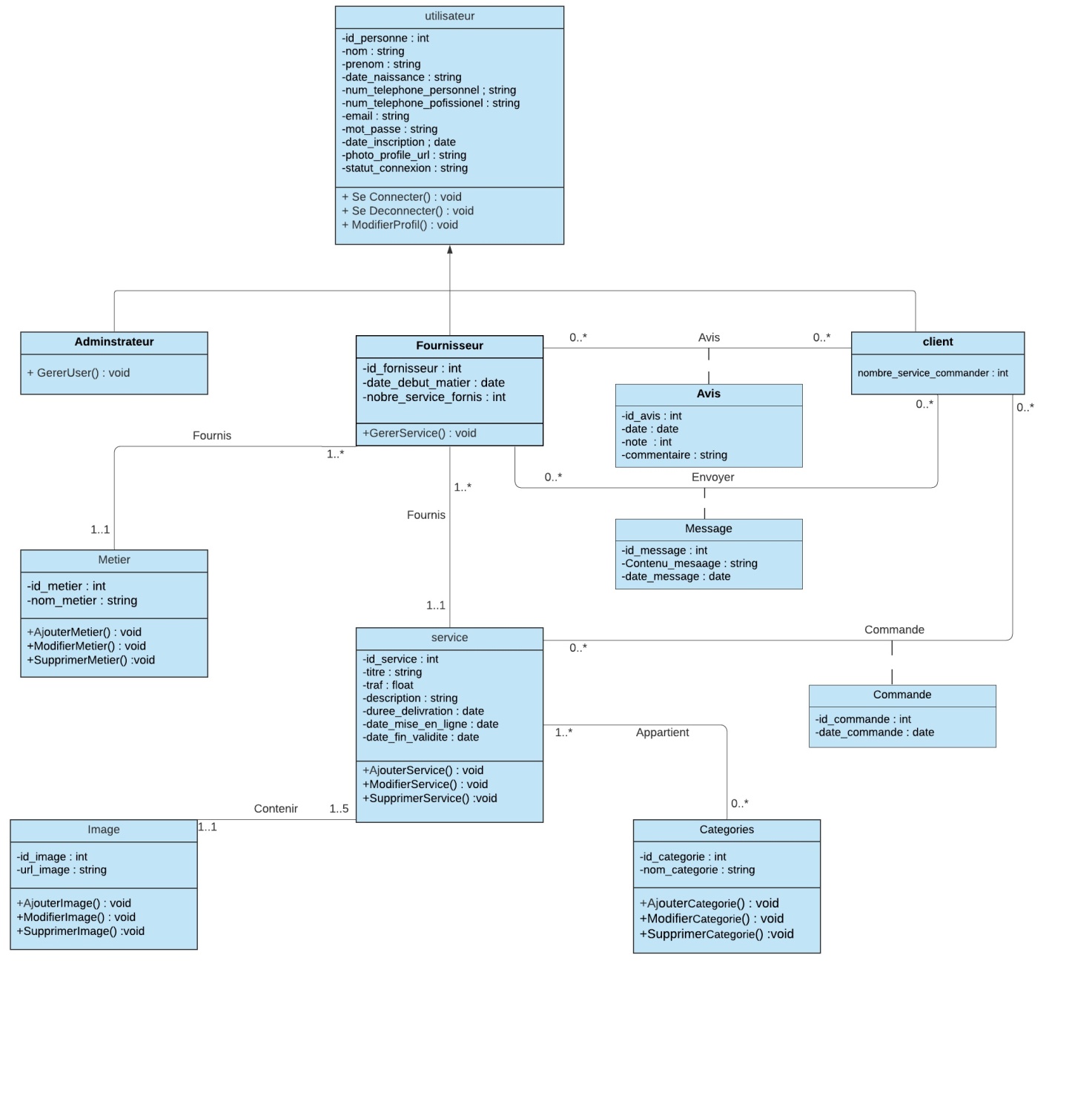
Le diagramme de séquence suivant illustre les interactions nécessaires pour la suppression d'un utilisateur. En effet, l'administrateur choisit l'utilisateur à supprimer, le système vérifie son existence dans la base de données, si l'utilisateur existe la suppression sera appliquée sinon un message d'erreur sera affiché.



**Figure 2.14 Diagramme de séquence du cas d'utilisation « Supprimer un utilisateur»**

**2.3.5 Diagramme de classe**

Le digramme de classe permet de donner la représentation statique du système à développer. Cette représentation est centrée sur les concepts de classe et d'association. Chaque classe se décrit par les données et les traitements dont elle est responsable. Nous avons pu construire le diagramme de classe indiqué ci-dessous ensuite le passage au modèle logique est effectué pour former la base de données de l'application.



**Figure 2.15 Diagramme de classe**

**2.3.6 Le schéma relationnel**

Pour réaliser ce passage, nous avons suivi des règles strictes et précises permettant de traduire le contenu conceptuel du diagramme de classe en modèle relationnel. Ces règles sont :

**2.3.6.1 Règle de passage du diagramme de classe au modèle relationnel**

Nous avons pu passer du diagramme de classes au modèle relationnel en se basant sur les règles suivantes :

* Règle 1 :

Transformation des classes : Chaque classe devient une table. L'identifiant de la classe devient la clé primaire et chaque attribut de classe se transforme en un champ de table.

* Règle 2 :

Association plusieurs-à-plusieurs : Toute association sous forme ( ?..\*) à ( ?..\*), donne naissance à trois tables, dont la table dérivée de l'entité association possédera les clés primaires des deux autres tables comme clé primaire, plus ses propres attributs s'il s'agit d'une classe d'association.

* Règle 3 :

Association d'un-à-plusieurs : Toute association binaire ( ?..1) à ( ?..\*) donne naissance à une table dérivée de l'entité de la cardinalité (1..1), et sa clé primaire est déposé comme clé étrangère dans la table dérivée de l'entité de la cardinalité ( ?..\*).

**2.3.6.2 Le modèle relationnel**

**Personne**(#id\_personne,nom,prenom,date\_naissance,num\_telephone\_personnel,num\_telephone\_pofissionel,email,mot\_passe,date\_inscription,dernier\_connection,photo\_profile\_url,statut\_connexion).

**Adminstrateur**(#id\_adminstrateur,géré\_user).

**Fornisseur**(#id\_fornisseur ,date\_debut\_matier,nobre\_service\_fornis).

**Metiers**(#id\_metier,nom\_metier).

**Possede**(#id\_fornisseur,#id\_metier ).

**Service**(#id\_service,titre,traf,description,duree\_delivration,date\_mise\_en\_ligne,

date\_fin\_validite,#id\_fornisseur ,#id\_image).

**Client**(#id\_client ,nombre\_service\_commander).

**Message**(#id\_message,contenu\_message , date\_message,#id\_fornisseur,#id\_client).

**Avis**(#id\_avis, date,note ,commentaire , #id\_fornisseur,#id\_client).

**Commande**(#id\_commande,date\_commande #id\_client,#id\_service).

**Categories**(#id\_categorie,nom\_categorie).

**Appartient**(#id\_service,#id\_categorie).

**Image**(#id\_image,url\_image).

**2.4 Conclusion**

Dans ce chapitre, nous avons présenté le processus de développement UP, le langage de modélisation UML, et fait la modélisation des différents diagrammes : diagrammes de cas d'utilisations, diagrammes de séquences et diagrammes de classes. Cette étape de modélisation nous a permis d'avoir une vue générale sur le comportement théorique des fonctionnalités offertes par notre application. Cette base théorique nous servira de guide pour le développement de l'application, qui fera l'objet du chapitre suivant.

**Chapitre 3**

**Réalisation**

**3.1 Introduction**

Depuis le début de ce projet, nous avons bien déterminé les perspectives de l'application en commençant par une étude préalable, qui nous a permis d'avoir un objectif bien concret. Puis en arrivant à l'analyse, nous avons pu avoir une idée bien claire sur comment va être notre application web, et c'est ce qui nous a mené à mieux comprendre ses fonctionnalités. Dans ce chapitre, nous allons nous appuyer sur la conception réalisée dans le chapitre précédent afin de présenter la concrétisation de l'application à travers son implémentation. Nous allons commencer par présenter les différents langages de développement ainsi que les outils de travail utilisés dans notre démarche, nous allons ensuite présenter en détail le principe de fonctionnement de ses outils, nous allons finir en présentant les interfaces des pages web de notre application ainsi que les différentes actions possibles à effectuer dans ces pages.

**3.2 Langages et environnements de développement**

**3.2.1 Le modèle MVC**

Le MVC (M : Modèle, V : Vue, C : Contrôleur) est destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective.

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.

* Modèle : cette partie gère les données de votre site. 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 entre autres 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. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.
* 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 à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès).



**Figure 3.1 Architecture MVC**

**3.2.2 PHP**

PHP signifie Personal Home Page, c'est un langage de script open source côté serveur utilisé pour créer des pages Web dynamiques. Les fichiers PHP peuvent contenir du texte, du HTML, du CSS, du JavaScript et du code PHP. Ce dernier est exécuté sur le serveur et le résultat est renvoyé au navigateur en format HTML. C'est probablement le langage de script le plus populaire.

**PHP peut :**

* Générer du contenu de page dynamique, créer, ouvrir, lire, écrire, supprimer et fermer des fichiers sur le serveur.
* Collecter des données de formulaire.
* Envoyer et recevoir des cookies.
* Ajouter, supprimer, modifier des données dans votre base de données.
* Être utilisé pour contrôler l'accès utilisateur.
* Chiffrer des données [10].

**3.2.3 SQL**

SQL (Structured Query Language) est un langage informatique standard pour la gestion de bases de données relationnelles et la manipulation de données. SQL est utilisé pour interroger, insérer, mettre à jour et modifier des données. La plupart des bases de données relationnelles prennent en charge SQL, ce qui constitue un avantage supplémentaire pour les administrateurs de base de données, car elles sont souvent nécessaires pour prendre en charge des bases de données sur plusieurs plates-formes différentes.

**3.2.4 HTML5**

HTML (Hyper Text Markup Language) est un langage qui permet la description de documents, crée par Tim Berners-Lee suite à l'invention du web. Il est en mesure d'afficher des éléments textuels, graphiques et vidéo dans un navigateur web. Il est constitué d'un ensemble de balises et d'attributs utilisés pour gérer et organiser le contenu d'une page web.

**3.2.5 CSS3**

CSS (Cascadding Style Sheets) est un langage complémentaire à celui du HTML5. Il se charge de la mise en forme et de la mise en page des éléments. Il permet de gérer l'apparence de la page tout en précisant les couleurs, le positionnement des objets, des marges, etc.

**3.2.6 Java Script**

JavaScript (JS) est un langage de script, principalement utilisé sur le Web. Il est utilisé pour améliorer les pages HTML et est généralement intégré au code HTML. C'est un langage interprété, ce qui signifie qu'il n'a pas besoin d'être compilé. Il rend les pages Web de manière interactive et dynamique, cela permet aux pages de réagir aux événements, d'afficher des effets spéciaux, d'accepter des textes variables, de valider des données, de créer des cookies, de détecter le navigateur de l'utilisateur, etc.

**3.2.7 jQuery**

jQuery est une bibliothèque JavaScript rapide, petite et riche en fonctionnalités. Il simplifie considérablement la traversée et la manipulation de documents HTML, la gestion des événements, l'animation et Ajax grâce à une API facile à utiliser qui fonctionne sur une multitude de navigateurs. Avec une combinaison de polyvalence et d'extensibilité, jQuery a changé la façon dont des millions de personnes écrivent du JavaScript.

**3.2.8 Ajax**

AJAX (Asynchronous JavaScript and XML) ce qui, transcrit en français, signifie « JavaScript et XML asynchrones»

Derrière ce nom se cache un ensemble de technologies destinées à réaliser de rapides mises à

jour du contenu d'une page Web, sans qu'elles nécessitent le moindre rechargement visible par l'utilisateur de la page Web. Les technologies employées sont diverses et dépendent du type de requêtes que l'on souhaite utiliser, mais d'une manière générale le JavaScript est constamment présent.

Ajax est une technologie JavaScript fort répandue qui permet d'envoyer des requêtes au serveur et de recevoir des réponses sans rechargement de la page. Il est par ce moyen possible de modifier dynamiquement le DOM (Document Object Model), donc une partie de la page.

**3.3 Outils de développement**

**3.3.1 XAMPP**

XAMPP est l'environnement de développement PHP le plus populaire. XAMPP est une distribution Apache entièrement gratuite et facile à installer contenant MySQL, PHP et Perl. Le paquetage open source XAMPP a été mis au point pour être incroyablement facile à installer et à utiliser.

**3.3.2 MYSQL**

C'est un système de gestion de base de données relationnelle (SGBDR) open source avec un modèle client-serveur. Il est développé, distribué et pris en charge par Oracle Corporation.

**3.3.3 PHPStorm**

Jetbrains PHPStorm est un environnement de développement intégré spécialisé pour le développement Web. Initialement conçu pour le développement PHP avec tout le support deWebstorm, grâce à son extensibilité, plusieurs plugins existent pour étendre ses fonctionnalités. Il supporte nativement les langages PHP, HTML, CSS et JavaScript.

Si plusieurs développeurs Web se penchent vers cet IDE, c'est grâce aux nombreux avantages

et fonctionnalités que possède PHPStorm qui rendent vraiment facile la vie des développeurs en occurrence :

1. Le support de la coloration syntaxique.

2. Un générateur de code.

3. L'auto compilation.

4. L'auto complétion intelligent du code.

5. Une documentation du langage PHP intégré.

6. La factorisation du code..

8. Un support de gestion de version (GIT, SVN) intégré.

9. Les supports des framework (Symfony) PHP intégré.

**3.3.4 Draw.io**

C'est une application gratuite en ligne, accessible via le navigateur qui permet de dessiner des diagrammes ou des organigrammes. Cet outil nous propose de concevoir toutes sortes de diagrammes, de dessins vectoriels, de les enregistrer ou de les exporter en format image, XML,PDF, etc.

**3.4 Frameworks utilisés**

Un Framework est, comme son nom l'indique en anglais, un «  cadre de travail » . L'objectif d'un Framework est généralement de simplifier le travail des développeurs informatiques, en leur offrant une architecture "prête à l'emploi" et qui leur permettre de ne pas repartir de zéro à chaque nouveau projet.

Les Frameworks sont comparables aux patrons de couture. Les principaux avantages sont donc :

* La réutilisation des codes.
* La standardisation de la programmation.
* La formalisation d'une architecture adaptée aux besoins de chaque entreprise.

À noter aussi que les Frameworks sont toujours « enrichis » de l'expérience de tous les développements antérieurs.

Nous avons utilisé deux Frameworks :

**3.4.1 tailwind ui** est un framework CSS complètement personnalisable, basé sur le principe de classes utilitaires

**3.4.2 CodeIgniter**

CodeIgniter est un framework de développement d'applications - une boîte à outils - pour les personnes qui créent des sites Web en utilisant PHP. Son objectif est de vous permettre de développer des projets beaucoup plus rapidement que si vous écriviez du code à partir de zéro, en fournissant un riche ensemble de bibliothèques pour les tâches courantes, ainsi qu'une interface simple et une structure logique pour accéder à ces bibliothèques. CodeIgniter vous permet de vous concentrer de manière créative sur votre projet en minimisant la quantité de code nécessaire pour une tâche donnée.

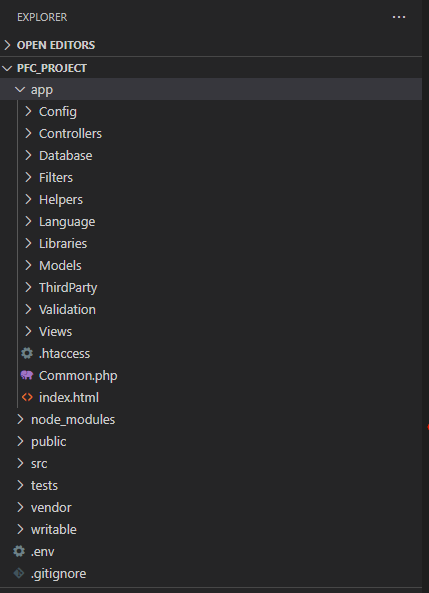
CodeIgniter requière l'installation d'un gestionnaire de dépendances « composer ».

Pour installer CodeIgniter nous devons écrire dans le composer la commande suivante :

|  |
| --- |
| composer create**-**project codeigniter4**/**appstarter NomDuProjet |

* Organisation de CodeIgniter

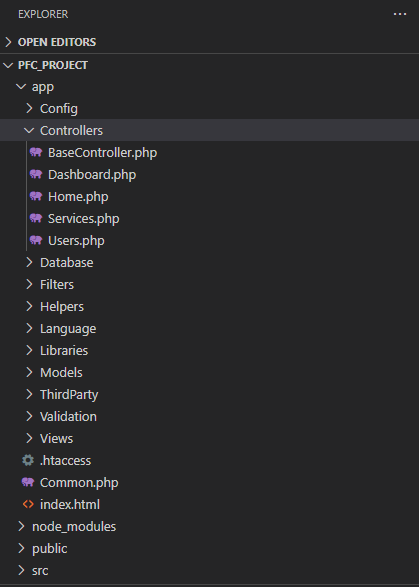
CodeIgniter est composé d'une hiérarchie de dossiers et fichiers représentée dans cette figure :



**Figure 3.2 Organisation de CodeIgniter**

* **Cache**: Ce dossier contient toutes les pages en cache de votre application. Ces pages mises en cache augmenteront la vitesse globale d'accès aux pages.
* **Config**: Ce dossier contient config.php, l'utilisateur peut configurer l'application.En utilisant le fichier database.php, l'utilisateur peut configurer la base de données de l'application.
* **Contrôleurs**: Ce dossier contient les contrôleurs de votre application. C'est la partie de base de votre application.
* **Helpers :** Dans ce dossier, vous pouvez mettre la classe d'assistance de votre application.
* **Database**: Le dossier de la base de données contient les pilotes de base de données de base et d'autres utilitaires de base de données.
* **Langue :** Ce Le dossier contient les fichiers relatifs à la langue.
* **Bibliothèques :** Ce dossier contient les fichiers des bibliothèques développées pour votre application.
* **Modèles :** la connexion à la base de données sera placée dans ce dossier.
* **Vues :** Les fichiers HTML de l'application seront placés dans ce dossier.
* **Les Controllers**

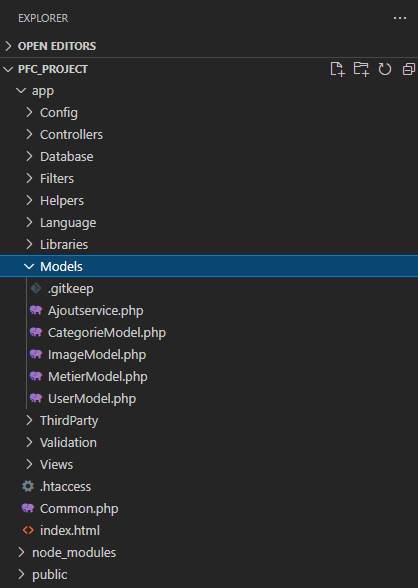
La grande majorité des actions qu'un utilisateur enclenche mènera à l'appel d'une méthode d'un contrôleur. Les Controllers vont permettre de gérer les différentes actions demandées par les utilisateurs lors d'une action



**Figure 3.3 Vue sur les Contrôleurs**

Il est assez simple de créer un contrôleur avec CodeIgniter. Pour effectuer cette opération:

* Il faut crcréer une classe qui hérite de la classe Controller de CodeIgniter.
* Il faut aussi créer un fichier PHP portant le même nom que le contrôleur que nous allons créer dans le dossier system/application/controllers (ce dossier se trouve dans le dossier de publication de votre serveur web).
* **Le model**
* Le Model est la partie qui permet de stocker et de traiter les donnes.  
  D'une manière générale en PHP et avec CodeIgniter, le model n'effectue que les opérations en base de donnes
* Pour créer un Modele avec CodeIgniter, il faut créer un fichier PHP dans le dossier system/application/models contenant une classe qui hérite de la classe Model du framework de CodeIgniter. Dans cette classe, on crée les fonctions qui effectuent les traitements sur les donnes et qui appellent les fonctions se chargeant de stocker les donnes (insertion SQL, écriture dans un fichier texte ou XML) ainsi qu'un constructeur appelant le constructeur de la classe Model.

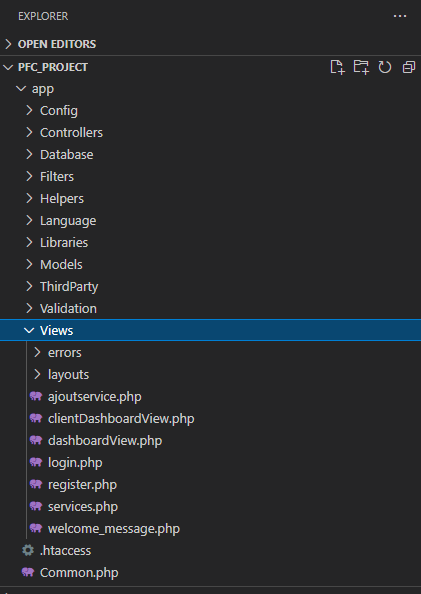
****

**Figure 3.4 Vue sur le Model**

* **Vues (views)**

Pour afficher correctement une page Web dans le navigateur, il faut utiliser une vue. C'est la vue qui est en charge de générer le code HTML.

Pour créer une vue avec CodeIgniter, il suffit de créer un script PHP ou HTML, qui affiche les donnes souhaites, dans le dossier system/application/views. Il n'y a pas de conventions particulières.



**Figure 3.5 Le dossier application/views**

**3.5 Présentation des interfaces graphiques**

Nous allons présenter dans ce qui suit, les imprimes-écran des principales interfaces réalisées pour notre application web.

**3.5.1 La page d'accueil**