République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/MIRA de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

MÉMOIRE DE MASTER PROFESSIONNEL

En Informatique

Option : ***Génie Logiciel***

Thème

Conception et réalisation d'une application Web *"iManager"* dédiée à la gestion des contrats fournisseurs.

"Cas d'étude : entreprise de Cevital de Bejaia".

Présenté par : M^{lle}. **GUEDJALI Lidia**
M^{lle}. **CHELLA Silya**

Devant le jury composé de :

Président	**L. HAMZA**	**Maître de Conf. A**	**U. A/Mira Béjaia.**
Examinateur	**K. AMROUNE**	**Professeur**	**U. A/MiRA Béjaia.**
Encadrant	**M. MOHAMMEDI**	**Maître de Conf. A**	**U. A/Mira Béjaia.**

Béjaia, 2021/2022.

⁂ *Remerciements* ⁂

Nous remercions Allah le tout puissant de nous avoir donné le courage et la volonté d'achever ce travail et sans lequel il n'aurait jamais été accompli.
Nous tenons à remercier chaleureusement notre encadrant au sein de la faculté monsieur **M. MOHAMMEDI** pour ses efforts considérables durant toute l'année pour ses précieux conseils, son aide moral sa gentillesse et surtout son aimable soutien.

Nous remercions également notre maitre de stage au sein de Cevital Bejaia monsieur **Y. CHIBOUTI** pour ses précieux conseils, encouragements, relectures, correction et surtout pour sa disponibilité et la confiance qu'il nous a accordé.

Nous remercions tous les membres du jury, pour avoir accepté d'examiner notre travail. Nous disons mille fois merci à nos familles pour leurs soutiens, leurs conseils et leurs encouragements. Nous n'oublions pas de remercier nos amis et tous ceux, qui ont contribué, de près ou de loin à l'aboutissement de ce travail.

⁕ *Dédicaces* ⁕

Je dédie ce mémoire à ma très chère mère qui as construit ma réussite avec tout son amour, son soutient et ses conseils en or.

Âmon défunt père, qui a su me transmettre les valeurs essentielles de la vie, et l'amour du travail.

Â mes frères et a mes adorables sœurs

Qui m'ont prodigué beaucoup d'aides et d'encouragements tout au longe mes études, et leurs conseils de valeur.

Â tous mes amis qui j'ai passé des moments inoubliables

Â ma chère binôme et amie Lidia avec qui j'ai passé des moments agréables et inoubliables et avec qui j'ai partagé ce travail.

Toutes les personnes que je connais et que je n'ai pas citées.

CHELLA Silya

⁕ *Dédicaces* ⁕

Je dédie ce travail :

Â ma très chère mère,

Â celle qui m'a fait voir la lumière, qui m'a fait goûter la joie, qui m'apprit le sourire, qui a toujours été là pour moi, qui a veillé durant mes nuits pour faire la réussite de mes jours, qui m'a tout donné sans jamais rien demander et à qui je dois tout et que rien ne suffira pour la remercier.

Â mon très cher Père,

Â celui qui m'a toujours soutenu, poussé et motivé dans mes études et à qui je dois ma place maintenant pour ses sacrifices.

Â mes cher frères et sœurs,

Qui m'ont prodigué beaucoup d'aides et d'encouragements tout au longe mes études, et leurs conseils de valeur.

Â tout ma famille.

Â ma chère binôme et amie Silya avec qui j'ai passé des moments agréables et inoubliables et avec qui j'ai partagé ce travail.

Â tous mes amis.

Â tous les membres du groupe **Help Tech**.

Toutes les personnes que je connais et que je n'ai pas citées.

GUEDJALI Lidia

TABLE DES MATIÈRES

LISTE DES TABLEAUX

TABLE DES FIGURES

LISTE DES ACRONYMES

A	API	Application Programming Interface.
C	CSS	Cascading Style Sheets.
D	DSI	Direction du Système d'Information.
M	MVC	Model View Controller.
	MySQL	My Structured Query Language.
P	PET	Polyéthylène Téréphatalate.
	PHP	Hypertext Preprocessor.
	POO	Programmation Orianté Objet.
R	RUP	Rational Unified Processus.
S	SPA	Société Par Action.
T	2TUP	Two Tracks Unified Process.
U	UML	Unified Modeling Language.
	UP	Processus Unifié.
V	VS Code	Visual Studio Code.

INTRODUCTION GÉNÉRALE

Aujourd'hui, toute entreprise est prête à investir des sommes considérables dans l'implantation des technologies logicielles afin d'améliorer ses services, d'accroitre son agilité et sa flexibilité, de réduire les couts, d'augmenter la production et de faire face aux défis du marché. En effet, vu la croissance des activités au sein des entreprises, la tâche de gérer efficacement toutes ces fonctions s'avère de plus en plus complexe et difficile.
Pour surpasser ces difficultés, une entreprise doit utiliser des outils optimisés facilitant les taches et offrant des fonctionnalités riches et utiles, des outils de gestion et d'analyse permettant d'optimiser la diffusion des information en interne, d'améliorer les processus de gestion et d'automatiser les tâches. Ce qui augmente énormément la réactivité des entreprises.

Parmi les entreprises algériennes qui sont confrontées à ce type de problèmes, nous trouvons Cevital, où nous avons effectué un stage et nous avons constaté que ces services traitent des contrats fournisseurs d'une manière manuelle, ce qui mène à une gestion inefficace de contrats fournisseurs et crée des risques de perturbation et surcharge ainsi que le manque de visibilité des échéances d'un contrat.
Le stockage de tous les contrats fournisseurs d'un unique espace dédié exige en effet une vérification une à une de chaque contrat pour détecter celui recherché. À l'origine d'une énorme perte de temps, la productivité de l'entreprise est aussi victime d'un ralentissement extrême.
C'est là que l'utilisation d'une solution de gestion de contrats permet d'améliorer l'efficacité des processus d'achat. Cependant améliorer la productivité en automatisant les processus collaboratifs de contractualisation.
C'est dans ce cadre que s'inscrit notre projet de fin de cycle qui a pour objectif de mettre en place une application web *iManager* dédiée à la gestion des contrats fournisseurs.

Ce présent mémoire se décompose en quatre chapitres, dans le premier chapitre nous présenterons l'organisme d'accueil et la méthodologie de conception ainsi que le sujet en décrivant clairement la problématique et les objectifs du projet.
Le deuxième chapitre concernera l'études préliminaire et analyse des besoins qui regroupe toutes les étapes de notre processus de développement, en définissant tout d'abord les besoins fonctionnels et non fonctionnels auxquels doit répondre notre application, ainsi que l'identification des acteurs et les différents cas d'utilisation. Ces derniers seront décrits ensuite par des diagrammes de séquences.

Dans le troisième chapitre on trouvera la conception du notre système en donnant la description de la conception exprimée à l'aide du diagramme de classe.
C'est dans le quatrième chapitre qu'on citera des différents outils de développement utilisé dans la réalisation de notre application ainsi que l'environnement de développement et quelques captures d'écrans qui permettront de présenter notre application.

Nous terminerons ce mémoire par une conclusion générale et des perspectives, que nous souhaiterons accomplir dans le future proche.

CHAPITRE 1

PRÉSENTATION DE L'ORGANISME D'ACCUEIL ET MÉTHODOLOGIE DE CONCEPTION.

1.1 Introduction

Dans ce chapitre nous allons présenter l'organisme d'accueil, son historique, ses activités et objectifs. Par la suite nous allons procéder à étudier la méthodologie de travail fondé sur le formalisme UML en raison de la flexibilité marquante qu'il offre, suivi de la présentation du processus unifié. Ensuite, nous allons exposer les problèmes de gestion rencontrés par Cevital.

1.2 Présentation de l'organisme d'accueil

1.2.1 Présentation de Cevital

Cevital est un complexe d'industrie agro-alimentaire. Il a été créé par l'entrepreneur Issad Rabrab en 1998 sous forme d'une société par actions(SPA) ; C'est le premier groupe privé Algerien, présent à l'internationale. Il renferme 26 filiales avec 18000 employés répartis sur trois continents.
Il se situe au niveau de l'arrière port de Bejaia, à proximité de RN 12.
Limité par Oued Ghir et Sonatrach au sud, le centre-ville de Bejaia au nord, la méditerranée à l'est, NAFTAL et ECOTEX au sud-ouest.
Cette place stratégique offre un grand avantage de proximité économique, puisque elle se trouve proche du port et de l'aéroport [3].

1.2.2 Historique de Cevital

Première entreprise privée algérienne à avoir investi dans des secteurs d'activités diversifiés, elle a traversé d'importantes étapes historiques pour atteindre sa taille et sa notoriété actuelle.

- **1971** :Lancement de la construction métallique.
- **1988** :Création de METAL SIDER.
- **1991** :Creation de quotidien d'information liberté / reprise des activités.

— **1997** :Création de Hyundai Motors Algérie.
— **1998** :Création de Cevital SPA Industies agroalimentaires.
— **2006** :Création de Numidis et Immobis / Acquisition de COJEK.
— **2007** :SAMHA -Production et distribution SAMSUNG/création MFG.
— **2008** :Nulis -Transport maritime / commercialisation du verre plat en Europe/ Création de NUMILOG.
— **2009** : Augmentation de la production de sucre de M T/AN.
— **2013** : OXXO (France)/ ALAS(Espagne).
— **2014** : BRANDT(France) / AFFERPI(Italie)/ ex LUCCHINI PIOMBINO.
— **2015** : Création de CTLOG / NUMILOG de France [2].

1.2.3 Les activités de Cevital

1. **Activité de Cevital au niveau de la commune de Bejaia**
   Au niveau de la commune de Bejaia, L'entreprise Cevital fait la contribution des installation suivantes
   — Le raffinage de sucre avec une capacité de 5000 T/jour.
   — Sucre liquide 500 T/jour.
   — Le raffinage de l'huile avec une capacité de 1800 T/jour.
   — Production de margarines (Fleurial, Matina...etc).
   — Fabrication d'emballage en PET de différentes dimensions(9000 unité/heure).
   — Silos Porturaires(24 silos pour stocker la matière brute et 1 silos pour stocker le sucre cristallisé de capacité 70000 Tonnes).
   — Trituration des graines oléagineuses(en cours de réalisation).
2. **Acitivité de Cevital au niveau de la commune d'EL-Kseur**
   Au niveau de la commune d'El-kseur (Béjaia) on retrouve les unités de production du jus Fruits COJECK.
3. **L'activité de Cevital au niveau de la commune d'Agueni Gueghane :**
   Au niveau de la commune de Agouni Gueghane (les montagnes de Djurdjura Tizi ouzou), on trouve L'unité de production d'eau minérale Lalla khedidja.

1.2.4 Objectifs de Cevital

Les objectifs du groupe Cevital sont les suivants :
— L'extension de ses produits à l'échelle nationale.
— Encourager les agriculteurs à produire les graines locale grâce à une aide financière.
— L'importation des graines oléagineuses pour l'extraction directe des huiles brutes.
— Optimiser ses opportunités d'emplois sur le marché de travail.
— Augmentation de chiffre d'affaire.
— Répondre aux besoins des clients.
— Modernisation des installations pour les marchés étrangers et les exportations.
— Améliorer partenariats étrangers.
— Améliorer en continu ces performances à tous les niveaux.

— Positionner ses produits sur les marchés étrangers par l'exportation.

1.2.5 Organigramme du groupe Cevital

La figure 1.1 illustre l'organigramme générale de l'entreprise Cevital et notre domaine d'étude est la direction du système d'information(DSI).

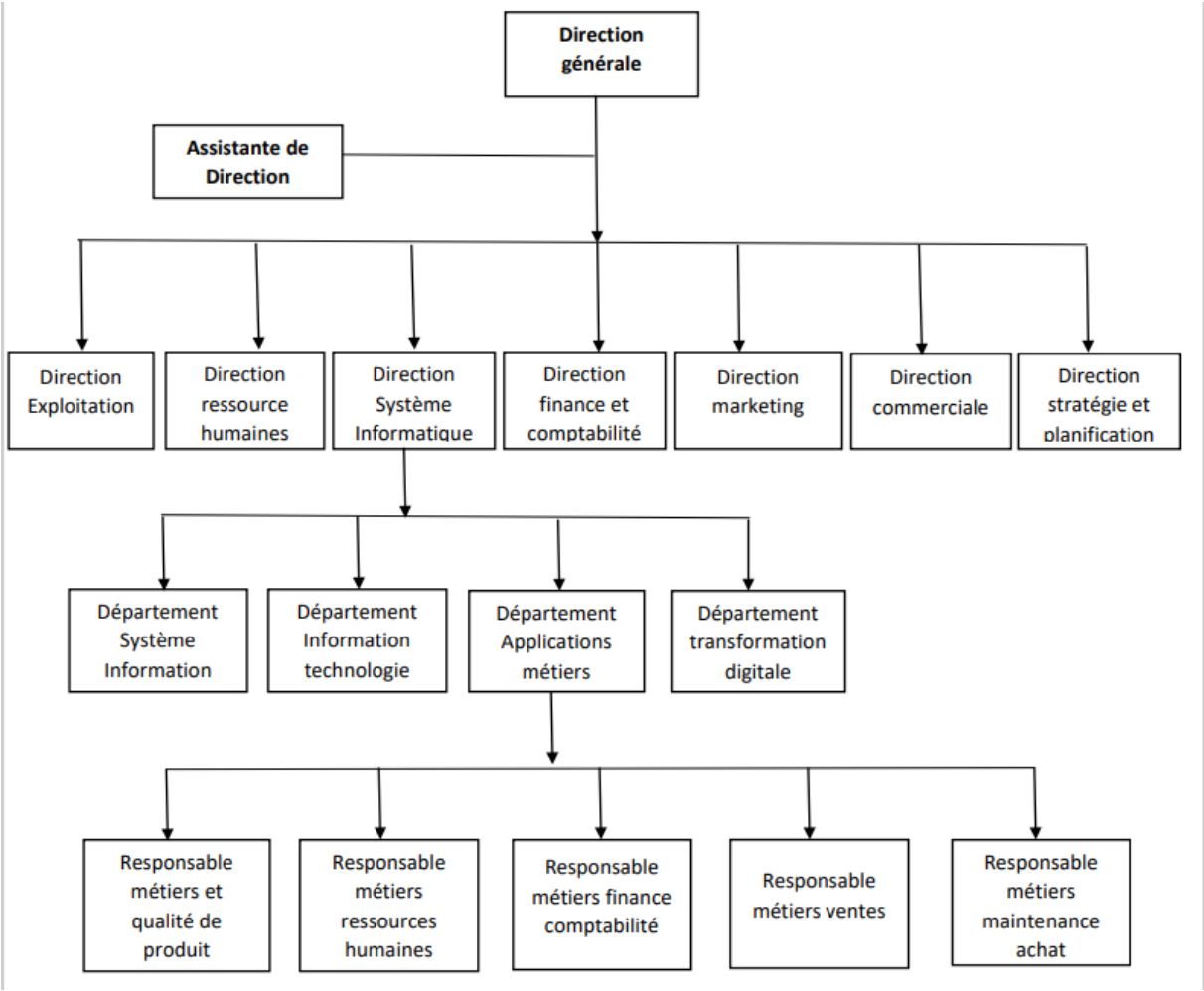


FIGURE 1.1 – Organigramme de l'entreprise Cevital.

.

1.3 Méthodologie de conception

1.3.1 Le formalisme UML

Présentation d'UML

UML est un acronyme anglais de Unified Modeling Language ou Le Langage de Modélisation Unifié en français.

UML est un langage visuel de modélisation à usage général composé d'un ensemble intégré de

diagrammes, développé pour aider les développeurs à spécifier, visualiser, construire et documenter le système et les objets qu'il contient, celui-ci est une partie très importante du développement de logiciels orientés objet et du processus de développement logiciel [9].

L'objectif principal d'UML est de définir une manière standard de visualiser la façon dont un système a été conçu et aide les équipes de projet à communiquer et à explorer les conceptions potentielles ainsi qu'à valider la conception architecturale du logiciel [17].

Différents diagrammes d'UML

UML propose treize diagrammes complémentaires qui permettent la modélisation d'un projet tout au long de son cycle de vie. Ces diagrammes sont répartis en deux catégories : Diagrammes structurels et comportementaux [5]

— **Diagrammes structurels :** montre la structure statique du système.
Cette catégorie regroupe les diagrammes suivants :
• *Diagramme de classe :* le diagramme de classe contient un ensemble de classes ainsi que leurs relations. L'intérêt majeur de ce diagramme est de représenter les entités du système d'information.
• *Diagramme de déploiement :* utilisé dans la modélisation des aspects physiques d'un système orienté objet, ces diagrammes montrent la disposition physique des différents matériels qui entrent dans la composition d'un système et la répartition des programmes exécutables sur ces matériels.
• *Diagramme d'objets :* les diagrammes objets représentent les instances des éléments et leurs relations
• *Diagramme de packages :* spécifie l'organisation logique du modèle et les relations entre packages.
• *Diagramme de structure :* désigne l'organisation interne d'un élément statique complexe.
• *Diagramme de composants :* décrivent les composants physiques et l'architecture interne d'un logiciel.

— **Diagrammes comportementaux :** montre le comportement dynamique des objets dans un système [6].
UML propose les diagrammes comportementaux suivants :
• *Diagramme de cas d'utilisation :* les diagrammes de cas d'utilisation représentent les fonctions du système selon les besoins de l'utilisateur.
• *Diagramme d'activité :* décrivent les comportements des actions au sein d'une activité.
• *Diagramme de vue d'ensemble des interactions :* fusionne les diagrammes d'activité et séquence pour combiner des fragments d'interaction avec des décisions et des flots.
• *Diagramme de séquence :* décrivent de manière temporelle les interactions entre objets et acteur.
• *Diagramme de communication :* désigne la communication entre objets dans le plan au sein d'une interaction.
• *Diagramme de temps :* fusionne les diagrammes d'états et de séquences pour montrer l'évolution de l'état d'un objet au cours du temps.
• *Diagramme d'états-transitions :* montre un automate à états finis qui met en évidence l'enchaî-

nement des différents états d'une classe.

Pour le cas de notre application les diagrammes utilisés sont les diagrammes de cas d'utilisation, de séquence, de classes et diagramme de déploiement.

1.3.2 Processus Unifié

Le processus unifié est une méthode générique de développement de logiciels. c'est-à-dire qu'il faut adapter l'UP au contexte d'un projet, d'une équipe, d'un domaine applicatif ou d'une organisation[19]. C'est une méthode pour soutenir le cycle de développement logiciel et aussi le développement de logiciels orientés objet. UP est un processus de développement incrémental et itératif, ce qui signifie que le projet est divisé en phases très courtes, et une nouvelle version incrémentale est livrée à la fin de chaque phase. L'objectif d'un tel processus, est de maîtriser la complixité des projets informatique en diminuant les risques [8].

Les principales caractéristiques du processus unifié

— Le processus unifié est à la base de composants.
— Le processus unifié utilise le langage UML (ensemble d'outils et de diagramme).
— Les processus unifiés sont pilotés par des cas d'utilisation.
— Centré sur l'architecture.
— Itératif et incrémental.

Le processus unifié 2TUP

2TUP est un processus de développement logiciel qui met en œuvre un processus unifié (l'approche générale du développement logiciel est une proche du mécanisme architectural du RUP "Rational Unified Process"). Chaque phase du cycle découle de la phase précédente. Il préconise un cycle de vie en y, semblable à un cycle de développement en cascade, et il est incrémental : à partir de la capture des besoins fonctionnels, plusieurs cas d'utilisation sont définis, et chaque cas d'utilisation représente un incrément d'un cycle de développement. Il facilite une forme de recherche de qualité et des propriétés intéressantes comme la réutilisation des services [7].

Pourquoi UML UP

UML est un langage de modélisation qui permet de définir clairement les besoins des clients, de généraliser les aspects de conception et de l'architecture, de structurer de manière cohérente les fonctionnalités et les données et également de faciliter la maintenance. UP est un processus itératif et incrémental qui se lit selon deux axes : vertical (enchainement de disciplines et d'activités au sein d'une itération) et horizontal (enchainement dynamique sur l'axe temporel de phases et d'itérations), ce qui nous permet de tester à chaque itération sans attendre la fin du projet.

1.4 Problématique

De nos jours, la gestion des contrats constitue un outil nécessaire pour les entreprises. Les contrats cadrent en effet la relation avec les fournisseurs, qui jouent un rôle de plus en plus important dans la gestion des achats.
La rédaction manuelle et sur papier des contrats fournisseurs est très souvent source d'erreurs. En effet, les entreprises déclarent que les erreurs résultant des transferts manuels de données pendant la phase de préparation des contrats fournisseurs créent des problèmes importants, parmi ces dérniers on trouve :
limitation de la connaissance des informations importantes et imposition des tâches répétitives,
risque de perdre des contrats et perturbations et surcharge.
Il peut être difficile de suivre et de vérifier les contrats une fois qu'ils ont été transmis à d'autres membres du personnel qui peuvent ne pas comprendre quels détails doivent être contrôlés.
Les mises à jour des documents originaux peuvent prendre des jours, voire des semaines. De plus, les equipes peut entraîner une augmentation des risques, le non-respect des délais, des erreurs et même des litiges.
Le traitement de contrats manuellement n'est pas conforme ce qui mène à avoir de graves conséquences juridiques et/ou financières.

1.5 Objectif de projet

La gestion des contrats est un aspect important des relations commerciales et constitue un outil nécessaire utilisé par les entreprises. Cependant elle semble impossible lorsqu'elle est réalisée à la main.
La mise en place d'une solution de gestion des contrats permet d'obtenir rapidement des données plus précises en temps réel et mieux qualifiées et d'établir ainsi des prévisions plus justes.
La gestion de contrats apporte une vision cohérente de l'ensemble des contrats en centralisant les informations juridiques et financières relatives aux fournisseurs. En outre, de nombreuses fonctionnalités facilitent la recherche.

1.6 Conclusion

Dans ce chapitre, nous avons donné une présentation générale de l'organisme d'accueil Cevital. Nous avons aussi parlé de ses activité et ses objectifs, Après nous sommes passés à la présentation générale de notre projet.
A présent, nous allons passer à l'étude préliminaire et analyse des besoins qui fait l'objet de chapitre suivant.

CHAPITRE 2

ETUDES PRÉLIMINAIRE ET ANALYSE DES BESOINS

2.1 Introduction

Dans ce chapitre, nous commençons la phase de l'analyse et la spécification des besoins qui représente la première étape pour passer à la conception d'une application. Elle sert à identifier les acteurs du système et leur associer à chacun l'ensemble d'action avec lesquelles il intervient. Avant d'aller plus loin nous allons modéliser le contexte du système, considéré comme une boite noire en identifiant les entités externes au système qui interagissent directement avec lui. Nous établirons ensuite un recueil initial des besoins fonctionnels et non fonctionnels, puis l'analyse des besoins à travers les diagrammes de cas d'utilisation. Enfin nous élaborerons les diagrammes de séquences.

2.2 Diagrammes de cas d'utilisations

2.2.1 Définition

Le diagramme de cas d'utilisation est un diagramme UML utilisé pour donner une vision globale du comportement de notre système, sa représentation est caractérisée par les concepts suivants :[20]

— **Acteur :** représente un rôle joué par une entité externe (utilisateur humaine, dispositif matériel ou autre système) qui interagit directement avec le système étudié afin d'effectuer une tache significative.

— **Cas d'utilisation :** : il permet de décrire l'interaction entre les acteurs et le système.

— **Les relations entre les acteurs :** la seule relation entre les acteurs est la relation de généralisation quand un acteur hérite d'un acteur père, il hérite en réalité toutes les associations du père.

— **Les relations entre les cas d'utilisation :**

• **Relation d'inclusion :** une relation d'inclusion d'un cas d'utilisation A par rapport au cas d'utilisation B signifié qu'une instance de A contient le comportement décrit dans B.

• **Relation d'extension :**une relation d'extension d'un cas d'utilisation A par apport a un cas

d'utilisation B signifier qu'une instance de A peut être étendue par le comportement décrit dans B.

• **Relation de généralisation :** les cas d'utilisation descendants héritent de description de leur parents communs chacun d'entre eux peut néanmoins comprendre des interactions spécifiques supplémentaire.

2.2.2 Identification des acteurs

Définition Un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif, matériel ou autre système) qui interagissent directement avec le système étudié.
Nous allons présenter dans le tableau 2.1 les acteurs de notre système et leurs principaux rôles :

Acteurs	**Rôle**
Administrateur	-Géstion des utilisateurs du système (ajouter, modifier,supprimer). -Géstion des accès. -Géstion des paramètres de système.
Structure	-Saisie les contrats (ajout, transmission).
Service Comptabilité	-Traitement de contrats (réception, rejet).
Fournisseur	-Suivie les contrats.

TABLE 2.1 – Les différents acteurs de l'application à réaliser et leurs rôles.

2.2.3 Diagramme de contexte

Dans la figure 2.1, nous illustrons les différents acteurs qui interagissent avec le système que nous allons mettre en place :

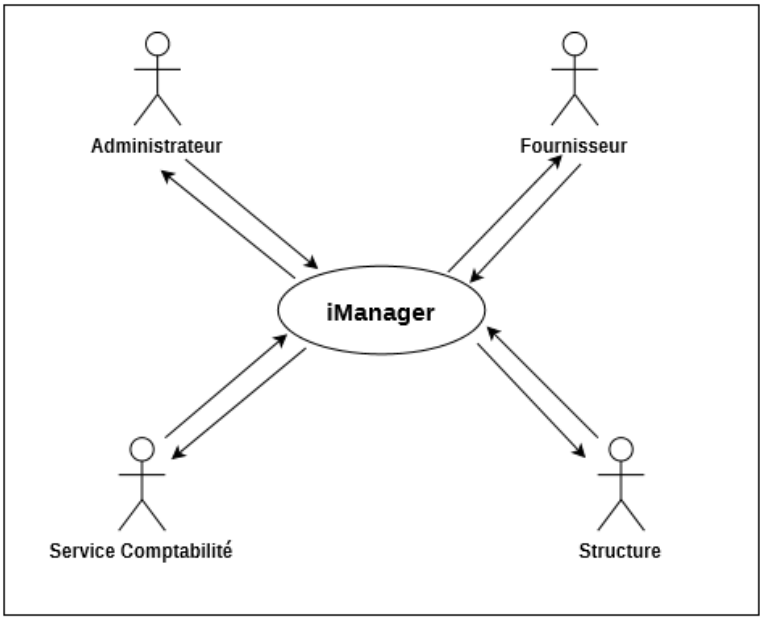


FIGURE 2.1 – Présentation de diagramme de contexe du système à réaliser.

2.2.4 Capture des besoins fonctionnels

Cette phase représente un point de vue « fonctionnel » de l'architecture du système à realiser. Nous allons identifier dans ce qui suit les cas d'utilisations, les différents acteurs associés et leurs interactions avec le système. Tableau 2.2 résume les différents cas d'utilisation.

N°	**Cas d'utilisations**		Acteur
1	Authentification		- Administrateur. - Structure. - Service comptabilité. - Fournisseur.
2	Gestion des utilisateurs	Ajouter	- Administrateur
		Modifier	
		Supprimer	
3	Gestion des droits d'accès		- Administrateur
4	Afficher la liste des structures		- Administrateur
5	Gestion des fournisseurs	Ajouter	- Administrateur
		Modifier	
		Supprimer	
6	Suivre les documents dans le workflow		- Administrateur. - Structure. -Service comptabilité.
7	Gestion des contrats	Créer	- Fournisseur. - Structure.
		Modifier	
		Supprimer	
8	Traiter un contrat		- Service comptabilité.
9	Transmettre un contrat		-Structure. - Service comptabilité.
10	Rejeter un contrat		- Service comptabilité.
11	Afficher la liste des contrats		- Administrateur.

2.2.5 Les diagrammes de cas d'utilisation

— **Le diagramme de cas d'utilisations « Administrateur »**

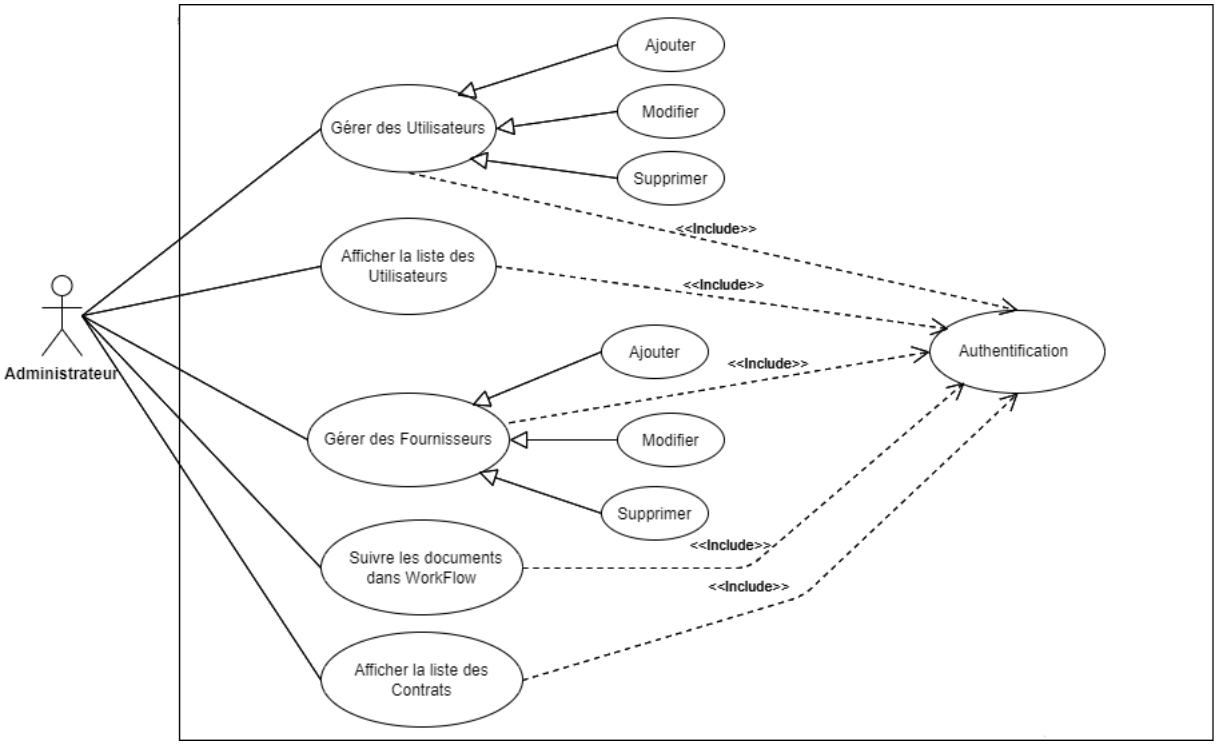


FIGURE 2.2 – Diagramme du cas d'utilisations associé à « administrateur ».

— **Le diagramme de cas d'utilisations « Structure »**

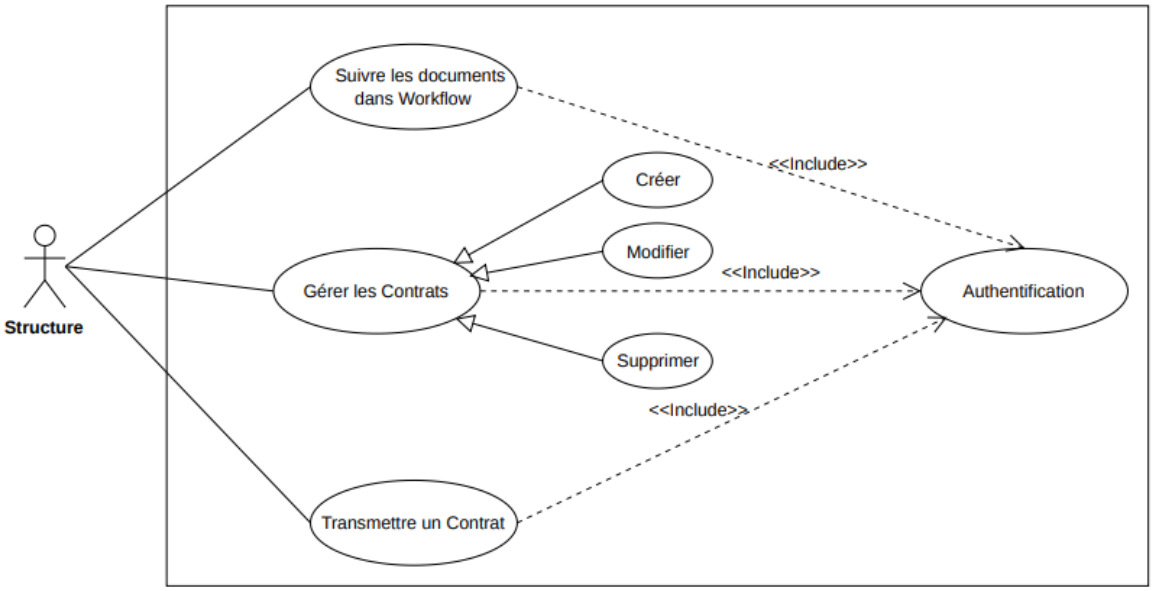


FIGURE 2.3 – Diagramme du cas d'utilisations associé à « Structure ».

— **Le diagramme de cas d'utilisations «Service comptabilité »**

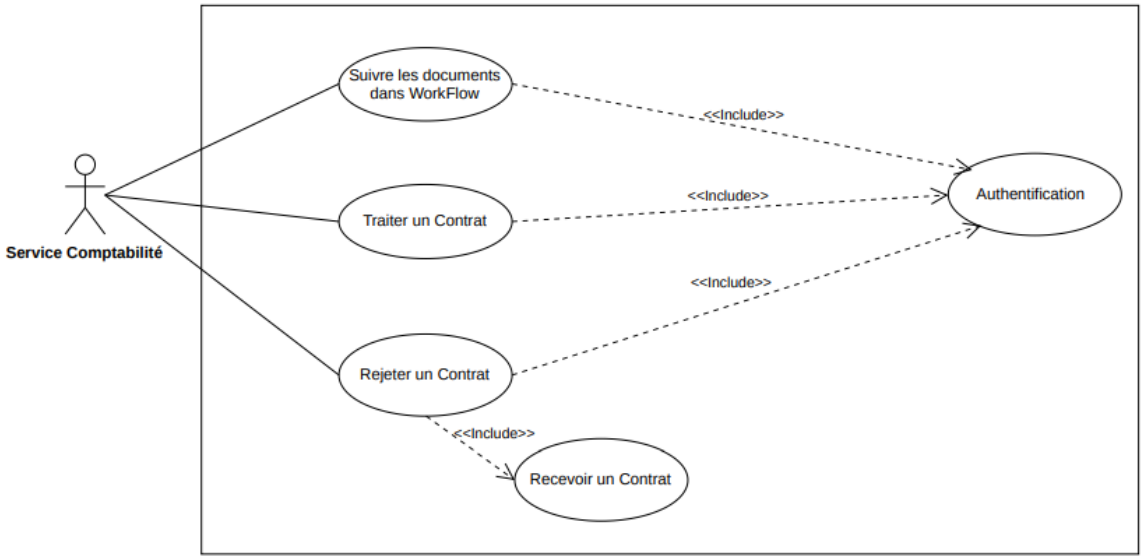


FIGURE 2.4 – Diagramme de cas d'utilisations associé à « Service comptabilité ».

— **Le diagramme de cas d'utilisations « Fournisseur »**

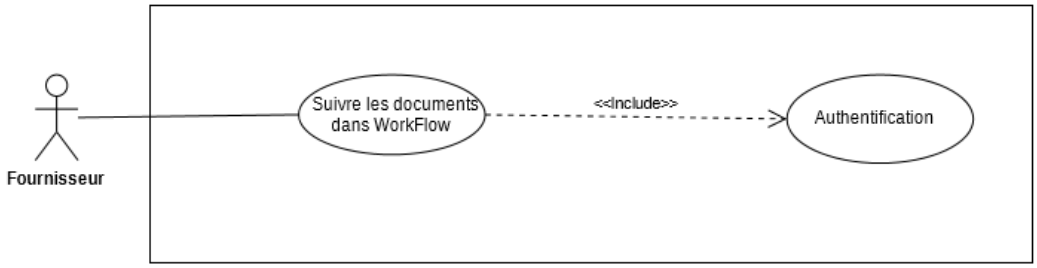


FIGURE 2.5 – Diagramme de cas d'utilisations associé à « Fournisseur »

2.2.6 Description textuelle des cas d'utilisation

Nous allons maintenant donner une description de chaque cas d'utilisation.

— **Cas d'utilisation authentification**

Cas d'utilisation N°1	**Authentification**
Objectif	Ce cas permet à l'utilisateur d'accéder à ses privilèges au niveau du système
Acteur	Administrateur, Structure, Service Comptabilité, Fournisseur
Précondition	L'utilisateur doit avoir un compte
Scenario nominal	1. Le système affiche la fenêtre d'authentification. 2. L'utilisateur introduit ses paramètres de connexion. 3. Le système confirme l'authentification.
Exception	Lors de l'étape trois des flux de base le système vérifier : -L'email ou le mot de passe (voir les deux) sont erroné donc l'accès au système sera refusé ; le système demande alors à l'utilisateur d'introduire à nouveau son email et son mot de passe.
Poste condition	L'utilisateur est authentifié et a accès au système

TABLE 2.3 – Déscription textuelle du Cas d'utilisation « Authentification ».

— **Cas d'utilisation gestion des utilisateurs :** Le cas d'utilisation « Gestion des utilisateurs » est caractérisé par trois scenarios suivants :
  — Ajouter un nouvel utilisateur.
  — Modifier un utilisateur.
  — Supprimer un utilisateur.

Cas d'utilisation N°2	**Ajouter d'un nouvel utilisateur.**
Objectif	Ce cas permet d'ajouter un nouvel utilisateur.
Acteur	Administrateur
Précondition	L'administrateur s'est bien authentifié.
Scenario nominal	1. Le système affiche le formulaire de saisie du nouvel utilisateur. 2. L'administrateur renseigne : Nom, Adresse email, Mot de passe, et valide. 3. Le système confirme l'ajout.
Scenario alternatif	Lors de l'étape trois des flux de base le système vérifie : -Si l'un des champs obligatoires n'est pas renseigné le système nous indique les champs manquant à remplir. - Que le mot de passe n'est pas utilisé - A tout moment l'administrateur peut annuler l'enregistrement .
Poste condition	Création d'un nouvel utilisateur.

TABLE 2.4 – Description textuelle du Cas d'utilisation « Ajouter un utilisateur ».

Cas d'utilisation N°3	**Modifier un utilisateur**
Objectif	Ce cas permet de modifier un utilisateur
Acteur	Administrateur
Précondition	L'administrateur s'est bien authentifié.
Scenario nominal	1. Le système affiche la liste de tous les utilisateurs. 2. L'administrateur choisi l'utilisateur à consulter ou à mettre à jour. 3. Le système affiche la fiche de l'utilisateur. 4. L'administrateur peut modifier les informations. personnelles, le profil ou les droits d'accès des utilisateurs existants et valide. 5. Le système confirme les mises à jour effectuées.
Scenario alternatif	Lors de l'étape deux l'administrateur : -Peut supprimer carrément l'utilisateur.
Poste condition	L'utilisateur concerné est modifié ou bien supprimé carrément.

TABLE 2.5 – Description textuelle du cas d'utilisation « Modifier un Utilisateur ».

— **Cas d'utilisation gestion des droits d'accès**

Cas d'utilisation N°4	**Gérer les droits d'accès**
Objectif	Ce cas permet de gérer les accès.
Acteur	Administrateur
Précondition	L'administrateur s'est bien authentifié.
Scenario nominal	1. L'utilisateur accède à la gestion des accès. 2. Le système affiche l'ensemble de privilège. 3. L'utilisateur choisi le privilège à consulter ou bien ajouter un nouveau privilège selon le besoin. 4. Le système confirme la mise à jour effectuée.
Exception	Lors de l'étape trois des flux de base qu'il n'y a pas de profils ayant le même code ou bien la même désignation.
Poste condition	Gestion des droits d'accès.

TABLE 2.6 – Description textuelle du cas d'utilisation « Gestion des droits d'accés ».

— **Cas d'utilisation afficher la liste des structures**

Cas d'utilisation N°5	**Afficher la liste de structures**
Objectif	Ce cas permet d'afficher la liste des structures.
Acteur	Administrateur
Précondition	L'administrateur s'est bien authentifié.
Scenario nominal	1. L'utilisateur demande l'affichage de liste de structure. 2. Le système affiche la liste de toutes les structures.
Poste condition	La liste de structures est affichée.

TABLE 2.7 – Description textuelle du cas d'utilisation « Afficher la Liste des structures ».

— **Cas d'utilisation gestion des fournisseurs**

Cas d'utilisation N°6	**Gérer les fournisseur**
Objectif	Ce cas permet de gérer les fournisseurs.
Acteur	Administrateur
Précondition	L'utilisateur s'est bien authentifié.
Scenario nominal	1. L'administrateur demande le formulaire de gestion de fournisseur. 2. Le système affiche le formulaire. 3. Remplir et envoyer (ou « modifier » pour la mise à jour ou « supprimer » pour la suppression).
Poste condition	Gestion des fournisseurs.

TABLE 2.8 – Description textuelle du cas d'utilisation « Gestion des fournisseurs ».

— **Cas d'utilisation suivre un document dans un workflow**

Cas d'utilisation N°7	**Suivre un document dans un workflow**
Objectif	Ce cas permet de suivre les contrats
Acteur	Administrateur, structure, service comptabilité, fournisseur.
Précondition	L'utilisateur s'est bien authentifié.
Scenario nominal	1. L'utilisateur demande de suivre les étapes de contractualisation. 2. Le système affiche les étapes .
Poste condition	Le document est suivi.

Table 2.9 – Description textuelle du cas d'utilisation « Suivre un decument dans un workflow ».

— **Cas d'utilisation gestion de Contrats :** Le cas d'utilisation « gestion des Contrats » est caractérisé par 3 scenarios suivants :
  — Créer un Contrat.
  — Modifier un Contrat.
  — Supprimer un Contrat.

Cas d'utilisation N°8	**Créer un contrat**
Objectif	Ce cas permet de créer un contrat.
Acteur	Structure
Précondition	L'utilisateur s'est bien authentifié.
Scenario nominal	1. Le système affiche un formulaire de saisie de contrat. 2. L'utilisateur rempli les informations et valide. 3. Le système confirme l'enregistrement.
Poste condition	Un noveau contrat ajouté.

Table 2.10 – Description textuelle du cas d'utilisation « Créer un contrat ».

Cas d'utilisation N°9	**Modifier un contrat**
Objectif	Ce cas permet modifier ou supprimer un contrat.
Acteur	Structure
Précondition	L'utilisateur s'est bien authentifié.
Scenario nominal	1. Le système affiche la liste des contrats. 2. L'utilisateur sélectionne le contrat à modifier ou à supprimer. 3. Le système confirme la mise à jour (la suppression).
Poste condition	Un contrat est modifié ou supprimé.

Table 2.11 – Description textuelle du cas d'utilisation « Modifier un contrat ».

— **Cas d'utilisation traiter un contrat**

Cas d'utilisation N°10	**Traiter un contrat**
Objectif	Ce cas permet de traiter un contrat.
Acteur	Service Comptabilité
Précondition	L'utilisateur s'est bien authentifié.
Scenario nominal	1. L'utilisateur sélectionne le contrat à traiter. 2. Vérifier la conformité de contrat.
Poste condition	Le contrat est vérifié.

Table 2.12 – Description textuelle du cas d'utilisation « Traiter un contrat ».

— **Cas d'utilisation transmettre un contrat**

Cas d'utilisation N°11	**Transmettre un contrat**
Objectif	Ce cas permet de transmettre un contrat.
Acteur	Structure, Service comptabilité
Précondition	L'utilisateur s'est bien authentifié.
Scenario nominal	1. L'utilisateur sélectionne le contrat et valide. 2. Le système confirme la transmission de demande.
Poste condition	Le contrat est transmis.

Table 2.13 – Description textuelle du cas d'utilisation « Transmettre un contrat ».

— **Cas d'utilisation rejeter un contrat**

Cas d'utilisation N°12	**Rejeter un contrat**
Objectif	Ce cas permet de rejeter un contrat
Acteur	Service comptabilité.
Précondition	L'utilisateur s'est bien authentifié.
Scenario nominal	1. L'utilisateur sélectionne le contrat à rejeter et valide. 2. Le système confirme le rejet de contrat et envoie une notification à la structure avec un motif de refus.
Poste condition	Le contrat est rejeté.

Table 2.14 – Description textuelle du cas d'utilisation « Rejeter un contrat ».

— **Cas d'utilisation afficher la liste des contrats**

Cas d'utilisation N°13	**Afficher la liste des contrats**
Objectif	Ce cas permet d'afficher la liste des contrat.
Acteur	Administrateur
Précondition	L'administrateur s'est bien authentifié.
Scenario nominal	1. L'utilisateur demande l'affichage de liste des contrats. 2. Le système affiche la liste de tous les contrats.
Poste condition	La liste des contrats est affichée.

TABLE 2.15 – Description textuelle du cas d'utilisation « Afficher la liste des contrats ».

2.3 Diagrammes de séquence

2.3.1 Définition

Un diagramme de séquence est une représentation graphique des interactions entre les acteurs et le système. Il montre les interactions entre les objets du point de vue temporel ainsi il décrit de scénarios du diagramme du cas d'utilisation [4]

Concepts principaux de diagramme de la séquence :

Une ligne de vie : représente un participant à une interaction (objet ou acteur)

Les messages : représenté par une flèche munie d'un texte, c'est un outil de dialogue entre les objets (utilisateur, système).

Flèche continue : demande ou requête (message d'envoi)

Flèche discontinue : réponse (message de retour).

— **Diagramme de séquence du cas d'utilisation « Authentification »**
Pour accéder à son espace de travail, l'utilisateur entre son login et son mot de passe et valide l'opération, le système effectue une vérification. Si l'un des champs est vide ou incorrecte un message d'erreur est envoyé, sinon, l'interface correspondante à l'utilisateur s'affiche.

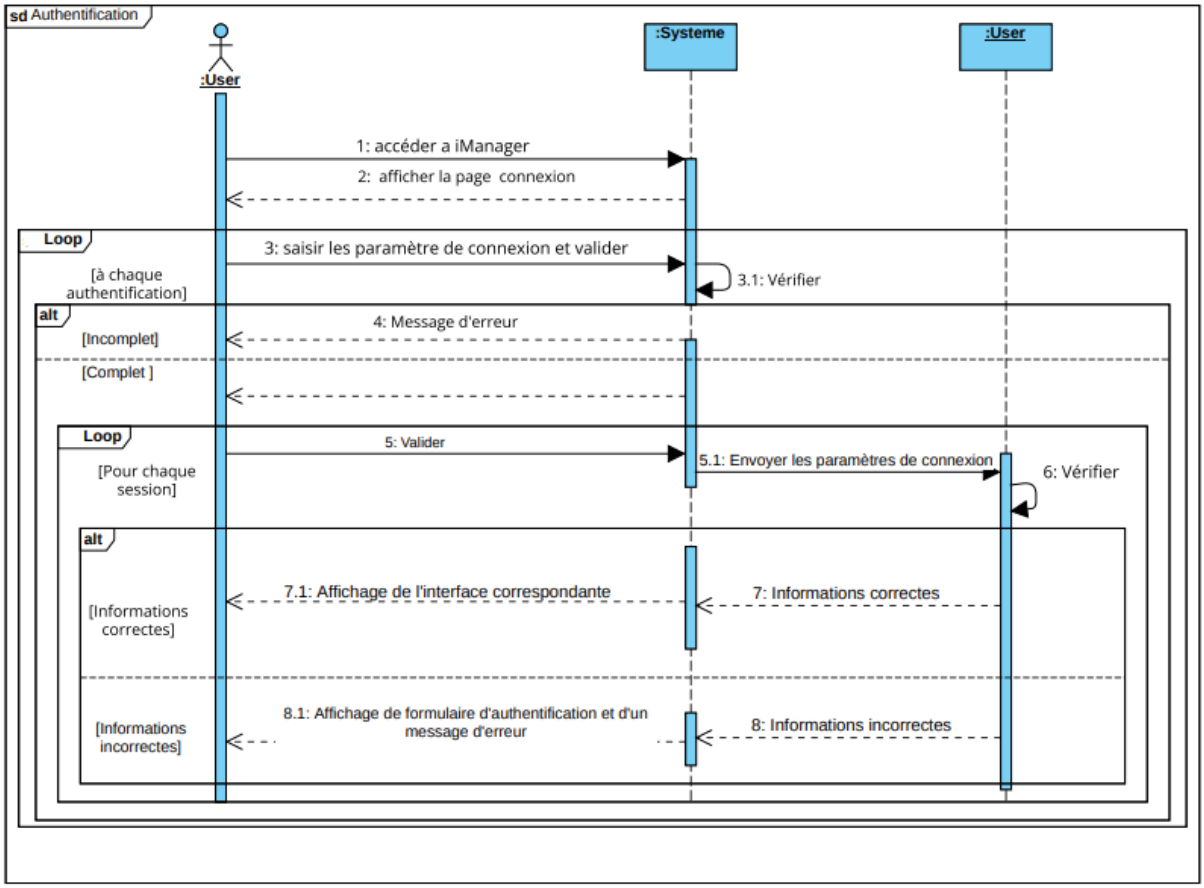


FIGURE 2.6 – Diagramme de séquence du cas d'utilisation « Authentification ».

— **Diagramme de sequence du cas d'utilisation « Gestion des contrats »**

La figure 2.7 représente le diagramme de sequence « Gestion des contrats » :

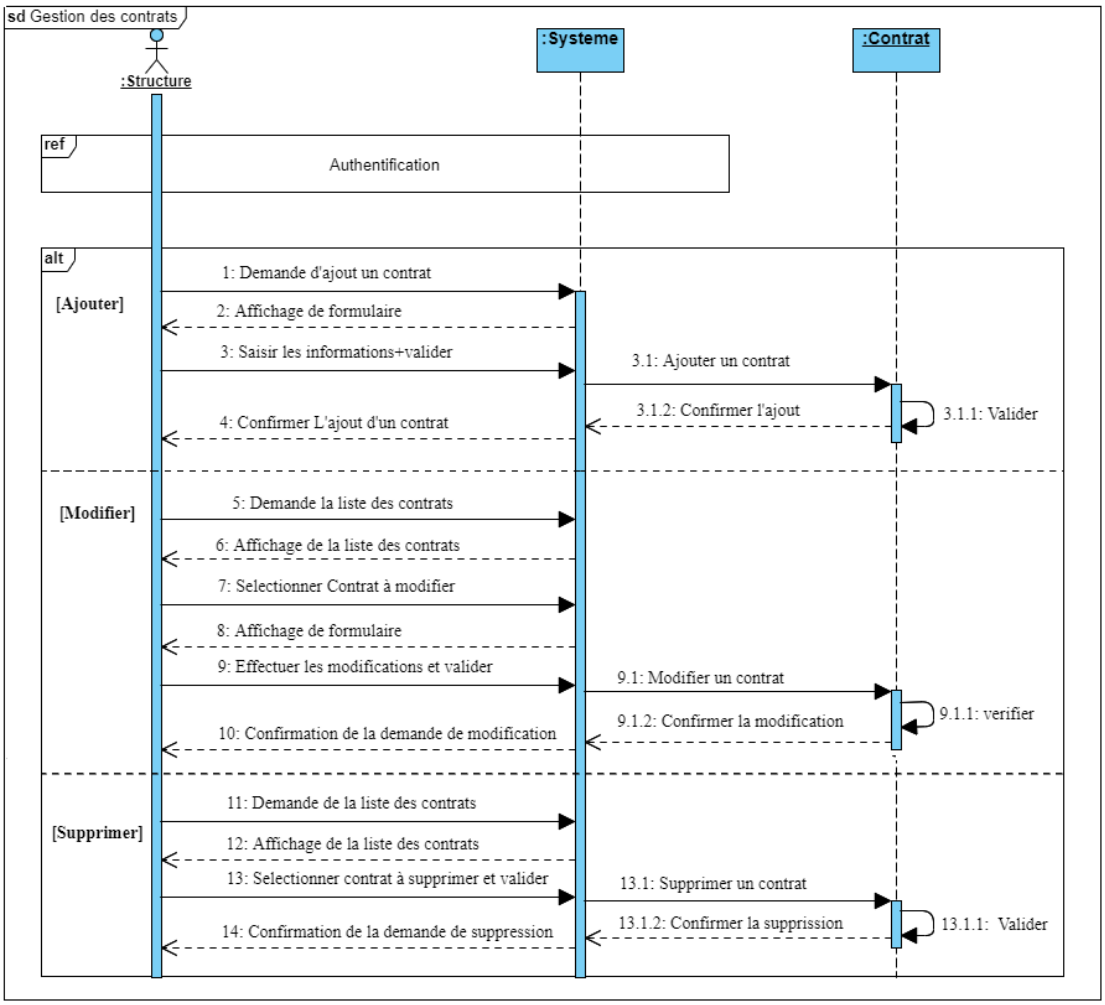


FIGURE 2.7 – Diagramme de séquence du cas d'utilisation « Gestion des contrats ».

— **Diagramme de sequence du cas d'utilisation « Gestion des utilisateurs »**

La figure 2.8 représente le diagramme de sequence « Gestion des utilisateurs » :

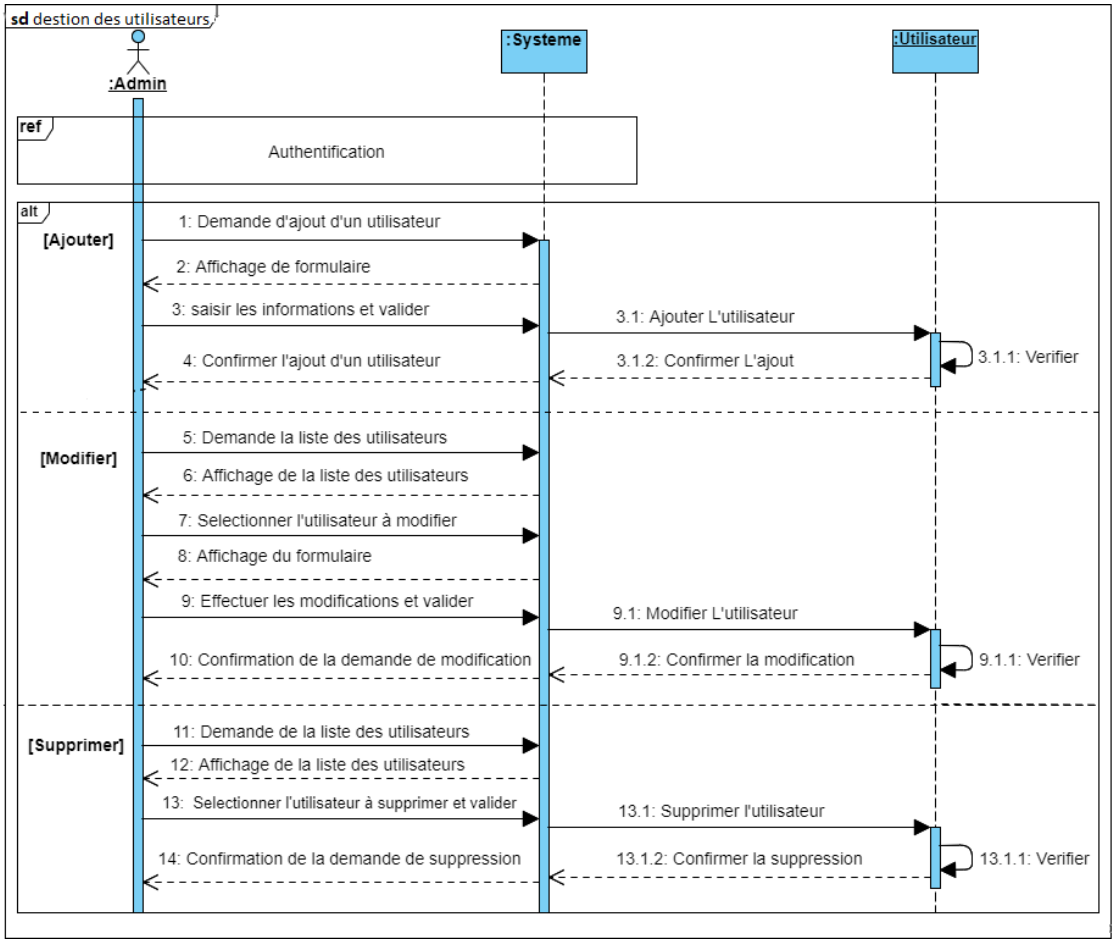


FIGURE 2.8 – Diagramme de séquence « Gestion des utilisateur ».

— **Diagramme de séquence du cas d'utilisation « Transmettre un contrat »**
La figure 2.9 représente le diagramme de séquence « Transmettre un contrat » :

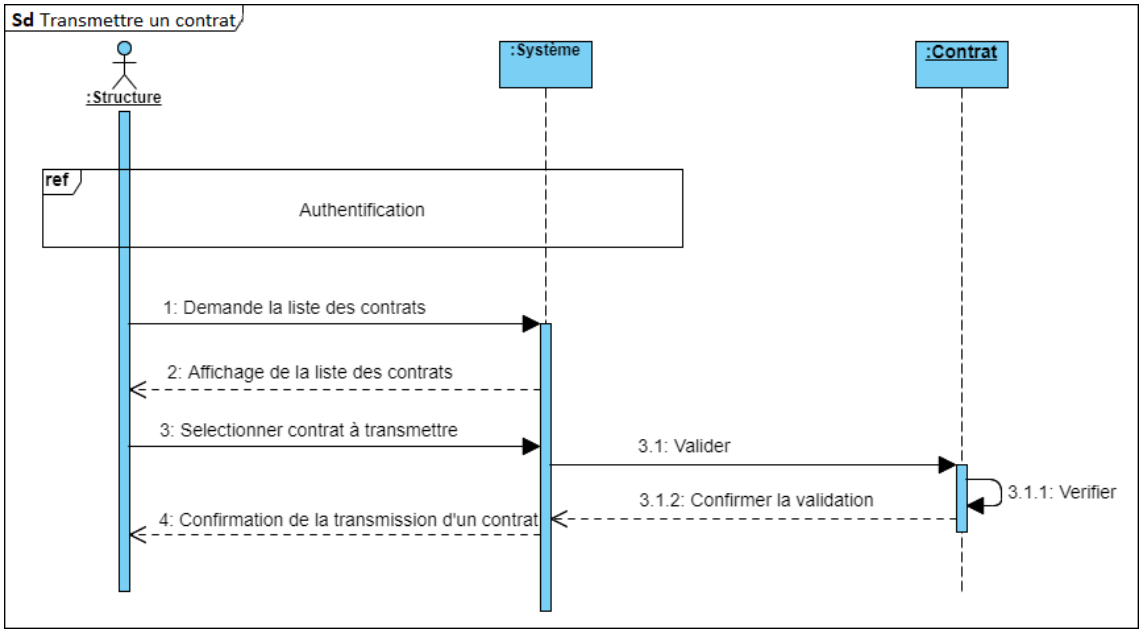


FIGURE 2.9 – Diagramme de Sequence « Transmettre un contrat ».

2.4 Capture des besoins non fonctionnels

Les besoins non fonctionnels concernent les contraintes à prendre en considérations pour mettre en place une solution adéquate aux attentes des concepteurs des architectures dynamiques Notre application doit nécessairement assurer les besoins suivants :

L'extensibilité : l'application devra être extensible c'est dire qu'il pourra avoir une possibilité d'ajouter ou de modifier des fonctionnalités.

La sécurité : l'application devra être hautement sécurisée, les informations ne devront pas être accessibles à tout le monde c'est-à-dire que l'application est accessible par un identificateur et un mot de passe attribuer à une personne physique.

L'interface : avoir une application qui respecte le principe de l'interface homme/machine tel que l'ergonomie et la fiabilité.

La performance : l'application devra être performant c'est-à-dire que le système doit réagit dans un délai précis quel que soit l'action de l'utilisateur.

La convivialité : l'application doit être simple et facile à manipulée même par les non Experts.

L'ergonomie : le thème adopté par l'application doit être inspiré des couleurs et du logo de type de l'entreprise d'accueil.

2.5 Conclusion

Ce chapitre nous a permis de couvrir en détails les besoins de notre système, nous avons fourni une analyse détaillée de ces besoins grâce aux diagrammes des cas d'utilisations et diagrammes de séquences. Nous passerons maintenant au chapitre qui suit à la phase de conception, nous allons élaborer le diagramme de classe ainsi que le modèle relationnel de données qui nous permet d'avoir le schéma de la base de données de notre l'application .

CHAPITRE 3

CONCEPTION

3.1 Introduction

La phase de conception est une phase critique lors du processus de la réalisation de notre application. Elle utilise les informations collectées lors des étapes précédentes pour établir l'architecture de la future application. Dans cette étape détaillée nous allons élaborer le diagramme des classes qui est essentiel à l'implémentation de notre application.

3.2 Définition du diagramme de classe

Les diagrammes de classes sont les diagrammes les plus utilisés, ils font partie du diagramme de structure représentent la structure interne du système. Un diagramme de classes décrit le type des objets qui composent le système ainsi les différentes relations entre eux, c'est le diagramme le plus important de la modélisation orienté objet (UML) [10].

3.2.1 Les éléments constituants d'un diagramme de classe

Dans ce qui suit, nous définissons les concepts utilisés dans un diagramme de classe[10] :

Objet : est une entité aux frontières bien définies. Il possède une identité et encapsule un état et un comportemet. Un objet est une instance (ou occurrence) d'une classe.

Classe : Description formelle d'un ensemble d'objet ayant une sémantique et des caractéristiques communs (un objet est une instance de class) représenter par un rectangle divisé en trois champs nom de classe, attributs et méthode .

Opération : est une foction applicable aux objets d'un classe et permet de décrire le comportement de ces objets.

Méthode : troisième ligne d'une forme de classe. On les appelle aussi opérations ; elles apparaissent sous forme de liste, chaque opération occupant une ligne différente.

Attribut : deuxième ligne d'une forme de classe. Chaque attribut de la classe apparaît sur une ligne distincte.

Classe-association : Permet de décrire soit des attributs, soit des opérations propres à l'association.

Relation : il existe plusieurs types de realtions entre les classes :

— **Association :** est une relation entre deux classes ou plus décrivant les connexions structurelles entre leurs instances, représentée par un trait (relier a ou moins deux classes) muni du nom de la relation et la cardinalité de chaque classe participante .

— **Agrégation :** si en enlève la classe composée on n'enlève pas la classe composante

— **La composition :** si on enlève la classe composée on doit enlever la classe composante .

— **Généralisation :** modélise la relation d'héritage .la classe héritée prend tous les attributs et les méthodes de la classe mère .

3.3 Diagramme de classes de l'application à réaliser

3.3.1 Dictionnaire de données

Dans ce qui suit, nous allons décrire les différentes classes schématisées dans le tableau. Cette description sera présentée sous forme d'un tableau, comme présenté dans la table ci-dessous.

Classe	**Attribut**	**Définition de l'attribut**	**Type**	**Méthodes**
Utilisateur	id_utilisateur	Identifiant de l'utilisateur	Int	Ajouter() Modifier() Supprimer()
	nom	Nom de l'utilisateur	Varchar	
	email	Adresse email de l'utilisateur	Email	
	mot_de_passe	Mot de passe de l'utilisateur	Password	
Contrat	id_contrat	Identifiant de contrat	Int	Ajouter() Modifier() Supprimer() Transmettre()
	type_contrat	Type de contrat	Varchar	
	date_debut	Date début de contrat	Date	
	date_fin	Date fin de contrat	Date	
	nature_achat	Nature d'achat	Varchar	
	devise	Devise de paiement(euro, dollars, dinars)	Varchar	
	delai_paiement	Le délai de paiement	Int	
	objet_contrat	L'objet de contrat	Varchar	
	montant	Le Montant de la facture	Float	
	montant_avancé	Le Montant avancé de la facture	Float	
	delai_preavis	Le délai préavis	Int	
	chargé_contrat	Le chargé de contrat	Varchar	
	chargé_contrat_dfc	Le chargé de contrat dfc	Varchar	
Fournisseur	id_fournisseur	L'identifiant de fournisseur	Int	Ajouter() Modifier() Supprimer()
	code_tière	Le code tière	Varchar	
	designation_tière	La désignation de fournisseur	Varchar	
	delai_paiement	Délai de paiement	Int	
	date_creation	Date de création de fournisseur	Date	
	date_modification	Date de modification de fournisseur	Date	
	utilisateur_create	L'utilisateur qui a créé le fournisseur	Varchar	
Structure	designation_structure	Désignation structure	Varchar	Ajouter() Modifier() Supprimer()
	entite	Entité de structure	Varchar	
	sous_entite	Sous entité de structure	Varchar	
Workflow	n°_piece	Numero de contrat a suivre	Int	
	source	Source de document	Varchar	
	destination	Destination de document	Varchar	

Liste deroulante	id_liste	L'identifiant de listes déroulantes	Int	Ajouter() Modifier() Supprimer()
	nom_liste	Nom de la liste déroulante	Varchar	
	desingnation_liste	Désignation de la liste déroulante	Varchar	
	designation$_s$tructure	Structure	Varchar	

TABLE 3.1 – Présentation des classes de l'application web à réaliser.

3.3.2 Diagramme de classe de l'application

la figure 3.1 présente le diagramme de classe de l'application à réaliser.

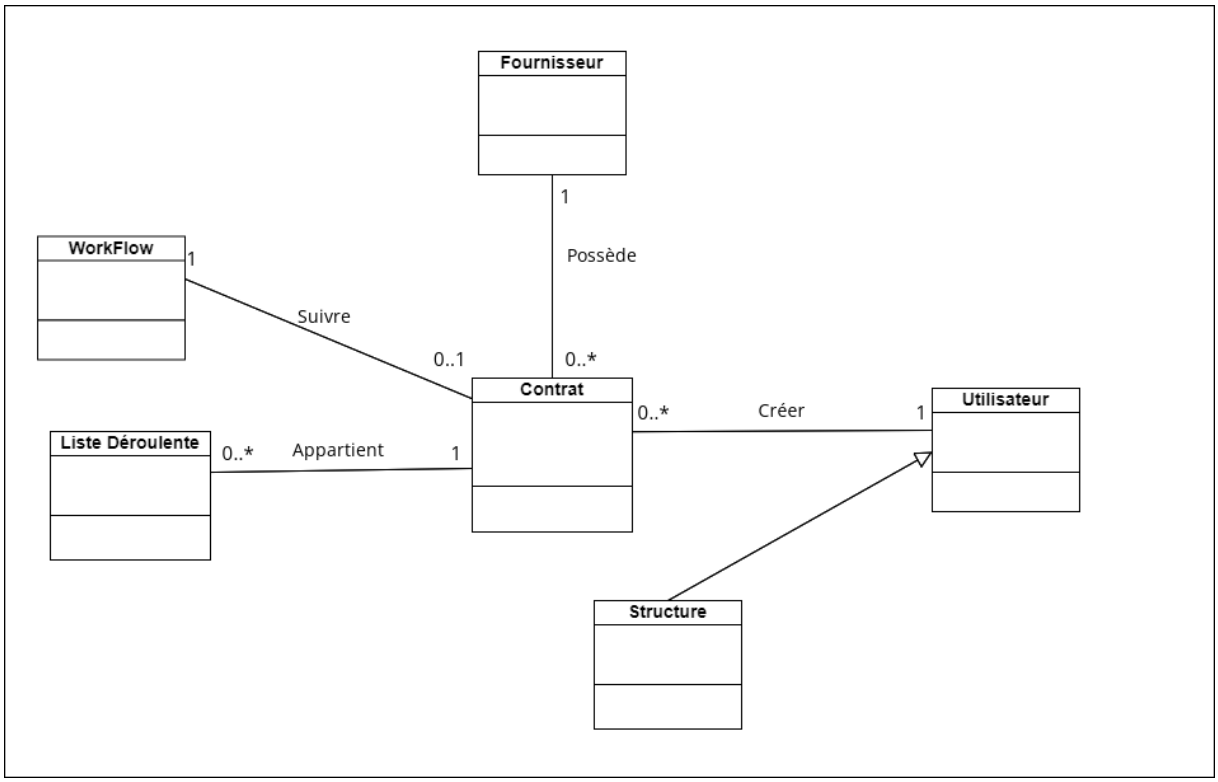


FIGURE 3.1 – Diagramme de classes.

3.4 Passage de l'UML au modèle relationnel

Pour pouvoir implémenter notre base de données nous allons réaliser le model relationnel à la base du diagramme de classes en s'appliquant les règles de passage suivantes [18] :

Règle1 : présence de la cardinalité (?..1) d'un côté de l'association

- Chaque classe se transforme en une table.
- Chaque attribut de classe se transforme en un champ de table.
- L'identifiant de la classe qui est associée à la cardinalité (?..1) devient une clé étrangère de l'autre

classe.

Règle2 : présence de (?..*) des deux côtés de l'association

- Chaque classe se transforme en une table.
- Chaque attribut de classe se transforme en un champ de table.
- L'association se transforme en une table. Cette table a comme champs l'identifiant de chacune des deux classes, plus d'éventuels autres attributs.

Règle3 : présence d'une généralisation.

- Créer une table avec tous les attributs des classes.
- Ajouter un attribut pour distinguer les types des objets.

3.4.1 Modèle relationnel

Le model relationnel est basé sur l'organisation des données sous forme de tables, elle est constituée d'un ensemble d'opération formelles sur les relations. L'opération relationnelle permettant de créer une nouvelle relation (table) à partir d'opération élémentaire sur d'autres tables.

• **Contrat**(id_contrat, type_contrat, date_debut, date_fin, nature_achat, devise, delai_paiement, objet_contrat, montant, montant_avance, delai_préavis, chargé_contrat, chargé_contrat_dfc, id_fournisseur, id_utilisateur, n°_pièce).

• **Liste_Déroulente** (id_liste, nom_liste, designation_liste, designation_structure, id_contrat).

• **Fournisseur** (id_fournisseur, code_tière, designation_tière, délai_paiement, date_création, date_modification, utilisateur_création).

• **Utilisateur** (id_utilisateur, nom, email, mot_de_passe) .

• **Structure**(id_utilisateur, designation_structure, entité, sous_entité).

• **WorkFlow**(id_workflow, source, destination).

3.5 Conclusion

Dans ce chapitre nous avous présenté le diagramme de classe utilisé pour concevoir notre application, cette étape nous a permis de finaliser la modélisation de notre application.

Nous sommes maintenant capables d'entamer la partie implémentation qui fera l'objet du chapitre suivant, nous allons présenter les outils et les environnements utilisés. Des captures d'écran sont ajoutés pour montrer les fonctionnalités de notre application.

CHAPITRE 4

RÉALISATION

4.1 Introduction

Après avoir conçu notre solution dans les parties précédentes, nous entamerons l'étape de la réalisation de l'application où nous allons présenter l'environnement de développement, l'architecture de déploiement et tous les choix techniques que nous avons adoptés pour sa réalisation.

4.2 Outils de développement

Pour mettre en place notre solution, nous nous sommes basés sur des technologies fiables, performantes et extensibles, afin de pouvoir satisfaire au mieux les besoins fonctionnels ainsi que les exigences en termes de qualité et de performance.

• **Visual paradigm online**

Visual Paradigm un outil de modélisation UML dans le cadre de programmation. Tout en un, il possède plusieurs options permettant une large possibilité de modélisation en UML. Laquelle offre de nombreux outils pour créer différents types de schémas comme les diagrammes cas d'utilisation, de classe ...etc. aussi il permet de générer des codes sources en divers langages comme le Java ou C++ à partir du modèle créé. Inversement, il permet de produire un modèle à partir de codes sources.

- **Visual studio code**

Visual Studio Code est un éditeur de code source léger mais puissant qui s'exécute sur votre bureau et est disponible pour Windows, macOS et Linux. Il est livré avec un support intégré pour JavaScript, TypeScript et Node.js et dispose d'un riche écosystème d'extensions pour d'autres langages (tels que C++, C, Java, PHP).
Parmi les forces de VS Code, nous trouvons l'intégration par défaut d'un terminal dans l'éditeur, ce qui permet de réaliser le projet via une et même fenêtre (ne plus avoir à basculer sur un invité de commande externe). Aussi, la capacité de déboguer les applications grâce à une console de débogage intégrée qui permet de résoudre les problèmes directement dans l'éditeur sans avoir recours aux navigateurs. De plus, Les utilisateurs peuvent effectuer plusieurs options telles que : modifier le thème et installer des extensions qui ajoutent des fonctionnalités supplémentaires[16].

- **WampServer**

WampServer est une platforme de développement web de type Wamp qui est l'acronyme de (Windows Apache MySQL, PHP), permettant de faire fonctionner localement (sans se connecter à un serveur externe) des scripts PHP. WampServer n'est pas en soi un logiciel mais un environnement comprenant deux serveurs (Apache et MySQL), un interpréteur de script (PHP), ainsi que phpMyAdmin pour l'administration Web des bases MySQL [14].

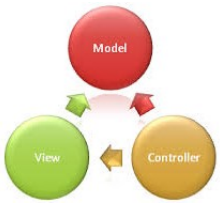
- **MySQL**

MySQL est un serveur de bases de données relationnelles open source qui fonctionne sous Windows et Linux. Il stocke les données dans des tables séparées au lieu de tout rassembler dans une seule table. Le serveur MySQL est très souple, rapide, facile et beaucoup plus simple à utiliser. De plus, il est performant et prévu pour fonctionner parfaitement avec PHP[21].

- **GitHub**

GitHub est une plateforme open source de gestion de versions et collaboration destinée aux développeurs de logiciels. Livrée en tant que logiciel à la demande, la solution GitHub a été lancée en 2008. Elle repose sur Git, un système de gestion de code open source créé par Linus Torvald dans le but d'accélérer le développement logiciel [11].

- **L'architecteur MVC**



Le Model View Controller (MVC) est une architecture désigne pattern qui décompose l'application en trois composants logiques principaux : le modèle, la vue et le contrôleur. Il est utilisé dans de très nombreux langages[15].

- **Laravel**

Laravel est un Framework PHP open-source qui respecte le principe MVC et entièrement développé en POO. Il est distribué sous licence MIT, avec ses sources hébergées sur GitHub. Nous avons adopté ce framework pour développer la partie logique (Back-end) en raison de ses capacités à fournir une sécurité de haut niveau. Il est considéré comme l'un des meilleurs framework PHP car il possède différentes bibliothèques telles que celle d'authentification, aussi, il dispose de nombreuses documentations [12].

- **Bootstrap**

Bootstrap est un Framework CSS gratuit et open-source (sous licence MIT) destiné au développement Web frontal réactif et mobile. Il contient des modèles de conception CSS et (éventuellement) JavaScript pour la typographie, les formulaires, les boutons, la navigation et d'autres composants d'interface.
Ce Framework est pensé pour développer des sites avec un design responsif, qui s'adapte à tout type d'écran, et en priorité pour les smartphones [13].

4.3 Description du diagramme de déploiement

Un diagramme de déploiement est une vue statique qui sert à décrit la disposition physique des ressources matérielles qui composent le système et montre la répartition des composants sur ces matériels. Ce diagramme précise comment les composants sont répartis sur les nœuds (ressources) et quelles sont les connexions entre les composants ou les nœuds. On a utilisé le diagramme de déploiement pour décrire l'architecture de notre application[1].

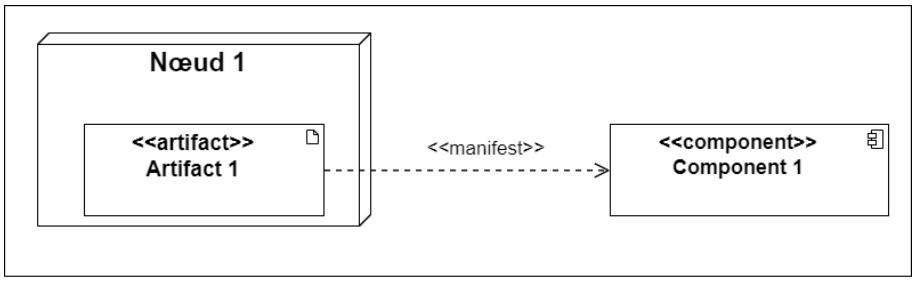


FIGURE 4.1 – Notation du diagramme de déploiement en UML.

4.3.1 Diagramme de déploiement de l'application réalisée

La figure 4.1 représente le diagramme de déploiement qui montre la disposition physique des ressources matérielles qui sont :

Serveur web Apache : permet d'accéder aux pages web à partir d'un navigateur existant deja sur l'ordinateur.

Serveur Base de données MySQL : permet de stocker les données de façon structuré, ces données sont gérés par un système de gestion de la base de données, ce système permet l'accès et les manipulations des informations présentes dans la base de données.

Machine utilisateur : permet d'envoie des requêtes vers le serveur web et afficher les requêtes reçues.

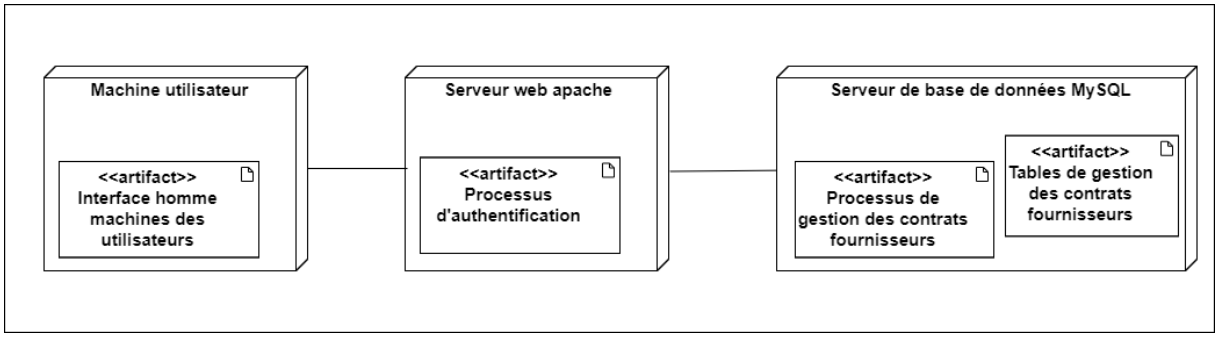


FIGURE 4.2 – Diagramme de déploiement de l'application web réalisée.

4.4 Présentation des interfaces du l'application réalisée

4.4.1 Page d'authentification

Cette interface s'affiche au lancement de notre application, les champs de saisie nous demandera d'introduire les paramètres de connexion. Cette étape met en valeur l'aspect sécurité de l'application.

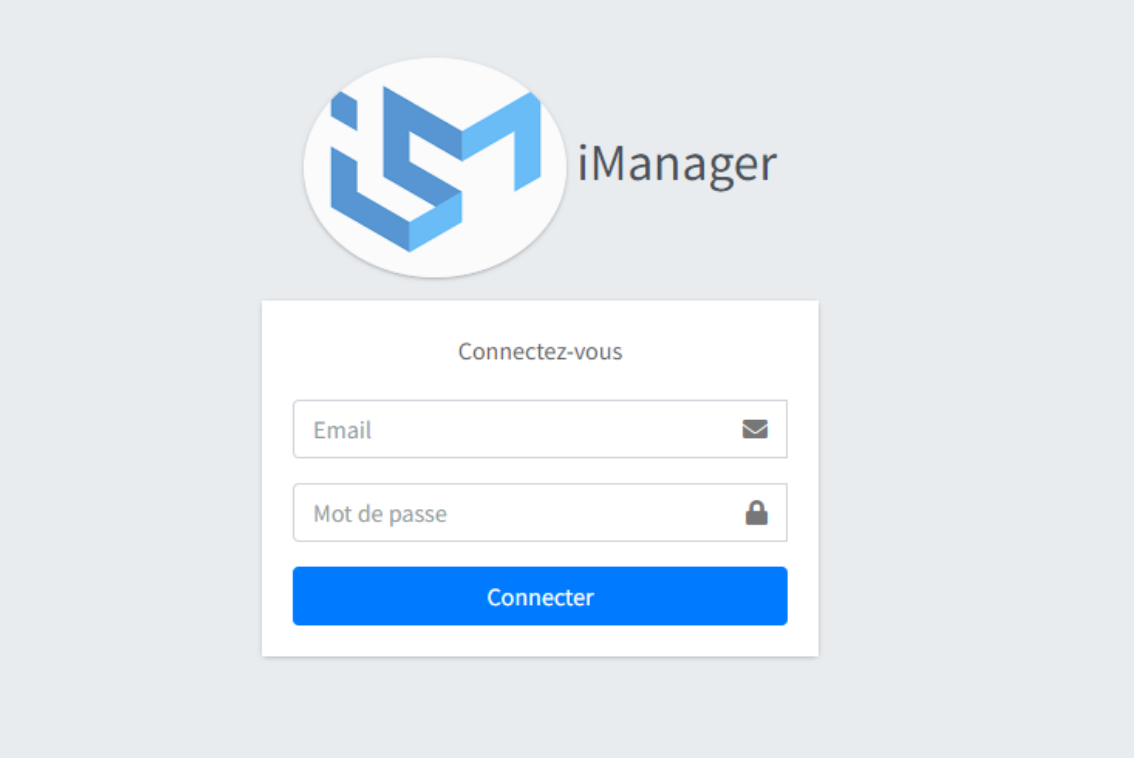


FIGURE 4.3 – Interface « Authentification ».

4.4.2 Page d'accueil

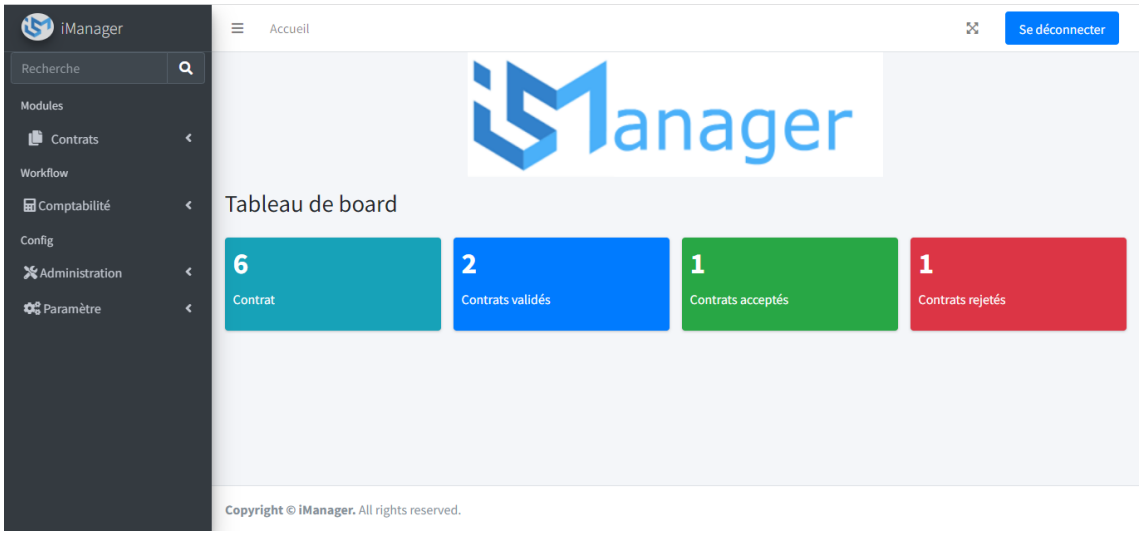


FIGURE 4.4 – Interface « Accueil ».

4.4.3 Interface d'ajout d'un contrat

L'interface suivante permet d'ajouter un nouveau contrat. Il lui suffira de remplir le formulaire et de rechercher les champs qu'il souhaite attribuer à un contrat dans le panel sélectionner cela en choisissant dans la liste déroulante.

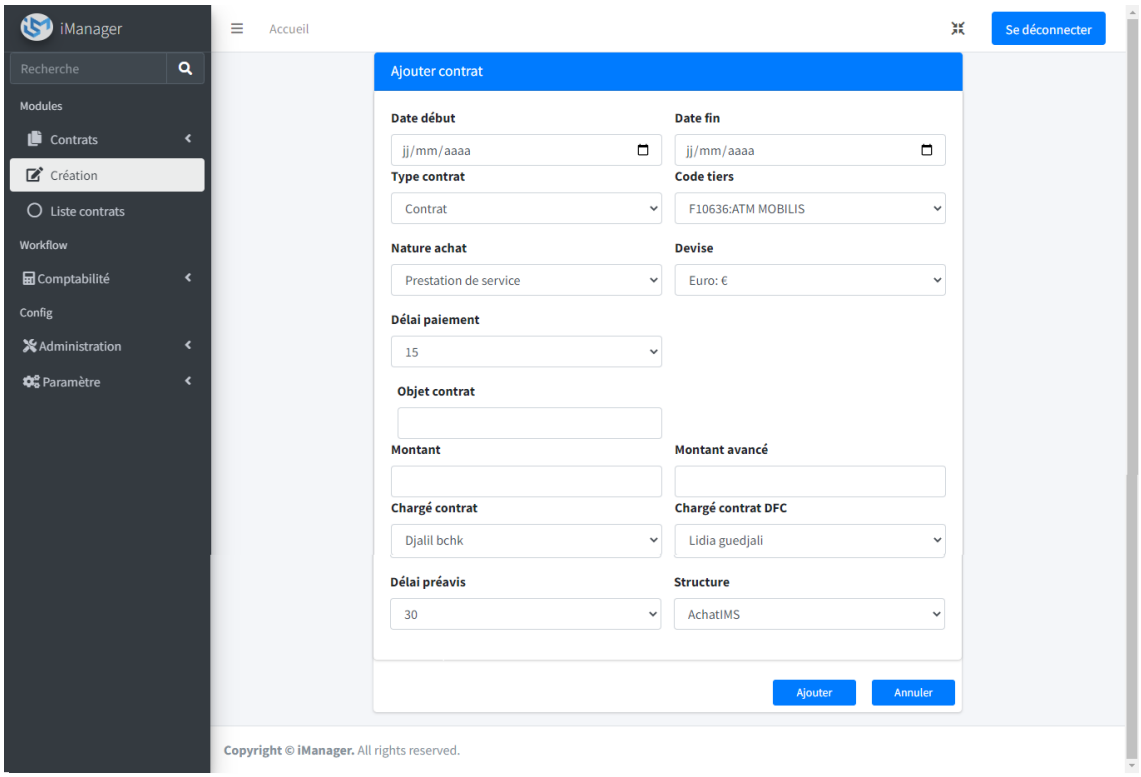


FIGURE 4.5 – Interface « Ajouter un contrat ».

4.4.4 Interface d'affichage de la liste des contrats

Cet interface affiche la liste des contrats avec la possibilité d'ajouter, modifier, supprimer ou de transmettre un contrat.
Le clic sur ce bouton "transmettre" nous affiche dans le champ statut les étapes workflow.

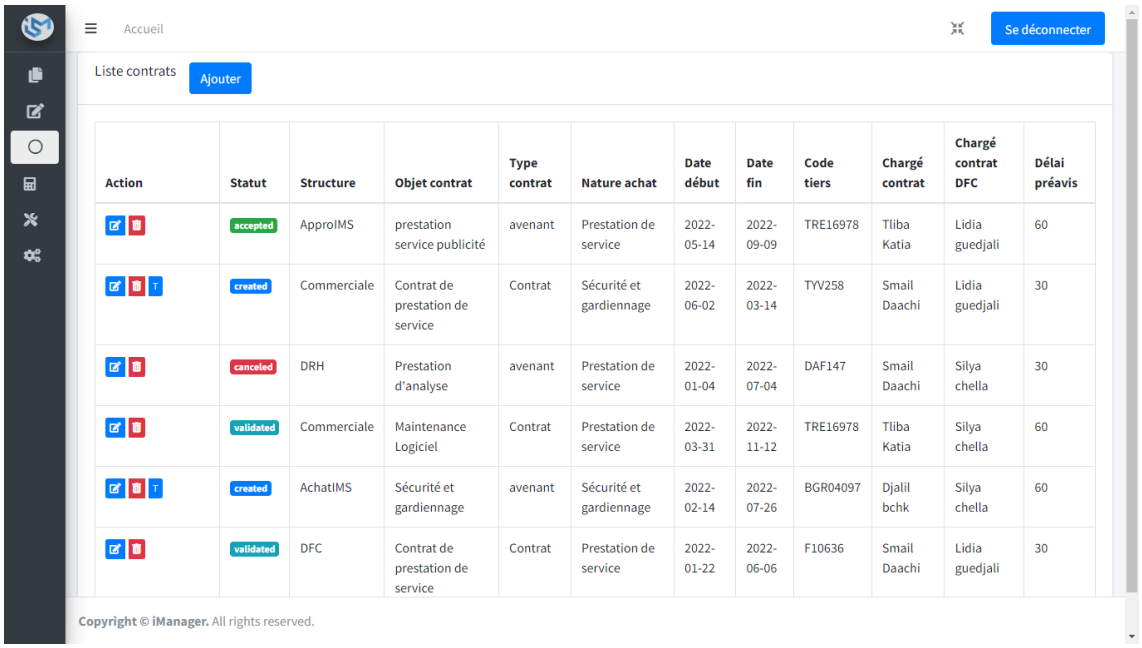


FIGURE 4.6 – Interface « Afficher la liste des contrats ».

4.4.5 Interface de modification d'un contrat

L'utilisateur a la possibilité de modifier un contrat.

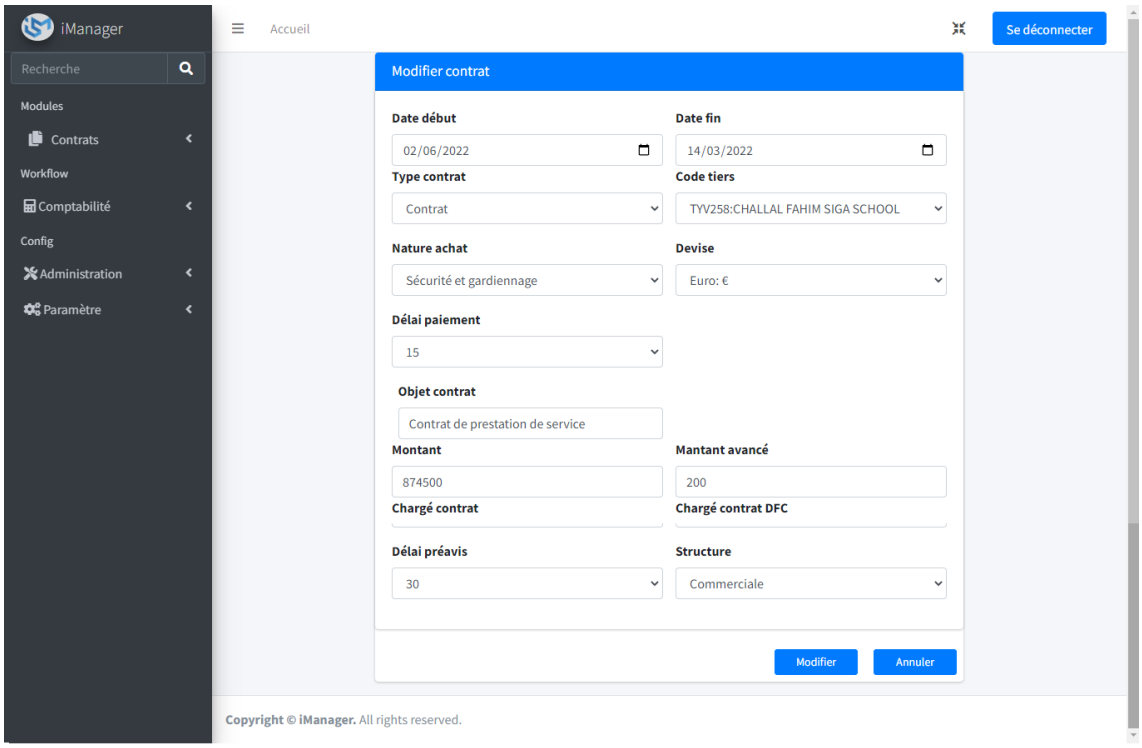


FIGURE 4.7 – Interface « Modifier un contrat ».

4.4.6 Interface de transsmission d'un contrat

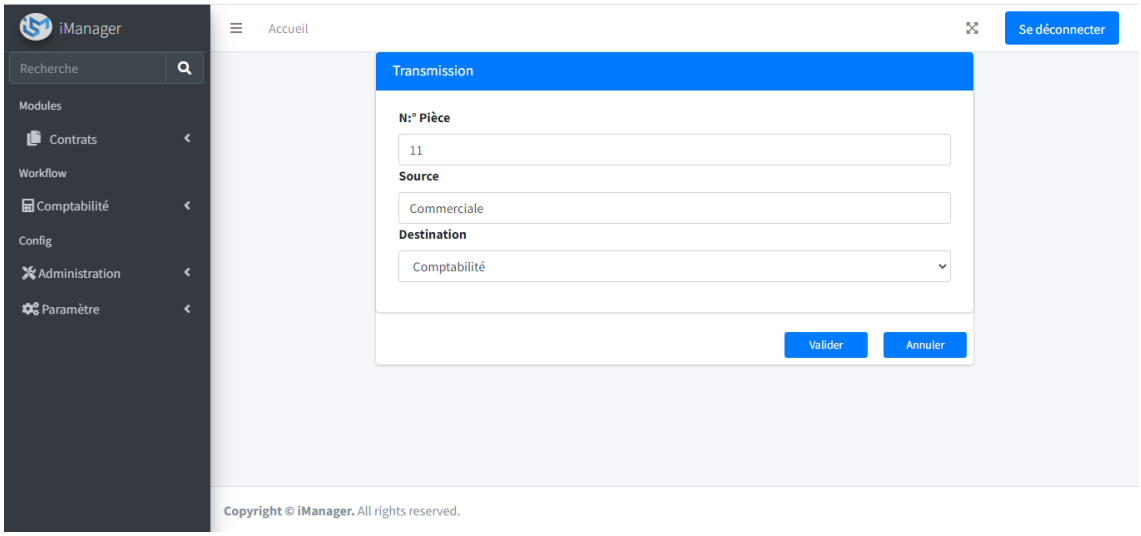


FIGURE 4.8 – Interface « Transmettre un contrat ».

4.4.7 Interface de service comptabilité

Grâce à cette interface, le service comptabilité peut voir uniquement la liste des contrats transmis avec la possibilité de valider ou rejeter un contrat.

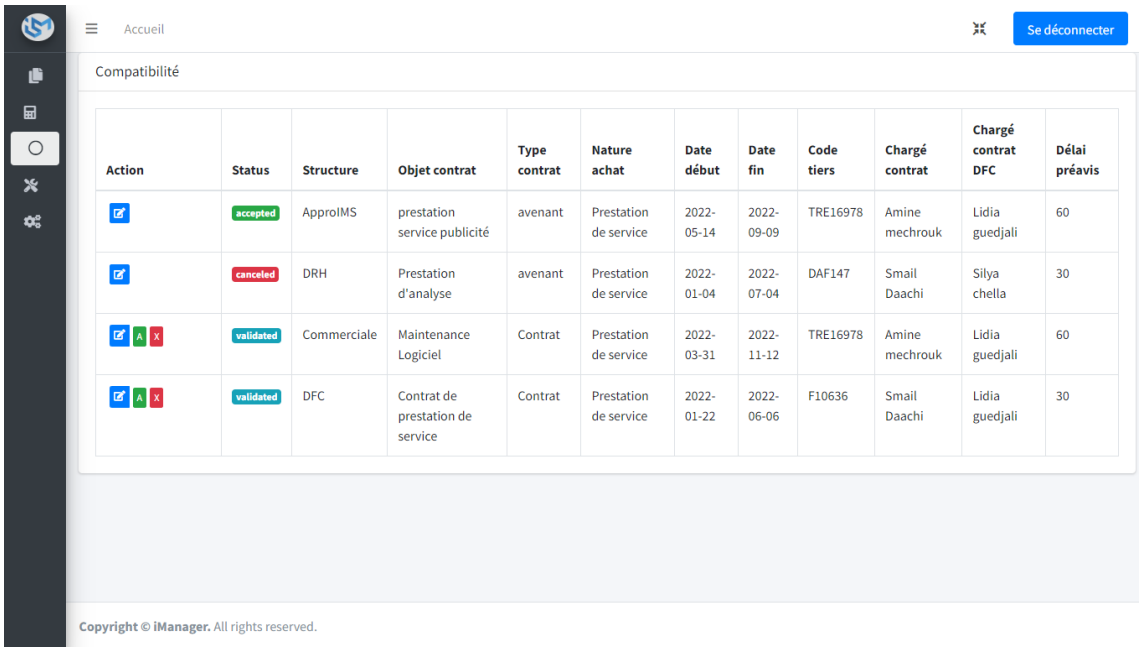


FIGURE 4.9 – Interface « Service comptabilité ».

4.4.8 Interface d'ajout d'un utilisateur

Cette interface permet à l'administrateur de l'application d'ajouter un nouveau utilisateur.

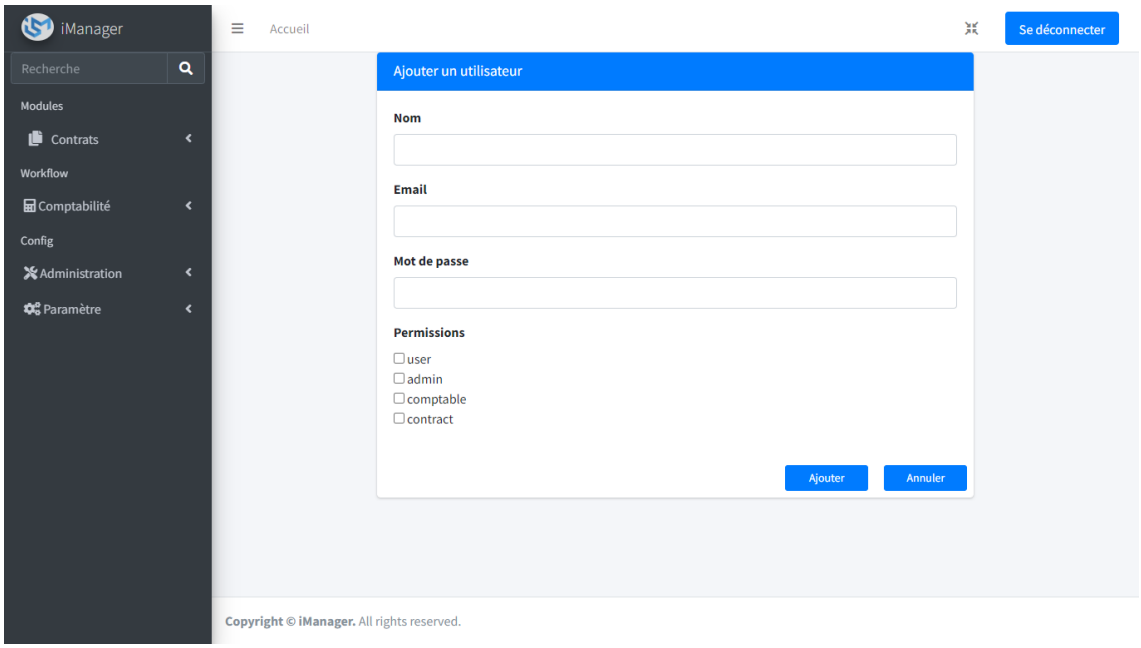


FIGURE 4.10 – Interface « Ajouter un utilisateur ».

4.4.9 Interface d'affichage de la liste des utilisateurs

Cette interface affiche la liste des utilisateurs, l'administrateur a la possibilité d'ajouter, mettre à jour ou modifier un utilisateur.

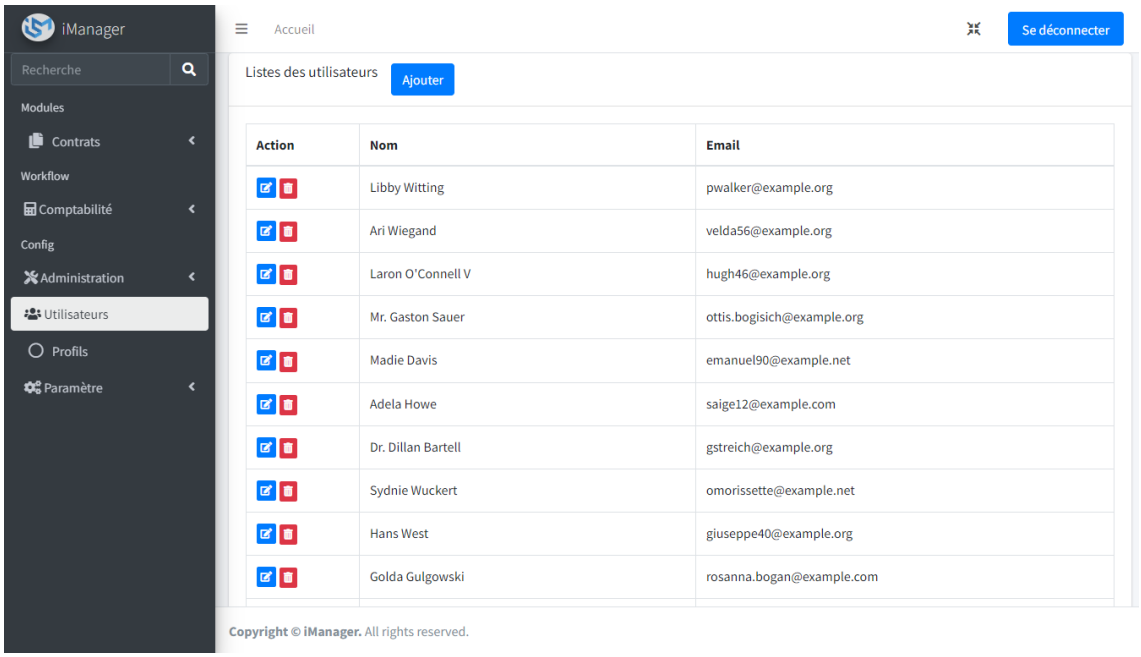


FIGURE 4.11 – Interface « Afficher liste des utilisateurs ».

4.4.10 Interface d'affichage de liste des fournisseurs

Le but de cette interface est d'afficher la liste de tous les fournisseurs.

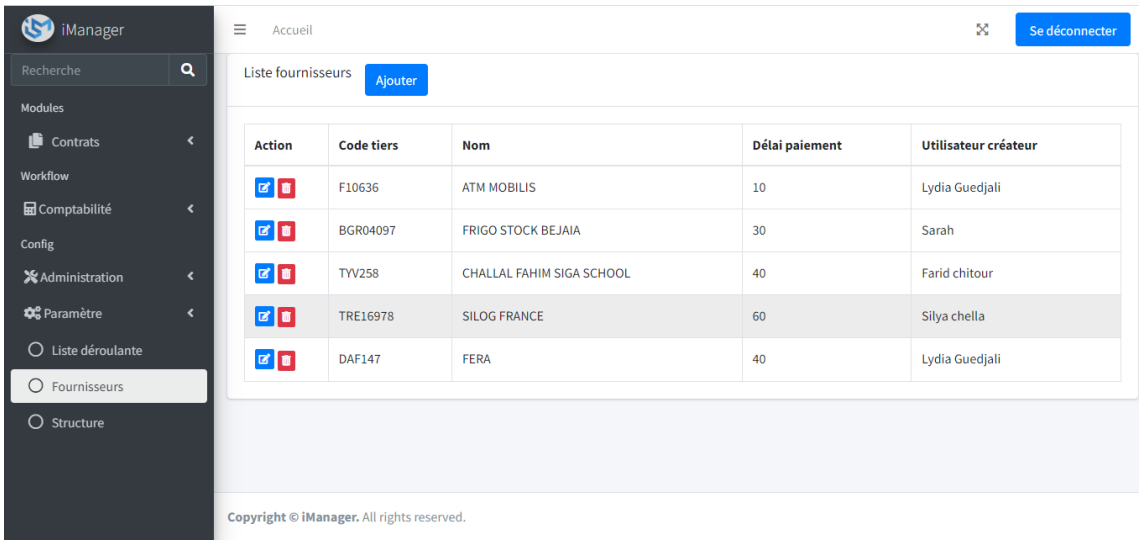


FIGURE 4.12 – Interface «Afficher liste des fournisseurs ».

4.4.11 Interface d'ajout d'un fournisseur

Cette interface permet à l'administrateur de l'application d'ajouter un nouveau fournisseur.

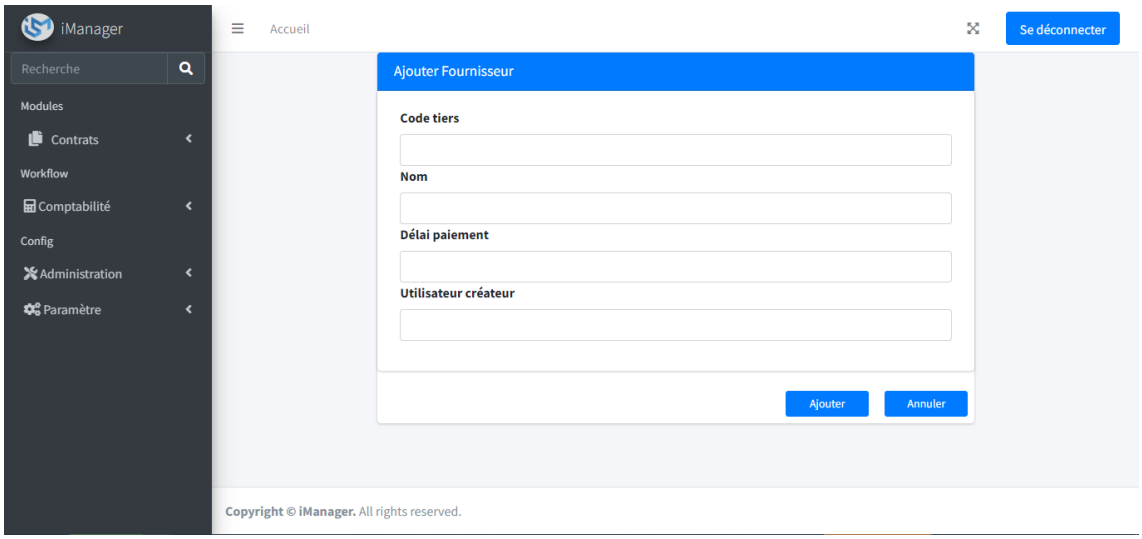


FIGURE 4.13 – Interface « Ajouter un fournisseur ».

4.4.12 Interface d'affichage de liste des structures

Cette interface affiche la liste des structures, l'administrateur a la possibilité d'ajouter, mettre à jour ou modifier une structure.

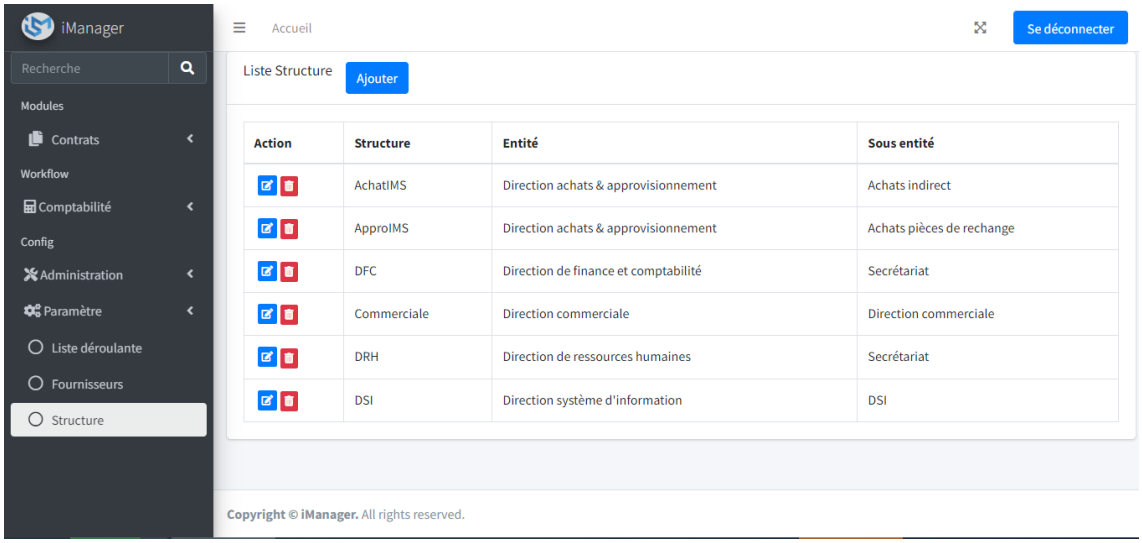


FIGURE 4.14 – Interface «Afficher liste des structures ».

4.4.13 Interface d'affichage de liste déroulantes

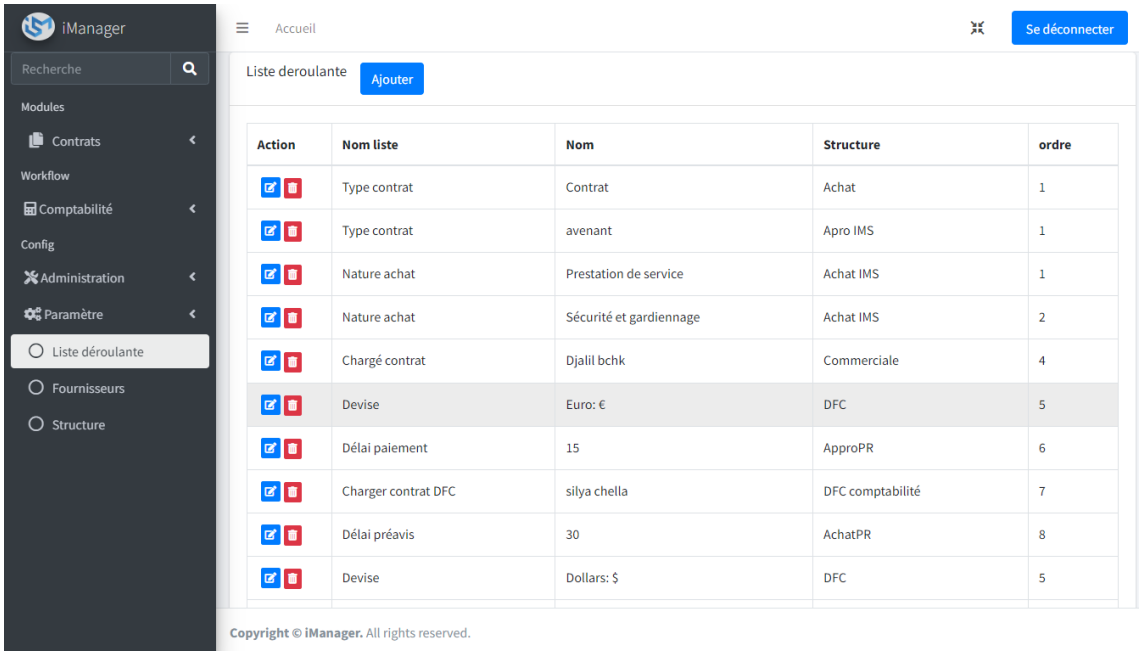


FIGURE 4.15 – Interface «Afficher la liste des déroulantes ».

4.5 Conclusion

Dans ce chapitre nous avons consacré la partie l'implémentation de notre application où nous avons présenté les différents outils de développements ainsi que l'environnement de travail utilisé. Par la suis nous avons présenté quelques interfaces graphiques de notre application.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Ce mémoire de fin d'étude représente un fruit de toute une période d'études et un fruit d'effort acquis pendant cette période. Elle peut être considérée comme une mise en pratique des connaissances théoriques acquises durant tout un cycle en réalisant une étude bien détaillée.
Le principal but de cette étude est et la mise en place d'une application web *iManager* pour la gestiont des contrats fournisseurs.

D'abord nous avons commencé par la présentation de l'organisme d'accueil ainsi que la méthodologie de conception. Après nous avons expliqué les solutions proposées. Nous avons modélisé notre application à l'aide du langage de modélisation UML en commençant par l'analyse des besoins et d'élaborer le diagramme de cas d'utilisation ainsi que les diagrammes des séquences. Ensuite, nous sommes passés à la conception en réalisant le diagramme de classes. Enfin nous sommes arrivés à présenter l'architecture de notre application à travers le diagramme de déploiement et à mettre en œuvre notre application.

Bien que nous ayons atteints les buts fixés au départ, notre application pourrait être enrichie par des fonctionnalités supplémentaires qui augmenteront l'efficacité de la gestion des contrats fournisseurs. Parmi ces améliorations, nous pouvons citer :pouvoir gérer d'autre documents dynamiquement. D'autre perspective que nous voudrions implémenter serait d'utiliser des API pour communiquer avec d'autre solutions pour mettre à jour des données de base (liste des fournisseurs, délais de paiement, liste du personnel, etc).

Enfin, nous pouvons dire que ce projet de fin d'étude est notre premier pas vers la vie professionnelle, ils nous a permis d'exploiter nos connaissances théoriques acquises pendant le cycle de notre formation, de nous familiariser avec un environnement dynamique et d'avoir une idée plus approfondie et plus pratique sur l'importance des systèmes d'informations dans les entreprises.

BIBLIOGRAPHIE

[1] Diagramme-de-deploiement-uml. [https://www.lucidchart.com/pages/fr/diagramme-de-deploiement-uml]..

[2] Cevital. [https://www.cevital.com/lhistoire-du-groupe/]., (Consulté le 01 04 2022).

[3] Cevitalentreprises. [https://www.cevitalentreprises.dz/index.php/fr/cevital-entreprises/presentation]., (Consulté le 01 04 2022).

[4] Radstudio. [https://docwiki.embarcadero.com/RADStudio/Sydney/fr/D%C3%A9finition_des_diagrammes_de_s%C3%A9quence_UML_1.5]., (Consulté le 04 05 2022).

[5] Diagrammes-uml. [https://gitmind.com/fr/types-diagrammes-uml.html]., (Consulté le 12 06 2022).

[6] Ditstream. [https://www.ummto.dz/dspace/bitstream/handle/ummto/12926/AmitoucheSabrina.pdf?sequence=1&isAllowed=y]., (Consulté le 12 06 2022).

[7] processus unifi. [https://www.academia.edu/39598956/Processus_Unifi%C3%A9_UP_and_2TUP]., (Consulté le 15 06 2022).

[8] Processus unifie. [https://www.rapport-gratuit.com/le-processus-unifie-up/]., (Consulté le 15 06 2022).

[9] Remy-manu. [http://remy-manu.no-ip.biz/UML/Cours/coursUML1.pdf]., (Consulté le 15 06 2022).

[10] remy-manu. [http://laurent\T1\textendashAudibert.devlopper.com/coursUML/?page=diagrammedeclasses]., (Consulté le 15 06 2022).

[11] Définition github. [https://www.lemagit.fr/definition/GitHub]., (Consulté le 16 06 2022).

[12] base-de-connaissances. [https://kinsta.com/fr/base-de-connaissances/qu-est-ce-que-laravel/]., (Consulté le 19 06 2022).

[13] developpeur. [https://www.journaldunet.com/web-tech/developpeur/1159810-bootstrap-definition-tutoriels-astuces-pratiques/]., (Consulté le 19 06 2022).

[14] wampserver. [https://www.africmemoire.com/part.6-chapitre-troisieme-presentation-de-lapplicatio
html]., (Consulté le 19 06 2022).

[15] L'architecture mvc. [https://www.irif.fr/~carton/Enseignement/InterfacesGraphiques/Cours/Swing/mvc.html#:~:text=L'architecture%20Mod%C3%A8le%2FVue%2F,r%C3%B4le%20pr%C3%A9cis%20dans%20l'interface.]., (Consulté le 20 06 2022).

[16] visualstudio. [https://code.visualstudio.com/docs]., (Consulté le 20 06 2022).

[17] Yann Thierry-Miege Benoit Charoux Aomar Osmani. *« UML2 pratique pour la modélisation 2éme édition », collection Synthex.* PhD thesis.

[18] Stéphane Crozat. Introduction au passage uml-relationnel : classes et associations. pages 3–8, 12 janvier 2018.

[19] DI GALLO Frédéric. Méthodologie des systèmes d'information - uml. *CNAM ANGOULEME*, pages 5–12, 2000-2001.

[20] Nathale Gaetner pierre Alain muller. « modilisation objet avec uml » édition eyrolles.

[21] Christian Soutou. Apprendre sql avec mysql. *ÉDITIONS EYROLLES 61, bd Saint-Germain 75240 Paris Cedex 05 www.editions-eyrolles.com*, Paris, 2006.

RÉSUMÉ

Une bonne gestion des contrats fournisseurs joue un rôle essentiel dans le cycle d'achat de toute entreprise, les conséquences d'une gestion inefficace des contrats fournisseurs sont multiples. Parmi ceux-ci figurent les risques procéduraux accrus, l'inflation des coûts, la non-conformité, l'augmentation ou le non-respect du délai de contractualisation, etc. C'est dans ce contexte que l'entreprise de Cevital de Bejaia souhaite mettre en place une gestion efficace des contrats fournisseurs qui couvre les principales phases de leur cycle de vie. Pour cela, il nous a été proposé de concevoir et réaliser une application web *« iManager »* assurant une gestion efficace des contrats fournisseurs. Pour ce faire, nous avons choisi de faire la modélisation avec le formalisme UML (Unified Modeling Language) par rapport à sa simplicité et ses performances dans la conception. Parmi les démarches de développement des logiciels existantes notre choix est basé sur le processus unifié (UP pour Unified Process). Ce dernier, est un processus de développement moderne, itératif, efficace sur des projets informatiques de toutes tailles. De plus, très complet, il couvre l'ensemble des activités, depuis la conception du projet jusqu'à la livraison de la solution. Quant à la conception des interfaces, nous avons utilisé le langage de programmation PHP via le framework "laravel", "Mysql" comme système de gestion de base de données (SGBD) et le framework Bootstrap.

Mots clés : Cevital ; Gestion des contrats ; Fournisseurs ; UML ; UP ; MySQL.

ABSTRACT

Good management of supplier contracts plays an essential role in the purchase cycle of any business, the consequences of ineffective management of supplier contracts are multiple. Among these are increased procedural risks, cost inflation, non-compliance, increase or non-compliance with the contractualization period, etc. It is in this context that the Cevital's company of Bejaia wishes to set up effective management of supplier contracts which covers the main phases of their life cycle. To do this, we have been offered to design and carry out a "iManager" web application ensuring effective management of supplier contracts. To do this, we have chosen to make the modeling with the UML (Unified Modeling Language) formalism compared to its simplicity and its performance in the design. Among the existing software development approaches our choice is based on the Unified Process (UP). The latter is a process of modern, iterative development, effective on computer projects of all sizes. In addition, very complete, it covers all activities, from the design of the project to the delivery of the solution. As for the design of interfaces, we used the PHP programming language via the "Laravel" framework, "MySQL" as a database management system (SGBD) and the Bootstrap framework.

Keywords : Cevital ; Contract management ; Suppliers ; UML ; UP ; MySQL.