



## **Filière de licence professionnelle**

Méthodes informatiques appliquées à la gestion des entreprises

Stage Professionnel

# **« *Application web pour la gestion du syndicat* »**

Réaliser par :

**BABA HAMOU Abdessadiq**

**Le 06/07/2022**

**Devant la commission :**

ELMZABI Amal	Professeur, à la faculté FSJES Mohammedia	Examineur
SIFI Fatima Zahra	Professeur, à la faculté FSJES Mohammedia	Examineur
ETTAIBI Charani	Professeur, à la faculté FSJES Mohammedia	Encadrant

**Année d'étude : 2021/2022**



# Dédicace

*Je dédie cet ouvrage*

*A mes parents qui m'ont soutenu et encouragé durant ces années d'études.*

*Qu'ils trouvent ici le témoignage de ma profonde reconnaissance.*

*A mes frères, mes grands-parents et Ceux qui ont partagé avec moi tous les*

*Moments d'émotion lors de la réalisation de ce travail. Ils m'ont*

*Chaleureusement supporté en encourageant tout au long de mon parcours.*

*A ma famille, mes proches et à ceux qui me donnent de l'amour et de la*

*Vivacité.*

*A tous mes amis qui m'ont toujours encouragé, et à qui je souhaite plus de*

*Succès.*

# Remerciement

Je tiens à remercier toute l'équipe pédagogique de la faculté de science juridique économique et sociale de Mohammedia et en particulier tout le corps professoral intervenant dans la filière « Méthodes informatiques appliquées à la gestion des entreprises » pour l'effort fourni pour réussir notre formation et mieux atteindre tous les objectifs attendus des différentes matières.

Mes remerciements les plus distingués sont à l'égard de mon encadrant Mr ETTAIBI Charani, qui en tant que professeur, a bien voulu accepter de suivre mon travail, me diriger afin de pouvoir mener ce stage à terme.

Je remercie vivement mon maître de stage, Mr Abdelali AHBIB, co-fondateur de l'entreprise Wings Technologies, pour son accueil, le temps passé ensemble et le partage de son expertise au quotidien. Grâce aussi à sa confiance j'ai pu m'accomplir totalement dans mes missions. Il fut d'une aide précieuse dans les moments les plus délicats.

En fin je remercier toutes les personnes, qui, de près ou de loin, se sont impliquées dans la réalisation de ce rapport, tant par le soutien opérationnel, que professionnel. J'adresse également mes sincères remerciements aux membres de ma famille qui n'ont jamais hésités de m'offrir le meilleur d'eux-mêmes, et surtout mes parents qui m'ont supporté au cours de mon parcours estudiantine.

# Résumé

La gestion de syndicat c'est un problème fréquent dans notre vie quotidienne l'objectif de projet de facilité ce travail en essayant de développer une application web qui permet gérer et organiser ce fonctionnement. Toute en essayant d'offrir la plupart des services qui peuvent y avoir besoin.

## **Abstract:**

The management of syndicate of a condominium it is a frequent problem in our daily life the project objective to facilitate this work by trying to develop a web application that allows manage and organize this operation. While trying to offer most of the services that may be needed there.

# Liste des figures

Figure 1 : Les package du système.....	14
Figure 2 : Diagramme de packages .....	15
Figure 3 : Diagramme de cas d'utilisation .....	16
Figure 4 : Diagramme de cas d'utilisation 2 .....	17
Figure 5 : Diagramme de classe .....	22
Figure 6 : : Modèle conceptuel de données, MCD .....	26
Figure 7 : Modèle logique de données, MLD.....	27
Figure 8 : Les composants de spring .....	35
Figure 9 : Une conversation entre le client, l'API et la base de données .....	38
Figure 10 : Architecture de l'application web.....	42
Figure 11 : La page d'accueil .....	42
Figure 12 : Page d'authentification.....	43
Figure 13 : Page d'inscription.....	44
Figure 14 : Page de tableau de bord .....	45
Figure 15 : L'interface de l'ajout .....	45
Figure 16 : Page de la modification .....	46

# Sommaire

<b>LISTE DES FIGURES.....</b>	<b>5</b>
<b>SOMMAIRE .....</b>	<b>6</b>
<b>INTRODUCTION GENERALE .....</b>	<b>6</b>
<b>CHAPITRE I : ETUDE, SPECIFICATION ET ANALYSE DE BESOINS.....</b>	<b>9</b>
1.2 INTRODUCTION .....	10
1.2 ETUDE DE L'EXISTANT.....	10
1.3 SPECIFICATION DES BESOINS .....	11
1.4 CONCLUSION .....	17
<b>CHAPITRE II : CONCEPTION ET MODELISATION .....</b>	<b>19</b>
2.1 INTRODUCTION.....	20
2.2 DIAGRAMME DE CLASSE .....	21
2.3 MODELISATION DES DONNEES .....	22
2.4 CONCLUSION .....	28
<b>CHAPITRE III : REALISATION .....</b>	<b>29</b>
3.1 INTRODUCTION.....	30
3.2 ENVIRONNEMENT LOGICIEL .....	30
3.3 ETAPE DE REALISATION .....	40
3.4 LES INTERFACES DE L'APPLICATION .....	42
3.5 CONCLUSION .....	47
<b>CONCLUSION GENERALE .....</b>	<b>48</b>

# Introduction générale

## Cadre générale du projet

Un syndicat est une association de personnes destinée à la défense de leurs intérêts communs. Le mot "syndicat", se retrouve, notamment dans le droit de la copropriété immobilière.

Chaque copropriété doit avoir un syndic qui assure la gestion administrative, comptable et financière de la copropriété, quand il s'agit de la gestion administrative, le syndic établi et tient à jour une liste de tous les copropriétaires, il doit convoquer un assemblé général au moins une fois par an, puis rétablir un compte rendu de la réunion pour envoyer le procès-verbal aux copropriétaires, de plus il doit conserver les archives de la copropriété. Généralement il s'agit d'une gestion complexe vu les contraintes multiples et les paramètres dont il faut tenir compte.

## Problématique

Souvent on trouve que le syndic a des problèmes dans la gestion administrative et financière d'une copropriété. Notamment quand le syndic décide de programmer un assemblé général il doit prévenir chaque membre (habitant dans la copropriété) de la date et éventuellement le lieu de la réunion, la même chose quand il va envoyer le compte rendu ou les décisions prises au cours de la réunion, et le même problème revient quand il s'agit par exemple d'un changement du programme, ce qui rend son travail plus frustrant et pas pratique.



Quand il s'agit d'un entretien de l'immeuble le syndic a pour mission de tenir au courant tous les copropriétaires de la réparation et les travaux nécessaire.

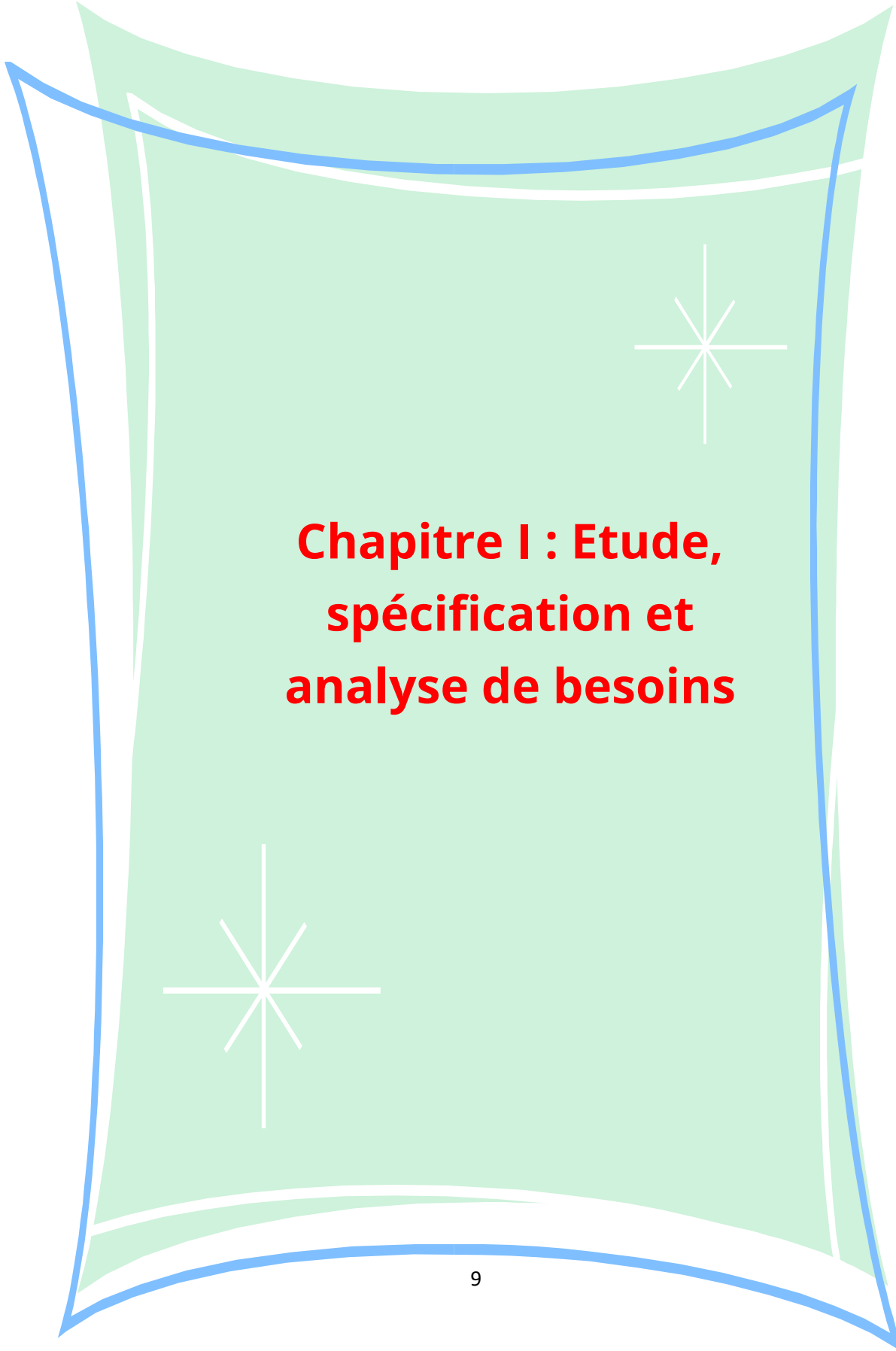
En outre les copropriétaires ont l'impression de ne pas connaître tous les habitants de l'immeuble, et quand un entre eux veut consulter le règlement interne, il doit le chercher chez le syndic. Donc le but est de développer une application web qui permet de gérer et bien organiser le fonctionnement du syndicat,

## **Plan du rapport**

La première partie de ce rapport présente le contexte de notre projet, le travail à réaliser et la solution que nous proposons pour réaliser ce travail.

La deuxième partie « Etude, Spécification et analyse des besoins » se focalise sur la présentation du projet à réaliser, ensuite, l'identification et l'analyse des fonctionnels et non fonctionnels.

La troisième partie expose tous les outils utilisés pour développer cette application et les différents pages développer ainsi que les détails de l'implémentation.



# **Chapitre I : Etude, spécification et analyse de besoins**

## 1.2 Introduction

Jusqu'aujourd'hui les logiciels qui ont été conçus pour gérer les services du syndicat d'une copropriété sont rare, même si comme ces logiciels se base sur un principe simple, que ce soit au niveau de base de données ou au niveau des traitements.

La gestion du syndicat d'une copropriété est l'un des exemples les plus fréquents des problèmes dans la vie des gens.

La gestion de services de syndicat d'une copropriété pose un problème qu'on doit résoudre.

D'une manière générale, pour résoudre un problème donné, on commence par la recherche de la classe à laquelle appartient ce problème, les problèmes relatifs aux gestions de syndicat d'une copropriété augmentent en fonction de contraintes, ce qui rend difficile leur résolution. De tels problèmes peuvent être résolus par des approches classique ou moderne.

Dans ce chapitre, nous découvrirons les problèmes de la gestion des services de syndicat d'une copropriété.

## 1.2 Etude de l'existant

La tâche de la gestion administrative est confiée au syndic, il réalise tout le travail manuellement et se charge de toute la gestion qui concerne une copropriété.

Grace à son expérience il arrivent à tout faire et à satisfaire la plupart des vœux des copropriétaires mais, d'une part cela lui nécessite un effort considérable :

La gestion de fonctionnement du syndicat est un plan représentatif qui nécessite :

- Un syndic qui va assurer la gestion administrative et financière d'une copropriété.
- Une copropriété va être gérée par un syndic.
- Un copropriétaire consulte leur cotisation et le règlement interne.

En pratique toutes ces fonctionnalités vont être disponibles après l'authentification de l'utilisateur, pour des raisons de la sécurité.

## **1.3 Spécification des besoins**

### **1.3.1 Introduction**

Une méthode de conception est une démarche générale reflétant une philosophie de présentation et de suivi du système. Elle propose des outils spécifiques permettant un suivi efficace de l'information relative au système. Et notre choix se porte sur le langage UML (Unified Modeling Language) qui facilite l'interactivité avec la base de données à l'aide des diagrammes de cas d'utilisation et des diagrammes de classes

L'UML (Unified Modeling Language) est un langage de modélisation orientée objet, elle est développée dans le but de définir la notation standard pour la modélisation des applications construites à l'aide des objets. Elle est utilisée pour spécifier un logiciel ou

pour le concevoir, le modèle décrit les classes et les cas d'utilisation vus de l'utilisateur final du logiciel.

Le modèle produit par une conception orientée objet est en général une extension du modèle issu de la spécification, il l'enrichit de classe dites techniques qui n'intéressent pas l'utilisateur final du logiciel mais seulement ses concepteurs.

## 1.3.2 Spécification de besoins

### 1.3.2.1 Les acteurs

Un acteur est une entité externe qui interagit avec le système (opérateur, centre distant, autre système...). En réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin. Les acteurs peuvent être classés (hiérarchie).

**Le syndic :** c'est la personne primordiale dans le système de gestion de syndicat d'une copropriété. Il permet de manipuler toutes les tâches proposées et possibles ; tels que l'ajout, La modification, la suppression et consultation.

- ✓ Ajouter un nouvel habitant
- ✓ Ajouter et mettre à jour les cotisations
- ✓ Réaliser et mettre à jour la liste de tous les copropriétaires
- ✓ Réaliser et mettre à jour le règlement interne
- ✓ Ajouter un copropriétaire en tant que admin

**Le copropriétaire :** toutes personnes qui habitent dans la copropriété concerner.

- ✓ Consulter leur cotisation
- ✓ Consulter le règlement interne

### 1.3.3 Décomposer le système en partie

Les besoins très différents des acteurs et le nombre de fonctionnalités dont le futur logiciel devra disposer nous semble assez souvent compliqué. Pour y voir clair et pour nous faciliter la tâche, on peut découper le logiciel en parties distinctes, en fonction des « familles » de fonctionnalités et de façon à pouvoir les analyser séparément. Chacune de ces parties correspond à un domaine fonctionnel ou **package**.

On décompose le logiciel à développer en plusieurs packages, dès que possible. En essayant d'identifier les parties bien distinctes parmi les fonctionnalités de l'application pour cela, on pose cette question : « Est-ce que l'application comporte des parties différentes qui pourraient être analysées séparément ? »

Pour notre application, il y a deux parties distinctes :

- La partie « Gestion administrative » qui contiendrait la gestion de toutes les fonctionnalités du site à proprement parler, la gestion des cotisations, financière et la mise à jour des données.
- La partie « Gestion copropriétaire », incluant les tâches liées à la consultation des cotisations et le règlement ainsi et les nouvelles qui sont partagées dans le site.

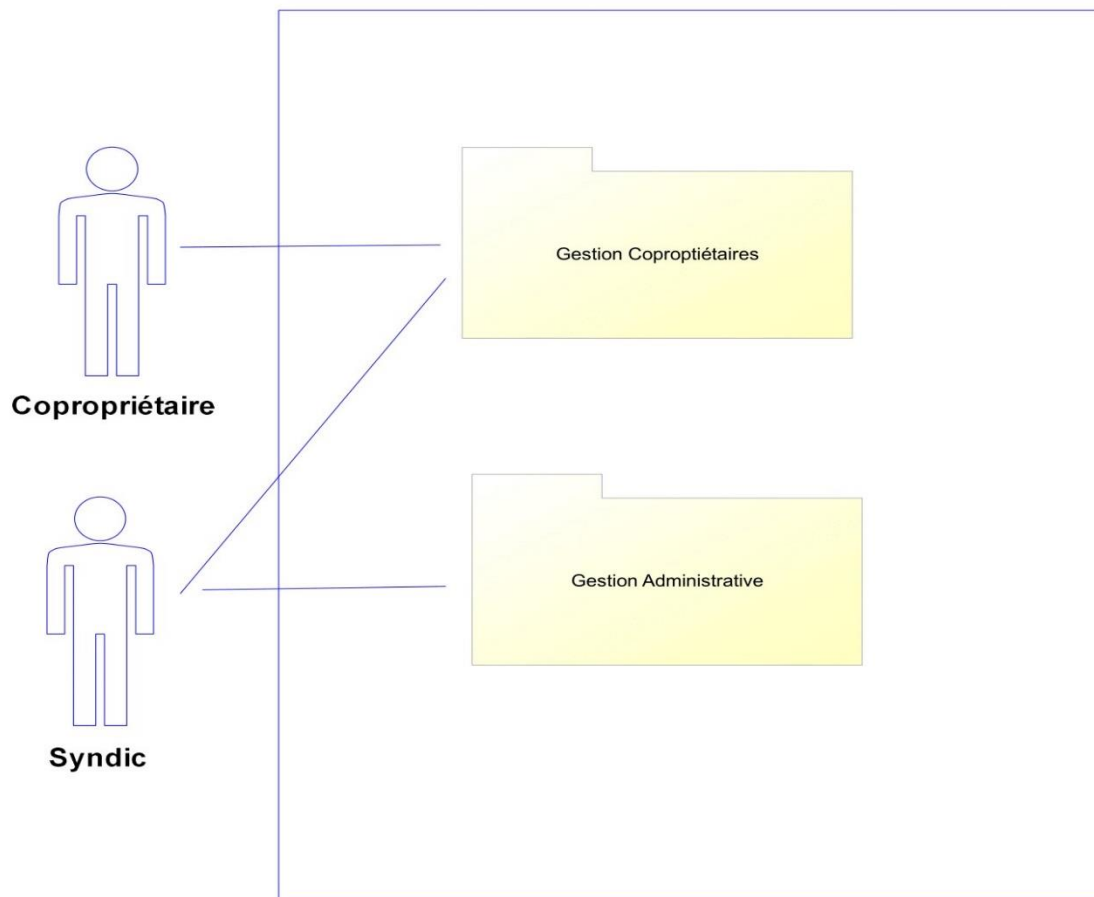
Nous aurons donc deux packages. Un package est représenté sous forme de dossier :



**Figure 1 : Les package du système**

Il ne nous reste plus qu'à réaliser le **diagramme de packages**, en mettant en évidence les acteurs qui interviennent dans chacun de ces packages. Pour cela, on représente au centre le Système, qui comprend les deux packages que nous avons trouvés : la gestion des copropriétaires et la Gestion administrative. Autour de cela, nous devons donc relier les acteurs principaux au package correspondant !

Ici, le Syndic et le copropriétaire, sont les acteurs principaux,



**Figure 2 : Diagramme de packages**

### **1.3.4 Diagramme des cas d'utilisation**

Le diagramme de cas d'utilisation décrit la succession des opérations réalisées par un acteur. C'est le diagramme principal du modèle UML, celui qui assure la relation entre l'utilisateur et les objets que le système met en œuvre.

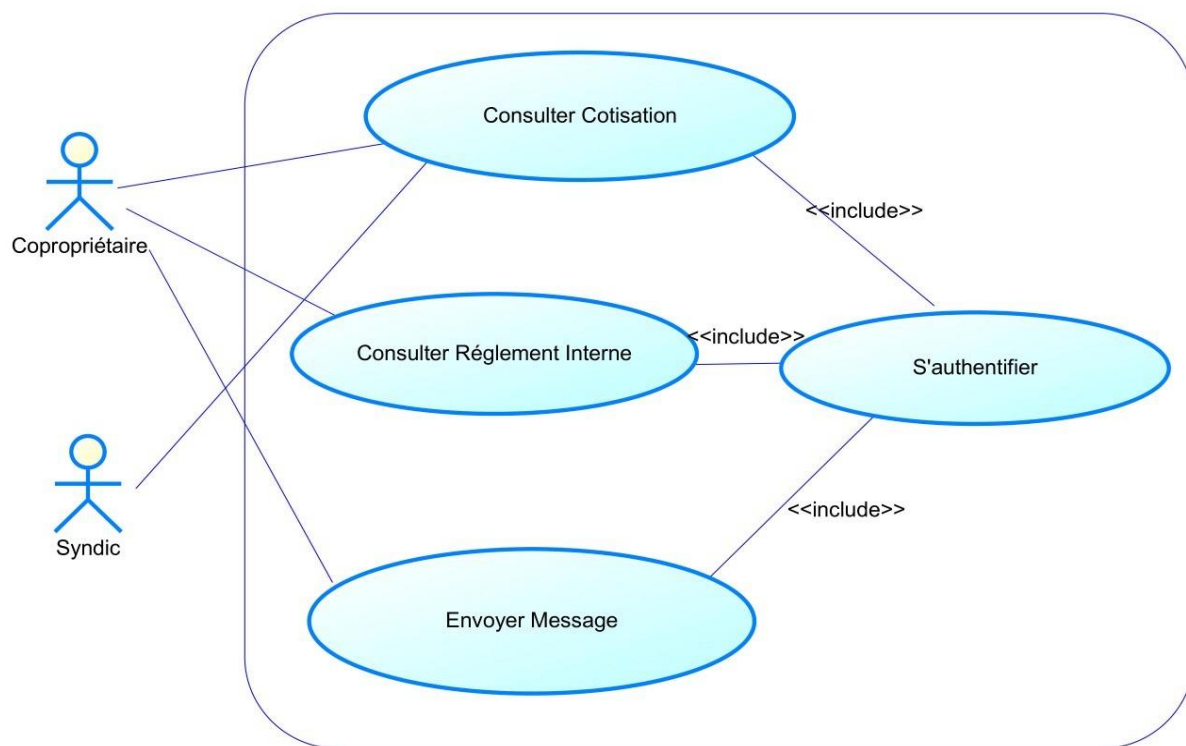
La description de l'interaction est réalisée suivant le point de vue de l'utilisateur, et les cas d'utilisation permettent de recueillir et de décrire les besoins des acteurs aux



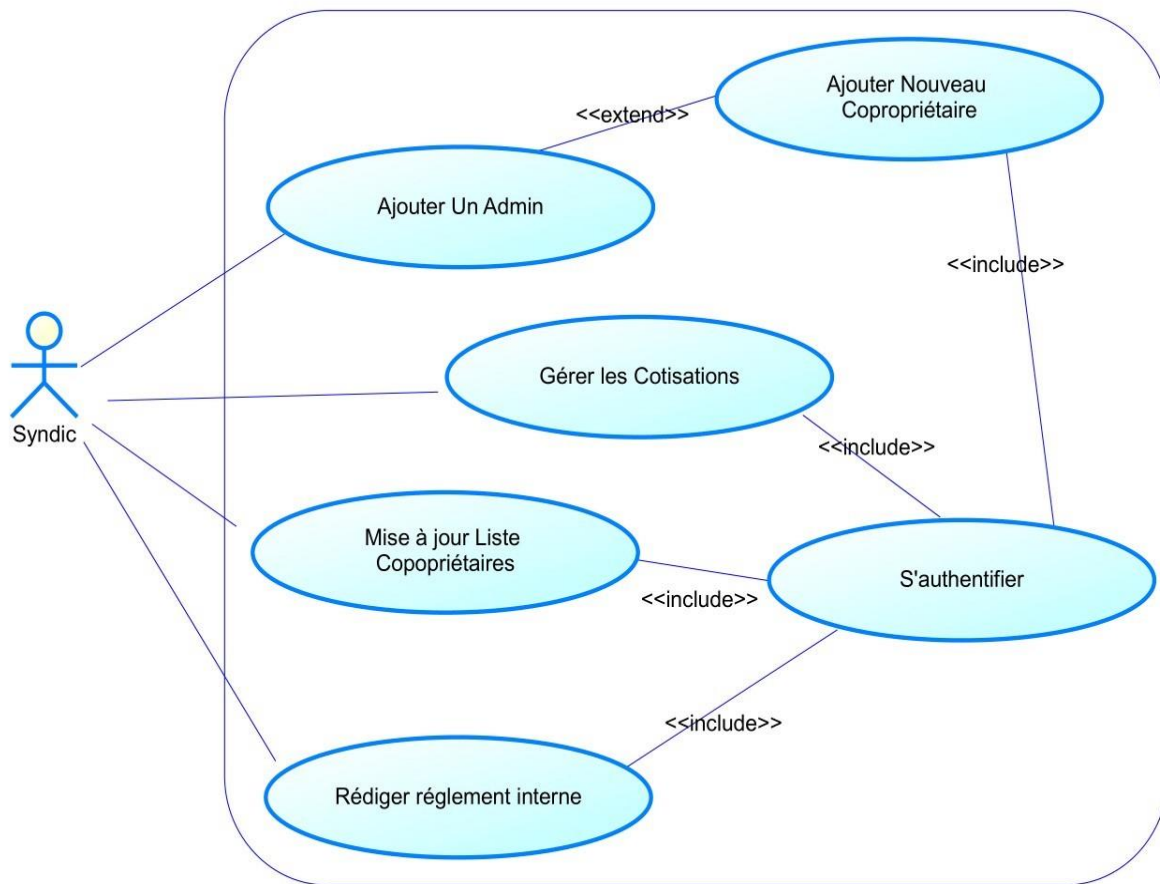
systèmes, il permet aussi de faciliter la structuration des besoins des utilisateurs et d'exprimer les limites et les objectifs du système.

Les cas d'utilisation principaux sont liés aux acteurs qui en ont besoin. Ils détaillent les diagrammes de packages. On se concentre sur le package « Gestion des copropriétaires » pour en réaliser **le diagramme des cas d'utilisation**.

Il faut donc identifier toutes les fonctionnalités dont les différents acteurs concernés par le package auront besoin. Dans notre projet, nous indiquons qu'un copropriétaire devra pouvoir utiliser le site pour consulter leur cotisation et le règlement interne. Nous **pouvons en déduire que « Consulter les cotisations » est un cas d'utilisation**.



**Figure 3 : Diagramme de cas d'utilisation**



**Figure 4 : Diagramme de cas d'utilisation 2**

## 1.4 Conclusion

La phase d'analyse a duré presque deux semaines, au cours de cette période, nous avons essayé de structurer et définir les besoins attendus du futur système. Il s'agissait de formuler, d'affiner et d'analyser la plupart des cas d'utilisation via les diagrammes d'UML.

Il faut noter que le dégagement des grandes fonctionnalités du système n'a pas suffi pour aborder la phase de conception, il fallait dégager plus de besoins. Il nous a fallu interroger les différents acteurs du système d'information pour enrichir notre diagramme de cas d'utilisation. Et là nous étions confrontés à un problème délicat : la dissimulation de l'information. La solution réside dans le Processus Unifié. Les éléments à livrer au terme de la phase d'analyse (acteurs, besoins fonctionnels, besoins non fonctionnels).



## **Chapitre II : Conception Et Modélisation**

## 2.1 Introduction

Il est aujourd'hui admis que le processus de développement d'un système d'information comporte deux phases essentielles : la phase de conception et la phase de réalisation. La phase de conception est centrée sur l'expression de besoins et la recherche de solution pour satisfaire à ces besoins, elle vise à produire différentes spécifications du système d'information, quant à la phase de conception elle se consiste à implémenter la solution pour aboutir à un système logiciel. La phase de conception est devenue essentielle dans le processus de développement des S.I. Des études bien qu'empiriques montrent que 82% des erreurs qui surviennent dans le développement des SI proviennent de la conception. Ces erreurs sont les plus difficiles et les plus coûteuses à résoudre.

La conception est la tâche la plus créative et la plus difficile car étant un processus de décisions complexes concernant généralement :

- Les fonctions que le S.I doit assurer et les informations que le S.I doit gérer
- Les techniques de traitement, de communication et de diffusion de l'information
- Les règles de mémorisation, de traitement, de communication et de diffusion de l'information.
- Les structures de travail et les comportements attendus.

## 2.2 Diagramme de classe

Le diagramme de classe représente la description statique du système à développer en intégrant dans chaque classe la partie dédiée aux données et celle consacrée au traitement.

C'est un diagramme pivot de l'ensemble de modélisation d'un système, cette représentation est concentrée sur le concept de classe et d'associations, les traitements sont matérialisés par des opérations.

Une classe est une description abstraite d'un ensemble d'objet ayant des propriétés similaires, un comportement commun et des relations communes avec d'autres objets.

### **Règle de gestion :**

- ✓ Un utilisateur (Copropriétaire et syndic) peut envoyer un ou plusieurs messages.
- ✓ Un utilisateur peut payer une ou plusieurs cotisations
- ✓ Un syndic (admin) peut gérer un ou plusieurs copropriétés
- ✓ Un utilisateur peut résider dans un seul appartement
- ✓ Une caisse concerne un utilisateur et un immeuble

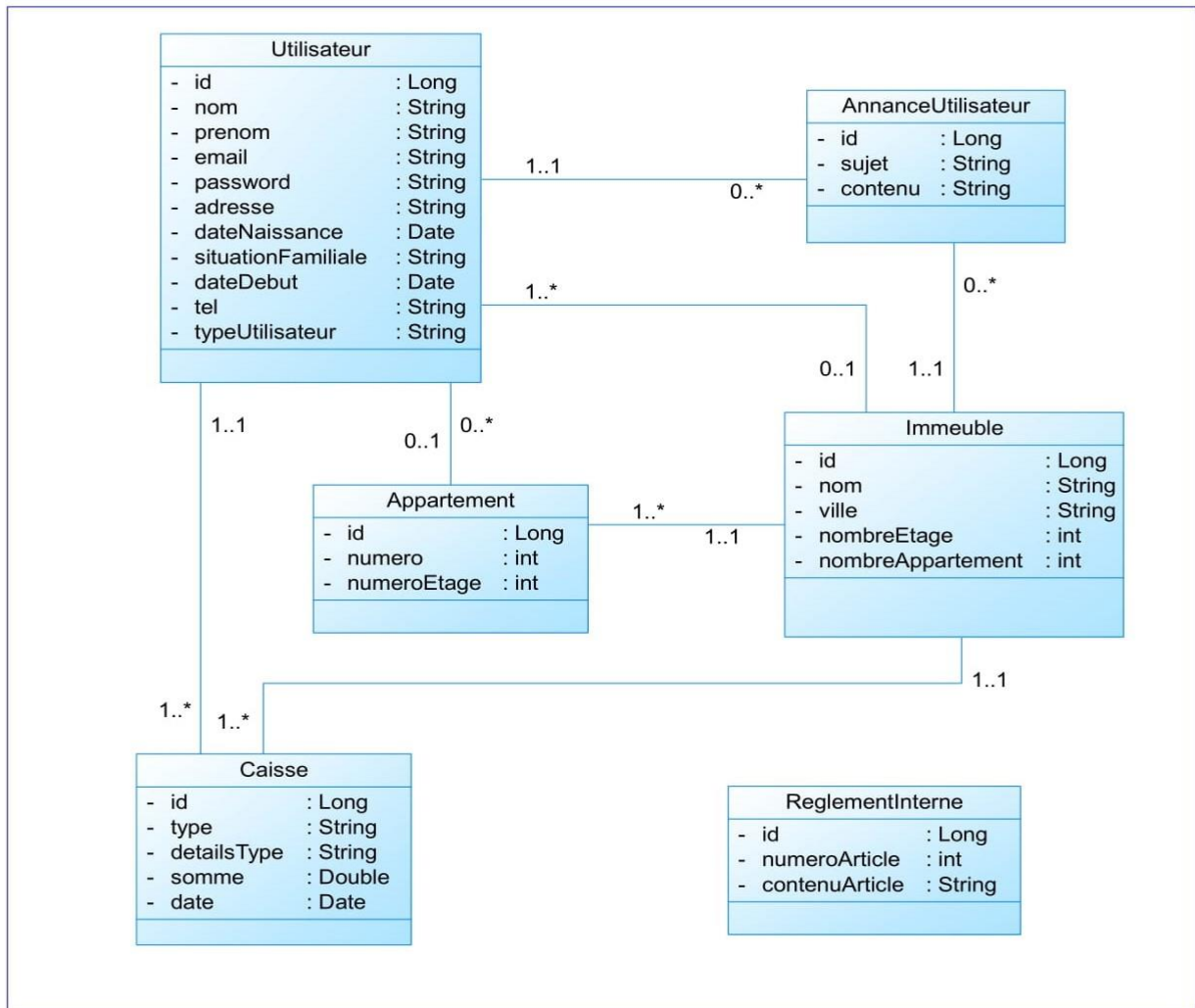


Figure 5 : Diagramme de classe

## 2.3 Modélisation des données

### 2.3.1 Description des données

#### a) Table Utilisateur

Utilisateur		Type
Attributs	Désignation	
id	Identité de l'utilisateur	Integer
nom	Nom de l'utilisateur	String
prénom	Prénom de l'utilisateur	String
email	e-mail de l'utilisateur	String
password	Mot de passe de l'utilisateur	String
adresse	Adresse de l'utilisateur	String
dateNaissance	Date de naissance de l'utilisateur	Date
tel	Numéro de téléphone de l'utilisateur	String
dateDebut	La date d'entrée	Date
situationFamiliale	La situation familiale de l'utilisateur	String
typeUtilisateur	Le type de l'utilisateur (Résident, Syndic, Syndic Externe)	String

## b) Table Immeuble



Propriétés		Type
Attributs	Désignation	
id	L'identifiant de l'immeuble	Integer
nom	Le nom de l'immeuble	String
Ville	La ville de l'immeuble	String
nombreEtage	Le nombre d'étages dans l'immeuble	Integer

### c) Table AnnonceUtilisateur

Propriétés		Type
Attributs	Désignation	
id	L'identité de message	Integer
sujet	Le sujet de l'annonce	String
Contenu	Le contenu d'annonce	String

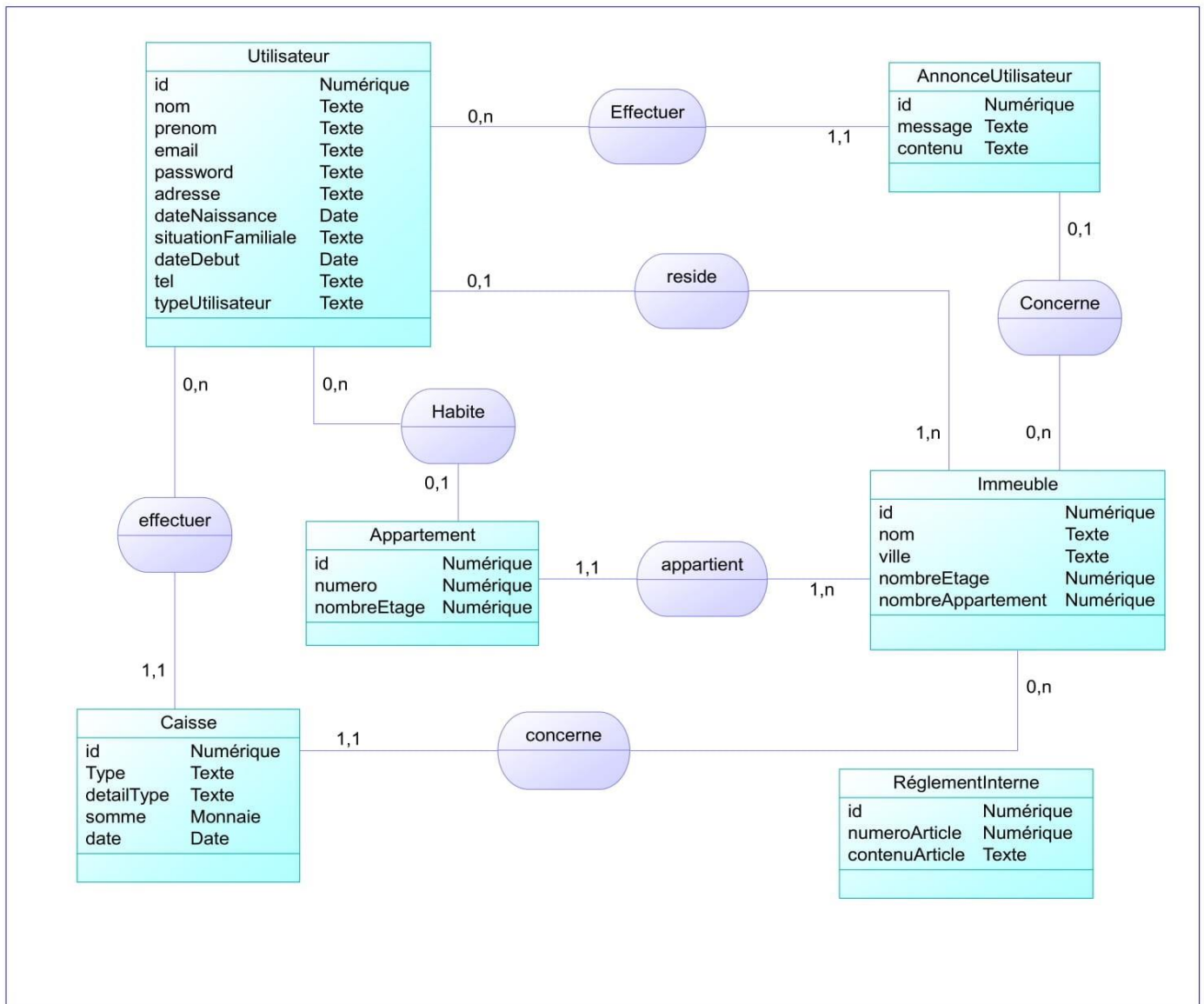
### d) Table Caisse

Propriétés		Type
Attributs	Désignation	
id	L'identité de caisse	Integer
TypeCaisse	Le type de caisse (entrée ou charge)	String
detailType	Le détail de type de caisse (travaux, cotisation, ménage, don ...)	String
somme	Le somme à payer	Réel
date	La date de paiement	Date

### e) Table règlement

Propriétés		Type
Attributs	Désignation	
id	L'identité du règlement	Integer
numeroArticle	Numéro de l'article	Integer
contenuArticle	Le contenu de l'article	String

## 2.3.2 Modèle conceptuel de données MCD



**Figure 6 : Modèle conceptuel de données, MCD**

## 2.3.3 Modèle Logique de données, MLD

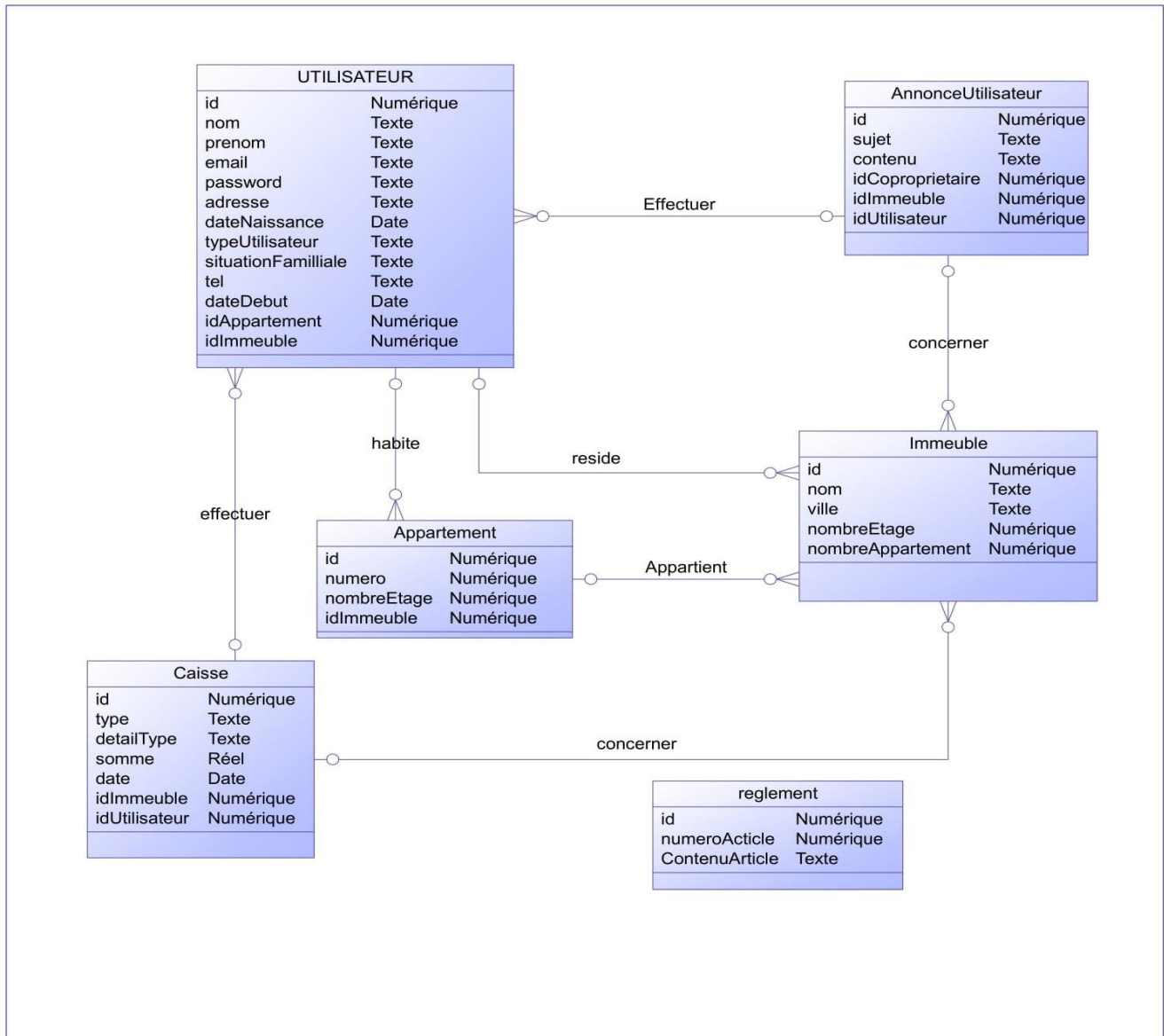


Figure 7 : Modèle logique de données, MLD

## 2.4 Conclusion

Dans ce chapitre on a entamé la phase de conception. Dans cette phase, nous avons déjà un modèle final des cas d'utilisation. Il s'agissait alors d'étendre la représentation effectuée au niveau de l'analyse en y intégrant les aspects techniques les plus proches des préoccupations physiques. L'élément principal à livrer au terme de cette phase est le diagramme de classe ainsi que le schéma relationnel.



## **Chapitre III : Réalisation**

## 3.1 Introduction

Ce dernier chapitre présente la partie de la réalisation et la mise en œuvre des différents composants décrits au niveau du chapitre précédent. Dans un premier temps, on présente l'environnement logiciel. Ensuite, on décrit le travail réalisé en détaillant quelques captures d'écrans des fonctionnalités réalisées.

## 3.2 Environnement logiciel

### 3.2.1 Choix de technologies de développement coté frontend

#### 3.2.1.1 Framework Angular



Angular est un Framework de développement des applications web, utilisé par des milliers d'entreprises, qui permet de créer des applications dynamiques complètes.

Angular permet de créer la partie front end des applications web de type SPA (Single Page Application réactive)

Une SPA est une application qui contient une seule page HTML (index.html) récupérée du serveur. C'est dans cette page qu'on fait toute la logique de présentation de notre application web.

Pour naviguer entre les différentes parties de cette application, JavaScript est utilisé pour envoyer des requêtes http vers le serveur pour récupérer du contenu dynamique généralement au format JSON. Ce contenu JSON est ensuite affiché côté client au format HTML dans la même page.

Généralement les applications Angular sont écrites en Type Script qui est compilé et traduit en java script avant d'être exécuté par les browsers web. Angular est basé sur une programmation appuyée sur les Composants web (Web Component).

Le TypeScript est un langage de programmation open-source développé par Microsoft. Le langage se présente comme un sur-ensemble du JavaScript notamment avec l'apport d'un typage statique optionnel des variables et des fonctions, la création de classes et d'interfaces, la création de namespace et de modules. Le type Script est un langage de script structuré et orienté objet qui permet de simplifier le développement d'application Java Script.

Le développement Angular passe par trois langages principaux :

- Le **HTML** pour structurer nos pages dans l'application ;
- Le **SCSS** pour les styles – le SCSS est une surcouche du CSS qui y apporte des fonctionnalités supplémentaires, mais qui permet également d'écrire du CSS pur si on le souhaite ;
- Le **TypeScript** pour tout ce qui est dynamique, comportement et données – un peu comme le JavaScript sur un site sans Framework.



### 3.2.1.2 Bibliothèque Bootstrap



Bootstrap est une bibliothèque de *composants* frontend qui permet de *prototyper* nos idées et de créer un site web entier à l'aide de HTML, CSS et JavaScript. Bootstrap est utile pour le développement *rapide* de prototypes de nos sites web et applications web, afin que nous puissions nous concentrer sur la fonctionnalité principale et la structure du site sans préoccuper de la *compatibilité* entre navigateurs.

### 3.2.1.3 Outils de développement

#### ❖ NodeJS et npm



NodeJS est une plateforme construite sur le moteur JavaScript V8 de Chrome qui permet de développer des applications en utilisant du JavaScript. Il se distingue des autres plateformes grâce à une approche non bloquante permettant d'effectuer des entrées/sorties (I/O) de manière asynchrone.

npm (Node Package Manager) est le gestionnaire de paquets par défaut pour l'environnement d'exécution JavaScript Node.js de Node.js.

npm se compose d'un client en ligne de commande, également appelé npm, et d'une base de données en ligne de paquets publics et privés payants, appelée le registre npm. Le registre est accessible via le client, et les paquets disponibles peuvent être parcourus et recherchés via le site Web de npm. Le gestionnaire de paquets et le registre sont gérés par npm.

### ❖ CLI d'Angular

Le CLI (Command Line Interface, ou interface en ligne de commande) d'Angular est un outil indispensable pour créer, gérer et déployer les applications Angular. On l'installe globalement sur notre ordinateur depuis une ligne de commande avec :

```
npm i -g @angular/cli
```

### ❖ Visual studio code IDE

Visual Studio Code est un éditeur de code open-source développé par Microsoft supportant un très grand nombre de langages grâce à des extensions. Il supporte l'auto-complétions, la coloration syntaxique, le débogage, et les commandes git.

## 3.2.2 Choix de technologies de développement coté backend

### 3.2.2.1 JAVA EE



JAVA EE est une plat-forme pour développer des site web solides, robustes et bien structurés, elle est notamment très utilisée dans la finance par exemple des banques ou les états. Car elle a acquis une très grande maturité professionnelle. JAVA EE est basé sur le langage de développement JAVA

### ***3.2.2.2 Framework Spring et Spring Boot***



Spring est un Framework libre très riche, parmi les plus réputés au monde. Il permet de construire l'infrastructure d'une application Java et d'en faciliter le développement.

Spring est un Framework, c'est-à-dire un cadre de travail existant que les développeurs peuvent utiliser. C'est comme un grand élément spécialisé dont on peut trouver d'autre composant.

## Spring Security



## Spring Data



## Spring Core



## Spring Cloud



## Spring Boot



Figure 8 : Les composants de spring

### ❖ Le Design-pattern IOC Container

C'est un patron de conception qui permet d'évoluer notre code. Avec IOC appelé aussi *contexte spring*, on pourra créer des objets dynamiquement, et de les injecter

dans d'autres objets. De plus on pourra facilement modifier l'implémentation d'un objet, avec quasiment zéro impact sur les objets qui utilisent ce dernier.

IoC est le sigle "de Inversion of Control". Cette expression indique un principe de programmation qui correspond au fait de déléguer à un Framework le flux de construction et d'appels des objets.

### ❖ **Spring Boot**

C'est un composant très particulier de Spring Framework, dans la mesure où il nous permet de mettre en œuvre tous les autres. Les avantages de Spring Boot sont :

- ✓ L'autoconfiguration automatique de Spring ;
- ✓ Des starters de dépendances ;
- ✓ Des Endpoint Actuator pour fournir des données sur l'application.

Spring Boot œuvre pour la simplification du développement de nos projets avec Spring Framework.

La gestion des dépendances est simplifiée grâce aux starters qui regroupent plusieurs dépendances et homogénéisent les versions.

L'autoconfiguration permet de se concentrer sur le code métier, et simplifie énormément la mise en œuvre des composants Spring qui sont utilisés. Spring Boot Actuator permet de monitorer et gérer une application pendant son exécution. Et le déploiement de l'application est facilité par la génération d'un JAR, et pour les projets web, un tomcat est embarqué.

### ❖ **JPA : Java Persistence API**

Java Persistence API (abrégée en JPA), est une interface de programmation Java permettant aux développeurs d'organiser des données relationnelles dans des applications utilisant la plateforme Java.

Une entité JPA est, par définition, une classe Java qui doit avoir les propriétés suivantes :

- ✓ Elle doit posséder un constructeur vide, public ou protected. Rappelons que ce constructeur vide existe par défaut si aucun constructeur n'existe dans la classe. Dans le cas contraire, il doit être ajouté explicitement.
- ✓ Elle ne doit pas être finale, et aucune de ces méthodes ne peut être finale.
- ✓ Une entité JPA ne peut pas être une interface ou une énumération.
- ✓ Une entité JPA peut être une classe concrète ou abstraite.
- ✓ Une classe possède des champs. Ces champs sont en général associés à des valeurs en base, rangées dans les colonnes d'une table. L'objet EntityManager est capable de positionner lui-même les valeurs de ces champs. Il le fait par injection, soit directement sur le champ, soit en passant par les getters / setters.

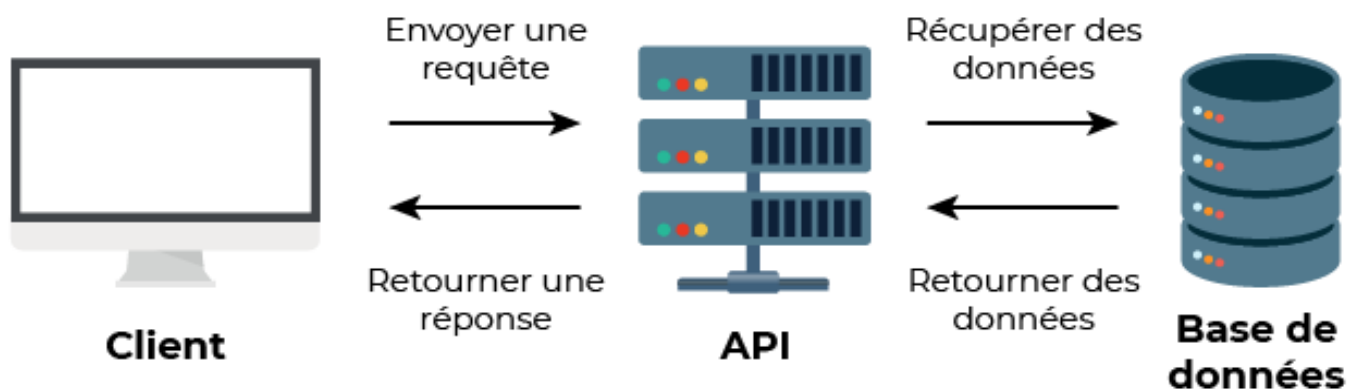
Il est possible de définir des entités JPA sur des classes qui héritent les unes des autres. Sur toute une hiérarchie de classes, on doit avoir une unique définition de clé primaire.

### 3.2.2.3 Les API REST



API est une abréviation et signifie Application Programming Interface (ou interface de programmation d'application, en français). C'est un moyen de communication entre deux logiciels, que ce soit entre différents composants d'une application ou entre deux applications différentes.

En web, un service web et une API sont tous les deux des moyens de communication. Un service web standard facilite seulement la communication entre deux machines via un réseau. Une API facilite l'interaction entre deux applications différentes afin qu'elles puissent communiquer entre elles : elle sert d'intermédiaire. Le client va demander à l'API une information, celle-ci va aller chercher cette information dans la base de données puis la renvoyer au client dans un second temps.



**Figure 9 : Une conversation entre le client, l'API et la base de données**

REST est un type d'architecture d'API qui signifie *REpresentational State Transfer*. Il a été inventé par l'Américain Roy Fielding dans les années 2000, période charnière dans la reconnaissance du potentiel des API web, afin de mettre en place des méthodes simples pour accéder à des services web.

Ce type d'API permet à des logiciels qui sont incompatibles, qui ne parlent pas le même langage, de communiquer facilement. REST peut être considéré comme un langage commun à ces différents logiciels. Par exemple, une API REST peut être réalisée dans le langage Java ou .NET.

L'API REST est une architecture qui repose sur le protocole HTTP, c'est-à-dire que l'API REST permet d'accéder à un service web via les 4 opérations de HTTP, ou *verbes* (POST, GET, PUT, DELETE). Ces opérations ont donc 4 fonctions résumées par l'acronyme **CRUD** (**C**reate, **R**ead, **U**ppdate, **D**eleter) :

- Opération HTTP "POST".  
Fonction : créer (Create) : permet de créer une ressource ;
- Opération HTTP "GET".  
Fonction : afficher (Read) : permet de lire une ressource ;
- Opération HTTP "PUT".  
Fonction : mettre à jour (Update) : permet de modifier la valeur d'une ressource ;
- Opération HTTP "DELETE".  
Fonction : supprimer (Delete) : permet de supprimer une ressource sur le serveur.
- **MySQL**

MySQL est une base de données relationnelle libre qui est très employée sur le Web, souvent en association avec PHP (langage) et Apache (serveur web). MySQL fonctionne



indifféremment sur tous les systèmes d'exploitation. Il a le principe d'une base de données relationnelle et d'enregistrer les informations dans des tables, qui représentent des regroupements de données par sujets. Les tables sont reliées entre elles par des relations.

Le langage SQL (Structured Query Language) est un langage reconnu par MySql et les autres bases de données et permettant de modifier le contenu d'une base de données.

### 3.2.2.4 Outils de development

#### ❖ IntelliJ IDEA



IntelliJ IDEA est un environnement de développement intégré (en anglais Integrated Development Environment - IDE) destiné au développement de logiciels informatiques reposant sur la technologie Java. Il est développé par JetBrains (anciennement « IntelliJ ») et disponible en deux versions, l'une communautaire, open source, sous licence Apache 2 et l'autre propriétaire, protégée par une licence commerciale. Tous deux supportent les langages de programmation Java, Kotlin, Groovy et Scala.

## 3.3 Etape de réalisation

Les caractéristiques de Spring Boot peuvent être résumées comme suit :

l'intégration directe d'applications de serveur Web/de conteneur comme Apache Tomcat ou Jetty sans utiliser de fichiers WAR (Web Application Archive) la configuration simplifiée de Maven grâce à des POM (Project Object Models) « Starter »

Au cœur de Spring Boot se trouvent les annotations qui servent à simplifier la tâche au développeur. Pour créer un contrôleur, il suffit de créer une classe et de l'annoter `@RestController` et de lui affecter un point d'accès. Chacune des méthodes aura l'annotation `@RequestMapping` qui indique quel chemin de l'API la méthode couvre et quelle méthode HTTP lui correspond.

Ces annotations permettent à simplifier le code et à le rendre plus lisible. Le framework s'occupe de démarrer le serveur web et de rediriger les requêtes aux méthodes concernées.

Angular s'occupe de la couche présentation (vue) de notre application. Le bloc de base d'une application Angular est le module (NgModules) qui sert de contexte de compilation et d'exécution à un autre élément nommé Composant. Un composant peut être vu comme la combinaison :

- D'une **Vue** : du contenu HTML
- D'un **Modèle** de données : les informations qui vont être affichées dans le contenu HTML
- D'un **Contrôleur**, qui va se charger de la logique derrière l'affichage des données dans la vue.

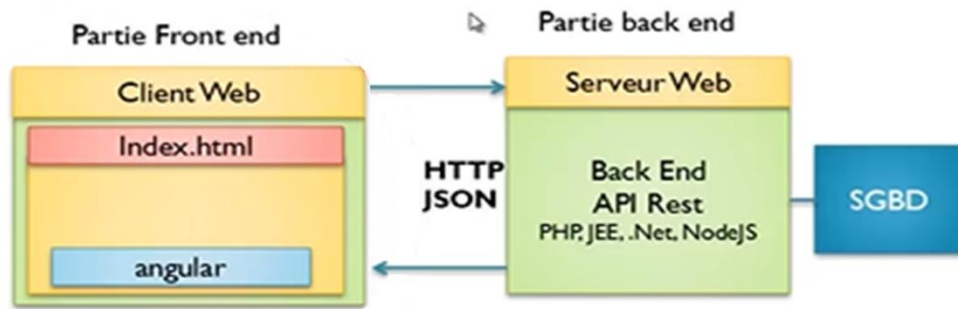


Figure 10 : Architecture de l'application web

## 3.4 Les Interfaces de l'application

- ❖ Page d'accueil de l'application c'est le point d'entrée de l'application web.

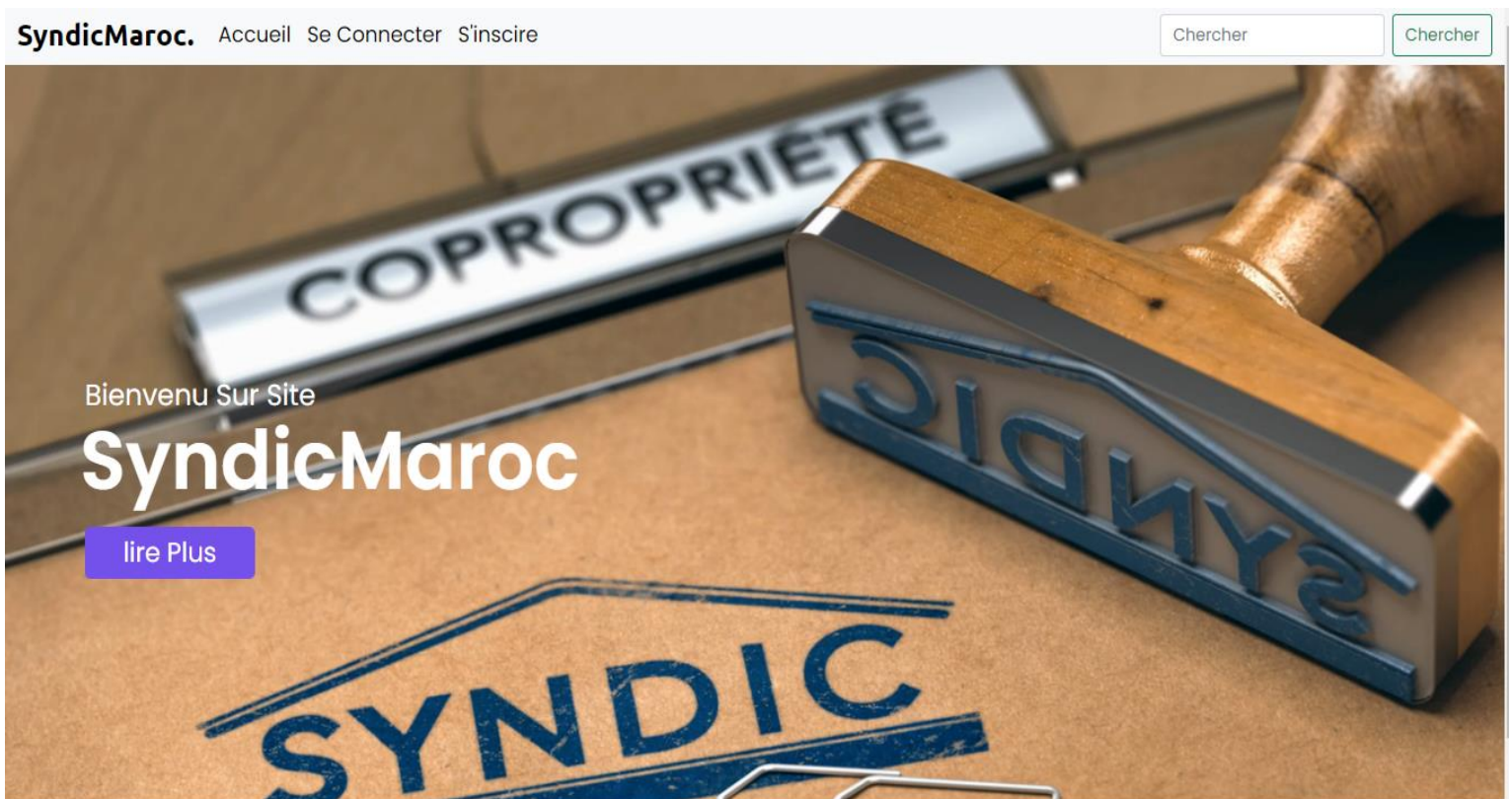



Figure 11 : La page d'accueil

- ❖ L'interface Authentification de l'application. Cette interface est décomposée deux champs pour la saisie du l'email et du mot de passe pour pouvoir accéder à l'application.



Se Connecter

E-mail\*  
exmple@gmail.com

Mot De Passe\*  
.....

Connexion

Créer un compte

**Figure 12 : Page d'authentification**

- ❖ La page de l'inscription de l'application. Cette page consiste à créer un nouveau compte pour pouvoir se connecter à l'application.



### Créer un compte

Nom *	😊	Prénom *	😊
Date de Naissance *	jj/mm/aaaa	E-mail *	✉
Mot de passe *	🔑	Confirmer Votre mot de passe *	🔑
Téléphone *	📞	Adresse *	📍
Situation Familiale : <input type="radio"/> Marié(e) <input type="radio"/> Divorvé(e) <input type="radio"/> Célibataire			
Date de D'entrée *	jj/mm/aaaa	Type Utilisateur	▼

**S'inscrire**

**Figure 13 : Page d'inscription**

- ❖ L'interface de tableau de bord pour la gestion administrative c'est la partie qui se charge de la mise à jour des données dans l'application web.

## Tableau de bord

- 🏠 Accueil
- 🏠 Immeubles
- 🏠 Appartements
- 👤 Utilisateurs
- 💰 La Caisse
- 📢 Les Annonces
- ⚙️ Régléments

[Accueil](#) [Déconnecter](#)

Chercher

## Gesion Adminstrative

[Ajouter un Utilisateur](#)  Chercher

### Liste des Utilisateurs

ID	Nom	Prenom	Email	Téléphone	immeuble	N° Appartement	Editer	Supprimer
1	Elyaaqobi	Ahmed	Elyaaqobi2022@gmil.com	0620202020	Résidance Amane	16		
2	Yassir	Ayoub	Yassir2022@gmil.com	0630303030	Résidance Farid Lmotawakil	18		
3	benhamou	brahim	benhamou2022@gmil.com	0640404040	Résidance Sarra	15		
4	hakimi	adil	hakimi2022@gmil.com	0630303030	Résidance Amane	20		
5	Said	Ahmed	said@gmail.com	0644434878	Résidance Gala	17		

0 1

Figure 14 : Page de tableau de bord

- ❖ L'interface l'ajout qui permet de faire l'ajout d'un élément dans la base de données.

### Ajouter Une Annonce

Sélectionner Un Immeuble

Sélectionner Un Utilisateur

Sujet \*

Contenu \*

Ajouter

Figure 15 : L'interface de l'ajout

- ❖ L'interface de la modification qui permet de faire la modification d'un élément dans la base de données.



The screenshot shows a web form titled "Modifier l'Appartement" in a large, bold, black font. Below the title is a horizontal blue line. The form contains three input fields: a text box for "Numero \*" with the value "15", a text box for "Numero Etage \*" with the value "15", and a dropdown menu for "Résidence Amane" with a downward arrow. Below these fields is a large blue button with the text "Modifier" in white.

## Modifier l'Appartement

Numero \*  
15

Numero Etage \*  
15

Résidence Amane ▼

Modifier

**Figure 16 : Page de la modification**

## 3.5 Conclusion

Nous voici au terme de notre travail qui a porté sur la mise en place d'une application web de la gestion de syndicat.

Ce projet nous a permis d'avoir une approche complète du développement d'une application et une bonne initiation au cycle complet du développement de l'application, de la conception à la validation en passant par les différentes étapes incrémentales de codage et de tests et nous a appris aussi à concevoir une base de données complète.

Ce travail avait comme objectif principal de concevoir une application web pour permettre au gérer les services du syndicat.

Une bonne stratégie pour partager ces données serait de centraliser la base de données sur une machine (serveur distant), mais accessible par tous.



# Conclusion générale

Au terme de ce projet de fin d'études, je tiens à souligner que sa réalisation était d'un très grand bénéfice pour moi car c'était une bonne occasion pour consolider mes connaissances théoriques dans le domaine de conception et la réalisation des applications informatiques. Il est évidemment que ce projet n'est pas une œuvre parfaite mais j'ai tenu à ce qu'il soit à la hauteur de mes espérances professionnelles ainsi aux responsables de l'entreprise wings technologies en espérant qu'ils trouvent dans mon travail une bonne solution pour les différents problèmes de gestion de syndicat.

# BIBLIOGRAPHIE

## **Livres :**

- Langage de modélisation UML, Frédéric Julliard
- Servlets : Programmation d'applications Web avec Java(J2EE) Benjamin AUMAILLE
- Java la synthèse : concepts, architectures, Framework
- My SQL 5: installation mise en œuvre administration programmation, Cyril Thibaud

## **Sites Web :**

- <http://uml.free.fr/index-cours.html>
- <http://www.codejava.net/>
- <https://www.mysql.fr/>
- <https://spring.io/>
- <https://openclassrooms.com/fr/>
- <https://angular.io/>
- <https://www.w3schools.com/>

# ANNEXE A

Faculté des Sciences Juridiques, Economiques et Sociales Mohammedia : (FSJES Mohammedia ou FSJESM), est un établissement d'enseignement supérieur dont le but est de développer des programmes d'enseignement et de recherche dans les domaines juridique, économique et social. A cette fin, elle remplit trois missions fondamentales : l'enseignement, la recherche, la culture et l'information

Faculté des Sciences Juridiques, Economiques et Sociales Mohammedia (FSJES Mohammedia ou FSJESM) est un établissement d'enseignement supérieur ayant pour objectif de développer des programmes d'enseignement et de recherche dans les domaines juridique, économique et social. A cette fin, elle remplit les missions suivantes :

- La formation fondamentale et professionnelle pour les différents cycles d'enseignement (Licence, Master, Master Spécialisé, Doctorat)
- La formation continue à travers les Diplômes et Certificats d'université appropriés et adaptés, en termes d'objectifs et d'approches pédagogiques avec le monde socioprofessionnel (secteur public et privé);
- L'encouragement et la promotion de la recherche scientifique
- La promotion des activités Scientifiques, culturelles et sportives
- La mise en place de partenariats et conventions de coopération avec différents établissements nationaux et internationaux pour le montage de projets d'intérêt communs.

# Table de Matière

<b>LISTE DES FIGURES.....</b>	<b>5</b>
<b>SOMMAIRE .....</b>	<b>6</b>
<b>INTRODUCTION GENERALE .....</b>	<b>7</b>
CADRE GENERALE DU PROJET .....	7
PROBLEMATIQUE .....	7
PLAN DU RAPPORT .....	8
<b>CHAPITRE I : ETUDE, SPECIFICATION ET ANALYSE DE BESOINS.....</b>	<b>9</b>
1.2 INTRODUCTION .....	10
1.2 ETUDE DE L'EXISTANT.....	10
1.3 SPECIFICATION DES BESOINS .....	11
1.3.1 Introduction .....	11
1.3.2 Spécification de besoins .....	12
1.3.2.1 Les acteurs .....	12
1.3.3 Décomposer le système en partie .....	13
1.3.4 Diagramme des cas d'utilisation .....	15
1.4 CONCLUSION .....	17
<b>CHAPITRE II : CONCEPTION ET MODELISATION .....</b>	<b>19</b>
2.1 INTRODUCTION.....	20
2.2 DIAGRAMME DE CLASSE .....	21

2.3 MODELISATION DES DONNEES .....	22
2.3.1 Description des données .....	22
2.3.2 Modèle conceptuel de données MCD.....	25
2.3.3 Modèle Logique de données, MLD .....	27
2.4 CONCLUSION .....	28
<b>CHAPITRE III : REALISATION .....</b>	<b>29</b>
3.1 INTRODUCTION.....	30
3.2 ENVIRONNEMENT LOGICIEL .....	30
3.2.1 Choix de technologies de développement coté frontend .....	30
3.2.1.1 Framework Angular.....	30
3.2.1.2 Bibliothèque Bootstrap .....	32
3.2.1.3 Outils de développement .....	32
3.2.2 Choix de technologies de développement coté backend .....	33
3.2.2.1 JAVA EE.....	33
3.2.2.2 Framework Spring et Spring Boot .....	34
3.2.2.3 Les API REST .....	38
3.2.2.4 Outils de development .....	40
3.3 ETAPE DE REALISATION .....	40
3.4 LES INTERFACES DE L'APPLICATION .....	42
3.5 CONCLUSION .....	47
<b>CONCLUSION GENERALE .....</b>	<b>48</b>

<b>BIBLIOGRAPHIE .....</b>	<b>49</b>
<b>ANNEXE A .....</b>	<b>50</b>
<b>TABLE DE MATIÈRE .....</b>	<b>51</b>