



UNIVERSITÉ MOHAMMED-V DE RABAT
ÉCOLE NATIONALE SUPÉRIEURE
D'INFORMATIQUE ET D'ANALYSE DES
SYSTÈMES
ENSIAS



DÉPARTEMENT GÉNIE LOGICIEL

PROJET DE FIN D'ANNÉE

Plateforme de Gestion de Syndic de Copropriété

Élèves ingénieurs :
KARMOUCHI ASMAE
KENDI MOHAMMED AMINE

Professeur Encadrant :
M.BAINA SALAH

Jury Composé De :

M.BAINA SALAH
M.MAHMOUD NASSAR

Année Académique 2023/2024

Remerciements

Nous tenons à exprimer notre profonde gratitude envers notre encadrant, M.BAINA SALAH , pour son soutien, ses conseils précieux et son dévouement tout au long de la réalisation de ce project.

Nous souhaitons également remercier chaleureusement le jury de notre soutenance M. MAHMOUD NASSAR et M.BAINA SALAH pour leur temps ,leur attention et leur évaluation de ce project. Vos commentaires éclairés et votre expertise seront d'une grande valeur pour nous améliorer et affiner notre travail.

Nous sommes profondément reconnaissants envers toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce project et à notre parcours académique à l'école ENSIAS.

Résumé

Ce rapport présente le développement d'une plateforme de gestion de syndic de copropriété, conçue pour répondre aux défis et aux besoins spécifiques de la gestion des immeubles en copropriété au Maroc, conformément à la loi n° 18-00 relative au statut de la copropriété des immeubles bâties.

L'objectif principal du projet était de créer un système d'information intégré et automatisé qui facilite la gestion des résidences, des charges, des tâches, des incidents, du reporting, des alertes et notifications des assemblées, ainsi que des paiements des résidents. Ce système vise à offrir une solution robuste, sécurisée et conviviale pour l'ensemble du processus de gestion.

En suivant la méthodologie MERISE, nous avons conçu et mis en œuvre une architecture modulaire et évolutive, en prenant en compte les impératifs de sécurité et de fiabilité. Le travail collaboratif a été facilité par l'utilisation de plateformes modernes de gestion de code et de développement, garantissant une intégration continue et une gestion efficace des versions.

Le rapport détaille les étapes de conception, de réalisation et de déploiement de la plateforme, mettant en avant les avantages obtenus et les défis relevés au cours du projet. En somme, cette plateforme représente une avancée significative dans la gestion des copropriétés au Maroc, offrant une solution pratique et innovante adaptée aux exigences des utilisateurs et du cadre légal en vigueur.

Mots-clés : Gestion de copropriété, plateforme de gestion, loi n° 18-00, MERISE, système d'information, automatisation, sécurité, fiabilité, architecture modulaire, intégration continue, gestion des résidences, gestion des charges, gestion des tâches, gestion des incidents, reporting, alertes et notifications, paiements des résidents

Table des matières

Remerciements	1
Résumé	2
Table des acronymes	7
Introduction	8
1 Contexte général du projet	10
1.1 Contexte du sujet	10
1.2 Loi n° 18-00 relative au statut de la copropriété [1]	10
1.3 Problématique	11
1.4 Objectifs du projet [7] [8]	11
1.5 Conclusion	12
2 Analyse des besoins	13
2.1 Critique de l'existant	13
2.2 Présentation du travail demandé	14
2.3 Spécification des besoins	15
2.3.1 Besoins fonctionnels	15
2.3.2 Besoins non fonctionnels	15
2.4 Analyse des besoins	16
2.4.1 Identification des parties prenantes	16
2.4.2 Diagramme de cas d'utilisation	16
2.5 Diagramme de séquence	18
3 Conception	21
3.1 Dictionnaire de données	21
3.2 Modèle Conceptuel de Données(MCD)	26
3.3 Modèle Logique de Données(MLD)	28
3.4 Règles de traitement	29
3.5 Modèle conceptuel de traitement (MCT)	32
3.6 Patterns de Conception	35
4 Réalisation et mise en oeuvre	36
4.1 Environnement de développement	36
4.1.1 Espace de Travail : IntelliJ IDEA	36
4.1.2 Collaboration : GitHub	36
4.1.3 Base de Données : Mysql	37
4.1.4 Backend	37

4.1.5	Sécurité : jBCrypt	38
4.1.6	Tests unitaires et les tests d'intégration [10]	38
4.1.7	Frontend	38
4.2	Résultats	40
4.2.1	Structure de l'application	41
4.3	Présentation des interfaces	42
4.3.1	Admin :	43
4.3.2	Syndic :	46
4.3.3	Résident :	58
4.4	Déploiement de l'application	61
4.4.1	Déploiement avec Docker Compose	62
4.4.2	Déploiement avec Kubernetes	62
Conclusion		64

Table des figures

2.1	Exemple de tableau excel pour la gestion de copropriété	14
2.2	Diagramme de cas d'utilisation pour l'administrateur	17
2.3	Diagramme de cas d'utilisation pour le syndic	17
2.4	Diagramme de cas d'utilisation pour le résident	18
2.5	Diagramme de séquence pour scénario d'ajout d'un nouveau syndic	18
2.6	Diagramme de séquence pour scénario d'insertion d'un nouveau paiement et génération du rapport	19
2.7	Diagramme de séquence pour scénario d'insertion d'un incident et génération du rapport	19
2.8	Diagramme de séquence pour scénario de consultation des paiements et génération du rapport	20
3.1	MCD	27
3.2	MLD	28
3.3	MCT : Déclaration d'un nouvelle résidence	32
3.4	MCT : Déclaration du paiement d'un résident	33
3.5	MCT : Signalement d'un incendie	33
3.6	MCT : Plannification d'un assemblé général	34
3.7	MCT : Déclaration d'une tâche	34
4.1	intelij	36
4.2	github	37
4.3	MySQL	37
4.4	JUnit et Mockito	38
4.5	HTML et CSS	39
4.6	JavaScript	39
4.7	Tailwind CSS	39
4.8	Chart.js	40
4.9	Moment.js	40
4.10	Résultat	40
4.11	Architecture de l'application	41
4.12	Landing Page	43
4.13	Login page	43
4.14	Dashboard - Admin	44
4.15	Add Syndic - Admin	44
4.16	Add Member - Admin	45
4.17	Liste des syndics - Admin	45
4.18	Liste des membres - Admin	46
4.19	Payment Flow - Admin	46
4.20	Dashboard Syndic	47

4.21 Profil Syndic	48
4.22 Liste des membres - Syndic	49
4.23 Info membre - Syndic	49
4.24 Mail Alerte de paiement	49
4.25 Page des paiement - syndic	50
4.26 Add payment - syndic	51
4.27 Mail de confirmation de paiement	51
4.28 PDF - Bulletin de paiement	52
4.29 Interface Incendie - syndic	52
4.30 Mail Notif Incendie	53
4.31 Interface meeting - Syndic	54
4.32 Mail Invitation meet	55
4.33 PDF - PV meeting	55
4.34 Pages des tâches - Syndic	56
4.35 Pages des charges - Syndic	57
4.36 Page de gestion de flux de paiement de la résidence	57
4.37 PDF - flux de paiement	58
4.38 Profile - Member	59
4.39 Info Résidence Member	59
4.40 Page Incident - Member	60
4.41 Page Reclamation - Member	60
4.42 Page des tâches - Member	61
4.43 Page News - Member	61
4.44 Enviroment avec les deux conteneurs	62
4.45 Architecture deployment kubernetes	63

Table des acronymes

MCD : Modèle Conceptuel des Données

MLD : Modèle Logique des Données

MCT : Modèle Conceptuel des Traitements

IDE : Integrated Development Environment

HTTP : Hypertext Transfer Protocol

CRUD : CREATE, READ, UPDATE and DELETE

API : Application Programming Interface

HTML : HyperText Markup Language

CSS : Cascading Style Sheets

JSON : JavaScript Object Notation

KPI : Key Performance Indicator

YAML : Yet Another Markup Language

Introduction

La gestion des syndicats de copropriété au Maroc est un domaine essentiel pour assurer le bon fonctionnement et l'entretien des immeubles partagés. Encadrée par la loi n° 18-00, cette gestion vise à organiser et à réglementer les relations entre les copropriétaires, en définissant clairement les droits et les obligations des syndicats de copropriété. Le syndicat de copropriété joue un rôle crucial en représentant les intérêts collectifs des copropriétaires, en veillant à l'entretien des parties communes de l'immeuble et en prenant les décisions nécessaires pour le bon fonctionnement de la copropriété.

Cependant, la gestion des syndicats de copropriété peut être complexe et laborieuse. La multiplicité des parties prenantes, les nombreuses tâches administratives à accomplir, et les éventuels conflits entre copropriétaires rendent cette gestion particulièrement difficile. C'est dans ce contexte que l'informatisation de la gestion des syndicats de copropriété prend tout son sens [3].

Enoncé du problème

La gestion des syndicats de copropriété au Maroc, bien que encadrée par la loi, souffre de plusieurs défis majeurs. Parmi ceux-ci, on trouve la difficulté de coordination entre les différents copropriétaires et le syndic, les lourdeurs administratives liées à la gestion quotidienne et à la tenue des assemblées générales, le manque d'alertes et de notifications pour des événements importants tels que les assemblées générales et les paiements, et les conflits fréquents dus à la mauvaise gestion des ressources et à la répartition des charges [5].

Ces défis soulignent la nécessité d'une solution moderne et efficace pour la gestion des syndicats de copropriété. Comment peut-on automatiser les tâches administratives courantes pour réduire la charge de travail des syndics ? Quels outils de communication et de reporting peuvent être mis en place pour garantir une meilleure transparence et faciliter la résolution des conflits ? Comment peut-on optimiser la gestion des charges, des incidents, et des tâches pour une gestion plus efficace des parties communes et des finances de la copropriété ? Quelles solutions d'alertes et de notifications peuvent être intégrées pour les événements clés tels que les assemblées générales et les échéances de paiement ? Comment peut-on proposer une interface utilisateur intuitive accessible via différents supports ?

Objectifs

Face à ces problématiques, notre projet vise à développer une Plateforme de Gestion de Syndic de Copropriété qui permettra de simplifier la gestion administrative en automatisant les tâches administratives courantes pour réduire la charge de travail des syndics. Il cherchera également à améliorer la communication et la transparence en mettant en place des outils de communication et de reporting pour garantir une meilleure transparence et faciliter la résolution des conflits. De plus, le projet vise à optimiser la gestion des ressources en fournissant des fonctionnalités avancées de gestion des charges, des incidents, et des tâches

pour une gestion plus efficace des parties communes et des finances de la copropriété. Il facilitera le suivi et les notifications en intégrant des systèmes d'alertes et de notifications pour les événements clés tels que les assemblées générales et les échéances de paiement. Enfin, le projet offrira un accès facilité en proposant une interface utilisateur intuitive accessible via différents supports.

Plan

Ce rapport est structuré comme suit :

Chapitre 1 présente le contexte général du projet.

Chapitre 2 se concentre sur l'analyse des besoins.

Chapitre 3 détaille la conception.

Chapitre 4 aborde la réalisation et la mise en œuvre.

Chapitre 1

Contexte général du projet

Ce chapitre plonge dans la réalité de la gestion des syndicats de copropriété au Maroc, encadrée par la loi numéro 18-00. Ce cadre législatif pose les bases essentielles pour comprendre les enjeux et les défis spécifiques rencontrés dans ce domaine. Ce chapitre explore ainsi les besoins fondamentaux de la gestion des syndicats, les objectifs clés du projet, son périmètre d'action et propose une vue d'ensemble structurée du plan détaillé à suivre.

1.1 Contexte du sujet

Au Maroc, la gestion des syndicats de copropriété est régie par la loi numéro 18-00 relative au statut de la copropriété des immeubles bâtis. Cette loi établit un cadre juridique visant à organiser et à réglementer les relations entre les copropriétaires, ainsi qu'à définir les droits et les obligations des syndicats de copropriété.

La mise en place d'un syndicat de copropriété est une étape essentielle dans la gestion d'un immeuble en copropriété au Maroc. Ce syndicat est chargé de représenter les intérêts collectifs des copropriétaires, de veiller à l'entretien et à la gestion des parties communes de l'immeuble, ainsi que de prendre toutes les décisions nécessaires au bon fonctionnement de la copropriété.

Cependant, la gestion d'un syndicat de copropriété peut s'avérer complexe et laborieuse, notamment en raison du nombre de parties prenantes impliquées, des multiples tâches administratives à accomplir et des éventuels conflits entre copropriétaires. C'est dans ce contexte que l'informatisation de la gestion des syndicats de copropriété prend tout son sens.

1.2 Loi n° 18-00 relative au statut de la copropriété [1]

La loi numéro 18-00, adoptée par le Dahir n° 1-02-298 du 25 rejeb 1423 (3 octobre 2002), est le texte fondamental qui régit la copropriété au Maroc. Elle fixe les règles relatives à la création, l'organisation, et le fonctionnement des syndicats de copropriété. Cette loi vise à garantir une gestion transparente et efficace des immeubles en copropriété, tout en protégeant les droits des copropriétaires.

Les principales dispositions de la loi 18-00 incluent :

- **La constitution du syndicat de copropriété :** La loi exige la création d'un syndicat dès lors qu'un immeuble est divisé en lots de copropriété. Ce syndicat est composé de l'ensemble des copropriétaires.

- **Les organes du syndicat** : Le syndicat est dirigé par une assemblée générale des copropriétaires, qui élit un syndic chargé de l'exécution des décisions de l'assemblée et de la gestion quotidienne de l'immeuble.
- **Les droits et obligations des copropriétaires** : La loi définit les droits des copropriétaires, notamment en matière de jouissance des parties privatives et communes, ainsi que leurs obligations, telles que la contribution aux charges de copropriété.
- **La gestion des parties communes** : Le syndic est responsable de l'entretien, de la réparation et, le cas échéant, de la rénovation des parties communes de l'immeuble.
- **Les assemblées générales** : La loi fixe les modalités de convocation, de tenue et de prise de décision des assemblées générales des copropriétaires.

Ces dispositions législatives visent à créer un cadre de gestion harmonieux et à prévenir les litiges entre copropriétaires. Toutefois, en pratique, leur application peut rencontrer des difficultés, notamment en raison de la complexité des tâches administratives et de la diversité des intérêts des copropriétaires. D'où l'importance d'une solution informatique pour optimiser la gestion des syndicats de copropriété.

1.3 Problématique

La gestion des syndicats de copropriété au Maroc, bien que encadrée par la loi, souffre de plusieurs défis majeurs [2]. Parmi ceux-ci, on trouve :

- La difficulté de coordination entre les différents copropriétaires et le syndic.
- Les lourdeurs administratives liées à la gestion quotidienne et à la tenue des assemblées générales.
- Le manque d'alertes et de notifications pour des événements importants tels que les assemblées générales et les paiements.
- Les conflits fréquents dus à la mauvaise gestion des ressources et à la répartition des charges.

Ces défis soulignent la nécessité d'une solution moderne et efficace pour la gestion des syndicats de copropriété.

1.4 Objectifs du projet [7] [8]

Face à ces problématiques, notre projet vise à développer une Plateforme de Gestion de Syndic de Copropriété qui permettra de :

- **Simplifier la gestion administrative** : Automatiser les tâches administratives courantes pour réduire la charge de travail des syndics.
- **Améliorer la communication et la transparence** : Mettre en place des outils de communication et de reporting pour garantir une meilleure transparence et faciliter la résolution des conflits.
- **Optimiser la gestion des ressources** : Fournir des fonctionnalités avancées de gestion des charges, des incidents, et des tâches pour une gestion plus efficace des parties communes et des finances de la copropriété.
- **Faciliter le suivi et les notifications** : Intégrer des systèmes d'alertes et de notifications pour les événements clés tels que les assemblées générales et les échéances de paiement.

- **Offrir un accès facilité** : Proposer une interface utilisateur intuitive accessible via différents supports

1.5 Conclusion

Ce premier chapitre a pour but de poser le contexte général et de souligner l'importance d'un cadre législatif clair pour la gestion des syndicats de copropriété au Maroc, tout en introduisant la nécessité de moderniser cette gestion par le biais de l'informatisation. Le projet de Plateforme de Gestion de Syndic de Copropriété que nous proposons répond directement aux problématiques identifiées et vise à atteindre des objectifs précis pour améliorer significativement la gestion des copropriétés.

Chapitre 2

Analyse des besoins

L'analyse des besoins est une étape cruciale dans le développement de tout système d'information. Elle permet d'identifier et de comprendre les exigences des utilisateurs et des parties prenantes, afin de concevoir une solution qui réponde efficacement à leurs attentes. Dans le cadre de notre projet de plateforme de gestion de syndic de copropriété, cette analyse se focalise sur les besoins des différents acteurs impliqués dans la gestion des copropriétés au Maroc.

2.1 Critique de l'existant

Avant de développer notre propre solution, il est essentiel de comprendre les limitations et les lacunes des systèmes de gestion de syndic de copropriété actuellement disponibles au Maroc. Cette critique de l'existant permet de mieux cerner les besoins non satisfaits et d'orienter le développement de notre plateforme pour répondre de manière optimale aux attentes des utilisateurs.

Systèmes de gestion traditionnels

Les systèmes de gestion traditionnels pour les syndics de copropriété reposent souvent sur des méthodes manuelles ou semi-informatisées. Voici les principales limitations de ces approches :

- **Gestion manuelle** : La plupart des syndicats de copropriété utilisent encore des méthodes papier ou des tableurs pour gérer les informations, ce qui entraîne des erreurs humaines, une perte de documents, et une inefficacité générale.
- **Communication inefficace** : Les notifications et les communications importantes sont souvent envoyées par courrier postal ou par téléphone, ce qui peut entraîner des retards et des malentendus.
- **Manque de transparence** : Les copropriétaires n'ont pas toujours un accès facile aux informations financières et aux décisions prises par le syndic, ce qui peut générer de la méfiance et des conflits.
- **Difficulté de suivi des paiements** : Le suivi manuel des paiements des charges peut entraîner des erreurs et des oublis, rendant difficile la gestion des arriérés et des litiges financiers.

Systèmes informatisés existants

Bien que certains syndicats aient adopté des solutions informatisées, celles-ci reposent souvent principalement sur des tableurs tels qu'Excel. Ces solutions présentent également des limitations notables :

- **Dépendance à Excel** : L'utilisation d'Excel pour gérer les informations de copropriété est courante, mais ce logiciel n'est pas conçu pour gérer des bases de données complexes ou pour offrir une collaboration en temps réel. Les feuilles de calcul peuvent devenir difficiles à gérer et sujettes aux erreurs. Comme illustré dans la figure 2.1

cours-gratuit.com--id-10836 [Protected View] - Excel									
File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do									
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing									
C42	:	X	✓	f(x)	D	E	F	G	H
A	B	C	D	E	F	G	H	I	J
4	Lots/descriptions	Propriétaires	Générales	Escalier	Eau	Charge1	Charge2	Travaux	Provision
5	Tantièmes	Tantièmes	Prorata	Prorata	Prorata	Prorata	Prorata	Tantièmes	
6	Lot n°1	Nom	ATCHOUM	143	143	143	143	143	143
7	RDC : Appartement - Jardin	Adres.							
8	Ville								
9	Tel								
10	Email		1000	1000	1000	1000	1000	1000	1000
11	Lot n°2	Nom	ATCHOUM	162	162	162	162	162	162
12	RDC : Appartement - Jardin - Chaufferie	Adres.							
13	Ville								
14	Tel								
15	Email		1000	1000	1000	1000	1000	1000	1000
16	Lot n°3	Nom	GRINCHEUX	91	91	91	91	91	91
17	RDC : Local	Adres.							
18	Ville								
19	Tel								
20	Email		1000	1000	1000	1000	1000	1000	1000

FIGURE 2.1 – Exemple de tableau excel pour la gestion de copropriété

- **Complexité d'utilisation** : Certains systèmes sont complexes et peu intuitifs, nécessitant une formation approfondie pour les utilisateurs, ce qui peut décourager leur adoption.
- **Personnalisation limitée** : Les solutions génériques ne sont pas toujours adaptées aux spécificités légales et culturelles du Maroc, notamment en ce qui concerne la loi n° 18-00 relative à la copropriété des immeubles bâtis.
- **Sécurité insuffisante** : La protection des données sensibles des copropriétaires est souvent insuffisante, exposant les informations à des risques de violation de confidentialité.

La critique des solutions existantes met en évidence les défis et les lacunes actuels dans la gestion des syndicats de copropriété au Maroc. Ces limitations justifient la nécessité de développer une plateforme innovante et adaptée aux besoins spécifiques du marché marocain. Notre projet vise à surmonter ces obstacles en offrant une solution sécurisée, conviviale et conforme aux exigences légales, tout en améliorant la transparence et l'efficacité de la gestion des copropriétés.

2.2 Présentation du travail demandé

À la lumière des lacunes identifiées dans la section précédente, le projet a pour objectif de développer une plateforme intégrée et automatisée pour la gestion des syndicats de copropriété au Maroc. Contrairement aux systèmes actuels, qui reposent principalement sur

des outils comme Excel, cette nouvelle solution vise à simplifier la gestion administrative, améliorer la communication entre les copropriétaires et les syndics, optimiser la gestion financière et assurer une transparence accrue.

Les fonctionnalités attendues de cette plateforme incluent la gestion des résidences, des charges, des tâches et des incidents, ainsi que la planification et la tenue des assemblées générales. En outre, elle doit permettre l'envoi d'alertes et de notifications automatiques. La solution doit également proposer des options de gestion de paiement, fournir des outils de reporting détaillés, et garantir une interface utilisateur intuitive. Enfin, elle doit respecter des critères stricts de performance et de sécurité pour répondre aux besoins des utilisateurs de manière efficace et fiable.

2.3 Spécification des besoins

Cette section détaille les besoins fonctionnels et non fonctionnels de la plateforme de gestion des syndicats de copropriété. Elle vise à définir précisément les attentes des utilisateurs et les contraintes techniques à respecter pour garantir le succès du projet.

2.3.1 Besoins fonctionnels

Les besoins fonctionnels décrivent les fonctionnalités que le système doit offrir pour répondre aux attentes des utilisateurs :

- **Gestion des résidences** : Permettre l'enregistrement et la mise à jour des informations sur les résidences et les copropriétaires.
- **Gestion des syndics** : Faciliter la gestion et le suivi des syndics, en offrant des outils pour gérer leurs tâches, leurs responsabilités, et leurs interactions avec les copropriétaires.
- **Gestion des membres** : Permettre la gestion efficace des informations et des interactions des membres de la copropriété, en assurant une communication fluide et une gestion transparente des données des membres.
- **Gestion des charges** : Faciliter la répartition, la collecte et le suivi des charges de copropriété.
- **Gestion des tâches et des incidents** : Suivre les tâches d'entretien et gérer les incidents signalés par les résidents.
- **Assemblées générales** : Planifier et organiser les assemblées générales, y compris les notifications, des ordres du jour, et des comptes rendus.
- **Alertes et notifications** : Envoyer des notifications automatiques pour les événements importants comme les assemblées générales et les échéances de paiement.
- **Gestion des paiements** : Offrir des options de paiement en ligne sécurisées et suivre les paiements effectués par les résidents.
- **Reporting** : Générer des rapports détaillés sur la gestion des charges, des incidents, des paiements, et d'autres aspects de la copropriété.

2.3.2 Besoins non fonctionnels

Les besoins non fonctionnels définissent les critères de performance, de sécurité, et d'utilisabilité que le système doit respecter :

- **Sécurité** : Assurer la protection des données des utilisateurs et garantir la confidentialité des informations personnelles et financières.
- **Performance** : Offrir des temps de réponse rapides et une disponibilité élevée pour les utilisateurs.
- **Utilisabilité** : Fournir une interface utilisateur intuitive et accessible, compatible avec différents supports (ordinateurs, tablettes, smartphones).
- **Scalabilité** : Permettre l'ajout de nouvelles fonctionnalités et le support d'un nombre croissant d'utilisateurs sans dégradation des performances.
- **Fiabilité** : Garantir un fonctionnement stable et minimiser les risques de dysfonctionnements ou de pannes.
- **Conformité légale** : Respecter les réglementations en vigueur, notamment la loi n° 18-00 relative au statut de la copropriété des immeubles bâties.

Ces spécifications des besoins sont essentielles pour guider la conception et le développement de la plateforme, en assurant qu'elle répondra aux attentes des utilisateurs tout en respectant les contraintes techniques et réglementaires.

2.4 Analyse des besoins

Cette section vise à identifier les parties prenantes du projet, à prioriser les besoins fonctionnels et non fonctionnels, et à présenter le diagramme de cas d'utilisation pour illustrer les interactions entre les acteurs du système, ainsi que le diagramme de séquence.

2.4.1 Identification des parties prenantes

Les parties prenantes du projet sont les individus ou les entités qui seront affectés par le développement et l'utilisation de la plateforme de gestion de syndic de copropriété. Elles peuvent inclure :

- Administrateurs : L'utilisateur responsable de la configuration et de la maintenance de la plateforme.
- Syndics : Les personnes ou les entreprises chargées de la gestion quotidienne de la copropriété.
- Résidents : Les personnes habitant dans les résidences en copropriété, utilisant la plateforme pour signaler des incidents,etc.
- Fournisseurs : Les individus responsables de la réalisation des travaux dans la copropriété.

2.4.2 Diagramme de cas d'utilisation

Ci-dessous, vous trouverez les diagrammes de cas d'utilisation pour les différents acteurs du système :

La figure 2.2 présente les fonctionnalités disponibles pour l'administrateur, y compris la gestion des résidences, des charges, des tâches, et des assemblées générales.

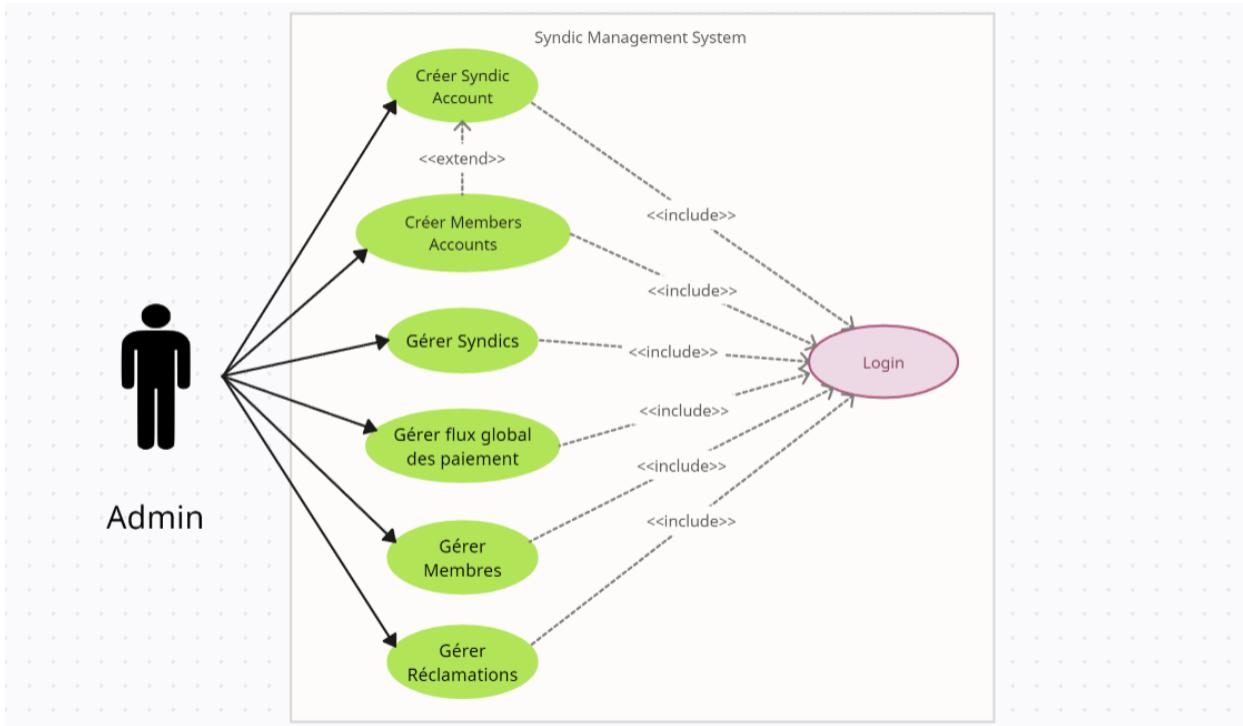


FIGURE 2.2 – Diagramme de cas d'utilisation pour l'administrateur

La figure 2.3 illustre les différentes interactions possibles pour le syndic, telles que la gestion des paiement, des charges, et des tâches de maintenance.

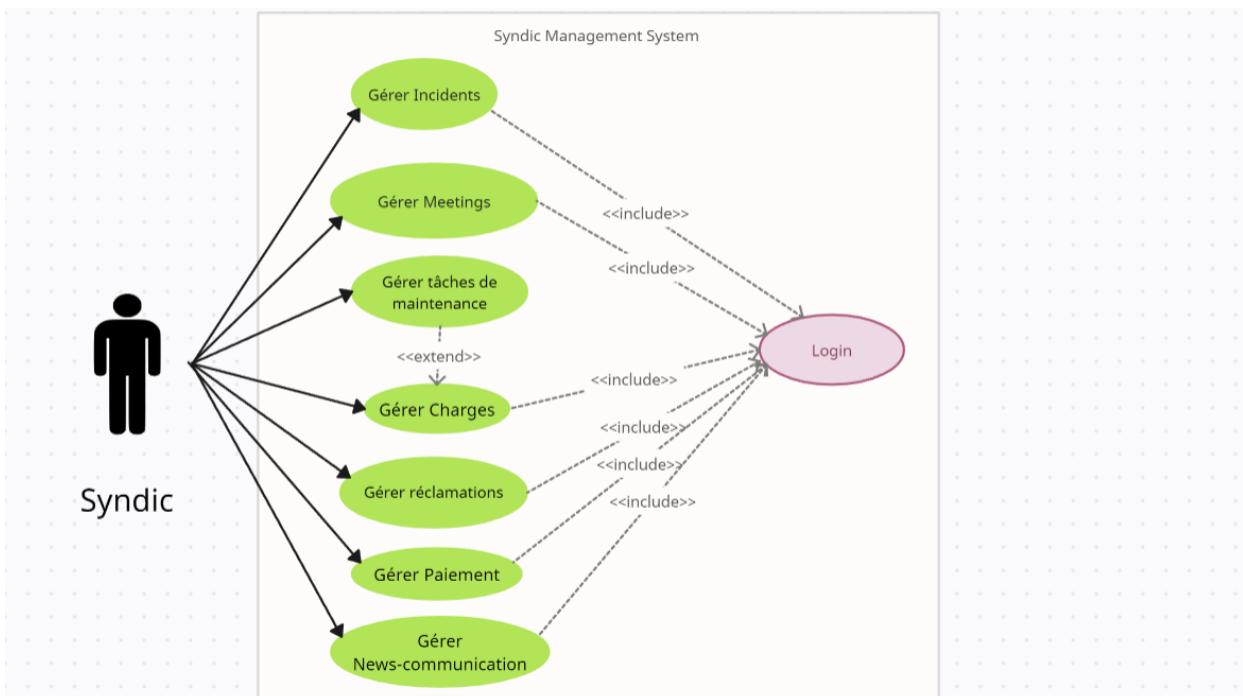


FIGURE 2.3 – Diagramme de cas d'utilisation pour le syndic

La figure 3.7 décrit les fonctionnalités accessibles pour le résident, notamment l'annonce des incidents et la participation aux assemblées générales.

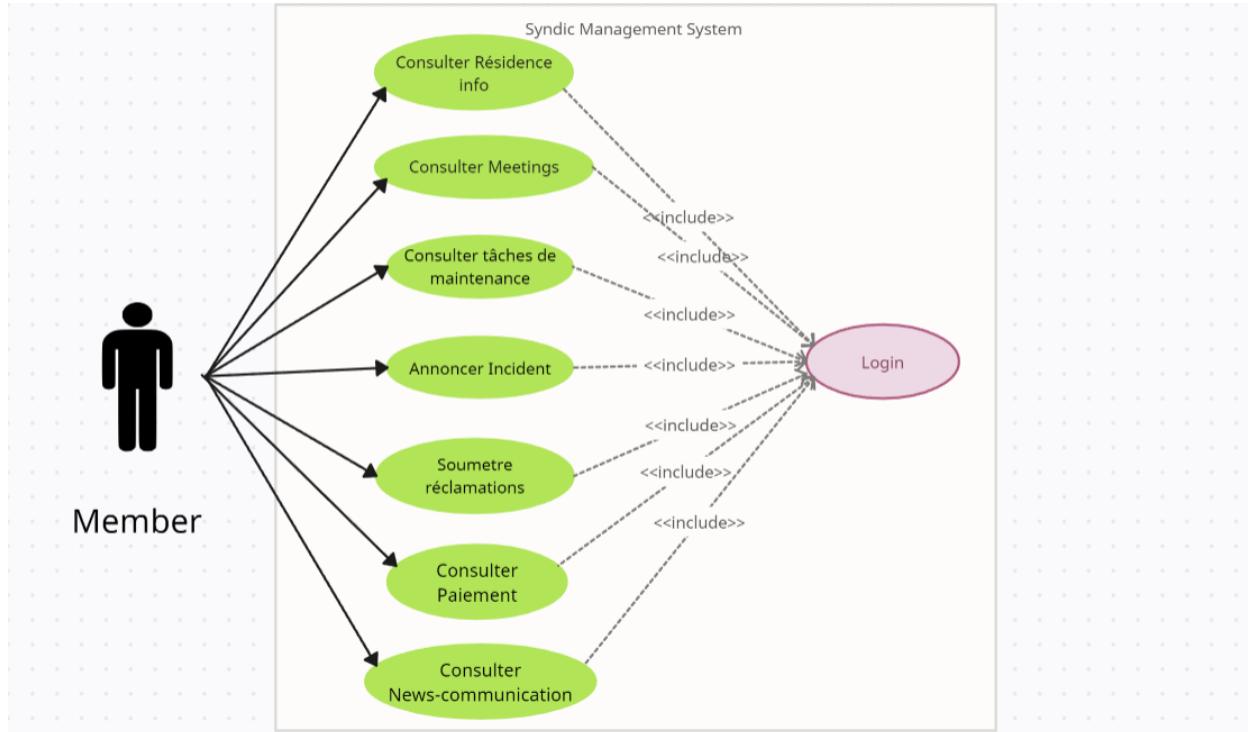


FIGURE 2.4 – Diagramme de cas d'utilisation pour le résident

2.5 Diagramme de séquence

Admin : add Syndic

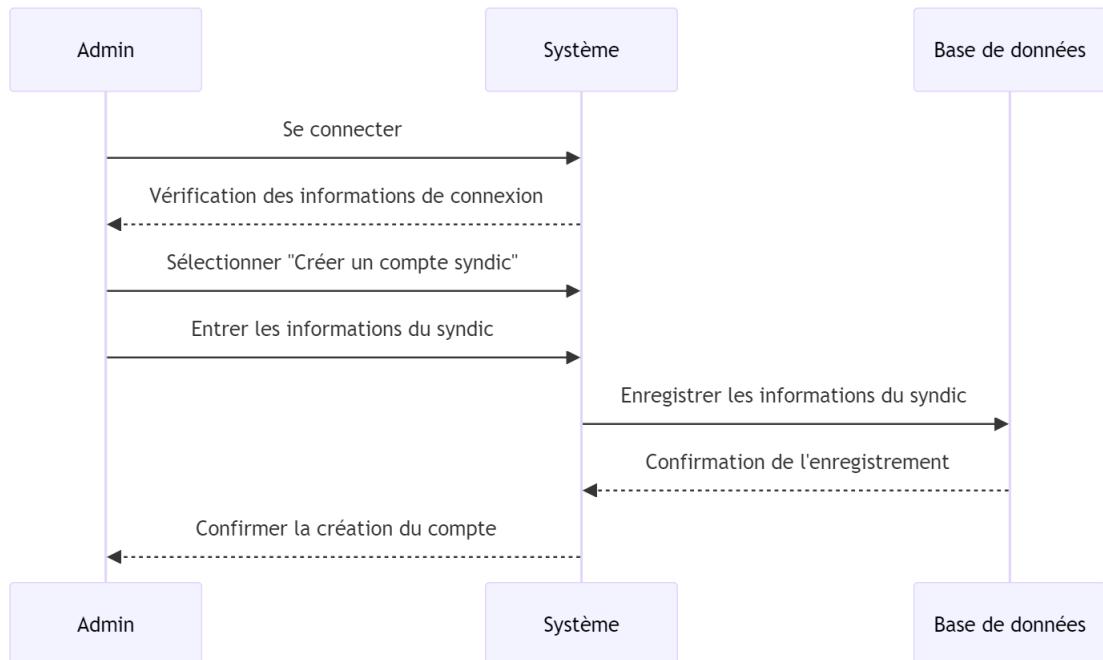


FIGURE 2.5 – Diagramme de séquence pour scénario d'ajout d'un nouveau syndic

Syndic : add Payment

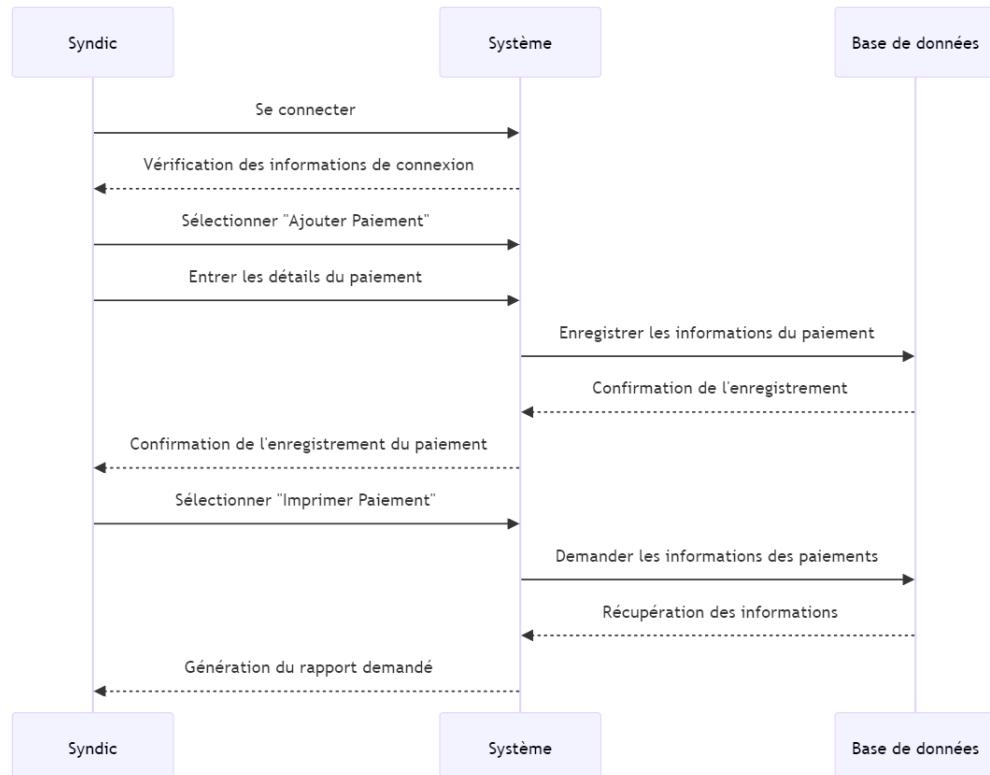


FIGURE 2.6 – Diagramme de séquence pour scénario d'insertion d'un nouveau paiement et génération du rapport

Syndic : add Incident

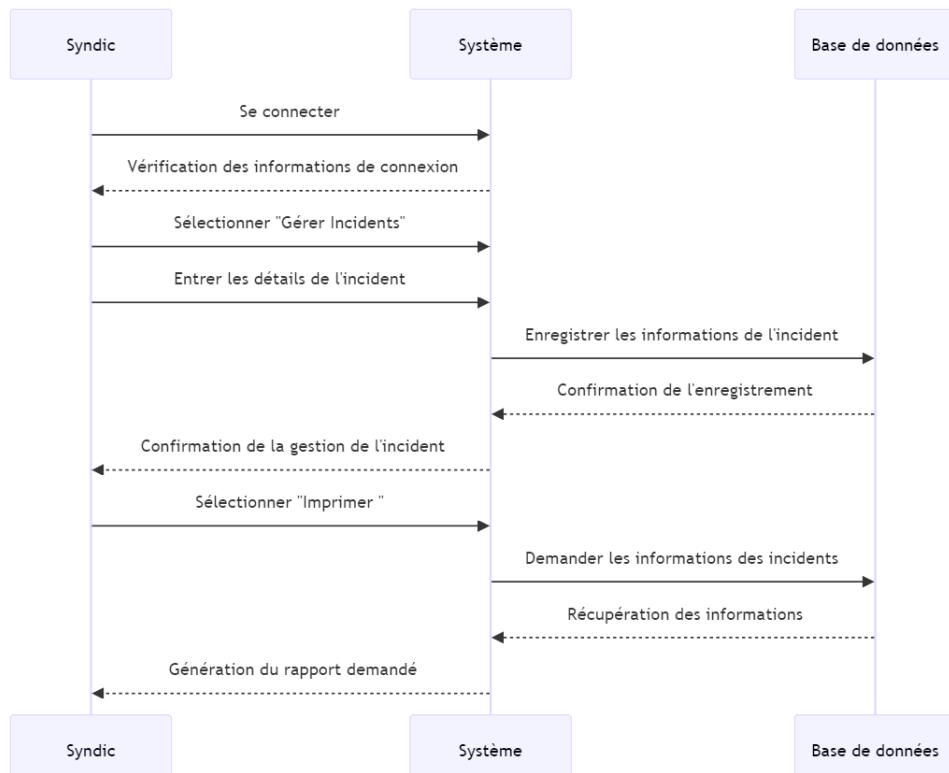


FIGURE 2.7 – Diagramme de séquence pour scénario d'insertion d'un incident et génération du rapport

Résident : Consultation of payments

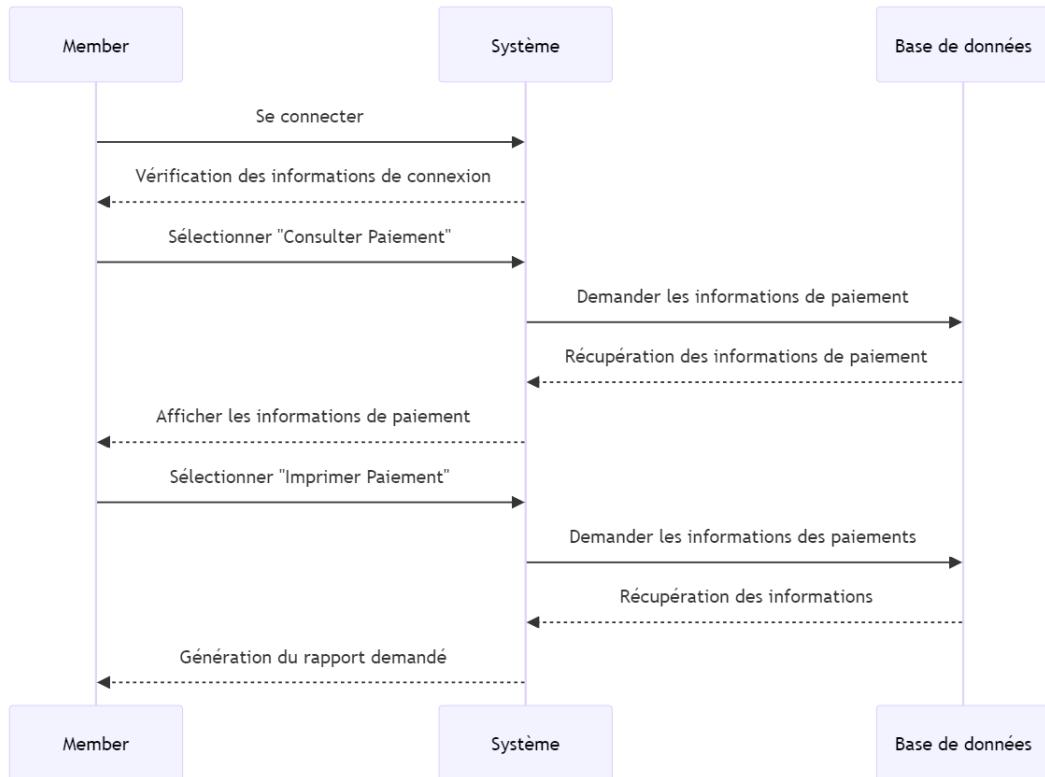


FIGURE 2.8 – Diagramme de séquence pour scénario de consultation des paiements et génération du rapport

Chapitre 3

Conception

La méthodologie Merise a été utilisée comme guide pour cette étape cruciale du projet. En s'appuyant sur cette méthode structurée, cette section du rapport détaille la manière dont les informations ont été organisées, représentées et structurées pour répondre aux besoins opérationnels.

La conception de la plateforme de gestion de syndic de copropriété est un pilier fondamental pour assurer l'efficacité, la cohérence et la facilité d'utilisation de la future application. Ce chapitre offre un aperçu complet des choix de conception effectués, en justifiant chaque décision en fonction des exigences identifiées et des meilleures pratiques .

3.1 Dictionnaire de données

Ce premier axe présente le dictionnaire de données exhaustif, offrant une référence complète décrivant chaque élément d'information.

Le dictionnaire de données joue un rôle crucial dans la représentation détaillée des informations relatives aux .

Entité	Attribut	Type de données	Description
Incidents	incident_id	Entier	Identifiant unique de l'incident
	incident_date	Date	Date de l'incident
	incident_type	Chaîne de caractères	Type d'incident
	incident_description	Texte	Description de l'incident
	incident_status	Chaîne de caractères	Statut de l'incident
	incident_resolution_date	Date	Date de résolution de l'incident
	incident_user_id	Entier	Identifiant de l'utilisateur qui a rapporté l'incident
Properties	property_id	Entier	Identifiant unique de la propriété
	property_code	Chaîne de caractères	Code de la propriété
	property_address	Texte	Adresse de la propriété
	property_type	Chaîne de caractères	Type de propriété
	property_size	Réel	Taille de la propriété
	co-ownership_fee	Réel	Frais de copropriété
	property_m_id	Entier	Identifiant du membre propriétaire
	property_s_id	Entier	Identifiant du syndic responsable
Members	m_id	Entier	Identifiant unique du membre

Entité	Attribut	Type de données	Description
	m_firstname m_lastname m_email m_phone m_address m_join_date m_iduser	Chaîne de caractères Chaîne de caractères Chaîne de caractères Chaîne de caractères Texte Date Entier	Prénom du membre Nom du membre Email du membre Numéro de téléphone du membre Adresse du membre Date d'adhésion du membre Identifiant utilisateur associé
Payments	payment_id payment_code payment_date payment_amount payment_method payment_type payment_account_id member_id payment_status	Entier Chaîne de caractères Date Réel Chaîne de caractères Chaîne de caractères Entier Entier Chaîne de caractères	Identifiant unique du paiement Code de paiement Date du paiement Montant du paiement Méthode de paiement Type de paiement Identifiant du compte de paiement Identifiant du membre Statut du paiement
Meetings	meeting_id meeting_date meeting_topic meeting_location meeting_type meeting_residence	Entier Date Chaîne de caractères Texte Chaîne de caractères Entier	Identifiant unique de la réunion Date de la réunion Sujet de la réunion Lieu de la réunion Type de réunion Identifiant de la résidence associée
Users	u_id u_name u_pwd u_email checkadmin	Entier Chaîne de caractères Chaîne de caractères Chaîne de caractères Booléen	Identifiant unique de l'utilisateur Nom de l'utilisateur Mot de passe de l'utilisateur Email de l'utilisateur Statut administrateur
Syndics	s_id s_firstname s_lastname s_email s_phone s_adresse s_join_date s_iduser	Entier Chaîne de caractères Chaîne de caractères Chaîne de caractères Chaîne de caractères Texte Date Entier	Identifiant unique du syndic Prénom du syndic Nom du syndic Email du syndic Numéro de téléphone du syndic Adresse du syndic Date d'adhésion du syndic Identifiant utilisateur associé
Documents	document_id document_name document_type document_path document_s_id document_created_at	Entier Chaîne de caractères Chaîne de caractères Texte Entier Date	Identifiant unique du document Nom du document Type de document Chemin d'accès au document Identifiant du syndic associé Date de création du document
Alerts	alert_id alert_type alert_description alert_status alert_s_id	Entier Chaîne de caractères Texte Chaîne de caractères Entier	Identifiant unique de l'alerte Type d'alerte Description de l'alerte Statut de l'alerte Identifiant du syndic associé

Entité	Attribut	Type de données	Description
	alert_created_at	Date	Date de création de l'alerte
Residences	residence_id	Entier	Identifiant unique de la résidence
	residence_name	Chaîne de caractères	Nom de la résidence
	residence_address	Texte	Adresse de la résidence
	residence_type	Chaîne de caractères	Type de résidence
	residence_size	Réel	Taille de la résidence
	apartment_count	Entier	Nombre d'appartements
	villa_count	Entier	Nombre de villas
	garden_count	Entier	Nombre de jardins
	pool_count	Entier	Nombre de piscines
	parking_count	Entier	Nombre de parkings
	elevators_count	Entier	Nombre d'ascenseurs
	security_system	Booléen	Système de sécurité
Tasks-maintenance	residence_s_id	Entier	Identifiant du syndic responsable
	task_id	Entier	Identifiant unique de la tâche
	task_name	Chaîne de caractères	Nom de la tâche
	task_description	Texte	Description de la tâche
	task_due_date	Date	Date limite de la tâche
	task_status	Chaîne de caractères	Statut de la tâche
	task_s_id	Entier	Identifiant du syndic associé
	task_created_at	Date	Date de création de la tâche
Suppliers	supplier_id	Entier	Identifiant unique du fournisseur
	supplier_name	Chaîne de caractères	Nom du fournisseur
	supplier_email	Chaîne de caractères	Email du fournisseur
	supplier_phone	Chaîne de caractères	Numéro de téléphone du fournisseur
	supplier_type	Chaîne de caractères	Type de fournisseur
	supplier_active	Booléen	Statut du fournisseur (actif/inactif)
	supplier_rating	Réel	Évaluation du fournisseur
	supplier_s_id	Entier	Identifiant du syndic associé
Expenses	expense_id	Entier	Identifiant unique de la dépense
	expense_code	Chaîne de caractères	Code de la dépense
	expense_supplier_id	Entier	Identifiant du fournisseur
	expense_amount	Réel	Montant de la dépense
	expense_category	Chaîne de caractères	Catégorie de la dépense
	expense_payment_date	Date	Date de paiement de la dépense
	expense_task_id	Entier	Identifiant de la tâche associée
Charges	charges_id	Entier	Identifiant unique de la charge
	charge_code	Chaîne de caractères	Code de la charge
	charge_name	Chaîne de caractères	Nom de la charge
	charge_description	Texte	Description de la charge
	charge_amount	Réel	Montant de la charge
	charge_frequency	Chaîne de caractères	Fréquence de la charge
	charge_category	Chaîne de caractères	Catégorie de la charge
	charge_due_date	Date	Date d'échéance de la charge

Entité	Attribut	Type de données	Description
	charge_residence_id	Entier	Identifiant de la résidence associée
Payment_flows	id syndic_id flow_type amount description transaction_date	Entier Entier Chaîne de caractères Réel Texte Date	Identifiant flux paiement Identifiant du syndic associé Type de flux Montant du flux Description du flux Date de la transaction
News-communicati	news_id news_title news_content news_date news_category news_attachment news_iduser	Entier Chaîne de caractères Texte Date Chaîne de caractères Chaîne de caractères Entier	Identifiant unique de la nouvelle Titre de la nouvelle Contenu de la nouvelle Date de la nouvelle Catégorie de la nouvelle Pièce jointe associée Identifiant de l'utilisateur qui a publié la nouvelle
Reclamations	reclaim_id reclaim_date reclaim_type reclaim_description reclaim_status reclaim_resolution_date reclaim_s_id reclaim_m_id	Integer Date String Text String Date Integer Integer	Identifiant unique de réclamation Date de la réclamation Type de réclamation Description de la réclamation Statut de la réclamation Date de résolution de réclamation Identifiant du syndic associé à la réclamation Identifiant du membre associé à la réclamation

Dépendances Fonctionnelles

Cette section présente les dépendances fonctionnelles des différentes entités du modèle de données pour une application de gestion de syndic de copropriété. Une dépendance fonctionnelle décrit une relation entre deux ensembles d'attributs, où la connaissance de la valeur d'un attribut (ou d'un ensemble d'attributs) permet de déterminer de manière unique la valeur d'un autre attribut (ou d'un ensemble d'attributs). Ces dépendances sont essentielles pour garantir l'intégrité et la cohérence des données au sein du système.

- incident_id** → incident_date, incident_type, incident_description, incident_status, incident_resolution_date, incident_user_id
- property_id** → property_code, property_address, property_type, property_size, co-ownership_fee, property_m_id, property_s_id
- m_id** → m_firstname, m_lastname, m_email, m_phone, m_address, m_join_date, m_iduser
- payment_id** → payment_code, payment_date, payment_amount, payment_method, payment_type, payment_account_id, member_id, payment_status
- meeting_id** → meeting_date, meeting_topic, meeting_location, meeting_type, meeting_residence

u_id	→ u_name, u_pwd, u_email, checkadmin
s_id	→ s_firstname, s_lastname, s_email, s_phone, s_adresse, s_join_date, s_iduser
document_id	→ document_name, document_type, document_path, document_s_id, document_created_at
alert_id	→ alert_type, alert_description, alert_status, alert_s_id, alert_created_at
residence_id	→ residence_name, residence_address, residence_type, residence_size, apartment_count, villa_count, garden_count, pool_count, parking_count, elevators_count, security_system, residence_s_id
task_id	→ task_name, task_description, task_due_date, task_status, task_s_id, task_created_at
supplier_id	→ supplier_name, supplier_email, supplier_phone, supplier_type, supplier_active, supplier_rating, supplier_s_id
expense_id	→ expense_code, expense_supplier_id, expense_amount, expense_category, expense_payment_date, expense_task_id
charges_id	→ charge_code, charge_name, charge_description, charge_amount, charge_frequency, charge_category, charge_due_date, charge_residence_id
id	→ syndic_id, flow_type, amount, description, transaction_date
news_id	→ news_title, news_content, news_date, news_category, news_attachment, news_iduser
reclaim_id	→ reclaim_date, reclaim_type, reclaim_description, reclaim_status, reclaim_resolution_date, reclaim_s_id, reclaim_m_id

Explications des Dépendances Fonctionnelles

Incidents : Chaque incident est identifié de manière unique par *incident_id*, qui détermine tous les autres attributs de l'incident tels que la date, le type, la description, le statut, la date de résolution et l'utilisateur qui l'a rapporté.

Properties : L'attribut *property_id* identifie de manière unique chaque propriété et permet de déterminer son code, son adresse, son type, sa taille, ses frais de copropriété ainsi que les identifiants du membre et du syndic associés.

Members : L'attribut *m_id* identifie de manière unique chaque membre et permet de déterminer son prénom, son nom, son email, son numéro de téléphone, son adresse, sa date d'adhésion et l'identifiant utilisateur associé.

Payments : Chaque paiement est identifié de manière unique par *payment_id*, qui détermine tous les autres attributs du paiement tels que le code, la date, le montant, la méthode, le type, le compte de paiement, l'identifiant du membre et le statut du paiement.

Meetings : Chaque réunion est identifiée de manière unique par *meeting_id*, qui détermine la date, le sujet, le lieu, le type et l'identifiant de la résidence associée.

Users : L'attribut *u_id* identifie de manière unique chaque utilisateur et permet de déterminer son nom, son mot de passe, son email et son statut d'administrateur.

Syndics : Chaque syndic est identifié de manière unique par *s_id*, qui détermine son prénom, son nom, son email, son numéro de téléphone, son adresse, sa date d'adhésion et l'identifiant utilisateur associé.

Documents : L'attribut *document_id* identifie de manière unique chaque document et permet de déterminer son nom, son type, son chemin d'accès, l'identifiant du syndic associé et la date de création.

Alerts : Chaque alerte est identifiée de manière unique par *alert_id*, qui détermine le type, la description, le statut, l'identifiant du syndic associé et la date de création.

Residences : L'attribut *residence_id* identifie de manière unique chaque résidence et permet de déterminer son nom, son adresse, son type, sa taille, le nombre d'appartements, de villas, de jardins, de piscines, de parkings, d'ascenseurs, son système de sécurité et l'identifiant du syndic responsable.

Tasks-maintenance : Chaque tâche de maintenance est identifiée de manière unique par *task_id*, qui détermine le nom, la description, la date limite, le statut, l'identifiant du syndic associé et la date de création.

Suppliers : L'attribut *supplier_id* identifie de manière unique chaque fournisseur et permet de déterminer son nom, son email, son numéro de téléphone, son type, son statut (actif/inactif), son évaluation et l'identifiant du syndic associé.

Expenses : Chaque dépense est identifiée de manière unique par *expense_id*, qui détermine le code, l'identifiant du fournisseur, le montant, la catégorie, la date de paiement et l'identifiant de la tâche associée.

Charges : L'attribut *charges_id* identifie de manière unique chaque charge et permet de déterminer le code, le nom, la description, le montant, la fréquence, la catégorie, la date d'échéance et l'identifiant de la résidence associée.

Payment_flows : Chaque flux de paiement est identifié de manière unique par *id*, qui détermine l'identifiant du syndic associé, le type de flux, le montant, la description et la date de transaction.

News-communication : L'attribut *news_id* identifie de manière unique chaque nouvelle et permet de déterminer le titre, le contenu, la date, la catégorie, la pièce jointe associée et l'identifiant de l'utilisateur qui a publié la nouvelle.

3.2 Modèle Conceptuel de Données(MCD)

Le deuxième axe consiste à élaborer le modèle conceptuel de données (MCD), représentant graphiquement les entités, leurs attributs et leurs relations. Ce schéma offre une vision structurée et visuelle de la base de données.

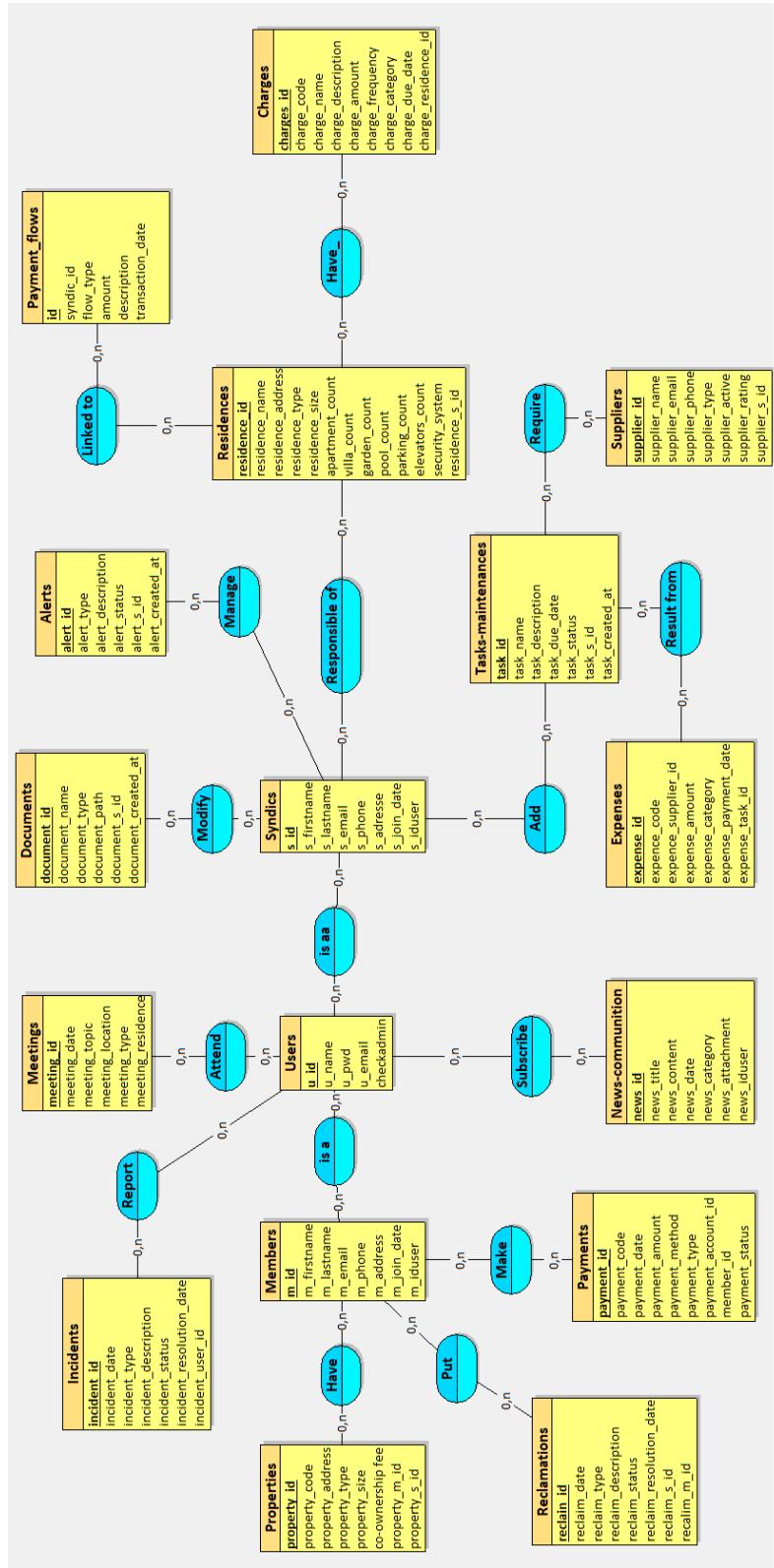


FIGURE 3.1 – MCD

Le MCD identifie les entités telles que les , ainsi que leurs attributs et leurs relations. Il illustre les connexions entre ces entités, mettant en évidence la manière dont elles interagissent dans le système de gestion de.

3.3 Modèle Logique de Données(MLD)

Le troisième axe concerne Le Modèle Logique de Données (MLD), étendant la représentation détaillée de la structure de la base de données qui sera implémentée .

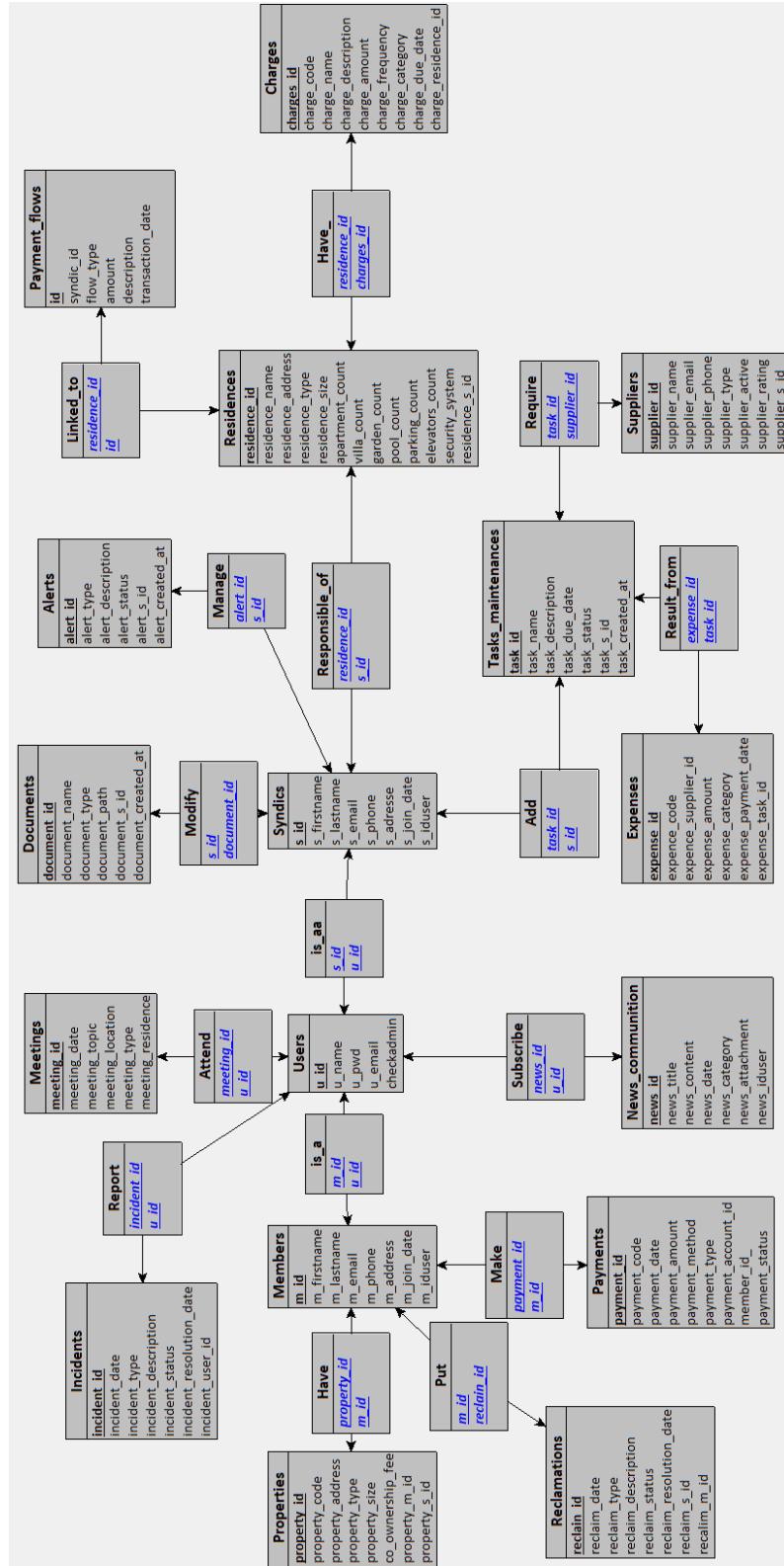


FIGURE 3.2 – MLD

Le MLD spécifie la manière dont les données seront stockées dans la base de données. Il détaille les tables, les colonnes, les types de données et les relations, offrant ainsi une structure logique pour l'implémentation dans un système de gestion de bases de données spécifique, tel que pour la plateforme de gestion de.

3.4 Règles de traitement

For Admin :

- Connexion
 - Règle : Authentifier les informations de l'utilisateur avant d'accorder l'accès.
 - Données requises : Nom d'utilisateur, Mot de passe.
 - Action : Valider par rapport aux informations d'identification stockées.
- Créer un compte Syndic
 - Règle : Accessible uniquement après une connexion réussie.
 - Données requises : Détails du syndic (nom, coordonnées, etc.).
 - Action : Valider les informations et les stocker dans la base de données.
- Créer des comptes Membres
 - Règle : Ne peut être effectué que par un administrateur.
 - Données requises : Détails du membre (nom, coordonnées, etc.).
 - Action : Valider et stocker dans la base de données, envoyer les informations d'identification aux membres.
- Gérer les Syndics
 - Règle : L'administrateur peut ajouter, modifier ou supprimer les détails du syndic.
 - Données requises : Détails du syndic, détails de la modification.
 - Action : Effectuer l'opération demandée sur la base de données des syndics.
- Gérer le flux global des paiements
 - Règle : Gérer et consulter le flux de paiement global.
 - Données requises : Détails du paiement, informations sur la transaction.
 - Action : s'assurer que les transactions sont valides.
- Gérer les Membres
 - Règle : L'administrateur peut ajouter, modifier ou supprimer les détails des membres.
 - Données requises : Détails du membre, détails de la modification.
 - Action : Effectuer l'opération demandée sur la base de données des membres.

For Syndic :

- Connexion
 - Règle : Authentifier les informations de l'utilisateur avant d'accorder l'accès.
 - Données requises : La résidence, Nom d'utilisateur, Mot de passe.
 - Action : Valider par rapport aux informations d'identification stockées.
- Gérer les Incidents

- Règle : Rapporter et gérer les incidents dans la propriété.
- Données requises : Détails de l'incident.
- Action : Enregistrer l'incident, notifier les parties concernées, suivre la résolution.

- **Gérer les Réunions**

- Règle : Planifier et gérer les réunions.
- Données requises : Détails de la réunion (date, heure, ordre du jour).
- Action : Envoyer des notifications, enregistrer les minutes.

- **Gérer les tâches de maintenance**

- Règle : Gérer les tâches de maintenance.
- Données requises : Détails de la tâche (description, personne responsable).
- Action : Assigner des tâches, suivre les progrès.

- **Gérer les Charges**

- Règle : Gérer les charges financières et leur allocation.
- Données requises : Détails de la charge.
- Action : Allouer les charges, mettre à jour les enregistrements.

- **Gérer les Réclamations**

- Règle : Gérer et répondre aux réclamations/plaintes.
- Données requises : Détails de la plainte, détails de la réponse.
- Action : Enregistrer la plainte, notifier les parties concernées, suivre la résolution.

- **Gérer les Paiements**

- Règle : Gérer la collecte et les enregistrements des paiements.
- Données requises : Détails du paiement.
- Action : Valider et enregistrer les paiements, mettre à jour les enregistrements.

- **Gérer la communication des Nouvelles**

- Règle : Communiquer les nouvelles et les mises à jour aux membres.
- Données requises : Détails de la communication.
- Action : Diffuser les nouvelles, enregistrer la communication.

For Member :

- **Connexion**

- Règle : Authentifier les informations de l'utilisateur avant d'accorder l'accès.
- Données requises : La résidence, Nom d'utilisateur, Mot de passe.
- Action : Valider par rapport aux informations d'identification stockées.

- **Consulter les informations sur la Résidence**

- Règle : Les membres peuvent consulter les informations relatives à la résidence.
- Données requises : Aucune (consultation uniquement).
- Action : Récupérer et afficher les informations.

- **Consulter les Réunions**

- Règle : Les membres peuvent consulter les horaires et les comptes rendus des réunions.
- Données requises : Aucune (consultation uniquement).

- Action : Récupérer et afficher les informations.

- **Consulter les tâches de maintenance**

- Règle : Les membres peuvent consulter les tâches de maintenance.
- Données requises : Aucune (consultation uniquement).
- Action : Récupérer et afficher les informations.

- **annoncer un Incident**

- Règle : Les membres peuvent signaler des incidents.
- Données requises : Détails de l'incident.
- Action : Enregistrer l'incident, notifier les parties concernées.

- **Soumettre des réclamations**

- Règle : Les membres peuvent soumettre des réclamations/plaintes.
- Données requises : Détails de la plainte.
- Action : Enregistrer la plainte, notifier les parties concernées.

- **Consulter les Paiements**

- Règle : Les membres peuvent consulter leur statut de paiement.
- Données requises : Aucune (consultation uniquement).
- Action : Récupérer et afficher les informations.

- **Consulter les Nouvelles-communications**

- Règle : Les membres peuvent consulter les nouvelles et les communications.
- Données requises : Aucune (consultation uniquement).
- Action : Récupérer et afficher les informations.

3.5 Modèle conceptuel de traitement (MCT)

Le quatrième axe se concentre sur le modèle conceptuel de traitement (MCT), décrivant les processus et les fonctionnalités du système. les schémas ci-dessous illustrent les opérations à réaliser sur les données.

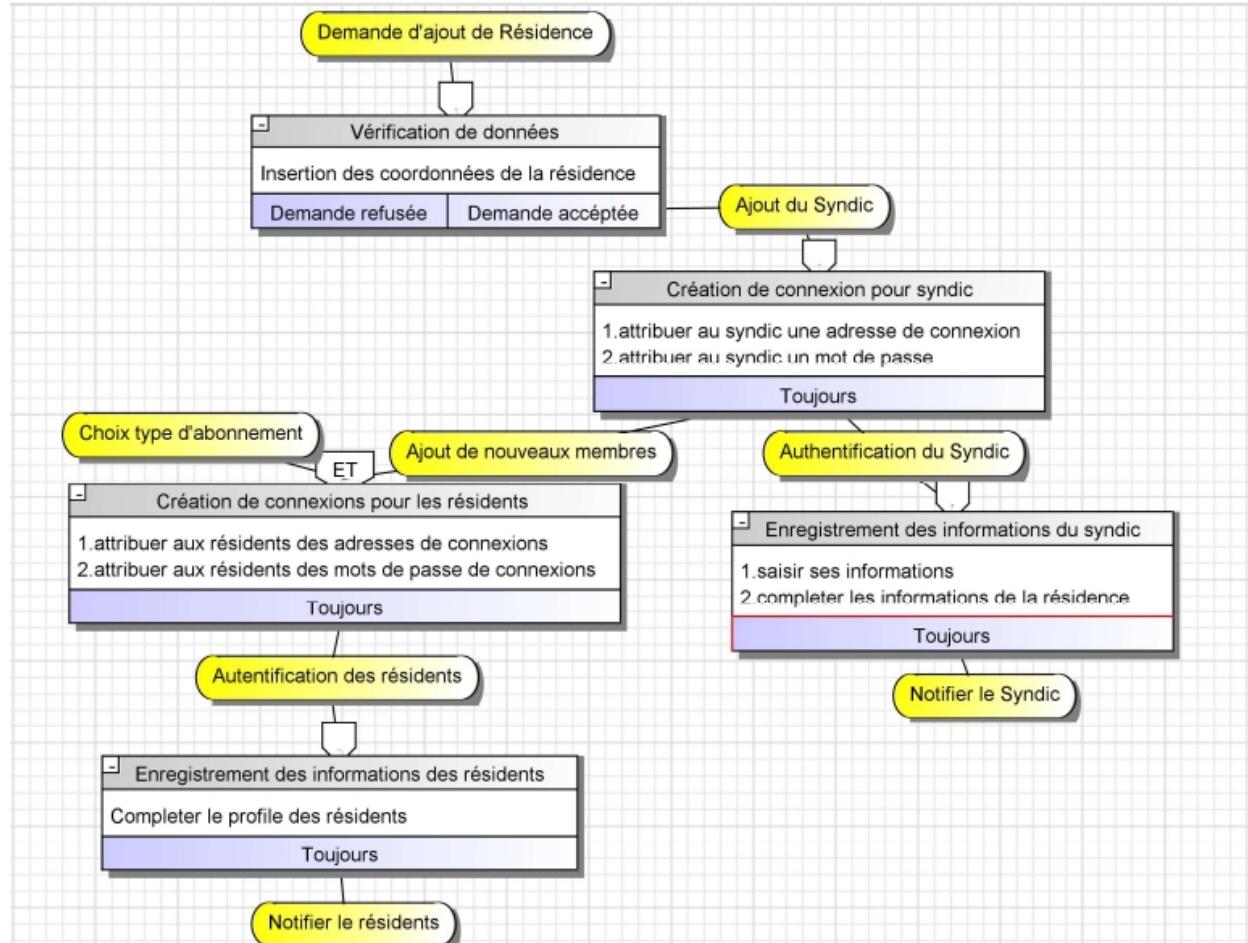


FIGURE 3.3 – MCT : Déclaration d'un nouvelle résidence

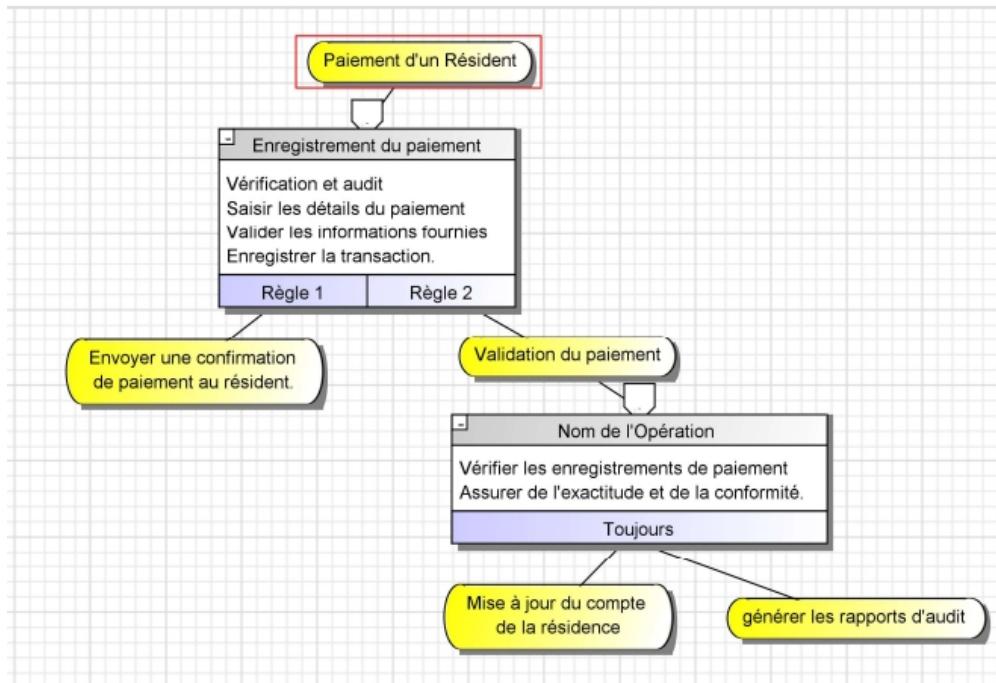


FIGURE 3.4 – MCT : Déclaration du paiement d'un résident

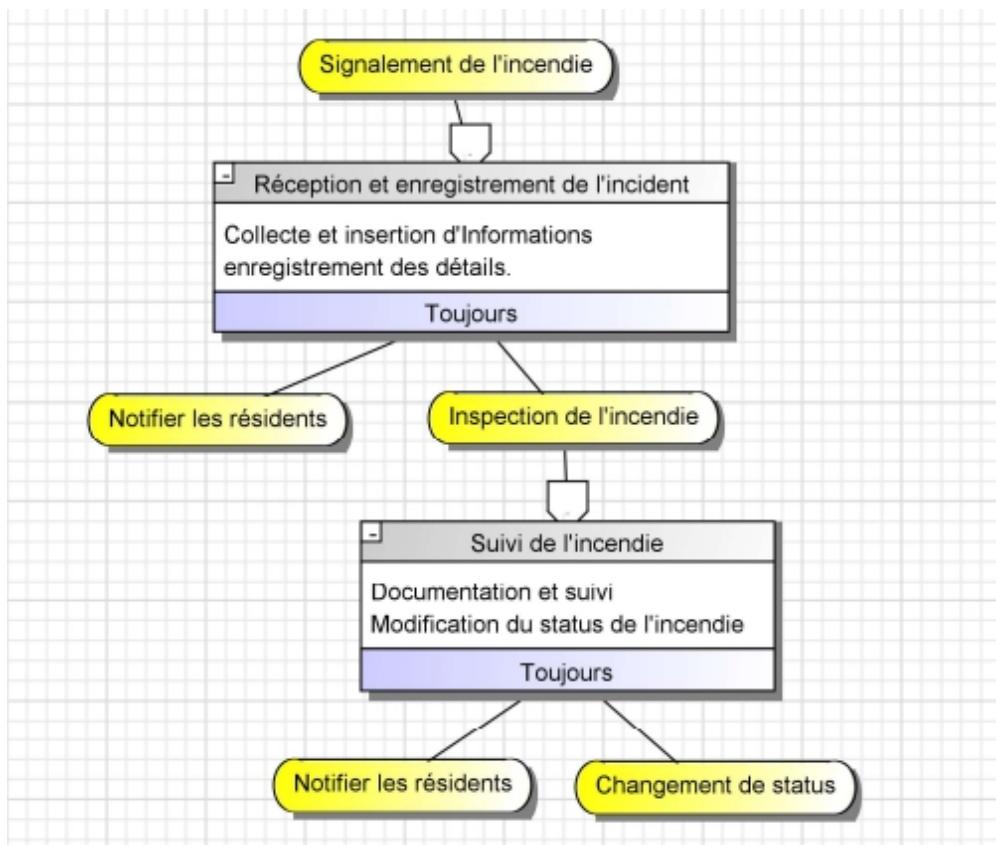


FIGURE 3.5 – MCT : Signalement d'un incendie

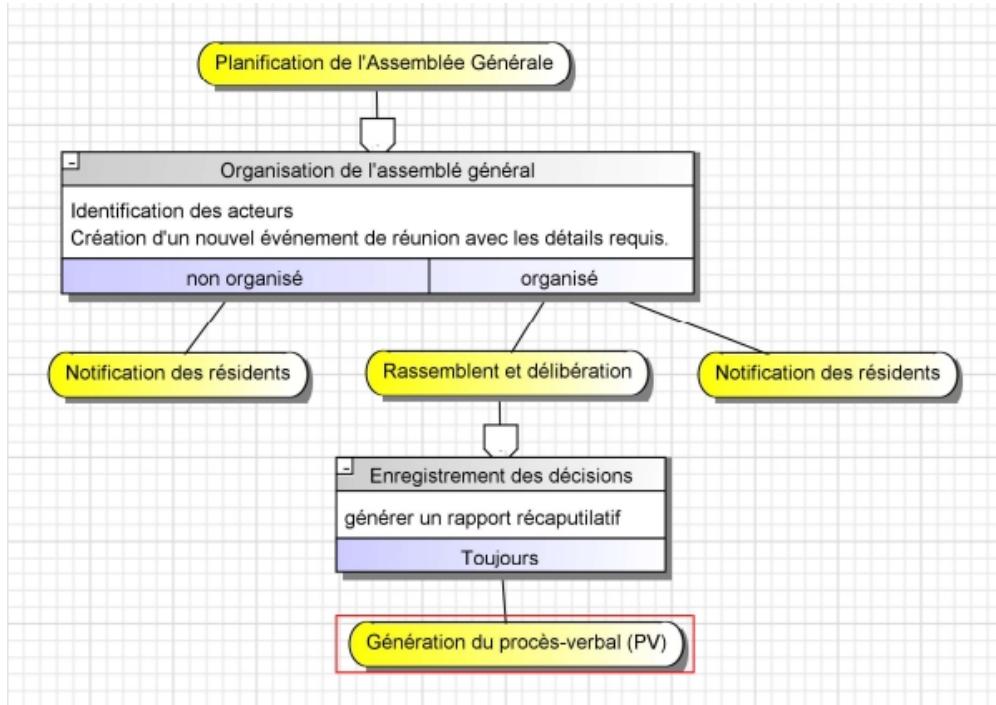


FIGURE 3.6 – MCT :Plannification d'un assemblé général

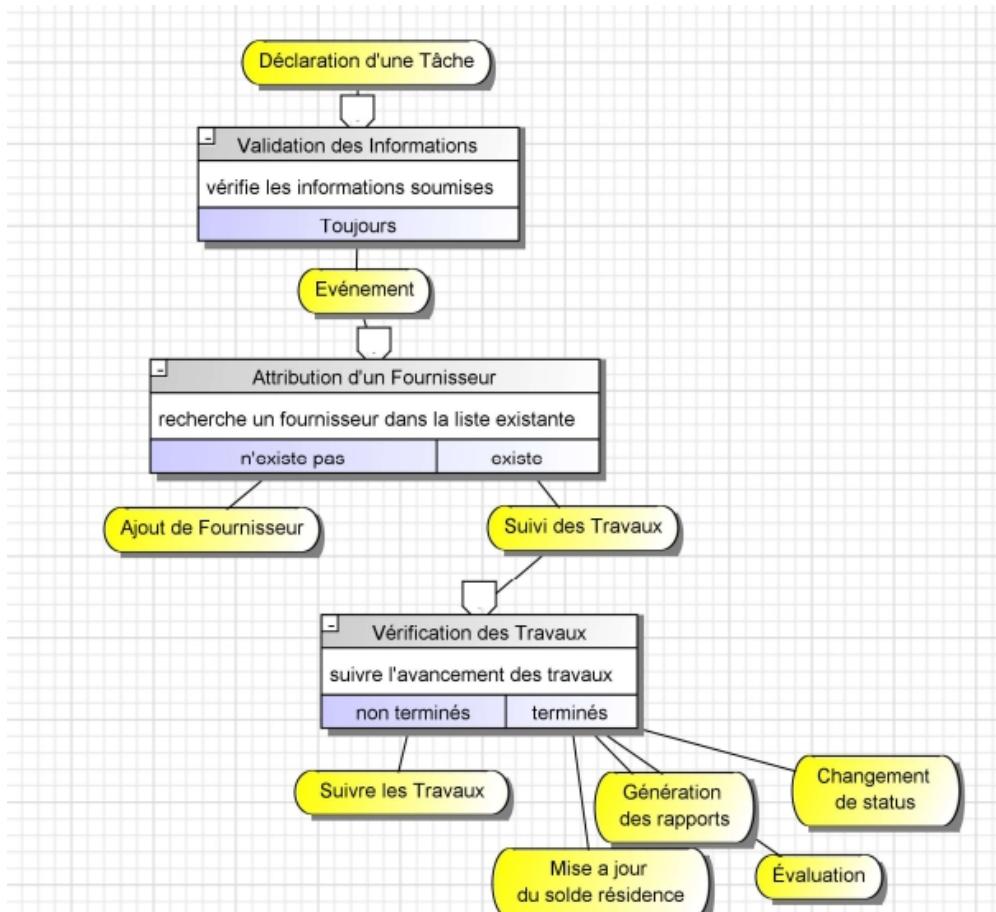


FIGURE 3.7 – MCT :Déclaration d'une tâche

3.6 Patterns de Conception

Les patterns de conception sont des solutions réutilisables et éprouvées pour des problèmes courants rencontrés dans le développement de logiciels. Ils permettent d'améliorer la maintenabilité, la flexibilité et la robustesse du code.

Voici les patterns que nous avons appliqués dans notre plateforme de gestion de syndic de copropriété :

Pattern Singleton :

Le pattern Singleton garantit qu'une classe n'a qu'une seule instance et fournit un point d'accès global à cette instance. Cela est particulièrement utile pour gérer des ressources partagées, telles que les connexions à une base de données.

Dans la classe Syndic_con, le constructeur est privé pour empêcher la création de nouvelles instances. La méthode statique getConnection crée une seule instance de connexion à la base de données si elle n'existe pas déjà, et la retourne.

Pattern Factory :

Le pattern Factory permet de créer des objets sans avoir à spécifier la classe exacte de l'objet qui sera créé. Il utilise des méthodes pour créer des objets à partir de différents paramètres.

Les classes utilisent plusieurs constructeurs pour créer des objets utilisateur avec différents ensembles de paramètres, ce qui simplifie la création d'instances des classes avec les données disponibles.

Pattern Facade :

Le pattern Facade fournit une interface simplifiée à un ensemble de classes ou de sous-systèmes complexes. Dans ce contexte, les classes DAO (interfacesDAO et classesDAOImpl) agissent comme des façades pour les opérations sur la base de données, simplifiant ainsi l'accès aux données pour les autres parties de l'application.

Conclusion

Le chapitre de conception a détaillé les étapes cruciales pour la structuration et l'organisation des informations dans le développement de la plateforme de gestion de syndic de copropriété, en utilisant la méthodologie Merise comme guide. Nous avons élaboré un dictionnaire de données exhaustif, identifié les dépendances fonctionnelles essentielles pour garantir l'intégrité et la cohérence des données, et développé des modèles conceptuels et logiques de données (MCD et MLD) pour représenter graphiquement les entités, leurs attributs et leurs relations. De plus, le modèle conceptuel de traitement (MCT) a été conçu pour décrire les processus et les fonctionnalités du système. Enfin, l'application de patterns de conception comme Singleton, Factory et Facade a permis d'améliorer la maintenabilité, la flexibilité et la robustesse du code. Ces choix de conception, justifiés par les exigences identifiées et les meilleures pratiques, posent les bases solides pour une application efficace, cohérente et facile à utiliser.

Chapitre 4

Réalisation et mise en oeuvre

Ce chapitre décrit les étapes de la réalisation et de la mise en œuvre de la plateforme de gestion de syndic de copropriété, en mettant en avant les technologies et les outils utilisés pour son développement. Il offre un aperçu détaillé des choix technologiques effectués, justifiant chaque décision en fonction des besoins fonctionnels et des exigences de sécurité de notre plateforme.

4.1 Environnement de développement

4.1.1 Espace de Travail : IntelliJ IDEA

Pour le développement de notre plateforme, nous avons choisi IntelliJ IDEA comme environnement de développement intégré (IDE). Les avantages d'IntelliJ IDEA incluent :

- **Complétion de code intelligente** : Facilite l'écriture de code rapide et précis.
- **Refactorisation** : Outils puissants pour refactoriser le code en toute sécurité.
- **Intégration avec les systèmes de versionnement** : Compatibilité avec Git et autres systèmes de gestion de versions.
- **Support étendu des frameworks** : Support natif pour de nombreux frameworks utilisés dans le projet.

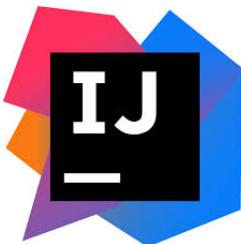


FIGURE 4.1 – intelij

4.1.2 Collaboration : GitHub

Nous avons utilisé GitHub comme plateforme de collaboration pour notre projet. Il nous a permis de partager le code source, de gérer les versions et de collaborer efficacement dans l'équipe. Les fonctionnalités telles que les requêtes de tirage, les problèmes ont facilité la communication et la coordination des efforts de développement.



FIGURE 4.2 – github

4.1.3 Base de Données : Mysql

La base de données est le cœur de toute application de gestion. Pour notre plateforme, nous avons choisi MySQL, une base de données relationnelle robuste et performante, qui offre les avantages suivants :

- **Fiabilité et robustesse** : MySQL est reconnu pour sa stabilité et sa capacité à gérer de grandes quantités de données.
- **Compatibilité** : Compatible avec de nombreux systèmes d'exploitation et langages de programmation.
- **Facilité d'utilisation** : Une syntaxe SQL bien connue et une grande communauté d'utilisateurs.



FIGURE 4.3 – MySQL

4.1.4 Backend

Le backend de notre plateforme est développé en utilisant Java et les technologies associées pour garantir la performance et la scalabilité :

- **Java** : Utilisé comme langage de programmation principal pour le développement du backend.
- **Maven [4]** : Utilisé pour la gestion des dépendances et la construction du projet.
- **Servlets** : Pour traiter les requêtes et les réponses HTTP.
- **Tomcat** : Serveur web Apache Tomcat pour déployer et exécuter les applications Java.
- **JavaMail API [11]** : Bibliothèque pour la gestion et l'envoi d'emails.
- **Spring JDBC** : Framework pour l'intégration avec la base de données, facilitant les opérations CRUD.

- **MySQL Connector** : Bibliothèque pour la connexion et les interactions avec la base de données MySQL.
- **iText [9]** : Bibliothèque pour la génération de documents PDF.
- **Gson** : Bibliothèque pour la manipulation et la conversion des données JSON.

4.1.5 Sécurité : jBCrypt

La sécurité des données des utilisateurs est primordiale. Pour garantir la protection des mots de passe, nous utilisons jBCrypt, une bibliothèque Java de hachage sécurisé :

- **Hachage sécurisé** : Utilisation d'algorithmes de hachage robuste pour stocker les mots de passe de manière sécurisée.
- **Salage des mots de passe** : Ajout de sel unique pour chaque mot de passe afin de protéger contre les attaques par dictionnaire et par force brute.

4.1.6 Tests unitaires et les tests d'intégration [10]

Pour assurer la qualité et la fiabilité du code, nous avons mis en place des tests unitaires et des tests d'intégration en utilisant **JUnit** et **Mockito** :

- **Tests unitaires** : Création de tests pour vérifier le bon fonctionnement des différentes parties du code.
- **Tests d'intégration** : Vérification de l'interaction entre les différents modules de l'application.
- **Intégration continue** : Exécution régulière des tests pour détecter et corriger rapidement les erreurs.



FIGURE 4.4 – JUnit et Mockito

4.1.7 Frontend

Dans le développement de notre application, nous avons utilisé une combinaison de technologies front-end pour assurer une expérience utilisateur optimale. Voici les principales technologies que nous avons utilisées :

HTML et CSS

Nous avons utilisé HTML et CSS pour la structure et le style de nos pages web. HTML5 fournit une structure claire et sémantique pour notre contenu, tandis que CSS3 nous a permis de styliser notre application avec des mises en page modernes et des animations.



FIGURE 4.5 – HTML et CSS

JavaScript

JavaScript est au cœur de notre application web. Nous l'avons utilisé pour rendre nos pages interactives, gérer les événements utilisateur et effectuer des requêtes asynchrones vers notre serveur.



FIGURE 4.6 – JavaScript

Tailwind CSS

Nous avons adopté Tailwind CSS comme framework CSS pour simplifier le développement et la maintenance de notre interface utilisateur. Tailwind CSS offre une approche basée sur les classes pour styliser les composants, ce qui nous a permis de personnaliser facilement l'apparence de notre application.



FIGURE 4.7 – Tailwind CSS

Chart.js [6]

Pour la visualisation des données et la création de graphiques, nous avons intégré Chart.js. Cette bibliothèque JavaScript nous a permis de générer facilement des graphiques interactifs pour présenter les informations de manière claire et attrayante.



Chart.js

FIGURE 4.8 – Chart.js

Moment.js

Moment.js a été utilisé pour manipuler, valider et formater les dates dans notre application. Cette bibliothèque JavaScript est particulièrement utile pour les opérations de gestion du temps, ce qui était essentiel pour notre fonctionnalité de gestion des tâches avec des dates d'échéance.



FIGURE 4.9 – Moment.js

4.2 Résultats



FIGURE 4.10 – Résultat

4.2.1 Structure de l'application

Voici la structure de fichiers de notre application comme représenté dans la figure 4.11 :

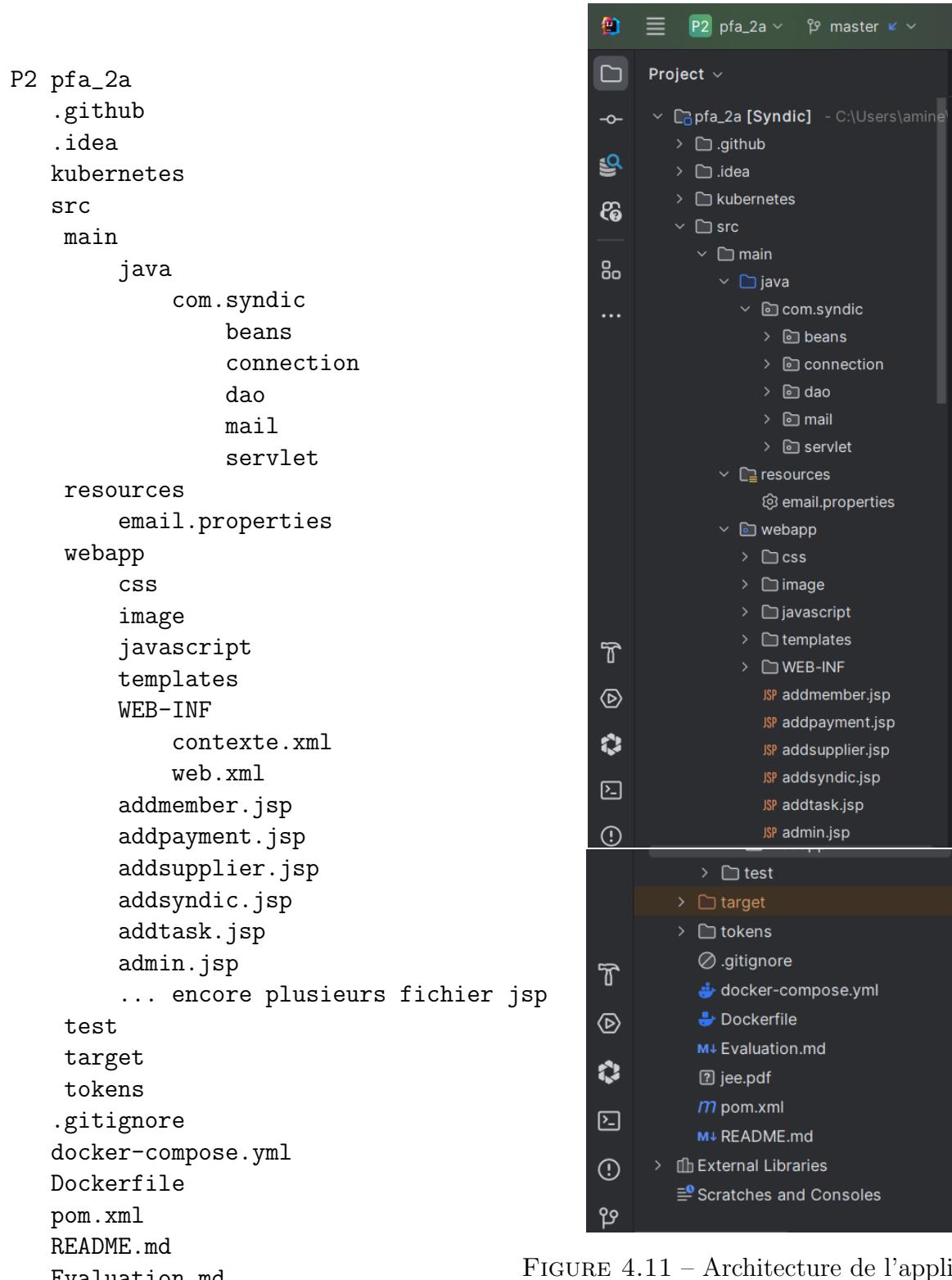


FIGURE 4.11 – Architecture de l'application

D'après la structure de notre application, on peut constater que cette dernière se compose principalement de :

- **src/main/java/com.syndic** : Ce répertoire contient le code source Java de l'application, organisé en différents packages :
 - **beans** : Contient les classes qui définissent les objets métier de l'application, comme les entités de base de données ou les objets de transfert de données.
 - **connection** : Contient les classes et les configurations liées à la connexion à la base de données.
 - **dao** : Contient les classes d'accès aux données, responsables de la manipulation des données dans la base de données.
 - **mail** : Contient les classes et les configurations pour l'envoi d'e-mails à partir de l'application.
 - **servlet** : Contient les servlets Java qui gèrent les requêtes HTTP entrantes et contrôlent le flux de l'application.
- **src/main/resources** : Ce répertoire contient les ressources non Java de l'application, telles que les fichiers de configuration, les fichiers de propriétés, etc.
- **webapp** : Ce répertoire contient les ressources Web de l'application, telles que les fichiers HTML, CSS, JavaScript, et les modèles de vues.
- **WEB-INF** : Ce répertoire contient les fichiers de configuration spécifiques à la servlet et d'autres ressources privées de l'application, tels que les fichiers JSP, ainsi que les fichiers de configuration comme `context.xml` et `web.xml`.
- **kubernetes** : Contient les fichiers de configuration Kubernetes pour le déploiement de l'application sur un cluster Kubernetes.
- **docker-compose.yml** : Fichier de configuration Docker Compose pour l'orchestration des conteneurs de l'application.
- **pom.xml** : Fichier de configuration Maven contenant les dépendances et les paramètres de construction du projet.
- **Dockerfile** : Fichier de configuration pour la création de l'image Docker de l'application.

4.3 Présentation des interfaces

La première interface de notre application de gestion de syndic est la landing page illustrée dans la figure 4.12.

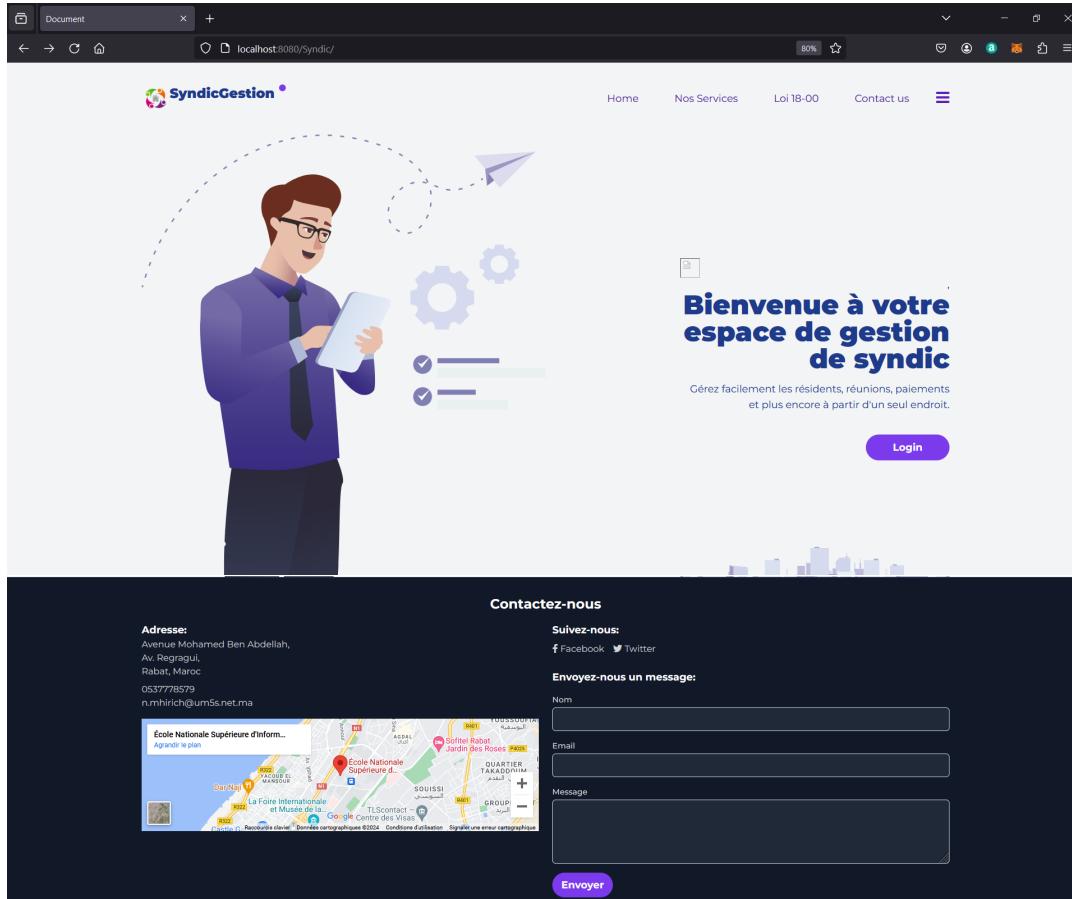


FIGURE 4.12 – Landing Page

En cliquant sur le boutton Login, on se redirige vers Login page (figure 4.13).

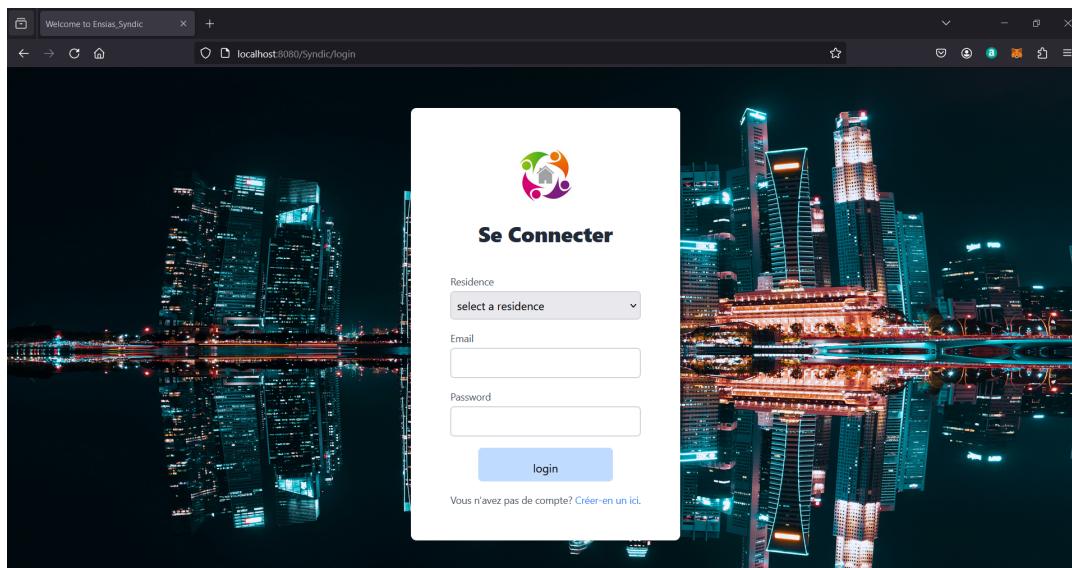


FIGURE 4.13 – Login page

4.3.1 Admin :

Après la connexion en tant que Admin, voilà les interfaces de ce dernier :

Dashboard

La figure 4.14 représente le dashboard de l'admin avec les informations principales et des graphes résumant la situation actuel.

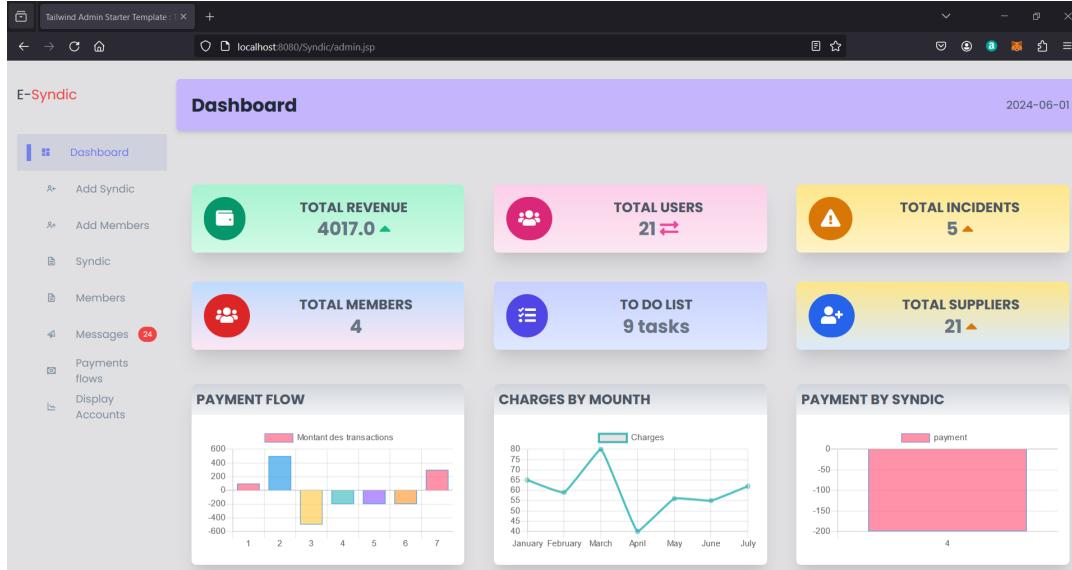


FIGURE 4.14 – Dashboard - Admin

Add syndic

La figure 4.15 représente l'interface où l'admin peut ajouter un nouveau syndic en lui associant un mail et un mot de passe pour se connecter à la plateforme, ainsi que la spécification du type d'abonnement.

The screenshot shows the 'Add New Syndic' page. The sidebar includes 'Dashboard', 'Add Syndic', 'Add Members', 'Syndic', 'Members', 'Messages (24)', 'Payments flows', and 'Display Accounts'. The main form has a dark background with white text. It includes fields for 'USER NAME' (Syndic2), 'RESIDENCE' (Jasmine), 'RESIDENCE TYPE' (select a residence), 'EMAIL ADDRESS' (redacted), 'PASSWORD' (redacted), and buttons 'New Syndic to Add' and 'Confirm Add'. To the right is a sidebar titled 'Analyse des Syndics' with sections for 'NOUVEAUX SYNDICS ENRÉGISTRÉS' (234), 'SYNDICS ACTIFS' (1100), and 'NOUVELLES DEMANDES D'ADHÉSION' (32). A button 'Ajouter un Nouveau Syndic' is also present.

FIGURE 4.15 – Add Syndic - Admin

Add member

La figure 4.16 représente l'interface où l'admin peut ajouter un nouveau member en lui associant un mail et un mot de passe pour se connecter, en spécifiant la résidence à laquelle

appartient ce nouveau membre.

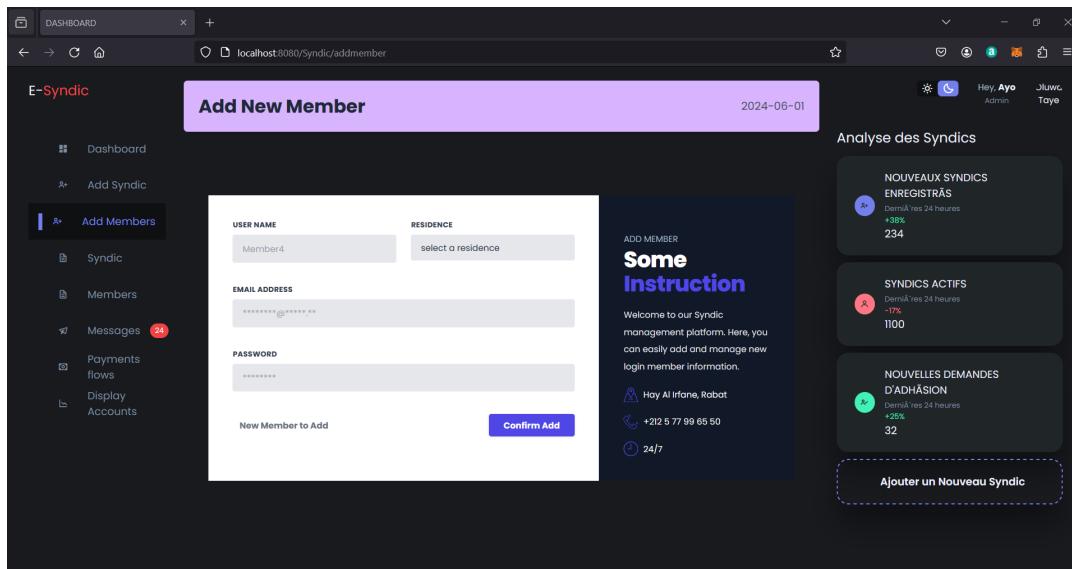


FIGURE 4.16 – Add Member - Admin

Liste des syndics

Afficher toutes les informations de tous les syndics et des résidences dont ils sont responsables.(figure 4.17).

FIGURE 4.17 – Liste des syndics - Admin

Liste des membres

Registered Members (2024-06-01)

Members List

ID	PRÉNOM	NOM	ADRESSE	TÉLÉPHONE	EMAIL	MEMBER_S_ID
2	Asmae	Karmouchi	Hay Nahda Rabat	06666666	asmae@gmail.com	4
3	Ahmed	Matjjour	Al Irfane Rabat	07777777	ahmed@gmail.com	4
4	chaimae	null	null	null	null	5
5	doha	null	null	null	test@gmail.com	6

Analyse des Syndics

- NOUVEAUX member ENREGISTRÉS**: Dernières 24 heures +10% 234
- members ACTIFS**: Dernières 24 heures -17% 1100
- NOUVELLES DEMANDES D'ADHÉSION**: Dernières 24 heures +20% 32

Ajouter un Nouveau member

FIGURE 4.18 – Liste des membres - Admin

Le flux de paiement de tout les syndics

La figure 4.19 représente les paiements associés à tous les syndics, avec la possibilité de filtrer par syndic et par date.

Administration of Expenses, Tasks, and Payments (2024-06-01)

Payments Flows

SYNDICID	FLOW TYPE	AMOUNT	DESCRIPTION	DATE TRANSACTION
4	0	500.0	Payment added with code: 3tpe: oal23By Member2	2024-06-08
4	1	-200.0	Task added with name: pool task2	2024-06-02
4	1	-200.0	Task added with name: pool task2	2024-06-02
4	1	-200.0	Task added with name: pool task3	2024-06-01

Analyse des Syndics

- NOUVEAUX SYNDICS ENREGISTRÉS**: Dernières 24 heures +38% 234
- SYNDICS ACTIFS**: Dernières 24 heures +10% 1100
- NOUVELLES DEMANDES D'ADHÉSION**: Dernières 24 heures +25% 32

Ajouter un Nouveau Syndic

FIGURE 4.19 – Payment Flow - Admin

4.3.2 Syndic :

Après la connexion en tant que Syndic, voici les interfaces disponibles :

Dashboard Syndic

La figure 4.20 représente le tableau de bord du syndic, avec plusieurs indicateurs clés de performance (KPI) et divers graphiques pour aider le syndic à avoir une vue d'ensemble de la situation actuelle et à prendre des décisions éclairées.

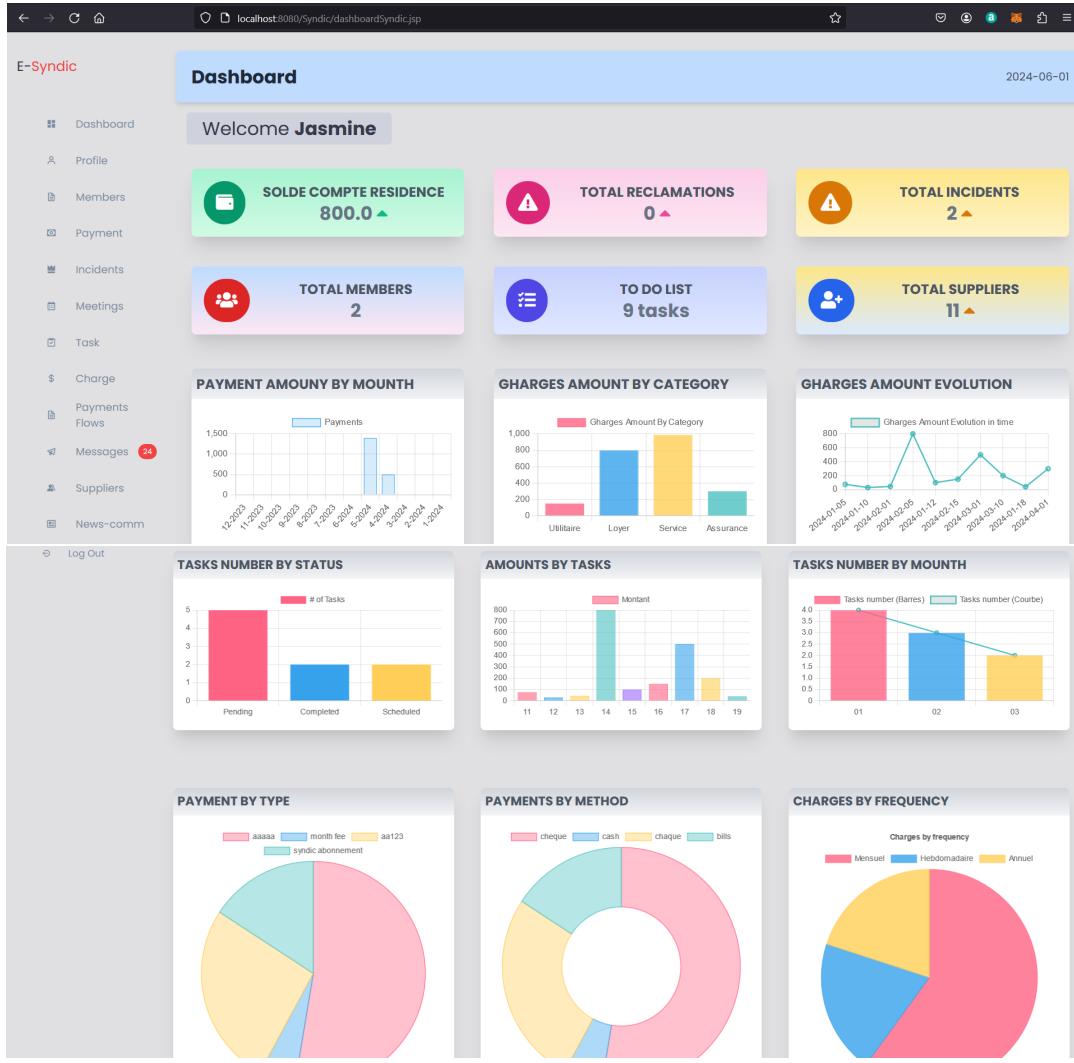


FIGURE 4.20 – Dashboard Syndic

Profil Syndic

Cette Interface représente les données du syndic et de la résidence ajoutées par le syndic lui-même et qu'il peut les modifier (figure 4.21).

FIGURE 4.21 – Profil Syndic

Liste des membres

Cette interface affiche la liste des membres en visualisent les paiements de chaqu'un d'eux comme illustré dans la figure 4.22. Les mois payés sont indiqués en vert, tandis que les mois non payés sont indiqués en rouge.

Pour plus d'infomation sur le membre on clique sur "More" (figure 4.23).

Pour alerter un membre de l'obligation d'effectuer ses paiements, il suffit de cliquer sur le bouton "Alerte" pour envoyer automatiquement un mail à ce dernier (figure 4.24).

The screenshot shows the 'Members of Jasmine Residence' section. It displays two members: Asmae Karmouchi and Ahmed Mahjour. Each member has a status indicator for each of 12 residents (M-1 to M-12). Asmae's status is: M-1 (red), M-2 (red), M-3 (red), M-4 (green), M-5 (green), M-6 (red), M-7 (red), M-8 (red), M-9 (red), M-10 (red), M-11 (red), M-12 (red). Ahmed's status is: M-1 (red), M-2 (red), M-3 (red), M-4 (red), M-5 (red), M-6 (red), M-7 (red), M-8 (red), M-9 (red), M-10 (red), M-11 (red), M-12 (red). Below each member's status is a red button labeled 'Alerte [Name] to pay'. To the right, there is a 'Syndic Analysis' sidebar with three cards: 'Monitor Payment Flow' (Follow-up +10%, 234), 'Residence Account Balance' (800.0 -17%), and 'Number of Residents in Jasmine' (2).

FIGURE 4.22 – Liste des membres - Syndic

This screenshot shows the detailed information for Asmae Karmouchi. It includes her full address (107 Nahda Rabat, 14000, Property Code: 122), email (asmae@gmail.com), property type (villa), size (250 sqm), and co-ownership fee (500 dh). Below this information is a status bar for all 12 residents, identical to Figure 4.22. A red button at the bottom left says 'Alerte Asmae Karmouchi to pay'. The right side features the same 'Syndic Analysis' sidebar as in Figure 4.22.

FIGURE 4.23 – Info membre - Syndic

The screenshot shows an email in the Gmail inbox titled 'Reminder to Pay' from 'ensiassyndic@gmail.com' to 'A moi'. The email body reads:

Dear asmae karmouchi,
We are reminding you to pay your dues.
Best regards,
The Syndic Team

The email is dated 'mer. 29 mai 14:05 (il y a 3 jours)'. The footer of the email says '© 2024 Ensias_Syndic. All rights reserved.'

FIGURE 4.24 – Mail Alerte de paiement

Ajouter un paiement

Cette interface affiche les paiements effectués par les membres de la résidence (4.25), avec plusieurs graphiques d'analyse. Pour ajouter un paiement, il suffit de cliquer sur le bouton "add payment" (figure 4.26). Après avoir effectué un paiement, le membre concerné reçoit un mail de confirmation (figure 4.27). Il est également possible de visualiser et d'imprimer les paiements, filtrés par membre et par date (figure 4.28).

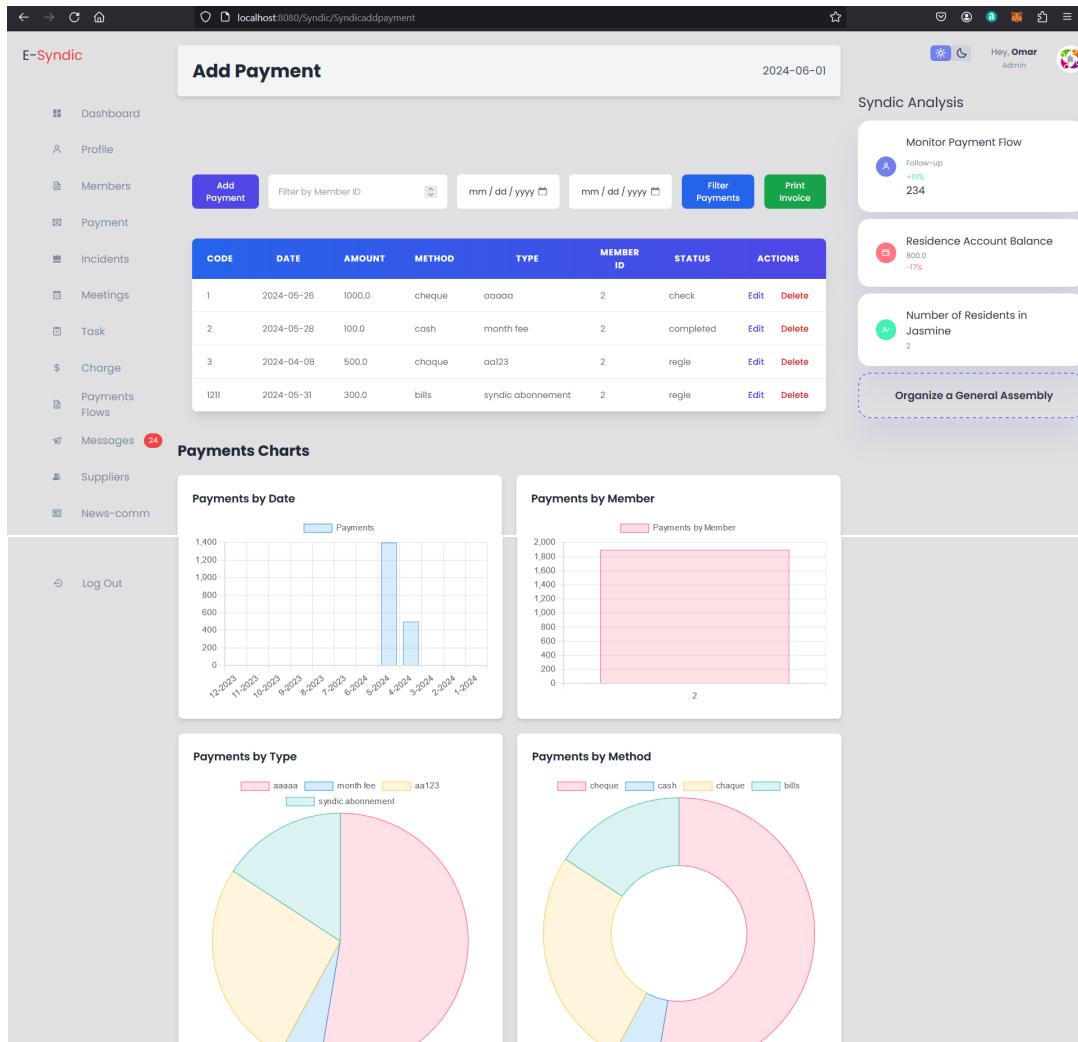


FIGURE 4.25 – Page des paiement - syndic

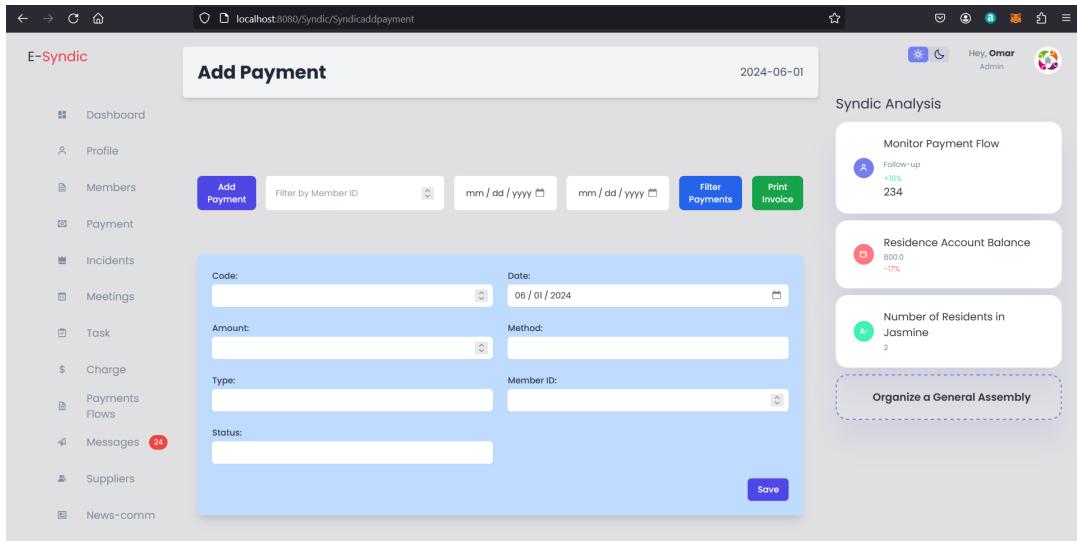


FIGURE 4.26 – Add payment - syndic

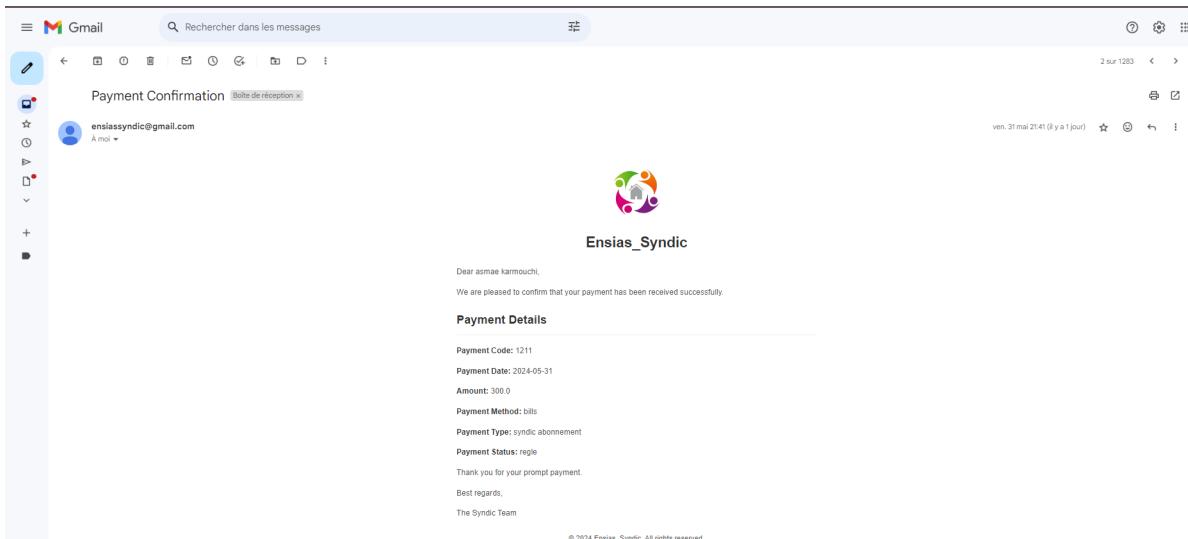


FIGURE 4.27 – Mail de confirmation de paiement

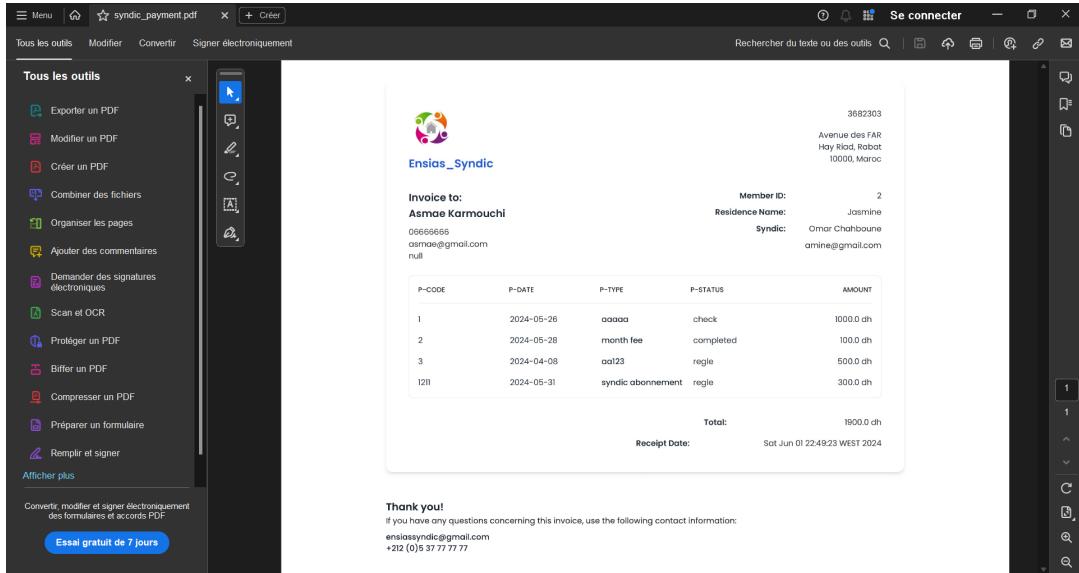


FIGURE 4.28 – PDF - Bulletin de paiement

Gestion des incidents

Dans cette interface, on peut ajouter un nouvel incident (figure 4.29), où tous les membres de la résidence seront notifiés par mail de l'incident (figure 4.30).

Il est également possible de visualiser tous les incidents et leurs détails, ainsi que de modifier le statut de chaque incident.

FIGURE 4.29 – Interface Incendie - syndic

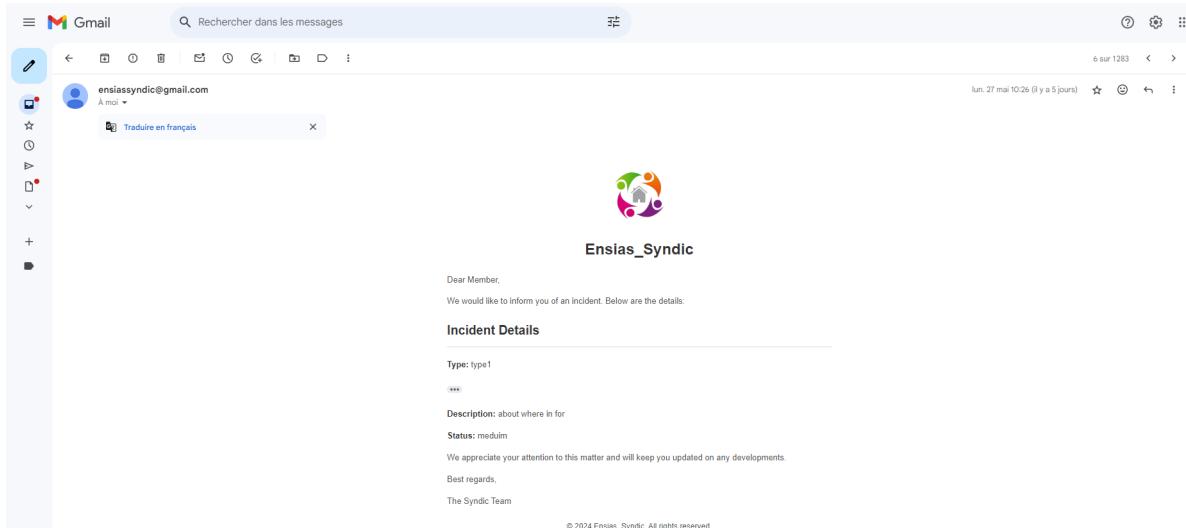


FIGURE 4.30 – Mail Notif Incendie

Gestion des réunions

Dans cette interface, on peut ajouter une nouvelle réunion (figure 4.31), accompagnée de l'envoi d'un mail d'invitation à tous les membres de la résidence (figure 4.32). Sans oublier la possibilité de visualiser les détails de toutes les réunions déjà passées, de les modifier et d'ajouter des conclusions à la réunion, ainsi que la possibilité d'imprimer un procès-verbal de la réunion (figure 4.33).

The screenshot displays the E-Syndic web application interface, specifically the 'Meeting' section. The top navigation bar shows the URL 'localhost:8080/Syndic/meeting', a battery icon at 67%, and a user profile for 'Hey Omar Syndic'. The main content area is titled 'Meeting' and shows the date '2024-06-01'. On the left, a sidebar menu includes 'Dashboard', 'Profile', 'Members', 'Payment', 'Incidents', 'Meetings' (selected), 'Task', 'Charge', 'Payments Flows', 'Messages' (with 24 notifications), 'Suppliers', and 'News-comm'. The central panel features a red header 'Meeting Regulation' and a sub-section 'Law No. 18-00 concerning the status of condominiums in built-up areas.' It also contains a 'Rules of the General Meeting of Condominium Owners' section with several bullet points. A large blue button labeled 'Add Meeting' is prominent. To the right, a 'Syndic Analysis' sidebar provides monitoring insights: 'Monitor Payment Flow' (Follow-up +10%, 234), 'Residence Account Balance' (800.00 -17%), and 'Number of Residents in Jasmine' (2). A dashed blue box labeled 'Organize a General Assembly' is also present. Below the main content is a modal window for sending files via email, showing fields for recipient ('example@domain.com'), subject ('Send files to this email:'), and file attachments ('PV_meeting5.pdf' and 'PV_meeting6.pdf'). A large blue 'Send File' button is at the bottom. At the bottom of the page is a 'List of Meetings' section with two entries: 'Réunion mensuelle' and 'Assemblée générale', each with 'Edit' and 'Print' buttons.

FIGURE 4.31 – Interface meeting - Syndic

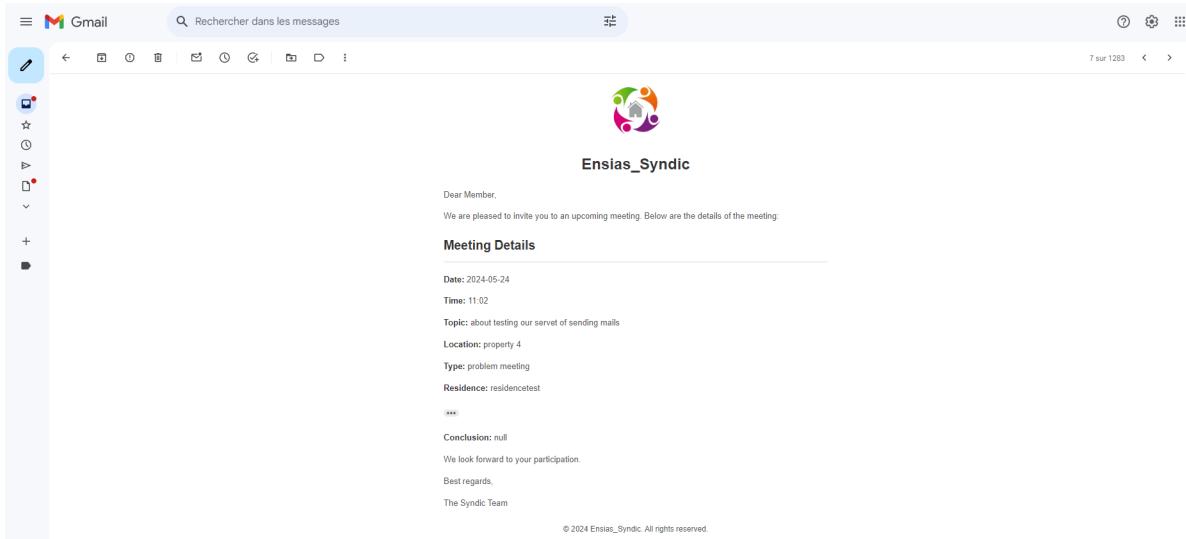


FIGURE 4.32 – Mail Invitation meet

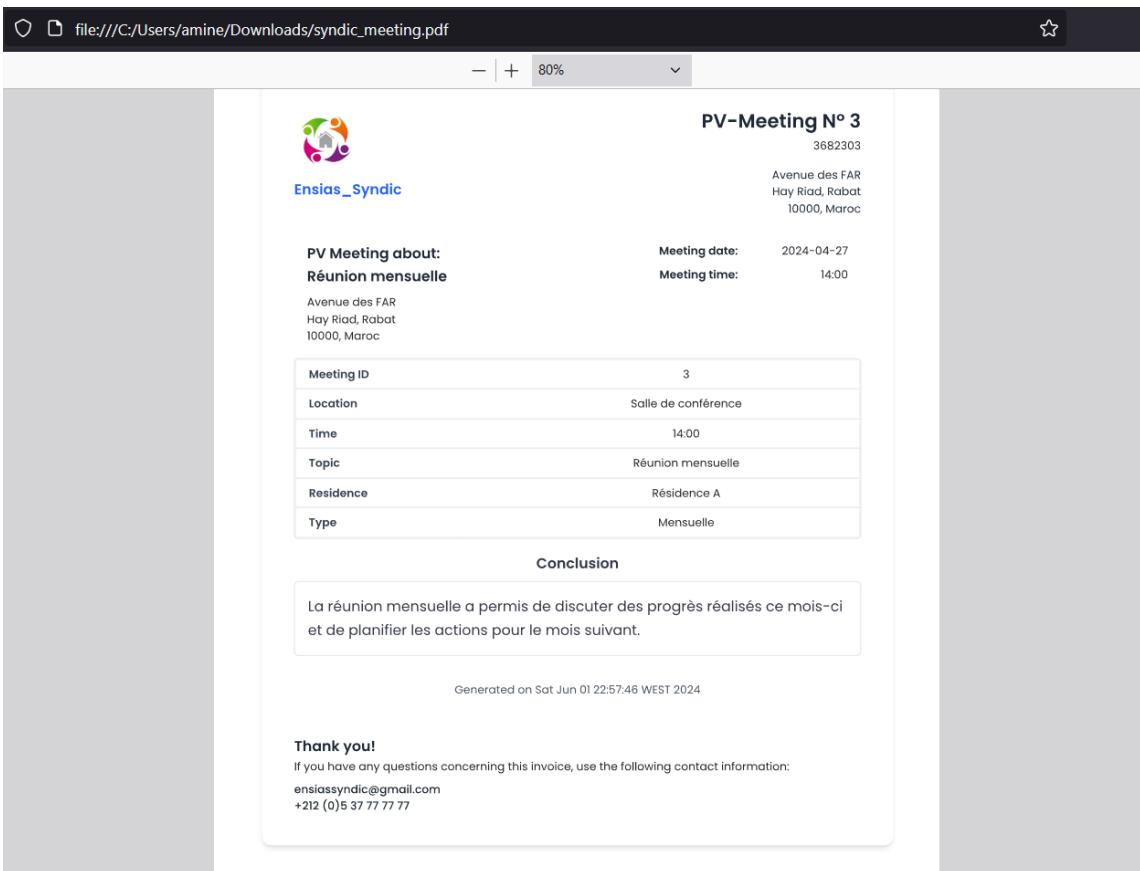


FIGURE 4.33 – PDF - PV meeting

Gestion des tâches

La figure 4.34 représente l'interface de gestion des tâches, qui affiche les informations sur les tâches dans la résidence et plusieurs graphes d'analyses, avec la possibilité d'ajouter de nouvelles tâches, ainsi que de nouveau fournisseur.

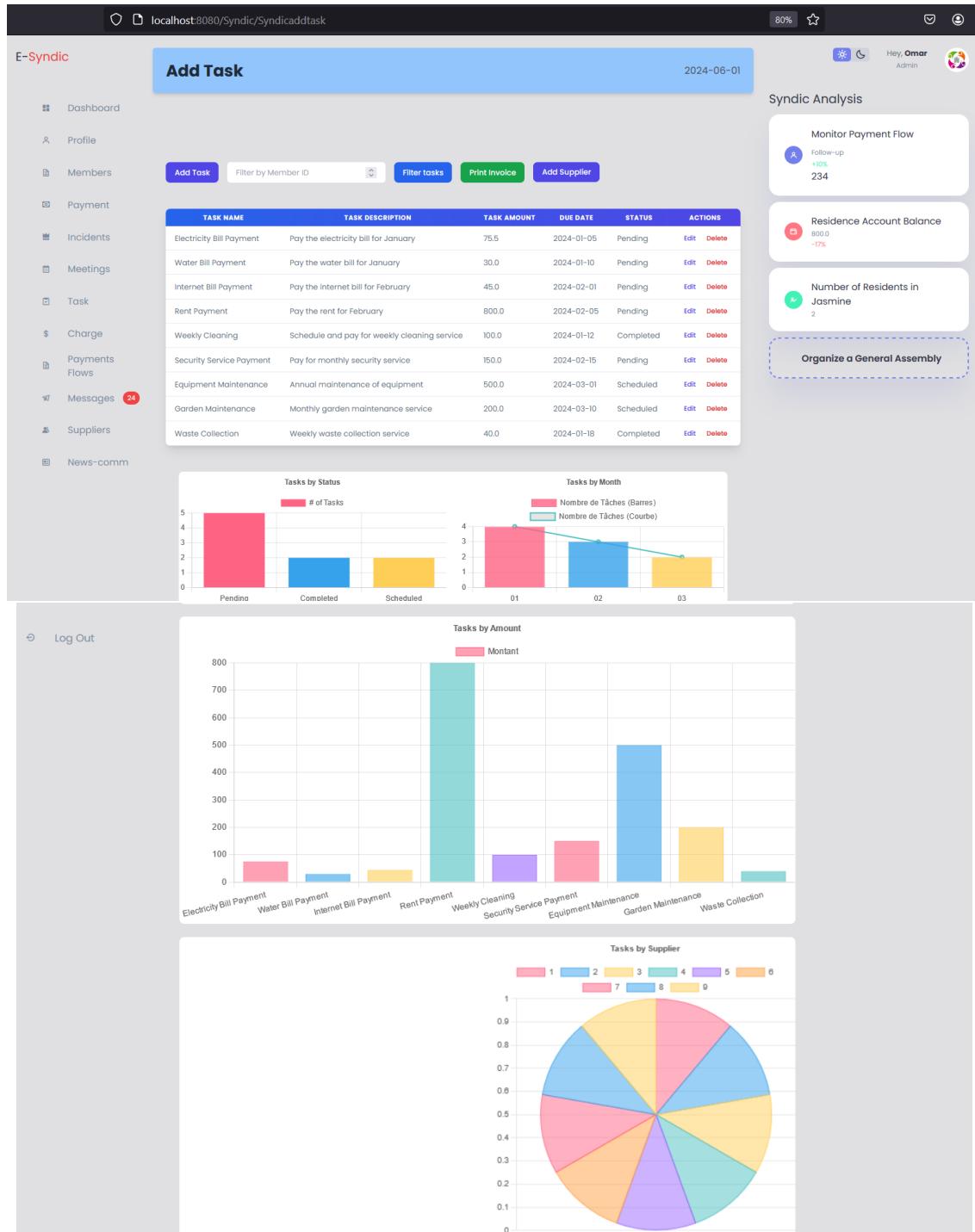


FIGURE 4.34 – Pages des tâches - Syndic

Gestion des charges

Cette interface représente les charges de la résidence avec la possibilité d'ajouter une autre charge (figure 4.35).

CHARGE NAME	CHARGE DESCRIPTION	CHARGE AMOUNT	FREQUENCY	CATEGORY	DATE	ACTIONS
Électricité	Paiement mensuel de l'électricité	75.5	Mensuel	Utilitaire	2024-01-05	Edit Delete
Eau	Facture d'eau mensuelle	30.0	Mensuel	Utilitaire	2024-01-10	Edit Delete
Internet	Abonnement internet	45.0	Mensuel	Utilitaire	2024-02-01	Edit Delete
Loyer	Loyer mensuel de la résidence	800.0	Mensuel	Loyer	2024-02-05	Edit Delete
Nettoyage	Service de nettoyage hebdomadaire	100.0	Hebdomadaire	Service	2024-01-12	Edit Delete
Sécurité	Service de sécurité mensuel	150.0	Mensuel	Service	2024-02-15	Edit Delete
Maintenance	Maintenance annuelle de l'équipement	500.0	Annuel	Service	2024-03-01	Edit Delete
Jardinage	Service de jardinage mensuel	200.0	Mensuel	Service	2024-03-10	Edit Delete
Déchets	Collecte des déchets hebdomadaire	40.0	Hebdomadaire	Service	2024-01-18	Edit Delete
Assurance	Assurance habitation annuelle	300.0	Annuel	Assurance	2024-04-01	Edit Delete

FIGURE 4.35 – Pages des charges - Syndic

Gestion de flux de paiement de la résidence

Cette interface affiche tout le flux d'argent de la résidence qui peut être filtrer par type et par date (figure 4.36).

Avec la possibilité d'impression d'un PDF (figure 4.37).

FlowType	Amount	description	Date Transaction
0	100.0	Payment added with code: 2type: month feeBy Member2	2024-05-28
0	500.0	Payment added with code: 3type: aal23By Member2	2024-06-08
1	-500.0	Task added with name: pool task	2024-05-31
1	-200.0	Task added with name: pool task2	2024-06-02
1	-200.0	Task added with name: pool task2	2024-06-02
1	-200.0	Task added with name: pool task3	2024-06-01
0	300.0	Payment added with code: l2ll type: syndic abonnement By Member: 2	2024-05-31

FIGURE 4.36 – Page de gestion de flux de paiement de la résidence

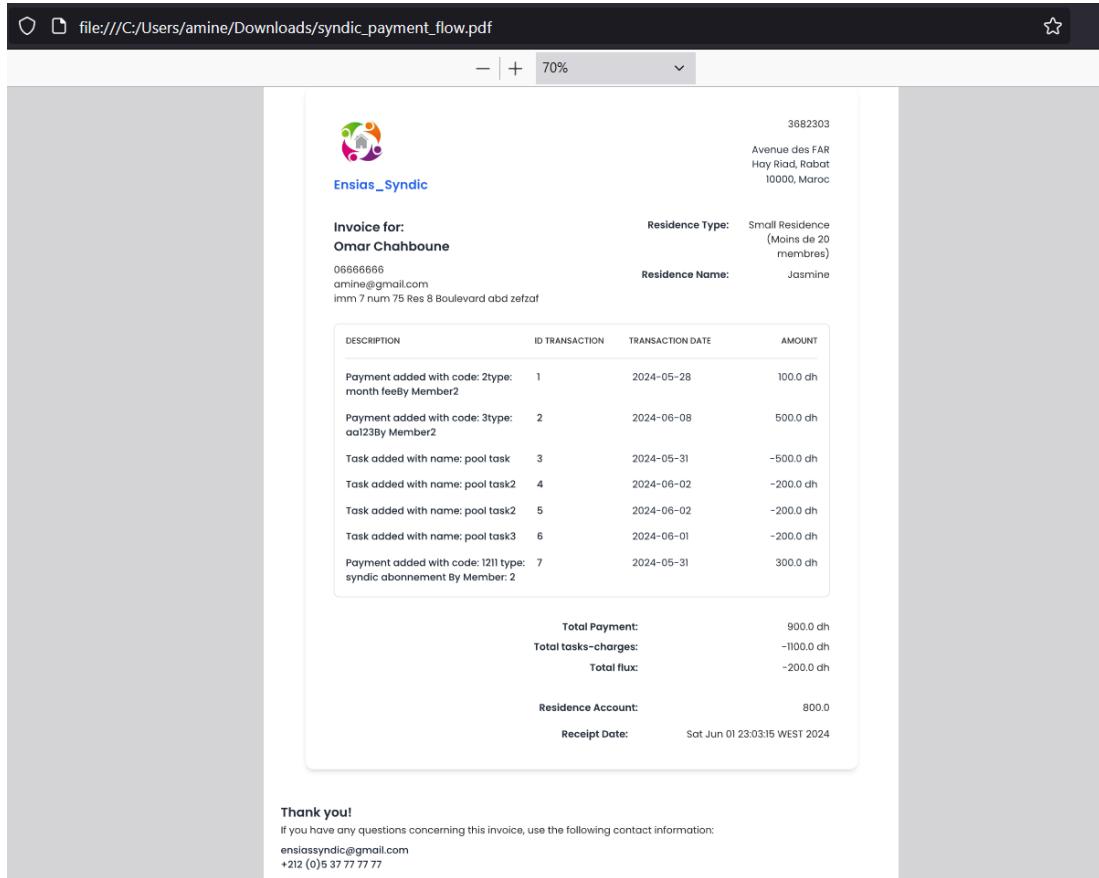


FIGURE 4.37 – PDF - flux de paiement

4.3.3 Résident :

Après la connexion en tant que Membre de la résidence, voilà les interfaces de ce dernier :

Profil Member

Cette Interface représente les données du membre et de sa propriété ajouter par le résident lui même et qu'il peut les modifier (figure 4.38).

The screenshot shows the E-Syndic member profile interface. At the top, there's a navigation bar with icons for back, forward, search, and refresh, followed by the URL "localhost:8080/Syndic/memberprofile". The title bar says "Profile of Asmae Karmouchi" and the date "2024-06-01". On the left, a sidebar menu includes "Profile", "Property", "Meetings", "My payments", "add reclamation", "Residence News", "Report Incident", and "Tasks". A "Log Out" button is also present. The main content area has a green header "Welcome Asmae!". Below it is a "Member Informations" section with fields for First Name (Asmae), Last Name (Karmouchi), Postal Code (14000), Phone Number (06666666), Full Address (Hay Nahda Rabat), and E-mail (asmoe@gmail.com). To the right, there's an "Analyse des Syndics" section with three boxes: "NOUVEAUX SYNDICS ENREGISTRÉS" (234), "SYNDICS ACTIFS" (1100), and "NOUVELLES DEMANDES D'ADHÉSION" (32). A blue dashed box labeled "Ajouter un Nouveau Syndic" is shown. Below this is a "About Property" section with fields for Property Code (122), Property Address (Hay el fath Residence Jasmine villa 2 Rabat), Property Size (250 m²), Property Type (Selectionner une option), and Co-Ownership Fee (500). A "SAVE" button is at the bottom.

FIGURE 4.38 – Profile - Member

Info Résidence Member

Cette Interface représente les données du syndic et de la résidence (figure 4.39).

The screenshot shows the E-Syndic residence information interface. At the top, there's a navigation bar with icons for back, forward, search, and refresh, followed by the URL "localhost:8080/Syndic/property". The title bar says "Property of Asmae Karmouchi" and the date "2024-06-01". On the left, a sidebar menu includes "Profile", "Property", "Meetings", "My payments", "add reclamation", "Residence News", "Report Incident", and "Tasks". A "Log Out" button is also present. The main content area has a green header "About Syndic: Omar Chahboune". Below it is an "Informations Syndic" section with fields for First Name (Omar), Last Name (Chahboune), Postal Code (13000), Phone Number (06666666), and E-mail (amine@gmail.com). To the right, there's an "Analyse des Syndics" section with three boxes: "NOUVEAUX SYNDICS ENREGISTRÉS" (234), "SYNDICS ACTIFS" (1100), and "NOUVELLES DEMANDES D'ADHÉSION" (32). A blue dashed box labeled "Ajouter un Nouveau Syndic" is shown. Below this is an "About Residence: Jasmine" section with a pink header "About Residence". It contains a grid of fields for Residence Name (Jasmine), Residence Address (Hay el fath Rabat), Residence Size (1400 m²), Number of Apartments (17), Number of Villas (4), Number of Gardens (2), Number of Pools (1), Number of Parkings (2), and Number of Elevators (4).

FIGURE 4.39 – Info Résidence Member

Incident Member

The screenshot shows the 'Incidents' section of the application. On the left, a sidebar menu includes 'Profile', 'Property', 'Meetings', 'My payments add reclamation', 'Residence News', 'Report Incident' (which is highlighted in blue), and 'Tasks'. The main content area has a green header 'Incidents' and a date '2024-06-01'. A box titled 'Incident Reporting Regulation' contains guidelines: 'Reporting', 'Documentation', 'Communication', and 'Confidentiality'. Below this is a button 'Add Incident'. To the right, there's a sidebar titled 'Analyse des Syndics' with three sections: 'NOUVEAUX SYNDICS ENREGISTRÉS' (234), 'SYNDICS ACTIFS' (1100), and 'NOUVELLES DEMANDES D'ADHÉSION' (32). A dashed box highlights a button 'Ajouter un Nouveau Syndic'.

FIGURE 4.40 – Page Incident - Member

Reclamation Member

Dans cette interface le résident peut ajouter une réclamation (figure 4.41).

The screenshot shows the 'Add Reclamation' section. The sidebar is identical to Figure 4.40. The main form has a title 'Add Reclamation' and a date '2024-06-01'. It includes fields for 'Resolution Date' (mm / dd / yyyy), 'Reclamation Type', 'Reclamation Description', and 'Reclamation Status'. Buttons for 'Add Reclamation', 'Filter by Member ID', 'Filter tasks', and 'Print Invoice' are at the top. A 'Save' button is at the bottom right. The right sidebar 'Analyse des Syndics' is also present.

FIGURE 4.41 – Page Reclamation - Member

Tâches Member

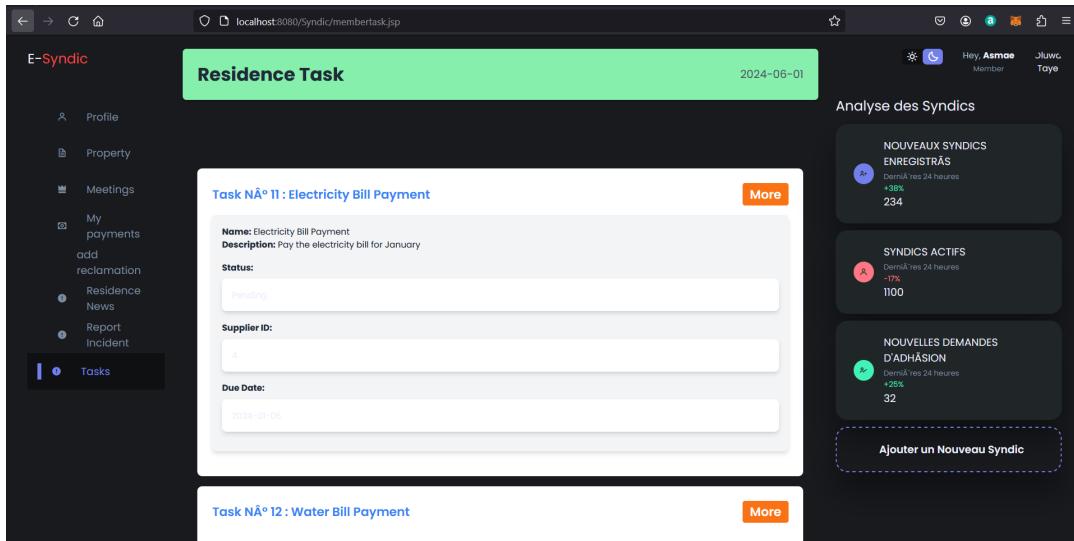


FIGURE 4.42 – Page des tâches - Member

News Member

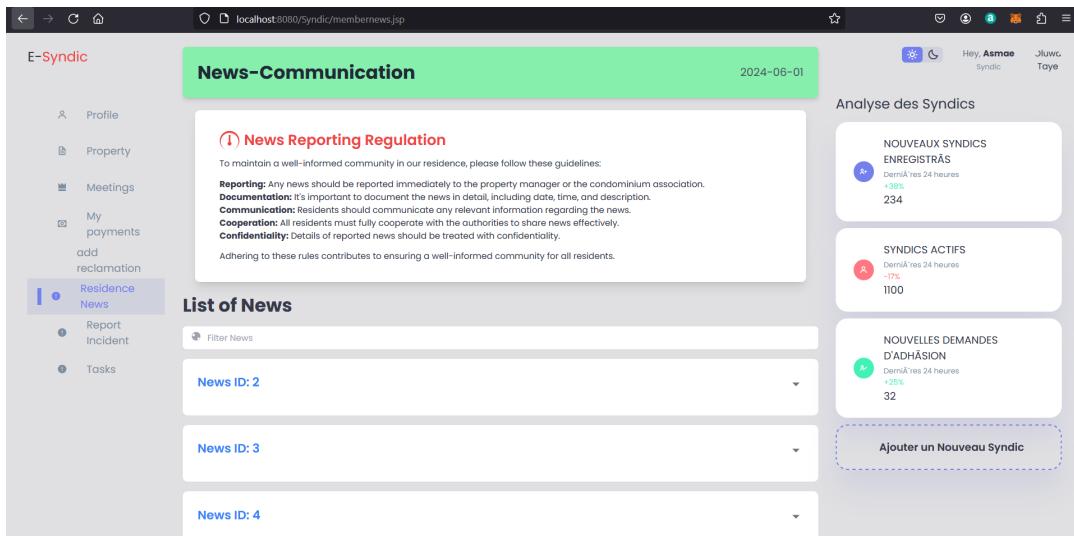


FIGURE 4.43 – Page News - Member

4.4 Déploiement de l'application

Le déploiement de l'application de gestion de syndic peut être effectué en utilisant deux méthodes principales : Docker Compose et Kubernetes. Ces deux méthodes permettent d'orchestrer les conteneurs nécessaires pour l'application, en assurant une mise en production cohérente et reproductible.

Pour déployer notre application de gestion de syndic, nous avons suivi plusieurs étapes. Tout d'abord, nous avons utilisé Maven pour construire l'application en exécutant la commande suivante :

```
mvn clean package
```

Cela a généré un fichier `.war` contenant notre application dans le répertoire `target`.

4.4.1 Déploiement avec Docker Compose

Docker Compose est utilisé pour définir et gérer des applications multi-conteneurs. Dans cette configuration, deux principaux services sont déployés : Tomcat et MySQL.

- **Service Tomcat** : Ce service utilise l'image officielle de Tomcat pour déployer l'application web. Le fichier WAR de l'application est monté directement dans le répertoire des applications web de Tomcat, ce qui permet une mise en production rapide et fiable.
- **Service MySQL** : La base de données MySQL est configurée avec des variables d'environnement pour définir le mot de passe root, le nom de la base de données, ainsi que l'utilisateur et le mot de passe de la base de données. Les données de la base de données sont persistées à l'aide de volumes Docker, garantissant la durabilité des données même si les conteneurs sont arrêtés ou recréés.

Les deux services sont connectés via un réseau Docker pour permettre une communication efficace et sécurisée entre eux.

Lancement des Services

Pour lancer les services définis dans le fichier `docker-compose.yml`, utilisez la commande suivante :

```
docker-compose up
```

Cette commande démarre les conteneurs en arrière-plan.

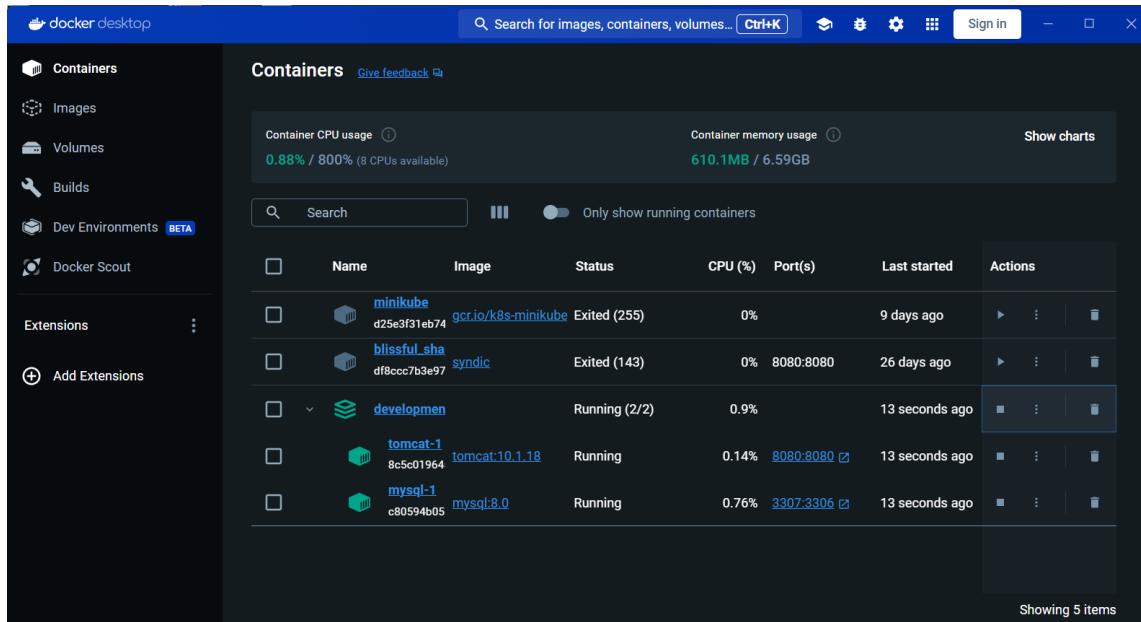


FIGURE 4.44 – Enviroment avec les deux conteneurs

4.4.2 Déploiement avec Kubernetes

Kubernetes est une plateforme d'orchestration de conteneurs qui permet de déployer, gérer et faire évoluer des applications conteneurisées. Le déploiement de l'application se fait à l'aide de fichiers de configuration YAML, notamment `mysql-deployment.yaml` et `tomcat-deployment.yaml`, ainsi qu'un `Dockerfile` pour construire l'image de l'application.

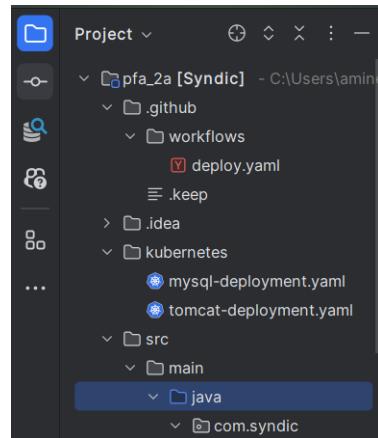


FIGURE 4.45 – Architecture deployment kubernetes

Fichiers de Déploiement

- **mysql-deployment.yaml** : Ce fichier définit un déploiement pour MySQL, incluant les spécifications de la base de données et les volumes pour persister les données.
- **tomcat-deployment.yaml** : Ce fichier définit un déploiement pour Tomcat, incluant la configuration pour monter le fichier WAR de l'application et les ports nécessaires.
- **Dockerfile** : Utilisé pour construire l'image Docker de l'application web.

Conclusion

En conclusion, ce projet de Système d'Informations dédié à la gestion des syndicats de copropriété a été conçu et mis en œuvre avec succès, en suivant la méthodologie MERISE. L'objectif principal était d'automatiser et de simplifier la gestion des résidences, des charges, des tâches, des incidents, du reporting, des alertes et notifications des assemblées, ainsi que des paiements des résidents, en fournissant une solution robuste, sécurisée et conviviale pour l'ensemble du processus.

Le rapport détaille de manière exhaustive les différentes composantes du système, en commençant par un dictionnaire complet des données associées. Chaque étape de la réalisation a été soigneusement documentée, des choix technologiques aux aspects de sécurité, en passant par les tests et l'architecture du système.

Des choix technologiques solides ont été effectués, avec l'utilisation de MySQL pour la base de données, Java pour le développement backend, et des technologies modernes et des frameworks associés. La sécurité des données a été assurée par l'utilisation de jBCrypt pour le hachage des mots de passe.

De plus, la mise en place de Docker et Kubernetes a permis de déployer l'application de manière efficace et scalable. La création de conteneurs pour MySQL et Tomcat a facilité la gestion des environnements de développement et de production, et l'architecture en micro-services a offert une grande flexibilité pour l'évolution future du système.

Enfin, notre projet s'inscrit pleinement dans le cadre juridique marocain, en particulier la loi n° 18-00 relative au statut de la copropriété des immeubles bâties. Cette loi a établi les bases nécessaires à l'organisation et à la réglementation des relations entre les copropriétaires, et notre plateforme vise à faciliter l'application de ces règles en simplifiant la gestion quotidienne des syndicats de copropriété.

En somme, cette plateforme de gestion de syndic de copropriété représente une avancée significative vers une gestion plus efficace et transparente des immeubles en copropriété au Maroc, répondant ainsi aux besoins des gestionnaires et des copropriétaires tout en respectant le cadre légal en vigueur.

Bibliographie

- [1] Loi n° 18-00 relative au statut de la copropriété des immeubles bâtis (2002), 2002. Dahir n° 1-02-298 du 25 rejab 1423 (3 octobre 2002).
- [2] Étude sur la gestion des copropriétés au maroc. Technical report, Ministère de l'Habitat et de la Politique de la Ville, 2017.
- [3] Mike Adams and Jonathan Golant. *Property Management*. Routledge, 2016.
- [4] Apache Software Foundation. Maven. <https://maven.apache.org/>, 2023. Outil de gestion de projet et de dépendances.
- [5] Claude Brunet. *La Copropriété : Réglementation et Pratique*. Éditions Dalloz, 2014.
- [6] Nick Downie. Chart.js. <https://www.chartjs.org/>, 2023. Bibliothèque JavaScript pour la création de graphiques.
- [7] Ahmed El Hajjam. *La gestion informatisée des copropriétés au Maroc : Défis et perspectives*. PhD thesis, Université Mohammed V, 2019.
- [8] Samira Hammadi. *L'impact de la digitalisation sur la gestion des syndicats de copropriété au Maroc*. PhD thesis, Université Hassan II, 2018.
- [9] iText Group NV. itext. <https://itextpdf.com/>, 2023. Bibliothèque pour la génération de documents PDF.
- [10] JUnit Team. Junit. <https://junit.org/junit5/>, 2023. Framework de tests unitaires pour Java.
- [11] Oracle Corporation. Javamail api. <https://javaee.github.io/javamail/>, 2023. Bibliothèque pour la gestion et l'envoi d'emails.