

RAPPORT
de Projet de Fin d'Année
Spécialité GENIE –INFORMATIQUE

Réalisation d'un projet du
Gestion d'immobiliers

Réalisé par :

BENALI Mohamed-Amine

HAMOUICH Walid

EL-MOUSSAOUI Chakir

BELMADANI Mohamed

BENMASAOUD Ilyas

Encadrant : KHADIRI Issam

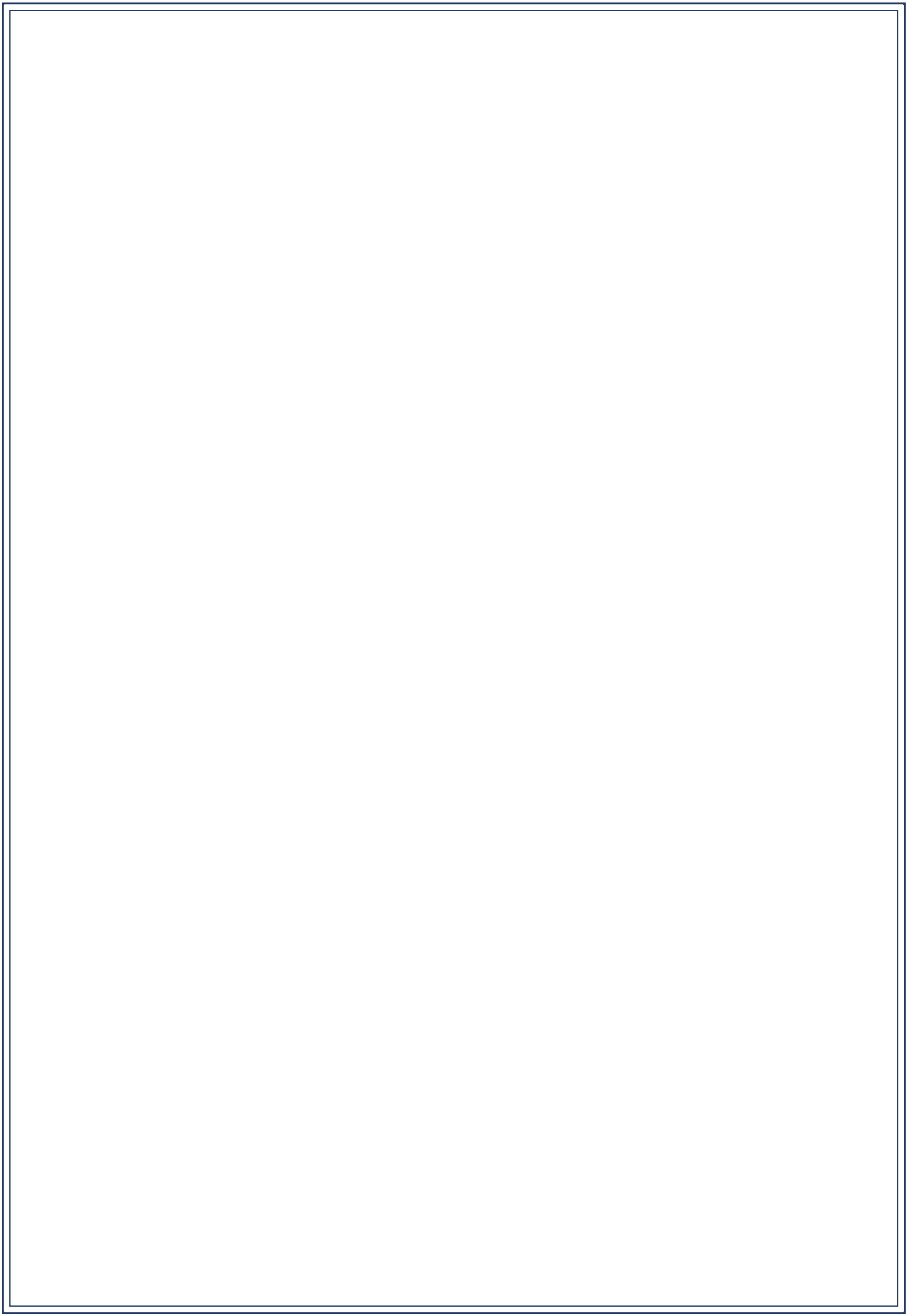
Soutenu le 01/06/2024 devant le jury composé de :

Mr. MOUHIB Imad

Mr. SALHI Khalid

Mr. MAZOUZ Ilyes

Année Universitaire : 2023/2024



RAPPORT
de Projet de Fin d'Année
Spécialité GENIE –INFORMATIQUE

Réalisation d'un projet du
Gestion d'immobiliers

Réalisé par :

BENALI Mohamed-Amin

HAMOUICH Walid

EL-MOUSSAOUI Chakir

BELMADANI Mohamed

BENMASSAOUD Ilyas

Encadrant : Mr. KHADIRI Issam

Soutenu le 01/06/2024 devant le jury composé de :

Mr. MOUHIB Imad

Mr. SALHI Khalid

Mr. MAZOUZ Ilyes

Année Universitaire : 2023/2024

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

« Dans les affaires, Les grandes choses ne sont jamais faites par une seule personne, elles sont faites par toute une équipe. » Steve Jobs.

Remerciement

Tout d'abord, nous remercions ALLAH source de toute connaissance.

Nos remerciements les plus cordiaux s'adressent à notre encadrant interne Mr. KHADIRI ISSAM à l'Ecole des Hautes Etudes d'Ingénierie, pour son aide, son attention et son orientation. Ainsi que pour ses précieux conseils concernant les missions évoquées dans ce rapport, pour sa disponibilité et ses directives.

Nous adressons nos remerciements les plus sincères aux professeurs de l'Ecole des Hautes Etudes d'Ingénierie de nous avoir soutenus tout au long de notre formation et de nous avoir transmis toutes les connaissances et compétences qui nous ont permis de réaliser ce travail, nous les saluons également pour leurs qualités et intelligences émotionnelles qui ont fait de notre formation une expérience très enrichissante personnellement.

Finalement, nous remercions tous les membres du jury pour nous avoir honorés en acceptant de participer à cette soutenance et d'évaluer ce travail.

Dédicace

Nous dédions ce modeste travail à :

Nos très chers parents :

« Que nulle dédicace ne puisse exprimer notre sincères sentiments, pour leur patience illimitée, leur encouragement contenu, leur aide, en témoignage de notre profond amour et respect pour leurs grands sacrifices, que dieu vous protège et vous garde pour nous. »

Nos chers frères et sœurs :

« Pour leur grand amour et leur soutien qu'ils trouvent ici l'expression de notre haute gratitude. »

Nos chers ami(e)s :

« Avec lesquels nous avons partagé nos moments de joie et de bonheur pour votre fidélité et votre soutien. »

Nos enseignants :

« Pour votre enseignement et vos conseils tout au long de notre parcours éducatif et professionnel. »

Résumé

Notre projet de fin d'année a pour ambition la création d'une application web facilitant la vente et la location des biens immobiliers. Ce rapport met en exergue notre parcours en matière de la conception et de la réalisation de ladite application Web.

Dans la perspective de bien structurer notre vision du projet, nous avons opté pour l'utilisation du langage de modélisation UML, marqué par sa capacité à offrir une représentation détaillée et puissante, à même d'assurer une planification méthodique et une compréhension approfondie des différents aspects.

Pour que cette vision devient concrète, nous allons juger opportun d'utiliser de nombreuses technologies modernes telles que : Api Platform, Docker, Symfony, Angular, JavaScript, CSS, et MySQL...

L'association de ces technologies susmentionnées, avec une modélisation UML détaillée, positionne le présent projet pour répondre aux exigences élevées de la création d'une plateforme web moderne.

Mot clé : Plateforme , application web, UML, technologie moderne.

Abstract

Our end-of-studies project aims to create a web application facilitating the sale and rental of real estate. This report highlights our journey in the design and creation of said web application.

With a view to properly structuring our vision of the project, we opted for the use of the UML modeling language, marked by its ability to offer a detailed and powerful representation, capable of ensuring methodical planning and an in-depth understanding of the different aspects.

To make this vision a reality, we will consider it appropriate to use many modern technologies such as: Api Platform, Docker, Symfony, Angular, JavaScript, CSS, and MySQL...

The combination of these aforementioned technologies, with detailed UML modeling, positions the present project to meet the high demands of creating a modern web platform.

Keyword : platform, web application, UML, modern technologie.

Table des matières

Remerciement	I
Dédicace.....	II
Résumé.....	III
Abstract	IV
Table des matières	V
Liste des figures	VIII
Liste des acronymes	X
Introduction générale	1
Chapitre 1 : Présentation et cadrage du projet.....	2
1.1 Introduction	3
1.2 Présentation et cadrage du projet.....	3
1.2.1 Contexte et définition du projet	3
1.2.1.1 Contexte du projet	3
1.2.1.2 Définition du projet	3
1.2.2 Problématique	3
1.2.3 Objectifs	4
1.2.4 Besoin fonctionnel	4
1.2.5 Besoin non fonctionnel	5
1.3 Déroulement du projet.....	5
1.3.1 Planification opérationnelle	5
1.3.2 Tableau des tâches.....	5
1.3.3 Diagramme de Gantt	6
1.4 Conclusion.....	7
chapitre 2 : Conception et modélisation	8
2.1 Introduction	9
2.2 Langage de modélisation UML	9
2.3 Diagramme de cas d'utilisation.....	9
2.3.1 Définitions.....	9
2.3.2 Les composants du diagramme de cas d'utilisation.....	9

2.4	Diagramme de classe.....	11
2.5	Diagramme de séquence.....	12
2.5.1	Définitions.....	12
2.5.2	Diagramme de séquence du cas d'utilisation se connecter	13
2.5.3	Diagramme de séquence du cas d'utilisation ajouter annonce	14
2.5.4	Diagramme de séquence cas d'utilisation sponsoriser.....	14
2.5.5	Diagramme de séquence du cas d'utilisation valider annonce	15
2.6	Diagramme de séquence MVC.....	16
2.6.1	Diagramme de séquence MVC du cas d'utilisation Ajouter à la liste noire 16	
2.6.2	Diagramme de séquence MVC du cas « Ajouter à la liste des favoris » ...	18
2.6.3	Diagramme de séquence MVC du cas Supprimer de la liste des favoris .	19
2.6.4	Diagramme de séquence MVC du cas Supprimer une annonce	20
2.6.5	Diagramme de séquence MVC du cas d'utilisation Créer un compte	21
2.6.6	Diagramme de séquence MVC du cas d'utilisation Ajouter Avis.....	22
2.6.7	Diagramme de séquence MVC du cas Consulter les biens les plus vus	23
2.6.8	Diagramme de séquence MVC du cas Rechercher des biens + Consulter les biens	23
2.7	Architecture Système	25
2.8	Conclusion.....	26
Chapitre 3 : Implémentation et réalisation		27
3.1	Introduction	28
3.2	Environnement de développement	28
3.2.1	Environnement matériel	28
3.2.2	Architecture utilisée	28
3.3	Pourquoi utiliser un Framework.....	29
3.4	Environnement logiciel	30
3.4.1	Adobe Photoshop	30
3.4.2	Git et GitHub.....	30
3.4.3	Draw.io	32
3.4.4	Visual Studio Code:	32
3.4.5	Discord	33
3.4.6	MySQL.....	33
3.4.7	Docker.....	34
3.4.8	Doctrine.....	34
3.4.9	Nginx.....	35
3.5	Langage et Framework utiliser	35

3.5.1	Définition PHP	35
3.5.2	Définition Symfony.....	36
3.5.3	Définition de Angular.....	37
3.5.4	Définition Api Platform	38
3.5.5	Définition de JavaScript.....	39
3.5.6	Définition de TypeScript	40
3.5.7	Définition de CSS	40
3.5.8	Définition HTML	41
3.3	Les principales interfaces du Front End (visiteur et adhérent).....	42
3.3.1	Interface principale pour visiteur et adhérent.....	42
3.3.2	Interface de connexion pour l'adhérent	42
3.3.3	Interface d'inscription pour le visiteur	43
3.3.4	Interface de liste des propriétés pour l'adhérent et le visiteur	44
3.3.5	Interface d'Ajout d'une propriété pour l'adhérent	44
3.3.6	Interface des annonces d'adhérents offreurs	45
3.4	Les principales interfaces du Back End pour Admin et Super Admin	46
3.5	Interface Api Platform	49
3.6	Conclusion.....	49
	Conclusion générale.....	50
	Bibliographie.....	51

Liste des figures

Figure 1 - liste des acronymes	X
Figure 2 – Table des tâches	6
Figure 3 - Diagramme de Gantt	7
Figure 4 - Diagramme de cas d'utilisation	11
Figure 5 - Diagramme de classe	12
Figure 6 - Diagramme de séquence du cas d'utilisation « se connecter »	13
Figure 7 - Diagramme de séquence cas d'utilisation « ajouter annonce »	14
Figure 8 - Diagramme de séquence cas d'utilisation « sponsoriser »	15
Figure 9 - Diagramme de séquence cas d'utilisation « valider l'annonce »	16
Figure 10 - Diagramme de séquence MVC « Ajouter à la liste noire »	17
Figure 11 - Diagramme de séquence MVC pour «Ajouter à la liste des favoris »	18
Figure 12 - Diagramme de séquence MVC pour « supprimer liste des favoris»	19
Figure 13 - Diagramme de séquence MVC pour «Supprimer une annonce»	20
Figure 14 - Diagramme de séquence MVC «Créer un compte»	21
Figure 15 - Diagramme de séquence MVC «Ajouter Avis»	22
Figure 16 - Diagramme de séquence MVC du cas «Consulter les biens les plus vus»	23
Figure 17 - Diagramme de séquence du cas «Rechercher les biens» + «Consulter les biens»	24
Figure 18 - Schéma Architecture Système	25
Figure 19 - Architecture MVC	29
Figure 20 - Adobe Illustrator	30
Figure 21 - Logo Git	31
Figure 22 - Logo GitHub	31
Figure 23 – Logo de Draw.io	32
Figure 24 - Logo Visual Studio Code	32
Figure 25 - logo Discord	33
Figure 26 - Logo MySQL	33
Figure 27 - Logo Docker	34
Figure 28 - Logo Doctrine	34
Figure 29 - Logo Nginix	35
Figure 30 - Logo PHP	36
Figure 31 - Logo Symfony	36
Figure 32 - Logo Angular	37
Figure 33 - Logo ApiPlatform	38
Figure 34 - Logo JavaScript	39
Figure 35 - Logo TypeScript	40
Figure 36 - Logo CSS	40
Figure 37 - Logo html	41
Figure 38 - Interface principale pour l'adhérent et visiteur	42

Figure 39 - Interface de connexion pour l'adhérent	43
Figure 40 - Interface d'inscription pour le visiteur	43
Figure 41 - Interface de liste des propriétés pour l'adhérent et le visiteur	44
Figure 42 - Interface d'Ajout d'une propriété pour l'adhérent	44
Figure 43 - Interface des annonces d'adhérents offreurs	45
Figure 44 - Interface du Dashboard pour l'adhérent	45
Figure 45 - Interface de connexion pour Admin et SuperAdmin	46
Figure 46 - Interface d'ajout un utilisateur par admin ou SuperAdmin	46
Figure 47 - Interface pour l'ajout des utilisateurs	47
Figure 48 - Interface de liste des annonces des adhérents	47
Figure 49 - Interface de supprimer l'annonce par admin ou SuperAdmin	48
Figure 50 - Interface de table de bord d'administrative	48
Figure 51 - Interface Api Platform	49

Liste des acronymes

UML	Unified Modeling Language
HTML	HyperText Markup Language,
CSS	Cascading Style Sheets
JS	JavaScript
PHP	Hypertext Preprocessor
MYSQL	My Structured Query Language
TS	TypeScript
API	Application Programming Interface

Figure 1 - liste des acronymes

Introduction générale

Au cours des dernières années, l'informatique s'est imposée d'une manière très impressionnante dans le monde professionnel. Ceci est dû à son apport dans le domaine de la gestion des systèmes d'information et des bases de données. Aujourd'hui, l'informatique est de plus en plus utilisée dans tous les domaines d'activités y compris le domaine de l'immobilier, domaine auquel est rattaché ce travail.

L'objectif de notre projet de fin d'année réalisé dans le cadre de notre formation à l'Ecole des Hautes Etudes d'Ingénierie (EHEI) était de développer une application web de gestion de l'immobilier.

Nous avons choisi d'utiliser le langage de modélisation UML pour décrire et organiser notre projet, car il offre un niveau de détail approprié à nos besoins. Pour l'implémentation, nous avons utilisé le logiciel tels que Visual Studio Code et Les langages de programmation que nous avons utilisés qui incluent HTML, CSS, JavaScript, ainsi que les Frameworks : Bootstrap, Angular et Symfony, API plateforme et aussi la plateforme docker.

Le résultat de notre travail est un site web permettant de présenter nos offres à un large public. Notre application web offre la possibilité à tout le monde de naviguer, d'acheter, et louer en ligne avec toute sécurité.

Dans la suite de ce rapport, nous présenterons en détail les différentes étapes de développement de notre projet, en expliquant les choix technologiques que nous avons faits et en mettant en évidence les fonctionnalités clés de notre système.

Nous espérons que notre Projet apportera une valeur ajoutée au domaine de l'immobilier en ligne, en offrant une expérience utilisateur intuitive et en facilitant l'accès pour les clients potentiels.

Chapitre 1 : Présentation et cadrage du projet

1.1 Introduction

Ce chapitre sera consacré à la planification et au cadrage général du projet de fin d'année. Nous commencerons par la définition du tableau des tâches, puis nous construirons le planning en utilisant le diagramme de GANTT. Ensuite, nous présenterons le cycle de vie du projet. En parallèle, nous décrirons les fonctionnalités du produit ainsi que les besoins fonctionnels et non fonctionnels du projet. Cette approche globale permettra de structurer efficacement notre travail, en garantissant que chaque aspect du projet est pris en compte et que nous avons une feuille de route claire pour atteindre nos objectifs.

1.2 Présentation et cadrage du projet

1.2.1 Contexte et définition du projet

1.2.1.1 Contexte du projet

Ce travail s'inscrit dans le cadre du projet de fin d'année pour quatrième année Génie Informatique au sein de l'Ecole des Hautes Etudes d'Ingénierie. Dans ce contexte, ce travail consiste à la mise en place d'un projet informatique pour la vente et location en ligne. La création de cette application web mettra en pratique nos connaissances théoriques et pratiques que nous avons acquises tout au long durant cette année d'études au sein de l'EHEI.

1.2.1.2 Définition du projet

Dans le cadre de notre savoir-faire maîtrisé ce semestre, nous voulons réaliser un projet visant la création d'une application web de vente et location en ligne des biens immobiliers. L'application va gérer les différentes offres et les différentes relations avec les clients grâce à une base de données. Ainsi, l'application permet de faciliter la tâche de la vente et location des offres. Dans le cadre de ce projet qui permettra de :

- Mettre en ligne des biens immobiliers ;
- Gérer les relations avec les clients ;
- Gérer les problèmes des offres.

1.2.2 Problématique

Le but de notre travail, la conception d'une application Web qui gère la vente et la location des biens immobiliers, est de répondre à de nombreux problèmes qui entravent la qualité du service offert aux clients, tant au niveau de l'exécution qu'au niveau de la 'post-production' (publicité/promotion).

Dans le présent projet, notre application propose des solutions aux problèmes mentionnés ci-après :

- Difficultés dans le processus de la demande : la perte du temps lorsqu'ils essayent de chercher les biens offerts.
- Manque d'informations détaillées : les clients peuvent rencontrer des difficultés liées au manque d'informations détaillées sur les biens. Cela peut inclure des descriptions incomplètes, des images de basse qualité ou l'absence d'informations cruciales telles que les plans d'étages, les commodités, etc.

1.2.3 Objectifs

Après avoir identifié les problèmes dans le processus d'achat et de location de biens immobiliers, notre application Web a pour objectif de mettre en œuvre des solutions innovantes afin d'améliorer significativement l'expérience des utilisateurs. Les objectifs spécifiques de notre projet comprennent :

- Fournir tous les outils nécessaires : Offrir une plateforme complète qui propose tous les outils et fonctionnalités essentiels dont les clients ont besoin pour naviguer, rechercher, acheter ou louer des biens immobiliers.
- Faciliter la recherche des propriétés : Simplifier le processus de recherche en mettant en place des filtres avancés, des options de tri, et en présentant des informations détaillées sur chaque propriété pour faciliter la prise de décision.

1.2.4 Besoin fonctionnel

Les besoins fonctionnels correspondent aux fonctionnalités du système. Ce sont les besoins spécifiant un comportement d'entrée / sortie du système. Le système doit permettre :

A l'administratif de :

S'authentifier : saisir son login et mots de passe pour accéder à son tableau de bord.

Gérer les offres : Ajouter, supprimer, modifier et consulter les offres avec leurs avis.

Au client de :

S'authentifier : saisir son login et mots de passe.

Gérer ses offres : Ajouter, supprimer, modifier et consulter les offres.

1.2.5 Besoin non fonctionnel

À part les besoins fonctionnels, notre système doit répondre aux critères suivants :

- Besoins de performance : il s'agit d'utiliser les bonnes pratiques de développement pour fournir un temps de réponse minime. En effet, l'application doit être capable de supporter un grand nombre de connexions simultanées,
- Besoins de disponibilité : la possibilité d'accès à l'application et à n'importe quelle information 24h/24 et 7j/7.
- Besoins de déploiement : dans le cadre de ce travail, l'application devra être accessible via un équipement équipé d'un serveur Web.
- Besoins de convivialité : le système de centralisation doit être facile à comprendre et à utiliser. En effet, l'interface devra être conviviale, interactive, cohérente du point de vue ergonomie et bien adaptée à l'utilisation,
- Besoins de sécurité : il faut que notre serveur garantisse la sécurité de notre application car son contenu ne doit être accessible qu'aux membres autorisés.

1.3 Déroulement du projet

Cette deuxième partie consiste à planifier le déroulement du projet. Nous commençons par définir le tableau des tâches ainsi le diagramme de GANTT (une technique très utilisée de construction de planning).

1.3.1 Planification opérationnelle

Nous proposons le planning de travail suivant qui contient les différentes tâches du projet, leurs ordres et leurs temps de réalisation d'une façon provisoire, et qui est susceptible d'être modifié à tout moment selon la progression du travail, les ajouts, les modifications et les difficultés rencontrées pendant les différentes itérations (Diagramme de GANTT prévu). Après qu'on a atteint les objectifs fixés, nous présentons la planification réelle de notre projet (Diagramme de GANTT réel)

1.3.2 Tableau des tâches

Permet d'avoir une vue pure du travail à accomplir, en cours et terminer. Il peut pareillement s'avérer lors des réunions quotidiennes. Le tableau facilite pareillement l'affectation des tâches par l'équipe en ayant une vision d'ensemble de sprint en un regard. Dispensable de se tourmenter à planifier à l'avance dans le détail de l'activité de chaque développeur sur toute la période du sprint.

1.3.3 Diagramme de Gantt

Le diagramme de Gantt, couramment utilisé en gestion de projet, est l'un des outils les plus efficaces pour représenter visuellement l'état d'avancement des différentes activités (tâches) qui constituent un projet. La colonne de gauche du diagramme énumère toutes les tâches à effectuer, tandis que la ligne d'en-tête représente les unités de temps les plus adaptées au projet (jours, semaines, mois etc.). Chaque tâche est matérialisée par une barre horizontale, dont la position et la longueur représentent la date de début, la durée et la date de fin. Ce diagramme permet donc de visualiser d'un seul coup d'œil : Les différentes tâches à envisager :

- La date de début et la date de fin de chaque tâche.
- La durée est comptée de chaque tâche.
- Le chevauchement éventuel des tâches, et la durée de ce chevauchement.
- La date de début et la date de fin du projet dans son ensemble.

En résumé, un diagramme de Gantt répertorie toutes les tâches à accomplir pour mener le projet à la bien, et indiquer la date à laquelle ces tâches doivent être effectuées.



Nom	Date de dé...	Date de fin
Collecte des donnees	05/12/2...	01/01/2...
Elaboration de cahier des ch...	05/12/2...	18/12/2...
Conception du projet	05/12/2...	08/01/2...
Creation des diagrammes de ...	05/12/2...	25/12/2...
Repartition des taches pour l...	05/12/2...	05/12/2...
Documentation sur chaque ta...	05/12/2...	27/12/2...
Realisation des taches	05/12/2...	11/03/2...

Figure 2 - Table des taches

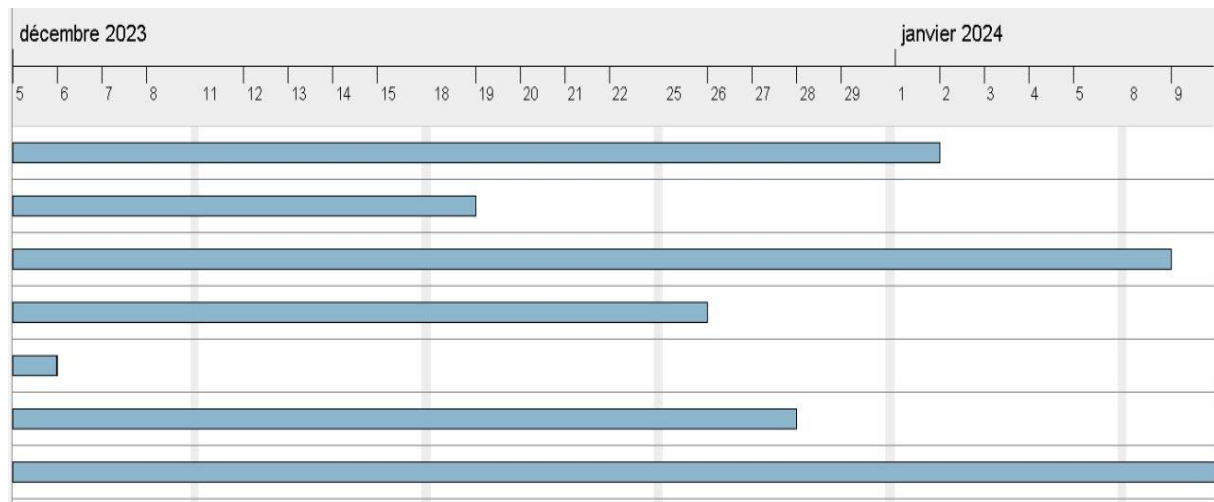


Figure 3 - Diagramme de Gantt

1.4 Conclusion

En conclusion, ce chapitre a été consacré au cadrage général du projet de fin d'année en décrivant les fonctionnalités de notre site, les besoins fonctionnels et non fonctionnels. On a réparti les tâches et organisé le temps de manière à maîtriser les ressources allouées au projet. Cette préparation minutieuse nous permet d'avancer avec une vision claire et structurée. Le chapitre suivant sera consacré à la spécification et à la conception du site web, où nous détaillerons davantage les aspects techniques et architecturaux du projet.

Chapitre 2 : Conception et modélisation

2.1 Introduction

Dans ce chapitre, nous explorons la conception et la modélisation du projet en détaillant les divers éléments clés. Nous examinons les interactions, les flux de données et la structure globale du système. Enfin, nous aboutissons à une représentation visuelle du contexte statique et dynamique, suivie de la modélisation conceptuelle à l'aide de l'UML.

2.2 Langage de modélisation UML

UML ou langage de modélisation unifié, est un langage de modélisation graphique. Il est utilisé en développement logiciel, et en conception orienté objet. Ce langage est constitué de diagrammes. Ces diagrammes sont tous réalisés à partir des besoins des utilisateurs :

- Diagramme de cas d'utilisation.
- Diagramme de séquence.
- Diagramme de séquence MVC.
- Diagramme de classe.

2.3 Diagramme de cas d'utilisation

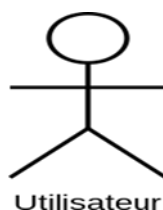
2.3.1 Définitions

Le rôle des diagrammes de cas d'utilisation est : de recueillir, d'analyser et d'organiser les besoins, ainsi que de recenser les grandes fonctionnalités d'un système. Il s'agit donc de la première étape UML pour la conception d'un système.

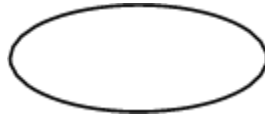
Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Il scinde la fonctionnalité du système en unités cohérentes; les cas d'utilisation, ayant un sens pour les acteurs. Ainsi, ces cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un système, ils sont donc une vision orientée utilisateur de ce besoin au contraire d'une vision informatique.

2.3.2 Les composants du diagramme de cas d'utilisation

Le diagramme se compose de trois éléments principaux



Un Acteur : c'est l'idéalisation d'un rôle joué par une personne externe, un processus ou une chose qui interagit avec un système. Il se représente par un petit bonhomme avec son nom inscrit dessous.



Un cas d'utilisation : c'est une unité cohérente représentant une fonctionnalité visible de l'extérieur. Il réalise un service de bout en bout, avec un déclenchement, un déroulement et une fin, pour l'acteur qui l'initie. Un cas d'utilisation modélise donc un service rendu par le système, sans imposer le mode de réalisation de ce service. Il représente par une ellipse contenant le nom du cas (un verbe à l'infinitif), et optionnellement, au-dessus du nom, un stéréotype.

Les relations : trois types de relations sont pris en charge par la norme UML et sont graphiquement représentées par des types particuliers de ces relations. Les relations indiquent que le cas d'utilisation source présente les mêmes conditions d'exécution que le cas issu. Une relation simple entre un acteur et un cas d'utilisation est un trait simple.

Les acteurs de notre projet :

Adhèrent : cet acteur est un visiteur ayant déjà créé un compte sur notre système. Il peut donc suivre le processus d'achat ou de location des offres en toute sécurité sachant que notre système doit être l'unique responsable de la confidentialité des données personnelles de ses clients et aussi peut gérer un avis (Ajouter, modifier, supprimer).

Administrateur et SuperAdmin : assure le dynamisme de l'application, celui qui gère le système et analyse les avis des visiteurs, et aussi gérer les comptes.

Visiteur : En tant que visiteur, vous avez la possibilité d'entreprendre plusieurs actions sur le site ou la plateforme immobilière. Ces actions incluent la création d'un compte, la consultation des biens les plus vus, ainsi que la visualisation des biens récemment consultés

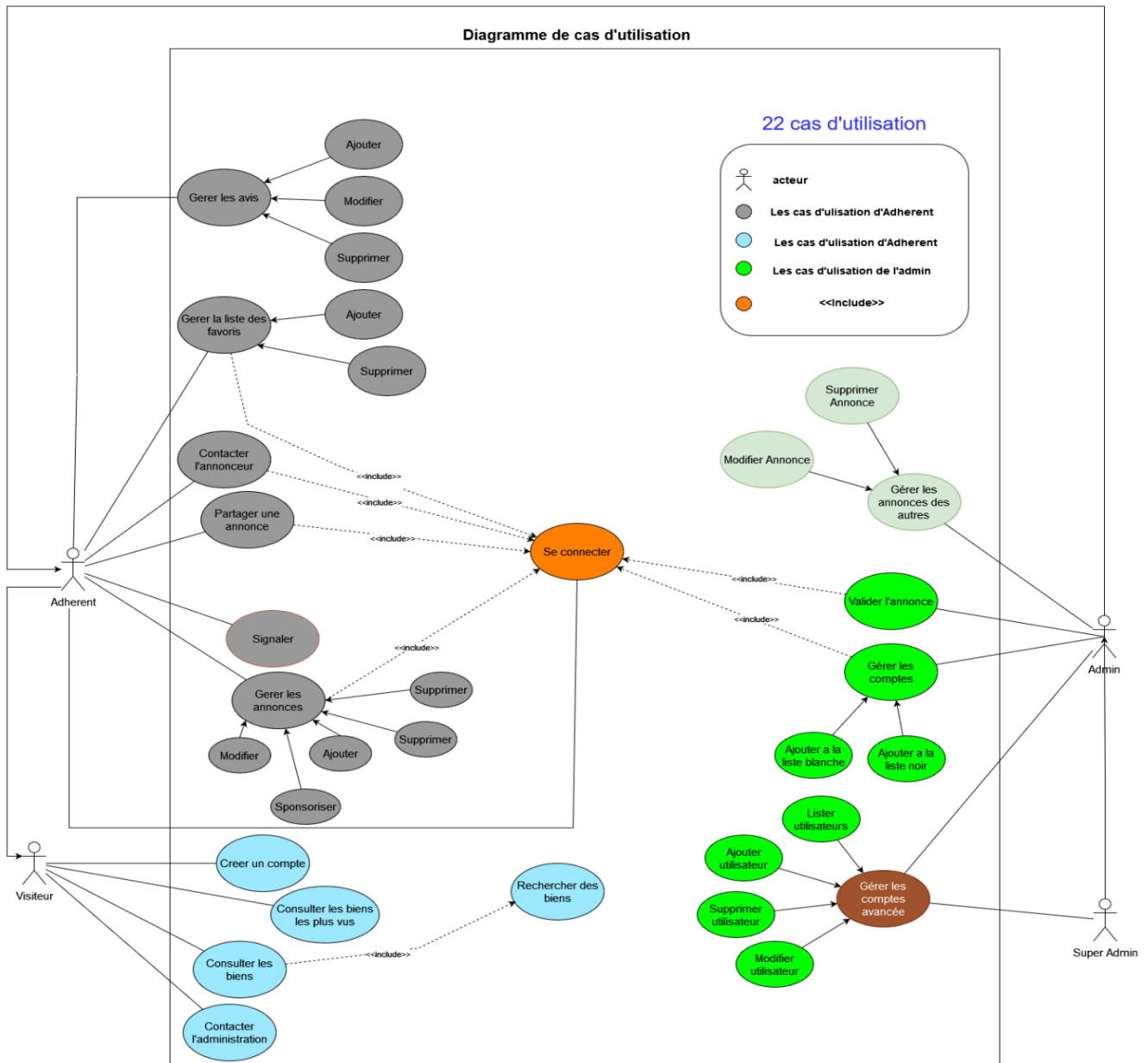


Figure 4 - Diagramme de cas d'utilisation

2.4 Diagramme de classe

Un diagramme de classes est une collection d'éléments de modélisation statique (classes, packages...), qui montre la structure d'un modèle. C'est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Ce diagramme fait partie de la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques.

Les classes peuvent être liées entre elles grâce au mécanisme d'héritage qui permet de mettre en évidence des relations de parenté. D'autres relations sont possibles entre les classes, chacune

de ces relations est représentée par un arc spécifique dans le diagramme de classes. Chaque classe est représentée par un rectangle comprenant trois parties : nom de la classe, attributs et opérations.

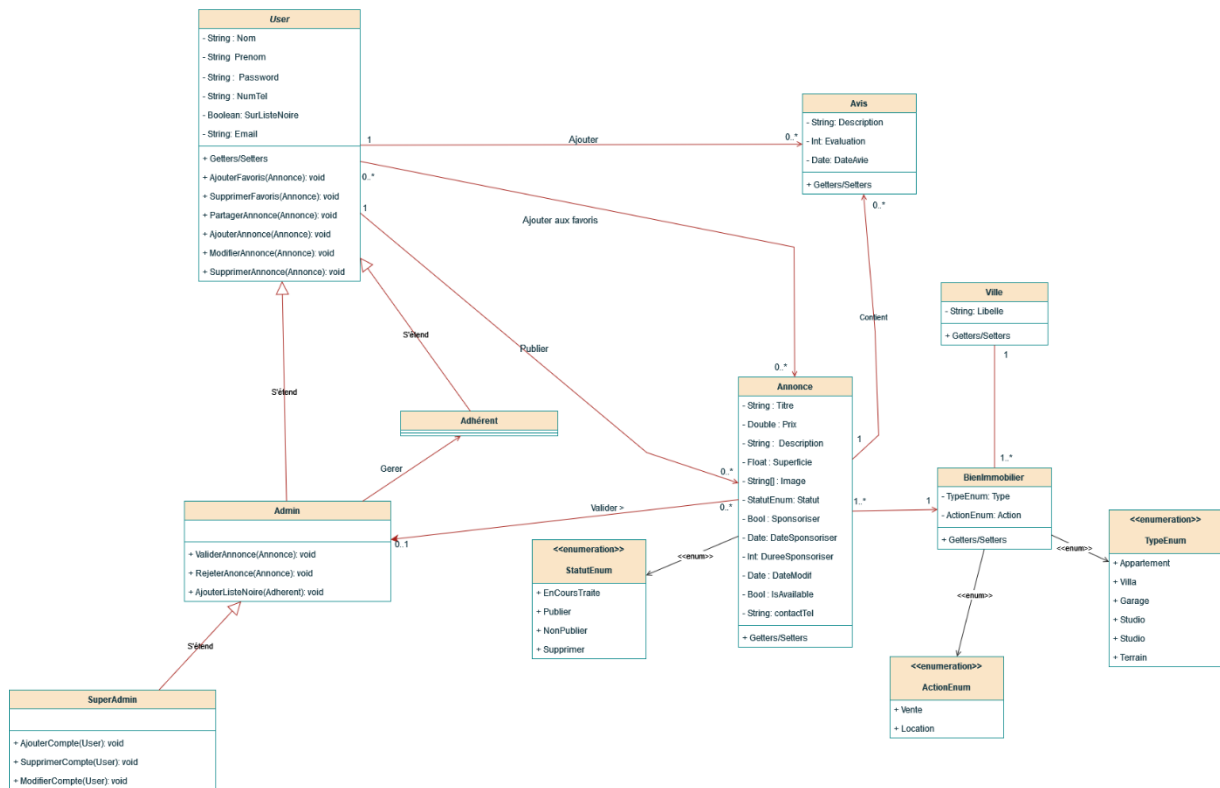


Figure 5 - Diagramme de classe

2.5 Diagramme de séquence

2.5.1 Définitions

Le diagramme de séquence est une description graphique des opérations d'un système sous un angle chronologique. C'est une vue dynamique qui contient les symboles d'objets (instances de classe), d'acteurs et de messages qu'ils échangent. La dimension verticale est l'axe temporel : les messages y sont représentés par ordre chronologique. La dimension horizontale montre des objets et des acteurs qui échangent des informations, le diagramme de séquence permet de représenter les échanges entre les composants et les objets du système, dans le cadre d'exécution des cas d'utilisation, de point de vue temporel.

Dans ce qui suit, nous présentons quelques fonctions de l'utilisateur à travers des différents diagrammes de séquences.

2.5.2 Diagramme de séquence du cas d'utilisation se connecter

Ce diagramme de séquence représente le processus de connexion d'un adhérent. L'adhérent fournit son nom d'utilisateur et son mot de passe, puis le système vérifie ces informations. En cas de succès, l'adhérent accède à l'espace réservé (tableau de bord) aux membres, sinon, un message d'erreur est affiché. Ce schéma illustre de manière succincte l'interaction entre l'adhérent et le système lors du processus d'authentification.

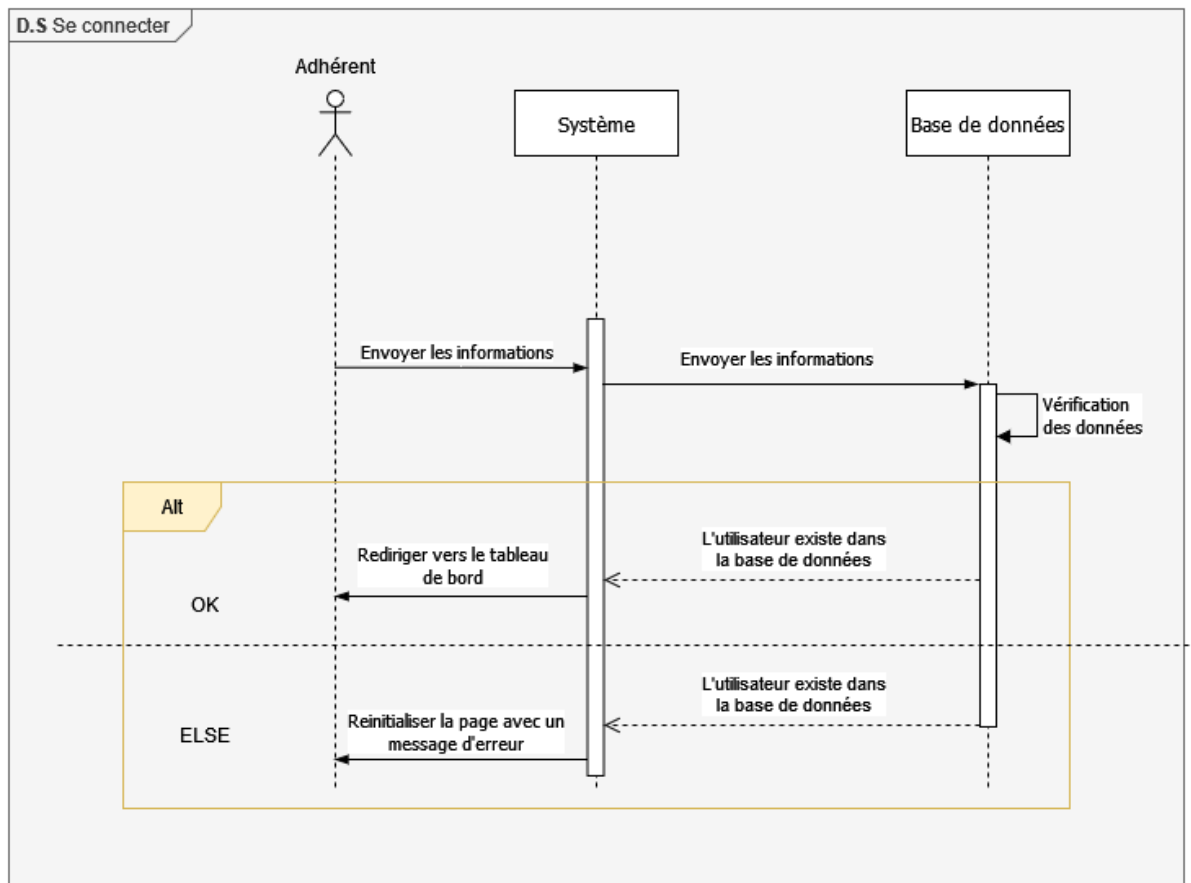


Figure 6 - Diagramme de séquence du cas d'utilisation « se connecter »

2.5.3 Diagramme de séquence du cas d'utilisation ajouter annonce

Ce diagramme de séquence décrit le processus d'ajout d'une annonce par un adhérent connecté. L'adhérent fournit les détails de l'annonce, tels que le titre et la description..., au système. Ensuite, le système valide ces informations. Si la validation est réussie, l'annonce est ajoutée, et un message de confirmation est renvoyé à l'adhérent. En cas d'échec de la validation, un message d'erreur indiquant la nature du problème affiché. Le schéma illustre de manière simple l'interaction entre l'adhérent et le système lors du processus d'ajout d'une annonce.

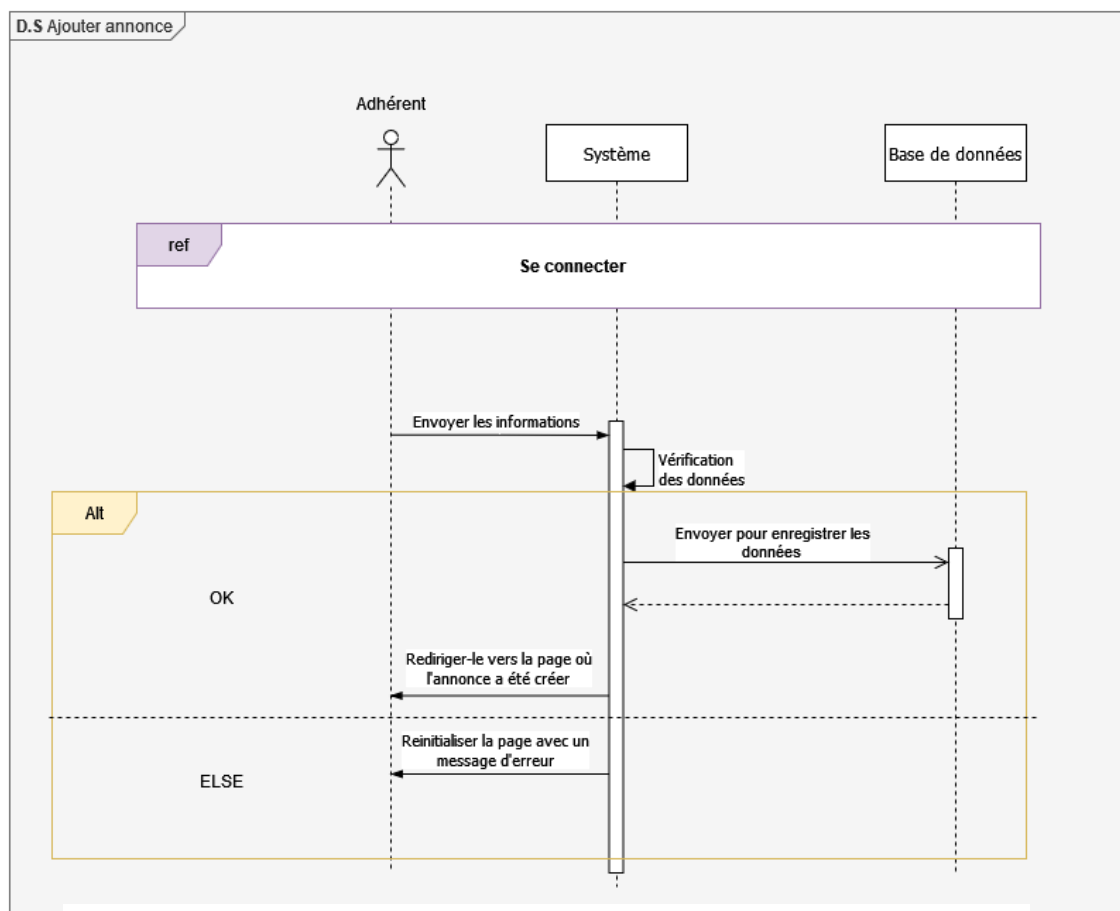


Figure 7 - Diagramme de séquence cas d'utilisation « ajouter annonce »

2.5.4 Diagramme de séquence cas d'utilisation sponsoriser

Ce diagramme de séquence représente la séquence d'actions pour sponsoriser une annonce par un adhérent connecté. L'utilisateur initie la demande de sponsoring pour une annonce spécifique. Il fournit ensuite les informations nécessaires pour effectuer le paiement. Une fois ces détails saisis, le système transmet les informations au fournisseur de la passerelle de paiement. Lorsque la transaction est effectuée avec succès, le système procède à la modification des données de l'annonce dans la base de données. Simultanément, la page est actualisée pour

refléter ces changements. Enfin, un message de succès est affiché à l'adhérent, confirmant que la transaction a été réalisée avec succès.

2.5.5 Diagramme de séquence du cas d'utilisation valider annonce

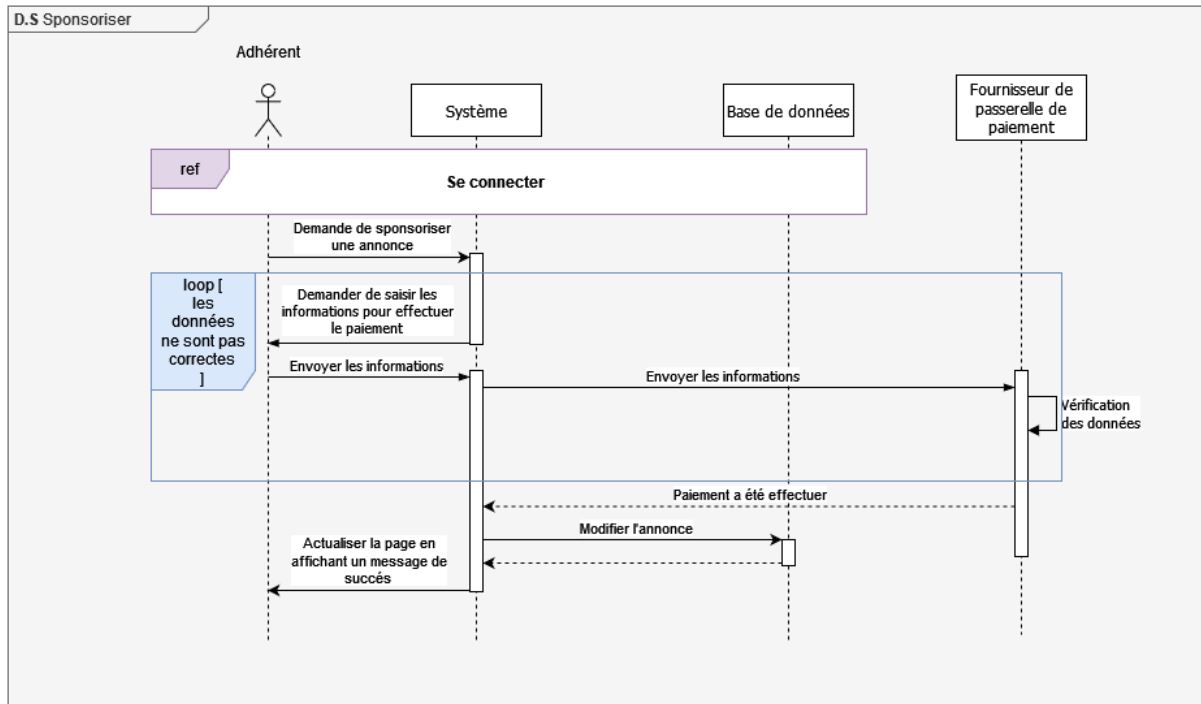


Figure 8 - Diagramme de séquence cas d'utilisation « sponsoriser »

Le processus commence par la connexion de l'administrateur. Une fois connecté, l'administrateur demande à visualiser toutes les annonces en attente à partir du système. Ensuite, il sollicite la base de données pour obtenir les informations nécessaires. Après avoir examiné les détails, l'administrateur valide l'annonce. Dans le cas contraire, l'administrateur la rejette. Ce processus assure un contrôle administratif sur la validation des annonces en attente dans le système.

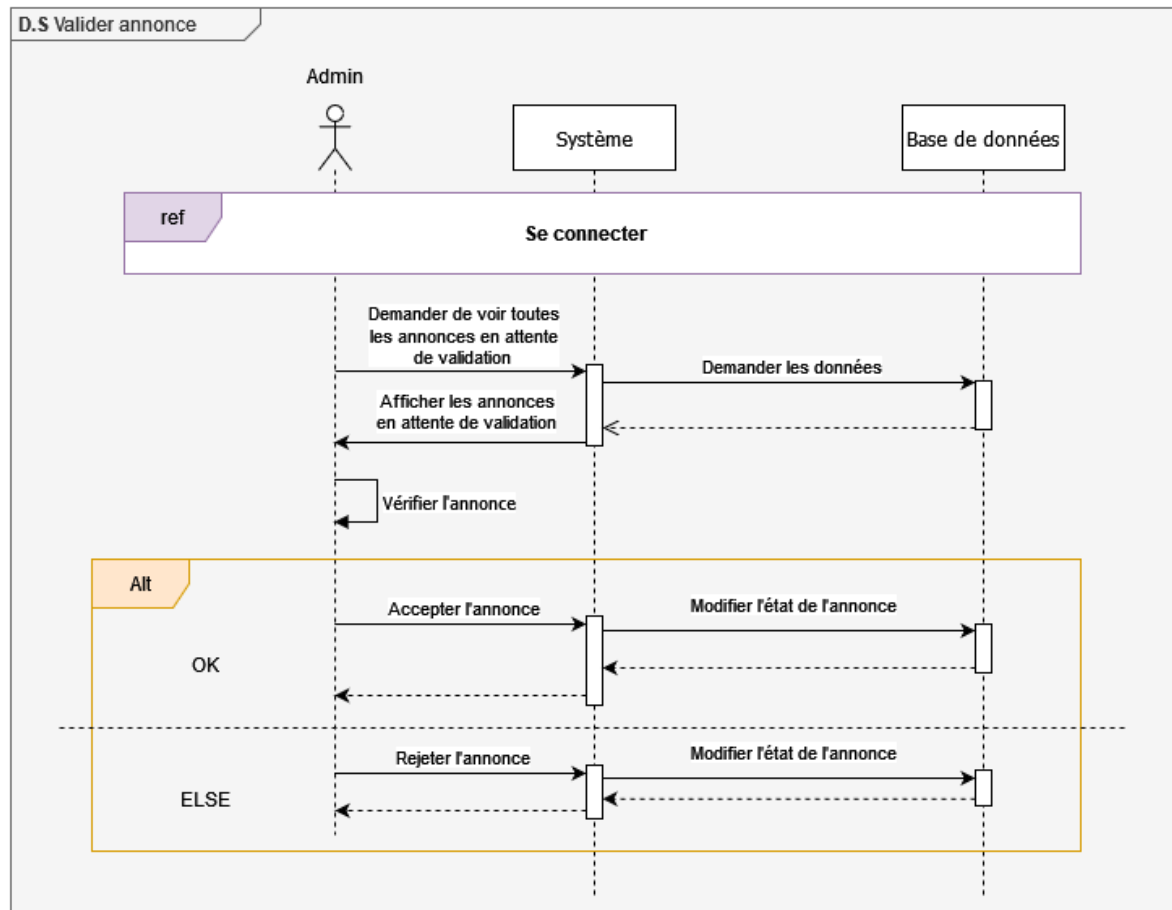


Figure 9 - Diagramme de séquence cas d'utilisation « valider l'annonce »

2.6 Diagramme de séquence MVC

Le diagramme de séquence MVC est une représentation visuelle qui montre comment les parties principales du Modèle-Vue-Contrôleur (MVC) interagissent pour accomplir des tâches dans une application. Il met en évidence comment le modèle gère les données, comment la vue gère l'interface utilisateur et comment le contrôleur coordonne les échanges entre le modèle et la vue.

2.6.1 Diagramme de séquence MVC du cas d'utilisation Ajouter à la liste noire

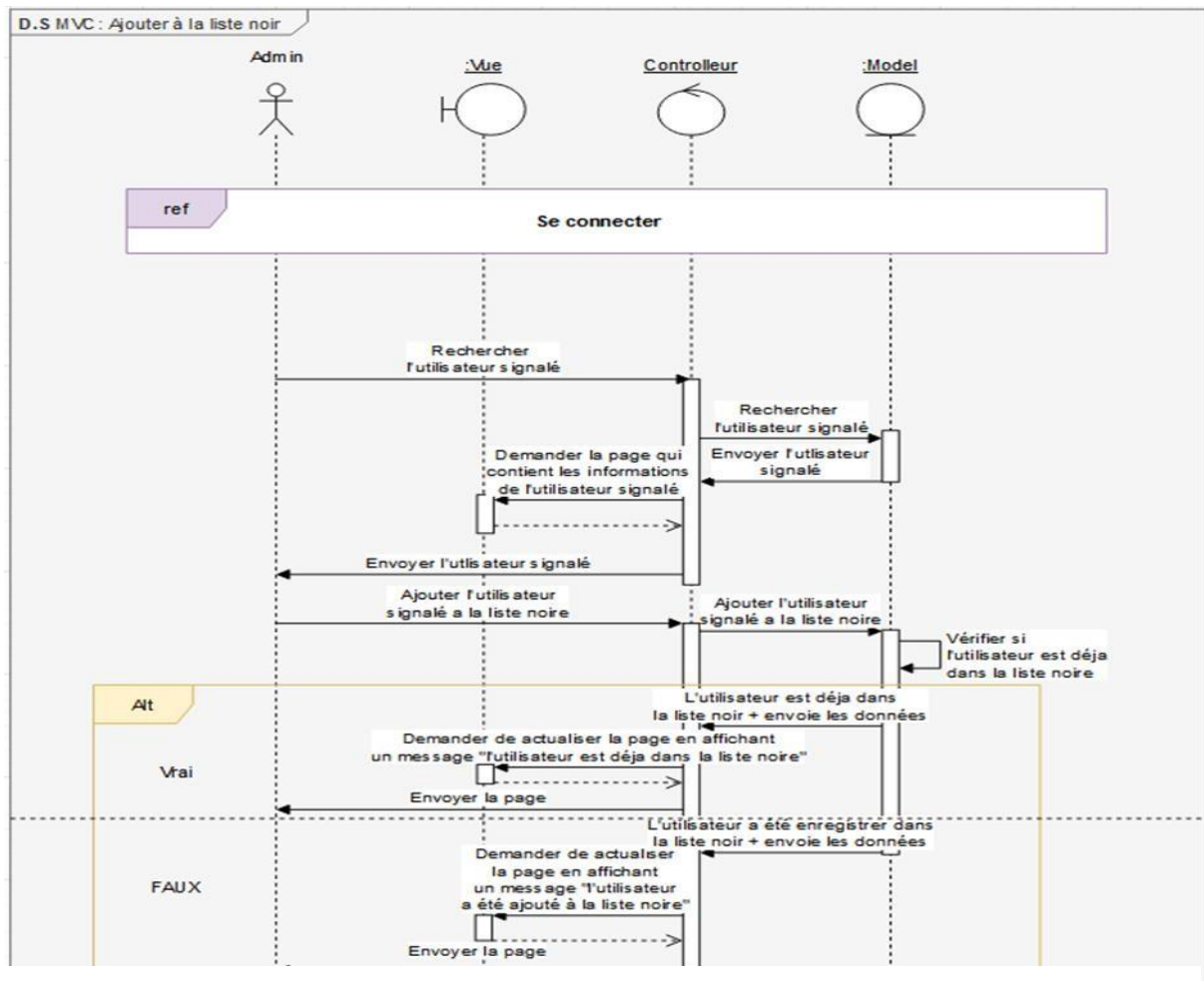


Figure 10 - Diagramme de séquence MVC « Ajouter à la liste noire »

Pour l'administrateur, le processus commence par se connecter à l'interface d'administration. Une fois connecté, l'administrateur demande au contrôleur de rechercher l'utilisateur signalé. Le contrôleur transmet cette demande au modèle, qui renvoie les informations de l'utilisateur signalé au contrôleur. Ce dernier demande alors à la vue de générer une page contenant les informations de l'utilisateur signalé, qu'il envoie ensuite à l'administrateur.

L'administrateur décide ensuite d'ajouter l'utilisateur signalé à la liste noire. Le contrôleur reçoit cette demande et interroge le modèle pour vérifier si l'utilisateur est déjà présent dans la liste noire. Si c'est le cas, le contrôleur demande à la vue d'afficher un message indiquant que l'utilisateur est déjà dans la liste noire, puis envoie la page actualisée à l'administrateur. Sinon, le modèle est chargé d'enregistrer l'utilisateur dans la liste noire, puis le contrôleur demande à la vue d'afficher un message indiquant que l'utilisateur a été ajouté avec succès à la liste noire. Enfin, le contrôleur envoie la page actualisée à l'administrateur.

2.6.2 Diagramme de séquence MVC du cas « Ajouter à la liste des favoris »

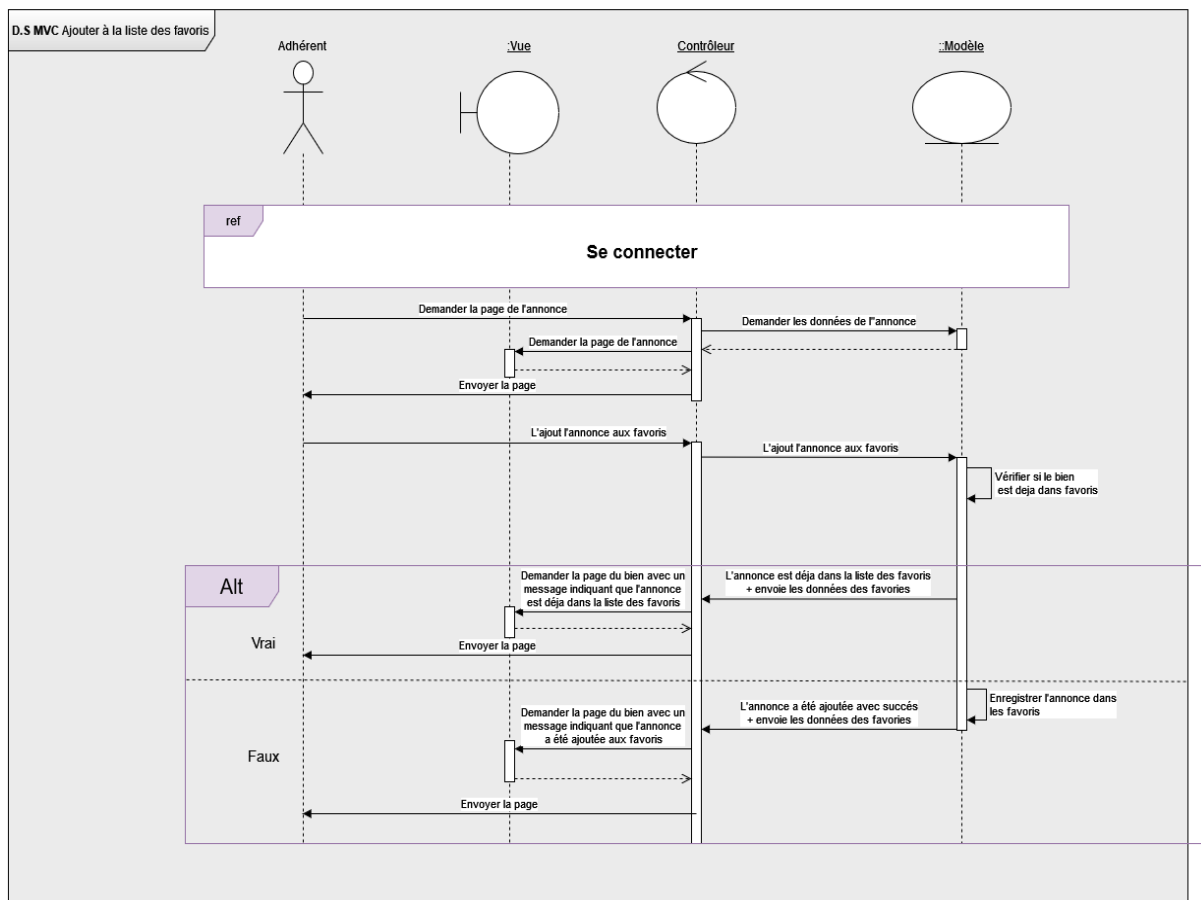


Figure 11 - Diagramme de séquence MVC pour «Ajouter à la liste des favoris »

Pour l'adhérent, le processus débute par la connexion à l'interface. Une fois connecté, l'adhérent demande au contrôleur la page de l'annonce souhaitée. Le contrôleur, à son tour, sollicite le modèle pour obtenir les données de l'annonce. Après réception, le contrôleur envoie la page contenant les détails de l'annonce à l'adhérent.

L'adhérent souhaite ensuite ajouter l'annonce à ses favoris. Il demande cette action au contrôleur, qui la transmet au modèle. Le modèle vérifie alors si l'annonce est déjà dans la liste des favoris. Si tel est le cas, il informe le contrôleur que l'annonce est déjà dans la liste. Le contrôleur demande alors à la vue d'afficher la page de l'annonce avec un message indiquant que l'annonce est déjà dans les favoris, puis envoie la page à l'adhérent.

Si l'annonce n'est pas déjà dans les favoris, le modèle enregistre l'annonce dans la liste des favoris. Il informe ensuite le contrôleur que l'ajout a été effectué avec succès. Le contrôleur

demande alors à la vue d'afficher la page de l'annonce avec un message indiquant que l'annonce a été ajoutée aux favoris, puis envoie la page à l'adhérent.

2.6.3 Diagramme de séquence MVC du cas Supprimer de la liste des favoris

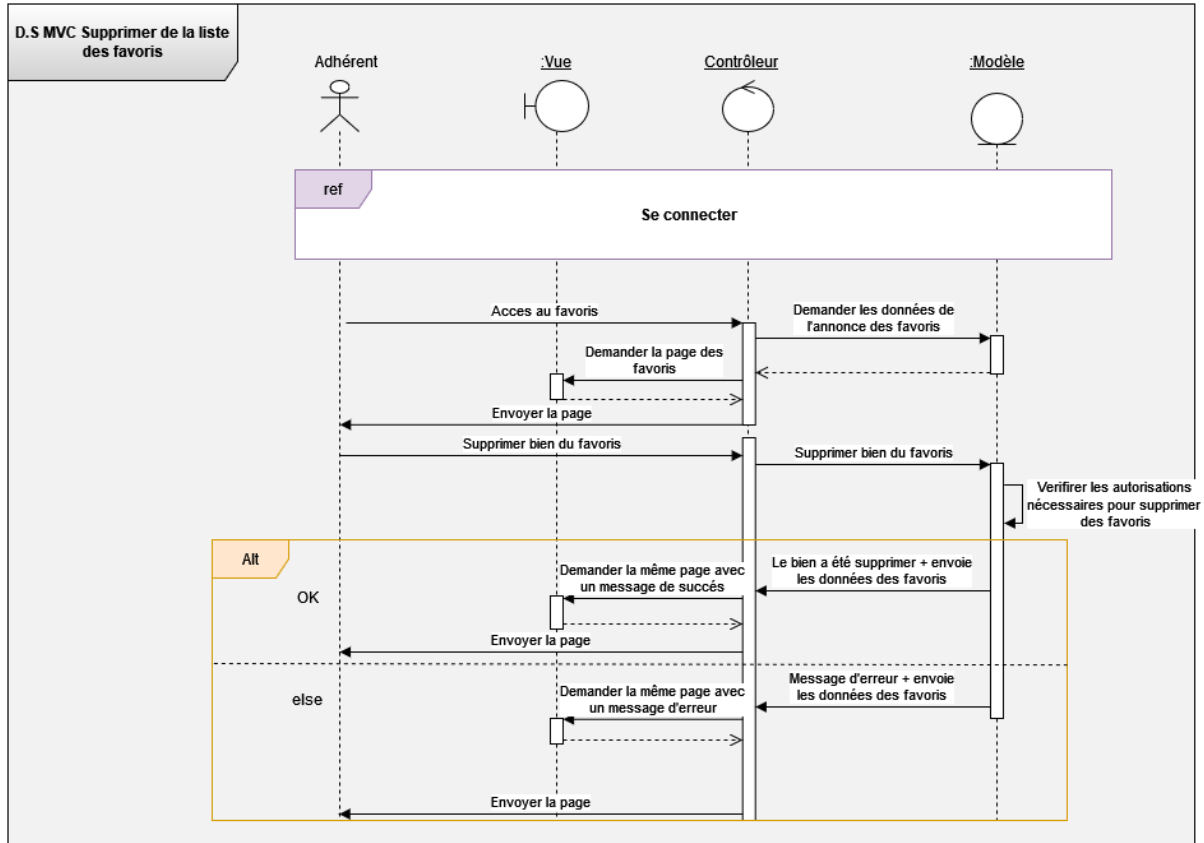


Figure 12 - Diagramme de séquence MVC pour « supprimer liste des favoris »

Pour l'adhérent, le processus commence par la connexion à l'interface. Une fois connecté, l'adhérent demande au contrôleur l'accès à ses favoris. Le contrôleur sollicite alors le modèle pour obtenir les données des annonces enregistrées dans les favoris. Ensuite, le contrôleur fournit la page des favoris à la vue, qui la transmet à l'adhérent.

L'adhérent souhaite ensuite supprimer un bien de ses favoris. Il demande cette action au contrôleur, qui procède à la suppression du bien des favoris dans le modèle. Le modèle effectue les vérifications nécessaires pour s'assurer que la suppression peut être effectuée. Si le bien est supprimé avec succès, le contrôleur fournit à la vue la page des favoris avec un message de succès, puis envoie la page à l'adhérent.

Si la suppression du bien n'est pas possible, le modèle envoie un message d'erreur au contrôleur. Ce dernier transmet alors ce message à la vue, qui affiche la page des favoris avec un message d'erreur. Enfin, le contrôleur envoie la page à l'adhérent.

2.6.4 Diagramme de séquence MVC du cas Supprimer une annonce

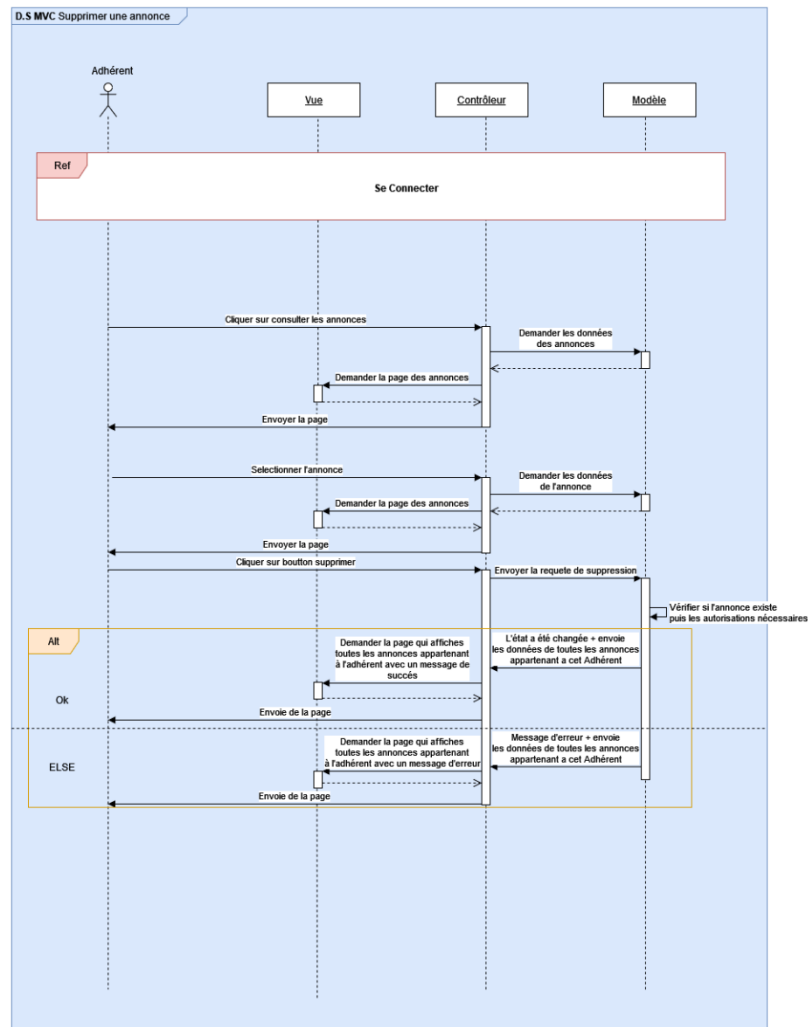


Figure 13 - Diagramme de séquence MVC pour «Supprimer une annonce»

Pour l'adhérent, le processus commence par la connexion à l'interface. Une fois connecté, l'adhérent accède à la liste des annonces en cliquant sur "Consulter les annonces". Le contrôleur demande alors au modèle les annonces disponibles, le modèle ensuite renvoie les annonces, si c'est le cas, le contrôleur fournit la page des annonces à la vue, qui la transmet à l'adhérent.

L'adhérent sélectionne ensuite l'annonce à supprimer et demande au contrôleur de la supprimer. Le contrôleur envoie la demande de suppression au modèle, qui vérifie les

autorisations nécessaires pour supprimer l'annonce. Si la suppression est autorisée, le modèle supprime l'annonce et informe le contrôleur. Ce dernier fournit alors à la vue la page des annonces avec un message de succès, puis envoie la page à l'adhérent.

Si la suppression de l'annonce n'est pas autorisée, le modèle informe le contrôleur par un message d'échec. Celui-ci transmet alors ce message à la vue, qui affiche la page des annonces avec un message d'erreur. Enfin, le contrôleur envoie la page adéquate à l'adhérent.

2.6.5 Diagramme de séquence MVC du cas d'utilisation Créer un compte

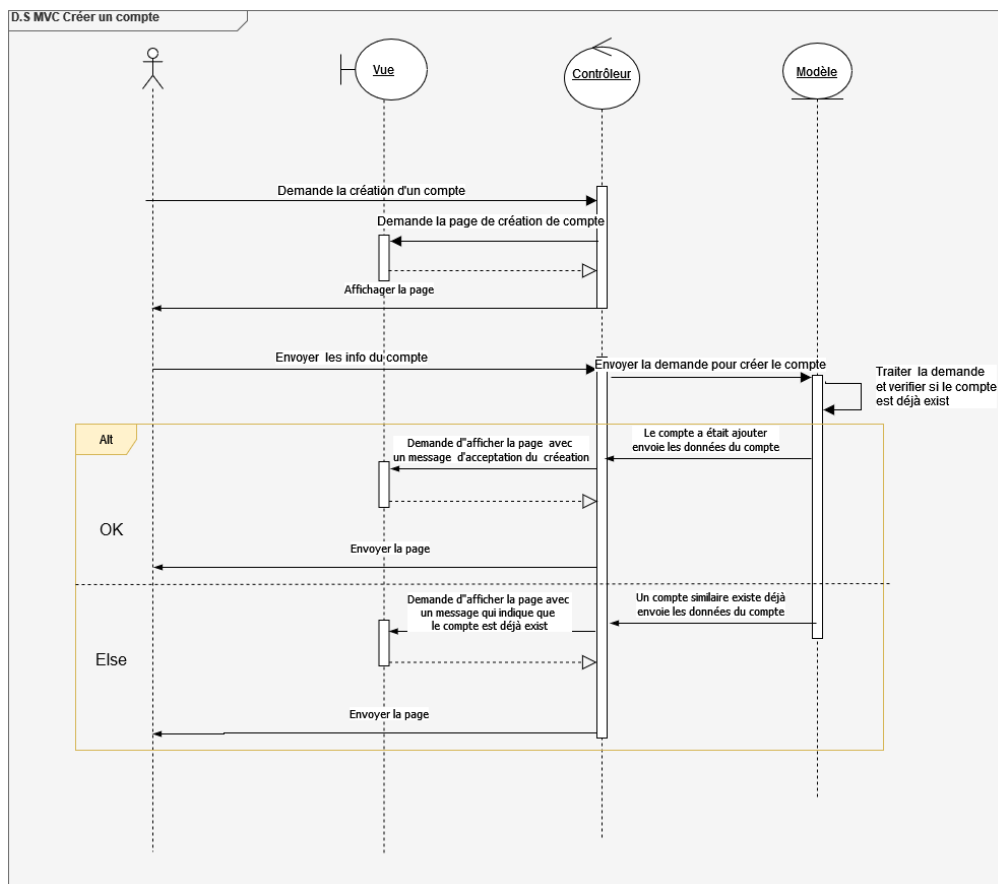


Figure 14 - Diagramme de séquence MVC «Créer un compte»

Lorsqu'un utilisateur souhaite créer un compte, le contrôleur de création de compte est sollicité. Ce contrôleur, à son tour, demande à la vue d'afficher la page de création de compte. Une fois affichée, la page est présentée au visiteur qui fournit les informations nécessaires pour le compte. Ces informations sont ensuite transmises au contrôleur.

Le contrôleur traite alors la demande en envoyant une requête au modèle pour la création du compte. Il vérifie également si le compte existe déjà. Si le compte est créé avec succès, les données du compte sont renvoyées au contrôleur. Cette dernière demande alors à la vue

d'afficher une page confirmant la création du compte, incluant un message d'acceptation. Cette page est ensuite envoyée au visiteur.

En revanche, si le compte existe déjà, le contrôleur informe la vue de lui fournir une page contenant un message indiquant que le compte existe déjà. Par la suite le contrôleur affiche cette page au visiteur.

2.6.6 Diagramme de séquence MVC du cas d'utilisation Ajouter Avis

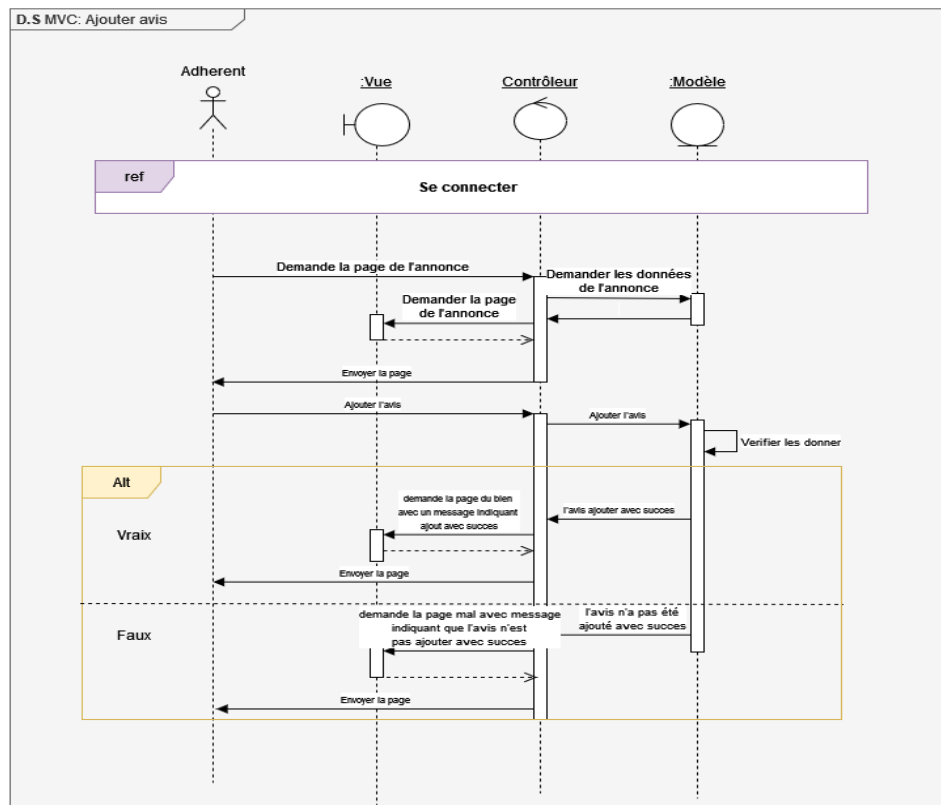


Figure 15 - Diagramme de séquence MVC «Ajouter Avis»

Pour le cas d'utilisation Ajouter un avis, l'adhérent doit d'abord se connecter. Une fois connecté, l'adhérent demande à la vue d'afficher la page des annonces au contrôleur. Le contrôleur, à son tour, sollicite le modèle pour obtenir les données nécessaires.

Une fois les données récupérées, le modèle les transmet au contrôleur qui les renvoie à l'adhérent sous forme de page. L'adhérent commence alors à ajouter son avis. Une fois l'avis a été ajouté, le contrôleur vérifie les données en les envoyant au modèle.

Si l'avis est ajouté avec succès, le contrôleur demande à la vue d'afficher la page du bien avec un message de succès. Cette page est ensuite envoyée par le contrôleur à l'adhérent.

En revanche, si l'avis n'est pas ajouté avec succès, le contrôleur demande à la vue d'afficher la page avec un message indiquant que l'ajout de l'avis a échoué. Cette page est également envoyée par le contrôleur à l'adhérent.

2.6.7 Diagramme de séquence MVC du cas Consulter les biens les plus vus

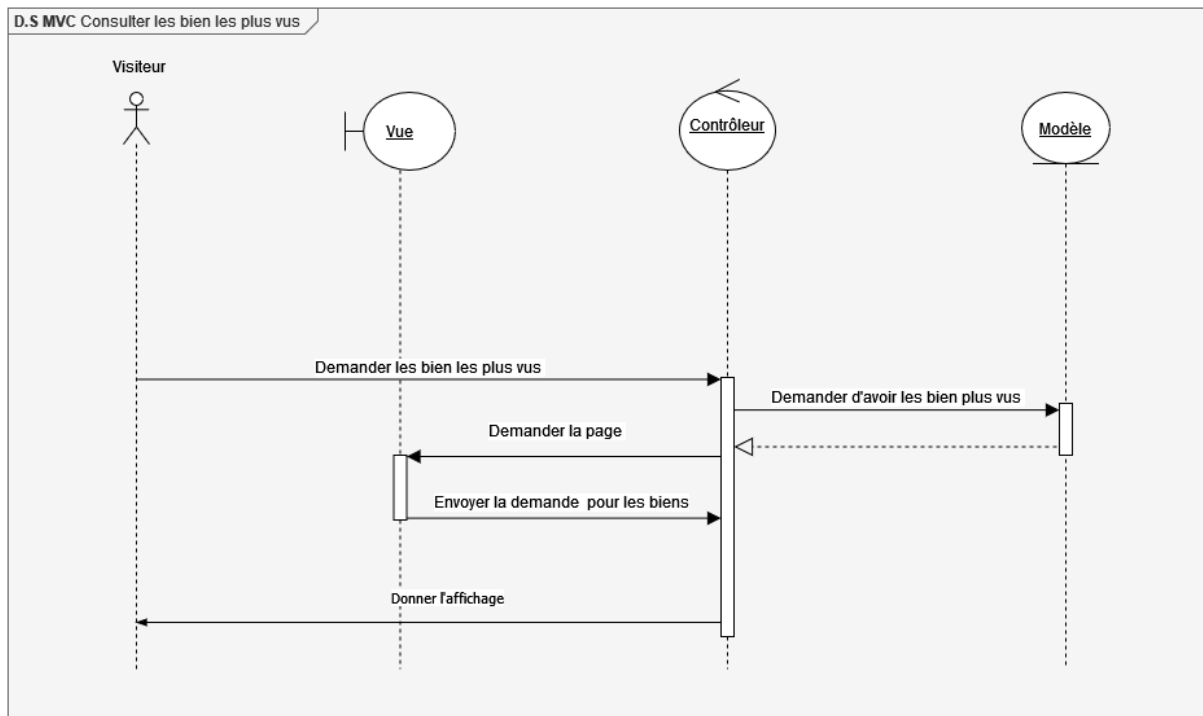


Figure 16 - Diagramme de séquence MVC du cas «Consulter les biens les plus vus»

Dans ce cas d'utilisation, un visiteur demande au contrôleur de lui fournir les biens les plus vus. Le contrôleur transmet alors cette demande au modèle pour récupérer les informations pertinentes. Une fois que le modèle a répondu avec les données des biens les plus vus, le contrôleur demande à la vue de lui fournir la page adéquate et par la suite le contrôleur la transmet au visiteur.

2.6.8 Diagramme de séquence MVC du cas Rechercher des biens + Consulter les biens

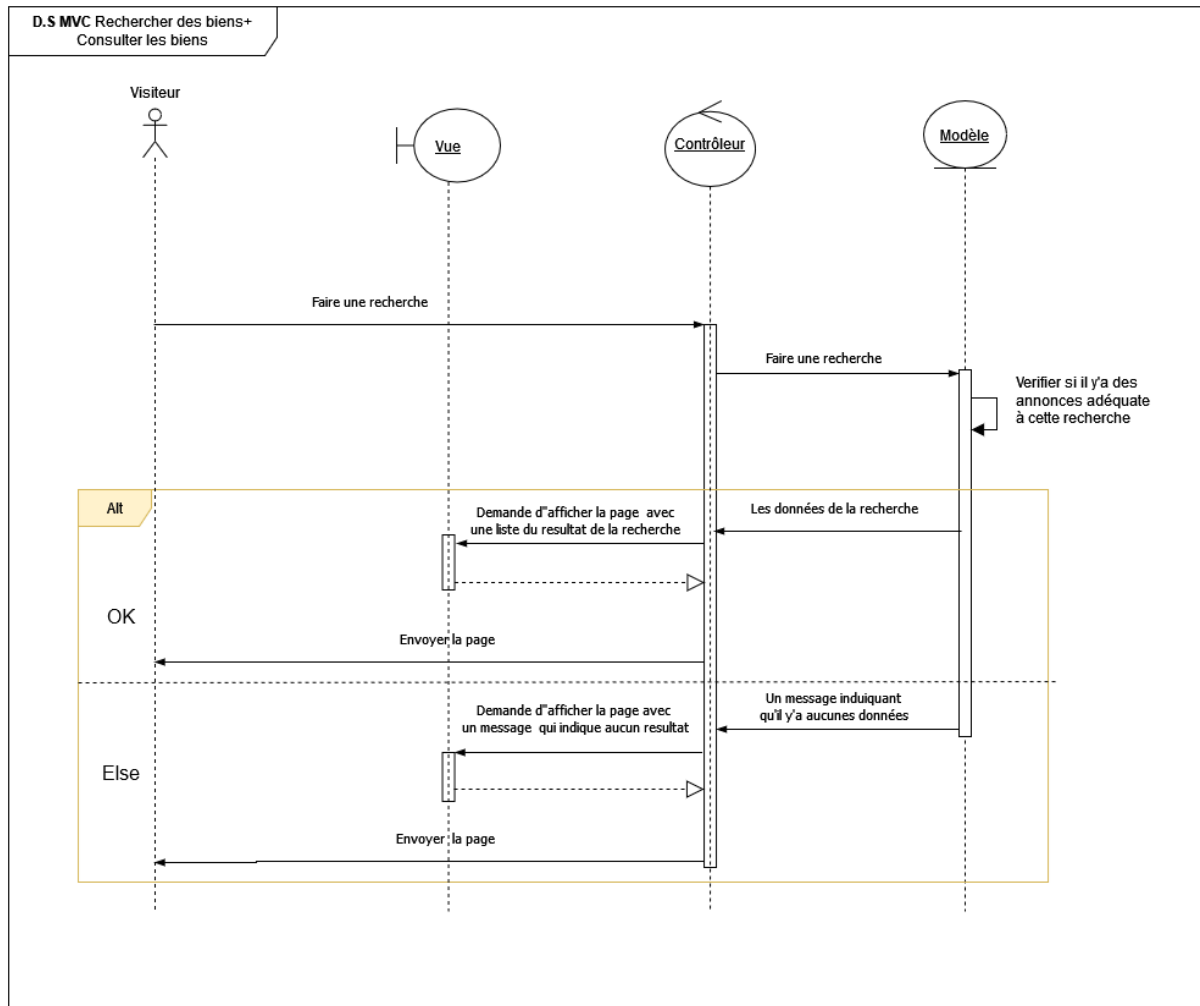


Figure 17 - Diagramme de séquence du cas «Rechercher les biens» + «Consulter les biens»

Dans ce cas d'utilisation, un visiteur demande au contrôleur d'effectuer une recherche de biens. Le contrôleur transmet cette demande au modèle pour vérifier si les annonces adéquate à cette recherche existent. Le modèle répond alors au contrôleur avec l'envoi de données. Si les annonces adéquate sont trouvées, le contrôleur demande à la vue d'afficher une page avec la liste des résultats. La vue répond au contrôleur, qui envoie ensuite la page au visiteur. Si aucun résultat n'est pas trouvé, le contrôleur demande à la vue d'afficher une page avec un message indiquant qu'aucun résultat n'a été trouvé.

2.7 Architecture Système

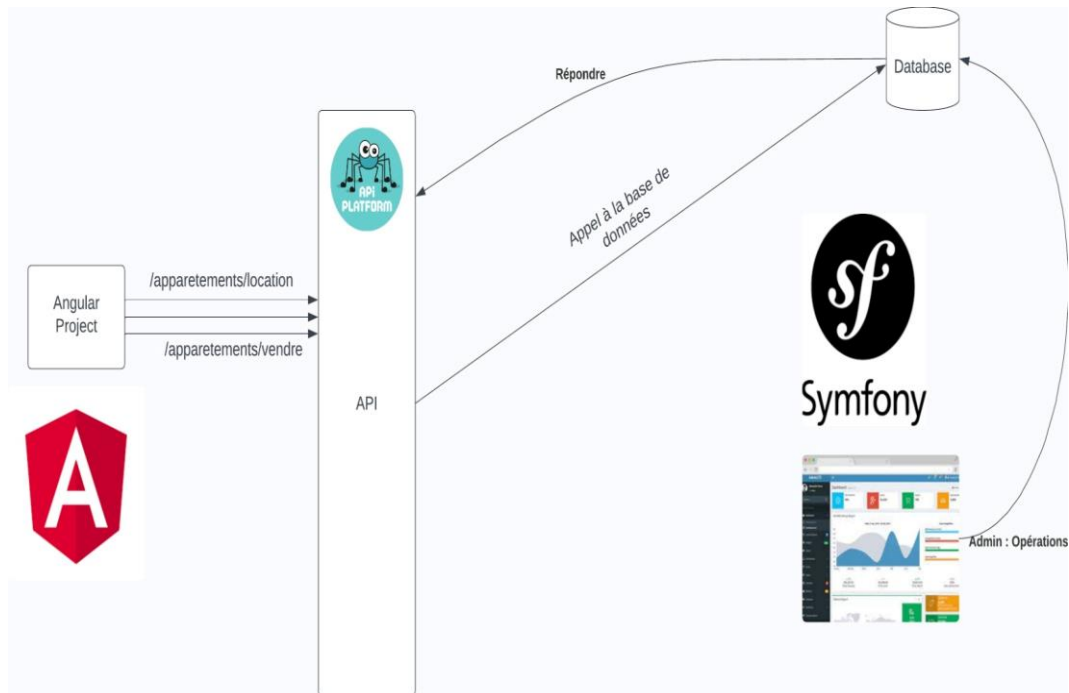
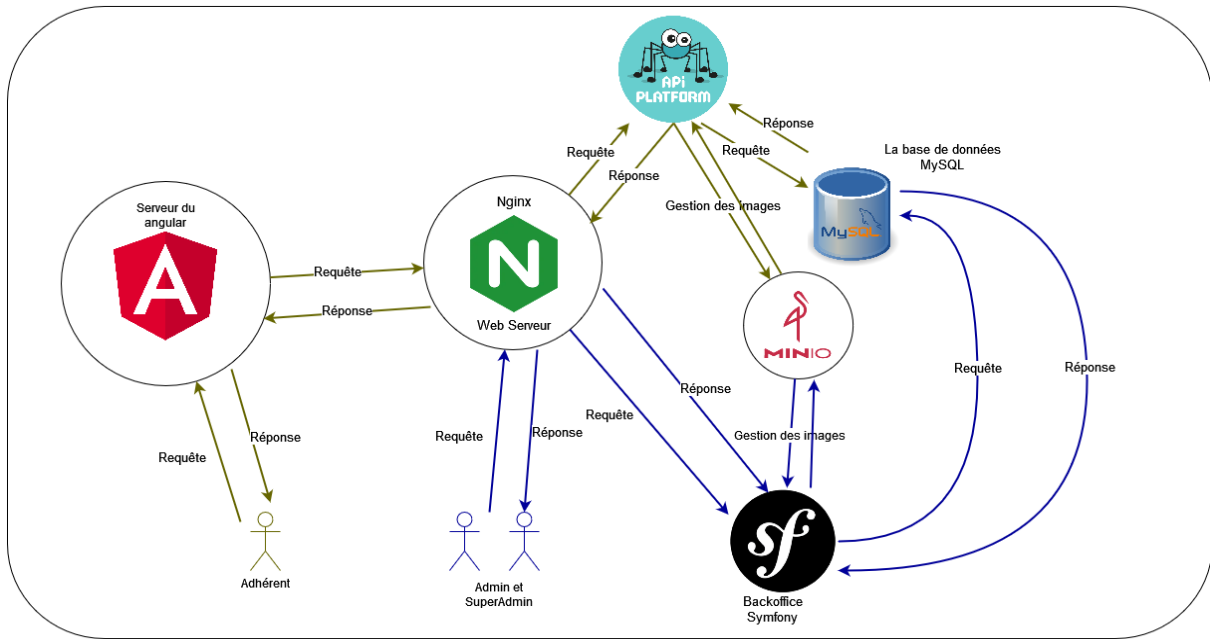


Figure 18 - Schéma Architecture Système

Dans l'image présentée, nous observons la partie frontale du système (partie client), développée avec Angular qui interagit avec l'API (développé avec API Platform). L'API traite ces requêtes et communique avec la base de données pour récupérer les informations nécessaires. Les réponses de la base de données, quelle que soit leur nature, sont ensuite renvoyées à l'API. De plus, la partie back-office, conçue avec le Framework Symfony, est chargée de la gestion des utilisateurs et des opérations administratives.

Voici une architecture plus détaillée concrète de notre projet :



2.8 Conclusion

Dans ce chapitre, on a présenté les différents diagrammes réalisés par notre groupe tels que le diagramme de cas d'utilisation, le diagramme de classe, les diagrammes de séquences et les diagrammes de séquences MVC afin d'arriver à un plan bien déterminé et clair. Le chapitre suivant sera consacré à la réalisation de l'application en utilisant les différents outils.

Chapitre 3 : Implémentation et réalisation

3.1 Introduction

La phase de réalisation ou mise en œuvre est la dernière phase du processus du développement, Elle comporte le codage et le test du système. Le présent chapitre a pour but la description de la phase de réalisation de la solution. Dans un premier temps, on présente les outils utilisés durant le développement pour passer, ensuite, à la réalisation effective de site web.

3.2 Environnement de développement

3.2.1 Environnement matériel

Pour le développement de notre application nous avons utilisé 5 PC portable « HP » dont la Configuration est la suivante :

	PC1	PC2	PC3	PC4	PC5
Quantité de mémoire vive	16 GO	32 GO	16 GO	16 GO	32GO
Capacité de disque	256GO	512GO	2TO	256GO	1TO
Processeur Intel Core	11th Gen Intel(R) Core(TM) i7-1185G7 - 3.00 GHz (8 CPUs) ~ 3.00 GHz	AMD Ryzen 5 PRO 4650U with Radeon Graphics 2.10 GHz	6 Gen Intel Core i(5-6400- 3.00 GHz (6 CPUs) ~ 3.00 GHz	6 Gen Intel Core i(5-6400- 3.00 GHz (6 CPUs) ~ 3.00 GHz	6 Gen Intel Core i(5-6400- 2.50 GHz (6 CPUs) ~ 0 G2.50 Hz

3.2.2 Architecture utilisée

Nous avons développé la partie web en se basant sur un patron de conception MVC (Modèle, Vue, Contrôleur) comme l'illustre la figure.

MVC est un modèle d'organisation du code telle que :

- Le modèle est chargé de gérer les données.
- La vue est chargée de la mise en forme pour l'utilisateur.
- Le contrôleur est chargé de gérer l'ensemble.

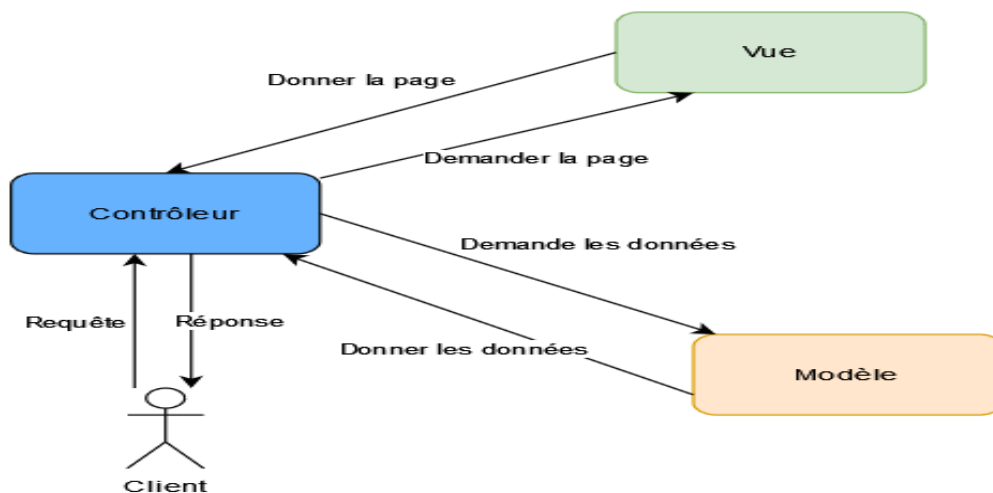


Figure 19 - Architecture MVC

Le modèle MVC décrit une manière d'architecturer une application informatique en la décomposant en trois sous-parties :

La partie Modèle : La couche Model représente la partie de l'application qui exécute la logique métier. Cela signifie qu'elle est responsable de récupérer les données, de les convertir selon des concepts chargés de sens pour votre application, tels que le traitement, la validation, l'association et beaucoup d'autres tâches concernant la manipulation des données.

La partie Vue : La Vue retourne une présentation des données venant du model. Etant séparée par les Objets Modèle , elle est responsable de l'utilisation des informations dont elle dispose pour produire une interface de présentation de votre application. Par exemple, de la même manière que la couche Modèle retourne un ensemble de données, la Vue utilise ces données pour fournir une page HTML qui les contenant.

La partie Contrôleur : La couche Contrôleur gère les requêtes des utilisateurs. Elle est responsable de retourner une réponse avec l'aide mutuelle des couches Modèle et Vue.

3.3 Pourquoi utiliser un Framework.

Un Framework est un ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture et des patterns, l'ensemble formant ou promouvant un « squelette » de programme. Il est souvent fourni sous la forme d'une bibliothèque logicielle, et accompagné du plan de l'architecture cible du Framework.

Les avantages des Framework sont nombreux. En effet, un Framework est portable, de la part de son abstraction de la base de données et de la gestion générique du cache. Les temps de développement avec un Framework sont réellement plus courts.

Tous les outils essentiels sont déjà écrits. Le développement des applications sécurisées est facile.

Les Frameworks sont des outils communautaires. De plus vu que les Frameworks sont largement déployés, la chance de trouver les correctifs des problèmes rencontrés est plus grande.

3.4 Environnement logiciel

3.4.1 Adobe Photoshop



Figure 20 - Adobe Illustrator

Adobe Illustrator est un logiciel de création graphique vectorielle. Il fait partie de la gamme Adobe, peut être utilisé indépendamment ou en complément de Photoshop, et offre des outils de dessin vectoriel puissants. Les images vectorielles sont constituées de courbes générées par des formules mathématiques. L'un des outils principaux d'Illustrator étant « la plume » qui permet de tracer des courbes à l'aspect parfait grâce au placement de points d'ancrage et de tangentes qui vont en modifier la courbure. Un des avantages des images vectorielles est qu'elles sont indépendantes de la résolution, c'est-à-dire qu'elles ne perdent pas en qualité lorsqu'on les agrandit. Adapté aussi bien à la création de document papier qu'à celle d'illustrations pour Internet (logos, affiches, etc.) ce logiciel est orienté vers le marché professionnel, il intègre de nombreuses options propres à améliorer la productivité.

3.4.2 Git et GitHub



Figure 21 - Logo Git



Figure 22 - Logo GitHub

Définition de Git :

Git est un logiciel libre qui a été créé en 2005 par le créateur de Linux, Linus Torvalds. Au départ, il a créé ce logiciel de gestion de version pour gérer les sources de son noyau open source.

Il s'agit ainsi d'un logiciel de versioning qui permet de conserver un historique des modifications. Ce type de logiciel est indispensable, car il permet de gérer facilement les modifications effectuées sur un projet. En utilisant Git, le professionnel de la data ou du développement web va pouvoir identifier rapidement les changements effectués et revenir, si besoin, à une ancienne version. Git fait partie de la famille des logiciels de versioning dits décentralisés, car chaque développeur va disposer d'une copie de l'historique de son code source.

Définition de GitHub :

Il s'agit du plus grand hébergeur de dépôts Git au monde. GitHub est étroitement lié à Git. En effet, comme Git est un logiciel de gestion de version, il va impliquer un enregistrement des différentes modifications effectuées sur le projet. C'est de cette manière que les professionnels vont pouvoir retourner à une version précédente.

3.4.3 Draw.io

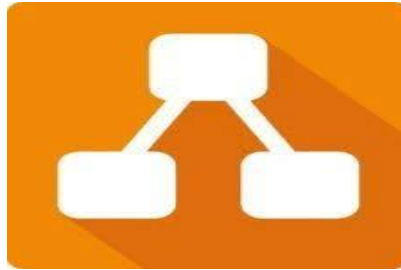


Figure 23 – Logo de Draw.io

Draw.io est un logiciel de diagramme en ligne simple et puissant qui permet de créer facilement des diagrammes, des organigrammes, des schémas et bien plus encore. Avec une interface intuitive et conviviale, Draw.io offre une large gamme d'outils de dessin et de formes prédéfinies pour vous aider à donner vie à vos idées visuelles. Que vous soyez un professionnel à la recherche d'un outil de visualisation de données ou un étudiant qui souhaite créer des présentations claires et attrayantes, Draw.io vous permet de créer des diagrammes de qualité professionnelle sans effort. Grâce à son réel avec d'autres utilisateurs et travailler ensemble sur des projets. Avec Draw.io, vous disposez de tous les outils nécessaires pour donner vie à vos concepts et communiquer efficacement vos idées visuelles. Compatibilité avec diverses plateformes et sa facilité de partage, vous pouvez collaborer en temps.

3.4.4 Visual Studio Code:

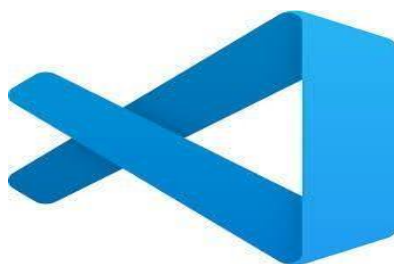


Figure 24 - Logo Visual Studio Code

Visual Studio Code (VS Code) est un éditeur de code source gratuit et open-source développé par Microsoft, largement utilisé pour la rédaction et le développement de code dans divers langages de programmation. Il offre une interface utilisateur intuitive et des fonctionnalités puissantes telles que la coloration syntaxique, l'autocomplétions, et le débogage intégré. Grâce à sa bibliothèque d'extensions, Visual Studio Code peut être facilement

personnalisé pour répondre aux besoins spécifiques des développeurs. Ce logiciel a été choisi pour ce projet en raison de sa flexibilité, de sa facilité d'utilisation, et de son large support communautaire.

3.4.5 Discord



Figure 25 - logo Discord

Discord est une plateforme permettant aux personnes ayant des intérêts similaires de partager et de communiquer. Il est populaire parmi la communauté des joueurs car il offre aux joueurs de jeux vidéo un moyen de communiquer entre eux et de développer une communauté en dehors des jeux eux-mêmes.

3.4.6 MySQL



Figure 26 - Logo MySQL

MySQL est un système de gestion de base de données relationnelle open-source largement utilisé dans le domaine des technologies de l'information. Il offre une solution robuste et flexible pour la création et la manipulation de bases de données, utilisant le langage de requête SQL pour interagir avec les données de manière efficace. MySQL a été choisi comme système de gestion de base de données pour ce projet en raison de sa fiabilité, de sa performance et de sa compatibilité avec les besoins spécifiques de l'application. Son architecture ouverte et son

large écosystème de support en font un choix populaire pour une variété d'applications, allant des petites applications web aux systèmes d'entreprise complexes.

3.4.7 Docker

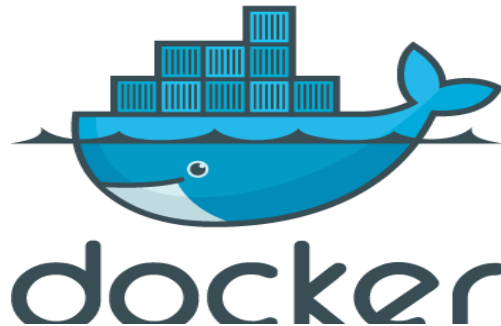


Figure 27 - Logo Docker

Docker est une plateforme open-source utilisée pour automatiser le déploiement, la gestion et l'exécution d'applications au sein de conteneurs logiciels. Les conteneurs créés avec Docker sont des environnements légers et portables qui contiennent toutes les dépendances nécessaires pour que l'application fonctionne de manière uniforme sur divers systèmes. Grâce à Docker, les développeurs peuvent créer, tester et déployer des applications plus rapidement et de manière fiable, indépendamment des spécificités de l'infrastructure sous-jacente. Cette technologie a été choisie pour ce projet en raison de sa capacité à simplifier les processus de développement et de déploiement, assurant ainsi une meilleure efficacité et une cohérence accrue des environnements de production.

3.4.8 Doctrine



Figure 28 - Logo Doctrine

Doctrine ORM est un mappeur Object-Relational (ORM) pour PHP qui fournit une persistance transparente pour les objets PHP. Il utilise le Data Mapper pattern au cœur, visant à une séparation complète de votre domaine/entreprise logique de la persistance dans un système de gestion de base de données relationnelle.

L'avantage de la doctrine pour le programmeur est la capacité de se concentrer sur la logique commerciale orientée vers l'objet et ne s'inquiéter de la persévérance que en tant que problème secondaire. Cela ne veut pas dire que la persistance est minimisée par la Doctrine 2, nous pensons toutefois qu'il y a des avantages considérables pour programmation orientée objet si la persistance et les entités sont maintenues séparés.

3.4.9 Nginx



Figure 29 - Logo Nginx

Nginx est un logiciel serveur web open-source utilisé pour héberger des sites web et gérer le trafic internet. Il est connu par sa rapidité et aussi il est capable de gérer beaucoup de connexions en même temps.

3.5 Langage et Framework utiliser

3.5.1 Définition PHP



Figure 30 - Logo PHP

Le PHP, pour Hypertext Preprocessor, désigne un langage informatique, ou un langage de script, utilisé principalement pour la conception de sites web dynamiques. Il s'agit d'un langage de programmation sous licence libre qui peut donc être utilisé par n'importe qui de façon totalement gratuite.

Créé au début des années 1990 par le Canadien et Groenlandais Rasmus Lerdorf, le langage PHP est souvent associé au serveur de base de données MySQL et au serveur Apache. Avec le système d'exploitation Linux, il fait partie intégrante de la suite de logiciels libres LAMP.

Sur un plan technique, le PHP s'utilise la plupart du temps côté serveur. Il génère du code HTML, CSS ou encore XHTML, des données (en PNG, JPG, etc.) ou encore des fichiers PDF. Il fait, depuis de nombreuses années, l'objet d'un développement spécifique et jouit aujourd'hui une bonne réputation en matière de fiabilité et de performances.

3.5.2 Définition Symfony



Figure 31 - Logo Symfony

Symfony est un Framework de développement d'applications web open source et performant écrit en PHP. Il fournit un ensemble d'outils et de composants permettant aux

développeurs de créer efficacement des applications web robustes et évolutives. Symfony suit le paradigme de conception Modèle-Vue-Contrôleur (MVC) et se concentre sur la réutilisation de code, la modularité et la maintenabilité du projet. De plus, il offre des fonctionnalités telles que la gestion des sessions, un routage flexible, la manipulation des formulaires, la sécurité intégrée et le support des bases de données. Symfony est largement utilisé dans la communauté du développement web en raison de sa flexibilité, de sa documentation détaillée et de sa communauté active.

3.5.3 Définition de Angular



Figure 32 - Logo Angular

Angular est un Framework côté client, open source, basé sur Type Script, et co-dirigé par l'équipe du projet « Angular » à Google et par une communauté de particuliers et de sociétés.

Angular est une réécriture complète d'AngularJS, cadriciel construit par la même équipe. Il permet la création d'applications Web et plus particulièrement de ce qu'on appelle des « Single Page Applications » : des applications web accessibles via une page web unique qui permet de fluidifier l'expérience utilisateur et d'éviter les chargements de pages à chaque nouvelle action.

Le Framework est basé sur une architecture du type MVC et permet donc de séparer les données, le visuel et les actions pour une meilleure gestion des responsabilités. Un type d'architecture qui a largement fait ses preuves et qui permet une forte maintenabilité et une amélioration du travail collaboratif.

3.5.4 Définition Api Platform

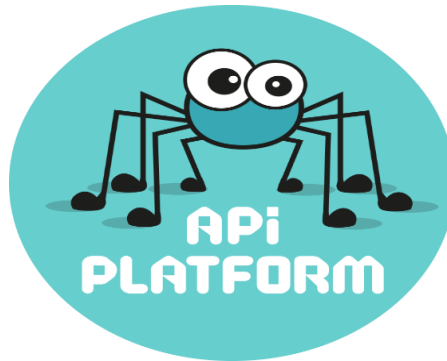


Figure 33 - Logo ApiPlatform

Une API (Application Programming Interface) est un ensemble de fonctions et de procédures qui permet à des applications de communiquer entre elles et d'échanger des données. Concrètement, les API définissent la façon dont deux logiciels peuvent interagir.

Une API Platform se distingue par sa capacité à simplifier le processus de développement et de gestion des API. Il propose une approche centrée sur le développeur, en fournissant des outils intuitifs et flexibles qui accélèrent la mise en place des API tout en assurant leur fiabilité et leur performance. Cela inclut une interface utilisateur simplifiée, des schémas de données automatiques, et une documentation API intégrée.

Ce Framework est conçu pour être polyvalent et s'adapte aux besoins spécifiques des projets, que ce soit pour créer des API REST, GraphQL ou même des applications hybrides. Il encourage également les

meilleures pratiques de l'industrie et respecte les normes et standards modernes, ce qui en fait un choix

privilégié pour les développeurs soucieux de la qualité et de l'évolutivité de leurs projets.

Les API jouent un rôle essentiel dans l'intégration et la connexion entre différents systèmes et applications. Elles permettent à des services web et applications de ne pas avoir à connaître les détails de mise en œuvre interne de chacun.

Grâce aux API, il est possible de faire communiquer facilement et rapidement des applications qui n'ont pas été conçues pour fonctionner ensemble à l'origine. Cela ouvre un monde de possibilités en termes d'innovation et de création de services.

3.5.5 Définition MinIO



Figure 34: logo Minio

MinIO est un magasin d'objets hautes performances compatible S3. Il est conçu pour les charges de travail d'IA/ML, de lacs de données et de bases de données à grande échelle. Il est défini par logiciel et s'exécute sur n'importe quelle infrastructure cloud ou sur site. MinIO est sous double licence sous open source GNU AGPL v3 et une licence d'entreprise commerciale.

3.5.6 Définition de JavaScript



Figure 35 - Logo JavaScript

JavaScript est un langage de programmation interprété et dynamique, essentiel dans le développement web pour rendre les pages interactives et dynamiques. Utilisé en conjonction avec HTML et CSS, JavaScript permet de manipuler les éléments du Document Object Model (DOM) pour créer des effets interactifs, valider des formulaires, et gérer des événements utilisateurs. En outre, grâce à des environnements tels que Node.js, JavaScript est également employé pour le développement côté serveur, faisant de lui un langage polyvalent pour le

développement full-stack. JavaScript a été choisi pour ce projet en raison de sa capacité à améliorer l'interactivité de l'application web et de sa large adoption dans l'industrie.

3.5.7 Définition de TypeScript



Figure 36 - Logo TypeScript

TypeScript est un langage de programmation fortement typé qui s'appuie sur JavaScript, vous offrant de meilleurs outils à toute échelle.

3.5.8 Définition de CSS



Figure 37 - Logo CSS

CSS (Cascading Style Sheets) est un langage de feuille de style essentiel pour la conception et la mise en forme des documents HTML. Il permet de définir et de contrôler l'apparence des éléments sur une page web, en spécifiant des styles pour la mise en page, les couleurs, les polices, et les espacements. En séparant le contenu HTML de sa présentation visuelle, CSS facilite la maintenance et l'évolution des sites web. De plus, CSS permet de créer des mises en page réactives qui s'adaptent à différents appareils et tailles d'écran, améliorant

ainsi l'expérience utilisateur. Pour ce projet, CSS a été utilisé pour garantir une interface utilisateur esthétique et cohérente à travers diverses plateformes.

3.5.9 Définition HTML



Figure 38 - Logo html

HTML signifie « HyperText Markup Language » qu'on peut traduire par « langage de balises pour l'hypertexte ». Il est utilisé afin de créer et de représenter le contenu d'une page web et sa structure. D'autres technologies sont utilisées avec HTML pour décrire la présentation d'une page (CSS) et/ou ses fonctionnalités interactives (JavaScript).

L'« hypertexte » désigne les liens qui relient les pages web entre elles, que ce soit au sein d'un même site web ou entre différents sites web. Les liens sont un aspect fondamental du Web. Ce sont eux qui forment cette « toile » (ce mot est traduit par web en anglais). En téléchargeant du contenu sur l'Internet et en le reliant à des pages créées par d'autres personnes, vous devenez un participant actif du World Wide Web.

3.3 Les principales interfaces du Front End (visiteur et adhérent)

3.3.1 Interface principale pour visiteur et adhérent

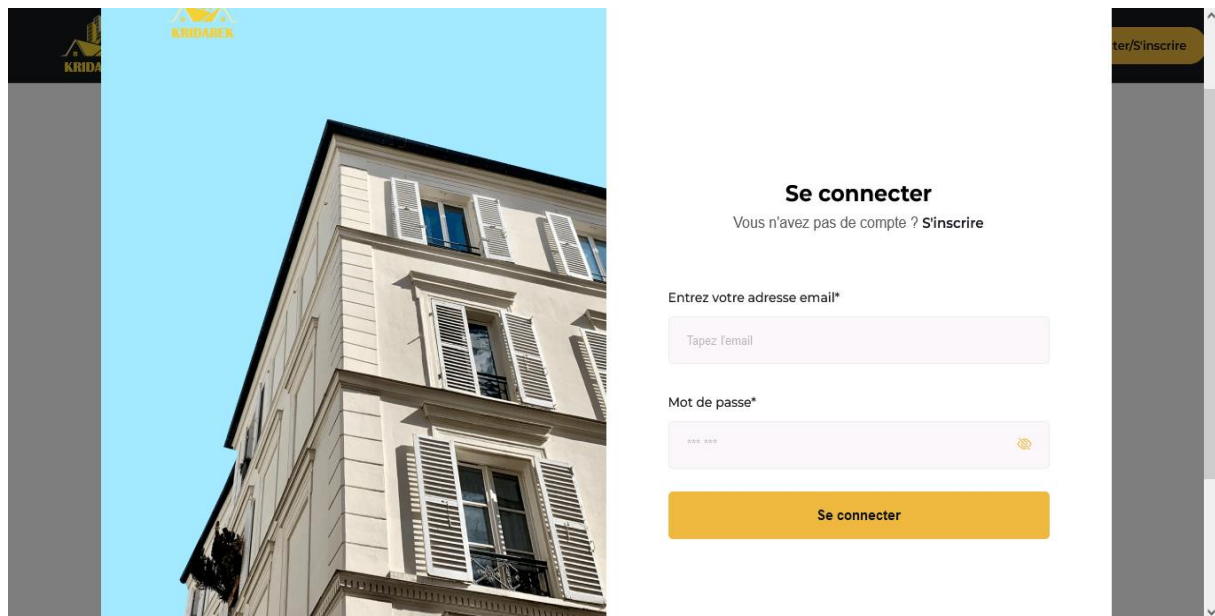
Cette interface décrit la première page d'accueil affichée après l'accès au site



Figure 39 - Interface principale pour l'adhérent et visiteur

3.3.2 Interface de connexion pour l'adhérent

L'interface ci-dessous de connexion est une fonctionnalité de sécurité provenant d'un utilisateur ait fournit un email et un mot de passe valides.



Se connecter

Vous n'avez pas de compte ? [S'inscrire](#)

Entrez votre adresse email*

Tapez l'email

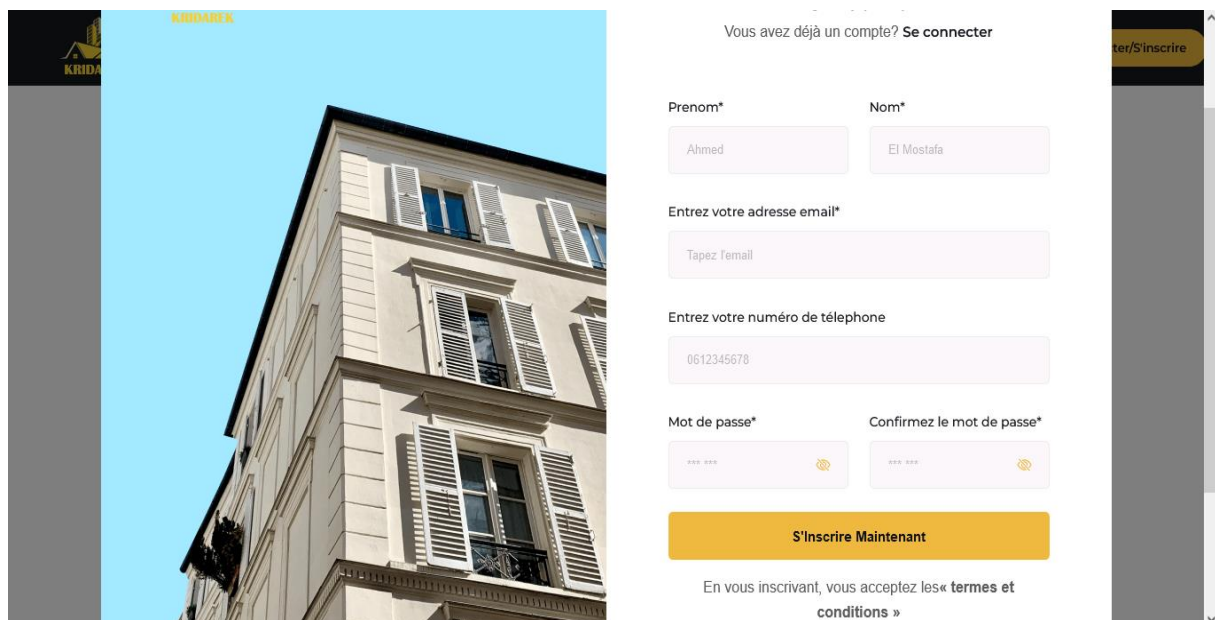
Mot de passe*

Se connecter

Figure 40 - Interface de connexion pour l'adhérent

3.3.3 Interface d'inscription pour le visiteur

Cette interface décrit la page d'inscription pour un visiteur qui n'a pas encore un compte avec le remplissage des champs affichés.



Vous avez déjà un compte? [Se connecter](#)

Prenom* Nom*

Ahmed El Mostafa

Entrez votre adresse email*

Tapez l'email

Entrez votre numéro de téléphone

0612345678

Mot de passe* Confirmez le mot de passe*

S'inscrire Maintenant

En vous inscrivant, vous acceptez les« [termes et conditions](#) »

Figure 41 - Interface d'inscription pour le visiteur

3.3.4 Interface de liste des propriétés pour l'adhérent et le visiteur

Cette interface décrit la page qui contient les biens disponibles pour le moment.

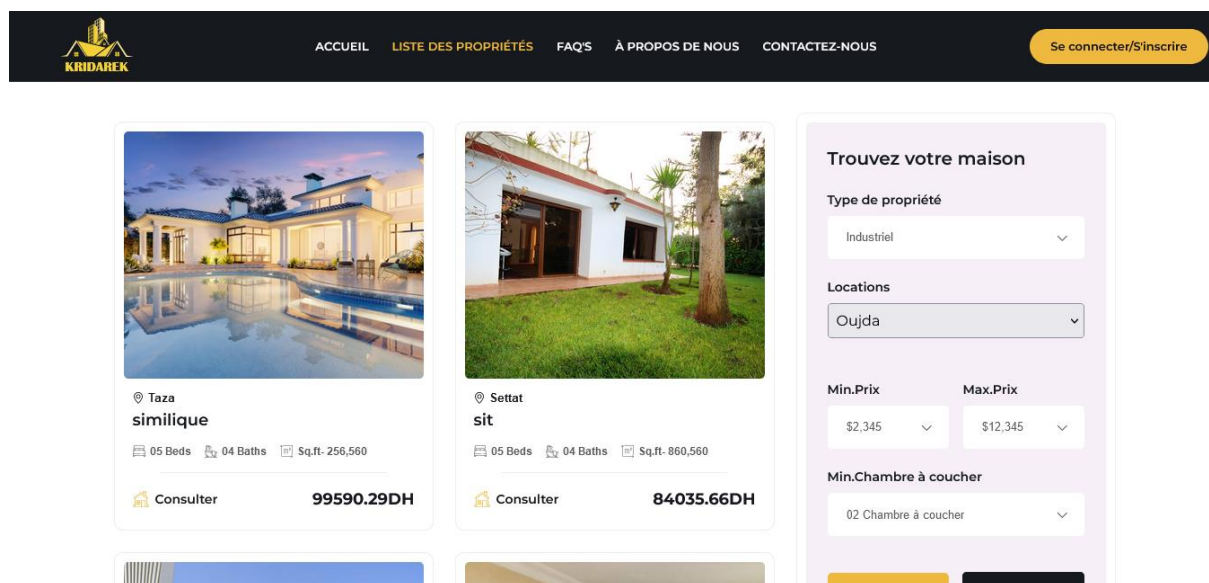


Figure 42 - Interface de liste des propriétés pour l'adhérent et le visiteur

3.3.5 Interface d'Ajout d'une propriété pour l'adhérent

Cette interface permet à l'adhérent d'ajouter une ou plusieurs propriétés tout en cliquant sur le bouton « ajouter une propriété » par la suite d'entrer les informations.

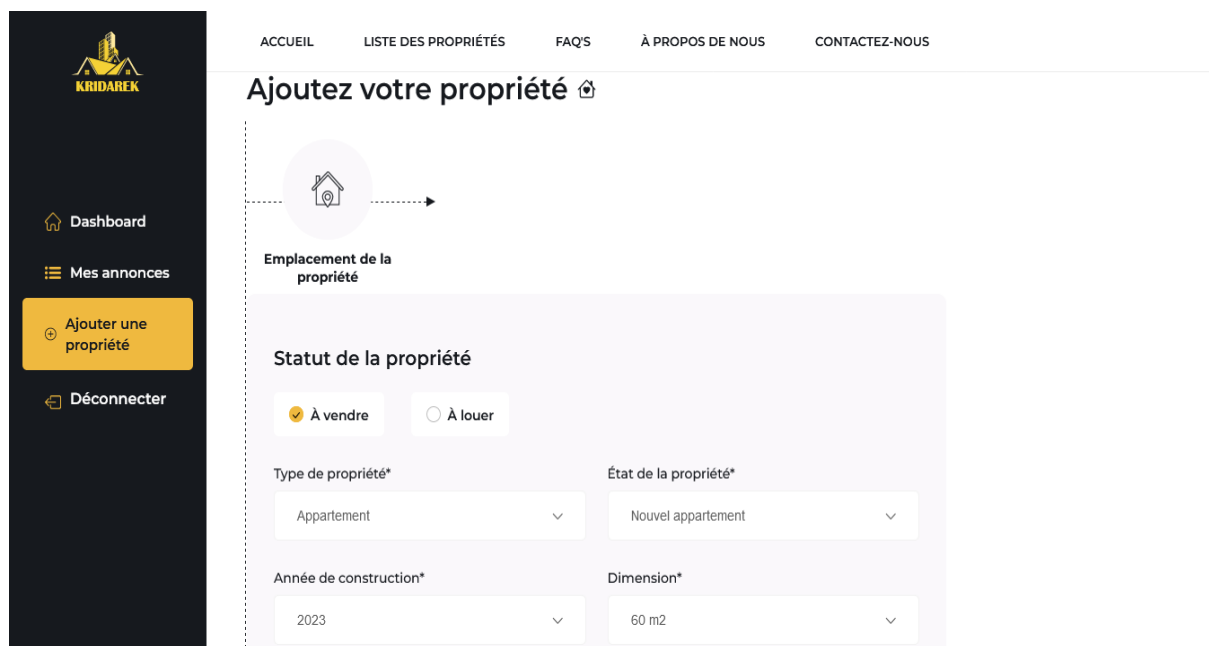


Figure 43 - Interface d'Ajout d'une propriété pour l'adhérent

3.3.6 Interface des annonces d'adhérents offreurs

Cette interface décrit les annonces ajoutées par l'adhérent qui est l'offreur de ces biens

Informations Sur La Propriété	Date D'inscription	Gamme De Prix	Statut De La Propriété	Superficie	Statut	Action
<p>Taza simlique Voir les détails</p>	5/25/2024	99590.29 MAD	Vent	250 m2	Actif	✓ ✕
<p>Settat sit Voir les détails</p>	5/25/2024	8900 MAD	Location	190 m2	Actif	✓ ✕
<p>Mohammedia consequatur Voir les détails</p>	5/25/2024	9000 MAD	Location	170 m2	Actif	✓ ✕

Figure 44 - Interface des annonces d'adhérents offreurs

3.3.6 Interface du Dashboard pour l'adhérent

Cette interface décrit le Dashboard des annonces que l'adhérent les a ajoutées.

Hi, Mohamed-Amine! ❤️

Propriétés en totale
4+

Propriété active
0+

Propriété en attente
1+

Supprimer la propriété
1+

Copyright 2024 © Technova

[À propos de nous](#)
[Avis](#)
[Termes & Conditions](#)

Figure 45 - Interface du Dashboard pour l'adhérent

3.4 Les principales interfaces du Back End pour Admin et Super Admin

3.4.1 interface de connexion pour admin et SuperAdmin

L'interface ci-dessous de connexion est une fonctionnalité de sécurité provenant d'un admin ou Super Admin ait fournit un email et un mot de passe valides.

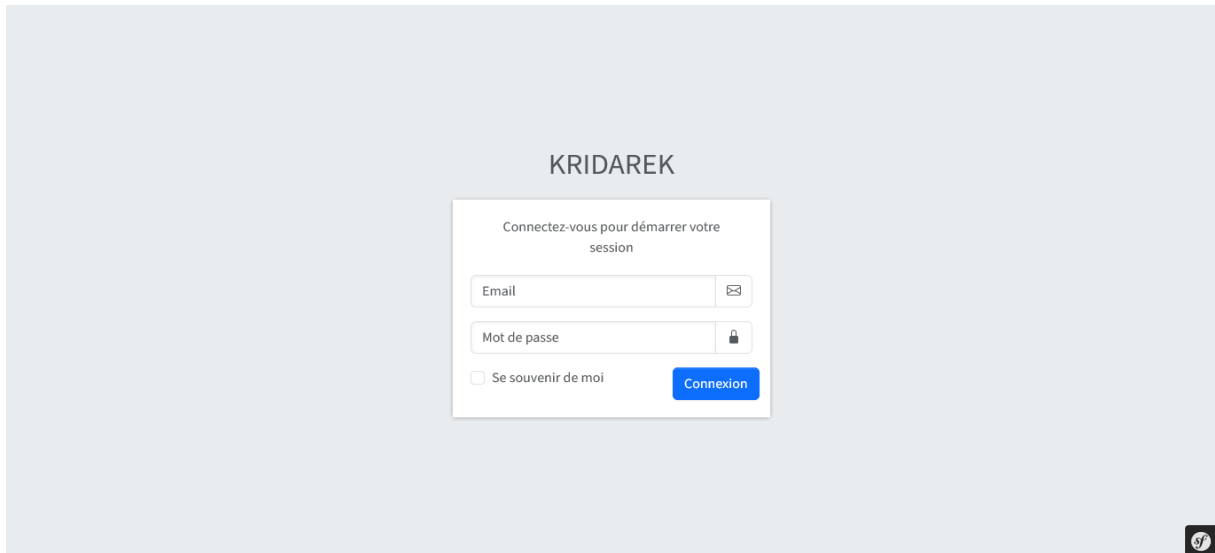


Figure 46 - Interface de connexion pour Admin et SuperAdmin

3.4.2 Interface d'ajout d'un utilisateur par Admin ou SuperAdmin

Cette interface décrit la page qui sera affichée après le clique sur le bouton ajouter un nouveau utilisateur ,elle inclut les champs de saisie pour le nouveau utilisateur

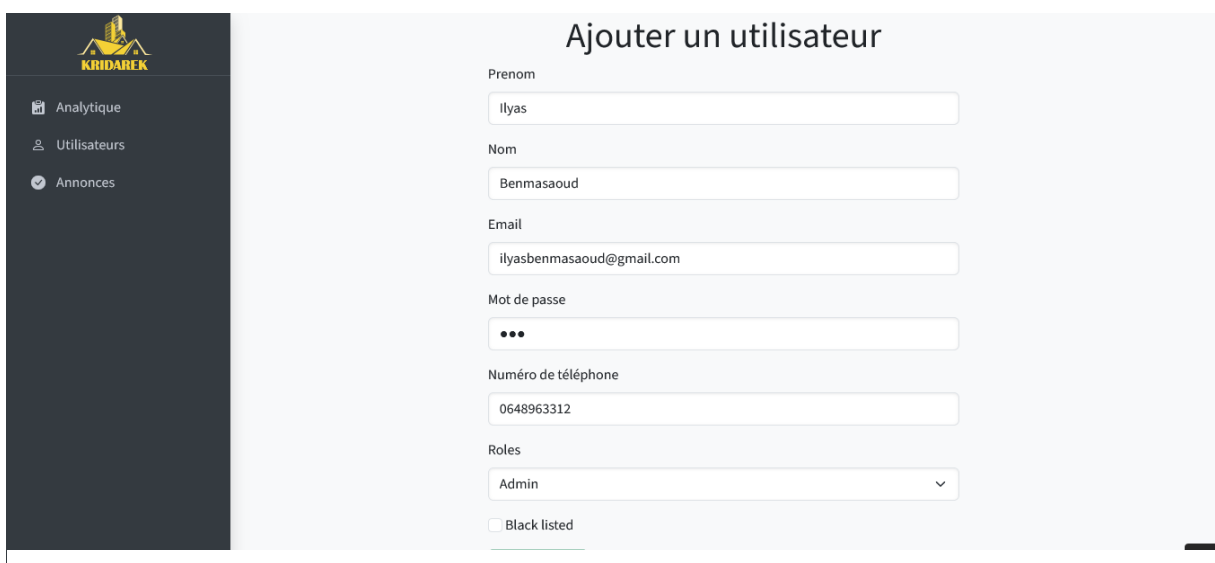


Figure 47 - Interface d'ajout un utilisateur par admin ou SuperAdmin

3.4.3 Interface pour l'ajout des utilisateurs.

Cette interface décrit la page où l'administrateur peut ajouter un utilisateur avec l'affichage d'un message de succès

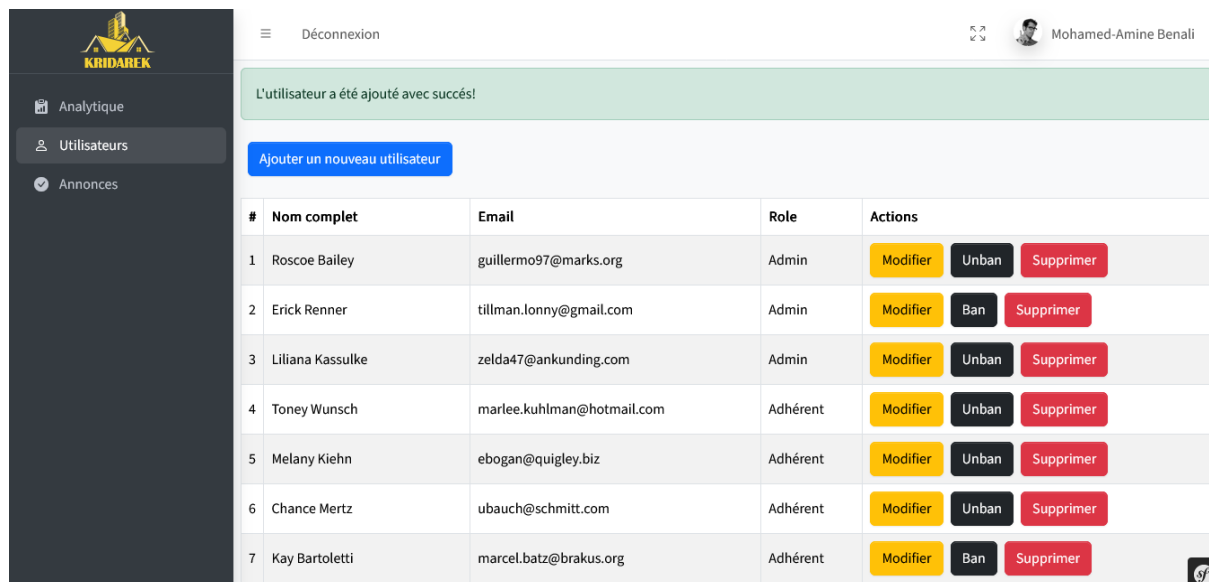


Figure 48 - Interface pour l'ajout des utilisateurs

3.4.4 Interface de listes des annonces des adhérents

Cette interface est disponible seulement pour l'administrateur, elle décrit le processus de gestion des annonces.

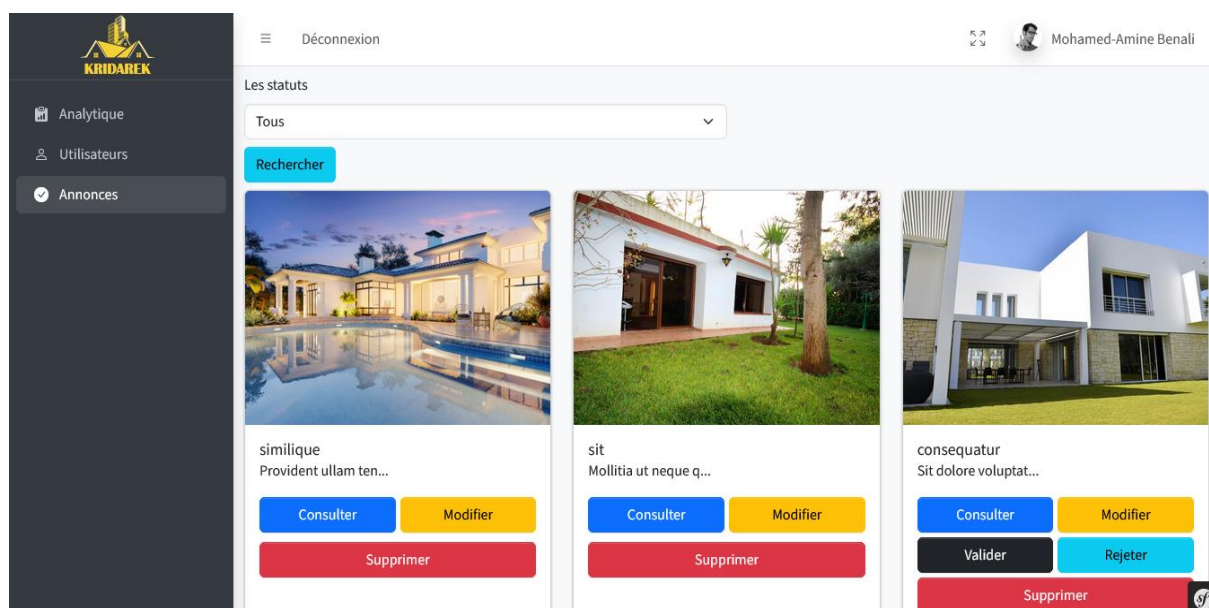


Figure 49 - Interface de liste des annonces des adhérents

3.4.5 Interface de supprimer l'annonce par admin ou superAdmin

Cette interface décrit le message affiché après la suppression d'une annonce par l'administrateur

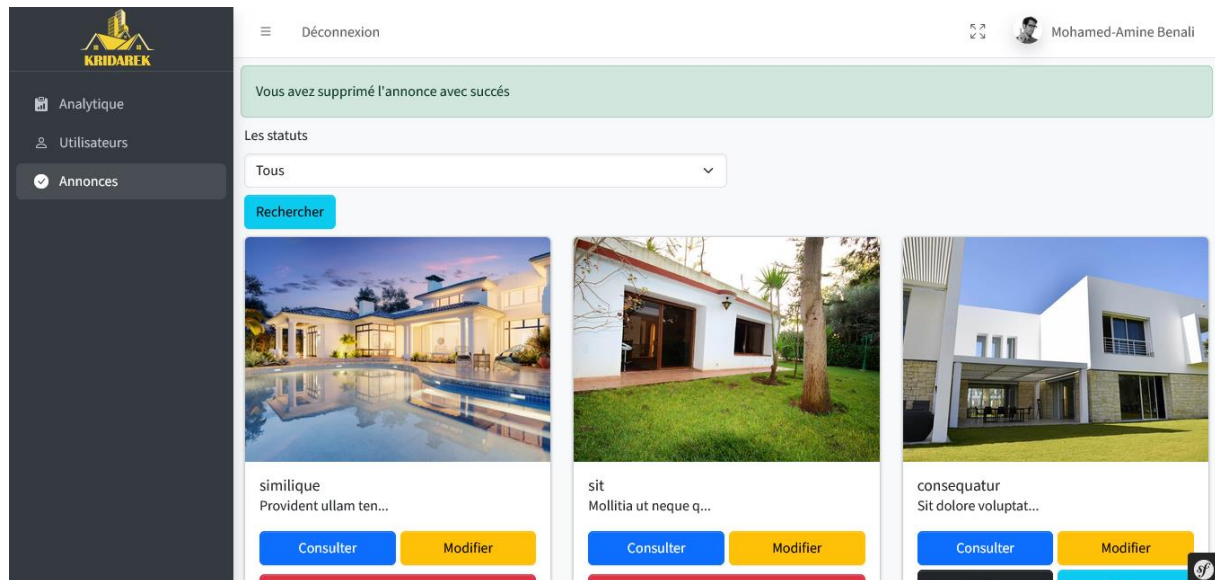


Figure 50 - Interface de supprimer l'annonce par admin ou SuperAdmin

3.4.6 Interface de table de bord d'administrative

Cette interface décrit le tableau de bord de l'administrateur qui affiche tous ce qui déroule en arrière-plan du site

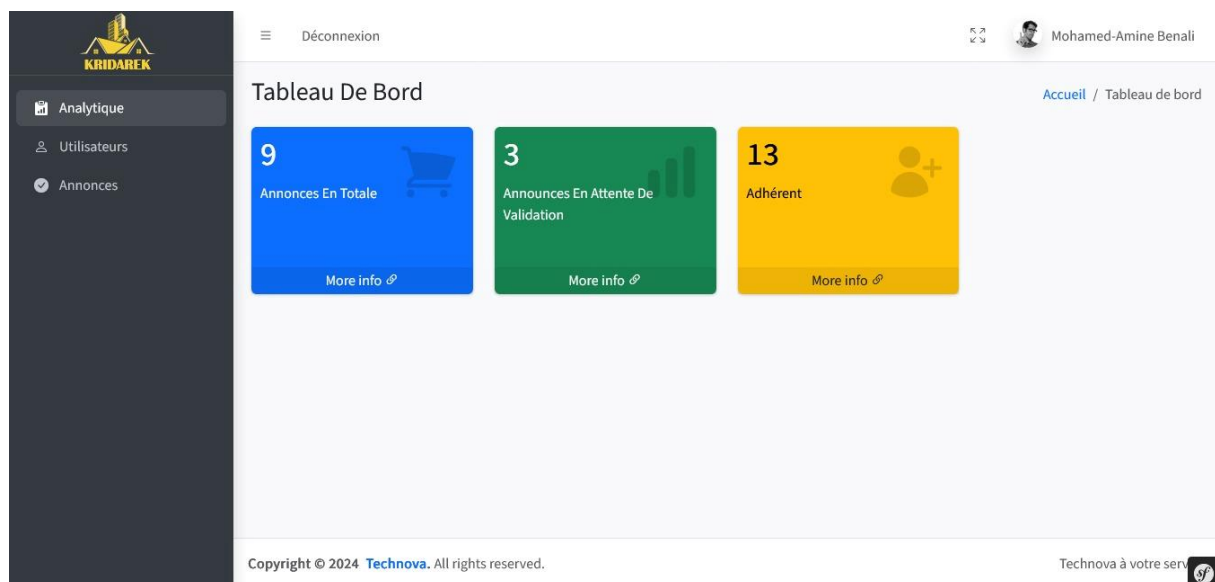


Figure 51 - Interface de table de bord d'administrative

3.5 Interface Api Platform

Cette interface décrit les différentes requêtes API qui peuvent être utilisées pour interagir avec

Les différentes données de chaque propriété et chaque avis.

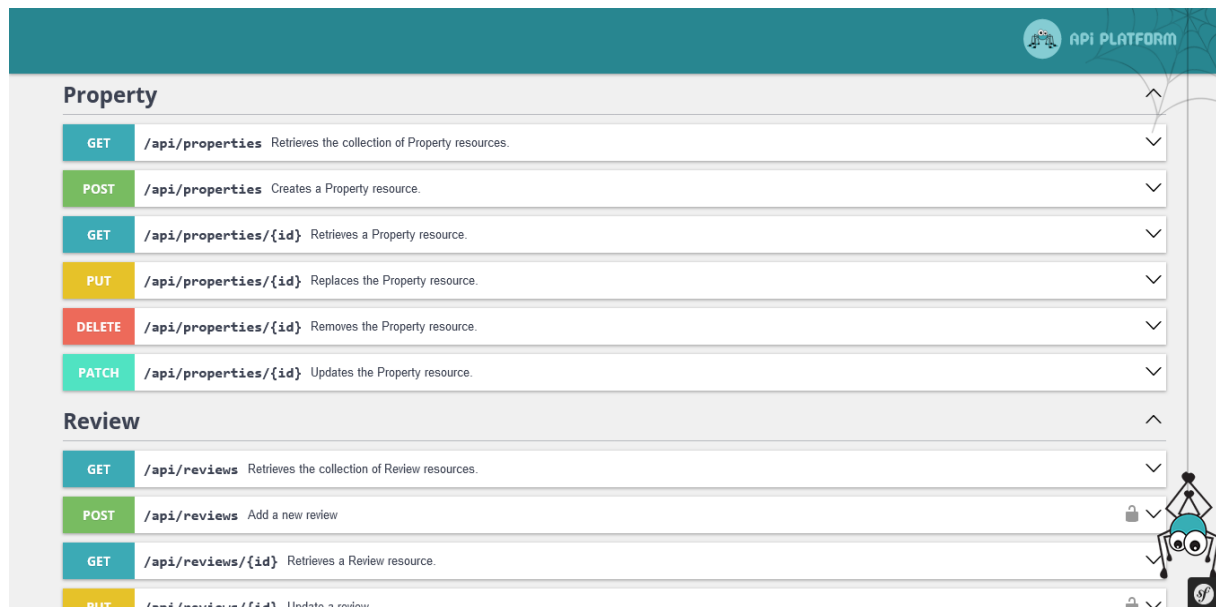


Figure 52 - Interface Api Platform

3.6 Conclusion

Dans ce dernier chapitre, on a réalisé les étapes qui sont basées sur la présentation des différents outils de développement utilisés, ainsi la représentation des captures d'écrans de l'application.

Conclusion générale

L'objectif de ce projet est de concevoir et de développer une plateforme web pour la gestion de biens immobiliers, couvrant les aspects de location et de vente.

Pour atteindre cet objectif, nous avons commencé par présenter le contexte général du projet et à étudier les besoins des utilisateurs. Nous avons également effectué une étude critique des plateformes de gestion immobilière existantes afin d'identifier les différentes fonctionnalités et critères essentiels.

Nous avons ensuite décrit la phase de déroulement du projet avant de passer à la modélisation et à la spécification des besoins de l'application.

Nous avons modélisé toutes les fonctionnalités identifiées en utilisant la modélisation UML (diagramme de classes, diagrammes de cas d'utilisation, diagrammes de séquence et diagramme de séquence MVC).

Enfin, nous avons implémenté les bases de données, les spécifications techniques et les interfaces de l'application en utilisant Symfony, Angular, API Platform et Docker.

Ce projet a été une opportunité qui nous a permis de vivre une expérience professionnelle enrichissante et de découvrir le secteur du développement des applications web, ainsi que d'améliorer nos connaissances et d'apprendre à utiliser d'autres outils de programmation qui sont plus avancés.

Bibliographie

<https://symfony.com/doc/current/index.html>

<https://angular.io/docs>

<https://docs.docker.com/>

<https://www.salesforce.com/fr/resources/definition/api-platform/#topic1>

<https://developer.mozilla.org/fr/docs/Web/HTML>

<https://bpesquet.developpez.com/tutoriels/php/evoluer-architecture-mvc/>

<https://www.jedha.co/blog/git-et-github-definitions-differences-utilite>

<https://www.datarockstars.ai/16309-2/>

<https://welovedevs.com/fr/app/tests/angular-4-legacy>

<https://www.appvizer.fr/services-informatiques/diagramme/drawio>

<https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203597-php-hypertext-preprocessor-definition/>