ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

IIR

INGÉNIERIE INFORMATIQUE & RÉSEAUX

ANNÉE UNIVERSITAIRE

2024/2025

OPTION

MÉTHODES INFORMATIQUES APPLIQUÉES À LA GESTION DES ENTREPRISES (MIAGE)

PROJET DE FIN D'ÉTUDES

THÈME

Realisation d'une plateforme d'investissement immobilier

RÉALISÉ PAR

Elmouhtadi Feirouz

Kelladi Fatimaezzahra

Ouhmida Soulaimane

Ettakadoumi Hamza

Résumé

À une époque où la digitalisation transforme profondément le secteur financier, l'investissement immobilier n'échappe pas à cette mutation. Face à la complexité des processus traditionnels, ce projet ambitieux vise à concevoir une plateforme numérique moderne d'investissement immobilier, accessible via le web et le mobile, combinant technologies innovantes et expérience utilisateur intuitive.

L'objectif est de fournir aux utilisateurs une interface attractive leur permettant de découvrir des biens, de simuler des investissements, de gérer leur portefeuille, et d'effectuer des transactions sécurisées. Pour cela, une architecture technique complète a été mise en place, incluant un backend Java Spring Boot exposant des API REST sécurisées par JWT, une base de données relationnelle MySQL pour la gestion structurée des utilisateurs, portefeuilles et investissements, ainsi qu'une base MongoDB NoSQL pour les biens immobiliers, propriétés aimées ou sauvegardées.

Le traitement des paiements est assuré par Stripe, garantissant une intégration fluide et fiable. Le tout est hébergé sur le cloud AWS, assurant évolutivité, haute disponibilité et sécurité, grâce à des services comme EC2, S3, RDS ou encore CloudFront.

Ce projet met en lumière des compétences techniques avancées en développement full-stack, modélisation UML, gestion agile des tâches via Azure DevOps, et intégration cloud. Il explore également les enjeux de l'ergonomie, de la performance applicative, et de la sécurité des données financières, éléments clés dans la conception d'une solution d'investissement moderne.

Mots clés :

Investissement immobilier, Plateforme numérique, API REST, Spring Boot, MongoDB, MySQL, React, Stripe, Transactions sécurisées, JWT, UX, Cloud AWS, Scalabilité, Architecture MVC, Azure DevOps, Sprints agiles, Modélisation UML, Performances applicatives, Écosystème full-stack.

Abstract

At a time when digitalization is profoundly transforming the financial sector, real estate investment is no exception. Faced with the complexity of traditional processes, this ambitious project aims to design a modern digital real estate investment platform, accessible via web and mobile, combining innovative technologies and an intuitive user experience.

The goal is to provide users with an attractive interface allowing them to discover properties, simulate investments, manage their portfolios, and conduct secure transactions. To achieve this, a comprehensive technical architecture has been implemented, including a Java Spring Boot backend exposing REST APIs secured by JWT, a MySQL relational database for structured user, portfolio, and investment management, as well as a MongoDB NoSQL database for real estate, favorite, and saved properties.

Payment processing is handled by Stripe, ensuring seamless and reliable integration. Everything is hosted on the AWS cloud, ensuring scalability, high availability, and security, thanks to services such as EC2, S3, RDS, and CloudFront. This project highlights advanced technical skills in full-stack development, UML modeling, agile task management via Azure DevOps, and cloud integration. It also explores the challenges of usability, application performance, and financial data security, key elements in the design of a modern investment solution.

Keywords:

Investissement immobilier, Plateforme numérique, API REST, Spring Boot, MongoDB, MySQL, React, Stripe, Transactions sécurisées, JWT, UX, Cloud AWS, Scalabilité, Architecture MVC, Azure DevOps, Sprints agiles, Modélisation UML, Performances applicatives, Écosystème full-stack.

Liste des Figures

Liste des tableaux

Table des matières

Introduction Générale

Dans un contexte économique où la recherche de placements rentables et sécurisés devient une préoccupation majeure pour de nombreux particuliers, l'investissement immobilier se présente comme une alternative attractive. Néanmoins, l'accès à ce type d'investissement reste souvent limité par des barrières financières, administratives ou techniques. Le projet présenté dans ce rapport vise à concevoir une plateforme numérique innovante permettant aux utilisateurs d'investir facilement dans des biens immobiliers, de suivre leurs placements, et d'accéder à des informations détaillées sur les projets immobiliers proposés.

En combinant les avantages des technologies modernes telles que les bases de données hybrides (relationnelles et NoSQL), les services web RESTful, et une architecture back-end robuste, cette solution a pour objectif de démocratiser l'investissement immobilier, d'optimiser la gestion des investissements, et de garantir la transparence des données. Le présent rapport détaille le contexte, la problématique ciblée, la solution développée, ainsi que les outils techniques employés pour sa mise en œuvre.

Contexte

L'investissement immobilier, bien que considéré comme l'un des moyens les plus sûrs de faire fructifier son capital, reste peu accessible à une grande partie de la population en raison de la complexité des démarches, du manque de transparence, ou encore du besoin d'un capital de départ élevé. Par ailleurs, de nombreux investisseurs souhaitent pouvoir diversifier leurs investissements dans plusieurs projets, tout en suivant en temps réel l'évolution de leur portefeuille.

Dans ce cadre, les plateformes numériques représentent une opportunité majeure pour réinventer l'accès à l'investissement immobilier. En permettant aux utilisateurs d'investir de manière fragmentée et transparente, ces plateformes offrent un nouveau modèle participatif qui favorise l'inclusion financière et la gestion dématérialisée des investissements.

Problématique

Les difficultés identifiées dans le domaine de l'investissement immobilier incluent :

- L'inaccessibilité de certains projets à cause de la barrière du capital initial requis ;
- Le manque de transparence sur les projets et la difficulté d'accéder à des informations fiables et détaillées :
- L'absence d'outils numériques simples permettant aux investisseurs de suivre leurs investissements en temps réel ;
- La nécessité de gérer efficacement de grandes quantités de données immobilières avec des structures flexibles et performantes.

Face à ces constats, il devient essentiel de proposer une solution numérique complète, intuitive, et sécurisée, qui facilite l'investissement et le suivi des projets immobiliers pour tous les profils d'investisseurs.

Objectifs

Le projet a pour objectifs de :

- Offrir une plateforme intuitive pour permettre aux utilisateurs d'investir dans différents projets immobiliers.
- Intégrer une base de données hybride (MySQL + MongoDB) afin de gérer efficacement les investissements et les fiches techniques des biens immobiliers.
- Associer chaque investissement utilisateur (deal) à un bien immobilier et enrichir dynamiquement les données affichées.
- Garantir la cohérence et la performance de la plateforme grâce à l'architecture Spring Boot et aux technologies modernes de développement web.

Structure du Rapport

Ce rapport est organisé en plusieurs chapitres, chacun décrivant une étape spécifique du projet.

1. Présentation de l'organisme d'accueil ;
2. Analyse et Conception ;
3. Étude Technique ;
4. Mise en œuvre et Réalisation.

CHAP1 : Présentation de l'organisme d'accueil

Introduction

Le premier chapitre de ce rapport présente l'organisme d'accueil dans lequel nous avons réalisé ce projet, l'EMSI. Ce chapitre vise à fournir un aperçu général de l'institution, de ses valeurs, de ses secteurs d'activité et de ses contributions dans le domaine de l'éducation et du développement des compétences.

EMSI

ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

FIGURE 1: LOGO DE L'EMSI

Présentation

L'EMSI (École Marocaine des Sciences de l'Ingénieur) est un établissement d'enseignement supérieur privé qui se spécialise dans la formation d'ingénieurs et de cadres dans diverses disciplines. Depuis sa création, l'EMSI a su s'imposer comme l'un des leaders dans le domaine de l'éducation en ingénierie au Maroc.

L'EMSI propose des programmes de formation dans plusieurs domaines, incluant l'informatique, les réseaux et télécommunications, le génie civil, et bien d'autres encore. Ces programmes sont conçus pour répondre aux besoins croissants du marché du travail en formant des professionnels compétents et prêts à affronter les défis technologiques de demain. L'EMSI dispose de plusieurs campus à travers le pays, ce qui lui permet d'avoir une portée nationale et de s'adresser à un large public.

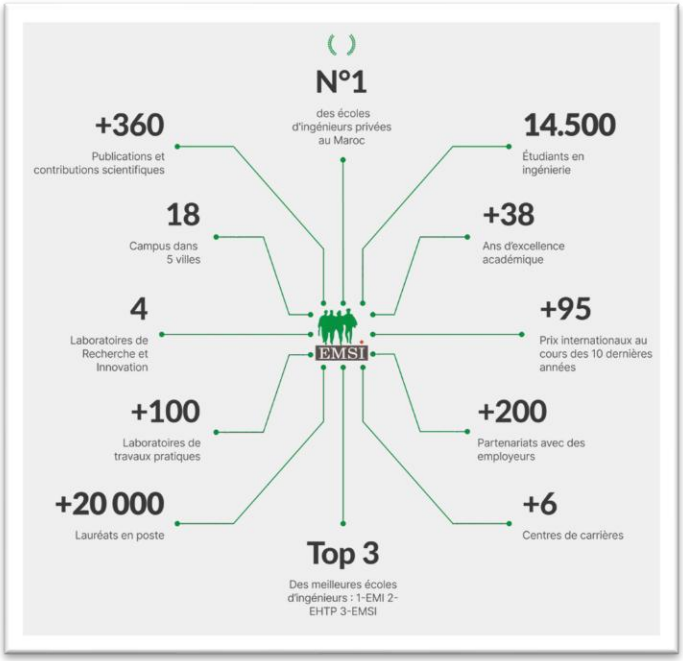

FIGURE 2: LES CHIFFRES CLES

Les valeurs de l'EMSI

L'EMSI se base sur plusieurs valeurs fondamentales qui guident son fonctionnement et son approche pédagogique :

- **Excellence :** La recherche constante de la qualité dans l'enseignement et la formation des étudiants.
- **Innovation :** L'adaptation aux évolutions technologiques et la mise en œuvre de solutions innovantes dans le domaine de l'enseignement.
- **Responsabilité sociale :** Un engagement envers la société en formant des professionnels responsables et éthiques.
- **Esprit d'équipe :** Encourager la collaboration et le travail en groupe afin de développer des compétences interpersonnelles chez les étudiants.

Secteurs d'activité

L'EMSI intervient principalement dans les secteurs de l'éducation, de la recherche et de l'innovation technologique. Elle forme des étudiants dans des domaines techniques en forte demande, ce qui contribue à renforcer les compétences des jeunes talents dans des secteurs tels que l'industrie, les technologies de l'information, la construction, et bien d'autres. Par ailleurs, l'EMSI s'engage activement dans la recherche scientifique, notamment dans les domaines de l'intelligence artificielle, des systèmes embarqués, et de la gestion des réseaux.

Conclusion

Ce premier chapitre a permis de mieux comprendre l'organisme d'accueil, l'EMSI, en exposant ses principales missions, ses valeurs, et ses domaines d'intervention. L'EMSI constitue un environnement propice à l'apprentissage et à l'innovation, et joue un rôle clé dans la formation des futurs professionnels qui contribueront au développement technologique du pays.

CHAP 2 : Analyse et Conception

Introduction

L'analyse et la conception sont des étapes fondamentales dans le processus de développement d'un système. Elles permettent de transformer les besoins exprimés dans le cahier des charges en une solution technologique efficace et robuste. Dans ce chapitre, nous détaillerons l'analyse du projet, en mettant l'accent sur la modélisation UML pour représenter les différentes facettes du système. Nous commencerons par la modélisation UML, qui constitue une approche standardisée et compréhensible pour toutes les parties prenantes, puis nous détaillerons les diagrammes spécifiques qui facilitent la conception du système.

I. Modélisation UML

Un diagramme UML est un moyen de visualiser des systèmes et des logiciels à l'aide du langage de modélisation unifié (UML). Les ingénieurs logiciels créent des diagrammes UML pour comprendre la conception, l'architecture du code et la mise en œuvre proposée de systèmes logiciels complexes [1].

Les diagrammes utilisés sont :

- **Diagramme des cas d'utilisation :** Il permet de visualiser les fonctionnalités principales que l'application offre aux utilisateurs ;
- **Diagramme de classes :** Il montre la structure du système en termes de classes et leurs relations.

Diagrammes de cas d'utilisation

Les diagrammes de cas d'utilisation représentent les interactions entre les utilisateurs (acteurs) et le système. Ils aident à définir les besoins fonctionnels du système en détaillant les actions que chaque acteur peut effectuer [2].

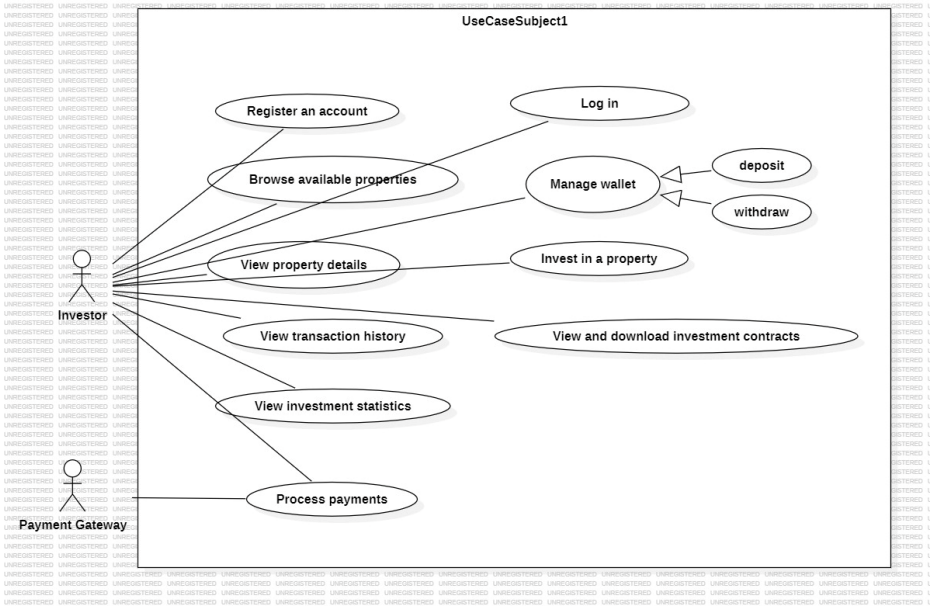

FIGURE 3: DIAGRAMME DE CAS D'UTILISATION

Diagrammes de classes

Les diagrammes de classes décrivent les objets du système et les relations entre eux. Ils fournissent une vue statique du système, permettant de comprendre la structure du code et des composants du projet [2].

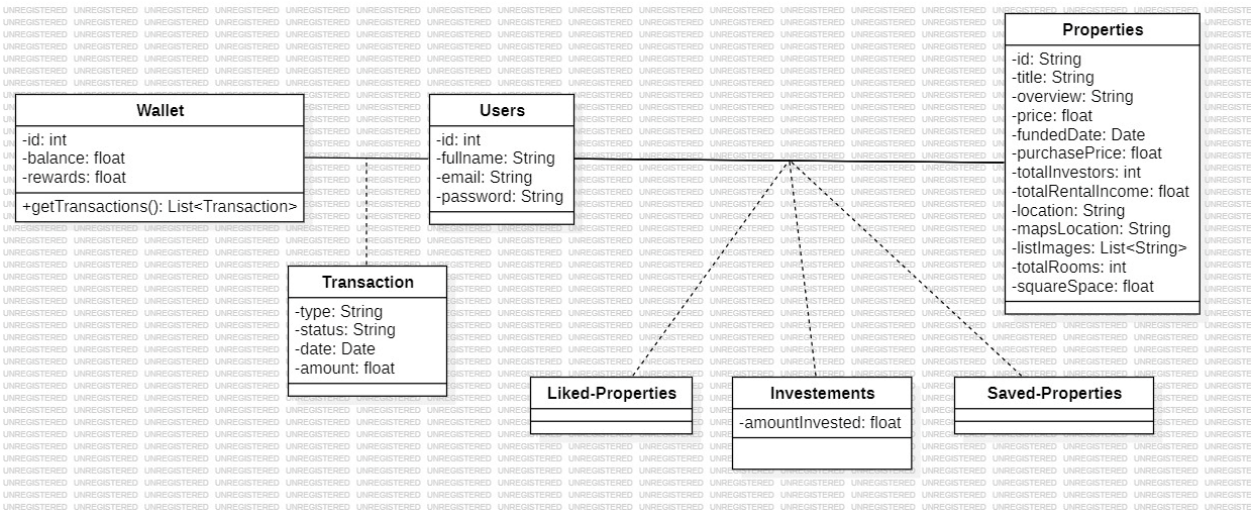

FIGURE 4: DIAGRAMME DE CLASSES

II. Modélisation Cloud

En plus des diagrammes UML, l'architecture cloud du projet est un élément clé pour assurer la performance et la scalabilité du système. Le déploiement de l'application se fait sur AWS, en utilisant différents services pour gérer les interactions entre les composants et garantir une expérience utilisateur fluide.

Qu'est-ce que le cloud computing ?

Le terme « cloud » désigne les serveurs accessibles sur Internet, ainsi que les logiciels et bases de données qui fonctionnent sur ces serveurs. Les serveurs situés dans le cloud sont hébergés au sein de datacenters répartis dans le monde entier. L'utilisation du cloud computing (informatique cloud) permet aux utilisateurs et aux entreprises de s'affranchir de la nécessité de gérer des serveurs physiques eux-mêmes ou d'exécuter des applications logicielles sur leurs propres équipements [3].

Le diagramme d'architecture cloud inclut les éléments suivants :

Service	Description
AWS EC2	Hébergement du backend Spring Boot avec plusieurs instances pour garantir la haute disponibilité.
AWS Load Balancer	Distribution du trafic entre plusieurs instances pour équilibrer la charge
AWS RDS	Base de données relationnel pour stocker les informations utilisateur et le contenu.
Amazon DocumentDB	Hébergement des bases de données MongoDB.
AWS S3	Stockage des fichiers vidéo et autres ressources.
AWS CloudFront	Diffusion des vidéos et contenu statique via un CDN pour minimiser la latence.

TABLEAU 1: SERVICES CLOUD AWS

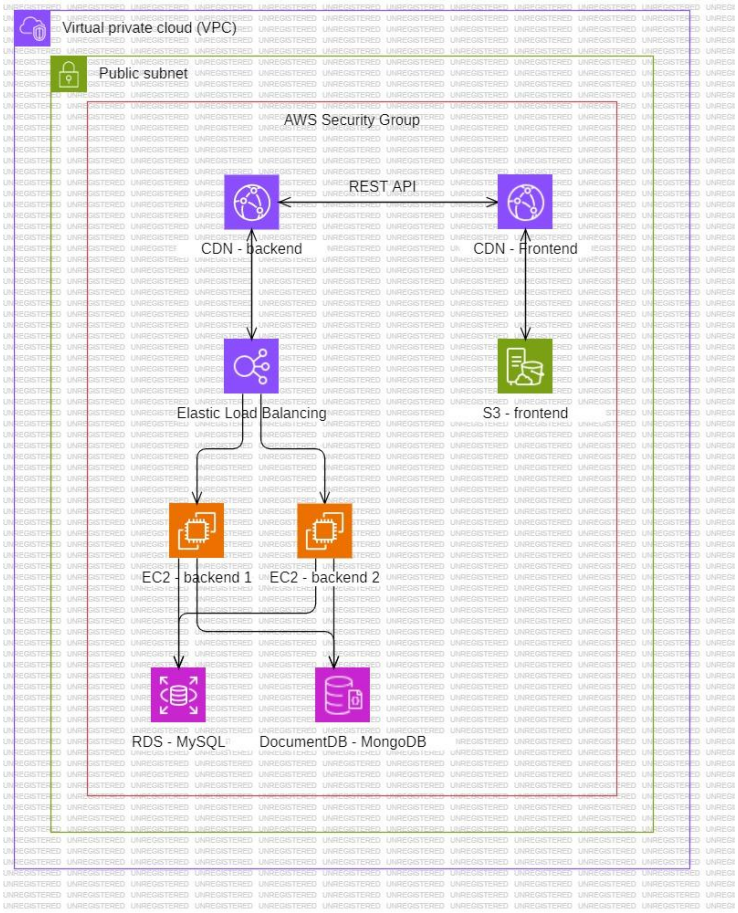

FIGURE 5: DIAGRAMME DU CLOUD

III. Gestion de projet

a. Diagramme de Gantt

Un diagramme de Gantt, couramment utilisé en gestion de projet, est l'un des moyens les plus populaires et les plus utiles pour représenter les activités (tâches ou événements) dans le temps [4].

Task	Start date	End date	Days	Progress	1-Apr Wk 1	8-Apr Wk 2	15-Apr Wk 3	22-Apr Wk 4	29-Apr Wk 5	6-May Wk 6	13-May Wk 7
Planification											
Etude de marche	4/1/2025	4/7/2025	6	100%							
Realisation du cahier des ch	4/1/2025	4/7/2025	6	100%							
les masuettes	4/1/2025	4/7/2025	6	100%							
Développement Backend											
Authentification	4/6/2025	4/27/2025	21	10%							
API	4/6/2025	4/27/2025	21	10%							
Intergation du paiement	4/6/2025	4/27/2025	21	0%							
Développement Frontend											
Intégration du design	4/21/2025	5/12/2025	21	10%							
Task 2	4/21/2025	5/12/2025	21	0%							
Tests & Déploiments											
Tests unitaires	5/13/2025	5/13/2025	0	0%							
Déploiment cloud	5/13/2025	5/13/2025	0	0%							

FIGURE 6: DIAGRAMME DE GANTT

b. Gestion des Sprints

A sprint is a short, time-boxed period when a scrum team works to complete a set amount of work. Sprints are at the very heart of scrum and agile methodologies, and getting sprints right will help your agile team ship better software with fewer headaches [5].

Le projet a été géré en suivant une méthodologie agile, avec des sprints définis sous **Azure DevOps.** Chaque sprint couvre une étape du développement:

- **Sprint 1** : Setup du projet, base frontend et backend ;
- **Sprint 2** : Development backend et Intégration Stripe et paiements ;
- **Sprint 3** : Development web et mobile;
- **Sprint 4** : Tests, déploiement et documentation.

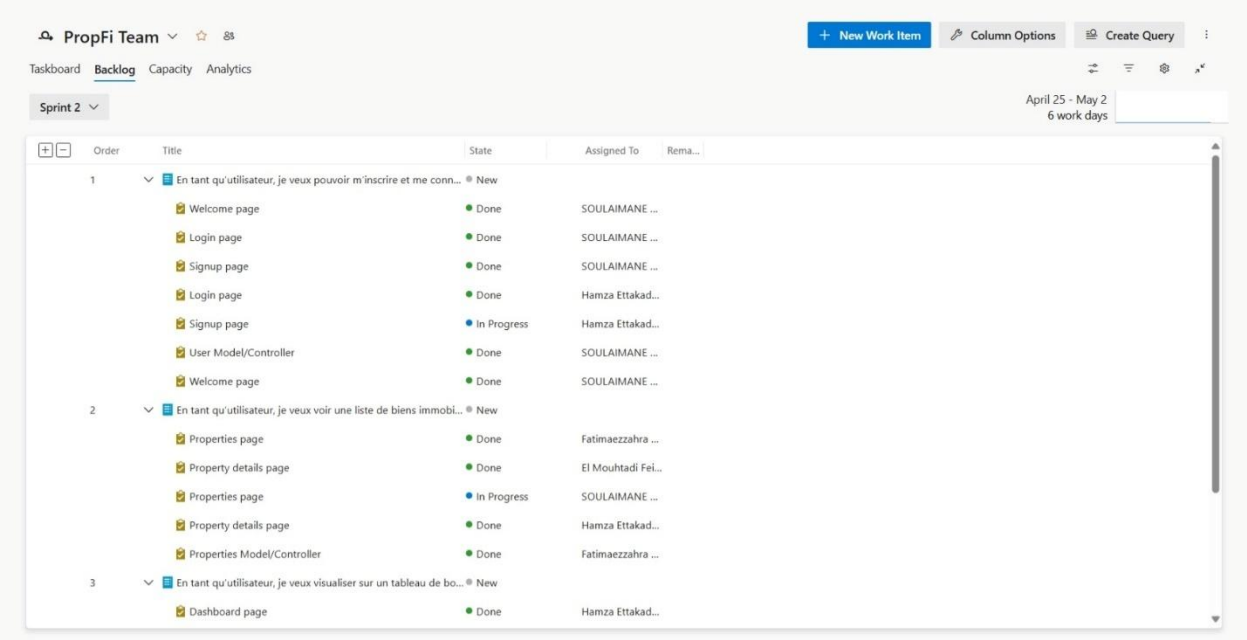

FIGURE 7: CAPTURE D'ECRAN AZURE DEVOPS "SPRINT 2"

Conclusion

La modélisation UML dans ce chapitre a permis de mieux comprendre l'architecture et le fonctionnement du système de recyclage de l'eau. Les diagrammes de cas d'utilisation, de classes et de séquence permettent de clarifier les besoins du système, d'organiser les données et d'illustrer le processus d'interaction entre les différents acteurs et composants. Cette analyse détaillée pose les bases d'une conception technique robuste et constitue un guide pour la mise en œuvre du système.

CHAP 3 : Étude Technique

Introduction

L'étude technique est une étape essentielle dans la mise en œuvre d'un projet numérique, car elle permet d'identifier et d'évaluer les solutions et technologies qui seront utilisées pour concrétiser le système proposé. Dans le cadre de ce projet de plateforme d'investissement immobilier, cette étude a pour but de détailler l'environnement de travail, de comparer les solutions techniques existantes et de présenter les outils et technologies choisis pour garantir la réussite du projet.

I. Présentation de l'environnement de travail

L'environnement de développement du projet repose sur une combinaison d'outils et de technologies adaptés à un développement web et mobile moderne, en mettant l'accent sur la performance, la scalabilité et la facilité de déploiement.

Système d'exploitation	• Windows 11 (local) • Android • Amazon Linux 2 (serveur sur AWS EC2).
IDE	Visual Studio Code.
Langages de programmation	• Java : Utilisé pour le backend, avec le framework Spring Boot. • JavaScript (ES6+) : Utilisé pour le développement frontend avec React.
Bases de données	• MySQL pour les données relationnelles (utilisateurs, transactions, portefeuilles, offres).
Cloud	Amazon Web Services (AWS) pour l'hébergement, le stockage et les services associés.

TABLEAU 2: ENVIRONEMMENT DE TRAVAIL

II. Comparaison entre les solutions techniques existantes

Avant de sélectionner les technologies spécifiques utilisées dans ce projet, il est essentiel de comparer les solutions techniques existantes pour déterminer celles qui sont les plus adaptées.

Voici quelques-unes des solutions analysées :

1. Backend : Spring Boot vs Node.js

- **Spring Boot (choix retenu)** : Ce framework Java a été sélectionné pour sa robustesse, sa capacité à gérer des architectures complexes et son intégration fluide avec des systèmes de sécurité et de gestion des transactions. Son écosystème mature facilite également la création d'API RESTful sécurisées et évolutives.

- **Node.js** : Bien que performant pour des applications en temps réel, Node.js a été écarté en raison de la complexité des transactions à gérer et des exigences de sécurité élevées du système, notamment liées à la fiabilité des données environnementales.

2. Frontend : React vs Angular

- **React (choix retenu)** : Cette bibliothèque JavaScript a été préférée pour sa flexibilité et sa rapidité de développement. Elle permet une interface utilisateur dynamique, idéale. L'intégration avec Redux a également facilité la gestion d'état dans des interfaces complexes.

- **Angular** : Bien qu'Angular soit un framework complet, sa structure plus lourde et sa courbe d'apprentissage plus longue le rendaient moins adapté à un projet développé individuellement dans un délai restreint.

3. Base de données relationnelle : MySQL vs PostgreSQL

- **MySQL (choix retenu) :** MySQL a été retenu pour gérer les données structurées du système, telles que les utilisateurs, les transactions, les portefeuilles, les offres. Elle est bien supportée sur AWS RDS, ce qui facilite son intégration cloud.

- **PostgreSQL :** Bien que très performant et riche en fonctionnalités, PostgreSQL a été écarté pour ce projet au profit de la simplicité et de la popularité de MySQL dans l'écosystème AWS.

4. Stockage de données volumineuses : Cassandra vs MongoDB

- **MongoDB (choix retenu) :** MongoDB a été choisi pour stocker des données non structurées et évolutives, telles que les propriétés et les documents liés aux investissements (rapports PDF, commentaires, préférences utilisateur, etc.). Sa flexibilité permet de faire évoluer le modèle de données sans contrainte, ce qui est idéal pour un projet où les types et formats de données peuvent varier au fil du temps.

- **Cassandra** : Plus adapté à des systèmes nécessitant une écriture massive et distribuée, Cassandra a été écarté car ses capacités dépassaient les besoins du projet, tout en introduisant une complexité inutile.

5. Infrastructure Cloud : AWS vs Google Cloud

- **AWS (choix retenu) :** L'écosystème AWS a été choisi pour héberger l'ensemble de l'architecture technique. Les services comme EC2, RDS, S3, IoT Core, ElastiCache et Keyspaces ont été déterminants pour la mise en place d'un système IoT scalable, sécurisé et interconnecté. De plus, AWS est largement documenté et dispose d'un support actif.

- **Google Cloud :** Bien qu'il propose aussi des services performants, Google Cloud a été écarté en raison de la moindre maturité de son offre IoT et de la familiarité plus grande avec AWS dans le cadre de ce projet.

III. Technologies et outils utilisés

Les technologies et outils sélectionnés pour ce projet sont choisis en fonction de leur capacité à répondre aux exigences de performance, de scalabilité et de sécurité. Voici un aperçu des choix faits :

1. Frontend

React.js

FIGURE 8: LOGO DE REACT JS

React est une bibliothèque JavaScript open-source qui est utilisée pour construire des interfaces utilisateur spécifiquement pour des applications d'une seule page. Elle est utilisée pour gérer la couche d'affichage des applications web et mobiles. React a été créé par Jordan Walke, un ingénieur logiciel travaillant pour Facebook. React a été déployé pour la première fois sur Facebook en 2011 et sur Instagram en 2012 [6].

React Native

FIGURE 9: LOGO DE REACT NATIVE

React Native est un framework d'applications mobiles populaire basé sur JavaScript qui permet de créer des applications mobiles natives pour iOS et Android. Ce framework permet de créer une application pour différentes plateformes en utilisant le même code source [7].

Expo

FIGURE 10: LOGO DE EXPO

Il s'agit d'un ensemble de nombreux outils dont l'objectif principal est de faciliter le travail des développeurs utilisant React Native. Il comprend, entre autres, des bibliothèques (Expo SDK), grâce auxquelles il n'est pas nécessaire d'écrire du code de toutes pièces pour les fonctions présentes dans la plupart des applications mobiles [8].

Javascript

FIGURE 11: LOGO DE JAVASCRIPT

JavaScript est un langage de programmation qui permet d'implémenter des mécanismes complexes sur une page web. À chaque fois qu'une page web fait plus que simplement afficher du contenu statique [9].

HTML

FIGURE 12: LOGO DE HTML

HTML est l'abréviation de « hypertext markup language » (langage de balisage hypertexte) et est un langage relativement simple utilisé pour créer des pages web. Comme il n'autorise pas les variables ou les fonctions, il n'est pas considéré comme un « langage de programmation », mais plutôt comme un « langage de balisage » [10].

CSS

FIGURE 13: LOGO DE CSS

CSS désigne Cascading Style Sheets (pour Feuilles de style en cascade). Il s'agit d'un langage de style dont la syntaxe est extrêmement simple mais son rendement est remarquable. En effet, le CSS s'intéresse à la mise en forme du contenu intégré avec du HTML [11].

Tailwind JS

FIGURE 14: LOGO DE TAILWIND

Tailwind CSS est un framework permettant aux développeurs de personnaliser totalement et simplement le design de leur application ou de leur site web. Avec ce framework CSS, il est possible de créer un design d'interface au sein même du fichier HTML [12].

2. Backend

Java

FIGURE 15: LOGO DE JAVA

Java est un langage de programmation multiplateforme, orienté objet et largement utilisé pour coder des applications Web. Il s'agit d'un choix populaire parmi les développeurs depuis plus de deux décennies [13].

Spring Boot

FIGURE 16: LOGO DE SPRING BOOT

Spring Boot est un framework de développement JAVA. C'est une déclinaison du framework classique de Spring qui permet essentiellement de réaliser des microservices (ce sont la majeure partie du temps des services web qui sont regroupés en API) [14].

3. Databases

MySQL

FIGURE 17: LOGO DE MYSQL

MySQL est un système de gestion de bases de données relationnelles SQL open source développé et supporté par Oracle. Peut être utilisé par les développeurs et des administrateurs de bases de données pour gérer efficacement les données de leurs projets [15].

Mongo DB

FIGURE 18: LOGO DE MONGO DB

MongoDB est une base de données de documents NoSQL open source populaire utilisée pour stocker et récupérer des données dans un format flexible et orienté document [16].

4. Traitement des paiements

FIGURE 19: LOGO DE STRIPE

Stripe est une société de traitement des paiements qui permet aux commerçants d'accepter les cartes de crédit et de débit, ainsi que d'autres types de paiement [17].

5. Cloud

AWS

FIGURE 20: LOGO DE AWS

AWS (Amazon Web Services) est une plateforme de cloud computing fournie par Amazon. Offre des outils tels que la puissance de calcul, le stockage de bases de données et les services de diffusion de contenu [18].

6. Gestion du Code Source

Git

FIGURE 21: LOGO DE GIT

Git est un outil DevOps utilisé pour la gestion du code source. Il s'agit d'un système de contrôle de version gratuit et open source utilisée pour gérer efficacement des projets de petite à très grande envergure. Git est habitué à suivre les modifications du code source [19].

7. Outils de Collaboration

Azure devops

FIGURE 22: LOGO DE AZURE DEVOPS

Azure DevOps offre une traçabilité de bout en bout, permettant aux développeurs de suivre le travail tout au long du cycle de vie, des exigences au déploiement. Azure Pipelines assure la livraison, l'intégration et la gestion du déploiement [20].

Conclusion

Cette étude technique a permis de poser les fondations d'une architecture **fiable, performante et évolutive**, parfaitement adaptée aux besoins d'une plateforme d'investissement immobilier. Les choix technologiques ont été guidés par la volonté d'assurer **sécurité**, **scalabilité**, **maintenabilité** et **expérience utilisateur fluide**. Grâce à la combinaison de React.js pour le frontend, Spring Boot pour le backend, et AWS pour l'infrastructure cloud, la solution développée est prête à être utilisée par des utilisateurs réels et à évoluer vers de nouveaux services ou marchés.

CHAP 4 : Mise en œuvre et Réalisation

Introduction

Dans ce chapitre, nous détaillerons les étapes de l'implémentation de la plateforme d'investissement immobilier, en nous concentrant sur le développement de l'interface utilisateur (frontend), l'application mobile et de l'API backend. L'objectif est de fournir une vue d'ensemble sur les technologies et outils utilisés pour concevoir une solution numérique intuitive, performante et sécurisée.

Implémentation et solution développée

I. Backend: API REST avec Spring Boot

Le backend est géré par Spring Boot, un Framework Java qui permet de créer et gérer des API RESTful robustes. Les principales fonctionnalités du backend incluent :

- **API de gestion des utilisateurs :** Authentification sécurisée par JWT et gestion des portefeuilles, transactions, investissements et les biens sauvegardés ;
- **API des biens immobiliers :** Exposition des informations sur les biens immobiliers ;
- **Intégration de Stripe :** Gestion des paiements.

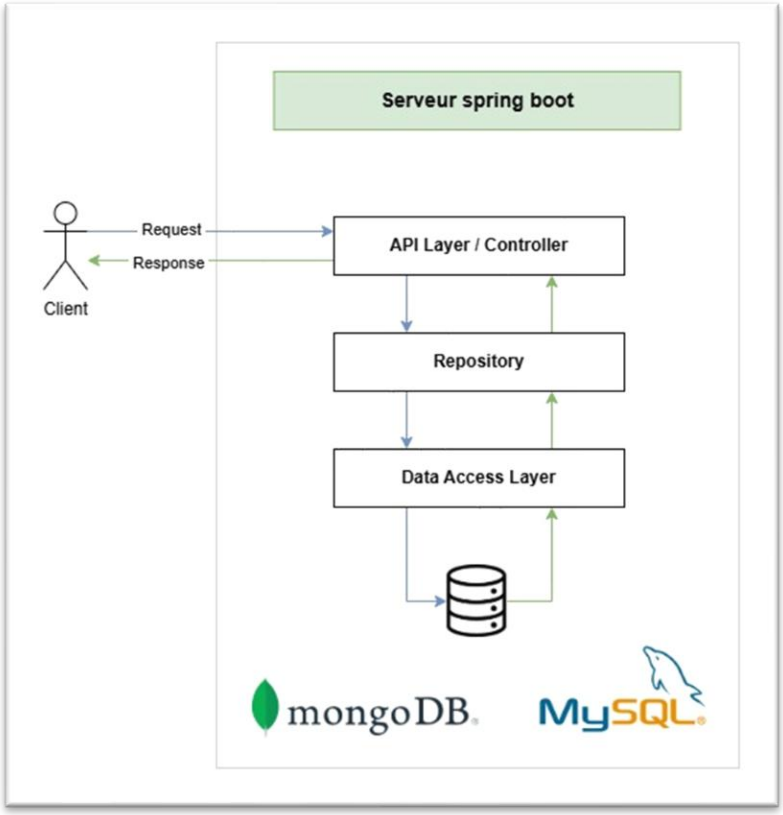

FIGURE 23: ARCHITECTURE DU BACKEND

Backend Architecture

L'architecture suivie est basée sur le modèle **MVC (Model-View-Controller)**, qui permet de séparer les responsabilités :

Model	Représente les entités métier comme Property, User, SavedProperty, etc.
Controller	Gère les requêtes HTTP, expose les endpoints RESTful.
Repository	Interface avec la base de données.

TABLEAU 3: BACKEND ARCHITECTURE

Cette architecture permet une bonne séparation des responsabilités, facilitant la maintenance et les évolutions futures.

Connexion à la base de données

MYSQL

La base de données relationnelle MySQL est utilisée pour toutes les entités structurées et transactionnelles :

- users : Informations utilisateur (fullname, email, mot de passe) ;
- user_wallet : Suivi des portefeuilles utilisateurs ;
- transactions : Historique des opérations financières ;
- investments : Enregistrements des investissements réalisés.

L'accès à ces données est facilité par **Spring Data JPA**, qui permet d'effectuer des opérations CRUD (Create, Read, Update, Delete) sans écrire de requêtes SQL manuelles.

MONGODB

MongoDB est utilisé pour stocker des données non relationnelles, notamment les informations sur les biens immobiliers, avec une plus grande souplesse de structure :

- properties : Informations principales sur les biens ;
- saved_properties : Biens enregistrés par les utilisateurs ;
- liked_properties : Biens aimés par les utilisateurs.

L'intégration se fait via **Spring Data MongoDB**, qui simplifie l'interaction avec la base NoSQL.

Gestion des paiements avec Stripe

Pour permettre aux utilisateurs d'accéder à des fonctionnalités premium, le backend intègre Stripe, une plateforme sécurisée de paiement en ligne.

FONCTIONNEMENT GLOBAL :

Création d'une session de paiement	Un endpoint backend (/api/payment/create-session) utilise le SDK Stripe pour créer une **session Checkout**.
Redirection vers l'interface de paiement Stripe	L'URL renvoyée est utilisée par le frontend pour rediriger l'utilisateur vers une interface de paiement sécurisée.
Enregistrement dans MySQL	Les détails du paiement (utilisateur, montant, date) sont enregistrés dans la table subscriptions.

TABLEAU 4: FONCTIONNEMENT GLOBAL DE STRIPE

FIGURE 24: COMMENT FONCTIONNE STRIPE ?

II. Développement de l'interface web

L'interface web constitue la principale porte d'entrée vers la plateforme d'investissement immobilier. Elle permet aux utilisateurs d'explorer les biens disponibles, de gérer leur compte et de suivre leurs investissements de manière fluide et sécurisée.

a. Technologies utilisées

React.js	Framework JavaScript moderne, utilisé pour construire une interface utilisateur dynamique et réactive.
Redux	Pour la gestion centralisée de l'état de l'application, en particulier pour l'authentification, les propriétés sauvegardées et les données de l'utilisateur.
Axios	Client HTTP pour la communication entre le frontend et le backend (API REST Spring Boot).
React Router	Pour la navigation entre les différentes pages de l'application.
Tailwind CSS	Framework CSS utilitaire pour une personnalisation rapide et responsive du design.
React icons	Bibliothèque d'icônes modernes intégrée dans les composants (ex. : cœur pour "aimer", étoile, panier).

TABLEAU 5: TECHNOLOGIES UTILISEES

b. Fonctionnalités principales

Inscription / Connexion utilisateur	• Formulaires réactifs avec validation des champs. • Authentification via JWT.
Tableau de bord utilisateur	• Visualisation du portefeuille, des investissements, et des transactions. • Accès aux biens sauvegardés et aimés.
les biens immobiliers	• Affichage des biens. • Consultation des détails de chaque propriété.
Fonction "Sauvegarder" ou "Aimer" un bien	• Interaction en temps réel avec MongoDB. • Suivi personnalisé par utilisateur.
Paiement via Stripe	• Intégration avec l'API Stripe. • Redirection sécurisée vers la page de paiement.

TABLEAU 6: FONCTIONNALITES PRINCIPALES

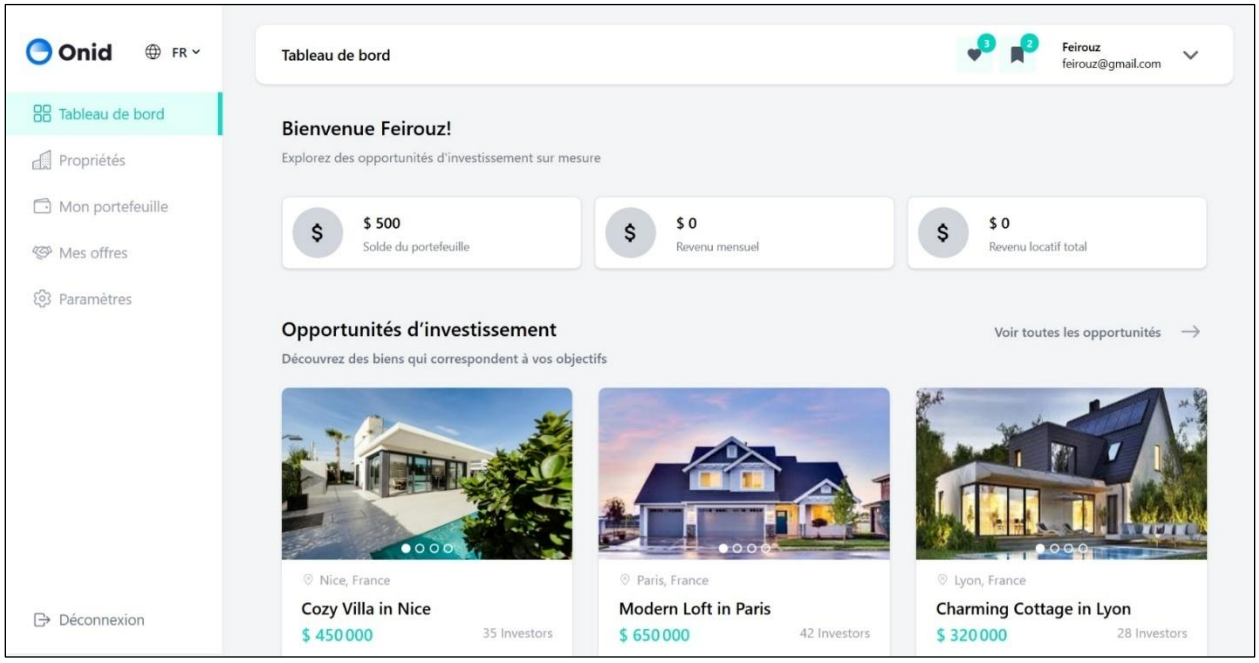

FIGURE 25: CAPTURE D'ECRAN " TABLEAU DE BORD "

III. Développement de l'interface mobile

a. Objectifs de l'application mobile

- Permettre aux utilisateurs de parcourir le catalogue de films et séries.
- Offrir une lecture fluide du contenu en streaming.
- Gérer les profils utilisateurs, listes de favoris et historiques de visionnage.
- Synchroniser les données avec la version web grâce à une architecture unifiée.

b. Architecture de l'application

L'application mobile adopte une architecture basée sur les composants, suivant les bonnes pratiques de séparation des responsabilités :

- **Pages principales** : Accueil, Détail du contenu, Recherche, Profil.
- **Composants réutilisables** : Carrousel de films, cartes de contenu, lecteur vidéo intégré.
- **Store Redux** : contient les informations utilisateur, la liste des contenus, les favoris, etc.
- **Appels API** : sécurisés avec token JWT et interfacés avec le backend via Axios.

c. Fonctionnalités principales

React Native	Le développement multiplateforme Android/iOS.
Axios	Client HTTP pour la communication entre le frontend et le backend (API REST Spring Boot).
React Navigation	Pour gérer la navigation entre les différentes pages.
Expo	Pour accélérer le développement et les tests.

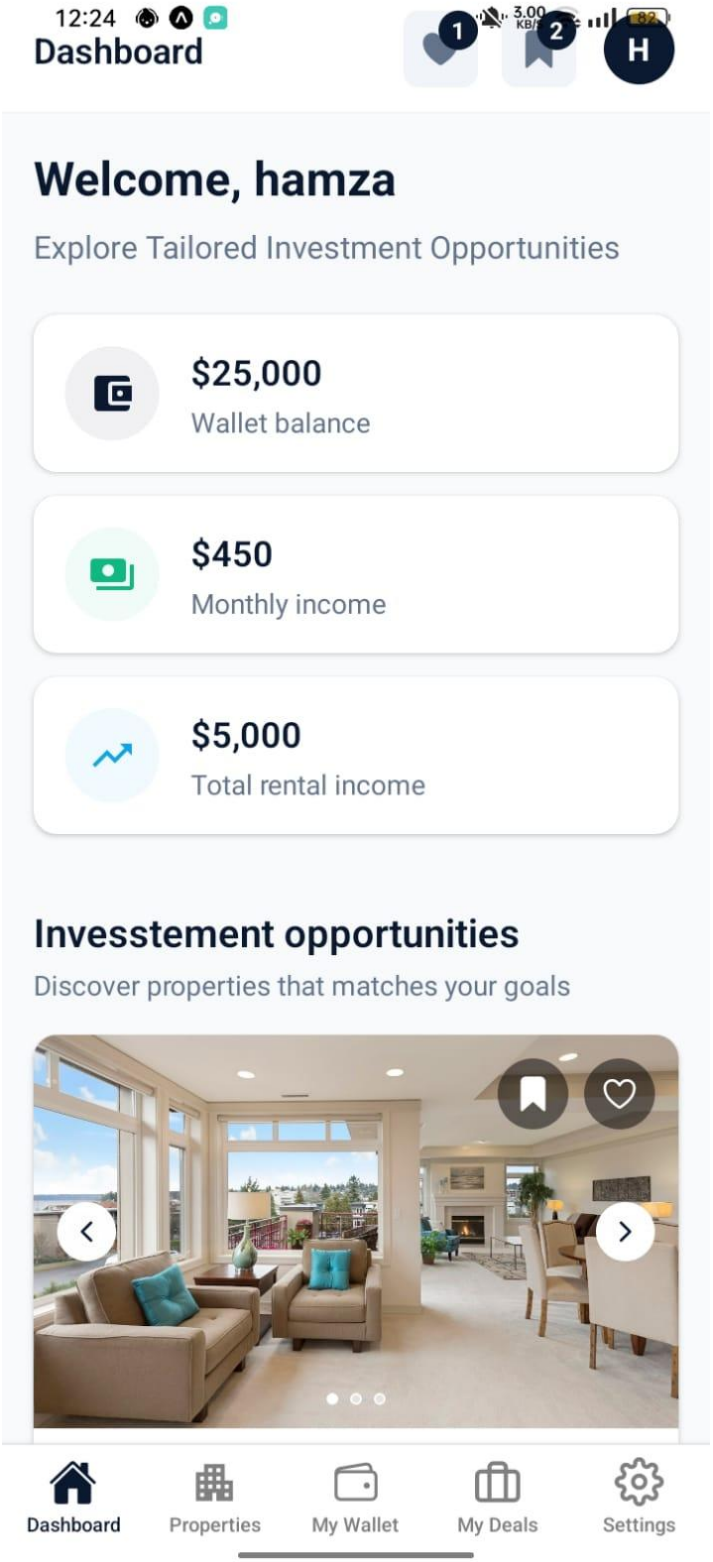

FIGURE 26: DASHBOARD MOBILE

IV. Infrastructure Cloud

Le projet est déployé sur AWS en utilisant plusieurs services pour assurer la disponibilité, la scalabilité, et la performance de l'application :

Amazon EC2	Utilisé pour héberger le backend Spring Boot. Deux instances EC2 sont déployées dans différentes zones de disponibilité pour assurer la redondance et la haute disponibilité.
AWS Elastic Load Balancer	Assure la répartition du trafic entre plusieurs instances EC2 pour équilibrer la charge et gérer les pics de trafic.
AWS S3	Stockage des fichiers images (affiches des biens immobiliers).
AWS CloudFront	Distribution des contenus (vidéo et images) via un CDN (Content Delivery Network) pour minimiser la latence et offrir une expérience de streaming fluide, même avec des utilisateurs répartis géographiquement.
AWS RDS et AWS DocumentDB	Hébergement des bases de données MySQL et MongoDB respectivement.

Défis rencontrés :

Optimisation des coûts : Ajustement des services utilisés et des configurations pour minimiser les coûts tout en assurant une performance optimale.

Conclusion

L'implémentation de la plateforme repose sur une architecture bien structurée combinant un **frontend React**.js moderne et une **API backend en Spring Boot**, soutenue par une base de données hybride (relationnelle et NoSQL). Le frontend permet aux utilisateurs de gérer efficacement leurs investissements grâce à une interface claire et interactive. Le backend assure quant à lui la sécurité, la cohérence et la disponibilité des données, tout en facilitant les échanges entre les utilisateurs et la plateforme. Cette solution, conçue pour être scalable et maintenable, pose les bases d'une application fiable et extensible à d'autres types d'investissements à l'avenir.

Conclusion Générale et Perspectives

Ce projet de développement d'une plateforme d'investissement immobilier a constitué une expérience complète, mêlant **analyse métier, conception logicielle, mise en œuvre technique** et **déploiement cloud**. Il a permis de relever de nombreux défis techniques et organisationnels tout en aboutissant à une solution fonctionnelle, sécurisée et scalable.

Le Chapitre 1 a permis de présenter l'organisme d'accueil, son domaine d'activité, ainsi que les objectifs du projet. Cette première partie a posé le contexte général, en mettant en lumière le besoin d'une solution numérique moderne pour optimiser la gestion des investissements immobiliers.

Dans le Chapitre 2, nous avons mené une analyse approfondie du projet à travers des modélisations UML (cas d'utilisation, classes, séquences), une modélisation cloud adaptée aux exigences de disponibilité et de sécurité, ainsi qu'un suivi de projet agile via des sprints organisés avec Azure DevOps. Cette étape a jeté les bases solides de la solution à développer.

Le Chapitre 3 a exploré les aspects techniques du projet. Nous avons sélectionné et justifié les technologies utilisées : React avec Redux pour un frontend dynamique, Spring Boot pour un backend robuste, MySQL et MongoDB pour une gestion mixte des données relationnelles et non relationnelles, et Stripe pour les paiements. L'infrastructure repose sur des services AWS, assurant performance, résilience et scalabilité.

Le Chapitre 4 a détaillé la réalisation du projet, avec l'implémentation des différentes fonctionnalités : gestion des utilisateurs, création de portefeuilles d'investissement, affichage des biens, simulation d'investissement, intégration du paiement sécurisé, et gestion des favoris. Une attention particulière a été portée à la sécurité des données et à l'expérience utilisateur.

En conclusion, ce projet a permis de développer une application complète et moderne, répondant aux exigences fonctionnelles du secteur de l'investissement immobilier. Il a également constitué une opportunité de renforcer mes compétences en **développement full-stack**, **gestion de projet agile**, **intégration cloud** et **sécurité des données**.

Perspectives :

- **Ajout de fonctionnalités avancées** : système de recommandations basé sur le profil d'investissement, alertes sur les nouveaux biens disponibles, messagerie intégrée avec les gestionnaires de projets immobiliers.
- **Optimisation des performances** : mise en cache via Redis pour accélérer les accès aux données fréquentes, amélioration des temps de réponse via des optimisations backend.
- **Renforcement de la sécurité** : authentification multifactorielle (2FA), chiffrement complet des données sensibles, audit de sécurité régulier.

Annexe

I. Documentation interactive avec Swagger

Pour permettre aux développeurs et testeurs d'interagir facilement avec l'API REST du backend, une documentation interactive a été mise en place grâce à Swagger (via Springdoc OpenAPI). Cette interface permet de visualiser les endpoints, leurs paramètres, leurs réponses, et même de tester les requêtes directement depuis le navigateur [21].

Lien d'accès local:

http://localhost:8080/swagger-ui.html

Fonctionnalités:

- Visualisation de tous les endpoints de l'API (GET, POST, PUT, DELETE, etc.) ;
- Description des paramètres et des réponses ;
- Test direct des appels API (avec body, headers, etc.) ;
- Support des statuts de réponse HTTP (200, 400, 404, 500…).

Exemples:

FIGURE 27: CAPTURE D'ECRAN DE SWAGGER

Références

[1] « What is Unified Modeling Language (UML)? » Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
[2] « What is Unified Modeling Language (UML)? » Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/
[3] « What Is Cloud Computing? | Microsoft Azure ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing
[4] « What is a Gantt Chart? Gantt Chart Software, Information, and History ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://www.gantt.com/
[5] « What Is Agile Methodology? (A Beginner's Guide) [2025] • Asana ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://asana.com/resources/agile-methodology
[6] « React – A JavaScript library for building user interfaces ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://legacy.reactjs.org/
[7] « What Is React Native? Complex Guide for 2024 ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://www.netguru.com/glossary/react-native
[8] ideo- www.ideo.pl, « React Native – What is Expo and is it worth using? / Digitization blog | Project-based Software Development ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://www.ideosoftware.com//blog/react-native-what-is-expo-and-is-it-worth-using,275.html
[9] « What is JavaScript? - Learn web development | MDN ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/What_is_JavaScript
[10] « HyperText Markup Language (HTML): What It Is and How It Works », Investopedia. Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://www.investopedia.com/terms/h/html.asp
[11] « CSS Introduction ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://www.w3schools.com/css/css_intro.asp
[12] « Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. » Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://tailwindcss.com/
[13] « What is Java and why do I need it? » Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://www.java.com/en/download/help/whatis_java.html
[14] « Spring Boot », Spring Boot. Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://spring.io/projects/spring-boot
[15] « MySQL: Understanding What It Is and How It's Used ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://www.oracle.com/mysql/what-is-mysql/
[16] « What Is MongoDB? | MongoDB ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://www.mongodb.com/company/what-is-mongodb
[17] « Stripe | Financial Infrastructure to Grow Your Revenue ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://stripe.com/
[18] « Free Cloud Computing Services - AWS Free Tier ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://aws.amazon.com/free/?trk=15faae9b-ab87-4e8f-8946-c46e8264e383&sc_channel=ps&ef_id=Cj0KCQjw8vvABhCcARIsAOCfwwomdxqA0ICCxqd91wNekm0pg0ziSeISgtXTyk076PA_dx5ou0XZoIwaAudGEALw_wcB:G:s&s_kwcid=AL!4422!3!645208863529!e!!g!!aws!19572078132!145087520813&gad_campaignid=19572078132&gbraid=0AAAADjHtp8-FJFN_gNnKVHnPDj1UW1Y5&gclid=Cj0KCQjw8vvABhCcARIsAOCfwwomdxqA0ICCxqd91wNekm0pg0ziSeISgtXTyk076PA_dx5ou0XZoIwaAudGEALw_wcB&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all

[19] « Git - What is Git? » Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F
[20] chcomley, « What is Azure DevOps - Azure DevOps ». Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://learn.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops
[21] « What is Swagger », Swagger Docs. Consulté le: 10 mai 2025. [En ligne]. Disponible sur: https://swagger.io/docs/specification/v2_0/what-is-swagger/