École Marocaine des Sciences de l'Ingénieur

Rapport de Projet

FlyHigh

Plateforme de réservation de vols en ligne

Présenté par :

Wissal Lamriss
Mohammed Amine Bellar

Année universitaire 2024 – 2025

Remerciements

Nous tenons à exprimer nos remerciements les plus sincères à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce projet :

À M. Ahmed Azouaoui, notre encadrant, pour sa disponibilité, ses conseils précieux et son accompagnement bienveillant tout au long du PFA.

À notre équipe pédagogique, pour l'enseignement de qualité et les compétences transmises durant notre formation.

À nos camarades de classe, pour leurs retours constructifs, leur entraide, et l'esprit collaboratif.

À nos proches et familles, pour leur soutien moral, leur patience, et leur compréhension durant les moments intenses du projet.

Table des matières

Introduction

Dans un monde où la digitalisation transforme profondément les habitudes de consommation et les attentes des usagers, le secteur du transport aérien n'échappe pas à cette évolution. En particulier, les compagnies aériennes doivent désormais proposer des services en ligne efficaces, intuitifs et sécurisés, afin d'attirer et fidéliser une clientèle de plus en plus connectée et exigeante. C'est dans ce contexte que s'inscrit notre projet intitulé : « Fly High – Plateforme Digitale de Gestion d'une Compagnie Aérienne ». L'objectif est de concevoir et développer une application web complète permettant aux utilisateurs de consulter les compagnies disponibles, de réserver des vols, d'effectuer leurs paiements en ligne, et de recevoir leurs factures numériques. En parallèle, une interface d'administration permet aux gestionnaires de superviser les vols, les passagers, les sièges, et les données associées de manière centralisée. Cette solution vise à offrir une expérience fluide et performante pour les passagers, tout en garantissant aux administrateurs un outil de gestion moderne, sécurisé et facile à prendre en main.

Le présent rapport est structuré autour de quatre chapitres principaux :
Chapitre 1 : Présentation du cadre de projet – il expose le contexte général du secteur aérien, une étude critique de l'existant, ainsi que la solution proposée.
Chapitre 2 : Spécification des besoins – il définit les besoins fonctionnels et non fonctionnels du système, illustrés par des cas d'utilisation.
Chapitre 3 : Conception du système – il présente la modélisation UML (diagrammes de classes, de séquence, etc.), l'architecture logicielle et matérielle.
Chapitre 4 : Réalisation du système – il décrit l'environnement de développement, les choix techniques, ainsi que les interfaces réalisées.

Ce travail aboutira à un prototype fonctionnel de la plateforme Fly High, prêt à être déployé et amélioré dans un contexte professionnel.

Chapitre 1

Contexte du Projet

1.1 Introduction

Le secteur du transport aérien est en constante évolution, stimulé par l'essor du numérique et les nouvelles attentes des voyageurs en matière de simplicité, rapidité et accessibilité. Dans un monde ultra-connecté, les compagnies aériennes doivent s'adapter pour offrir une expérience client fluide, depuis la réservation du billet jusqu'à l'embarquement. Le projet Fly High s'inscrit dans cette dynamique en proposant une plateforme web moderne dédiée à la gestion des réservations de vols et à l'administration globale d'une compagnie aérienne.

1.2 But du projet

L'objectif principal de ce projet est de concevoir et développer une plateforme web complète permettant aux utilisateurs de :

- Consulter les vols disponibles selon des critères de recherche (destination, date, etc.) ;
- Réserver un vol en ligne de manière simple, rapide et sécurisée ;
- Gérer leur profil et leurs réservations ;
- Effectuer des paiements via une passerelle intégrée (comme Stripe).

Sur le plan technique, le projet nous permet de :

- Mettre en œuvre le framework **Django** pour la gestion du backend (modèles, vues, API REST) ;
- Développer une interface utilisateur fluide et moderne avec **React** ;
- Concevoir une architecture modulaire et maintenable ;
- Appliquer une méthodologie de travail en équipe.

1.3 Problématiques

Le développement d'une application de réservation de vols en ligne soulève plusieurs problématiques :

- **Ergonomie et expérience utilisateur** : comment offrir une interface intuitive et agréable à utiliser ?
- **Fiabilité des données** : comment s'assurer de la cohérence et de la disponibilité des informations de vol ?
- **Sécurité des transactions** : comment sécuriser les paiements en ligne et la gestion des comptes utilisateurs ?
- **Performance et scalabilité** : comment garantir des temps de réponse rapides et une architecture évolutive ?
- **Intégration entre frontend et backend** : comment assurer une communication fluide entre l'API Django et l'interface React ?

1.4 Objectifs

Les objectifs de ce projet peuvent être répartis sur deux plans : pédagogique et technique.

Objectifs pédagogiques

- Appliquer les concepts enseignés dans le module Django à travers un projet complet ;
- Renforcer nos compétences en développement backend (Django) et frontend (React) ;
- Travailler en collaboration, organiser les tâches, respecter les délais et les étapes d'un cycle de développement.

Objectifs techniques

- Concevoir une base de données relationnelle robuste avec Django ORM ;
- Développer une API RESTful sécurisée et performante ;
- Implémenter un frontend interactif avec React, utilisant Tailwind CSS et React Router ;
- Gérer l'authentification, les rôles utilisateurs et l'intégration d'un interface de paiement.

1.5 Conclusion du chapitre

Le projet **FlyHigh** s'inscrit dans un contexte concret et porteur, celui de la digitalisation des services de réservation aérienne. En identifiant les enjeux techniques et les attentes des utilisateurs, ce chapitre nous a permis de poser les bases de notre démarche de conception. Dans les prochains chapitres, nous détaillerons le cahier des charges, l'architecture technique, les choix de développement, ainsi que les résultats obtenus.

Chapitre 2

Modélisation et Conception

2.1 Introduction

La phase de modélisation et de conception est essentielle pour structurer et clarifier les besoins fonctionnels du système **FlyHigh**. Elle permet de représenter graphiquement les interactions entre les utilisateurs et le système, ainsi que la structure interne des données et le déroulement des processus.

Dans ce chapitre, nous présentons les principaux diagrammes UML utilisés pour formaliser les exigences et la conception du projet : le diagramme de cas d'utilisation, le diagramme de classes et le diagramme de séquence.

2.2 Diagramme de cas d'utilisation (Use Case)

Le diagramme de cas d'utilisation illustre les fonctionnalités offertes par le système **FlyHigh** ainsi que les interactions avec les différents acteurs (utilisateur et administrateur).

FIGURE 2.1 – Diagramme de cas d'utilisation du système FlyHigh

Use Case	**Acteur**	**Message / Action Émise**
Rechercher un vol	Utilisateur	L'utilisateur cherche des vols disponibles
Réserver un vol	Utilisateur	L'utilisateur réserve un vol (inclut l'authentification)
Payer un vol	Utilisateur	L'utilisateur effectue le paiement d'un vol réservé
Envoyer un message à l'administrateur	Utilisateur	L'utilisateur envoie une requête ou message
Gérer les utilisateurs	Administrateur	L'administrateur gère la liste des utilisateurs
Gérer les compagnies	Administrateur	Étend : Ajouter / Modifier / Supprimer une compagnie
→ Ajouter une compagnie	Administrateur	Ajout d'une compagnie aérienne
→ Modifier une compagnie	Administrateur	Modification des infos d'une compagnie
→ Supprimer une compagnie	Administrateur	Suppression d'une compagnie aérienne
Gérer les vols	Administrateur	Étend : Ajouter / Modifier / Supprimer un vol
→ Ajouter un vol	Administrateur	Création d'un vol
→ Modifier un vol	Administrateur	Mise à jour d'un vol
→ Supprimer un vol	Administrateur	Suppression d'un vol
Voir les réservations	Administrateur	Consultation des réservations effectuées
Consulter les paiements	Administrateur	Voir les détails des paiements effectués
Consulter les messages des clients	Administrateur	Lire les messages envoyés par les utilisateurs

TABLE 2.1 – Description des cas d'utilisation de FlyHigh

2.3 Diagramme de classes

Le diagramme de classes permet de représenter la structure statique du système **FlyHigh**, en montrant les différentes entités (classes), leurs attributs, leurs méthodes, ainsi que les relations qui existent entre elles.

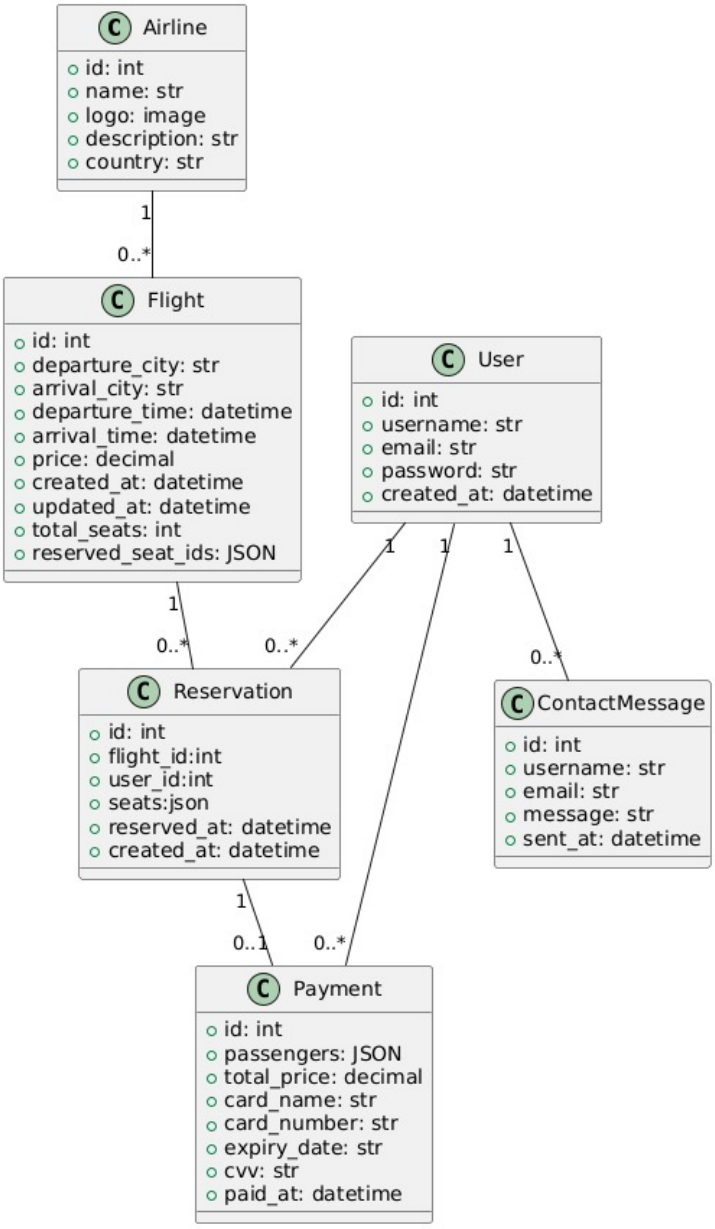

FIGURE 2.2 – Diagramme de classes du système FlyHigh

Classe	Attributs	Relations (associations)	Multiplicité
Airline	id, name, logo, description, country	Contient plusieurs **Flight**	1 **Airline** → 0..* **Flight**
Flight	id, departure_city, arrival_city, departure_time, arrival_time, price, created_at, updated_at, total_seats, reserved_seat_ids	Associée à 1 **Airline** Associée à plusieurs **Reservation**	1 **Flight** → 0..* **Reservation**
User	id, username, email, password, created_at	Peut faire plusieurs **Reservation** Peut envoyer plusieurs **ContactMessage**	1 **User** → 0..* **Reservation** 1 **User** → 0..* **ContactMessage**
Reservation	id, flight_id, user_id, seats, reserved_at, created_at	Liée à 1 **User**, 1 **Flight** Liée à 0..* **Payment**	1 **Reservation** → 0..* **Payment**
Payment	id, passengers, total_price, card_name, card_number, expiry_date, cvv, paid_at	Associée à une **Reservation**	1 **Reservation** → 0..* **Payment**
ContactMessage	id, username, email, message, sent_at	Liée à 1 **User** (par attributs)	1 **User** → 0..* **ContactMessage**

TABLE 2.2 – Description des classes du système FlyHigh

2.4 Diagrammes de séquence

Les diagrammes de séquence permettent de décrire la dynamique des interactions entre les objets du système au fil du temps. Ils illustrent l'ordre dans lequel les messages sont échangés lors de l'exécution d'un scénario particulier.

Dans ce projet, nous présentons deux scénarios typiques du système **FlyHigh** à travers des diagrammes de séquence.

1. Réservation d'un vol par un utilisateur

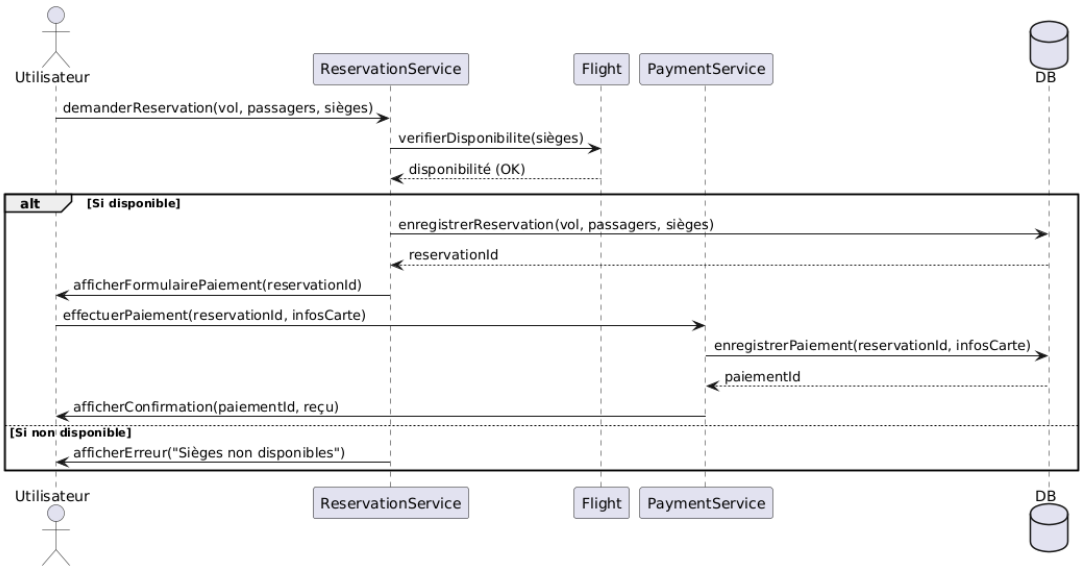

FIGURE 2.3 – Diagramme de séquence — Réservation d'un vol

Ce diagramme montre le processus de réservation d'un vol. L'utilisateur sélectionne un vol, s'authentifie si nécessaire, saisit les informations de réservation, puis le système enregistre la réservation dans la base de données.

2. Envoi d'un message

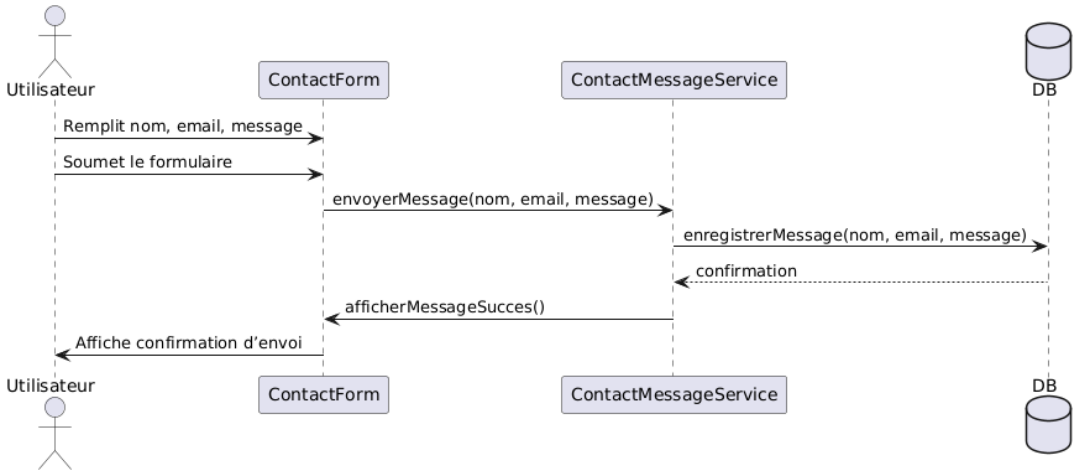

FIGURE 2.4 – Diagramme de séquence — Envoi d'un message au admin

Ce diagramme décrit le processus d'envoi d'un message au administrateur. L'utilisateur remplit le formulaire, saisit ses informations, et le système enregistre le message.

Conclusion

La phase de modélisation et de conception constitue une étape cruciale dans le développement du système **FlyHigh**. À travers les différents diagrammes UML présentés — diagramme de cas d'utilisation, diagramme de classes et diagrammes de séquence — nous avons pu formaliser les exigences fonctionnelles du système, organiser la structure des données, et anticiper les interactions principales entre les acteurs et le système.

Cette modélisation permet de mieux comprendre le fonctionnement global de l'application et sert de base solide pour les étapes suivantes, notamment l'implémentation et les tests. Elle garantit également une meilleure communication entre les membres de l'équipe et une documentation claire du projet.

Chapitre 3

Architecture du Projet

3.1 Introduction

L'architecture logicielle est un élément fondamental dans le développement d'une application performante, évolutive et maintenable. Dans le cadre du projet **FlyHigh**, nous avons adopté une architecture orientée services reposant sur une séparation claire entre les différentes couches de l'application : le front-end, le back-end et la base de données.

Ce choix permet non seulement une meilleure organisation du code, mais également une meilleure évolutivité du système, en facilitant les mises à jour, les tests, et l'intégration de nouvelles fonctionnalités.

Dans ce chapitre, nous présentons en détail l'architecture du projet FlyHigh, en décrivant les technologies utilisées, les interactions entre les différentes couches, ainsi que les outils et bibliothèques adoptés pour garantir robustesse et fluidité dans le fonctionnement du système.

3.2 Vue d'ensemble de l'architecture

L'application FlyHigh repose sur une architecture en trois couches principales :

— **Frontend** : développé avec *React.js* et *Tailwind CSS*, il offre une interface utilisateur moderne et réactive.

— **Backend** : construit avec *Django REST Framework*, il gère la logique métier, les contrôleurs d'API et les traitements côté serveur.

— **Base de données** : assurée par *MySQL*, elle stocke l'ensemble des informations nécessaires au bon fonctionnement du système (utilisateurs, vols, paiements, réser-

vations, etc.).

3.3 Schéma de l'architecture générale

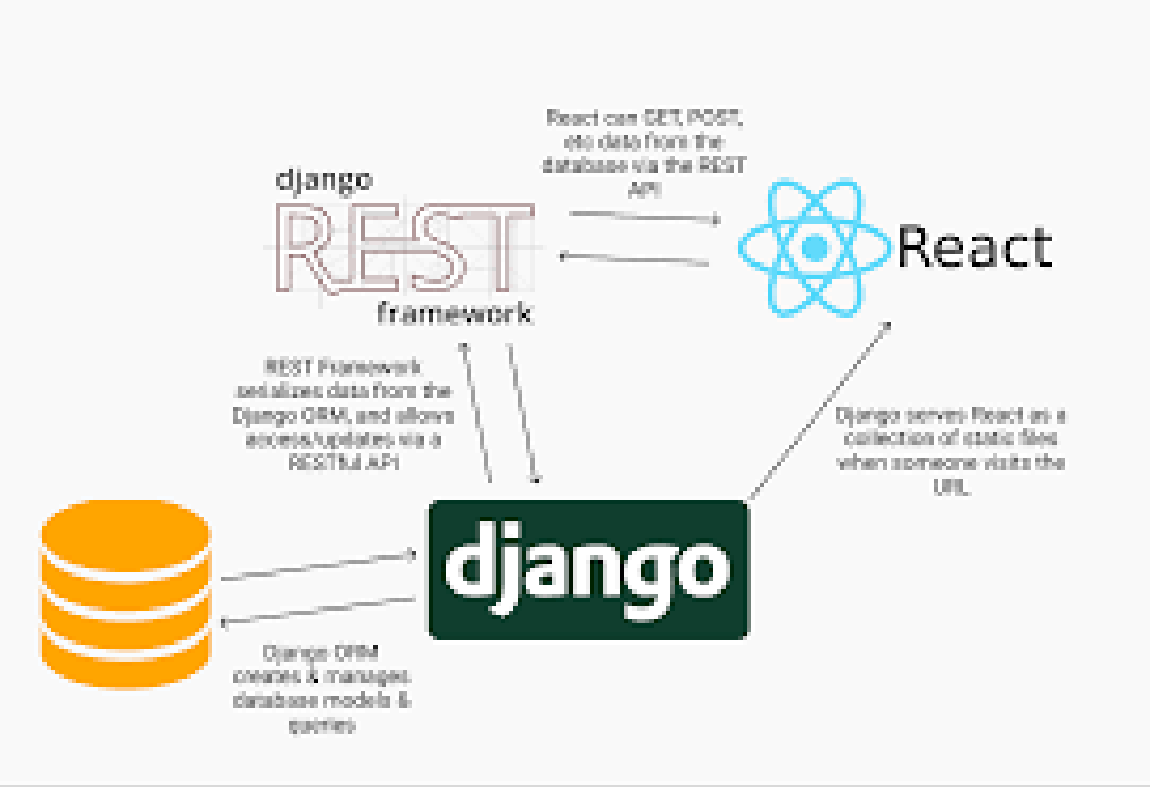

FIGURE 3.1 – Schéma global de l'architecture de FlyHigh

3.4 Détail des composants

3.4.1 Frontend

Le frontend constitue l'interface visible par l'utilisateur final. Grâce à **React.js**, nous avons pu créer une SPA (Single Page Application) avec une navigation fluide et rapide. **Tailwind CSS** a été utilisé pour un design moderne et réactif.

Le frontend interagit avec le backend via des appels API REST sécurisés. Il gère l'affichage des vols, la réservation, les formulaires de connexion/inscription, les messages de contact, ainsi que l'accès au tableau de bord administrateur.

3.4.2 Backend

Le backend, développé avec **Django REST Framework**, expose les API permettant de gérer toute la logique métier. Il est responsable de :

— L'authentification et la gestion des rôles (utilisateur, administrateur)

— La gestion des vols, compagnies, utilisateurs et paiements

— La sécurisation des données échangées via des permissions et des tokens (JWT ou session)

— Le traitement des réservations et des messages

3.4.3 Base de données

La base de données **MySQL** contient l'ensemble des entités du système. Elle est conçue pour garantir la cohérence et l'intégrité des données grâce à des relations bien définies entre les tables (voir chapitre précédent sur le diagramme de classes).

3.5 API RESTful

Les échanges entre le frontend et le backend sont basés sur une architecture **REST**. Chaque ressource (vols, utilisateurs, paiements, etc.) est accessible via une URL spécifique, avec des méthodes HTTP standard :

— `GET` : pour lire une ressource

— `POST` : pour créer une ressource

— `PUT/PATCH` : pour mettre à jour une ressource

— `DELETE` : pour supprimer une ressource

Conclusion

Cette architecture modulaire et bien découpée permet au projet FlyHigh d'être à la fois robuste, évolutif et facile à maintenir. Le découplage entre les composants favorise la réutilisabilité du code et simplifie le travail en équipe, en répartissant clairement les responsabilités entre le frontend, le backend et la base de données.

Chapitre 4

Mise en œuvre de l'application

4.1 Introduction

Cette partie du rapport décrit la mise en œuvre concrète de l'application **FlyHigh**, en se concentrant principalement sur les interfaces utilisateurs. Elle présente les différentes pages développées à l'aide de **React.js**, ainsi que les fonctionnalités associées à chaque vue.

L'objectif est de garantir une expérience utilisateur fluide, intuitive et cohérente avec les besoins fonctionnels définis lors de la phase de modélisation. Les interfaces sont également conçues pour être adaptatives (responsive) et facilement maintenables.

4.2 Interface client

L'interface client de l'application **FlyHigh** a été conçue pour offrir une navigation simple, fluide et agréable. Elle permet à l'utilisateur de rechercher, réserver et payer un vol, ainsi que de contacter l'administrateur pour toute assistance.

Voici les principales pages de l'interface client :