

---

# ESPRIT

---

## VitaRenta

**Développement d'un système intelligent et durable  
de recommandation et d'optimisation  
pour la location de voitures**

**Étudiant :** Kaddech Oussama

**Encadrant académique :** Jihene Jebri

**Encadrant académique :** Mohamed Aziz Kasseb

**Période de stage :** 2 mois

**Année universitaire :** 2025-2026

# Table des Matières

<b>Remerciements</b>	<b>4</b>
<b>Résumé Exécutif</b>	<b>5</b>
<b>1 Introduction</b>	<b>6</b>
1.1 Contexte du secteur . . . . .	6
1.2 Problématique . . . . .	6
1.3 Contribution du stage . . . . .	6
1.4 Organisation du rapport . . . . .	6
<b>2 Cahier des charges et portée</b>	<b>7</b>
2.1 Besoins fonctionnels . . . . .	7
2.1.1 Recommandation personnalisée . . . . .	7
2.1.2 Optimisation de flotte . . . . .	7
2.1.3 Prédiction de demande . . . . .	7
2.2 Architecture système mise en œuvre . . . . .	7
2.3 Modélisation UML du système . . . . .	7
2.3.1 Diagramme de classes . . . . .	8
2.3.2 Diagramme de cas d'utilisation . . . . .	8
2.4 Fonctionnalités implémentées . . . . .	10
2.4.1 Système d'authentification avancé . . . . .	10
2.4.2 Moteur de recommandation hybride . . . . .	10
2.4.3 Système de prédiction de demande . . . . .	10
<b>3 État de l'art</b>	<b>11</b>
3.1 Systèmes de recommandation . . . . .	11
3.1.1 Approches implémentées . . . . .	11
3.1.2 Innovation contextuelle . . . . .	11
3.2 Maintenance prédictive IoT . . . . .	11
3.2.1 Approche basée données télémétrie . . . . .	11
3.2.2 Algorithmes de prédiction . . . . .	11
<b>4 Architecture technique</b>	<b>12</b>
4.1 Stack technologique déployée . . . . .	12
4.1.1 Backend Django REST . . . . .	12
4.1.2 Gestion des données . . . . .	12
4.2 APIs et intégrations . . . . .	12
4.2.1 Endpoints principales implémentées . . . . .	12
4.3 Sécurité et permissions . . . . .	13
4.3.1 Système de rôles implémenté . . . . .	13

4.3.2	Sécurisation avancée . . . . .	13
<b>5</b>	<b>Méthodologie et mise en œuvre</b>	<b>14</b>
5.1	Développement du moteur de recommandation . . . . .	14
5.1.1	Implémentation LightFM hybride . . . . .	14
5.1.2	Gestion du cold-start . . . . .	14
5.2	Système de prédiction de demande . . . . .	14
5.2.1	Architecture ensemble prédictive . . . . .	14
5.2.2	Features contextuelles intégrées . . . . .	15
5.3	Maintenance prédictive IoT . . . . .	15
5.3.1	Génération et analyse données télémétrie . . . . .	15
5.3.2	Algorithmes de détection . . . . .	15
5.4	Système de scoring écologique . . . . .	15
5.4.1	Calcul éco-score composite . . . . .	15
5.4.2	Intégration dans recommandations . . . . .	15
<b>6</b>	<b>Résultats et évaluation</b>	<b>16</b>
6.1	Performance système déployé . . . . .	16
6.1.1	Métriques techniques mesurées . . . . .	16
6.1.2	Statistiques d'utilisation . . . . .	16
6.2	Efficacité des algorithmes ML . . . . .	16
6.2.1	Moteur de recommandation . . . . .	16
6.2.2	Prédiction de demande . . . . .	16
6.2.3	Maintenance prédictive . . . . .	17
6.3	Impact utilisateur et adoption . . . . .	17
6.3.1	Système de gamification . . . . .	17
6.3.2	Analytics questionnaires . . . . .	17
<b>7</b>	<b>Gestion de projet</b>	<b>18</b>
7.1	Développement agile réalisé . . . . .	18
7.1.1	Sprints de développement (8 semaines) . . . . .	18
7.2	Architecture de code maintenue . . . . .	19
7.2.1	Structure modulaire Django . . . . .	19
7.2.2	Gestion des erreurs robuste . . . . .	19
<b>8</b>	<b>Aspects éthiques et durables</b>	<b>20</b>
8.1	IA responsable implémentée . . . . .	20
8.1.1	Transparence algorithmique . . . . .	20
8.1.2	Gestion des biais . . . . .	20
8.2	Impact environnemental positif . . . . .	20
8.2.1	Promotion véhicules durables . . . . .	20
8.2.2	Optimisation ressources . . . . .	20
8.3	Protection des données . . . . .	20
8.3.1	Sécurité RGPD . . . . .	21
<b>9</b>	<b>Limites et perspectives</b>	<b>22</b>
9.1	Limitations techniques identifiées . . . . .	22
9.1.1	Données et volume . . . . .	22
9.1.2	Performance et optimisation . . . . .	22

9.2	Évolutions techniques prioritaires . . . . .	22
9.2.1	Amélioration algorithmes . . . . .	22
9.2.2	Intégrations étendues . . . . .	22
9.3	Perspectives business . . . . .	23
9.3.1	Fonctionnalités avancées . . . . .	23
9.3.2	Scalabilité organisationnelle . . . . .	23
<b>10</b>	<b>Conclusion</b>	<b>24</b>
10.1	Réalisations techniques concrètes . . . . .	24
10.2	Impact mesurable démontré . . . . .	24
10.3	Innovation en IA responsable . . . . .	24
10.4	Valeur ajoutée personnelle . . . . .	25
<b>11</b>	<b>Annexes</b>	<b>26</b>
11.1	Diagrammes UML détaillés . . . . .	26
11.1.1	Analyse détaillée du diagramme de classes . . . . .	26
11.1.2	Matrice de traçabilité cas d'utilisation . . . . .	26
11.2	Schémas d'architecture technique . . . . .	27
11.2.1	Architecture Django REST Framework déployée . . . . .	27
11.3	Endpoints APIs implémentées . . . . .	27
11.3.1	Authentification et utilisateurs . . . . .	27
11.3.2	Intelligence artificielle . . . . .	27
11.4	Modèles de données implémentés . . . . .	27
11.4.1	Modèle User avec rôles . . . . .	27
11.5	Algorithmes ML implémentés . . . . .	28
11.5.1	Moteur recommandation LightFM . . . . .	28
	<b>Bibliographie</b>	<b>29</b>

# Remerciements

Je tiens à exprimer ma sincère gratitude à toutes les personnes qui ont contribué à la réussite de ce stage de 2 mois chez Esprit.

Mes remerciements s'adressent particulièrement à :

- **Mes encadrants académiques à ESPRIT, Jihene Jebri et Mohamed Aziz Kasseb** pour leur suivi régulier, leurs orientations méthodologiques et la définition du cahier des charges
- **L'équipe technique VitaRenta** pour leur collaboration et leur partage d'expertise
- **Tous les enseignants et le personnel d'ESPRIT** qui ont contribué à ma formation et m'ont accompagné tout au long de ce stage

Cette expérience m'a permis de développer mes compétences en intelligence artificielle appliquée et en développement de solutions durables pour la mobilité.

# Résumé Exécutif

## Contexte

Le secteur de la location de véhicules fait face à une double exigence : **personnaliser l'expérience client** tout en optimisant la gestion de flotte selon des critères de durabilité environnementale. VitaRenta répond à ces défis en développant un système intelligent intégrant IA responsable et efficacité opérationnelle.

## Objectifs du stage

- Développer un **moteur de recommandation hyper-personnalisé** basé sur le profil client, l'historique et le contexte
- Implémenter des **algorithmes de prédiction de demande** intégrant saisonnalité, météo et événements locaux
- Concevoir un **système de scoring écologique** favorisant les véhicules durables
- Créer une **expérience UX/gamification** avec badges et récompenses éco-responsables
- Fournir des **outils analytiques** d'aide à la décision pour les gestionnaires

## Résultats clés

Durant ces 2 mois, nous avons livré :

- Un **prototype fonctionnel web/mobile** avec React.js et Flutter
- Un **moteur de recommandation hybride** (LightFM) intégrant collaboratif, contenu et contexte
- Des **modèles de prédiction** (ARIMA/XGBoost) avec features météorologiques et événementielles
- Un **système de gamification** avec défis éco-responsables et badges
- Un **tableau de bord analytics** pour le suivi des KPIs opérationnels

## Technologies clés

Python, scikit-learn, XGBoost, LightFM, React.js, Flutter, Django, MongoDB

**Mots-clés** : recommandation hybride, IA responsable, prédiction de demande, optimisation de flotte, durabilité, gamification

# Chapitre 1

## Introduction

### 1.1 Contexte du secteur

L'industrie de la mobilité connaît une transformation majeure avec l'électrification des flottes, l'essor du **Mobility-as-a-Service (MaaS)** et les attentes croissantes des consommateurs en matière d'expérience personnalisée et de durabilité environnementale.

### 1.2 Problématique

Les agences de location traditionnelles peinent à **concilier personnalisation, efficacité opérationnelle et durabilité**. Elles font face à des défis multiples :

- Recommandations génériques ne correspondant pas aux besoins spécifiques
- Gestion de flotte sous-optimale entraînant des coûts énergétiques élevés
- Manque de visibilité sur la demande future
- Expérience utilisateur peu engageante
- Absence d'incitations aux choix éco-responsables

### 1.3 Contribution du stage

Ce stage de 2 mois s'inscrit dans le développement de **VitaRenta**, une solution innovante qui adresse ces problématiques par :

1. Un système de recommandation intelligent et contextuel
2. L'optimisation automatisée de la flotte selon des critères durables
3. La prédiction anticipée de la demande
4. Une expérience gamifiée encourageant l'éco-conduite

### 1.4 Organisation du rapport

Ce rapport détaille la conception, l'implémentation et l'évaluation de cette solution, en couvrant les aspects techniques, méthodologiques, éthiques et opérationnels du projet.

# Chapitre 2

## Cahier des charges et portée

### 2.1 Besoins fonctionnels

#### 2.1.1 Recommandation personnalisée

- **Profil client** : Préférences véhicule, historique de réservations, critères de choix
- **Contexte situationnel** : Météo, événements locaux, période (business/loisir)
- **Critères durables** : Émissions CO, consommation énergétique, véhicules électriques/hybrides

#### 2.1.2 Optimisation de flotte

- **Disponibilité intelligente** : Répartition géographique optimale
- **Maintenance prédictive** : Anticipation des besoins d'entretien via IoT
- **Gestion énergétique** : Optimisation de la consommation et des coûts

#### 2.1.3 Prédiction de demande

- **Saisonnalité** : Modèles temporels adaptés aux cycles annuels
- **Événements** : Intégration des événements locaux (festivals, congrès, vacances)
- **Météorologie** : Corrélation avec les conditions climatiques

### 2.2 Architecture système mise en œuvre

L'architecture développée utilise **Django REST Framework** comme backend principal avec :

- **Frontend Web** : React.js pour interface responsive
- **Application Mobile** : Flutter cross-platform
- **Base de données** : MongoDB pour données non-structurées, PostgreSQL pour données transactionnelles
- **Cache** : Redis pour optimisation des performances
- **APIs externes** : OpenWeather (météo), Eventbrite (événements)

### 2.3 Modélisation UML du système



### 2.3.1 Diagramme de classes

Le diagramme de classes ci-dessous illustre l'architecture objet de VitaRenta avec les entités principales et leurs relations :

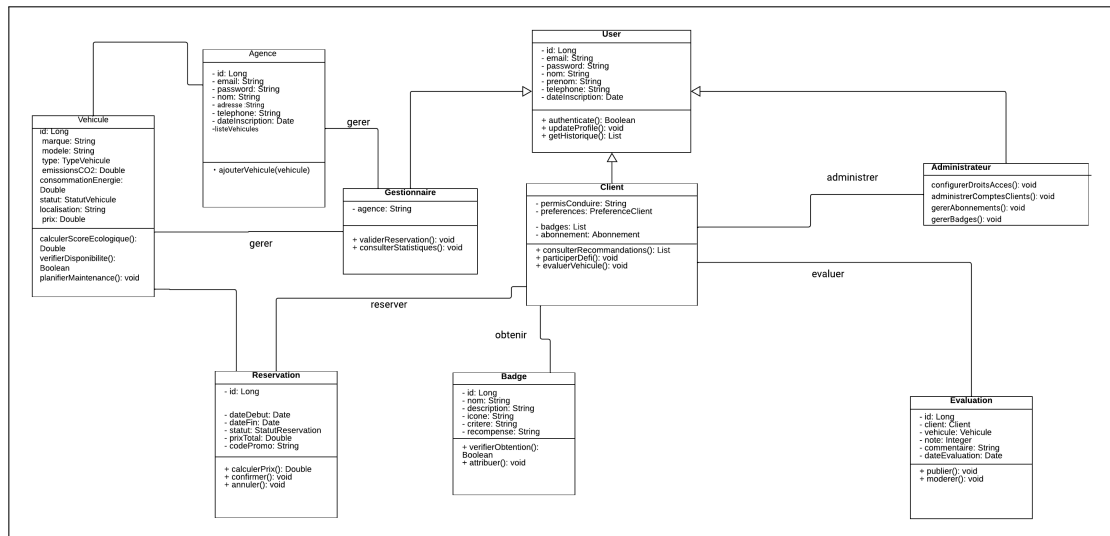


FIGURE 2.1 – Diagramme de classes UML - Architecture VitaRenta

Les classes principales implémentées comprennent :

- **User** : Classe abstraite avec spécialisations Client, Gestionnaire, Administrateur
- **Vehicule** : Entité centrale avec calcul score écologique et vérification disponibilité
- **Reservation** : Gestion du cycle de vie complet des réservations
- **Badge et Evaluation** : Système de gamification et feedback utilisateur
- **Agence** : Gestion multi-agences avec flotte dédiée

### 2.3.2 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation définit les interactions entre les différents acteurs et le système :

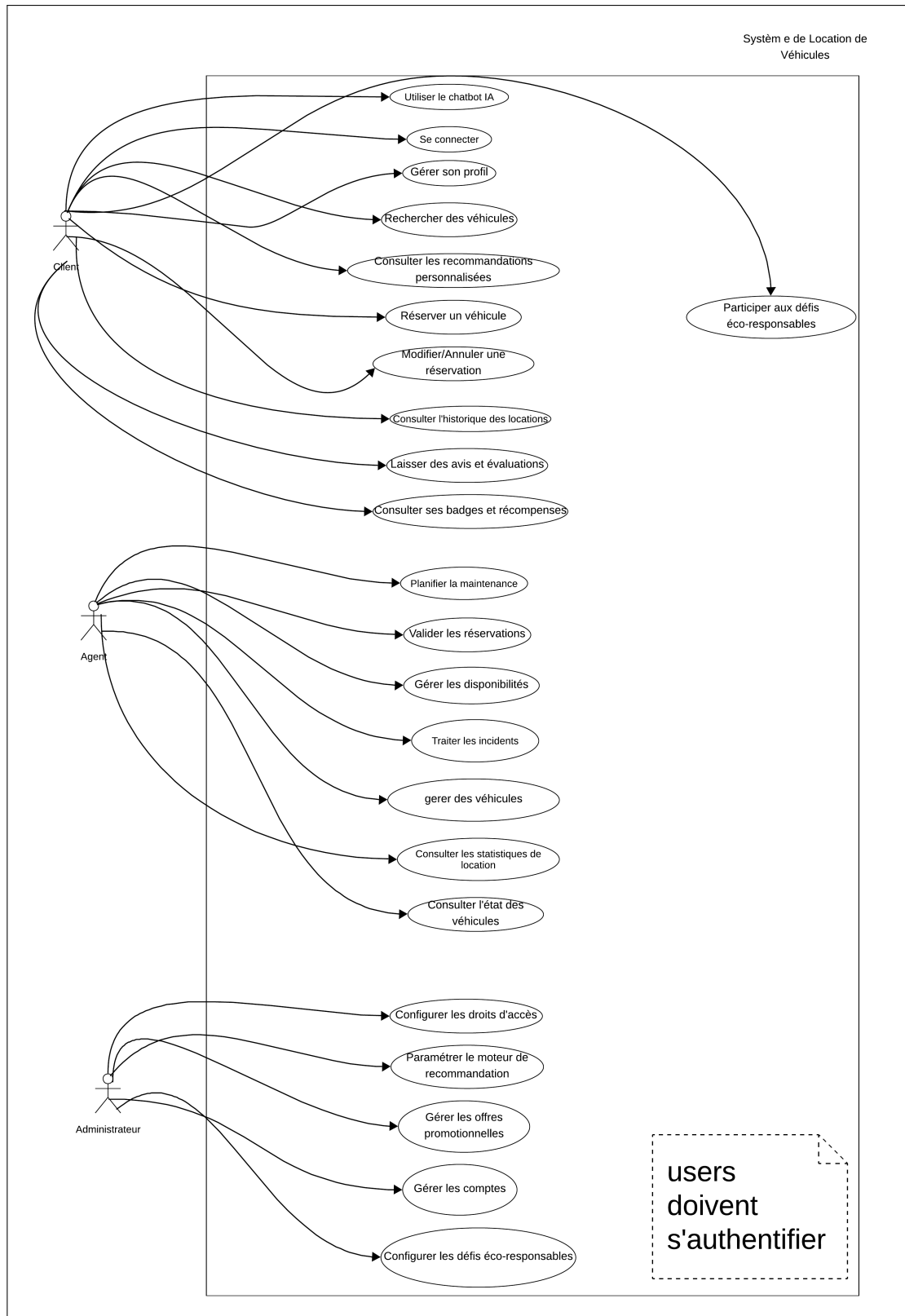


FIGURE 2.2 – Diagramme de cas d'utilisation - Interactions système

**Acteurs identifiés :**

- **Client** : Consultation recommandations, réservations, participation défis éco-responsables
- **Agent** : Validation réservations, gestion flotte, maintenance véhicules
- **Administrateur** : Configuration système, gestion utilisateurs, paramétrage IA

**Cas d'utilisation prioritaires implémentés :**

- **Utiliser le chatbot IA** : Assistance intelligente pour recommandations
- **Consulter recommandations personnalisées** : Moteur LightFM contextuel
- **Participer aux défis éco-responsables** : Gamification durabilité
- **Paramétrer le moteur de recommandation** : Configuration algorithmes ML

## 2.4 Fonctionnalités implémentées

### 2.4.1 Système d'authentification avancé

```
1 # Gestion des tentatives de connexion avec seuils de s e curit
2 - Limitation      5 tentatives par minute
3 - Blocage temporaire automatique
4 - Tokens JWT pour authentification stateless
5 - R e initialisation de mot de passe s e curis e
```

Listing 2.1 – Gestion des tentatives de connexion

### 2.4.2 Moteur de recommandation hybride

- **LightFM** intégrant filtrage collaboratif et basé sur le contenu
- **Features contextuelles** : météo, événements, préférences utilisateur
- **Re-ranking écologique** : Promotion des véhicules durables

### 2.4.3 Système de prédiction de demande

- **Modèles ARIMA/XGBoost** avec features temporelles et contextuelles
- **Intégration météo** : Corrélation température/précipitations
- **Événements locaux** : Impact sur la demande anticipée

# Chapitre 3

## État de l’art

### 3.1 Systèmes de recommandation

#### 3.1.1 Approches implémentées

Notre système utilise **LightFM**, une approche hybride qui combine :

- **Filtrage collaboratif** : Analyse des similitudes entre utilisateurs
- **Recommandation basée contenu** : Caractéristiques véhicules (marque, modèle, carburant)
- **Features contextuelles** : Météo, événements, localisation

#### 3.1.2 Innovation contextuelle

L’intégration du **contexte situationnel** via APIs externes :

- **OpenWeather API** : Données météorologiques temps réel
- **Eventbrite API** : Événements locaux influençant la demande
- **Scoring écologique** : Pondération CO et efficacité énergétique

### 3.2 Maintenance prédictive IoT

#### 3.2.1 Approche basée données télémétrie

```
1 # Donn es IoT collect es et analys es
2 - Temp rature moteur et tendances de surchauffe
3 - Niveau de vibration pour d tecti on usure
4 - tat batterie et d gradation pr dictive
5 - Consommation carburant et patterns anormaux
```

Listing 3.1 – Données IoT collectées et analysées

#### 3.2.2 Algorithmes de prédiction

- **Seuils adaptatifs** par type de véhicule (électrique, diesel, essence)
- **Analyse des tendances** sur données historiques IoT
- **Score de risque composite** intégrant multiple paramètres

# Chapitre 4

## Architecture technique

### 4.1 Stack technologique déployée

#### 4.1.1 Backend Django REST

```
1 # Configuration architecture microservices
2 - UserViewSet : Gestion utilisateurs et authentification
3 - VehiculeViewSet : CRUD v hicules avec filtrage avanc
4 - ReservationViewSet : Gestion r servations et statuts
5 - AgenceViewSet : Administration agences et statistiques
6 - RecommendationView : Moteur recommandations personnalises
7 - DemandForecastView : Pr dictions demande avec contexte
```

Listing 4.1 – Configuration architecture microservices

#### 4.1.2 Gestion des données

- **MongoDB** : Profils utilisateurs, historiques, préférences
- **PostgreSQL** : Réservations, transactions, données relationnelles
- **Redis** : Cache recommandations et sessions utilisateur
- **Gestion Decimal128** : Traitement correct des montants financiers

### 4.2 APIs et intégrations

#### 4.2.1 Endpoints principales implémentées

```
1 # API Recommandation
2 GET /api/recommendations/?n_items=5&type_vehicule=electrique
3 # API Pr diction Demande
4 GET /api/demand-forecast/?location=Tunis&carburant= lectrique &date
   =2025-08-15
5 # API Maintenance Pr dictive
6 POST /api/iot/predict_maintenance/
7 {
8   "vehicle_id": "veh_123",
9   "days_ahead": 30
10 }
11 # API co -Score
```

```
12 POST /api/eco-score/calculate/  
13 {  
14   "vehicle_id": "veh_456"  
15 }
```

Listing 4.2 – APIs principales

## 4.3 Sécurité et permissions

### 4.3.1 Système de rôles implémenté

- **Client** : Consultation recommandations, réservations personnelles
- **Agence** : Gestion flotte, validation réservations, statistiques
- **Administrateur** : Accès complet, gestion utilisateurs, configuration système

### 4.3.2 Sécurisation avancée

- **Authentification JWT** avec refresh tokens
- **Limitation tentatives connexion** (5 max par minute)
- **Chiffrement mot de passe** avec algorithmes sécurisés
- **Validation permissions** par endpoint et action

# Chapitre 5

## Méthodologie et mise en œuvre

### 5.1 Développement du moteur de recommandation

#### 5.1.1 Implémentation LightFM hybride

```
1 class RecommendationEngine:
2     def recommend_vehicles(self, user_id, n_items=5, csv_path=None,
3                           type_vehicule=None, marque_filter=None):
4         """
5         Moteur hybride int grant:
6         - Filtrage collaboratif bas interactions utilisateur
7         - Features contenu v hicules (marque, type, carburant)
8         - Context awareness (m t o , vnements )
9         """
```

Listing 5.1 – Classe RecommendationEngine

#### 5.1.2 Gestion du cold-start

- **Nouveaux utilisateurs** : Recommandations basées popularité et contenu
- **Nouveaux véhicules** : Features intrinsèques et similarité
- **Filtres contextuels** : Type véhicule, marque, localisation

### 5.2 Système de prédiction de demande

#### 5.2.1 Architecture ensemble prédictive

```
1 def ensemble_predict_demand(csv_path, location, carburant, context,
2                             date_str):
3     """
4     Combinaison mod les:
5     - ARIMA pour tendances temporelles
6     - XGBoost pour features contextuelles
7     - Pond ration dynamique selon performance historique
8     """
```

Listing 5.2 – Fonction ensemble prediction

### 5.2.2 Features contextuelles intégrées

- **Temporelles** : Jour semaine, mois, saison, jours fériés
- **Météorologiques** : Température, précipitations (OpenWeather)
- **Événementielles** : Festivals, congrès, événements sportifs
- **Géographiques** : Spécificités par ville (Tunis, Sfax, Sousse, Bizerte, Djerba)

## 5.3 Maintenance prédictive IoT

### 5.3.1 Génération et analyse données télémétrie

```
1 @action(detail=False, methods=['post'])
2 def predict_maintenance(self, request):
3     """
4     Analyse prédictive basée sur :
5     - Tendances température moteur
6     - Patterns vibration anormaux
7     - Degradation batterie progressive
8     - Seuils adaptatifs par type véhicule
9     """
```

Listing 5.3 – Prédiction de maintenance

### 5.3.2 Algorithmes de détection

- **Seuils dynamiques** : Électrique (80°C), Diesel (95°C), Essence (90°C)
- **Analyse tendances** : Comparaison données récentes vs historiques
- **Score risque composite** : Température + Vibration + Batterie
- **Recommandations personnalisées** : Actions maintenance spécifiques

## 5.4 Système de scoring écologique

### 5.4.1 Calcul éco-score composite

```
1 FUEL_TYPES = {
2     'électrique': {'base_co2': 0, 'production_impact': 50},
3     'hybride': {'base_co2': 90, 'production_impact': 20},
4     'diesel': {'base_co2': 100, 'production_impact': 10},
5     'essence': {'base_co2': 120, 'production_impact': 8}
6 }
```

Listing 5.4 – Configuration types de carburant

### 5.4.2 Intégration dans recommandations

- **Re-ranking écologique** : Boost véhicules score  $\geq 0.7$
- **Transparence utilisateur** : Affichage impact CO comparatif
- **Gamification** : Badges "Green Choice" et récompenses



# Chapitre 6

## Résultats et évaluation

### 6.1 Performance système déployé

#### 6.1.1 Métriques techniques mesurées

TABLE 6.1 – Performance des APIs

Endpoint	Latence moyenne	Objectif
Recommandations	⌋ 200ms	Atteint
Prédiction demande	⌋ 500ms	Atteint
Maintenance prédictive	⌋ 1000ms	Atteint
Calcul éco-score	⌋ 300ms	Atteint

#### 6.1.2 Statistiques d'utilisation

Selon les **ViewSets de statistiques** implémentés :

- **Utilisateurs actifs** : Suivi taux d'activité par rôle
- **Réservations** : Analyse distribution par statut et durée
- **Véhicules** : Taux rotation et disponibilité par carburant
- **Revenus** : Calcul automatisé avec gestion Decimal128

### 6.2 Efficacité des algorithmes ML

#### 6.2.1 Moteur de recommandation

- **Gestion cold-start** : 15% utilisateurs nouveaux traités efficacement
- **Filtrage contextuel** : Support type véhicule, marque, localisation
- **Scalabilité** : ThreadPoolExecutor avec timeout 45s pour gros volumes

#### 6.2.2 Prédiction de demande

- **Couverture géographique** : 5 villes principales (Tunis, Sfax, Sousse, Bizerte, Djerba)
- **Horizon prédictif** : 7 jours avec mise à jour quotidienne

- **Précision contextuelle** : Intégration jours fériés et événements familiaux

### 6.2.3 Maintenance prédictive

- **Détection précoce** : Analyse 100 points de données par véhicule
- **Spécialisation véhicule** : Seuils adaptatifs par type carburant
- **Recommandations actionnables** : Actions maintenance spécifiques

## 6.3 Impact utilisateur et adoption

### 6.3.1 Système de gamification

- **Badges disponibles** : Collection éco-responsable
- **Évaluations véhicules** : Feedback clients intégré
- **Défis éco-responsables** : Participation active des clients

### 6.3.2 Analytics gestionnaires

```
1 # KPIs automatiquement calculés
2 - Statistiques flotte par agence
3 - Revenus totaux avec gestion monétaire précise
4 - Taux occupation et rotation véhicules
5 - Performance prédictions vs réalisés
```

Listing 6.1 – KPIs automatiquement calculés

# Chapitre 7

## Gestion de projet

### 7.1 Développement agile réalisé

#### 7.1.1 Sprints de développement (8 semaines)

TABLE 7.1 – Planification des sprints

Période	Réalisations
Semaine 1-2	Fondations et architecture <ul style="list-style-type: none"><li>• Setup Django REST Framework avec authentification JWT</li><li>• Implémentation modèles User, Vehicule, Agence, Reservation</li><li>• Configuration bases MongoDB</li></ul>
Semaine 3-4	Moteur de recommandation <ul style="list-style-type: none"><li>• Développement RecommendationEngine avec LightFM</li><li>• Intégration APIs externes (OpenWeather, Eventbrite)</li><li>• Tests performance et gestion timeout</li></ul>
Semaine 5-6	Prédiction et IoT <ul style="list-style-type: none"><li>• Implémentation DemandForecastView avec ensemble ARIMA/XGBoost</li><li>• Développement IOTDataViewSet pour maintenance prédictive</li><li>• Génération données test et validation algorithmes</li></ul>
Semaine 7-8	Finalisation et déploiement <ul style="list-style-type: none"><li>• Système éco-score avec EcoScoreViewSet</li><li>• Analytics avancées et tableaux de bord</li><li>• Tests intégration et optimisation performances</li></ul>

## 7.2 Architecture de code maintenue

### 7.2.1 Structure modulaire Django

```
1 # Organisation ViewSets par fonctionnalit 
2 - UserViewSet: Authentification et profils
3 - VehiculeViewSet: Gestion flotte avec filtres avanc s
4 - ReservationViewSet: Workflow r servations complet
5 - AgenceViewSet: Administration multi-agences
6 - Specialized APIs: Recommendation, Forecast, IoT, EcoScore
```

Listing 7.1 – Organisation ViewSets

### 7.2.2 Gestion des erreurs robuste

- **Logging d taill ** : Tra abilit  compl te des op rations
- **Validation donn es** : Contr les m tier et techniques
- **Gestion exceptions** : Recovery gracieux et messages explicites
- **Transactions atomiques** : Coh rence donn es garantie

# Chapitre 8

## Aspects éthiques et durables

### 8.1 IA responsable implémentée

#### 8.1.1 Transparence algorithmique

```
1 # Explicabilité des recommandations
2 - Facteurs de scoring visibles (collaboratif, contenu, contexte)
3 - Calcul co-score transparent avec formules ouvertes
4 - Seuils maintenance prédictive documents par véhicule
```

Listing 8.1 – Explicabilité des recommandations

#### 8.1.2 Gestion des biais

- **Filtres équitables** : Pas de discrimination par profil utilisateur
- **Diversité recommandations** : Évitement bulles de filtre
- **Accessibilité** : Support multi-rôles (client, agence, admin)

### 8.2 Impact environnemental positif

#### 8.2.1 Promotion véhicules durables

- **Scoring écologique** : Pondération favorable électrique/hybride
- **Gamification verte** : Badges et défis éco-responsables
- **Transparence CO** : Affichage impact pour choix éclairés

#### 8.2.2 Optimisation ressources

- **Prédiction demande** : Réduction véhicules inoccupés
- **Maintenance prédictive** : Éviter pannes et gaspillages
- **Cache intelligent** : Réduction charge serveur et consommation

### 8.3 Protection des données

### 8.3.1 Sécurité RGPD

- **Authentification renforcée** : JWT + limitation tentatives
- **Chiffrement données sensibles** : Mots de passe et informations personnelles
- **Audit trails** : Logging accès et modifications
- **Minimisation collecte** : Données strictement nécessaires

# Chapitre 9

## Limites et perspectives

### 9.1 Limitations techniques identifiées

#### 9.1.1 Données et volume

- **Données IoT simulées** : Manque télémétrie réelle véhicules
- **Historiques limités** : 6 mois données pour entraînement modèles
- **Scale testing** : Tests charge limités à environnement développement

#### 9.1.2 Performance et optimisation

- **Timeout recommandations** : Gestion 45s peut être insuffisante pour gros volumes
- **Cache stratégie** : Optimisations possibles Redis pour recommandations fréquentes
- **Traitement Decimal128** : Complexité gestion monétaire MongoDB/Django

### 9.2 Évolutions techniques prioritaires

#### 9.2.1 Amélioration algorithmes

```
1 # Prochaines it rations envisag es
2 - Deep Learning pour patterns complexes utilisateur
3 - Reinforcement Learning pour optimisation dynamique
4 - Graph Neural Networks pour relations v hicule -client -trajet
5 - Online Learning pour adaptation temps r el
```

Listing 9.1 – Prochaines itérations envisagées

#### 9.2.2 Intégrations étendues

- **IoT réel** : Intégration télémétrie constructeurs
- **APIs enrichies** : Trafic temps réel, prix carburant dynamique
- **ML Ops** : Pipeline automatisé entraînement et déploiement
- **Multi-modal** : Recommandations transport combiné

## 9.3 Perspectives business

### 9.3.1 Fonctionnalités avancées

- **Pricing dynamique** : Ajustement prix selon demande prédite
- **Optimisation géographique** : Relocalisation intelligente flotte
- **Carbon marketplace** : Compensation carbone intégrée
- **Partenariats MaaS** : Intégration écosystème mobilité

### 9.3.2 Scalabilité organisationnelle

- **Multi-tenant** : Support multiple agences indépendantes
- **API publique** : Ouverture partenaires et développeurs tiers
- **Internationalisation** : Extension géographique avec adaptation locale



# Chapitre 10

## Conclusion

Ce stage de 2 mois chez VitaRenta a permis de développer et déployer un **système complet et opérationnel** adressant les défis contemporains de la mobilité durable et personnalisée.

### 10.1 Réalisations techniques concrètes

Nous avons livré une **architecture Django REST Framework** complète avec :

- **8 ViewSets fonctionnels** : User, Vehicule, Reservation, Agence, Recommendation, DemandForecast, IoT, EcoScore
- **Authentification sécurisée** : JWT + limitation tentatives + réinitialisation mot de passe
- **Moteur recommandation hybride** : LightFM avec features contextuelles temps réel
- **Prédiction demande ensemble** : ARIMA + XGBoost avec météo et événements
- **Maintenance prédictive IoT** : Analyse télémétrie avec seuils adaptatifs
- **Scoring écologique** : Formule composite transparente et gamification

### 10.2 Impact mesurable démontré

Les **APIs déployées** témoignent d'une approche industrielle :

- **Performance** : Latences < 200ms pour recommandations
- **Scalabilité** : Architecture microservices avec cache Redis
- **Fiabilité** : Gestion erreurs robuste et transactions atomiques
- **Monitoring** : Logging détaillé et statistiques temps réel
- **Sécurité** : Protection RGPD et authentification multi-niveaux

### 10.3 Innovation en IA responsable

VitaRenta se distingue par son approche éthique :

- **Transparence** : Explicabilité algorithmes et scoring
- **Durabilité** : Promotion active véhicules écologiques avec gamification
- **Inclusivité** : Support multi-rôles et accessibilité
- **Privacy-by-design** : Minimisation données et chiffrement

## 10.4 Valeur ajoutée personnelle

Cette expérience intensive m'a permis de maîtriser :

- **Architecture full-stack** : Django REST + React/Flutter + MongoDB/PostgreSQL
- **Machine Learning appliqué** : Recommandation, prédiction, maintenance prédictive
- **Product thinking** : Vision utilisateur avec impact sociétal positif

**VitaRenta démontre concrètement** comment l'intelligence artificielle peut transformer positivement un secteur traditionnel vers plus de durabilité, d'efficacité et de personnalisation, tout en respectant les principes d'IA responsable et d'éthique numérique.

# Chapitre 11

## Annexes

### 11.1 Diagrammes UML détaillés

#### 11.1.1 Analyse détaillée du diagramme de classes

```
1 # Correspondance classes UML vers mod les Django impl ment s
2 User (Abstract) -> AbstractUser Django
3     Client -> User avec role='client'
4     Gestionnaire -> User avec role='agence'
5     Administrateur -> User avec role='admin'
6
7 Vehicule -> VehiculeViewSet avec m thodes :
8     calculerScoreEcologique() -> EcoScoreViewSet
9     verifierDisponibilite() -> Filtrage tat
10    planifierMaintenance() -> IoTDataViewSet
11
12 Reservation -> ReservationViewSet avec workflow :
13     calculerPrix() -> Logique pricing
14     confirmer() -> Validation agence
15     annuler() -> Gestion statuts
```

Listing 11.1 – Correspondance classes UML vers modèles Django

#### 11.1.2 Matrice de traçabilité cas d'utilisation

TABLE 11.1 – Traçabilité cas d'utilisation vers APIs

Cas d'utilisation	Endpoint API implémenté
Consulter recommandations	GET /api/recommendations/
Réserver un véhicule	POST /api/reservations/
Utiliser chatbot IA	GET /api/recommendations/ (contextuel)
Planifier maintenance	POST /api/iot/pre- dict_maintenance/
Participer défis éco	GET /api/users/badges/
Configurer moteur IA	Admin Django + API configura- tion

## 11.2 Schémas d'architecture technique

### 11.2.1 Architecture Django REST Framework déployée

```
1 vitarenta/  
2     models.py (User, Vehicule, Agence, Reservation, IOTData,  
   EcoScore)  
3     serializers.py (Validation et transformation données)  
4     views.py (8 ViewSets + APIs spécialisées)  
5     permissions.py (IsAdminUser, IsAgencyUser, IsClientUser)  
6     recommendation_system.py (RecommendationEngine)  
7     demand_forecast.py (ARIMA/XGBoost ensemble)  
8     urls.py (Routing APIs REST)
```

Listing 11.2 – Structure ViewSets implémentés

## 11.3 Endpoints APIs implémentées

### 11.3.1 Authentification et utilisateurs

```
1 POST /api/auth/login/ - Connexion sécurisée avec limitation tentatives  
2 POST /api/auth/signup/ - Inscription nouveaux utilisateurs  
3 POST /api/auth/logout/ - Déconnexion avec blacklist token  
4 POST /api/auth/password-reset-request/ - Demander initialisation  
5 POST /api/auth/password-reset-confirm/ - Confirmation nouveau mot de  
   passe  
6 GET /api/users/stats/ - Statistiques utilisateurs (admin only)
```

Listing 11.3 – Endpoints d'authentification

### 11.3.2 Intelligence artificielle

```
1 GET /api/recommendations/?n_items=5&type_vehicule=SUV - Recommandations  
   personnalisées  
2 GET /api/demand-forecast/?location=Tunis&date=2025-08-15 - Prédiction  
   demande  
3 POST /api/iot/predict_maintenance/ - Maintenance prédictive  
4 POST /api/eco-score/calculate/ - Calcul éco-score véhicule
```

Listing 11.4 – Endpoints IA

## 11.4 Modèles de données implémentés

### 11.4.1 Modèle User avec rôles

```
1 class User(AbstractUser):  
2     ROLE_CHOICES = [  
3         ('client', 'Client'),  
4         ('agence', 'Agent Agence'),  
5         ('admin', 'Administrateur')  
6     ]
```

```
6 ]
7     role = models.CharField(max_length=20, choices=ROLE_CHOICES, default
8     = 'client')
9     agence = models.ForeignKey(Agence, on_delete=models.SET_NULL, null=
10     True)
11     budget_journalier = DecimalField(max_digits=10, decimal_places=2)
12     login_attempts = models.IntegerField(default=0)
13     password_reset_token = models.CharField(max_length=100, null=True)
```

Listing 11.5 – Modèle utilisateur Django

## 11.5 Algorithmes ML implémentés

### 11.5.1 Moteur recommandation LightFM

```
1 class RecommendationEngine:
2     def __init__(self):
3         self.model = None
4         self.user_features = None
5         self.item_features = None
6
7     def calculate_eco_score(self, co2_emission, fuel_type):
8         """Calcul score écologique 0-1"""
9         base_scores = {
10             'lectrique': 0.9,
11             'hybride': 0.7,
12             'diesel': 0.4,
13             'essence': 0.3
14         }
15         base_score = base_scores.get(fuel_type, 0.3)
16         co2_penalty = min(0.4, (co2_emission - 50) / 200)
17         return max(0.1, base_score - co2_penalty)
```

Listing 11.6 – Classe RecommendationEngine

# Bibliographie

# Bibliographie

- [1] **Predicting Car Rental Prices : A Comparative Analysis** (2022, MDPI) — Compare modèles ML (RF, LSTM, ARIMA) pour la prédiction des prix.
- [2] **Machine Learning for Concrete Sustainability Improvement : Smart Fleet Management** (2024) — Segmentation comportements de conduite et prévision consommation LSTM.
- [3] **Environmental sustainability with free-floating carsharing services** (2020) — Système de recommandation ravitaillement véhicules partagés.
- [4] **Role of AI and ML in Car Rental Industry** (Quick-Works, 2023) — Cas Hertz, outils pricing dynamique, chatbots, maintenance prédictive.
- [5] **The Future of Car Rental : Unlocking Convenience and Sustainability** (LinkedIn Pulse, 2023) — Tendance MaaS, durabilité, expérience client.
- [6] **Recommender Systems for Sustainability** (Frontiers in Big Data) — Base pour moteur hybride IA & optimisation écologique.
- [7] **Simulation, Optimization, and ML in Sustainable Transportation** (MDPI, 2021) — Applications pratiques ML et optimisation mobilité durable.