Flesk Car Sell

AI-Powered Car Sales Platform

Progressive Web Application Documentation

Development Team:

Hamdi MOKNI
Amen Allah JOUILA

https://github.com/amenallah0/myCar
https://github.com/amenallah0/mycar$_{a}pi$
https://github.com/amenallah0/TuniPlate
amenallah.jouila@esprim.tn
hamdi.mokni@esprim.tn

August 28, 2025

Résumé Exécutif

MyCar représente une application web progressive (PWA) de pointe conçue pour révolutionner l'expérience d'achat et de vente de voitures. Cette plateforme complète offre :

- Un catalogue complet de véhicules avec images et descriptions détaillées
- Une interface de recherche et de filtrage avancée
- Une authentification et une gestion sécurisées des utilisateurs
- Un blog informatif avec actualités et conseils
- Un support client complet avec FAQ

Développé avec React et des technologies web modernes, le système garantit :

- Une expérience utilisateur intuitive et responsive
- Une performance optimale
- Une scalabilité importante

Cette plateforme offre une solution complète pour :

- Les acheteurs de véhicules
- Les vendeurs particuliers et professionnels
- Les prestataires de services automobiles
- Les experts du secteur
- Les passionnés d'automobile

Contents

List of Figures

Introduction

1.1 Contexte du projet

Le projet MyCar est une plateforme e-commerce spécialisée dans la vente et l'achat de véhicules. Cette solution innovante vise à digitaliser le marché automobile en offrant une expérience utilisateur optimale tant pour les vendeurs que pour les acheteurs.

1.2 Organisation Hôte

Flesk est une entreprise de développement logiciel qui offre des services complets et des solutions sur mesure aux startups et aux entreprises dans divers secteurs, notamment la vente au détail, la banque, l'assurance, l'hôtellerie, l'éducation, la mode et la joaillerie.

Figure 1.1: Company Logo

1.3 Objectifs

- Créer une plateforme de mise en relation entre vendeurs et acheteurs de véhicules.
- Automatiser le processus de vente automobile.
- Optimiser l'expérience utilisateur sur les interfaces desktop.

Besoins Fonctionnels

2.1 Introduction

Ce chapitre présente les besoins fonctionnels de l'application, définissant les fonctionnalités principales que le système doit offrir à ses utilisateurs. Les besoins fonctionnels sont centrés sur les actions que les utilisateurs peuvent réaliser, les interactions avec le système, ainsi que les résultats attendus de ces actions. Ces besoins sont essentiels pour assurer que l'application réponde correctement aux exigences des utilisateurs et atteigne les objectifs du projet.

2.2 Authentification et Gestion des Utilisateurs

Le système doit permettre aux utilisateurs de créer un compte, de se connecter et de gérer leur profil personnel. Les fonctionnalités suivantes sont requises :

- **Inscription des utilisateurs** : Les utilisateurs doivent pouvoir créer un compte en fournissant leurs informations personnelles (nom, email, mot de passe).
- **Connexion sécurisée** : Les utilisateurs doivent pouvoir se connecter en utilisant leur email et leur mot de passe. Le système doit sécuriser cette connexion avec des mécanismes d'authentification tels que JWT.
- **Gestion des mots de passe** : Les utilisateurs doivent pouvoir réinitialiser leur mot de passe en cas d'oubli.
- **Profil utilisateur** : Chaque utilisateur doit pouvoir consulter et modifier ses informations personnelles, telles que le nom, l'email, et la photo de profil.
- **Rôles et permissions** : Le système doit permettre la gestion des rôles des utilisateurs (administrateur, utilisateur standard) pour restreindre l'accès à certaines fonctionnalités.

2.3 Gestion des Véhicules

Les utilisateurs doivent pouvoir ajouter, modifier, consulter et supprimer des véhicules dans leur compte. Les principales fonctionnalités à implémenter sont :

- **Ajout de véhicule** : L'utilisateur doit pouvoir ajouter un véhicule en renseignant des informations telles que la marque, le modèle, l'année de fabrication, le numéro d'immatriculation, et télécharger des photos du véhicule.
- **Consultation des véhicules** : L'utilisateur doit pouvoir consulter la liste de ses véhicules avec un aperçu des informations principales (nom, photo, modèle, etc.).

- **Modification des véhicules** : L'utilisateur doit pouvoir modifier les informations d'un véhicule existant, notamment pour corriger des erreurs ou mettre à jour les détails.
- **Suppression de véhicule** : L'utilisateur doit pouvoir supprimer un véhicule de son compte.

2.4 Traitement d'Images

Le système doit intégrer un microservice de traitement d'images automatisé qui intervient dès qu'un utilisateur ajoute un nouveau véhicule. Lors de l'ajout, l'utilisateur doit télécharger une image de son véhicule, et si une plaque d'immatriculation est détectée, celle-ci sera automatiquement supprimée et remplacée par une version floutée pour garantir la confidentialité. Les fonctionnalités associées sont les suivantes :

- **Téléchargement d'images** : L'utilisateur doit pouvoir télécharger l'image de son véhicule lors de l'ajout de ce dernier via une interface simple et rapide.
- **Détection et suppression des plaques d'immatriculation** : Le système doit détecter automatiquement toute plaque d'immatriculation présente sur l'image, la supprimer, puis la remplacer par une version floutée afin de préserver la vie privée de l'utilisateur.
- **Floutage des plaques d'immatriculation** : Si une plaque d'immatriculation est détectée, elle sera floutée pour garantir la confidentialité de l'utilisateur.
- **Visualisation des images traitées** : L'utilisateur pourra visualiser l'image après traitement, avec la plaque d'immatriculation remplacée par un flou.
- **Archivage des images** : Le système conservera les images traitées pour une consultation future, en veillant à ce que les informations sensibles soient protégées.

2.5 Système de Notifications

Le système doit notifier les utilisateurs de diverses actions et événements pertinents pour leur compte ou leurs véhicules. Cela inclut :

- **Notifications par email** : L'utilisateur doit recevoir des notifications par email pour des événements importants, comme la confirmation de l'inscription, la réinitialisation du mot de passe, ou la mise à jour des informations.
- **Notifications en temps réel** : L'utilisateur doit être informé en temps réel via l'interface de l'application pour des événements comme la réussite du téléchargement d'une image ou la fin du traitement d'une image.

2.6 Historique des Activités

Le système doit permettre à l'utilisateur de consulter l'historique de ses actions et événements importants liés à ses véhicules. Les principales fonctionnalités sont :

- **Suivi des traitements d'images** : L'utilisateur doit pouvoir consulter l'historique des images téléchargées et des traitements effectués sur ces images.
- **Historique des actions** : Le système doit garder une trace des actions effectuées par l'utilisateur, telles que l'ajout, la modification, et la suppression de véhicules.

2.7 Administration du Système

Les administrateurs doivent avoir des droits d'accès étendus pour gérer les utilisateurs, les véhicules, et le système dans son ensemble. Les fonctionnalités administratives comprennent :

- **Gestion des utilisateurs** : L'administrateur doit pouvoir consulter la liste des utilisateurs, ajouter, modifier, ou supprimer des utilisateurs, et gérer leurs rôles.
- **Gestion des véhicules** : L'administrateur doit pouvoir consulter et gérer les véhicules de tous les utilisateurs.
- **Surveillance du système** : L'administrateur doit avoir accès aux logs du système et à des outils de surveillance pour assurer la bonne marche du service.

2.8 Conclusion

Les besoins fonctionnels présentés dans ce chapitre décrivent les principales fonctionnalités que l'application doit offrir pour garantir une expérience utilisateur fluide et sécurisée. Ces exigences sont essentielles pour définir les bases du développement et orienter la mise en œuvre des différentes fonctionnalités du système. Le respect de ces besoins garantira la satisfaction des utilisateurs et la réussite du projet.

Technologies Utilisées

3.1 Frontend

Le frontend de l'application a été développé avec React, offrant une interface utilisateur moderne et réactive. Les technologies utilisées sont détaillées ci-dessous :

3.1.1 Technologies Principales

- **React 18.0** : Framework principal pour le développement de l'interface utilisateur.
- **JavaScript** : Langage dynamique permettant une grande flexibilité et rapidité de développement..
- **React Router v6** : Gestion de la navigation et du routage côté client.

3.1.2 Bibliothèques UI et Utilitaires

- **Material-UI v5** : Framework UI fournissant des composants réutilisables et cohérents pour une meilleure expérience utilisateur.
- **Styled-components** : Outil de stylisation basé sur le principe CSS-in-JS pour un code plus modulaire et maintenable.
- **Axios** : Client HTTP utilisé pour les appels et les requêtes API.
- **React-Query** : Gestion efficace du cache et des requêtes côté client pour améliorer les performances.

3.2 Backend

Le backend est construit sur Spring Boot, offrant une architecture robuste et scalable. Les principales technologies et caractéristiques sont détaillées ci-dessous :

3.2.1 Core Technologies

- **Spring Boot 3.0** : Framework principal pour le développement backend.
- **Spring MVC** : Gère les interactions entre l'utilisateur et l'application en organisant les requêtes HTTP, les réponses, et la logique d'affichage des pages.
- **Spring REST** : Facilite la création d'API RESTful pour permettre aux applications ou systèmes de communiquer entre eux via des formats standards comme JSON ou XML.

3.2.2 Caractéristiques Principales

- **Framework léger et flexible** : Spring Boot 3.0 permet de créer rapidement des applications performantes et facilement extensibles.
- **Gestion des requêtes utilisateur** : Spring MVC organise efficacement les requêtes HTTP et les réponses, tout en offrant une logique claire pour l'affichage des données.
- **Création et gestion des API RESTful** : Spring REST facilite la communication entre les systèmes avec des formats standards comme JSON, assurant une interopérabilité fluide.
- **Configuration simplifiée** : Spring Boot réduit la complexité des configurations initiales, permettant un démarrage rapide des projets.

3.2.3 Sécurité

- **JWT** : Utilisé pour l'authentification sécurisée.
- **Configuration CORS** : Gère les politiques de partage des ressources entre origines.

3.3 Traitement d'Images

Le microservice de traitement d'images utilise Flask et des bibliothèques d'intelligence artificielle pour effectuer diverses opérations avancées.

3.3.1 Technologies Core

- **Flask 2.0** : Framework Python léger pour la création d'API RESTful.
- **TensorFlow 2.x** : Framework puissant pour l'apprentissage automatique.
- **OpenCV** : Bibliothèque pour le traitement d'images.
- **NumPy** : Utilisée pour les calculs numériques complexes.

3.3.2 Fonctionnalités IA

- **Détection d'objets avec YOLO v5** : Pour identifier et localiser des objets dans les images.
- **Classification d'images avec ResNet** : Modèle avancé pour classifier les images.
- **Optimisation des images avec Pillow** : Pour redimensionner et améliorer la qualité des images.

3.3.3 API et Intégration

- **API RESTful Flask** : Fournit des services d'intégration.
- **Documentation Swagger** : Pour une documentation API conviviale.

3.4 Base de Données

L'architecture de la base de données est conçue pour assurer des performances optimales, une gestion efficace des données et une scalabilité adaptée aux besoins actuels et futurs du projet.

3.4.1 Technologies Principales

- **MySQL 8** : MySQL est utilisé comme la base de données relationnelle principale. Il offre une gestion robuste des données structurées grâce à des fonctionnalités avancées telles que les transactions ACID, les jointures complexes et un moteur de requêtes performant. La version 8 apporte des améliorations significatives en termes de sécurité, de performance et de support pour les structures JSON.
- **SQLite** : SQLite est intégré pour des scénarios spécifiques, notamment les tests locaux ou les applications légères où une base de données embarquée est plus adaptée. Grâce à son architecture sans serveur, SQLite est facile à configurer et consomme peu de ressources, ce qui le rend idéal pour le prototypage rapide et les environnements de développement.

3.4.2 Structure et Organisation

La base de données est conçue autour d'un modèle relationnel, avec des tables bien définies pour représenter les entités principales de l'application. Une attention particulière est portée sur la normalisation des données afin de réduire la redondance et d'améliorer leur cohérence.

3.4.3 Optimisation des Performances

Pour garantir des performances élevées :

- Des index sont utilisés sur les colonnes fréquemment interrogées.
- Les requêtes SQL sont optimisées pour minimiser les temps de réponse.
- Un système de sauvegardes régulières est mis en place pour éviter toute perte de données.

Fonctionnalités Clés

4.1 Gestion des Utilisateurs

La gestion des utilisateurs dans ce projet est bien structurée et sécurisée. Elle suit les bonnes pratiques de développement avec une séparation claire des responsabilités. Les principales fonctionnalités sont présentes et le système est extensible pour des améliorations futures.

Gestion des comptes :

- Création de compte utilisateur
- Mise à jour des informations
- Récupération des annonces publier par l'utilisateur

Sécurité :

- Cryptage des mots de passe avec BCrypt
- Validation des données

4.2 Gestion des Annonces de Voitures

La gestion des annonces de voitures constitue le cœur du système, permettant aux utilisateurs de créer, modifier et supprimer leurs annonces de véhicules. Chaque annonce contient des informations détaillées sur le véhicule (marque, modèle, année, prix, description) et son statut (disponible, vendu, réservé).
Le système implémente une recherche avancée avec filtres multiples (prix, année, marque, etc.) et une pagination efficace pour gérer de grands volumes d'annonces. Les annonces sont liées à leurs propriétaires via une relation `ManyToOne`, permettant un suivi précis et une gestion des droits d'accès appropriée.

Caractéristiques principales :

- Identifiant unique auto-généré
- Informations du véhicule (marque, modèle, année, prix)
- Relation avec l'utilisateur propriétaire
- Gestion des images

4.3 Gestion des images

Le système de gestion des images permet un traitement efficace des visuels associés aux annonces de voitures. Il prend en charge l'upload multiple d'images avec un traitement automatisé incluant le redimensionnement, la compression et la génération de thumbnails. Chaque image est validée pour la sécurité (type de fichier, taille) et stockée de manière optimisée. Le système maintient différentes versions des images (originale, thumbnail, moyenne résolution) pour optimiser le chargement selon le contexte d'utilisation.

- Upload multiple d'images
- Redimensionnement automatique
- Stockage sécurisé

4.4 AI Integration

Le projet intègre un système de détection de plaques d'immatriculation basé sur YOLO (You Only Look Once), un algorithme de détection d'objets en temps réel. Cette implémentation utilise OpenCV et le framework YOLO pour détecter et flouter automatiquement les plaques d'immatriculation dans les images de véhicules.
Key Features:

- Détection Automatique
- Identification précise des plaques d'immatriculation
- Support pour plusieurs formats d'images
- Traitement par lots possible
- Privacy Protection
- Floutage automatique des plaques détectées
- Utilisation du flou gaussien (99x99 pixels)
- Préservation de la qualité globale de l'image

4.5 Gestion des Favoris

La fonctionnalité de favoris permet aux utilisateurs de sauvegarder et suivre les annonces qui les intéressent. Cette feature facilite la navigation et la comparaison des véhicules.

Les utilisateurs peuvent ainsi :

- Ajouter/Retirer des annonces de leurs favoris
- Consulter leur liste de favoris
- Accéder rapidement aux annonces sauvegardées

Analyse Fonctionnelle des Cas d'Utilisation

Cette section présente les diagrammes UML utilisés pour concevoir et documenter l'architecture du système, ainsi que les principaux cas d'utilisation de l'application.

5.1 Diagramme de Classes

Le diagramme de classes ci-dessous illustre la structure statique du système, incluant les entités principales, leurs attributs, méthodes, et relations.

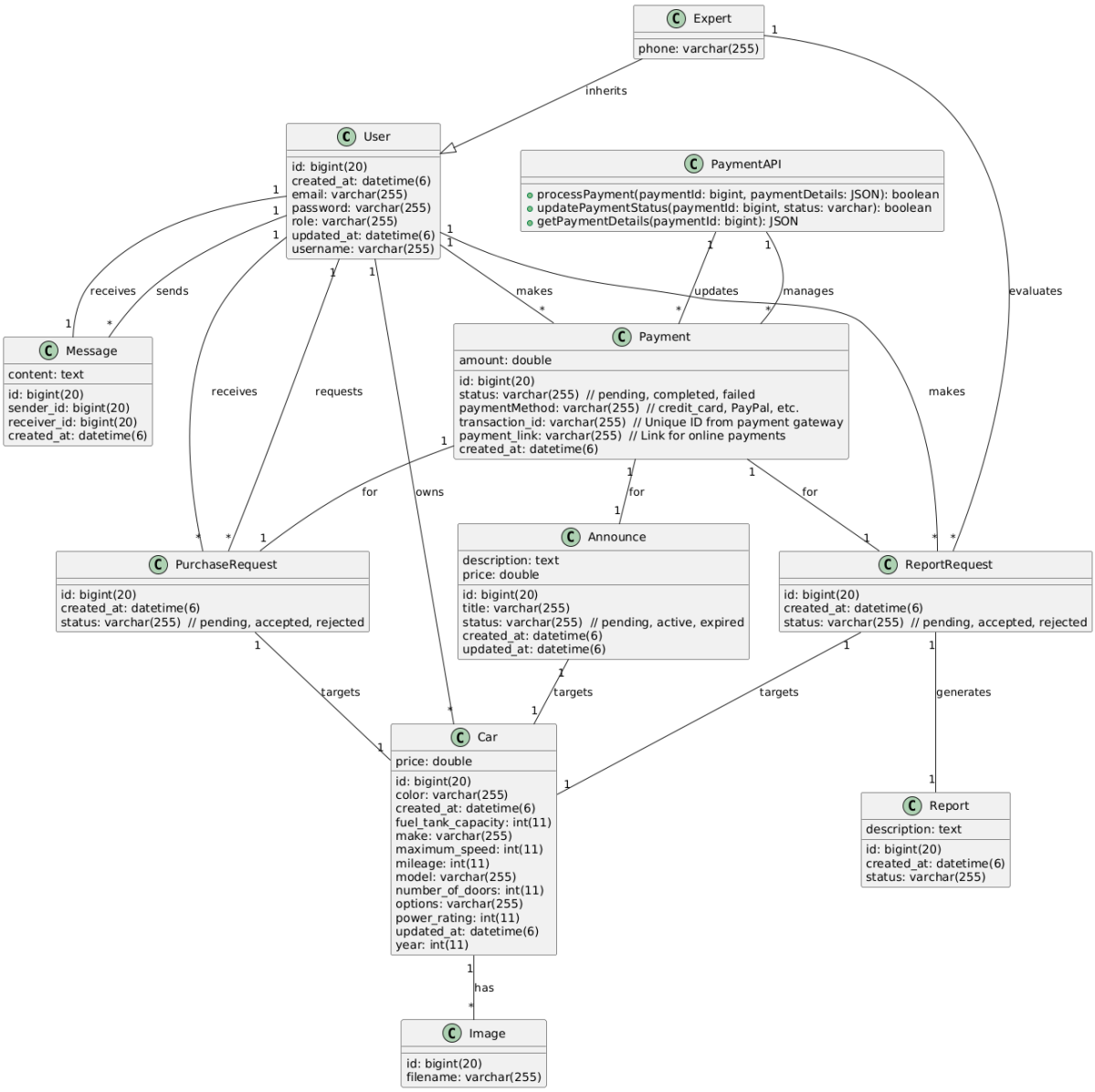

Figure 5.1: Diagramme de classes du système

5.2 Cas d'Utilisation

Cette section détaille les principaux cas d'utilisation du système, en mettant en avant les différents acteurs et leurs interactions via des diagrammes explicatifs. Chaque sous-section décrit un rôle spécifique et ses responsabilités au sein du système.

5.2.1 Cas d'Utilisation de l'Administrateur

L'administrateur joue un rôle central dans le système. Ses principales responsabilités incluent :

- Gestion des utilisateurs (ajout, suppression, modification des informations).
- Supervision et gestion des véhicules dans la base de données.
- Gestion des diagnostics et traitement des images pour garantir leur qualité et leur pertinence.

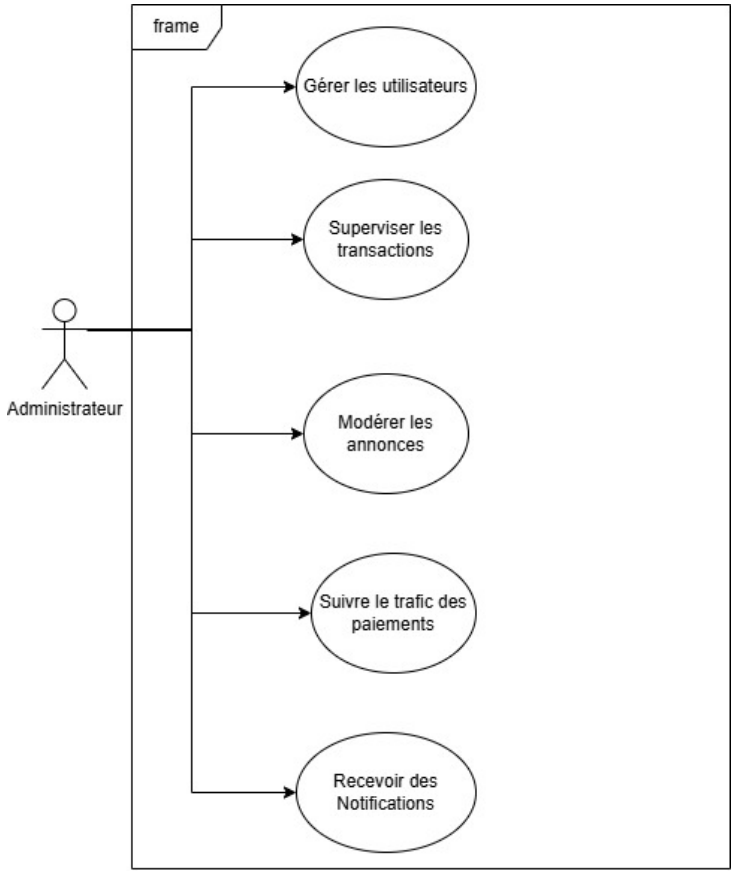

Figure 5.2: Cas d'utilisation de l'Administrateur

5.2.2 Cas d'Utilisation de l'Expert

L'expert est responsable de l'analyse et du diagnostic des véhicules. Ses activités principales comprennent :

- Consultation des véhicules listés.

- Réalisation d'analyses techniques sur les véhicules.
- Archivage et mise à disposition des rapports de diagnostic.

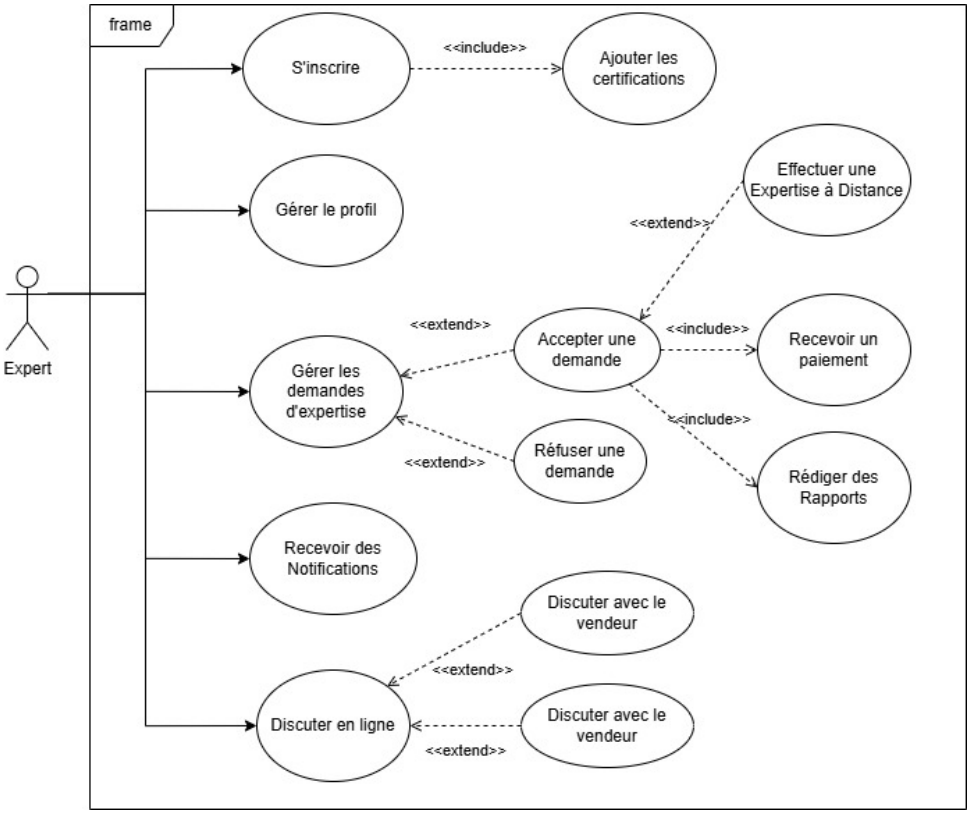

Figure 5.3: Cas d'utilisation de l'Expert

5.2.3 Cas d'Utilisation de l'Acheteur

L'acheteur interagit avec le système pour trouver et acheter des véhicules. Les actions disponibles pour ce rôle incluent :

- Recherche de véhicules à l'aide de filtres avancés (marque, prix, année, etc.).
- Consultation des détails des véhicules (description, images, rapports de diagnostic).
- Initiation d'un processus d'achat sécurisé.
- Accès aux images des véhicules traitées et floutées pour protéger la vie privée.

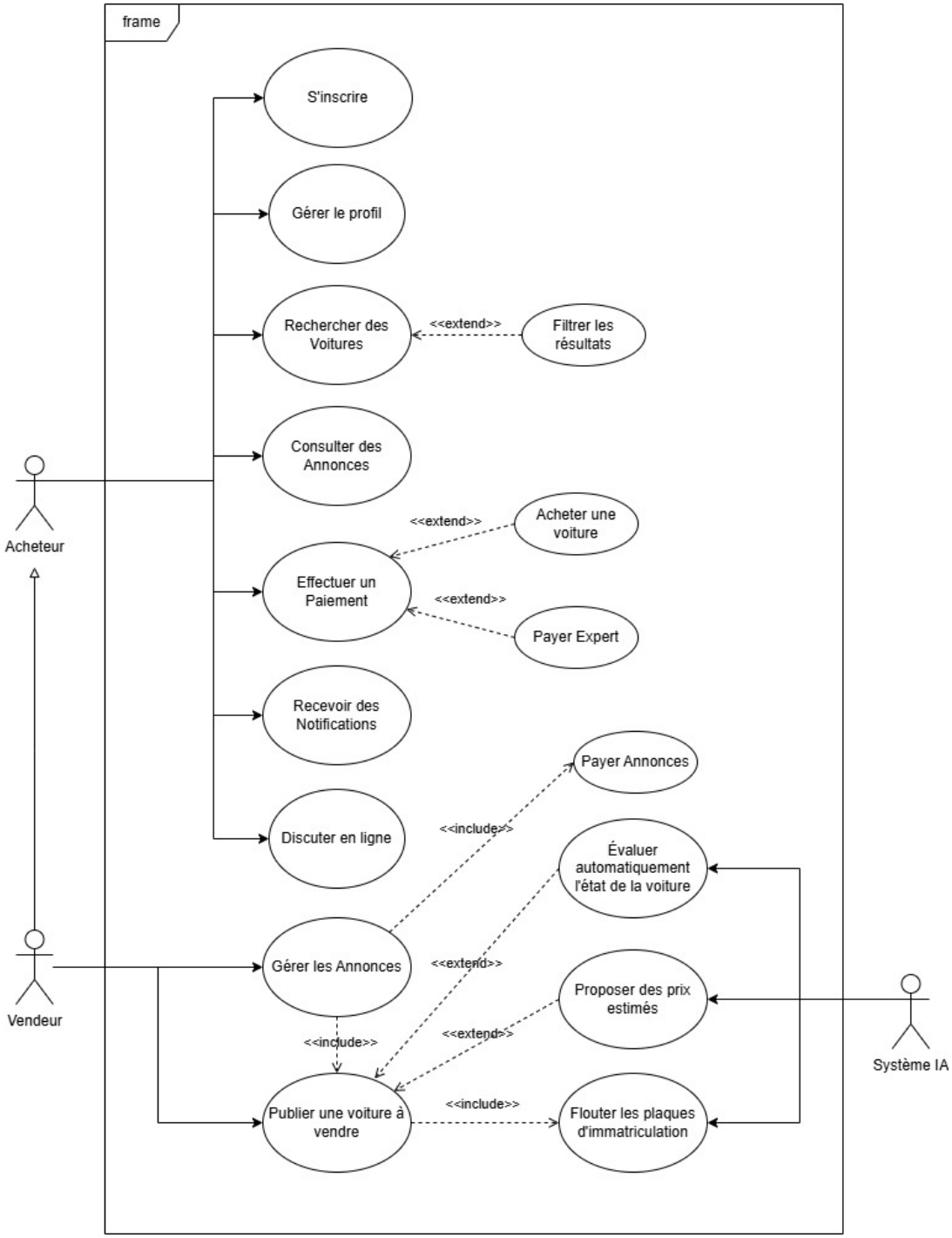

Figure 5.4: Cas d'utilisation de l'Acheteur

Interface Utilisateur

Cette section présente les principales interfaces utilisateur de l'application, illustrant les différentes fonctionnalités et le design global.

6.1 Accueil

La page d'accueil de l'application offre une vue claire et intuitive des principales fonctionnalités. Elle comprend :

- Une barre de navigation pour accéder rapidement aux différentes sections.
- Un tableau de bord résumant les informations clés.
- Des boutons d'action pour effectuer des tâches courantes.

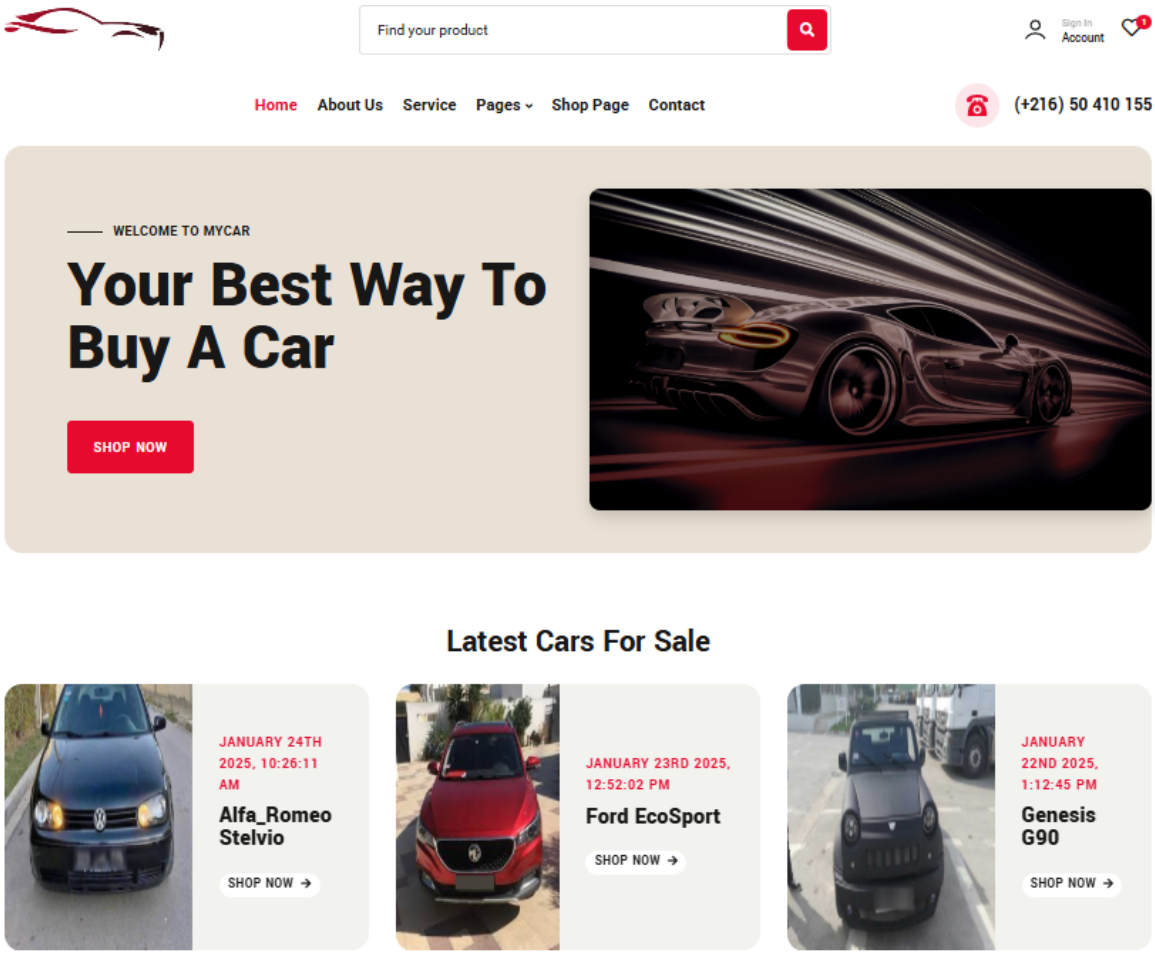

Figure 6.1: Page d'accueil de l'application.

6.2 Gestion du Profil

L'interface de gestion du profil offre aux utilisateurs la possibilité de gérer leurs informations personnelles de manière simple et intuitive. Elle comprend :

- Une section affichant les informations personnelles de l'utilisateur, telles que le nom, l'adresse e-mail, et le rôle.
- Un formulaire interactif permettant de mettre à jour les informations du profil.
- Une section dédiée à la gestion des annonces postées par l'utilisateur, incluant la possibilité de les consulter, les modifier ou les supprimer.

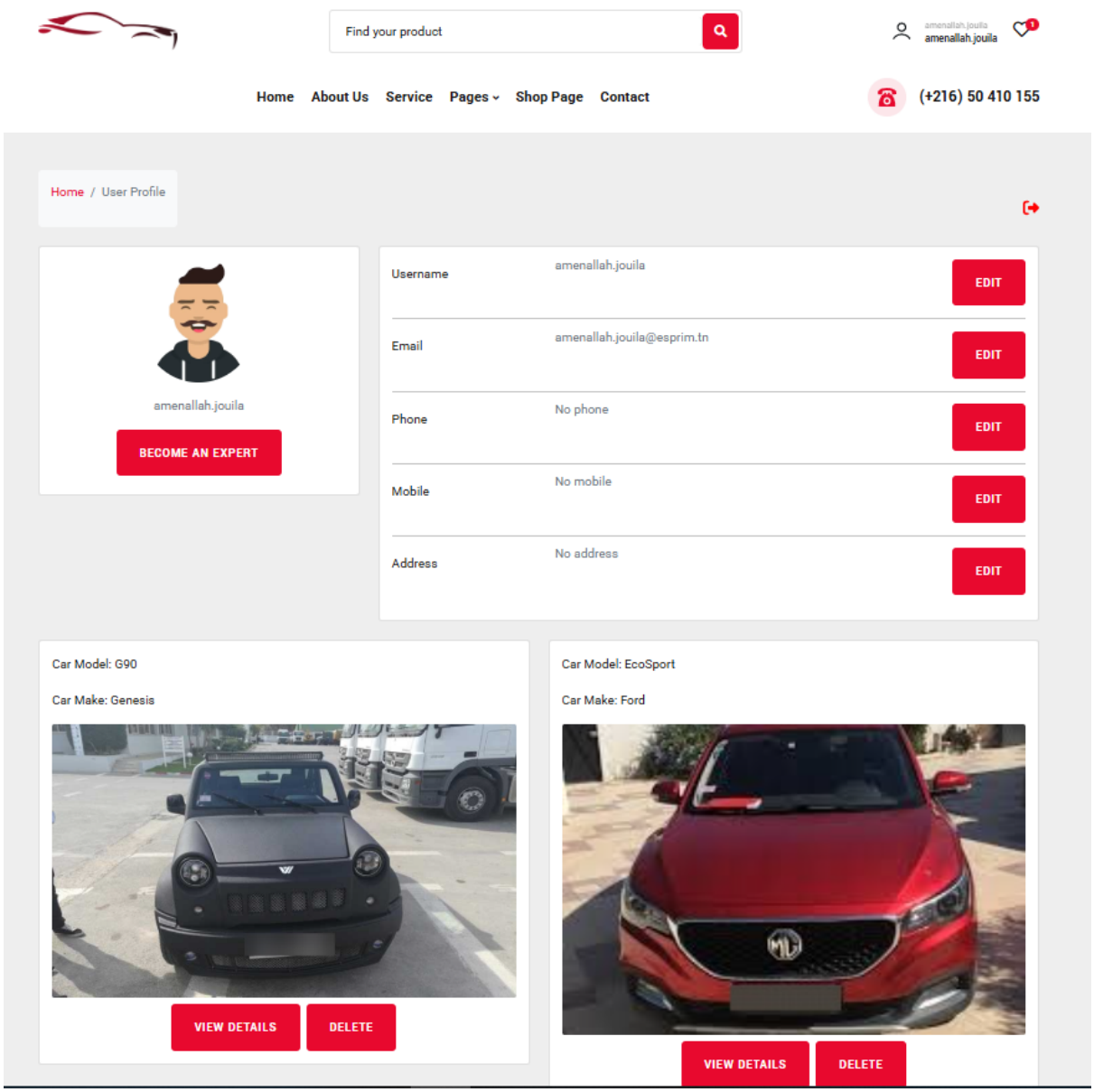

Figure 6.2: Interface de gestion du profil.

6.3 Gestion des Images

L'interface dédiée au traitement des images offre des fonctionnalités avancées pour assurer une manipulation efficace et sécurisée des visuels. Elle comprend :

- Un module d'upload permettant le téléchargement simultané de plusieurs images.
- Un système de traitement automatique des images incluant le floutage des plaques d'immatriculation pour la protection des données sensibles.
- Une fonctionnalité de redimensionnement automatique des images pour les adapter aux formats requis par la plateforme.

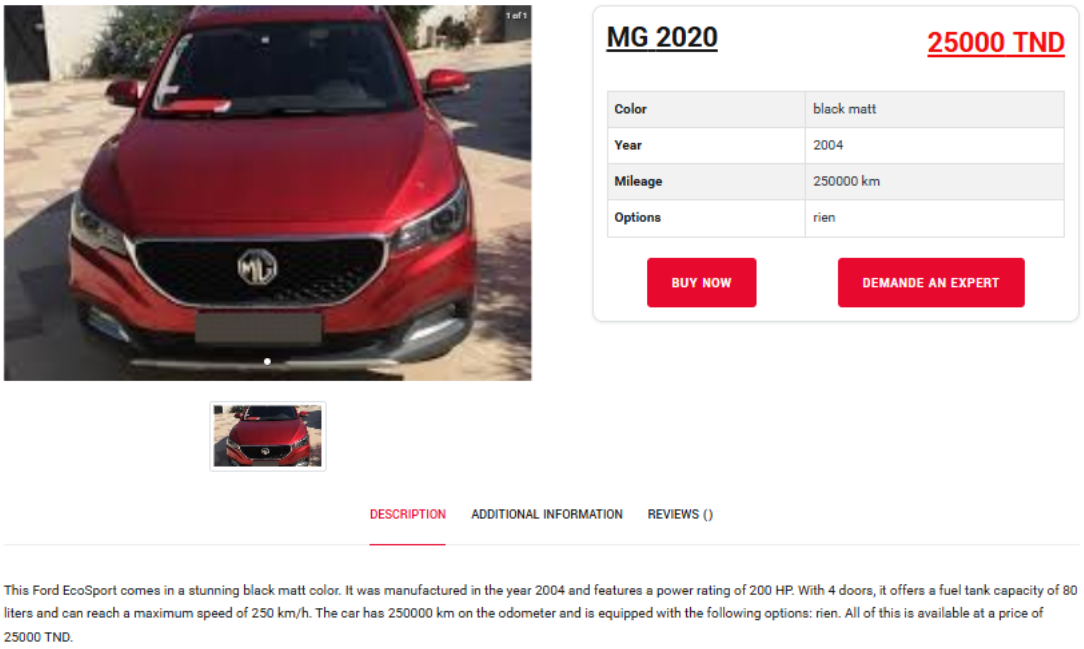

Figure 6.3: Interface de la page de detail.

6.4 Conclusion

Les interfaces utilisateur sont conçues pour offrir une expérience utilisateur fluide, intuitive et esthétique. Chaque écran est optimisé pour une accessibilité et une efficacité maximales.

Test et Validation

Cette section présente les tests effectués pour garantir la robustesse, la fiabilité et les performances du système.

7.1 Introduction

Les tests jouent un rôle crucial dans le cycle de développement logiciel. Ils permettent de vérifier la conformité aux spécifications, de détecter les erreurs et de s'assurer que le système fonctionne correctement dans différents scénarios.

7.2 Test de Concurrence

L'objectif de ce test est de valider le comportement du système dans des scénarios de concurrence, où plusieurs utilisateurs accèdent simultanément à une ressource partagée. Le test a été réalisé sur le service utilisateur pour simuler l'accès concurrent à une base de données contenant un utilisateur unique.

7.2.1 Code du Test

Voici le code utilisé pour le test de concurrence :

```
1  public class UserServiceConcurrencyTest {
2
3      @Mock
4      private UserRepository userRepository;
5
6      @InjectMocks
7      private UserService userService;
8
9      @BeforeEach
10     public void setUp() {
11         MockitoAnnotations.openMocks(this);
12     }
13
14     @Test
15     public void testConcurrentUserAccess() throws InterruptedException,
       ExecutionException {
16         // Simuler une base de donn es avec un utilisateur unique
17         User user = new User("username", "email@example.com", "password
       ");
18         when(userRepository.findById(1L)).thenReturn(Optional.of(user))
       ;
19
20         // D finir un pool de threads pour simuler des utilisateurs
       concurrents
21         int numberOfUsers = 100;
22         ExecutorService executor = Executors.newFixedThreadPool(10);
```

```
23          List<Future<User>> futures = new ArrayList<>();
24
25          // Simuler l'acc s simultan  de plusieurs utilisateurs
26          for (int i = 0; i < numberOfUsers; i++) {
27              futures.add(executor.submit(() -> userService.getUserById(1
        L)));
28          }
29
30          // V rifier que chaque thread a bien r cup r  l'utilisateur
31          for (Future<User> future : futures) {
32              User retrievedUser = future.get();
33              assertNotNull(retrievedUser);
34          }
35
36          // Arr ter le pool de threads
37          executor.shutdown();
38          executor.awaitTermination(1, TimeUnit.MINUTES);
39      }
40 }
```

Listing 7.1: Test de concurrence pour le service utilisateur

7.2.2 Description du Test

- **Scénario simulé :** Accès concurrent de 100 utilisateurs à un utilisateur unique dans la base de données.
- **Technologies utilisées :**
  - Mockito pour le mock des dépendances.
  - JUnit pour les assertions et la structure de test.
  - Executors pour gérer les threads concurrents.
- **Objectif :** Vérifier que le service utilisateur peut gérer plusieurs requêtes simultanées sans compromettre les performances ou la fiabilité.

7.2.3 Résultats du Test

Tous les threads ont réussi à récupérer l'utilisateur sans exception ni erreur, confirmant que le système est capable de gérer des scénarios d'accès concurrent.

7.3 Conclusion

Le test de concurrence a permis de valider la robustesse du système face à des charges importantes et des accès simultanés. Cela garantit que le système est prêt à être utilisé dans des environnements multi-utilisateurs.

Améliorations futures

Dans les versions futures de la plateforme, plusieurs améliorations sont envisagées pour enrichir l'expérience utilisateur et renforcer les capacités de l'intelligence artificielle. Parmi les principales améliorations à venir, on retrouve :

- **Amélioration de l'Analyse Prédictive :** Intégration de nouveaux modèles d'apprentissage automatique pour affiner les prévisions des prix du marché et des tendances en temps réel.
- **Expansion de la Vision par Ordinateur :** Développement de nouvelles fonctionnalités de reconnaissance d'images pour une analyse plus précise des véhicules, incluant la détection d'anomalies mécaniques à partir des photos et vidéos téléchargées.
- **Optimisation de l'Interface Utilisateur :** Une nouvelle version de l'interface utilisateur avec des options de personnalisation avancées, permettant aux utilisateurs de configurer leurs préférences d'affichage et de navigation.
- **Fonctionnalités Sociales Avancées :** Mise en place de forums de discussion interactifs, d'un système de recommandations basé sur les interactions des utilisateurs, et d'un réseau social permettant aux acheteurs et vendeurs d'échanger des avis et conseils.
- **Extension des Méthodes de Paiement :** Ajout de nouvelles solutions de paiement afin de faciliter les transactions pour les utilisateurs.

Ces améliorations permettront d'élargir les fonctionnalités de la plateforme tout en assurant une meilleure sécurité, accessibilité et satisfaction des utilisateurs. La plateforme restera ainsi à la pointe de l'innovation dans le domaine de l'achat et de la vente de véhicules.

Conclusion

La réalisation du projet MyCar a permis de mettre en place une plateforme innovante de commerce automobile en ligne, répondant aux besoins actuels du marché et intégrant les technologies modernes.

Réalisations Principales : Le projet a abouti à plusieurs réalisations majeures :

- Architecture technique robuste et évolutive
- Interface utilisateur intuitive et responsive
- Intégration réussie des fonctionnalités d'IA
- Plateforme conforme aux normes de sécurité

Résultats Obtenus : Les objectifs fixés initialement ont été atteints avec succès :

- Performance optimale de la plateforme
- Sécurisation des données utilisateurs
- Expérience utilisateur fluide et intuitive