

Université Paris Nanterre

L3 MIAE

# Cahier des charges

L3 MIAE

Projet Web

---

Soutenu le : 12/12/2025

Réalisé par : Baaboura Ritej, Thibaud Thomas Lamotte, Baibou Selma

## Système de Gestion de Garage

*Application Web Django pour la Gestion des Réparations  
Automobiles*

Année Universitaire 2024/2025

*Version 3.0 - Document actualisé après implémentation*



# Table des matières

## Liste des abréviations

<b>Introduction générale</b>	<b>1</b>
<b>1 Cadre Général du Projet</b>	<b>2</b>
1.1 Introduction . . . . .	2
1.2 Contexte général du projet . . . . .	2
1.3 Méthodologies de gestion de projet . . . . .	5
1.4 Conclusion . . . . .	6
<b>2 Sprint 0 : Analyse et Spécification des Besoins</b>	<b>7</b>
2.1 Introduction . . . . .	7
2.2 Acteurs du Système . . . . .	7
2.3 Besoins Fonctionnels . . . . .	9
2.4 Besoins Non Fonctionnels . . . . .	13
2.5 Diagramme de Cas d'Utilisation Global . . . . .	16
2.6 Écrans Principaux de l'Application . . . . .	17
2.7 Backlog Produit . . . . .	18
2.8 Rôles Scrum . . . . .	28
2.9 Planification des Sprints . . . . .	29
2.10 Architecture du Système et Environnement de Travail . . . . .	31
2.11 Règles de Gestion et Calculs . . . . .	38
2.12 Jeux de Données de Démonstration . . . . .	42
2.13 Conclusion . . . . .	44
<b>Conclusion Générale</b>	<b>45</b>

# Table des figures

2.1	Diagramme de cas d'utilisation global du système de gestion de garage . .	17
2.2	Architecture MVT de Django . . . . .	32
2.3	Architecture physique trois tiers . . . . .	33
2.4	Modèle de données conceptuel - Diagramme de classes UML . . . . .	35

# Liste des tableaux

1.1	Comparaison entre les méthodes Waterfall et Agile (Scrum) . . . . .	5
2.1	Identification des acteurs du système . . . . .	8
2.2	Règles de validation des champs de formulaire . . . . .	16
2.3	Backlog produit avec critères d'acceptation . . . . .	19
2.4	Répartition des rôles Scrum dans le projet . . . . .	28
2.5	Planification des sprints . . . . .	30
2.6	Configuration matérielle minimale . . . . .	36
2.7	Technologies et outils utilisés . . . . .	37
2.8	Spécifications de mise en page des PDF . . . . .	40
2.9	Barèmes complets des pannes configurées . . . . .	43

# Liste des abréviations

UML	Unified Modeling Language
MVP	Minimum Viable Product
RDV	Rendez-vous
CT	Contrôle Technique
MO	Main d'Œuvre
HT	Hors Taxes
TTC	Toutes Taxes Comprises
TVA	Taxe sur la Valeur Ajoutée
PDF	Portable Document Format
CRUD	Create, Read, Update, Delete
ETA	Estimated Time of Arrival
RGPD	Règlement Général sur la Protection des Données
TTFB	Time To First Byte
WCAG	Web Content Accessibility Guidelines

# Introduction générale

Dans un contexte où la digitalisation des services devient indispensable, les garages automobiles doivent moderniser leur gestion pour offrir une meilleure expérience client et optimiser leurs processus internes. La gestion traditionnelle, basée sur des documents papier et des échanges téléphoniques, présente de nombreuses limites en termes de traçabilité, de réactivité et d'efficacité.

Ce projet vise à développer une solution web complète permettant aux clients de déclarer leurs problèmes automobiles, d'obtenir des devis automatiques, de prendre rendez-vous et de suivre l'avancement des réparations. Parallèlement, le système offre aux gestionnaires du garage des outils pour valider les devis, gérer les interventions et générer la facturation.

Réalisé avec Django et suivant une méthodologie Agile Scrum, ce projet s'inscrit dans une démarche de professionnalisation et répond aux exigences d'un projet de fin d'études. Il permettra de démontrer les compétences acquises en développement web, en conception de systèmes et en gestion de projet.

Ce document présente le cadre général du projet, l'analyse des besoins, la planification des sprints selon la méthodologie Scrum, ainsi que les spécifications techniques et fonctionnelles de la solution à développer.

# Chapitre 1

## Cadre Général du Projet

### 1.1 Introduction

Ce chapitre vise à introduire le projet dans son contexte général. Nous présenterons d'abord le contexte et les objectifs du système de gestion de garage, puis nous analyserons l'existant et ses limites. Ensuite, nous définirons la problématique et la solution proposée. Enfin, nous justifierons le choix de la méthodologie Agile avec le framework Scrum pour piloter ce projet.

### 1.2 Contexte général du projet

La gestion d'un garage automobile implique de nombreuses tâches complexes : accueil des clients, diagnostic des pannes, établissement de devis, planification des interventions, suivi des réparations et facturation. Actuellement, ces processus sont souvent gérés de manière manuelle ou à l'aide d'outils disparates (téléphone, documents papier, tableurs Excel), ce qui génère des inefficacités et des risques d'erreurs.

Les clients, de leur côté, manquent de visibilité sur l'état d'avancement de leurs réparations et doivent multiplier les appels pour obtenir des informations. Cette situation impacte négativement leur satisfaction et la réputation du garage.

Dans ce contexte, le développement d'une plateforme web centralisée devient une nécessité pour moderniser la gestion du garage, améliorer la relation client et optimiser les processus internes.

#### 1.2.1 Analyse et critique de l'existant

Actuellement, la gestion des interventions dans la plupart des garages repose sur des méthodes traditionnelles présentant plusieurs inconvénients majeurs :

- **Absence de système centralisé** : Les informations sont dispersées entre différents supports (papier, téléphone, emails), rendant difficile la traçabilité et le suivi.
- **Processus manuel de devis** : L'établissement d'un devis nécessite des calculs manuels, des recherches de tarifs et prend du temps, avec des risques d'erreurs de calcul.
- **Manque de visibilité pour les clients** : Les clients n'ont aucun moyen de suivre l'avancement de leur dossier sans appeler le garage, générant une charge de travail supplémentaire pour le personnel.
- **Gestion inefficace des rendez-vous** : La planification se fait manuellement, avec des risques de double réservation ou d'oubli.
- **Difficulté d'archivage** : La conservation et la recherche des documents (devis, factures) sont fastidieuses.
- **Perte d'informations** : Les échanges verbaux ne laissent pas de trace, ce qui peut poser problème en cas de litige.

Ces limitations entraînent une perte de temps, des erreurs potentielles, une insatisfaction client et une image moins professionnelle pour le garage.

### 1.2.2 Problématique du projet

Face aux limites identifiées dans la gestion traditionnelle d'un garage automobile, la question principale qui se pose est :

**Comment concevoir et développer une solution web permettant d'automatiser et de centraliser la gestion des interventions automobiles, depuis la déclaration des problèmes jusqu'à la facturation, tout en offrant une expérience utilisateur optimale tant pour les clients que pour les gestionnaires du garage ?**

Cette problématique soulève plusieurs sous-questions :

- Comment permettre aux clients de déclarer facilement leurs problèmes sans connaissances techniques ?
- Comment automatiser le calcul des devis de manière fiable et transparente ?
- Comment gérer efficacement les rendez-vous et éviter les conflits ?
- Comment assurer un suivi en temps réel du statut des réparations ?
- Comment faciliter la gestion administrative pour le garage (devis, factures, historique) ?

### 1.2.3 Solution proposée

Pour répondre à cette problématique, nous proposons de développer une application web complète basée sur Django, offrant les fonctionnalités suivantes :

**Pour les clients :**

- Création de compte et authentification sécurisée
- Ajout et gestion de leurs véhicules avec informations complètes
- Déclaration intuitive des problèmes via une interface à boutons (sans saisie de texte libre)
- Génération automatique de devis détaillés (main d'œuvre + pièces)
- Acceptation explicite du devis avant réservation
- Téléchargement des devis au format PDF
- Réservation de créneaux de rendez-vous disponibles
- Annulation ou modification de rendez-vous selon conditions
- Suivi en temps réel de l'avancement des réparations via une timeline
- Notifications in-app des changements de statut
- Accès à l'historique complet de leurs documents (devis, factures)

**Pour les gestionnaires :**

- Tableau de bord centralisé de tous les dossiers
- Filtrage et recherche des dossiers par statut ou client
- Validation et modification des devis avant acceptation
- Gestion des statuts des interventions (Nouveau, Devis émis, Devis accepté, RDV confirmé, En cours, Prêt, Clôturé)
- Génération automatique de factures au format PDF
- Notification automatique des clients (e-mail et in-app) lors des changements de statut
- Clôture et archivage des dossiers terminés

**Pour l'administration :**

- Configuration des groupes de pannes et des barèmes
- Gestion des tarifs (taux horaire, TVA, prix des pièces)
- Définition des créneaux de rendez-vous disponibles (modèles récurrents et exceptions)
- Gestion complète des utilisateurs via l'interface Django Admin

Cette solution permettra de digitaliser l'ensemble du processus, d'améliorer l'expérience client, de réduire les erreurs et d'optimiser la productivité du garage.

## 1.3 Méthodologies de gestion de projet

Le choix d'une méthodologie appropriée est crucial pour la réussite du projet. Deux grandes approches existent : les méthodes traditionnelles (Waterfall) et les méthodes agiles (Scrum, Kanban). Le tableau ci-dessous compare ces approches :

TABLE 1.1 – Comparaison entre les méthodes Waterfall et Agile (Scrum)

Critère	Waterfall	Agile (Scrum)
Approche	Séquentielle et figée	Itérative et évolutive
Planification	Complète en début de projet	Progressive par sprint
Adaptation aux changements	Très difficile après la phase de conception	Très flexible à chaque sprint
Implication des utilisateurs	Limitée (début et fin)	Continue tout au long du projet
Livraisons	Livraison unique en fin de projet	Livraisons progressives et fréquentes
Visibilité	Faible pendant le développement	Haute avec des démos régulières
Risques	Découverte tardive des problèmes	Détection rapide et correction itérative
Documentation	Très complète	Suffisante et évolutive
Taille d'équipe	Adaptée aux grandes équipes	Idéale pour petites équipes (3-9 personnes)
Pertinence pour notre projet	Peu adaptée	Parfaitement adaptée

### 1.3.1 Choix de la méthodologie

Pour ce projet, nous avons choisi d'adopter la méthodologie **Agile avec le framework Scrum** pour les raisons suivantes :

1. **Flexibilité** : Scrum permet d'adapter le projet aux retours des utilisateurs et aux changements de besoins identifiés en cours de développement.
2. **Livraisons incrémentales** : Chaque sprint produit un incrément fonctionnel, permettant de valider progressivement les fonctionnalités et de détecter rapidement les problèmes.

3. **Collaboration renforcée** : Les rituels Scrum (daily stand-up, sprint review, rétrospective) favorisent la communication au sein de l'équipe de 3 personnes.
4. **Visibilité continue** : Le Product Owner et les parties prenantes peuvent suivre l'avancement réel du projet à chaque sprint.
5. **Gestion des risques** : L'approche itérative permet d'identifier et de résoudre rapidement les obstacles techniques ou fonctionnels.
6. **Adaptée aux petites équipes** : Scrum est particulièrement efficace pour des équipes de 3 à 9 personnes comme la nôtre.
7. **Amélioration continue** : Les rétrospectives permettent d'optimiser constamment les processus de travail.

**Organisation Scrum retenue :**

- **Durée des sprints** : 1 à 2 semaines
- **Nombre de sprints** : 3 sprints principaux
- **Rituels** : Sprint Planning, Daily Stand-up (10 min), Sprint Review (démonstration), Sprint Retrospective
- **Rôles** : Product Owner (enseignant/client fictif), Scrum Master (membre de l'équipe en rotation), Development Team (3 étudiants)

## 1.4 Conclusion

Ce premier chapitre a permis de poser le cadre général du projet. Nous avons identifié les limites de la gestion traditionnelle des garages automobiles et défini une solution digitale complète pour y répondre. Le choix de la méthodologie Agile Scrum assure une approche structurée, flexible et collaborative, adaptée à notre contexte de projet académique en équipe réduite. Le chapitre suivant détaillera l'analyse et la spécification des besoins du système.

# Chapitre 2

## Sprint 0 : Analyse et Spécification des Besoins

### 2.1 Introduction

Le Sprint 0, également appelé sprint de préparation, est une phase cruciale qui précède le développement. Il permet d'établir les fondations du projet en définissant clairement les besoins, l'architecture et l'environnement de travail. Ce chapitre présente les résultats de cette phase d'analyse, incluant l'identification des acteurs, la spécification des besoins fonctionnels et non fonctionnels, la modélisation UML, le backlog produit et l'architecture technique du système.

### 2.2 Acteurs du Système

Le système de gestion de garage implique trois types d'acteurs principaux, chacun ayant des rôles et des permissions spécifiques :

TABLE 2.1 – Identification des acteurs du système

Acteur	Description et rôle
<b>Client</b>	<p>Utilisateur externe qui possède un ou plusieurs véhicules. Il peut :</p> <ul style="list-style-type: none"> <li>— Créer un compte et s’authentifier</li> <li>— Ajouter, modifier et consulter ses véhicules</li> <li>— Déclarer des problèmes via l’interface à boutons</li> <li>— Consulter et télécharger ses devis (PDF)</li> <li>— Accepter ou refuser un devis</li> <li>— Réserver, modifier ou annuler un créneau de rendez-vous</li> <li>— Suivre l’évolution du statut de son dossier</li> <li>— Recevoir des notifications in-app</li> <li>— Accéder à l’historique de ses documents (devis, factures)</li> </ul>
<b>Gestionnaire</b>	<p>Employé du garage (garagiste, responsable atelier). Il peut :</p> <ul style="list-style-type: none"> <li>— S’authentifier avec son compte gestionnaire</li> <li>— Visualiser tous les dossiers clients</li> <li>— Filtrer et rechercher des dossiers par statut ou critères</li> <li>— Modifier et valider les devis (ajustement des lignes)</li> <li>— Verrouiller les devis après validation</li> <li>— Gérer les statuts des interventions (progression du workflow)</li> <li>— Définir une estimation de temps (ETA)</li> <li>— Générer des factures au format PDF</li> <li>— Notifier les clients par email et in-app</li> <li>— Clôturer et archiver les dossiers terminés</li> </ul>
<b>Administrateur</b>	<p>Super-utilisateur disposant d’un accès complet au système. Il peut :</p> <ul style="list-style-type: none"> <li>— Accéder à l’interface Django Admin</li> <li>— Gérer les utilisateurs (CRUD clients et gestionnaires)</li> <li>— Configurer les groupes de pannes et sous-pannes</li> <li>— Définir les barèmes (heures de main d’œuvre, prix pièces)</li> <li>— Paramétrer le taux horaire et le taux de TVA</li> <li>— Gérer les créneaux de rendez-vous (modèles récurrents et exceptions)</li> <li>— Superviser toutes les données du système</li> </ul>

## 2.3 Besoins Fonctionnels

Les besoins fonctionnels décrivent les fonctionnalités que le système doit offrir aux utilisateurs. Ils sont organisés par acteur et par module fonctionnel.

### 2.3.1 Besoins liés à l'authentification et la gestion des comptes

- **BF1 - Créer un compte** : Permettre à un nouveau client de créer un compte avec email, nom, prénom et mot de passe.
- **BF2 - Se connecter** : Permettre à tous les utilisateurs de s'authentifier avec email et mot de passe.
- **BF3 - Se déconnecter** : Permettre aux utilisateurs de terminer leur session de manière sécurisée.
- **BF4 - Réinitialiser le mot de passe** : Permettre la récupération de compte via email en cas d'oubli.

### 2.3.2 Besoins liés à la gestion des véhicules (Client)

- **BF5 - Ajouter un véhicule** : Permettre au client d'enregistrer un nouveau véhicule avec : marque, modèle, année, kilométrage, date du dernier contrôle technique, immatriculation (optionnel), surnom du véhicule (obligatoire si pas d'immatriculation, ex : "Voiture principale"), nom de l'assurance, date d'échéance de l'assurance.
- **BF6 - Modifier un véhicule** : Permettre au client de mettre à jour les informations d'un véhicule existant.
- **BF7 - Supprimer un véhicule** : Permettre au client de retirer un véhicule de son compte (suppression logique, conservation pour historique des dossiers).
- **BF8 - Consulter ses véhicules** : Afficher la liste des véhicules enregistrés par le client avec leurs informations principales.
- **BF8 bis - Historique Véhicule** : Traçabilité automatique des événements (création, modification kilométrage, désactivation) pour audit et suivi.

### 2.3.3 Besoins liés à la déclaration des problèmes et devis (Client)

- **BF9 - Sélectionner un véhicule** : Choisir le véhicule concerné par la déclaration de problème.
- **BF10 - Déclarer des problèmes via boutons** : Afficher une interface avec des groupes de pannes (Carrosserie, Pneus, Pare-brise, Moteur, Freinage, Électricité, Vidange, Climatisation) et leurs sous-catégories sous forme de boutons cliquables.

- **BF11 - Valider la sélection** : Empêcher la validation si aucun problème n'est sélectionné (au moins un requis).
- **BF12 - Générer un devis automatique** : Calculer automatiquement le devis basé sur les problèmes sélectionnés en appliquant les barèmes de main d'œuvre et les prix des pièces par défaut.
- **BF13 - Afficher le devis** : Présenter le détail du devis avec les lignes de main d'œuvre, les pièces, les montants HT, TVA et TTC, ainsi que la date de validité (15 jours).
- **BF14 - Télécharger le devis PDF** : Permettre au client de télécharger le devis au format PDF.
- **BF44 - Accepter le devis** : Permettre au client d'accepter explicitement le devis via un bouton "Accepter ce devis". L'acceptation est requise avant de pouvoir réserver un RDV.
- **BF45 - Refuser le devis** : Permettre au client de refuser un devis avec saisie optionnelle d'un motif. Le dossier passe alors en statut "Refusé" et peut être archivé.

### 2.3.4 Besoins liés aux rendez-vous (Client)

- **BF15 - Consulter les créneaux disponibles** : Afficher une liste de créneaux horaires prédéfinis non réservés.
- **BF16 - Réserver un créneau** : Permettre au client de choisir et confirmer un rendez-vous (uniquement si devis accepté).
- **BF17 - Mise à jour automatique** : Rendre indisponible un créneau une fois réservé pour éviter les doubles réservations.
- **BF46 - Modifier un rendez-vous** : Permettre au client de modifier son RDV (changer de créneau) jusqu'à 24 heures avant l'heure prévue.
- **BF47 - Annuler un rendez-vous** : Permettre au client d'annuler son RDV jusqu'à 24 heures avant l'heure prévue. Le créneau redevient disponible. Le dossier revient au statut "Devis accepté".

### 2.3.5 Besoins liés au suivi (Client)

- **BF18 - Consulter ses dossiers** : Afficher la liste des dossiers du client avec leurs statuts.
- **BF19 - Suivre l'évolution** : Présenter une timeline visuelle des statuts (Nouveau → Devis émis → Devis accepté → RDV confirmé → En cours → Prêt → Clôturé).
- **BF20 - Consulter l'ETA** : Afficher l'estimation de fin de réparation si définie par le gestionnaire.

- **BF21 - Télécharger les documents** : Accéder et télécharger les devis et factures de ses dossiers.
- **BF48 - Recevoir des notifications in-app** : Afficher un badge de notification dans le header et un centre de notifications listant les événements récents (changement de statut, devis prêt, véhicule prêt).

### 2.3.6 Besoins liés à la gestion des dossiers (Gestionnaire)

- **BF22 - Visualiser tous les dossiers** : Afficher un tableau récapitulatif de tous les dossiers avec informations clés (client, véhicule, statut, date).
- **BF23 - Filtrer par statut** : Permettre de filtrer l’affichage des dossiers selon leur statut.
- **BF24 - Rechercher un dossier** : Rechercher par nom de client ou immatriculation.
- **BF25 - Consulter le détail d’un dossier** : Afficher toutes les informations d’un dossier (véhicule, problèmes, devis, rendez-vous).

### 2.3.7 Besoins liés à la gestion des devis (Gestionnaire)

- **BF26 - Modifier un devis** : Permettre d’ajuster les lignes de main d’œuvre et de pièces (quantités, prix, ajout/suppression de lignes). Modification possible uniquement si le devis n’est pas encore accepté par le client.
- **BF27 - Valider un devis** : Verrouiller le devis après vérification (non modifiable ensuite par le gestionnaire). Déclenche la notification au client pour acceptation.
- **BF28 - Recalculer automatiquement** : Mettre à jour les totaux HT, TVA, TTC après chaque modification.
- **BF49 - Notifier les modifications majeures** : Si une modification du devis entraîne une variation de plus de 10% du montant TTC, envoyer automatiquement un email de notification au client.

### 2.3.8 Besoins liés à la gestion des statuts (Gestionnaire)

- **BF29 - Changer le statut** : Faire progresser le dossier dans le workflow (En cours, Prêt, Clôturé).
- **BF30 - Définir une ETA** : Indiquer une estimation de date/heure de fin de réparation.
- **BF31 - Historiser les changements** : Conserver un log horodaté des changements de statut.

### 2.3.9 Besoins liés à la facturation (Gestionnaire)

- **BF32 - Générer une facture** : Créer une facture au format PDF à partir du devis validé.
- **BF33 - Numérotation automatique** : Attribuer un numéro unique de facture (format FAC-YYYY-).
- **BF34 - Inclure les informations complètes** : La facture doit contenir : numéro, date d'émission, informations client, véhicule, détail des lignes, totaux HT/TVA/TTC.
- **BF34 bis - Gestion des Paiements** : Enregistrement des paiements (montant, méthode : Espèces/Carte/Virement, statut) liés aux factures.

### 2.3.10 Besoins liés à la notification (Gestionnaire)

- **BF35 - Notifier le client par email** : Envoyer un email automatique au client lors du passage au statut "Prêt" et lors de la validation du devis.
- **BF36 - Email console** : En développement, afficher l'email dans le terminal (backend console).
- **BF50 - Créer une notification in-app** : À chaque changement de statut, créer une entrée dans le centre de notifications du client.

### 2.3.11 Besoins liés à l'archivage (Gestionnaire)

- **BF37 - Clôturer un dossier** : Marquer un dossier comme terminé après paiement/livraison.
- **BF38 - Archiver automatiquement** : Passer le dossier en lecture seule et le conserver pour historique.

### 2.3.12 Besoins liés à la gestion des erreurs

- **BF51 - Erreurs de validation formulaire** : Afficher des messages d'erreur spécifiques sous chaque champ invalide, en français, indiquant clairement le problème (ex : "L'email est déjà utilisé", "Le kilométrage doit être un nombre positif").
- **BF52 - Échec génération PDF** : En cas d'échec de génération d'un PDF, réessayer automatiquement une fois. Si nouvel échec, afficher un message "La génération du document a échoué. Veuillez réessayer." avec un bouton "Réessayer".
- **BF53 - Échec envoi email** : Logger l'erreur dans le système, notifier le gestionnaire via le dashboard, permettre le renvoi manuel de l'email.
- **BF54 - Créneau RDV devenu indisponible** : Si un créneau est réservé par un autre client pendant la navigation, afficher un message "Ce créneau vient d'être

réservé. Veuillez en choisir un autre." et rafraîchir la liste des créneaux disponibles.

- **BF55 - Session expirée** : Rediriger vers la page de connexion avec le message "Votre session a expiré. Veuillez vous reconnecter." et conserver l'URL de destination pour redirection post-connexion.

### 2.3.13 Besoins liés à l'administration (Administrateur)

- **BF39 - Gérer les utilisateurs** : CRUD complet sur les comptes clients et gestionnaires.
- **BF40 - Configurer les pannes** : Créer, modifier, supprimer des groupes de pannes et leurs sous-catégories.
- **BF41 - Définir les barèmes** : Paramétrer les heures de main d'œuvre et les prix des pièces par défaut pour chaque sous-panne.
- **BF42 - Paramétrer les tarifs** : Modifier le taux horaire de main d'œuvre et le taux de TVA.
- **BF43 - Gérer les créneaux RDV** : Définir des modèles de créneaux récurrents (ex : "Tous les lundis de 9h à 10h") et gérer les exceptions (jours fériés, fermetures exceptionnelles).

## 2.4 Besoins Non Fonctionnels

Les besoins non fonctionnels définissent les critères de qualité et les contraintes techniques du système.

### 2.4.1 Ergonomie et accessibilité

- **BNF1 - Interface intuitive** : L'interface doit être simple, claire et ne nécessiter aucune formation particulière.
- **BNF2 - Navigation fluide** : Maximum 3 clics pour accéder à une fonctionnalité principale.
- **BNF3 - Messages explicites** : Afficher des messages d'erreur et de confirmation compréhensibles en français.
- **BNF4 - Responsive design** : L'application doit être utilisable sur ordinateur, tablette et mobile (breakpoints : 576px, 768px, 992px, 1200px).
- **BNF5 - Accessibilité** : Respecter le niveau A des WCAG 2.1 (labels sur les champs, navigation clavier, contraste minimum 4.5 :1, textes alternatifs sur les images).

### 2.4.2 Performance

- **BNF6 - Temps de réponse serveur** : Le TTFB (Time To First Byte) doit être inférieur à 500ms. Le temps de chargement complet des pages principales doit être inférieur à 2 secondes sur une connexion 4G standard (10 Mbps).
- **BNF7 - Calcul de devis** : La génération d'un devis ne doit pas excéder 3 secondes.
- **BNF8 - Génération PDF** : La création d'un PDF (devis ou facture) doit se faire en moins de 5 secondes.

### 2.4.3 Sécurité

- **BNF9 - Authentification** : Utiliser le système d'authentification Django avec sessions sécurisées.
- **BNF10 - Protection CSRF** : Activer la protection contre les attaques CSRF sur tous les formulaires.
- **BNF11 - Validation des entrées** : Valider toutes les données saisies côté serveur (voir section Règles de validation).
- **BNF12 - Isolation des données** : Un client ne peut accéder qu'à ses propres dossiers et véhicules.
- **BNF13 - Politique de mots de passe** :
  - Complexité : minimum 8 caractères, au moins 1 majuscule, 1 minuscule, 1 chiffre
  - Hachage : PBKDF2 avec sel (mécanisme Django par défaut)
  - Expiration session : déconnexion automatique après 30 minutes d'inactivité
  - Verrouillage compte : après 5 tentatives de connexion échouées consécutives, compte verrouillé pendant 15 minutes
- **BNF14 - HTTPS** : Utiliser HTTPS en production (facultatif en développement local).

### 2.4.4 Maintenabilité et évolutivité

- **BNF15 - Code structuré** : Respecter l'architecture MVT de Django et les bonnes pratiques PEP 8.
- **BNF16 - Documentation** : Fournir un README complet avec instructions d'installation et utilisation.
- **BNF17 - Tests** : Écrire des tests unitaires pour les fonctions critiques (calcul devis, génération facture, validation formulaires).
- **BNF18 - Modularité** : Organiser le code en applications Django distinctes par fonctionnalité.

- **BNF19 - Évolutivité** : Concevoir le système pour faciliter l'ajout de nouvelles fonctionnalités.

### 2.4.5 Fiabilité

- **BNF20 - Gestion des erreurs** : Gérer les exceptions et afficher des messages appropriés sans crash (voir BF51-55).
- **BNF21 - Cohérence des données** : Garantir l'intégrité des données (contraintes, transactions).
- **BNF22 - Disponibilité** : Le système doit fonctionner 24/7 en production (objectif 99% de disponibilité).

### 2.4.6 Règles de validation des données

Les données saisies par les utilisateurs doivent respecter les règles suivantes :

TABLE 2.2 – Règles de validation des champs de formulaire

Champ	Format	Contraintes
Email	RFC 5322	Obligatoire, unique en base, max 254 caractères
Mot de passe	Texte	Min 8 car., 1 majuscule, 1 minuscule, 1 chiffre
Nom / Prénom	Texte	Obligatoire, 2-50 caractères, lettres et tirets uniquement
Immatriculation	AA-123-AA	Regex : $\wedge [A-Z] \{2\} - [0-9] \{3\} - [A-Z] \{2\} \$$ , optionnel
Surnom véhicule	Texte	Obligatoire si pas d'immatriculation, 2-30 caractères
Kilométrage	Entier	Obligatoire, $> 0$ , max 999 999
Année véhicule	YYYY	Obligatoire, entre 1950 et année courante
Marque / Modèle	Texte	Obligatoire, 2-50 caractères
Date CT	Date	Format JJ/MM/AAAA, optionnel, ne peut pas être dans le futur de plus d'un an
Nom assurance	Texte	Optionnel, max 100 caractères
Date échéance assurance	Date	Optionnel, format JJ/MM/AAAA

### 2.4.7 Conformité

- **BNF23 - RGPD** : Respecter les principes de base (collecte minimale, droit de suppression via admin, information sur l'utilisation des données à l'inscription).
- **BNF24 - Archivage** : Conserver les données des dossiers clôturés pour l'historique et la comptabilité (minimum 5 ans pour conformité légale).

## 2.5 Diagramme de Cas d'Utilisation Global

Le diagramme de cas d'utilisation (Figure 2.1) représente une vue d'ensemble des interactions entre les acteurs et le système. Il identifie les principales fonctionnalités accessibles à chaque type d'utilisateur.

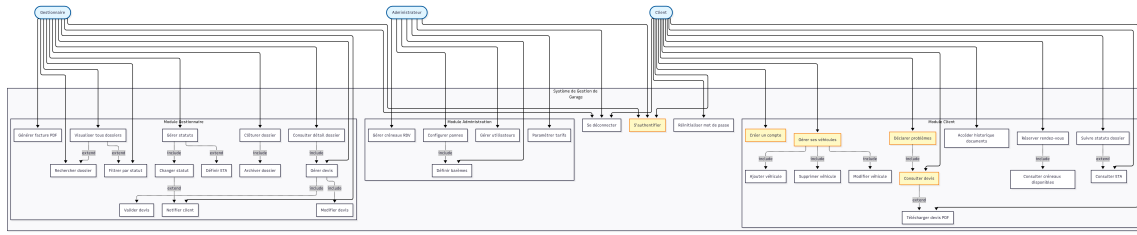


FIGURE 2.1 – Diagramme de cas d'utilisation global du système de gestion de garage

### Description des principaux cas d'utilisation :

- **S'authentifier** : Tous les acteurs doivent s'authentifier pour accéder au système.
- **Gérer véhicules** : Le client peut ajouter, modifier, supprimer et consulter ses véhicules.
- **Déclarer problèmes** : Le client sélectionne les pannes via l'interface à boutons.
- **Consulter/Télécharger devis** : Le client visualise et télécharge ses devis en PDF.
- **Accepter/Refuser devis** : Le client valide ou refuse formellement le devis proposé.
- **Réserver/Modifier/Annuler RDV** : Le client gère ses rendez-vous selon les conditions établies.
- **Suivre statuts** : Le client consulte la timeline d'avancement de son dossier.
- **Consulter notifications** : Le client accède à son centre de notifications in-app.
- **Gérer dossiers** : Le gestionnaire visualise, filtre et gère tous les dossiers clients.
- **Modifier/Valider devis** : Le gestionnaire ajuste et valide les devis avant intervention.
- **Générer facture** : Le gestionnaire crée la facture finale au format PDF.
- **Configurer système** : L'administrateur paramètre les pannes, tarifs et créneaux.

## 2.6 Écrans Principaux de l'Application

Les maquettes détaillées seront réalisées avec Figma durant le Sprint 0. Voici la liste des écrans à concevoir :

### 2.6.1 Écrans communs

- Page d'accueil / Landing page
- Page de connexion
- Page d'inscription (client)
- Page de réinitialisation de mot de passe
- Page d'erreur 404 / 500

### 2.6.2 Espace Client

- Dashboard client (vue d'ensemble des dossiers)
- Liste des véhicules
- Formulaire ajout/modification véhicule
- Interface de déclaration de problèmes (sélection par boutons)
- Page de visualisation du devis (avec actions accepter/refuser)
- Page de sélection de créneau RDV (calendrier)
- Page de suivi du dossier (timeline)
- Centre de notifications
- Historique des documents

### 2.6.3 Espace Gestionnaire

- Tableau de bord gestionnaire (liste des dossiers avec filtres)
- Page de détail d'un dossier
- Formulaire de modification de devis
- Interface de changement de statut
- Page de génération/visualisation de facture

### 2.6.4 Espace Administration

- Interface Django Admin personnalisée
- Gestion des pannes et barèmes
- Gestion des créneaux RDV
- Paramétrage des tarifs

## 2.7 Backlog Produit

Le backlog produit est la liste ordonnée de toutes les fonctionnalités à développer, exprimées sous forme de user stories avec leurs critères d'acceptation. Chaque user story est estimée en termes de complexité et priorisée selon sa valeur métier.

Le tableau 2.3 présente l'ensemble des user stories définies pour le projet, réparties selon les trois sprints de développement.

TABLE 2.3 – Backlog produit avec critères d'acceptation

ID	Acteur	Fonctionnalité	User Story	Critères d'acceptation	Prio.
<b>SPRINT 1 : Fondations &amp; Devis (2 semaines)</b>					
US1	Client	Authentification	En tant que client, je veux créer un compte afin d'accéder aux services du garage.	<p>Formulaire avec email, nom, prénom, mot de passe, confirmation</p> <p>Validation des règles de complexité MDP</p> <p>Email unique vérifié</p> <p>Message de succès affiché</p> <p>Redirection vers login</p>	Haute
US2	Tous	Authentification	En tant qu'utilisateur, je veux me connecter avec mon email et mot de passe afin d'accéder à mon espace.	<p>Formulaire email + MDP</p> <p>Message d'erreur si identifiants invalides</p> <p>Verrouillage après 5 échecs</p> <p>Redirection selon rôle (client/gestionnaire)</p>	Haute
US3	Client	Gestion véhicules	En tant que client, je veux ajouter un véhicule avec toutes ses informations afin de le déclarer au système.	<p>Tous les champs présents</p> <p>Validation selon règles (tableau 6)</p> <p>Surnom obligatoire si pas d'immat</p> <p>Véhicule visible dans la liste</p> <p>Message de confirmation</p>	Haute

ID	Acteur	Fonctionnalité	User Story	Critères d'acceptation	Prio.
US4	Client	Gestion véhicules	En tant que client, je veux consulter la liste de mes véhicules afin de les gérer.	<p>Liste affichée avec marque, modèle, immat/surnom</p> <p>Boutons modifier/supprimer visibles</p> <p>Message si aucun véhicule</p>	Moyenne
US5	Admin	Configuration	En tant qu'admin, je veux créer les groupes de pannes et sous-pannes avec barèmes afin de permettre les devis automatiques.	<p>Interface admin accessible</p> <p>CRUD groupes de pannes</p> <p>CRUD sous-pannes avec heures MO et prix pièces</p> <p>8 groupes minimum créés</p>	Haute
US6	Client	Déclaration problèmes	En tant que client, je veux sélectionner mon véhicule et déclarer les problèmes via des boutons afin d'obtenir un diagnostic.	<p>Dropdown sélection véhicule</p> <p>8 groupes de pannes affichés</p> <p>Sous-pannes en boutons cliquables</p> <p>Changement visuel à la sélection</p> <p>Bouton Valider désactivé si 0 sélection</p>	Haute

ID	Acteur	Fonctionnalité	User Story	Critères d'acceptation	Prio.
US7	Système	Calcul devis	En tant que système, je dois calculer automatiquement le devis (MO + pièces) afin de fournir une estimation au client.	Calcul MO = heures × taux horaire Somme des pièces correcte TVA appliquée (20%) Totaux HT/TVA/TTC exacts Temps < 3 secondes	Haute
US8	Client	Consultation devis	En tant que client, je veux visualiser mon devis détaillé (lignes, totaux HT/TVA/TTC, validité) afin de connaître le coût.	Lignes MO listées Lignes pièces listées Totaux affichés Date validité = +15 jours Boutons Accepter/Refuser visibles	Haute
US9	Client	Téléchargement PDF	En tant que client, je veux télécharger mon devis au format PDF afin de le conserver.	Bouton télécharger fonctionnel PDF généré < 5 secondes Format A4, lisible Toutes infos présentes Numéro DEV-YYYY- ###	Moyenne

ID	Acteur	Fonctionnalité	User Story	Critères d'acceptation	Prio.
US31	Client	Acceptation devis	En tant que client, je veux accepter explicitement le devis afin de pouvoir réserver un RDV.	Bouton "Accepter ce devis" Confirmation demandée Statut passe à "Devis accepté" Accès RDV débloqué Date d'acceptation enregistrée	Haute
<b>SPRINT 2 : RDV &amp; Gestion atelier (2 semaines)</b>					
US10	Admin	Configuration	En tant qu'admin, je veux définir des créneaux de rendez-vous types afin de gérer le planning.	Interface de création modèles récurrents Gestion des exceptions (jours fériés) Créneaux d'1h configurables Plages horaires 9h-12h / 14h-17h	Moyenne
US11	Client	Réservation RDV	En tant que client, je veux consulter les créneaux disponibles afin de choisir mon rendez-vous.	Calendrier/liste des créneaux Créneaux pris grisés Créneaux dispo cliquables Accès uniquement si devis accepté	Haute

ID	Acteur	Fonctionnalité	User Story	Critères d'acceptation	Prio.
US12	Client	Réservation RDV	En tant que client, je veux confirmer un créneau afin de réserver mon intervention.	Clic sur créneau + confirmation Créneau marqué indisponible Statut passe à "RDV confirmé" Récapitulatif RDV affiché Gestion conflit si double résa	Haute
US32	Client	Modification RDV	En tant que client, je veux modifier mon RDV afin de changer de créneau si besoin.	Bouton modifier si > 24h avant Ancien créneau libéré Nouveau créneau réservé Confirmation affichée	Moyenne
US33	Client	Annulation RDV	En tant que client, je veux annuler mon RDV afin de reporter mon intervention.	Bouton annuler si > 24h avant Demande confirmation Créneau redevient disponible Statut revient à "Devis accepté"	Moyenne
US13	Client	Suivi	En tant que client, je veux consulter la timeline des statuts de mon dossier afin de suivre l'avancement.	Timeline visuelle des 7 statuts Statut actuel mis en évidence Dates de passage affichées ETA visible si définie	Haute

ID	Acteur	Fonctionnalité	User Story	Critères d'acceptation	Prio.
US14	Gestionnaire	Tableau de bord	En tant que gestionnaire, je veux visualiser tous les dossiers dans un tableau afin de gérer l'atelier.	Tableau avec colonnes : client, véhicule, statut, date Tri par colonne Pagination si > 20 lignes Clic ouvre détail	Haute
US15	Gestionnaire	Filtrage	En tant que gestionnaire, je veux filtrer les dossiers par statut afin de prioriser mon travail.	Dropdown filtre par statut Filtre "Tous" par défaut Mise à jour instantanée Compteur par statut affiché	Moyenne
US16	Gestionnaire	Gestion devis	En tant que gestionnaire, je veux consulter le détail d'un dossier et son devis afin de le vérifier.	Infos client complètes Infos véhicule complètes Liste problèmes déclarés Détail devis avec lignes Historique statuts visible	Haute
US17	Gestionnaire	Modification devis	En tant que gestionnaire, je veux modifier les lignes du devis (quantités, prix) afin de l'ajuster si nécessaire.	Édition inline des quantités/prix Ajout/suppression de lignes Recalcul auto des totaux Possible si non accepté Notif si variation > 10%	Haute

ID	Acteur	Fonctionnalité	User Story	Critères d'acceptation	Prio.
US18	Gestionnaire	Validation devis	En tant que gestionnaire, je veux valider et verrouiller un devis afin de le soumettre au client.	Bouton "Valider le devis" Devis non modifiable après Statut passe à "Devis émis" Notification client déclenchée	Haute
US19	Gestionnaire	Gestion statuts	En tant que gestionnaire, je veux changer le statut d'un dossier (En cours, Prêt) afin de refléter l'avancement.	Boutons de progression Transitions valides uniquement Horodatage enregistré Notification in-app créée	Haute
US34	Client	Notifications	En tant que client, je veux voir mes notifications afin de suivre mes dossiers sans email.	Badge compteur dans header Centre de notifications accessible Liste chronologique Marquer comme lu Lien vers dossier concerné	Moyenne
<b>SPRINT 3 : Facture &amp; Clôture (2 semaines)</b>					
US20	Gestionnaire	Facturation	En tant que gestionnaire, je veux générer une facture PDF à partir du devis validé afin de finaliser le dossier.	Bouton "Générer facture" PDF créé < 5 secondes Format conforme (voir specs) Téléchargement automatique Copie stockée serveur	Haute

ID	Acteur	Fonctionnalité	User Story	Critères d'acceptation	Prio.
US21	Système	Numérotation	En tant que système, je dois attribuer un numéro unique à chaque facture (FAC-YYYY-###) afin d'assurer la traçabilité.	Format FAC-2025-001 Incrémentation automatique Pas de doublon possible Reset compteur par année	Moyenne
US22	Gestionnaire	Notification	En tant que gestionnaire, je veux notifier automatiquement le client par email quand le véhicule est prêt.	Email envoyé au statut "Prêt" Template avec infos dossier Lien vers suivi inclus Console en dev, SMTP en prod Gestion échec avec retry	Moyenne
US23	Client	Historique	En tant que client, je veux accéder à l'historique de mes documents (devis, factures) afin de les retrouver facilement.	Page historique accessible Liste des devis avec statut Liste des factures Téléchargement PDF possible Tri par date	Moyenne
US24	Gestionnaire	Clôture	En tant que gestionnaire, je veux clôturer un dossier après livraison/paiement afin de l'archiver.	Bouton clôturer (statut Prêt) Confirmation requise Facture doit exister Statut passe à "Clôturé"	Haute

ID	Acteur	Fonctionnalité	User Story	Critères d'acceptation	Prio.
US25	Gestionnaire	Archivage	En tant que gestionnaire, je veux que les dossiers clôturés soient en lecture seule afin de préserver l'historique.	Dossier clôturé = lecture seule Aucune modification possible Documents téléchargeables Conservation 5 ans minimum	Moyenne
US26	Client	Gestion véhicules	En tant que client, je veux modifier ou supprimer mes véhicules afin de maintenir mes informations à jour.	Bouton modifier fonctionnel Formulaire pré-rempli Bouton supprimer avec confirm Suppression logique (historique)	Basse
US27	Développeur	Tests	En tant que développeur, je veux écrire des tests unitaires sur le calcul de devis afin de garantir la fiabilité.	Tests calcul MO Tests calcul pièces Tests calcul TVA Tests cas limites (0, max) Couverture > 80%	Moyenne
US28	Développeur	Tests	En tant que développeur, je veux écrire des tests sur la génération de facture afin d'assurer sa conformité.	Tests numérotation Tests contenu PDF Tests gestion erreurs Tests performance < 5s	Moyenne

ID	Acteur	Fonctionnalité	User Story	Critères d'acceptation	Prio.
US29	Équipe	Documentation	En tant qu'équipe, nous voulons rédiger un README complet afin de faciliter l'installation et l'utilisation.	Prérequis listés Installation pas à pas Création comptes de test Lancement serveur Captures d'écran principales	Moyenne
US30	Admin	Configuration	En tant qu'admin, je veux paramétrer le taux horaire et la TVA via l'interface admin afin d'ajuster les tarifs.	Champ taux horaire éditable Champ taux TVA éditable Validation valeurs positives Effet immédiat nouveaux devis	Basse

## 2.8 Rôles Scrum

Pour la réalisation de ce projet, l'équipe de 3 étudiants adopte les rôles Scrum suivants :

TABLE 2.4 – Répartition des rôles Scrum dans le projet

Rôle	Responsabilité et personne
<b>Product Owner</b>	<b>Enseignant/Client fictif</b> Définit les besoins, priorise le backlog produit, valide les livrables à chaque sprint review.
<b>Scrum Master</b>	<b>Membre de l'équipe (rotation possible)</b> Facilite les rituels Scrum, supprime les obstacles, veille au respect de la méthodologie, anime les réunions.
<b>Development Team</b>	<b>P1 : Spécialiste Backend</b> – Modèles Django, calculs, génération PDF, API <b>P2 : Spécialiste Frontend Client</b> – Templates, formulaires, interface client, CSS <b>P3 : Spécialiste Gestionnaire</b> – Interface gestionnaire, dashboard, administration

**Note** : Bien que chaque membre ait une spécialisation, l'entraide et la polyvalence sont encouragées. Les membres peuvent contribuer aux tâches des autres selon les besoins du sprint.

## 2.9 Planification des Sprints

Le projet est divisé en 3 sprints de 2 semaines chacun, soit un total de 6 semaines de développement. Chaque sprint se termine par une démonstration fonctionnelle et une rétrospective.

TABLE 2.5 – Planification des sprints

Sprint	Période	Objectifs principaux
<b>Sprint 1</b>	Semaines 1-2	<b>Fondations &amp; Devis</b> <ul style="list-style-type: none"> <li>- Mise en place Django et architecture</li> <li>- Authentification et gestion comptes</li> <li>- Gestion véhicules (ajout)</li> <li>- Interface de déclaration (boutons pannes)</li> <li>- Calcul et génération devis automatique</li> <li>- Acceptation/refus devis par client</li> <li>- Téléchargement PDF devis</li> </ul> <b>Livrable</b> : Parcours client fonctionnel jusqu'au devis PDF accepté
<b>Sprint 2</b>	Semaines 3-4	<b>RDV &amp; Gestion atelier</b> <ul style="list-style-type: none"> <li>- Système de rendez-vous (créneaux, réservation, annulation)</li> <li>- Timeline de suivi des statuts côté client</li> <li>- Notifications in-app</li> <li>- Tableau de bord gestionnaire</li> <li>- Filtrage et recherche de dossiers</li> <li>- Modification et validation de devis</li> <li>- Changement de statuts (workflow)</li> </ul> <b>Livrable</b> : Parcours complet devis → RDV → suivi statuts
<b>Sprint 3</b>	Semaines 5-6	<b>Facture &amp; Clôture</b> <ul style="list-style-type: none"> <li>- Génération facture PDF</li> <li>- Numérotation automatique</li> <li>- Notification email (console)</li> <li>- Historique documents client</li> <li>- Clôture et archivage dossiers</li> <li>- Tests unitaires</li> <li>- Documentation README</li> </ul> <b>Livrable</b> : Application complète de bout en bout

### Rituels Scrum appliqués :

- **Sprint Planning** (début de sprint, 1h) : Sélection des user stories, estimation, répartition des tâches.
- **Daily Stand-up** (chaque jour, 10 min) : Point rapide sur hier/aujourd'hui/blocages.
- **Sprint Review** (fin de sprint, 30 min) : Démonstration des fonctionnalités au Product Owner.

- **Sprint Retrospective** (fin de sprint, 20 min) : Analyse du sprint, identification des points d'amélioration.

**Definition of Done (DoD) :**

Une user story est considérée comme terminée (Done) si :

- La fonctionnalité est implémentée et fonctionne sans erreur
- Tous les critères d'acceptation de la user story sont validés
- Les validations et messages d'erreur sont en place (conformes aux BF51-55)
- L'interface respecte le système de design Bootstrap et les principes WCAG 2.1 niveau A
- Le code est commenté et suit les conventions PEP 8
- Les tests unitaires sont écrits et passent (pour les fonctions critiques)
- La documentation README est mise à jour si nécessaire
- La démonstration est possible en live

## 2.10 Architecture du Système et Environnement de Travail

Cette section détaille l'architecture logicielle et physique du système, ainsi que l'environnement de développement utilisé.

### 2.10.1 Architecture logicielle adoptée

Le système adopte l'architecture **MVT (Model-View-Template)** propre à Django, une variante du pattern MVC adaptée au développement web.

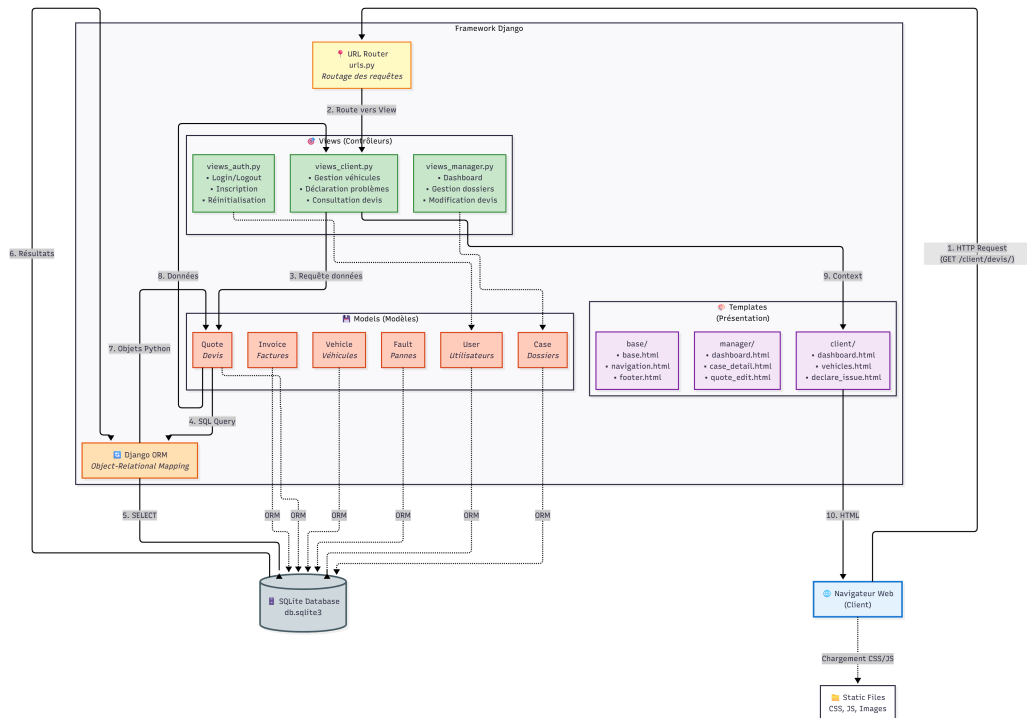


FIGURE 2.2 – Architecture MVT de Django

### Description des composants :

- **Model (Modèle)** : Représente la structure des données et la logique métier. Chaque modèle Django correspond à une table dans la base de données. Les modèles définissent les attributs, les relations et les méthodes métier.
- **View (Vue/Contrôleur)** : Gère la logique applicative. Les vues reçoivent les requêtes HTTP, interagissent avec les modèles, effectuent les traitements nécessaires et retournent des réponses (HTML, JSON, PDF, etc.).
- **Template (Gabarit)** : Responsable de la présentation. Les templates Django utilisent un langage de template pour générer dynamiquement du HTML en insérant les données fournies par les vues.
- **URL Router** : Fait correspondre les URLs aux vues appropriées selon les patterns définis.

### Organisation modulaire :

Le projet est organisé en plusieurs applications Django indépendantes :

- **accounts** : Gestion des utilisateurs, authentification, rôles
- **vehicles** : Gestion des véhicules clients
- **catalog** : Configuration des pannes, barèmes, tarifs
- **cases** : Dossiers, statuts, timeline
- **quotes** : Devis et lignes de devis
- **appointments** : Rendez-vous et créneaux
- **billing** : Factures et génération PDF

- **notifications** : Notifications in-app et gestion emails

### 2.10.2 Architecture physique

L'application suit une architecture **trois tiers (3-tier)** :

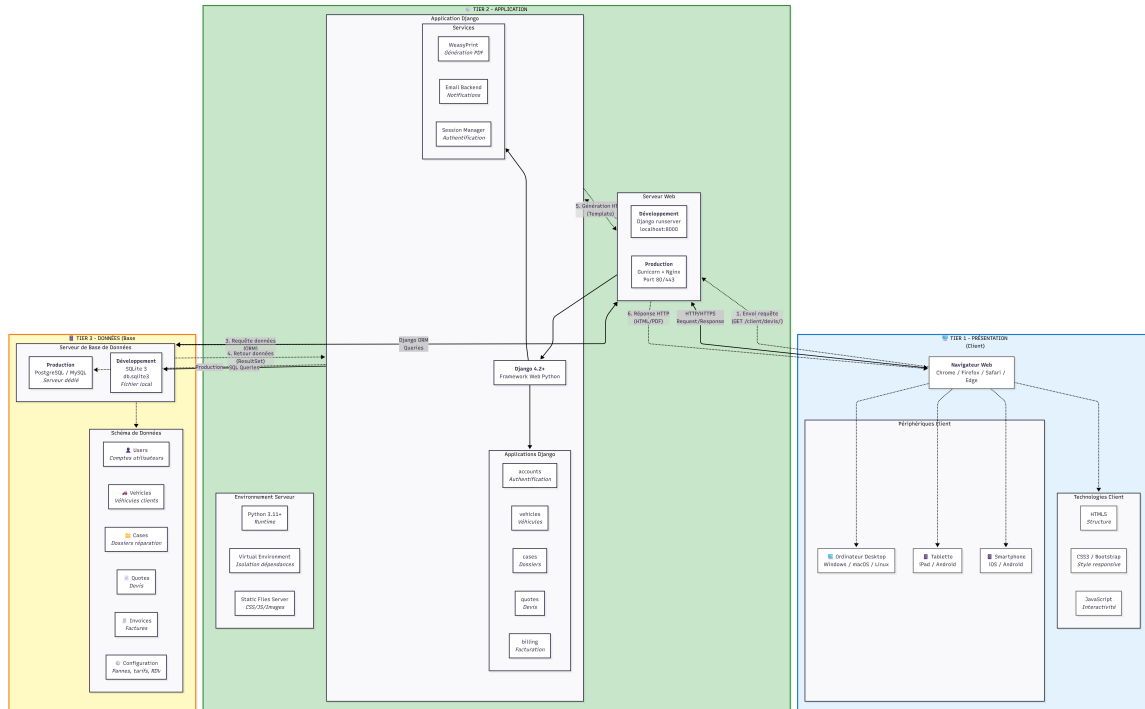


FIGURE 2.3 – Architecture physique trois tiers

### Description des tiers :

1. **Tier 1 - Couche présentation (Client) :**
  - Navigateur web moderne (Chrome, Firefox, Safari, Edge)
  - Interface HTML/CSS générée par les templates Django
  - JavaScript pour interactions dynamiques (optionnel)
  - Responsive design (Bootstrap ou Tailwind CSS)
2. **Tier 2 - Couche application (Serveur) :**
  - Serveur Django (développement : runserver, production : Gunicorn/uWSGI)
  - Traitement de la logique métier
  - Génération des PDF (WeasyPrint ou xhtml2pdf)
  - Gestion des sessions et authentification
  - Routage des URLs et traitement des requêtes
3. **Tier 3 - Couche données (Base de données) :**
  - SQLite en développement (fichier db.sqlite3)
  - PostgreSQL ou MySQL envisageable en production

- ORM Django pour l'abstraction des requêtes SQL
- Migrations automatiques pour évolution du schéma

### **2.10.3 Modèle de données conceptuel**

Le schéma ci-dessous présente les principales entités du système et leurs relations :

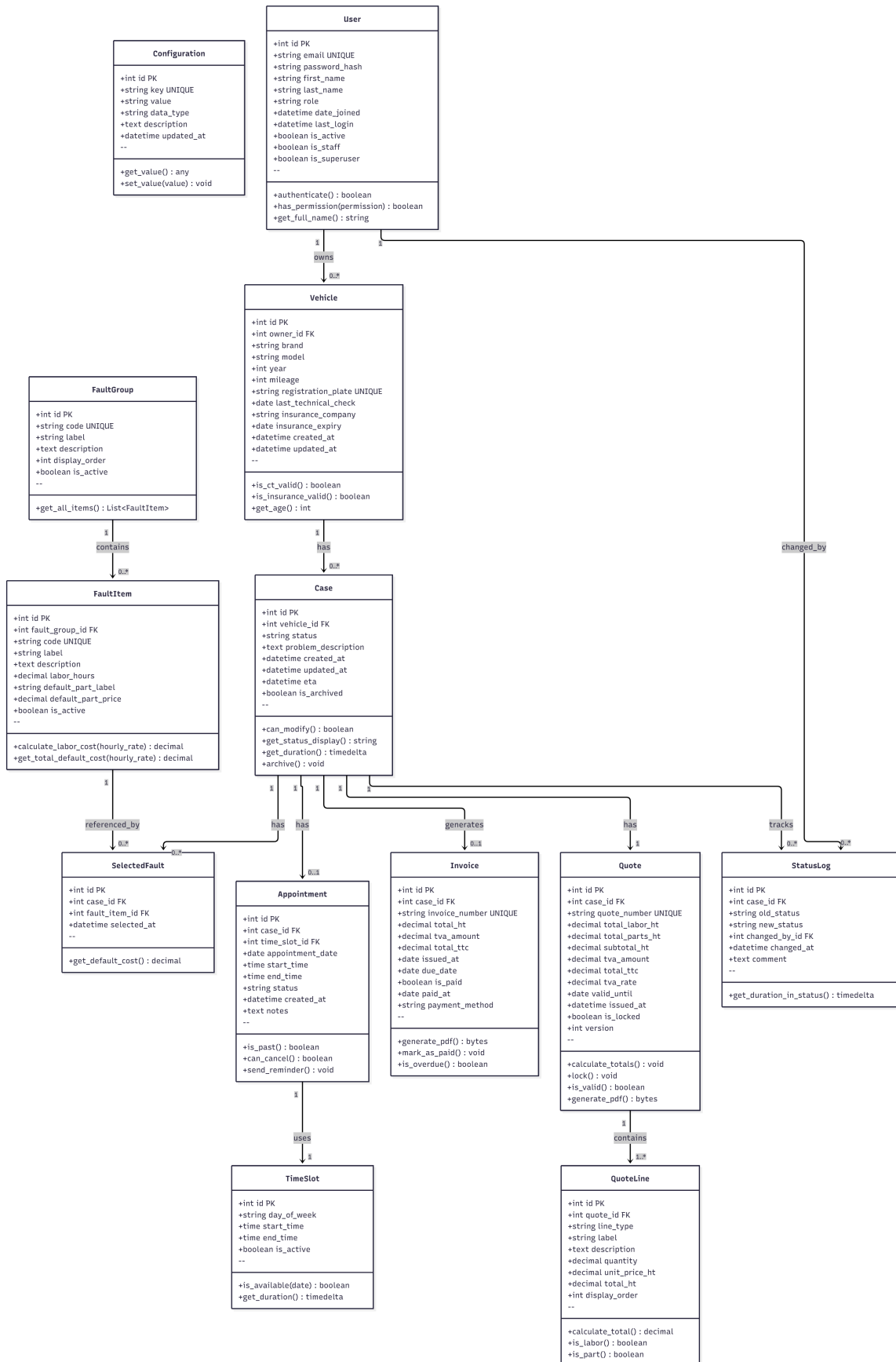


FIGURE 2.4 – Modèle de données conceptuel - Diagramme de classes UML

## 2.10.4 Environnement de développement

### Environnement matériel

Le développement peut être réalisé sur tout ordinateur moderne répondant aux spécifications minimales suivantes :

TABLE 2.6 – Configuration matérielle minimale

Composant	Spécification minimale
Processeur	Intel Core i3 / AMD Ryzen 3 ou supérieur
Mémoire RAM	4 Go minimum (8 Go recommandé)
Stockage	10 Go d'espace disque disponible
Système d'exploitation	Windows 10/11, macOS 10.15+, Linux (Ubuntu 20.04+)

## Environnement logiciel

TABLE 2.7 – Technologies et outils utilisés

Technologie/Outil	Version et description
<b>Python</b>	Version 3.11 ou supérieure Langage de programmation principal
<b>Django</b>	Version 4.2 ou supérieure (LTS) Framework web full-stack pour Python
<b>SQLite</b>	Version 3 (incluse avec Python) Base de données relationnelle légère pour le développement
<b>Bootstrap / Tailwind</b>	Dernière version stable Framework CSS pour le design responsive
<b>xhtml2pdf</b>	Version 0.2+ Bibliothèque de conversion HTML vers PDF (alternative compatible Windows à WeasyPrint)
<b>Git</b>	Version 2.40+ Système de contrôle de version
<b>VS Code</b>	Dernière version Éditeur de code avec extensions Python et Django
<b>Figma</b>	Version web Outil de design UI/UX pour les maquettes
<b>Draw.io</b>	Version web ou desktop Création de diagrammes UML
<b>Postman</b>	Version desktop Tests des endpoints API (si nécessaire)

### Dépendances Python (requirements.txt) :

```
Django>=4.2
pillow>=10.0
xhtml2pdf>=0.2
django-crispy-forms>=2.0
crispy-bootstrap5>=1.0
```

## 2.11 Règles de Gestion et Calculs

Cette section détaille les règles métier et les formules de calcul appliquées par le système.

### 2.11.1 Calcul du devis

Le montant d'un devis est calculé automatiquement selon les règles suivantes :

#### 1. Main d'œuvre (MO) :

$$MO = \sum_{i=1}^n (\text{heures\_barème}_i) \times \text{taux\_horaire} \quad (2.1)$$

Où :

- $n$  = nombre de sous-pannes sélectionnées
- heures\_barème = temps standard associé à chaque sous-panne
- taux\_horaire = tarif horaire du garage (paramétrable, par défaut 60€/h)

#### 2. Pièces :

$$\text{Pièces} = \sum_{j=1}^m \text{prix\_pièce\_défaut}_j \quad (2.2)$$

Où :

- $m$  = nombre de pièces nécessaires
- prix\_pièce\_défaut = prix prédéfini pour chaque type de pièce

#### 3. Montant Hors Taxes (HT) :

$$\text{Montant HT} = MO + \text{Pièces} \quad (2.3)$$

#### 4. TVA :

$$\text{TVA} = \text{Montant HT} \times \text{taux\_TVA} \quad (2.4)$$

Où taux\_TVA est paramétrable (par défaut 20% = 0.20)

#### 5. Montant Toutes Taxes Comprises (TTC) :

$$\text{Montant TTC} = \text{Montant HT} + \text{TVA} \quad (2.5)$$

#### 6. Validité du devis :

$$\text{Date de validité} = \text{Date d'émission} + 15 \text{ jours} \quad (2.6)$$

### 2.11.2 Gestion de l'expiration des devis

- Un devis non accepté après sa date de validité est marqué comme "Expiré"
- Un devis expiré affiche un badge visuel "Expiré" sur l'interface client
- L'acceptation d'un devis expiré est impossible (bouton désactivé)
- Le client peut demander un nouveau devis pour le même véhicule (les tarifs peuvent avoir changé)
- Les devis expirés sans suite sont automatiquement archivés après 30 jours

### 2.11.3 Exemple de calcul

Client sélectionne :

- Pneus → Usure (0.2h, 2 pneus = 200€)
- Pare-brise → Fissure (1.2h, Pare-brise = 150€)

Calculs :

- MO =  $(0.2 + 1.2) \times 60\text{€} = 1.4 \times 60 = 84\text{€}$
- Pièces =  $200\text{€} + 150\text{€} = 350\text{€}$
- HT =  $84\text{€} + 350\text{€} = 434\text{€}$
- TVA (20%) =  $434\text{€} \times 0.20 = 86.80\text{€}$
- TTC =  $434\text{€} + 86.80\text{€} = \mathbf{520.80\text{€}}$

### 2.11.4 Numérotation des documents

**Devis :**

$$\text{Numéro devis} = \text{DEV-YYYY-###} \quad (2.7)$$

Exemple : DEV-2025-001, DEV-2025-002...

**Factures :**

$$\text{Numéro facture} = \text{FAC-YYYY-###} \quad (2.8)$$

Exemple : FAC-2025-001, FAC-2025-002...

Où YYYY = année en cours, ### = compteur séquentiel sur 3 chiffres avec zéros initiaux.

### 2.11.5 Spécifications des documents PDF

Les documents PDF (devis et factures) respectent les spécifications suivantes :

TABLE 2.8 – Spécifications de mise en page des PDF

Élément	Spécification
Format	A4 portrait (210 × 297 mm)
Marges	20 mm de chaque côté
Police	Arial ou équivalent sans-serif
Taille police corps	10-11 pt
Taille police titres	14-16 pt

**Contenu de l'en-tête (commun devis/facture) :**

- Logo du garage (placeholder 150×50 px)
- Nom du garage : "Garage Auto Express"
- Adresse : 123 Avenue de la Réparation, 75001 Paris
- Téléphone : 01 23 45 67 89
- Email : contact@garage-auto-express.fr
- SIRET : 123 456 789 00012 (fictif)

**Contenu du pied de page :**

- Numéro de page (Page X/Y)
- Mention légale : "TVA non applicable, art. 293 B du CGI" ou "TVA : FR12 123456789"
- Conditions de paiement (facture uniquement)

**Corps du devis :**

- Numéro et date d'émission
- Date de validité
- Informations client (nom, adresse, email, téléphone)
- Informations véhicule (marque, modèle, immatriculation/surnom, kilométrage)
- Tableau des lignes : description, quantité, prix unitaire HT, total HT
- Récapitulatif : Total HT, TVA (taux + montant), Total TTC
- Mention "Devis gratuit et sans engagement"

**Corps de la facture :**

- Numéro et date d'émission
- Référence du devis associé
- Informations client et véhicule (idem devis)
- Tableau des lignes (idem devis)
- Récapitulatif avec totaux
- Modalités de paiement : "Payable à réception"
- Mention "En cas de retard de paiement, une pénalité de 3 fois le taux d'intérêt légal sera appliquée"

## 2.11.6 Configuration et templates email

### Configuration technique :

- Développement : `EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'`
- Production : SMTP avec paramètres serveur (à configurer)
- Expéditeur : `noreply@garage-auto-express.fr`

### Template "Devis disponible" (BF35) :

- Objet : "[Garage Auto Express] Votre devis n°{numero\_devis} est prêt"
- Corps : Bonjour {prenom}, votre devis pour {marque} {modele} est disponible.  
Montant TTC : {montant}€. Valable jusqu'au {date\_validite}. [Lien vers espace client]

### Template "Véhicule prêt" (BF35) :

- Objet : "[Garage Auto Express] Votre véhicule est prêt!"
- Corps : Bonjour {prenom}, les réparations de votre {marque} {modele} sont terminées. Vous pouvez venir récupérer votre véhicule. [Lien vers espace client]

### Template "Confirmation inscription" (BF1) :

- Objet : "[Garage Auto Express] Bienvenue!"
- Corps : Bonjour {prenom}, votre compte a été créé avec succès. [Lien connexion]

### Template "Réinitialisation mot de passe" (BF4) :

- Objet : "[Garage Auto Express] Réinitialisation de votre mot de passe"
- Corps : Bonjour, cliquez sur le lien suivant pour réinitialiser votre mot de passe.  
Ce lien expire dans 24h. [Lien reset]

## 2.11.7 Stockage et rétention des données

### Organisation du stockage fichiers :

```
/media/  
documents/  
  {user_id}/  
    devis/  
      {année}/  
        DEV-2025-001.pdf  
    factures/  
      {année}/  
        FAC-2025-001.pdf  
  ...  
...
```

### Politique de rétention :

- Devis acceptés et factures : 10 ans (obligation légale comptable française)
- Devis refusés ou expirés : 2 ans puis suppression
- Données personnelles des comptes inactifs : anonymisation après 3 ans d'inactivité
- Logs système : 1 an

**Sauvegarde (développement) :**

- Copie quotidienne du fichier db.sqlite3
- Script de backup automatique recommandé en production

### 2.11.8 Workflow des statuts

Le dossier suit un cycle de vie strict avec les statuts suivants (dans l'ordre) :

1. **NOUVEAU** : Dossier créé, problèmes déclarés, devis en cours de génération
2. **DEVIS\_\_ÉMIS** : Devis généré, validé par gestionnaire, en attente d'acceptation client
3. **DEVIS\_\_ACCEPTÉ** : Client a accepté le devis, peut réserver un RDV
4. **RDV\_\_CONFIRMÉ** : Client a réservé un créneau
5. **EN\_\_COURS** : Intervention en cours dans l'atelier
6. **PRÊT** : Véhicule réparé, prêt pour récupération
7. **CLÔTURÉ** : Dossier terminé, véhicule livré, facture générée

**Statuts alternatifs :**

- **REFUSÉ** : Client a refusé le devis (archivage possible)
- **EXPIRÉ** : Devis non accepté après date de validité

**Règles de transition :**

- Progression normale : chaque statut ne peut évoluer que vers le suivant dans la séquence
- Retour arrière autorisé : uniquement de EN\_COURS vers RDV\_CONFIRMÉ en cas d'erreur de manipulation (via gestionnaire)
- Retour arrière interdit : après génération de facture, aucun retour arrière possible
- Annulation RDV : fait revenir de RDV\_CONFIRMÉ à DEVIS\_ACCEPTÉ (via action client)

## 2.12 Jeux de Données de Démonstration

Pour faciliter les tests et la démonstration, des données de base seront créées :

### 2.12.1 Comptes utilisateurs

- **Admin** : admin@garage.com / Admin123!
- **Gestionnaire** : garagiste@garage.com / Gest123!
- **Client 1** : client1@example.com / Client123!
- **Client 2** : client2@example.com / Client123!

### 2.12.2 Groupes de pannes et sous-pannes

TABLE 2.9 – Barèmes complets des pannes configurées

Groupe	Sous-panne	Heures	Pièce	Prix
3*Carrosserie	Rayure légère	0.5h	Kit polish	25€
	Bosse	1.5h	-	0€
	Peinture portière	3.0h	Peinture	120€
2*Pneus	Usure	0.2h	2 pneus	200€
	Vibrations	0.3h	-	0€
2*Pare-brise	Fissure	1.2h	Pare-brise	150€
	Impact	0.4h	Résine	25€
2*Moteur	Surchauffe	2.0h	Thermostat	80€
	Bruit anormal	1.0h	-	0€
2*Freinage	Plaquettes usées	1.0h	Plaquettes	120€
	Disques usés	1.5h	Disques	180€
2*Électricité	Batterie faible	0.3h	Batterie	100€
	Phare HS	0.4h	Ampoule	30€
Vidange	Vidange standard	0.5h	Huile + filtre	45€
2*Climatisation	Recharge clim	0.5h	Gaz R134a	60€
	Fuite clim	1.5h	Joint + gaz	90€

### 2.12.3 Créneaux de rendez-vous

**Modèle récurrent (lundi-vendredi) :**

Créneaux d'une heure :

- 09 :00 - 10 :00
- 10 :00 - 11 :00
- 11 :00 - 12 :00
- 14 :00 - 15 :00
- 15 :00 - 16 :00
- 16 :00 - 17 :00

**Exceptions pré-configurées :**

- Jours fériés 2025 : 1er janvier, lundi de Pâques, 1er mai, 8 mai, Ascension, lundi de Pentecôte, 14 juillet, 15 août, 1er novembre, 11 novembre, 25 décembre

## 2.12.4 Paramètres système

- Taux horaire : 60€/h
- Taux TVA : 20%
- Validité devis : 15 jours
- Délai annulation RDV : 24 heures
- Expiration session : 30 minutes
- Tentatives connexion avant verrouillage : 5

## 2.13 Conclusion

Ce chapitre a permis d'établir les fondations solides du projet en définissant précisément les acteurs, les besoins fonctionnels et non fonctionnels, ainsi que l'architecture technique. Le backlog produit structuré en 34 user stories réparties sur 3 sprints, incluant les critères d'acceptation pour chaque story, offre une feuille de route claire et testable pour le développement.

Les diagrammes UML, les spécifications détaillées des documents PDF, les templates email et les règles de validation garantissent que chaque développeur dispose des informations nécessaires pour implémenter les fonctionnalités sans ambiguïté.

L'environnement technique basé sur Django, SQLite et WeasyPrint est adapté aux objectifs du MVP. Les règles de calcul, le workflow définis et la gestion des cas d'erreur garantissent la cohérence métier et la robustesse du système.

Nous sommes maintenant prêts à démarrer le Sprint 1 avec une vision claire des objectifs à atteindre et des critères de validation pour chaque fonctionnalité.

# Conclusion Générale

Ce cahier des charges a présenté de manière exhaustive le projet de système de gestion de garage automobile, depuis la définition du contexte jusqu'à la planification détaillée du développement.

Nous avons identifié une problématique concrète : la nécessité de digitaliser et d'automatiser les processus de gestion des interventions automobiles pour améliorer l'efficacité opérationnelle du garage et la satisfaction des clients. La solution proposée, une application web Django complète, répond à cette problématique en offrant des fonctionnalités adaptées aux besoins de chaque acteur (clients, gestionnaires, administrateurs).

Le choix de la méthodologie Agile Scrum assure une approche itérative et flexible, particulièrement adaptée à notre contexte de projet académique avec une équipe réduite. La planification en 3 sprints de 2 semaines permet des livraisons progressives et une validation continue des fonctionnalités développées grâce aux critères d'acceptation définis pour chaque user story.

L'architecture MVT de Django, combinée à une organisation modulaire en applications indépendantes, garantit la maintenabilité et l'évolutivité du système. Les spécifications détaillées des besoins fonctionnels et non fonctionnels, les règles de validation des données, les templates de documents PDF et d'emails, ainsi que les règles de gestion clairement définies, constituent un guide précis et non ambigu pour l'implémentation.

Les prochaines étapes consisteront à réaliser les sprints planifiés, en commençant par le Sprint 1 qui établira les fondations du système (authentification, gestion véhicules, déclaration problèmes, génération devis avec acceptation client). Chaque sprint se conclura par une démonstration fonctionnelle validant les critères d'acceptation et une rétrospective pour améliorer continuellement nos pratiques.

Ce projet représente une opportunité de mettre en pratique les compétences acquises en développement web, conception de systèmes d'information, gestion de projet Agile et travail en équipe. Il démontre également notre capacité à analyser un besoin réel, à proposer une solution technique appropriée et à la réaliser de manière professionnelle.

Nous sommes confiants que ce système apportera une réelle valeur ajoutée aux garages automobiles en modernisant leur gestion et en offrant une expérience client optimale.

# Bibliographie

- [1] Django Software Foundation, *Django Documentation*.  
Consulté en janvier 2025.  
<https://docs.djangoproject.com/>
- [2] Scrum.org, *The Scrum Guide*.  
Consulté en janvier 2025.  
<https://www.scrum.org/resources/scrum-guide>
- [3] Object Management Group, *Unified Modeling Language (UML)*.  
Consulté en janvier 2025.  
<https://www.uml.org/>
- [4] xhtml2pdf Documentation, *HTML to PDF converter*.  
Consulté en janvier 2025.  
<https://github.com/xhtml2pdf/xhtml2pdf>
- [5] Bootstrap Team, *Bootstrap Framework Documentation*.  
Consulté en janvier 2025.  
<https://getbootstrap.com/>
- [6] Agile Alliance, *Agile Manifesto and Principles*.  
Consulté en janvier 2025.  
<https://www.agilealliance.org/>
- [7] W3C, *Web Content Accessibility Guidelines (WCAG) 2.1*.  
Consulté en janvier 2025.  
<https://www.w3.org/TR/WCAG21/>