Année Universitaire: 2023/2024

École Supérieure de Management Appliqué

ESMA
Ecole Supérieure de Marrakech

Rapport

de Projet de Fin d'Étude

Présenté en vue de l'obtention du diplôme de
Master professionnel

Sujet

Développement d'une application de gestion d'agences de location de voitures

Élaboré par
QOREICHI Ahmed

Organisme d'accueil

FOXDIGIA
Technological Vision Defining the Future

FOXDIGIA

Encadrée par
SAJI Ismail

Remerciements

Je tiens tout d'abord a adresser mes plus vifs remerciements a mon encadreur Monsieur SAJI Ismail pour son encadrement ses conseils et ses corrections.

J’adresse mes vifs remerciements a Madame KERFA, Zakaria qui m'a soutenue tout au long de mon travail.

Je tiens également a remercier et exprimer mon profond respect aux membres de jury d’avoir accepte de juger ce travail.

Introduction generale

L'informatique est un domaine d'activité technique et industriel consacré au traitement de l'information. Il englobe l'ensemble des ressources d'une organisation : personnel, matériel et logiciels, pour collecter, stocker, traiter et diffuser les informations.

De nos jours, de nombreuses entreprises sont confrontées à des problèmes liés à la gestion des données et à la nécessité d'une application informatique capable de résoudre ces difficultés.

La puissance de l'informatique offre de multiples possibilités. L'évolution des logiciels permet d'automatiser les procédures et de fournir les informations adéquates en temps voulu.

À cet effet, notre projet de fin d'études consiste à développer une application web pour la gestion des agences de location de voitures.

Notre travail se concentre sur le développement d'une solution web permettant aux agences de gérer efficacement leur flotte de véhicules, les clients, les réservations, les paiements, ainsi que la planification des contrôles techniques des voitures. L'application inclura également des fonctionnalités pour le calcul automatique des coûts et des charges opérationnelles.

Notre rapport est élaboré en adoptant la méthodologie Scrum et se compose de quatre chapitres, en plus d'une introduction générale et d'une conclusion:

- Le premier chapitre, intitulé « Étude du projet », contient la présentation générale du projet, en commençant par la présentation de l'organisme d'accueil ainsi que du projet, suivie de la méthode de travail adoptée.
- Le deuxième chapitre, intitulé « Planification du projet », également appelé sprint 0, est consacré au recueil des besoins et à la planification complète des tâches.
- Le troisième chapitre, nommé « Réalisation du projet », est dédié au développement de l'application.
- Le dernier chapitre, intitulé « Phase de clôture », présente les différents outils et technologies utilisés pour le développement.

Chapitre 1 : Étude du projet

Introduction

Dans ce chapitre, nous présentons l'organisme d'accueil où s'est déroulé le stage, puis le cadre du projet. Nous y étudions l'existant en décrivant la situation actuelle, la solution proposée, et enfin, la méthode de travail adoptée.

Présentation de l'organisme d'accueil

1. Présentation

FOXDIGIA est une agence web spécialisée dans la création de sites internet, le développement d'applications web, et la stratégie digitale. Fondée en 2018, elle accompagne ses clients dans la réalisation de leurs projets numériques.

2. Organisation

L'agence est structurée comme suit :

- **Direction générale**: Elle est assistée par un coordinateur qui veille à la bonne transmission des informations entre les différents départements.
- **Service administratif et financier**: Ce service est en charge de la gestion administrative, comptable, et financière de l'agence, en veillant à la bonne gestion des ressources, à la rentabilité, et à la solvabilité de l'entreprise.
- **Service de création et design**: Les designers et créateurs s'occupent de la conception visuelle et ergonomique des sites web et des interfaces utilisateur pour offrir une expérience client optimale.
- **Service de développement web**: Ce service est dédié à la création de sites web, d'applications web sur mesure, et à l'intégration de solutions numériques adaptées aux besoins des clients.
- **Service marketing digital**: Il s'occupe de la stratégie digitale, du référencement (SEO), de la gestion des campagnes publicitaires en ligne et de l'accompagnement des clients dans leur visibilité sur le web.

3. Organigramme

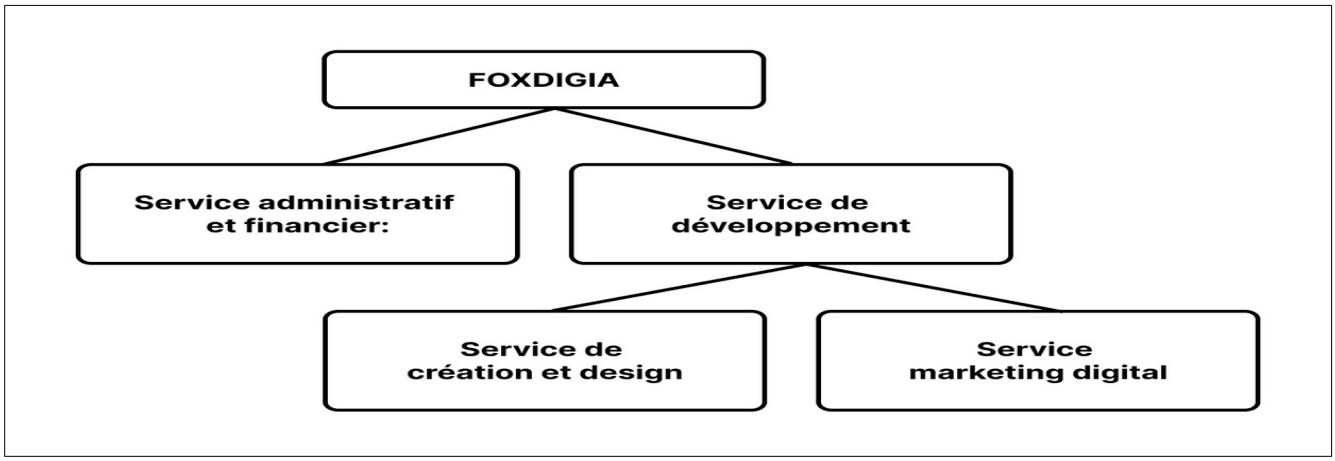

Figure 1: Organigramme FOXDIGIA

4. Activités

Depuis sa création, la direction générale de « FOXDIGIA » a choisi d'offrir à ses clients des services de création de sites web et de développement numérique à forte valeur ajoutée, en adoptant une approche pragmatique et axée sur les résultats.

FOXDIGIA s'appuie sur un réseau de designers, développeurs et experts en marketing digital hautement qualifiés, disposant d'une expérience significative dans le domaine, afin de répondre efficacement aux besoins de ses clients.

Les clients attendent de l'agence des solutions pratiques et opérationnelles pour atteindre leurs objectifs en ligne, plutôt que des théories sans résultats concrets.

Parmi les activités de l'agence, nous pouvons citer :

- Création et développement de sites web sur mesure.
- Optimisation du référencement naturel (SEO) pour améliorer la visibilité en ligne.
- Développement d'applications web et mobiles adaptées aux besoins spécifiques des clients.
- Conception d'identités visuelles et de supports marketing numériques.
- Formation et accompagnement des clients sur les outils numériques et les meilleures pratiques en matière de marketing digital.

Présentation du projet

1. Cadre du projet

Ce projet s'inscrit dans le cadre du projet de fin d'études pour l'obtention d'un diplôme de mastère professionnel à l'École Supérieure de Management Appliqué.

L'objectif de ce projet est de développer une application web nommée « Cars Rental », qui permettra de :

- Gérer les utilisateurs en créant et en administrant des comptes pour les administrateurs et les clients.
- Administrer les entreprises de location en enregistrant et en gérant plusieurs sociétés de location de voitures avec leurs informations détaillées.
- Gérer le parc automobile en ajoutant, modifiant et supprimant des véhicules, tout en maintenant des informations sur la marque, le modèle, l'année, le tarif journalier et l'état général.
- Faciliter la réservation en permettant aux clients de créer et de gérer des réservations, y compris la sélection des dates de prise en charge et de retour.
- Traiter les paiements en gérant les transactions liées aux réservations, y compris le montant total et les paiements effectués.
- Gérer les commentaires et les tickets de support en offrant aux clients la possibilité de laisser des commentaires et de soumettre des tickets.
- Suivre les récupérations de véhicules en enregistrant l'état des véhicules lors de leur retour, y compris le kilométrage et le niveau de carburant.
- Administrer les charges additionnelles en gérant les frais supplémentaires associés aux locations de véhicules.
- Fournir des rapports et des analyses en générant des rapports sur les réservations et les paiements.
- Envoyer des notifications et des rappels en mettant en place un système de notifications pour informer l'agence des réservations terminées et des rappels à venir.

2. Étude de l’existant

L’étude de l’existant est une étape primordiale qui permet de définir les forces et les faiblesses des systèmes de gestion des réservations de véhicules. Cette analyse est essentielle pour identifier les besoins des agences de location et les prendre en considération lors de la conception et de la réalisation de notre application.

2.1. Description de l’existant

Actuellement, la gestion des réservations de véhicules se fait de manière traditionnelle et manuelle, sans l'utilisation d'un logiciel spécifique, ce qui entraîne plusieurs problèmes.

La gestion des réservations repose sur des processus manuels. Les agences traitent les réservations par téléphone ou par e-mail, et les informations sont souvent enregistrées sur papier. Lorsqu'un client effectue une réservation, un dossier est créé pour chaque transaction, mais ces dossiers sont rarement centralisés. Chaque réservation nécessite des informations

détaillées, telles que la date de prise en charge, la date de retour, et les détails du client, qui peuvent être éparpillées dans différents documents.

Cette méthode entraîne un manque d'efficacité et de transparence, car il est difficile pour les agences de suivre les réservations en cours, de gérer les paiements et de communiquer efficacement avec les clients. De plus, le suivi des récupérations de véhicules et la gestion des états des véhicules lors de leur retour ne sont pas documentés de manière systématique, ce qui peut engendrer des litiges.

En conséquence, cette étude de l'existant met en lumière la nécessité d'une solution numérique intégrée pour améliorer la gestion des réservations, optimiser la communication et offrir une meilleure visibilité sur les opérations des agences de location.

Le travail se déroule de la manière suivante :

1. Analyse des besoins des agences de location pour identifier les exigences spécifiques liées à la gestion des réservations.
2. Élaboration d'une architecture technique pour l'application, en définissant les entités principales et leur interrelation.
3. Développement des fonctionnalités de l'application, y compris la gestion des réservations, le traitement des paiements, et le suivi des véhicules.
4. Intégration d'un système de notifications pour alerter les agences sur les réservations à venir et les récupérations de véhicules.
5. Réalisation de tests fonctionnels pour valider chaque fonctionnalité et s'assurer de leur conformité aux exigences.
6. Déploiement de l'application, permettant aux agences de commencer à l'utiliser dans leurs opérations quotidiennes.
7. Formation des utilisateurs sur l'application pour garantir une adoption efficace et une utilisation optimale.
8. Mise en place d'un suivi post-déploiement pour résoudre rapidement les problèmes et recueillir les retours des utilisateurs.
9. Évaluation continue de l'application et mises à jour régulières pour répondre aux besoins évolutifs du marché.

2.2. Critique de l'existant

La critique de l'existant permet d'évaluer les méthodes actuelles de gestion des réservations de véhicules, en mettant en lumière les lacunes et les inefficacités des systèmes en place.

1. **Manque d'efficacité** : Les processus manuels entraînent un temps considérable pour traiter chaque réservation, ce qui ralentit le service client et réduit la capacité d'une agence à gérer un grand volume de réservations.

2. **Risque d’erreurs humaines** : L’enregistrement des informations sur papier ou par e-mail augmente le risque d'erreurs, telles que des données manquantes ou incorrectes, ce qui peut causer des problèmes lors des transactions et des communications avec les clients.
3. **Absence de centralisation des données** : Les informations sont dispersées à travers différents documents, rendant difficile l'accès rapide aux données critiques. Cela complique également la prise de décisions informées basées sur des données consolidées.
4. **Communication inefficace** : La dépendance aux canaux de communication traditionnels (téléphone et e-mail) limite la capacité des agences à interagir efficacement avec les clients. Cela peut entraîner des retards dans la réponse aux demandes et une insatisfaction client.
5. **Suivi des véhicules non systématique** : L'absence de documentation précise sur l'état des véhicules lors de leur retour complique le processus de récupération et peut engendrer des litiges liés à l'état des véhicules.
6. **Incapacité à générer des rapports** : Les méthodes manuelles ne permettent pas de générer facilement des rapports analytiques sur les réservations, les paiements ou les retours de véhicules, limitant ainsi la capacité des agences à évaluer leur performance.

En résumé, la critique de l’existant met en avant la nécessité d'une solution numérique pour remédier aux faiblesses identifiées et optimiser la gestion des réservations de véhicules.

3. Solution proposée

La solution proposée vise à développer une application web dédiée à la gestion des réservations de véhicules, permettant d'automatiser et d'optimiser les processus actuels. Cette application répondra aux besoins identifiés lors de l'étude de l'existant et permettra d'améliorer l'efficacité opérationnelle des agences de location. Les principales fonctionnalités incluent :

1. **Interface utilisateur intuitive** : Développement d'une interface conviviale pour faciliter la navigation et l'utilisation de l'application par les administrateurs et les clients.
2. **Gestion centralisée des réservations** : Permettre aux agences de gérer toutes les réservations depuis un tableau de bord centralisé, incluant la création, la modification et l'annulation de réservations, ainsi que la consultation des historiques de réservation.
3. **Automatisation des notifications** : Mettre en place un système de notifications automatiques pour informer les agences et les clients des réservations à venir, des retours de véhicules, et des paiements à effectuer.
4. **Suivi en temps réel des véhicules** : Intégrer une fonctionnalité pour enregistrer l'état des véhicules lors de leur retour, y compris le kilométrage et le niveau de carburant, afin de garantir une gestion transparente des actifs.

5. **Gestion des paiements** : Proposer des options de paiement en ligne sécurisées et la possibilité de gérer les transactions liées aux réservations, incluant les frais additionnels et les remboursements.
6. **Système de rapport et d'analyse** : Fournir des outils de reporting qui permettent aux agences de générer des rapports sur les réservations, les paiements et l'état des véhicules, afin d'évaluer leur performance et de prendre des décisions éclairées.
7. **Support client intégré** : Intégrer un système de tickets de support qui permet aux clients de soumettre des demandes d'assistance et de suivre l'état de leurs requêtes.

Cette solution web intégrée permettra d'améliorer considérablement l'efficacité et la transparence des opérations des agences de location de véhicules, tout en offrant une expérience utilisateur améliorée pour les clients.

Méthodologie adoptée

Le bon choix de la méthodologie conduit à la bonne réalisation du projet. Plusieurs méthodes de conception existent : l'approche traditionnelle en cascade, ainsi que différentes approches agiles, telles que Extreme Programming (XP), Scrum, et d'autres.

Pour la réalisation de notre application, nous avons adopté la méthodologie agile.

1. Méthodes agiles

Une méthode agile est une approche itérative et collaborative, capable de prendre en compte à la fois les besoins initiaux du client et ceux liés aux évolutions. La méthode agile repose sur un cycle de développement centré sur le client, impliquant celui-ci tout au long du projet. Cette implication permet à l'équipe d'obtenir un retour d'information régulier afin d'appliquer directement les changements nécessaires. Grâce à la méthode agile, le demandeur bénéficie d'une meilleure visibilité sur la gestion des travaux par rapport à une méthode classique. Cette méthode vise à accélérer le développement d'un logiciel tout en garantissant la réalisation d'un produit fonctionnel tout au long de son élaboration.

2. Méthode agile adoptée : Scrum

Pour mener à bien notre projet et garantir le bon déroulement des différentes phases, nous avons adopté Scrum comme méthodologie de gestion de projet.

Aujourd'hui, « Scrum » est la méthode agile la plus populaire. Ce terme signifie « mêlée » au rugby. La méthode Scrum s'appuie sur des « sprints », qui sont des périodes de temps relativement courtes, allant de quelques heures à un mois, mais généralement d'une durée de deux semaines. À la fin de chaque sprint, l'équipe présente ce qu'elle a ajouté au produit.

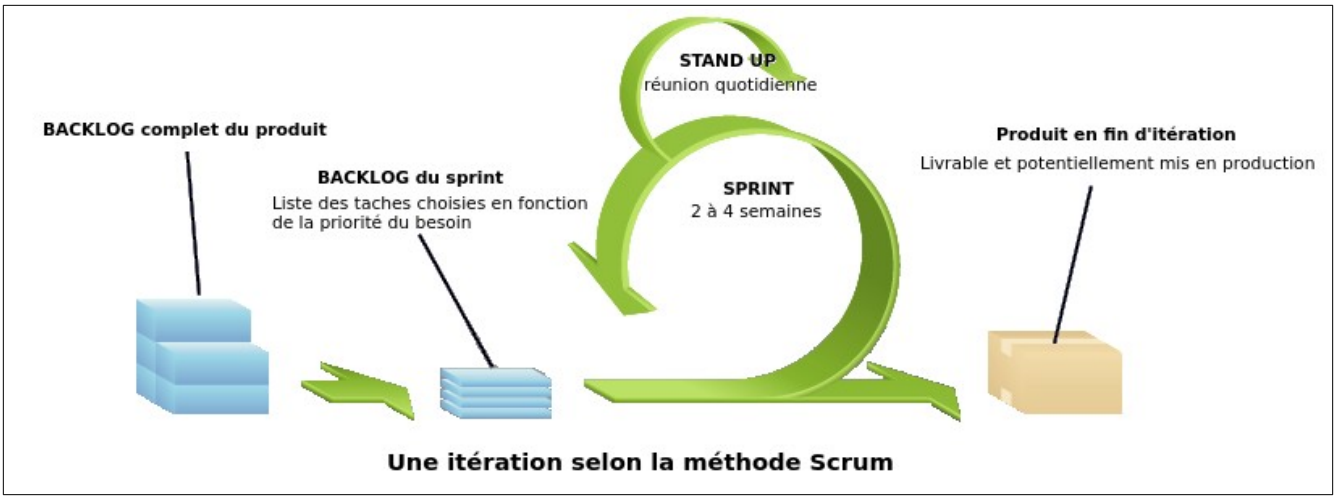

Figure 2: la méthode scrum

Scrum regroupe trois acteurs :

- **Product Owner** Le Product Owner est responsable de la définition et de la gestion du backlog produit. Il représente les intérêts des parties prenantes et des utilisateurs, en veillant à ce que l'équipe de développement se concentre sur les fonctionnalités les plus importantes. Le Product Owner priorise les éléments du backlog en fonction de la valeur ajoutée pour le client et des objectifs du projet.
- **Scrum Master** Le Scrum Master agit en tant que facilitateur pour l'équipe Scrum. Il aide à la mise en œuvre de la méthodologie Scrum et veille à ce que l'équipe respecte les pratiques agiles. Le Scrum Master est responsable de l'élimination des obstacles qui pourraient entraver le travail de l'équipe, tout en assurant la communication et la collaboration entre les membres de l'équipe et les parties prenantes.
- **Équipe de développement** L'équipe de développement est composée de professionnels qui travaillent ensemble pour livrer un produit fonctionnel à la fin de chaque sprint. L'équipe est auto-organisée et multidisciplinaire, ce qui signifie qu'elle possède toutes les compétences nécessaires pour réaliser le travail sans dépendre d'autres équipes. L'équipe s'engage à atteindre les objectifs fixés pour chaque sprint et à livrer un produit de haute qualité.

D’autres termes sont à connaître pour comprendre la méthode Scrum :

- **Le product backlog (carnet du produit)** : Ce document contient les exigences initiales dressées, puis hiérarchisées avec le client en début de projet. Néanmoins, il évolue tout au long de la durée du projet en fonction des divers besoins du client.
- **Le sprint backlog (carnet de sprint)** : Au début de chaque sprint, l’équipe définit un but. Lors de la réunion de planification du sprint, l’équipe de développement choisit les éléments du carnet à réaliser. L’ensemble de ces éléments constitue alors le sprint backlog.

- **User story** : Ce terme désigne les fonctionnalités décrites par le client, généralement formulées sous la forme d'une phrase simple qui exprime le besoin d'un utilisateur.
- **La mélée (scrum)** : C'est une réunion d'avancement organisée de manière quotidienne durant le sprint. Elle permet à l'équipe de se synchroniser et d'aborder les obstacles rencontrés.

Conclusion

Ce chapitre a été le point de départ du rapport pour la réalisation de notre projet. Il a permis d'identifier les enjeux et les opportunités liés à la gestion des réservations de véhicules. Le chapitre suivant sera consacré à la planification générale du projet, où nous détaillerons les étapes essentielles, les ressources nécessaires et les délais prévus pour mener à bien cette initiative.

Chapitre 2: Planification générale du projet

Introduction

Ce chapitre sera consacré à la réalisation de la première phase de la méthodologie Scrum, appelée « sprint 0 ». Cette phase présente les fonctionnalités de base du projet, le diagramme des cas d'utilisation global, ainsi que le *Product Backlog* qui servira de base pour la planification d'application. Nous y aborderons également l'analyse des besoins, étape essentielle pour définir les exigences du client et orienter les développements futurs. Ce travail préliminaire permettra de garantir que le projet « Cars Rental » réponde aux attentes tout en assurant une organisation efficace pour les sprints à venir.

Analyse des besoins

Cette section est destinée à la spécification des besoins fonctionnels et non fonctionnels de la solution ainsi qu'à l'identification des acteurs. L'objectif est de définir de manière claire et précise les attentes des utilisateurs finaux et les fonctionnalités nécessaires à l'application « Cars Rental ».

1. Identification des acteurs

- **Administrateurs** : Ils supervisent l'ensemble du système, gèrent les utilisateurs, les sociétés de location, les véhicules, et veillent au bon déroulement des réservations et des transactions financières.
- **Agences de location** : Elles administrent leurs propres véhicules, gèrent les réservations, les paiements et assurent le suivi des retours de véhicules ainsi que des éventuelles réparations.
- **Développeurs** : Responsables de l'évolution technique de l'application, assurant la maintenance, la mise à jour des fonctionnalités, et l'intégration des besoins des utilisateurs.

2. Besoins fonctionnels

- **Gestion des utilisateurs** : Création et gestion des comptes administrateurs et des agences de location.
- **Gestion des véhicules** : Ajout, modification, suppression de véhicules et mise à jour des informations associées.
- **Réservation de véhicules** : Création, gestion et annulation des réservations avec sélection des dates de prise en charge et de retour.
- **Paiements** : Suivi des transactions liées aux réservations, avec gestion des montants totaux, des acomptes et des paiements complets.
- **Suivi des véhicules retournés** : Gestion de l'état des véhicules à la restitution, avec suivi du kilométrage et du niveau de carburant.

- **Notifications** : Envoi de rappels et de notifications aux agences pour les réservations terminées ou nécessitant une intervention.

3. Besoins non fonctionnels

- **Sécurité** : Mise en place d’un système de gestion des accès sécurisé pour protéger les informations des utilisateurs et des transactions.
- **Performance** : L'application doit être capable de gérer un grand nombre de réservations et de véhicules sans altérer les temps de réponse.
- **Facilité d’utilisation** : Interface intuitive pour permettre aux utilisateurs de gérer leurs tâches sans complexité.

4. Langage de modélisation adopté

Pour modéliser les différents aspects de notre projet, nous avons opté pour l'utilisation du langage UML (Unified Modeling Language). UML est un langage de modélisation standard qui permet de représenter visuellement les composants d'un système d'information, et de mieux comprendre les relations entre eux. Il est largement utilisé dans la conception de logiciels et offre une approche visuelle permettant de simplifier la complexité des systèmes.

Figure 3: Logo UML

5. Architecture logicielle

Dans cette partie, nous expliquons le choix de l'architecture logicielle de notre application, qui repose sur le modèle **MVC (Modèle - Vue - Contrôleur)**. Ce modèle est l'une des architectures les plus couramment utilisées pour les applications web en raison de sa modularité, de sa maintenabilité et de sa simplicité d'utilisation.

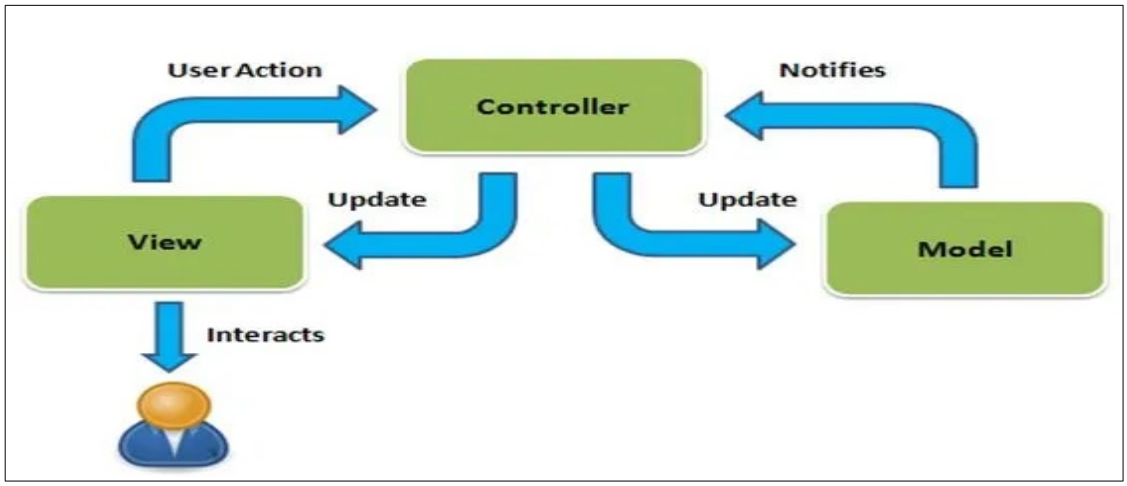

Figure 4: Architecture MVC

L'architecture MVC se compose de trois modules principaux :

- Le **modèle** représente les données de l'application ainsi que la logique métier qui leur est associée. C'est lui qui interagit avec la base de données pour créer, lire, mettre à jour et supprimer les informations.
- La **vue** correspond à la présentation des informations aux utilisateurs. Elle génère les interfaces visuelles basées sur les données reçues du modèle via le contrôleur.
- Le **contrôleur** agit comme un intermédiaire entre le modèle et la vue. Il reçoit les requêtes des utilisateurs (comme une réservation ou une modification de profil), traite la logique métier, et retourne une vue avec les données nécessaires.

Diagramme des cas d'utilisation global

Pour présenter les fonctionnalités de notre système de manière formelle, nous utilisons le diagramme de cas d'utilisation du langage de modélisation UML. Les diagrammes de cas d'utilisation sont des outils UML qui offrent une vision globale du comportement fonctionnel d'un système logiciel. Ils sont particulièrement utiles lors de présentations auprès de la direction ou des parties prenantes d'un projet, bien que pour le développement, des cas d'utilisation plus détaillés soient souvent plus appropriés. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et le système.

La **Figure 5** illustre le diagramme du cas d'utilisation général de notre application « Cars Rental ». Ce diagramme met en évidence les différents acteurs ainsi que les cas d'utilisation qui leur sont attribués.

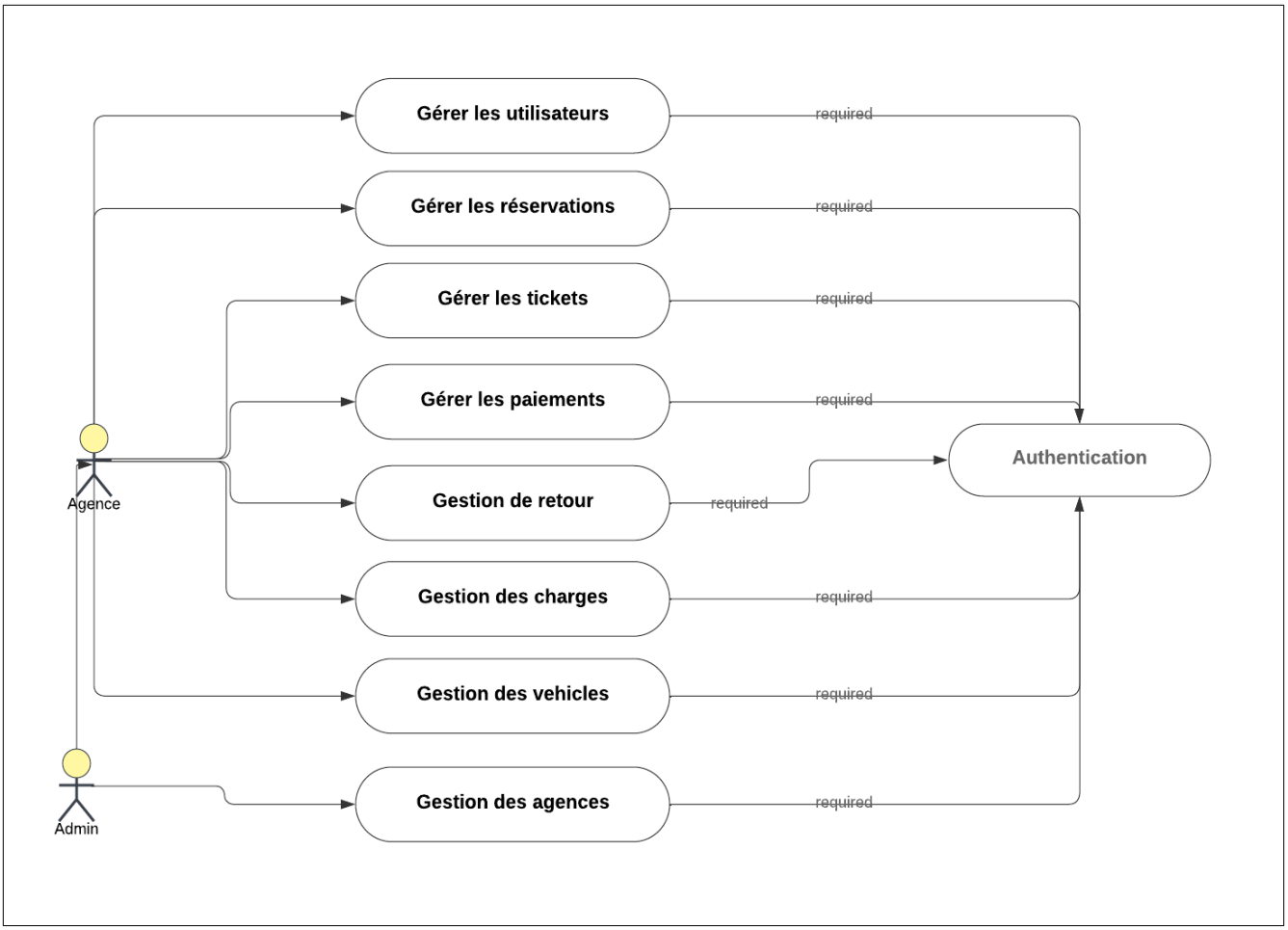

Figure 5: Diagramme cas d'utilisation global

Prototypage des interfaces de l'application web

Nous avons créé des maquettes pour les interfaces de l'application web afin d'obtenir une vision graphique et ergonomique dès le début du projet, en utilisant l'outil de prototypage « Figma ». Ces maquettes servent à illustrer le design et la disposition des éléments, facilitant ainsi la compréhension de l'interface par les utilisateurs et les parties prenantes.

Figure 6: Logo d'outil Figma

Voici quelques maquettes prévisionnelles des interfaces de l’application :

Adaptation du Cycle de Développement Scrum au Projet

1. Répartition des Rôles

Dans le cadre de l'adoption de la méthodologie Scrum pour le projet « Cars Rental », la répartition des rôles est essentielle pour assurer une collaboration efficace et une gestion optimale du projet. Voici les principaux rôles définis au sein de l’équipe :

- **Product Owner** : Foxdigia
- **Scrum Master**: SAJI Ismail
- **Équipe de Développemen**t : QOREICHI Ahmed

2. Product Backlog

Le tableau ci-dessous représente le backlog produit. Il présente les fonctionnalités de notre application. Dans ce tableau, les fonctionnalités sont classées selon des critères de classification :

Priorité:

- **Très Élevée**: Essentielle pour le fonctionnement de l'application.
- **Élevée**: Importante pour les utilisateurs, mais pas critique.
- **Moyenne**: Utile, mais peut être abordée plus tard.
- **Faible**: Peu de valeur immédiate, peut être développée en dernier.

Effort:

- L’effort est estimé en jours/homme et doit prendre en compte la complexité, le temps de développement et les tests nécessaires pour chaque fonctionnalité.

Fonctionnalité	Priorité	Effort (jours/homme)	Description
Gestion des utilisateurs	Élevée	5	Création et gestion des comptes pour les administrateurs et les agences de location.
Administration des agences de location	Élevée	4	Ajout et gestion des informations des agences, activation/désactivation.
Gestion du parc automobile	Élevée	6	Ajout, modification et suppression de véhicules avec détails (marque, modèle, etc.).
Système de réservation	Très Élevée	8	Création, modification et annulation de réservations par les agences.
Traitement des paiements	Élevée	5	Intégration d'une passerelle de paiement et enregistrement des paiements.

Suivi des récupérations de véhicules	Moyenne	4	Enregistrement de l'état des véhicules lors de leur retour.
Gestion des commentaires et tickets de support	Moyenne	3	Fonctionnalité permettant aux utilisateurs de laisser des commentaires et de soumettre des tickets de support.
Notifications et rappels	Moyenne	4	Système de notifications pour alerter des réservations à venir et des véhicules retournés.
Rapports et analyses	Faible	5	Génération de rapports sur les réservations et les paiements.
Interface utilisateur	Élevée	7	Conception d'une interface utilisateur intuitive et responsive.

Planification des sprints du projet

La planification des sprints du projet vise à organiser le calendrier de travail et à identifier le backlog de chaque sprint. Ce processus est essentiel pour assurer une progression fluide et efficace dans le développement de l'application. Le graphique suivant illustre la planification des sprints, mettant en évidence les différentes étapes et les objectifs associés à chaque sprint.

Sprint	Durée	Objectifs	Fonctionnalités
1	2 semaines	Mise en place de l'infrastructure de base	- Création de la base de données - Configuration de l'environnement de développement
2	2 semaines	Gestion des utilisateurs et des entreprises	- Création des comptes admin - Gestion des informations des agences
3	2 semaines	Gestion du parc automobile	- Ajout, modification et suppression de véhicules - Gestion des détails des véhicules
4	2 semaines	Réservation et gestion des paiements	- Fonctionnalités de réservation - Traitement des paiements
5	2 semaines	Gestion des retours de véhicules et des états	- Suivi de l'état des véhicules lors du retour - Gestion des kilométrages et du carburant
6	2 semaines	Notifications et rapports	- Système de notifications pour rappels - Génération de rapports sur les réservations et paiements

7	2 semaines	Tests et corrections de bugs	- Tests fonctionnels et non fonctionnels - Corrections des bugs identifiés

Conclusion

Au cours de ce chapitre, nous avons identifié les besoins fonctionnels et non fonctionnels de notre application. Nous avons également créé des maquettes prévisionnelles, élaboré le backlog produit et planifié la répartition des sprints à réaliser. Dans le chapitre suivant, nous allons passer à la phase de développement de l'application.

Chapitre 3 : Réalisation du projet

Introduction

Ce chapitre est consacré à la mise en œuvre du projet de développement de l'application **Cars Rental**. Après avoir identifié les besoins et planifié le projet dans les phases précédentes, nous passons maintenant à la réalisation pratique du système. Nous suivrons les étapes de la méthodologie Scrum, abordant les aspects techniques et les différentes fonctionnalités planifiées pour chaque sprint.

Spécification Fonctionnelle

La spécification fonctionnelle décrit en détail les fonctionnalités attendues de l'application web **Cars Rental**. Chaque fonctionnalité est associée à des exigences spécifiques, des entrées, des sorties, et des règles métier.

1. Gestion des Utilisateurs

- **Description** : Permettre aux agences et aux administrateurs de créer, gérer, et modifier leurs comptes.
- **Entrées** : Formulaire d'inscription, données de connexion, informations de profil.
- **Sorties** : Confirmation de création de compte, connexion réussie, affichage du profil.
- **Règles Métier** :
    - Un compte administrateur a accès à la gestion complète de l'application.
    - Un compte agence peut gérer uniquement les véhicules et les réservations liées à son entreprise.

2. Gestion des Véhicules

- **Description** : Gestion du parc automobile, ajout, modification, et suppression de véhicules.
- **Entrées** : Informations sur les véhicules (marque, modèle, année, état, prix journalier, etc.).
- **Sorties** : Liste des véhicules, confirmation d'ajout/modification/suppression.
- **Règles Métier** :
    - Les véhicules doivent être classés par disponibilité.
    - Un véhicule ne peut pas être supprimé s'il est lié à une réservation active.

3. Réservation de Véhicules

- **Description** : Les agences peuvent gérer les réservations de leurs clients en sélectionnant les dates de prise en charge et de retour du véhicule.
- **Entrées** : Dates de réservation, choix du véhicule, informations du client.
- **Sorties** : Confirmation de réservation, mise à jour des disponibilités.
- **Règles Métier** :

    - Un véhicule réservé doit être marqué comme indisponible durant la période sélectionnée.
    - Des règles de validation sont appliquées pour éviter les chevauchements de réservations.

4. Paiements

- **Description** : Gestion des paiements pour chaque réservation effectuée.
- **Entrées** : Montant total, méthodes de paiement.
- **Sorties** : Confirmation du paiement, récapitulatif des paiements.
- **Règles Métier** :
    - Le paiement doit être confirmé avant la validation de la réservation.
    - Un système de remboursement doit être prévu en cas d'annulation.

5. Suivi des Retours de Véhicules

- **Description** : Enregistrement de l'état des véhicules lors de leur retour (kilométrage, état général, niveau de carburant).
- **Entrées** : Données sur l'état du véhicule au retour.
- **Sorties** : Confirmation du retour, mise à jour de l'état du véhicule.
- **Règles Métier** :
    - Un rapport doit être généré pour tout dommage ou anomalie détectée au retour.

6. Notifications

- **Description** : Envoi de rappels et notifications aux agences concernant les réservations terminées ou en cours.
- **Entrées** : Dates des événements à notifier.
- **Sorties** : Notifications par email ou SMS.
- **Règles Métier** :
    - Les notifications doivent être envoyées à l'agence un jour avant la fin de la réservation.

7. Rapports et Analyses

- **Description** : Générer des rapports sur les réservations et les paiements.
- **Entrées** : Période de génération du rapport.
- **Sorties** : Rapports sur les réservations et revenus.
- **Règles Métier** :
    - Les rapports peuvent être filtrés par agence, période, ou véhicule.

8. Gestion des Charges Additionnelles

- **Description** : Ajouter des frais supplémentaires (dommages, carburant, etc.) à une réservation.
- **Entrées** : Type de charge additionnelle, montant.
- **Sorties** : Mise à jour du montant total de la réservation.
- **Règles Métier** :

- Toute charge additionnelle doit être justifiée par un événement documenté (ex. : inspection du véhicule).

9. Gestion des Tickets de Support

- **Description** : Permettre aux agences de soumettre des tickets de support pour signaler des problèmes ou demander de l'aide.
- **Entrées** : Détails du problème, description.
- **Sorties** : Confirmation de la création du ticket, suivi de la résolution.
- **Règles Métier** :
  - Chaque ticket doit être assigné à un administrateur pour suivi.

Diagrammes des classes

Les diagrammes des classes permettent de définir et de modéliser les différentes entités du système ainsi que les relations entre elles. Pour notre application Cars Rental, nous avons identifié plusieurs classes principales telles que Reservation, Vehicle, Agency, Admin, et Payment. Ces classes interagissent entre elles pour assurer la gestion complète du processus de réservation et de location des véhicules.

Le diagramme des classes ci-dessous montre les attributs et les méthodes de chaque classe ainsi que les associations, dépendances, et héritages entre les différentes entités du système.

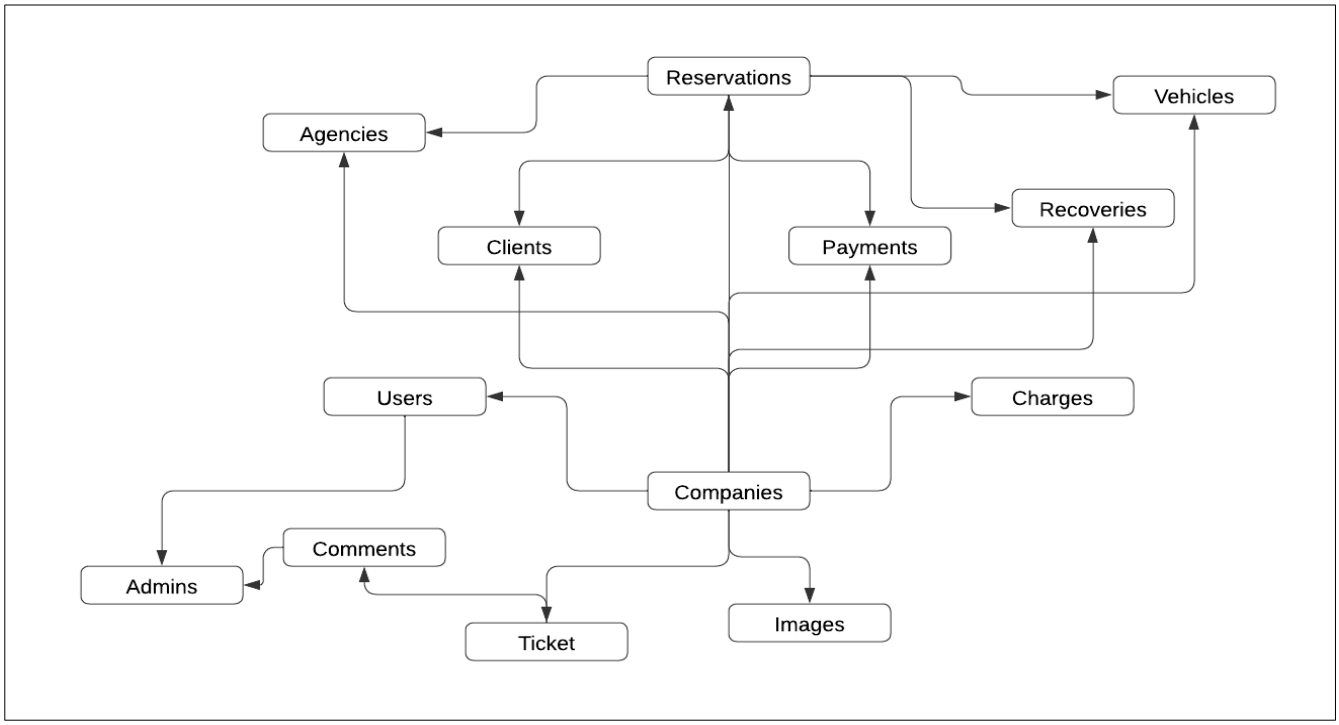

Figure 7: Diagrammes des classes

Admins (Administrateurs)

- Un **administrateur** peut recevoir plusieurs **commentaires**.
- Relation: 1 Administrateur --- * Commentaires
- Détails: Un administrateur peut superviser plusieurs tickets, et les utilisateurs peuvent laisser des commentaires sur leur interaction avec un administrateur.

Agencies (Agences)

- Une **agence** peut avoir plusieurs **réservations**.
- Relation: 1 Agence --- * Réservations
- Détails: Une agence gère les réservations pour les véhicules dont elle a la charge. Chaque réservation est associée à une agence spécifique.

Clients

- Un **client** peut faire plusieurs **réservations**.
- Relation: 1 Client --- * Réservations
- Détails: Un client peut effectuer plusieurs réservations de véhicules au fil du temps. Chaque réservation est liée à un client unique.

Companies (Sociétés)

- Une **société** peut avoir plusieurs **clients**, **réservations** et **paiements**.
- Relation: 1 Société --- * Clients, 1 Société --- * Réservations, 1 Société --- * Paiements
- Détails: Une société est responsable de la gestion des clients, des réservations de véhicules, et des paiements. Elle est donc reliée à ces entités.

Images (Images)

- Les **images** peuvent être associées de manière polymorphique à plusieurs entités (ex: sociétés, véhicules, etc.).
- Détails: Les images peuvent représenter des logos de sociétés, des photos de véhicules, etc. Le champ target_type permet de définir le type d'entité (Société, Véhicule, etc.) auquel l’image est associée.
- Relation polymorphique: Images --- Entités polymorphiques (Sociétés, Véhicules, etc.)

Reservations (Réservations)

- Une **réservation** est faite pour un **véhicule** spécifique.
- Relation: 1 Réservation --- 1 Véhicule
- Détails: Chaque réservation concerne un véhicule précis, géré par une société ou une agence. Un véhicule ne peut avoir qu’une réservation active à un moment donné.

Tickets

- Un **ticket** peut avoir plusieurs **commentaires**.
- Relation: 1 Ticket --- * Commentaires

- Détails: Un ticket est une demande ou un incident signalé par un client, un administrateur, ou une agence. Plusieurs commentaires peuvent être associés à un même ticket pour fournir des détails supplémentaires ou des mises à jour.

1. Exemple de relations détaillées en UML

- **Admins** (Administrateurs): chaque administrateur peut recevoir plusieurs **Commentaires** d'utilisateurs ou de tickets auxquels il a répondu.
- **Agences**: chaque agence peut avoir de multiples **Réservations**, indiquant qu'elle gère les véhicules réservés pour ses clients.
- **Clients**: chaque client peut faire plusieurs **Réservations**, représentant les différents véhicules qu'il a loués au fil du temps.
- **Sociétés** (Companies): chaque société a une responsabilité directe sur plusieurs **Clients**, **Réservations**, et **Paiements**.
- **Images**: chaque image est associée de manière polymorphique à une entité spécifique (par exemple, un véhicule ou une société).
- **Réservations**: chaque réservation concerne un **véhicule** particulier qui est loué par un client pour une période définie.
- **Tickets**: chaque ticket peut inclure plusieurs **Commentaires** pour documenter les interactions entre les utilisateurs et le personnel de support.

Ce modèle détaillé donne une vision claire des relations entre les entités dans le projet "Cars Rental".

Codage

Dans cette partie, nous allons présenter le schéma de la base de données ainsi que le codage des principales fonctionnalités implémentées lors du développement de notre projet Cars Rental. Nous allons détailler les méthodes, contrôleurs, et vues implémentées au cours des différents sprints.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	company	bigint(20)		UNSIGNED	Yes	NULL		
3	name	varchar(255)	utf8mb4_unicode_ci		No	None		
4	email	varchar(255)	utf8mb4_unicode_ci		No	None		
5	phone	varchar(255)	utf8mb4_unicode_ci		No	None		
6	secondary_phone	varchar(255)	utf8mb4_unicode_ci		Yes	NULL		
7	city	enum('casablanca', 'rabat', 'fez', 'marrakesh', 'a...	utf8mb4_unicode_ci		No	None		
8	zipcode	varchar(255)	utf8mb4_unicode_ci		No	None		
9	address	varchar(255)	utf8mb4_unicode_ci		No	None		
10	created_at	timestamp			Yes	NULL		
11	updated_at	timestamp			Yes	NULL		

Agneciess table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	company	bigint(20)		UNSIGNED	Yes	NULL		
3	vehicle	bigint(20)		UNSIGNED	Yes	NULL		
4	agency	bigint(20)		UNSIGNED	Yes	NULL		
5	client	bigint(20)		UNSIGNED	Yes	NULL		
6	secondary_client	bigint(20)		UNSIGNED	Yes	NULL		
7	reference	varchar(255)	utf8mb4_unicode_ci		No	None		
8	pickup_date	datetime			No	None		
9	pickup_location	varchar(255)	utf8mb4_unicode_ci		Yes	NULL		
10	dropoff_date	datetime			No	None		
11	dropoff_location	varchar(255)	utf8mb4_unicode_ci		Yes	NULL		
12	mileage	int(11)			No	None		
13	fuel_level	double(15,5)			No	None		
14	condition	longtext	utf8mb4_bin		No	None		
15	rental_period_days	int(11)			No	None		
16	status	varchar(255)	utf8mb4_unicode_ci		No	None		
17	created_at	timestamp			Yes	NULL		
18	updated_at	timestamp			Yes	NULL		

Reservations table

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	company	bigint(20)		UNSIGNED	Yes	NULL		
3	registration_type	enum('vehicle', 'WW')	utf8mb4_unicode_ci		No	None		
4	registration_number	varchar(255)	utf8mb4_unicode_ci		No	None		
5	brand	varchar(255)	utf8mb4_unicode_ci		No	None		
6	model	varchar(255)	utf8mb4_unicode_ci		No	None		
7	year	int(11)			No	None		
8	daily_rate	decimal(15,5)			No	None		
9	passenger_capacity	int(11)			No	None		
10	mileage	int(11)			No	None		
11	number_of_doors	int(11)			No	None		
12	cargo_capacity	int(11)			No	None		
13	transmission_type	enum('manual', 'automatic')	utf8mb4_unicode_ci		No	None		
14	fuel_type	enum('gasoline', 'diesel', 'gasoline hybrid', 'die...	utf8mb4_unicode_ci		No	None		
15	registration_date	date			No	None		
16	horsepower	enum('less than 8 cv', 'between 8 and 10 cv', 'bet...	utf8mb4_unicode_ci		No	None		
17	horsepower_tax	decimal(15,5)			No	None		
18	insurance_company	enum('company 1', 'company 2')	utf8mb4_unicode_ci		No	None		
19	insurance_issued_at	date			No	None		
20	insurance_cost	decimal(15,5)			No	None		
21	created_at	timestamp			Yes	NULL		
	dated_at	timestamp			Yes	NULL		

Console

Vehicles table

Figure 8: Database schema

Interface

Voici quelques exemples d'écrans réalisés pour notre application, montrant les principales interfaces utilisateur développées. Ces interfaces illustrent la gestion des véhicules, la création des réservations, ainsi que l'administration du système par les agences et les administrateurs.

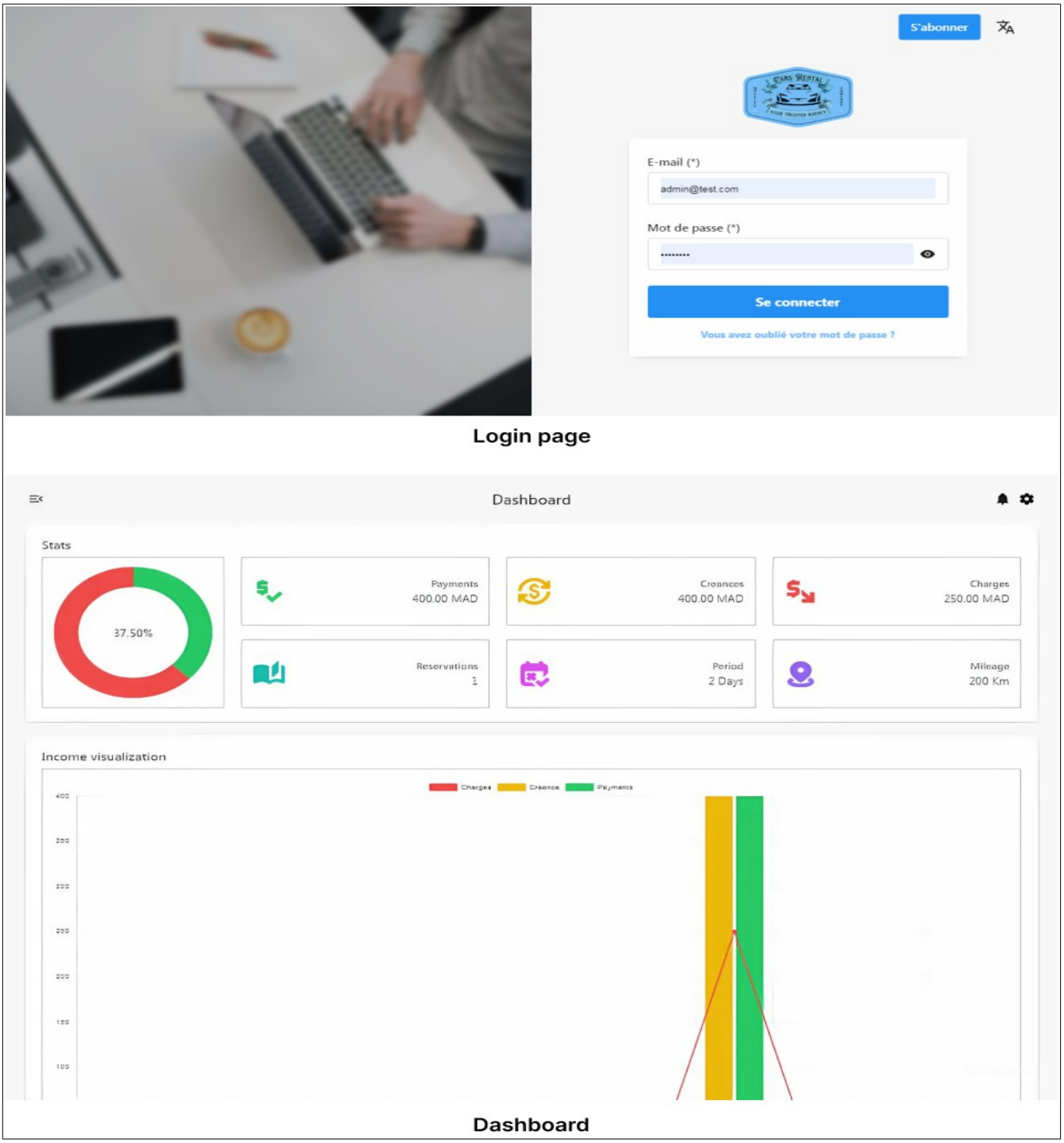

Figure 9: Interface

Conclusion

Ce chapitre a présenté la réalisation du projet **Cars Rental**, détaillant les fonctionnalités principales et le diagramme des classes. Nous avons également abordé les interfaces développées. Le prochain chapitre se concentrera sur la phase de clôture de l'application.

Chapitre 5 : Phase de cloture

Introduction

La phase de clôture est la dernière étape du projet, marquant la fin du développement et de la mise en œuvre de la plateforme Cars Rental. Elle se concentre sur l'évaluation des résultats obtenus et l'assurance que toutes les exigences du projet ont été satisfaites. Cette phase comprend également une analyse de l'environnement de travail, tant matériel que logiciel, qui a été utilisé tout au long du projet.

Nous y présentons également l'architecture opérationnelle de notre projet, en mettant en lumière les différents composants du système, leur interaction, et leur rôle dans le fonctionnement global de la plateforme. Cette récapitulation nous permettra de mieux comprendre les succès et les défis rencontrés, ainsi que de préparer le terrain pour les futures améliorations et itérations de la plateforme.

Environnement de travail

Au cours de la phase de développement de Cars Rental, plusieurs outils et technologies ont été utilisés pour assurer une mise en œuvre efficace.

a. Matériel

- Serveurs : Des serveurs dédiés ont été utilisés pour héberger l'application, garantissant une disponibilité et une performance optimales.
- Postes de travail : Les développeurs et designers ont travaillé sur des stations de travail équipées de ressources suffisantes pour exécuter des environnements de développement intégrés (IDE) et des outils de conception graphique.

b. Logiciel

- Système de gestion de base de données : MySQL a été choisi pour gérer les données de l'application, permettant un stockage sécurisé et une récupération rapide des informations.
- Framework : Laravel a été utilisé comme framework de développement pour sa robustesse, sa sécurité et ses fonctionnalités avancées facilitant le développement rapide d'applications web.
- Outils de gestion de projet : Des outils comme Trello ou Jira ont été employés pour suivre l'avancement des tâches et assurer une bonne communication au sein de l'équipe.

Architecture opérationnelle

L'architecture opérationnelle de Cars Rental repose sur une conception modulaire, permettant une flexibilité et une évolutivité dans le développement futur. Voici les principaux composants de l'architecture :

Frontend :

Technologies : Utilisation de HTML, CSS, et JavaScript pour construire une interface utilisateur réactive et intuitive.

Frameworks : Bootstrap a été intégré pour garantir que l'application soit responsive et accessible sur différents appareils.

Backend :
API REST : Le backend expose des services via une API REST, facilitant les échanges de données entre le client et le serveur.
Gestion des utilisateurs : Un système d'authentification et d'autorisation a été mis en place pour sécuriser l'accès aux différentes fonctionnalités de l'application.
Base de données :
La base de données est structurée en plusieurs tables interconnectées, incluant des entités telles que les utilisateurs, réservations, véhicules, et paiements.
Les relations entre les tables permettent de maintenir l'intégrité des données et de faciliter les opérations complexes.

Évaluation des résultats

À ce stade, il est essentiel d'évaluer les résultats obtenus par rapport aux objectifs initiaux. Cette évaluation inclut :

- Tests fonctionnels : Chaque fonctionnalité a été testée pour garantir qu'elle répond aux spécifications.
- Feedback des utilisateurs : Des retours ont été collectés auprès des utilisateurs pour identifier les points forts et les axes d'amélioration.
- Performance de l'application : Des analyses de performance ont été réalisées pour mesurer la rapidité et l'efficacité de l'application sous différentes charges.

Conclusion

La phase de clôture permet non seulement de récapituler les réalisations de Cars Rental, mais également d'identifier les leçons apprises qui guideront les futurs projets. Les résultats de l'évaluation aideront à prendre des décisions éclairées concernant les mises à jour et les nouvelles fonctionnalités à intégrer. En somme, cette phase est cruciale pour assurer la pérennité et le succès continu de la plateforme.

Conclusion générale

Notre projet de fin d'études a été réalisé dans le cadre de l'obtention du diplôme de Mastère professionnel, au sein de la société Foxdigia. Ce projet consistait à développer une application web pour la gestion des agences de location de voitures.

Tout au long de ce rapport, nous avons détaillé les différentes étapes que nous avons suivies pour construire cette application en adoptant la méthodologie Scrum. Nous avons commencé par une étude de l'existant, l'identification des objectifs du projet, et la proposition de notre solution. Par la suite, nous avons planifié le travail en identifiant les différents modules du projet tout en respectant les priorités des besoins.

Ensuite, nous avons mis en place des études spécifiques et conceptuelles pour chaque sprint. Nous avons terminé notre rapport par une description de l'environnement de travail, y compris le matériel et le logiciel utilisés, ainsi que les technologies et l'architecture de notre projet.

Enfin, durant la réalisation de notre projet, nous avons rencontré certaines limites, notamment une contrainte de temps qui a constitué un obstacle à l'ajout de certaines fonctionnalités supplémentaires. Cependant, ce projet a été une occasion précieuse pour mettre en pratique et déployer nos connaissances en informatique.

Dans le cadre de notre projet, plusieurs perspectives peuvent être envisagées pour son développement futur. Par exemple, l’intégration d’une application mobile pourrait améliorer l'accessibilité et l’expérience utilisateur, permettant ainsi aux agences de gérer les réservations de véhicules de manière plus fluide et intuitive depuis leurs appareils mobiles.