Dédicace

Je dédie ce travail :

*À ma chère mère **Henda** et à mon cher père **Ridha** qui n'ont jamais cessé de me supporter, me soutenir et m'encourager durant mes années d'études.*

Qu'ils trouvent ici le témoignage de ma profonde gratitude et reconnaissance.

*À mes soeurs **Achwak**, **Nawres** & **Farah** qui me donnent de l'amour et de la vivacité.*

*À ma merveilleuse grand-mère **Malika**, et à mes oncles, **Adel**, **Nizar, Anwar**, & **Faten**, fêtant mes compagnons d'aventure depuis le début.*

*À tous mes amis **Ahlem**, **Md.Ali**, **Refki**, **Aymen** & **Haythem** les gardiens de mes souvenirs les plus précieux.*

Vous êtes tous les piliers de ma vie, la source de mon bonheur et le réconfort de mon âme.

Merci !

Remerciements

En achevant mon parcours d'ingénieur en informatique en tant que stagiaire chez VIVIA CLINIAL NETWORK, je tiens à exprimer ma profonde gratitude envers toutes les personnes qui ont joué un rôle essentiel dans la réussite de cette expérience

MME. BEN JABRA SAOUSSEN mon encadrante académique pour m'avoir soutenu tout au long de stage et pour ses précieuses directives qu'elle m'a prodiguées tout au long de la rédaction de ce rapport. Je tiens à lui exprimer toute mon admiration et ma reconnaissance.

Mon superviseur professionnel **MME. FARJALLAH WISSAL** ainsi que mon tuteur **M. FAZAA MAJDEDDINE**, qui malgré leurs responsabilités et leurs occupations, ont toujours eu le temps pour m'écouter, conseiller et n'ont jamais renoncé à m'orienter vers les bonnes solutions avec un degré de patience et de professionnalisme sans égal. Pour leur présence et leur expertise qui m'ont permis d'approfondir mes connaissances et d'accomplir ce travail avec confiance.

Je tiens aussi à adresser mes plus sincères remerciements à l'ensemble du corps administratif et enseignant de l'École Pluridisciplinaire internationale de Sousse **(EPI),** pour avoir porté un vif intérêt à notre formation, et pour avoir accordé de l'attention et de l'énergie, et ce, dans un cadre agréable de respect.

Finalement, Je profite de cette occasion pour remercier **les membres du jury** pour l'honneur qu'ils me font de bien vouloir juger ce travail. Veuillez trouver dans ce rapport les qualités de clarté et de motivation qu'ils attendent.

Résumé

Ce projet a été mené dans le cadre du stage de fin d’étude en génie logiciel au sein de l'entreprise " VIVIA CLINICAL NETWORK" pour l'année universitaire 2023/2024.

Il vise à répondre à la problématique urbaine en créant une application mobile. Cette application a pour objectif d'améliorer la qualité de vie en simplifiant la recherche de stationnement, en encourageant le covoiturage et en facilitant l'accès aux transports publics grâce à l'utilisation de données en temps réel. Nos objectifs sont ainsi de réduire le stress et la pollution, tout en rendant la vie urbaine plus agréable et fluide pour tous les citadins.

Mots-clés: gestion urbaine, gestion du trafic, SCRUM, dévelopement mobile, Flutter, Clean Architecture, NodeJS.

Abstract

This project was carried out as part of the end-of-study internship in software engineering within the company "VIVIA CLINICAL NETWORK" for the 2023/2024 academic year.

It aims to respond to urban problems by creating a mobile application. This application aims to improve quality of life by simplifying the search for parking, encouraging carpooling and facilitating access to public transport through the use of real-time data. Our objectives are to reduce stress and pollution, while making urban life more pleasant and fluid for all city dwellers.

Keywords: urban management, traffic management, SCRUM, mobile development, Flutter, Clean Architecture, NodeJS.

Table des matières

Liste des Tableaux

Liste des Figures

Introduction Générale

Dans un monde de plus en plus urbanisé, la gestion efficace de l'espace, de la mobilité et des services est devenue essentielle pour garantir la qualité de vie dans les zones urbaines. Face à ces défis croissants, l'innovation technologique offre de nouvelles opportunités pour repenser la manière dont nous abordons l'urbanisme, le transport et la réservation dans nos villes.

C’est dans ce cadre que se dirige notre projet de fin d’études qui consiste à réaliser une application mobile qui sera nommée " Smart Way " et qui émerge comme une réponse à ces défis en offrant une approche intégrée pour améliorer la gestion urbaine. En fait, cette application polyvalente vise à simplifier la vie quotidienne des citadins en leur fournissant un accès facile à une gamme de services et d'informations pertinents.

En combinant des fonctionnalités innovantes pour l'urbanisme, le covoiturage, les réservations et la circulation, " Smart Way " cherchera à transformer la manière dont les individus interagissent avec leur environnement urbain. L'application facilitera la mise en relation des conducteurs et des passagers partageant un trajet similaire, contribuant ainsi à réduire les embouteillages et les émissions de carbone. De plus, elle fournira des alertes en temps réel sur les travaux routiers et les événements perturbateurs, aidant ainsi les utilisateurs à planifier leurs déplacements de manière efficace.

Ce rapport sera consacré à la présentation des différentes étapes de réalisation de projet et sera organisé en quatre chapitres principaux.

Dans le premier chapitre, nous présentons l’étude préalable où nous allons commencer par la présentation de l’entreprise d’accueil suivie d’une étude des applications existantes répondant à notre besoin. Enfin, nous terminons le chapitre par la présentation de la méthodologie de développement qui sera adoptée.

Dans le deuxième chapitre nous nous intéressons à la spécification des besoins en identifiant les acteurs de notre solution ainsi que les fonctionnalités qui leurs sont

offertes. Ces fonctionnalités seront par la suite modélisées avec des diagrammes de cas d'utilisations.

Le troisième chapitre présente l'étude conceptuelle de notre application et la description des principes de la solution à l'aide du langage UML.

Dans le dernier chapitre nous allons exposer l'environnement de travail adopté ainsi que les interfaces réalisées au cours de ce projet.

Enfin, nous clôturons le rapport par une conclusion générale et certaines perspectives potentielles du travail réalisé.

Chapitre 1 : Etude préalable

I. Introduction

Ce premier chapitre est consacré à l'étude préliminaire de notre projet qui est considérée comme une étape cruciale pour sa réalisation. Nous commencerons alors par présenter l'entreprise d'accueil, puis nous situerons le projet dans son contexte en identifiant sa problématique et ses objectifs. Ensuite nous passons à l'analyse de l'existant qui nous permettra de mieux mieux fixer nos objectifs en comparant les solutions actuelles. Enfin, nous aborderons la méthodologie de travail qui sera adopté pour notre projet.

II. Présentation de l'entreprise d'accueil

Ce projet a été réaliser au sein de **VIVIA Clinical Network** (VCN) [1] qui est une startup spécialisée dans les solutions logicielles innovantes, offrant des services de développement de logiciels et de consultation pour les entreprises en pleine transformation numérique.

Figure 1 : Logo Vivia

Avec une approche centrée sur le client et une expertise approfondie des réglementations et des technologies, VIVIA (figure 1) guide ses clients vers le succès dans un environnement professionnel dynamique et évolutif [2].

Informations techniques

VIVIA Clinical Network est basée à Technopôle, SoftTech de Sousse, 4000, Tunisie. Elle offre une gamme diversifiée de services comprenant le développement Web et Mobile, l'amélioration du référencement SEO, la Business Intelligence, l'Intelligence Artificielle ainsi que l'Assistance Technique.

III. Contexte général du projet

Dans cette section, nous allons présenter le contexte général de notre projet en définissant notre problématique et en fixant nos objectifs.

III.1 Problématique

La recherche de stationnement, les embouteillages et les retards des transports publics sont des problèmes majeurs qui impactent négativement la qualité de vie des citadins. En effet, ces problèmes augmentant le stress de la vie quotidienne des gens ainsi que la pollution des villes. Ces défis persistent en raison du manque de solutions intégrées et efficaces pour gérer la mobilité urbaine. Nous nous interrogeons sur la manière de développer une approche innovante et technologiquement avancée pour résoudre ces problèmes et améliorer la gestion du stationnement, promouvoir le covoiturage et simplifier les transports publics dans nos villes.

III.2 Objectifs

Dans le cadre de notre projet de fin d'études, nous chercherons à répondre à la problématique posée. En effet, nos objectifs consistent à développer une application mobile visant à améliorer la qualité de vie en ville. Cette application devra faciliter la recherche de stationnement, promouvoir le covoiturage et rendre les transports publics plus accessibles en se basant sur la manipulation d'informations en temps réel. Les objectifs de l'application à développer sont alors réduire le stress ainsi que la pollution et rendre la vie urbaine plus fluide pour tous.

IV. Etude de l'existant

L'analyse du marché a mis en évidence une variété d'applications dédiées à la gestion des rues et à la régulation du trafic. À titre illustratif, nous avons examiné trois applications existantes :

« Park Easy », « Split » et « Transport Urbain Bruxelles »

IV.1 Park Easy

Park Easy [3] est une application conçue pour faciliter le stationnement en permettant aux utilisateurs de réserver des places de parking à l'avance. Elle offre également des fonctionnalités de recharge pour véhicules électriques (VE). Figure 2 illustre l'interface d'accueil de cette application.

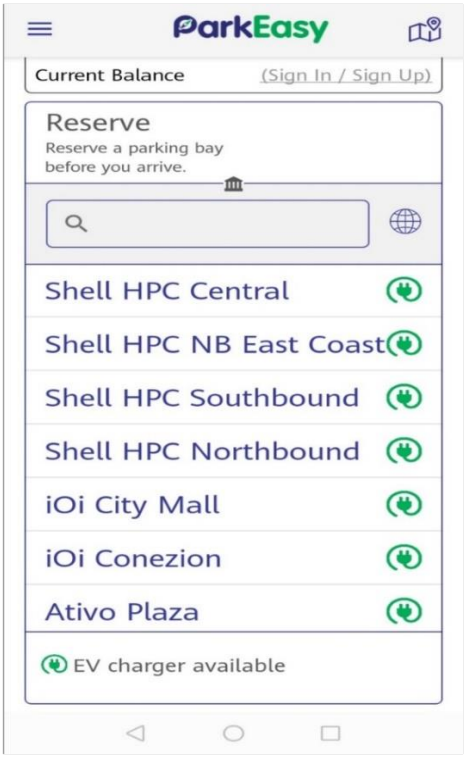


Figure 2 : Interface « Park Easy »

IV.2 Split

Split [4] est une application de covoiturage qui connecte les conducteurs et les passagers pour partager des trajets en voiture. Elle est utile soit pour les conducteurs ayant des places vides dans leurs voitures ou pour les passagers à la recherche d'un trajet. La figure 3 illustre l'interface d'accueil de l'application.

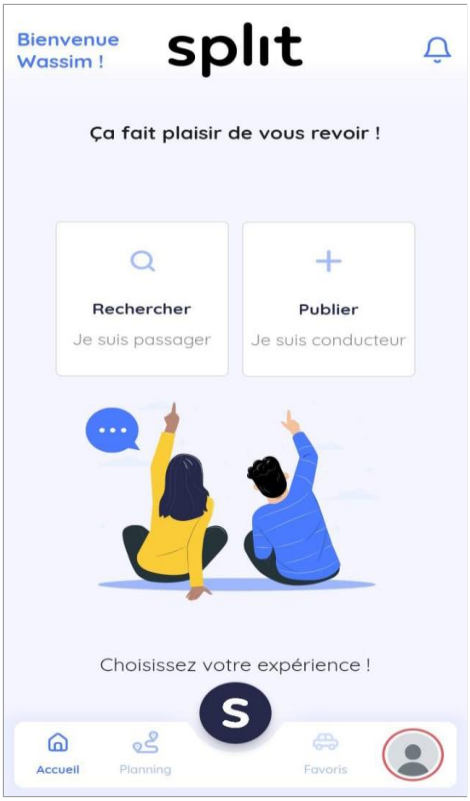


Figure 3: Interface « Split »

IV.3 Transport urbain Bruxelles

Transport urbain Bruxelles [5] est une application centralisant l'infrastructure de transport de Bruxelles, y compris les lignes de métro, de tramway et de bus. Elle est utilisée par les résidents locaux ainsi que les voyageurs occasionnels pour faciliter leurs déplacements quotidiens. La figure 4 illustre l'interface d'accueil de l'application.

Figure 4: Interface de « Transport urbain Bruxelles »

V. Critique de l'existant

La critique de l'existant met en évidence les principaux dysfonctionnements du système actuel. Les applications existantes présentent les inconvénients suivants :

V.1 Park Easy

- Interface mal conçue : Difficulté de navigation, instructions peu claires et bugs fréquents rendent l'application frustrante. Par exemple, certains utilisateurs ont signalé que l'application est boguée et nécessite parfois une réinstallation pour fonctionner correctement.
- Limitation aux services de stationnement : Park Easy se concentre uniquement sur les réservations de stationnement ce qui la rend moins polyvalente que les applications offrant des services de transport variés tels que le covoiturage, les informations sur les transports publics ou les mises à jour du trafic en temps réel

V.2 Split

- Manque d'options de communication : Peut entraîner des malentendus entre les utilisateurs.
- Limitations des fonctionnalités : elle repose uniquement sur le covoiturage.

V.3 Transport urbain Bruxelles

- Manque de clarté dans les fonctionnalités : Les fonctionnalités avancées peuvent ne pas être clairement expliquées dans l'application, ce qui rend difficile pour les utilisateurs de comprendre comment les utiliser efficacement.
- Manque de mises à jour en temps réel : Les horaires, les retards ou les perturbations du service, ainsi que les données de localisation, peuvent ne pas être mis à jour en temps réel, entraînant des informations incorrectes ou obsolètes pour les utilisateurs.

V.4 Etude comparative

Afin de bien dégager les inconvénients et les avantages des applications existantes nous avons élaborer une étude comparative entre elles. Le tableau 1 illustre cette comparaison en se basant sur différents critères que nous envisageons importantes selon les besoins de notre projet

Solutions / Critères	Park Easy	Split	Transport urbain Bruxelles
C1 : Notification et Alertes	Non	Non	Non
C2 : Réservations de Parkings	Oui	Non	Non
C3 : Gestion du Covoiturage	Non	Oui	Non
C4 : Feedback et Suggestions	Non	Non	Non
C5 : Suivi de infos en temps réels (incidents, retards...)	Non	Non	Oui
C6 : Transports Publics	Non	Non	Oui

Tableau 1 : Tableau comparatif de la solutions existantes

VI. Solution proposée

Afin de profiter des avantages des applications existantes, et d'éviter leurs inconvénients, nous proposons dans ce projet de développer une application mobile de gestion de trafic urbain qui devra offrir différents modules dont les plus importants sont :

- Le module de covoiturage : Permet aux utilisateurs de partager des trajets en voiture avec d'autres personnes se rendant dans la même direction, réduisant ainsi les coûts de transport et les émissions de CO2.
- Le Module de réservations de parkings : Facilite la recherche et la réservation de places de parking disponibles à proximité, optimisant ainsi le temps et les efforts des utilisateurs pour trouver un stationnement.
- Le module de transports publics : Fournit des informations en temps réel sur les horaires, les perturbations et les itinéraires des métros, trams, et bus, aidant les utilisateurs à planifier leurs trajets de manière plus efficace.

VII. Méthodologie de travail

Une gestion de projet [6] efficace implique l'utilisation d'une méthodologie appropriée pour garantir le bon déroulement du travail jusqu'à sa réalisation, en tenant compte des besoins du client et des changements éventuels. Pour répondre à ces exigences, l'équipe a opté pour une approche agile [7], en particulier SCRUM [8].

VII.1 Méthodologie agile

Pour répondre aux exigences de notre projet, nous avons opté pour un processus de développement agile qui est plus efficace et moins rigide que les méthodes classiques. Les méthodes agiles [9] offrent une plus grande flexibilité et une meilleure visibilité dans la gestion du projet, ce qui permet de générer un produit de haute qualité. De plus, elles permettent à l'équipe de parcourir les risques beaucoup plus rapidement qu'avec les outils de gestion de projet traditionnels

VII.2 Méthode Scrum

SCRUM illustrée par la figure 5, est une méthode agile qui est définie par un ensemble de pratiques qui doivent être inclues dans la phase de développement. SCRUM est subdivisée en des blocs temporaires appelés Sprint, qui durent généralement de deux à quatre semaines. Chaque Sprint est une entité à part et est planifié collectivement par tous les membres de l'équipe. Durant les Sprints, une réunion quotidienne se déroule pendant quinze minutes où chaque membre discute les tâches qu'il a réalisé, les différents obstacles qu'il a rencontrés et son futur plan pour les vingt-quatre heures à venir. [10].

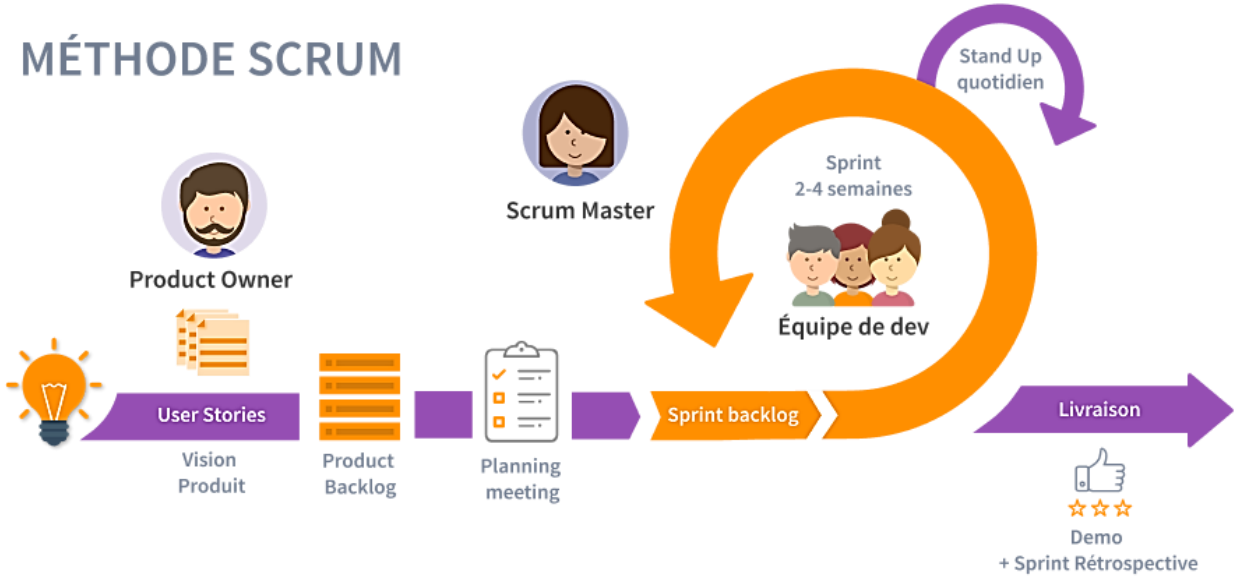


Figure 5: Exemple de cycle de développement Agile

Scrum définit 3 rôles qui sont :

- **Product Owner** : Le Product Owner est le représentant du client au sein de l'équipe projet. Il a pour responsabilité de définir et de hiérarchiser les fonctionnalités du produit, ainsi que de décider de la date et du contenu de chaque sprint, en se basant sur les estimations de charge fournies par l'équipe.

- **Scrum Master** : Son rôle est de veiller à ce que l'équipe Scrum suive correctement la méthodologie Scrum, d'éliminer les obstacles qui entravent le travail de l'équipe, et de faciliter la communication et la collaboration au sein de l'équipe pour atteindre les

objectifs du projet. Le Scrum Master agit comme un mentor et un défenseur de la méthodologie Scrum pour assurer son efficacité.

- Équipe de développement : L'équipe Scrum englobe tous les rôles traditionnellement requis dans un projet, y compris les concepteurs, les développeurs, les testeurs, et autres. Cette équipe est autoorganisée et demeure stable tout au long de la durée d'un sprint

VIII. Diagramme de GANTT

Le diagramme de Gantt [11] est un outil couramment employé en ordonnancement et gestion de projet pour afficher chronologiquement les différentes phases par lesquelles passent un projet. Il offre une représentation visuelle de sa progression. Pour notre cas, le diagramme de GANTT illustré par la figure 6 est composé de 7 phases incluant la phase préliminaire du projet, 4 sprints ainsi que la rédaction du rapport.

	1 - 25 Fév	26 Fév - 09 Mar	10 - 24 Mar	25 Mar - 08 Avr	09 - 23 Avr	24 Avr -08 Mai	09 - 23 mai	24 Mai - 07 Juin
Analyse et spécification du besoin								
Choix de d'architecture & Collecte de données								
Sprint 1 : Développemnt du module d'authentifcation & d'inscription								
Sprint 2 : Développemnt du module réservation								
Sprint 3 : Développemnt du module Transports publlics								
Sprint 4 : Développemnt du module Covoiturage								
Rédaction du rapport								

Figure 6 : Diagramme de GANTT

IX. Conclusion

Dans ce premier chapitre, nous avons commencé par la présentation de l'organisme d'accueil pour notre projet de fin d'études. Par la suite, nous avons exposé en détail la problématique qui a constitué le point de départ de notre mission, tout en dévoilant la solution que nous avons élaborée pour y répondre, ultérieurement les objectifs que nous nous sommes fixés. Finalement nous avons clôturer ce chapitre en exposant la méthodologie que nous avons consciencieusement adoptée tout au long de notre stage

Chapitre 2 : Spécification des besoins

I. Introduction

Dans ce chapitre nous intéressons de l'étape de spécifications de besoins [12], en effet nous allons Identifier les différents acteurs de notre application et analyser les besoins fonctionnels et non fonctionnels. Ces besoins seront par la suite exprimés sous forme des diagrammes de cas d'utilisation.

II. Identification des acteurs

Au cours de la phase de documentation et de recherche, nous avons réussi à repérer les différents acteurs qui pourraient interagir avec notre système. Nous distinguons alors deux acteurs pour notre système qui sont :

- **Utilisateur** : Au cœur de notre écosystème, l'utilisateur occupe une position centrale dans notre application. Il bénéficie de fonctionnalités spécialement conçues pour répondre à ses besoins, contribuant ainsi à une expérience enrichissante.
- **Administrateur** : Il joue un rôle crucial dans la gestion et la supervision du système, assurant son bon fonctionnement et répondant aux besoins de l'utilisateur

III. Identification des besoins fonctionnels

Les besoins fonctionnels [13] de notre application mobile englobent diverses fonctionnalités visant à garantir une expérience utilisateur complète et efficiente. Parmi ces éléments, nous identifions :

III.1 Fonctionnalités offertes à l'utilisateur

Les principales fonctionnalités qui seront offertes par notre application a l'utilisateur sont les suivantes :

- **Inscription et Authentification :** L'utilisateur peut effectuer une inscription conviviale avec vérification d'identité, sécurisation du processus, et obligation de fournir des informations valides.

- **Consulter le profil :** L'utilisateur peut effectuer des modifications sur son profil, que ce soit en éditant ses informations personnelles ou en changeant son mot de passe.
- **Consulter les parkings :** Les utilisateurs peuvent accéder à une liste de parkings avec leurs descriptions respectives. Ils ont la possibilité de rechercher ou de demander la réservation en saisissant les données nécessaires de manière flexible.
- **Consulter les activités :** Les utilisateurs ont la possibilité de consulter une liste de toutes les activités de réservation récentes, contenant les 5 dernières réservations avec leurs détails. Ils peuvent également supprimer ou modifier leur réservation si les conditions le permettent. Un code QR sera disponible sur chaque réservation pour faciliter l'entrée dans le parking.
- **Covoiturage sécurisé :** Les utilisateurs ont accès à une fonctionnalité de covoiturage, que ce soit en tant que passager ou conducteur, avec une planification sécurisée. Ils peuvent aussi voir le numéro d'immatriculation du véhicule et contacter facilement les autres parties en toute sécurité.
- **Consulter les horaires de transports publics :** L'intégration des horaires et retards des transports en commun seront assurées en temps réels pour l'utilisateur.
- **Notifications intégrées :** Des alertes et des notifications personnalisées seront proposées.
- **Feedback et réclamations :** Notre plateforme devra offrir une interface interactive permettant aux utilisateurs de fournir leurs retours.

III.2 Fonctionnalités offertes à l'administrateur

Concernant l'administrateur, les fonctionnalités offertes seront les suivantes :

- **Consulter les utilisateurs :** L'administrateur peut aisément consulter les utilisateurs, et effectuer des modifications.

- **Consulter des réclamations :** L'administrateur peut facilement consulter les réclamations et prendre des décisions, telles que bloquer les utilisateurs si nécessaire.
- **Consulter les parkings :** L'administrateur peut facilement consulter la liste des parkings et ajouter ou supprimer des entrées de parking.
- **Envoyer des notifications :** L'administrateur peut facilement envoyer des notifications à tous les utilisateurs connectés.
- **Consultation des horaires de transport public :** L'administrateur peut consulter les horaires de transports publics (métro, tramway, bus), et effectuer des modifications si nécessaire.

IV. Modélisation des besoins fonctionnels

Dans cette section, nous allons exposer les divers diagrammes de cas d'utilisation [14] qui nous permettant de structurer les besoins des utilisateurs et les objectifs du système visant à atteindre

IV.1 Vue globale de système

La figure 7 présente le diagramme de cas d'utilisation général de l'application, mettant en avant les principales fonctionnalités du système ainsi que ses interactions avec les différents acteurs.

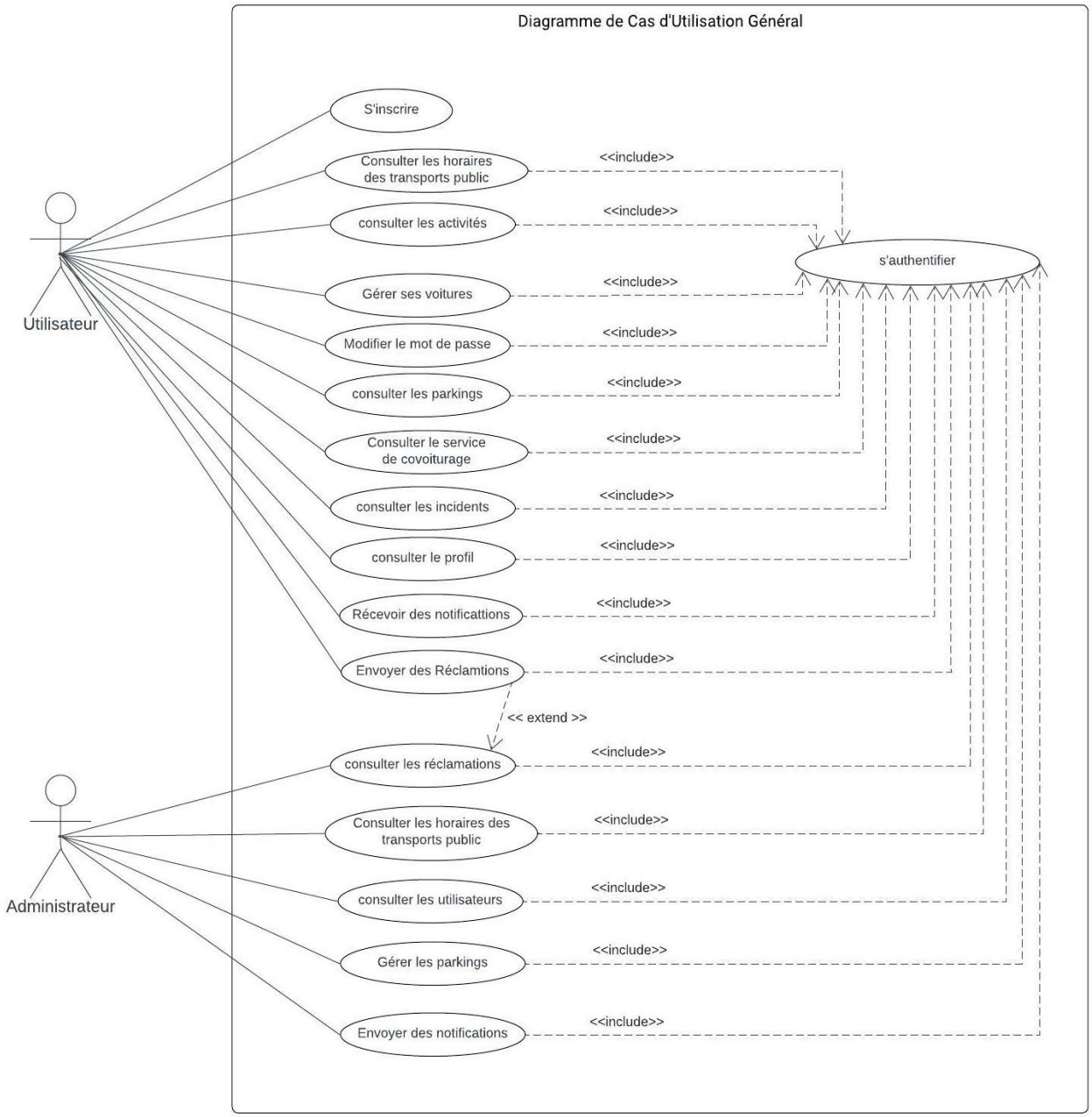


Figure 7 : Diagramme de cas d'utilisation général

IV.2 Raffinement de cas d'utilisation

Afin de mieux mettre en évidence les besoins de notre système, nous allons élaborer une décomposition détaillée [15] du diagramme de cas d'utilisation général en créant des sous Diagramme de cas d'utilisation pour chaque composant du système sur lequel nous avons travaillé

- **Raffinement de cas d'utilisation ' Réserver parking ' :**

Le diagramme de cas d'utilisation correspond est illustré dans la figure 8 :

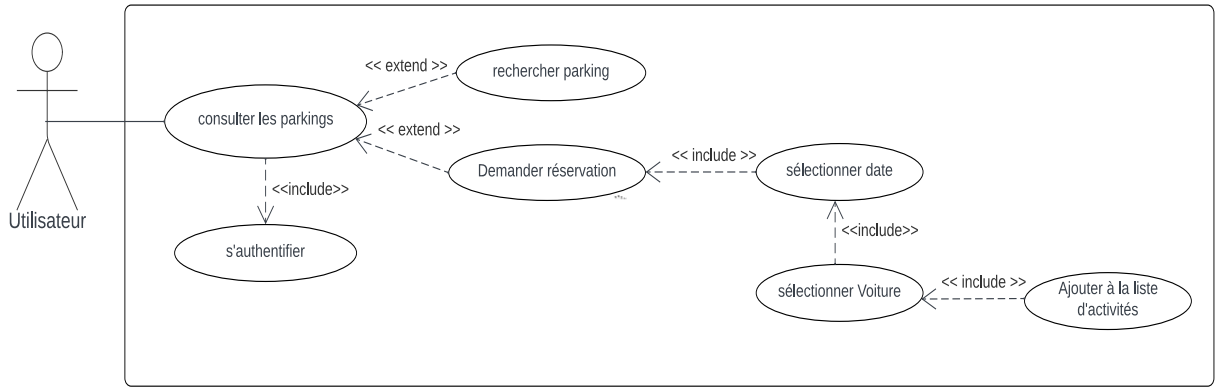


Figure 8 : Diagramme de cas d'utilisation raffiné « Réserver parking »

La description textuelle du cas d'utilisation est donnée par le tableau 2 :

Cas d'utilisation	Réserver parking
Acteur principal	Utilisateur
Pré Conditions	Liste de parkings ouverts
Post Conditions	La réservation est effectuée et le ticket est ajouté
Scénario nominal	1. L'utilisateur consulte la liste des parkings disponibles. 2. L'utilisateur sélectionne un parking. 3. Le système affiche les détails du parking. 4. L'utilisateur saisit les informations nécessaires pour faire une réservation (Date et Voiture). 5. Le système affiche un message de succès et ajoute un ticket de réservation contenant les détails avec un code QR dans la liste des activités de l'utilisateur.
Enchainements Alternatifs	Si la date de début est supérieure à la date de fin, un message d'erreur s'affiche. Si la date de début n'est pas au moins 1 heure après la date actuelle, un message d'erreur s'affiche. Si une autre réservation n'est pas encore terminée, un message d'erreur s'affiche.

Tableau 2 : Description du cas d'utilisation « Réserver parking »

- **Raffinement de cas d'utilisation ' Consulter les horaires de transports public ' :**

Le diagramme correspondant au cas d'utilisation ' consulter les horaires de transports publics ' est illustré par la figure 9

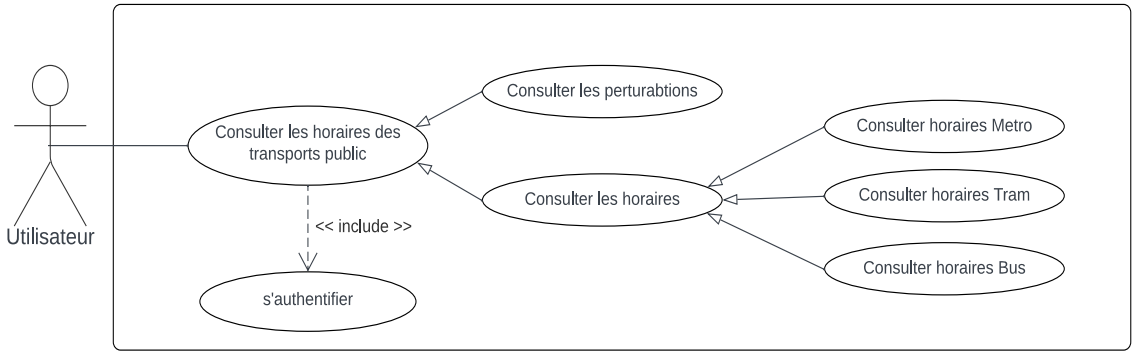


Figure 9 : Diagramme de cas d'utilisation raffiné « Consulter les horaires de transport public »

La description textuelle du cas d'utilisation est donnée par le tableau 3 :

Cas d'utilisation	Consulter les horaires de transport public
Acteur principal	Utilisateur
Pré Conditions	Interface contenant les informations nécessaires concernant les horaires et les perturbations en temps réels ouverte
Post Conditions	Affichage des données effectué avec succès
Scénario nominal	1. L'utilisateur consulte les horaires en temps réel. 2. Le système affiche les trois types de transport disponibles : métro, tram, et bus. 3. L'utilisateur sélectionne un type de transport (métro, tram, ou bus). 4. Le système affiche les informations détaillées pour le type de transport sélectionné.
Enchainements Alternatifs	Erreur lors de la récupération des horaires : Le système affiche un message d'erreur informant l'utilisateur de l'échec de la récupération des horaires.

Tableau 3 : Description du cas d'utilisation « Consulter les horaires de transport public »

- **Raffinement de cas d'utilisation ' Consulter le service de covoiturage ' :**

Le diagramme correspondant au cas d'utilisation ' Consulter le service de covoiturage ' est illustré par la figure 10 :

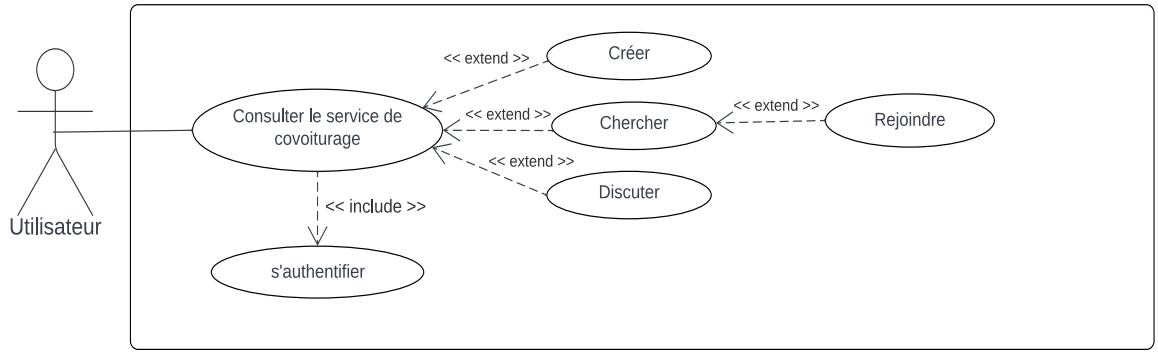


Figure 10 : Diagramme de cas d'utilisation raffiné « Consulter le service de covoiturage »

La description du cas d'utilisation raffiné pour la partie création d'un covoiturage est fournie par le tableau suivant :

Cas d'utilisation	Créer un covoiturage
Acteur principal	Utilisateur
Pré Conditions	Interface de création de covoiturage ouverte
Post Conditions	Covoiturage effectué et ajouté à la liste des trajets de covoiturage
Scénario nominal	1. L'utilisateur suit un ensemble d'étapes et remplit correctement les champs nécessaires. 2. Le système crée avec succès un covoiturage avec tous les détails requis. 3. Le covoiturage est ajouté à la liste des trajets.
Enchainements Alternatifs	Si les valeurs entrées ne respectent pas les conditions de saisie : Le système afficher un message d'erreur.

Tableau 4 : Description du cas d'utilisation « Créer un covoiturage »

- **Raffinement de cas d'utilisation ' Recevoir des notifications ' :**

Le diagramme correspondant au cas d'utilisation ' Recevoir des notifications ' est illustré par la figure 11.

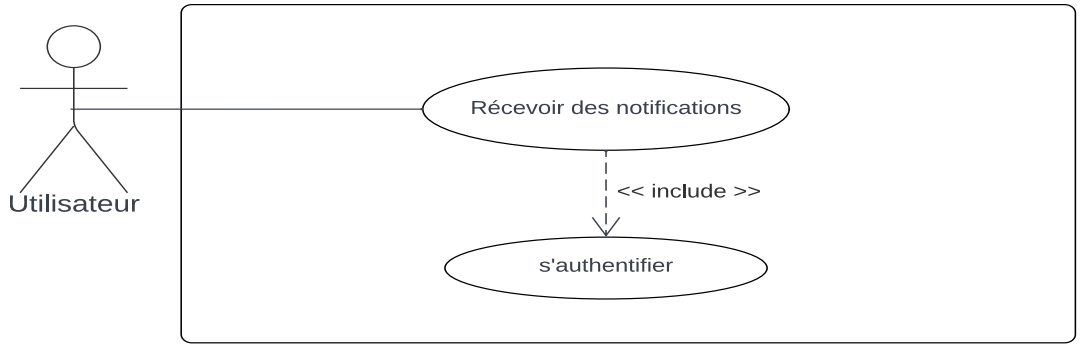


Figure 11 : Diagramme de cas d'utilisation raffiné « Recevoir des notifications »

- **Raffinement de cas d'utilisation ' Envoyer des notifications ' de l'administrateur :**

Le diagramme correspondant au cas d'utilisation ' Envoyer des notifications ' est illustré par la figure 12.

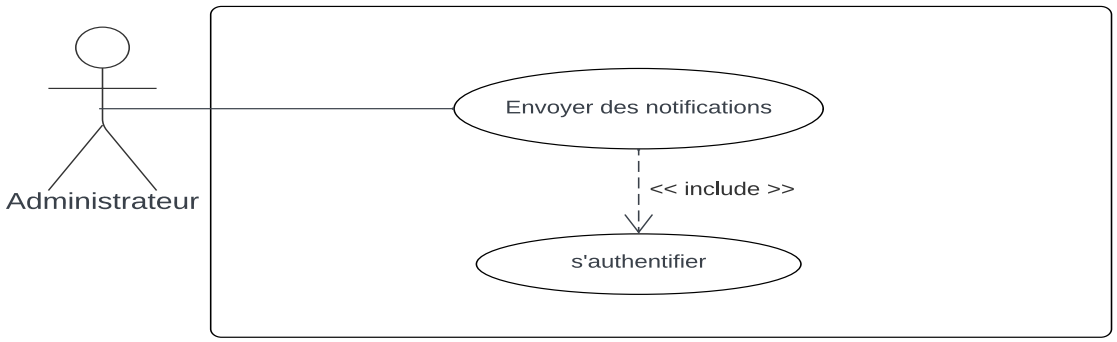


Figure 12 : Diagramme de cas d'utilisation raffiné « Envoyer des notifications »

- **Raffinement de cas d'utilisation ' Consulter les horaires des transports publics ' de l'administrateur :**

Le diagramme correspondant au cas d'utilisation ' Consulter les horaires des transports publics ' est illustré par la figure 13.

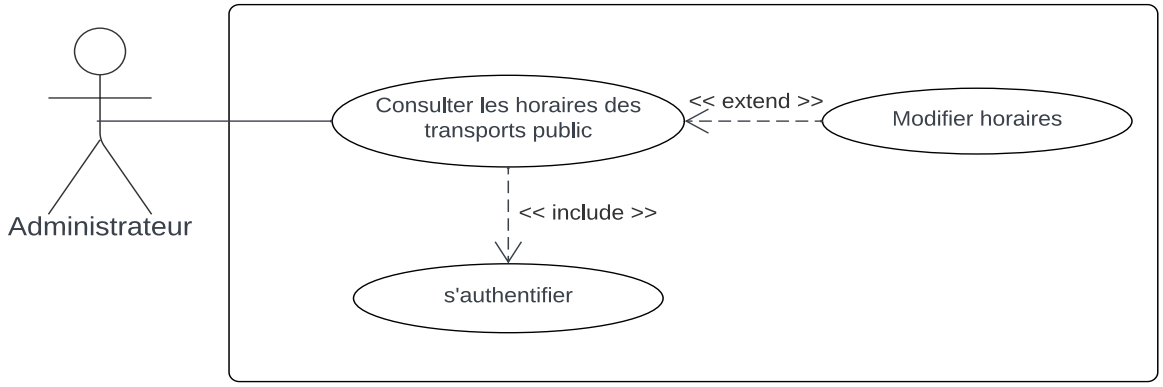


Figure 13 : Diagramme de cas d'utilisation raffiné « Consulter les horaires des transports publics »

V. Identification des besoins non fonctionnels

En plus des besoins fonctionnels notre application doit vérifier des besoins non fonctionnels [16]. Ces besoins présentent les contraintes et les considérations qui doivent être prises en compte lors de la conception et de la réalisation d'un système, afin d'assurer son fonctionnement et de fournir une meilleure expérience client. Dans notre cas, nous avons identifié les principales restrictions que notre solution doit respecter, à savoir :

- **Maintenabilité :** Le code doit être propre et lisible, bien structuré et commenté. L'application doit être maintenable, et donc facilement prise en main par un développeur junior.
- **Ergonomie :** Tous les standards d'ergonomie doivent être présents : interface utilisateur présentable, simple, compréhensible, bien claire et facile à utiliser.
- **Performance :** L'application doit être performante et robuste, signifiant que l'accès aux données doit être rapide.
- **Sécurité :** Les APIs de notre solution seront sécurisées et contrôlées par des tokens et des cookies.

Conclusion

Ce chapitre a été consacré à l'analyse et la spécification des besoins du projet. En premier lieu, nous avons identifié les différents acteurs du système. Ensuite, nous avons analysé les besoins fonctionnels et non fonctionnels de chaque partie sur laquelle nous avons travaillé. Finalement, nous sommes arrivés à tracer et à raffiner les diagrammes des cas d'utilisation correspondants au système global et les parties sur lesquelles nous avons travaillé.

Chapitre 3 : Etude Conceptuelle

I. Introduction

Après avoir décortiqué les besoins des utilisateurs, nous abordons dans ce chapitre l'étude conceptuelle. En effet dans une première partie, noua allons présenter la conception générale qui exposera l'architecture logique ainsi que physique de notre plateforme. Ensuite nous allons détailles la vue statique de notre projet en se basant sur le diagramme de classes. Enfin, la vue dynamique sera présentée et ceci en élaborant les diagrammes de séquences.

II. Conception générale

En reconnaissant son importance fondamentale, cette section se concentre sur la conception architecturale de notre solution [18]. Nous débutons en clarifiant les deux aspects essentiels de cette conception : l'architecture logique et l'architecture physique.

II.1 Architecture logique

Pour notre projet nous avons choisi l'architecture ' ***Clean Architecture*** ' *[19]* qui est une architecture dépasse le cadre d'une simple méthodologie de conception logicielle [20]. Elle représente une approche philosophique qui place les besoins métier au cœur de son architecture. En privilégiant la séparation des préoccupations, elle vise à rendre le code maintenable et évolutif en organisant les concepts autour des entités métier, des cas d'utilisation et des interfaces utilisateur. Cette approche se démarque par son engagement envers la maintenabilité à long terme, mettant en avant une séparation nette entre les concepts métier et l'infrastructure technique.

Comparée à d'autres architectures [21] telles que MVC, MVVM ou l'architecture hexagonale, la Clean Architecture offre une modularité accrue et une indépendance vis-à-vis des frameworks. Cette indépendance facilite les tests unitaires, la réutilisation du code et l'évolutivité de l'application, contribuant ainsi à une meilleure gestion de la complexité et des changements à long terme.

II.1.1 Pourquoi choisir la Clean Architecture ?

Clean Architecture a été choisis pour offrir une voie vers un code plus propre, plus ordonné et plus durable. En l'adoptant, les équipes de développement peuvent créer des applications qui non seulement répondent aux exigences actuelles, mais sont également prêtes pour les évolutions futures de la technologie. La figure 14 illustre cette architecture.

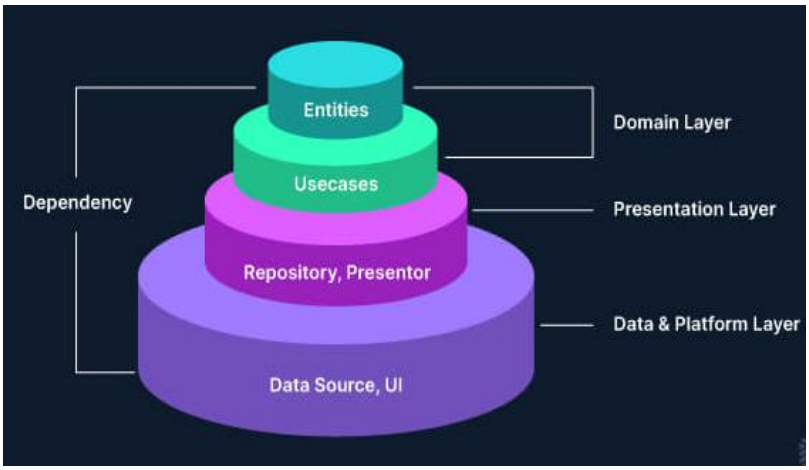


Figure 14 : Clean Architecture

II.1.2 Les Composants de la Clean Architecture

La Clean Architecture peut-être structurée en trois principaux domaines : **Domain, Data, et Présentation** [22] comme illustre la figure 15. Chacune de ces catégories contient des composants spécifiques qui remplissent des rôles précis.

Présentation

Cette couche est chargée de présenter les données à l'utilisateur et gérer les entrées de l'utilisateur. Il ne doit contenir aucune logique métier et doit être aussi simple que possible.

Domain

Cette couche contient la logique métier de l'application et définit les cas d'utilisation. Il doit être libre de toute dépendance vis-à-vis des autres couches et doit être testable isolément.

Data

Cette couche est responsable de la récupération et du stockage des données. Il devrait être indépendant de la couche de domaine et ne devrait concerner que l'accès et la persistance des données.

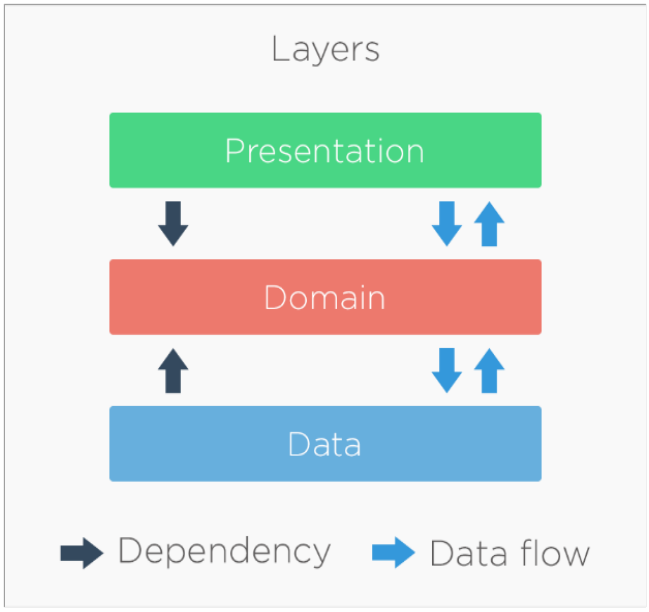


Figure 15 : Les Composants de Clean Architecture

II.2 Architecture physique

Concernant l'architecture physique, nous l'avons détaillé en se basant sur le diagramme de déploiement [24] qui permet de décrire la disposition physique des différents modules qui composent le système et les différentes connexions entre eux. La figure 16 représente les ressources et les connexions avec les différents composants qui appartiennent à notre application mobile « Smart Way ».

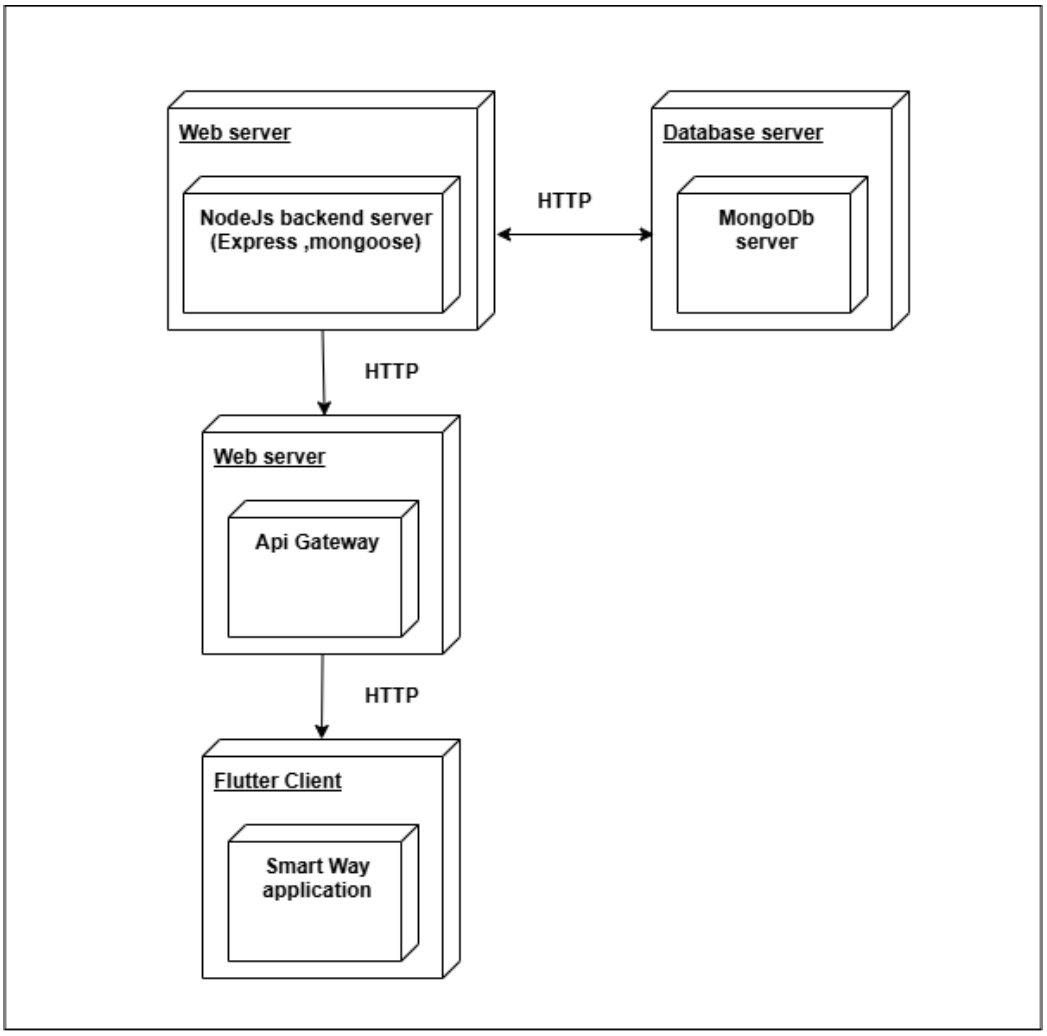


Figure 16 : Diagramme de Déploiement

III. Conception détaillé

Dans cette partie, nous nous intéressons de la conception détaillée de notre projet ou nous allons présenter les vues statique et dynamique à travers les diagrammes de classe et de séquences.

III.1 Vue statique

La figure 17 illustre le diagramme de classes composé de différentes interfaces et classes liées entre elles, intervenant dans la récupération des informations de l'utilisateur connecté au système.

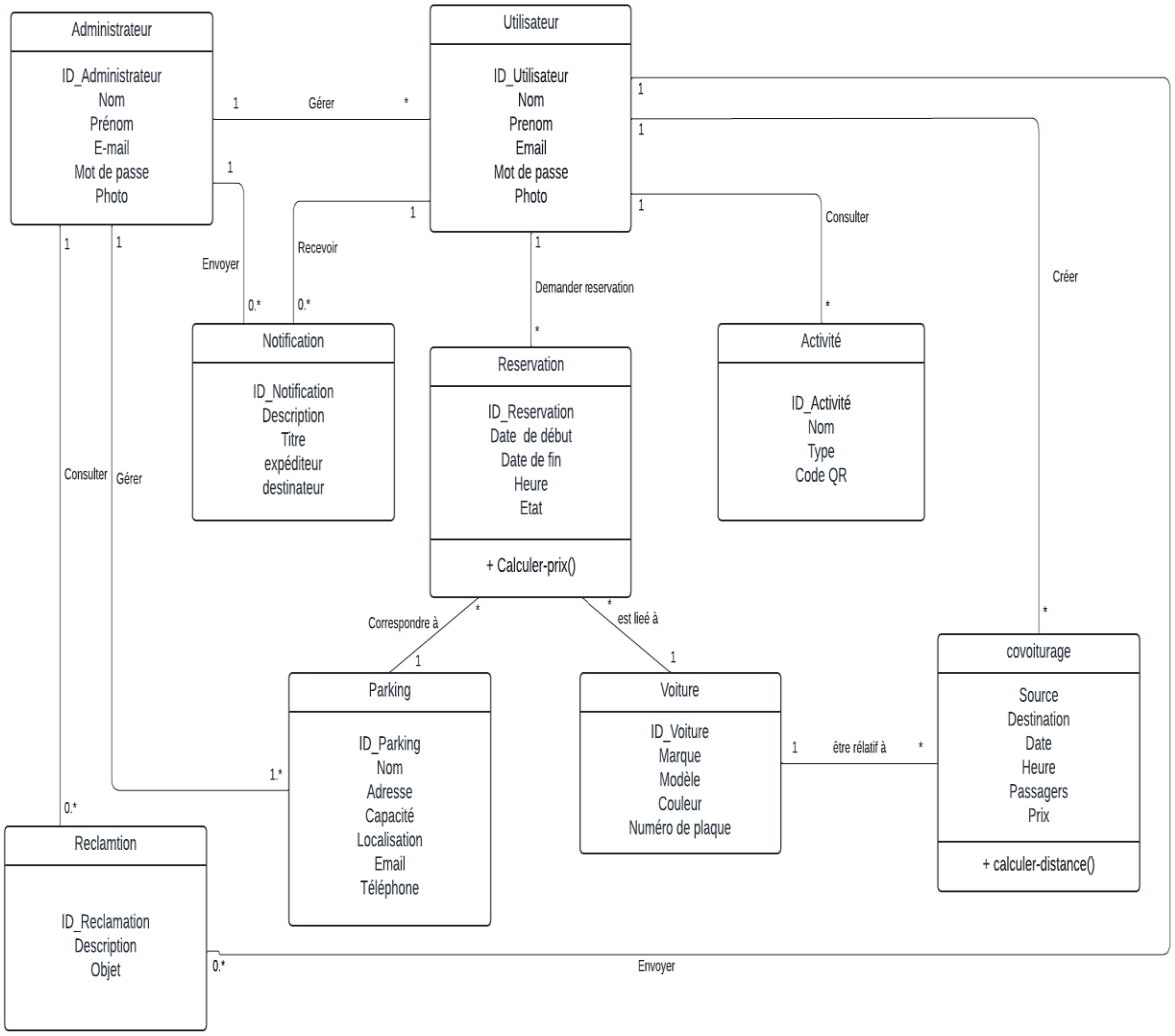


Figure 17 : Diagramme de classes

Parmi les classes principales du diagramme, nous distinguons :

Utilisateur : Un utilisateur représente une personne qui interagit avec le système. Les informations stockées pour chaque utilisateur incluent son ID, nom, prénom, email, mot de passe et une photo de profil. Un utilisateur peut gérer des actions comme demander des réservations et consulter des informations.

Notification : Une notification est un message envoyé pour informer d'un événement ou d'une mise à jour. Chaque notification a un ID, une description, un titre, un expéditeur et un destinataire.

Réservation : Une réservation représente une demande de réservation d'un service ou d'une ressource. Elle contient des informations comme l'ID de réservation, la date de début, la date de fin, l'heure et l'état de la réservation. Elle inclut aussi une méthode pour calculer le prix de la réservation.

Activité : Une activité représente une liste des actions que les utilisateurs peuvent consulter. Elle est définie par un nom, un type et un code QR.

Réclamation : Une réclamation est une plainte déposée par un utilisateur concernant un problème rencontré. Elle inclut une description et un objet pour spécifier le type de problème.

Parking : Un parking est un lieu où les voitures peuvent stationner. Les informations stockées incluent le nom du parking, son adresse, sa capacité, sa localisation, un email de contact et un numéro de téléphone.

Voiture : Une voiture représente un véhicule dans le système de covoiturage ou de réservation. Elle est définie par sa marque, son modèle, sa couleur et son numéro de plaque.

Covoiturage : Un covoiturage est une entente entre utilisateurs pour partager un trajet en voiture. Les détails incluent la source, la destination, la date, l'heure, les passagers et le prix. Une méthode pour calculer la distance du trajet est incluse.

III.2 Vue dynamique

Pour montrer la vue dynamique [23] de notre système, nous optons pour les diagrammes de séquence objet. Dans la suite de cette section, nous avons choisir d’exposer les diagrammes de séquences relatifs à l’authentification, l’inscription, la création de covoiturage et l'ajout de véhicule

III.2.1 Diagramme de séquence relatif à l'authentification

Pour mettre en évidence l'interaction entre les différents composants de notre système, nous exposons le diagramme de séquence présenté dans la figure 18, décrivant le processus d'authentification de l'utilisateur

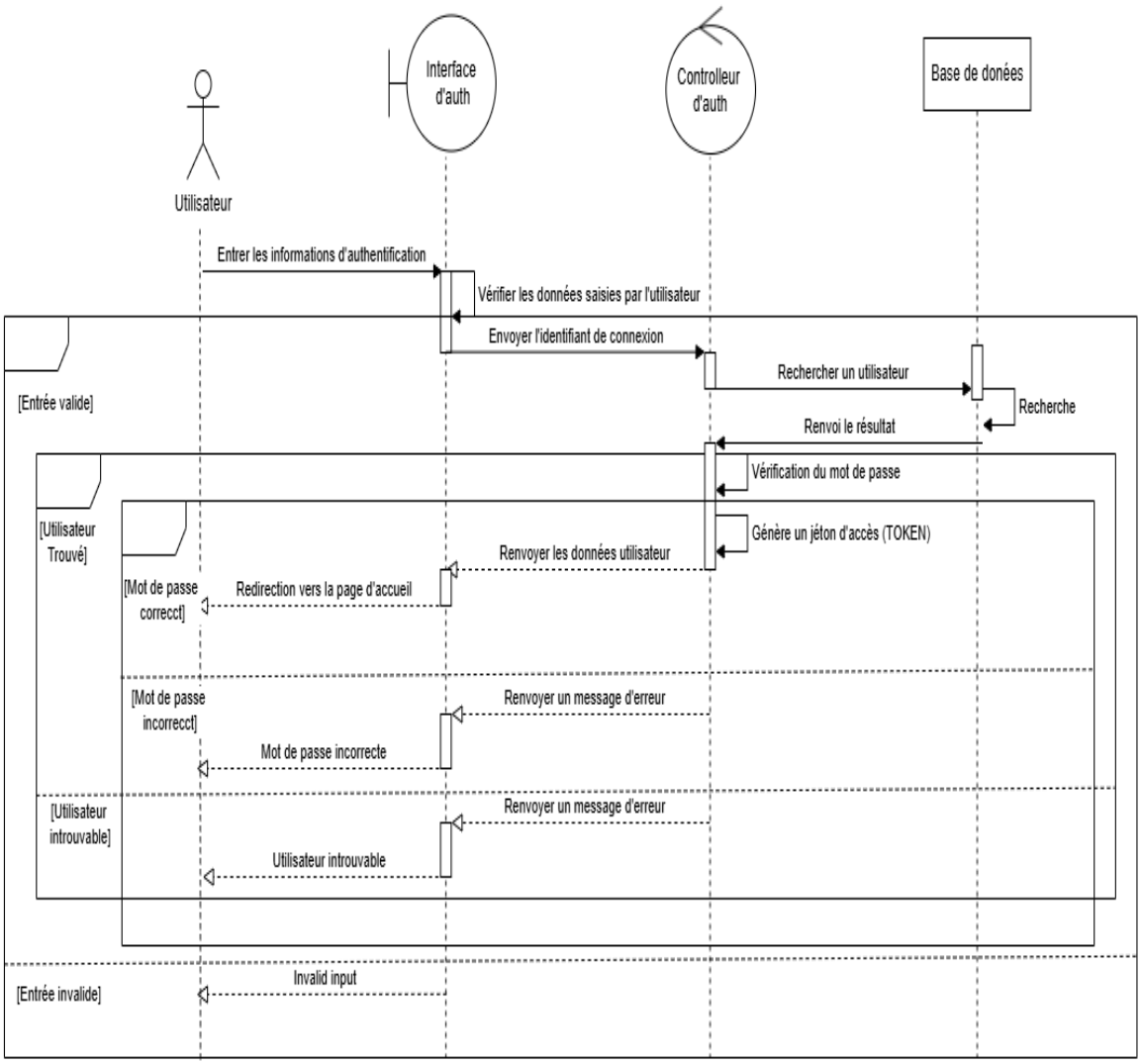


Figure 18 : Diagramme de séquence relatif à l'authentification

III.2.2 Diagramme de séquence relatif à l'ajout de véhicule

Afin de mettre en lumière l'interaction entre les divers composants de notre système, nous présentons le diagramme de séquence illustré dans la figure 19, décrivant le processus d'ajout de véhicule par l'utilisateur.

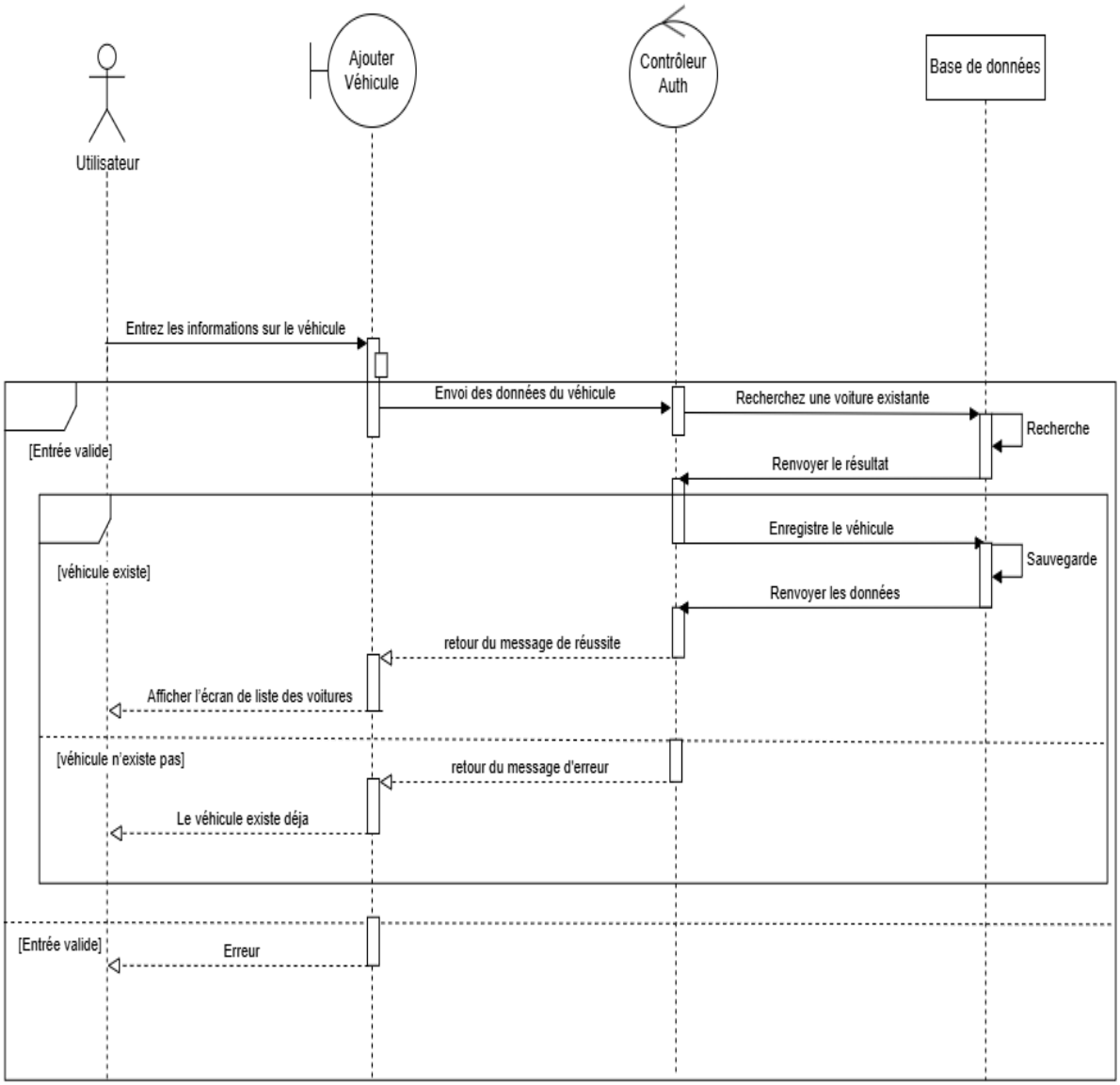


Figure 19 : Diagramme de séquence relatif à l'ajout de véhicule

III.2.3 Diagramme de séquence relatif à la création de covoiturage

Le diagramme de séquences, illustré par la figure 20 montre les différents échanges de messages nécessaires pour la fonctionnalité « création de covoiturage ».

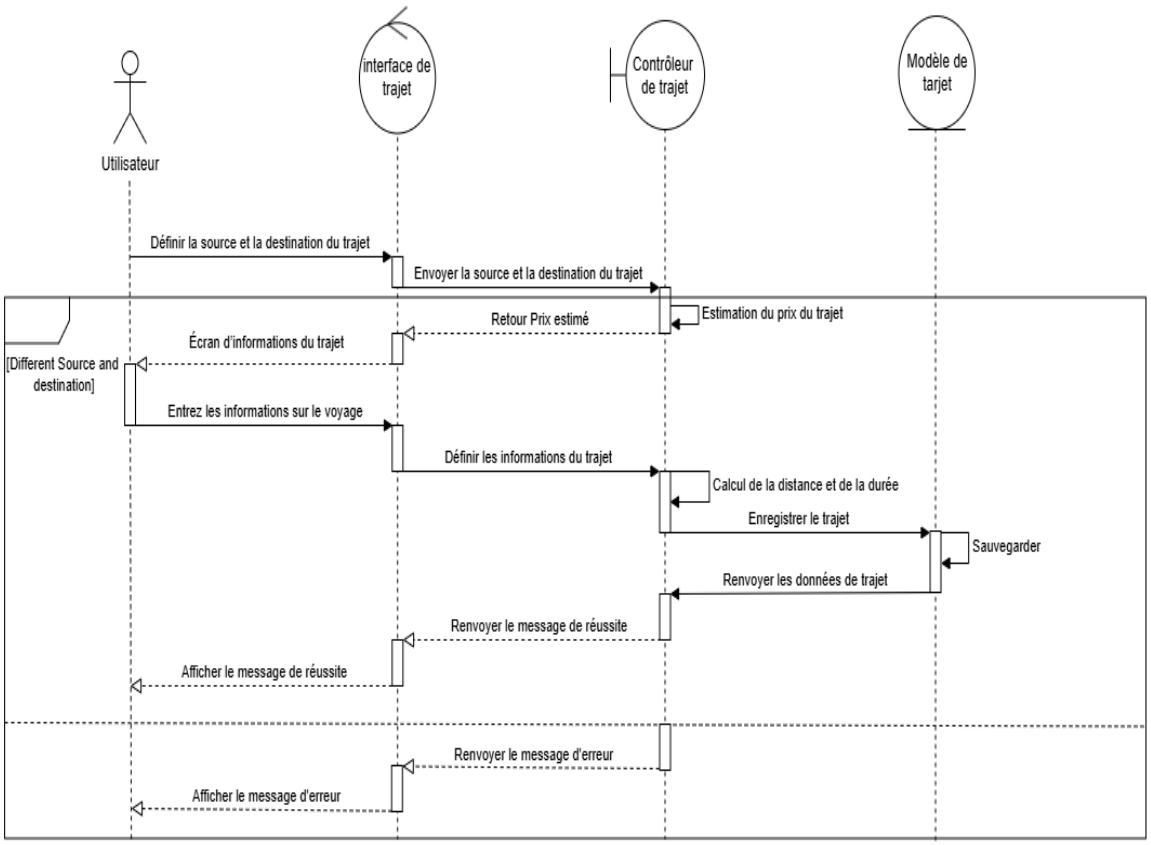


Figure 20 : Diagramme de séquence relatif à la création de covoiturage

III.2.4 Diagramme de séquence relatif à la suppression de véhicule

La figure 21 montre le diagramme de séquences relatif au cas d'utilisation « supprimer véhicule » qui permet à un utilisateur de supprimer un véhicule de sa liste.

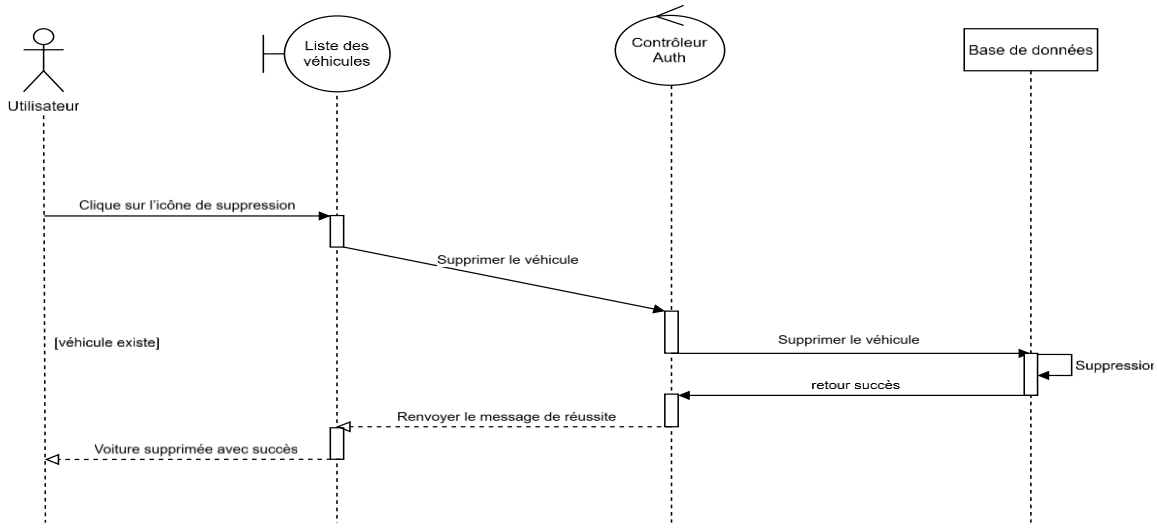


Figure 21 : Diagramme de séquence relatif à la suppression de véhicule

Conclusion

Ce chapitre a été consacré à la présentation de l'étude conceptuelle de la solution à mettre en place. Nous avons détaillé tout d'abord les architectures logique et physique. Ensuite nous avons exposé les vue statiques et dynamique en se basant respectivement sur le diagramme de classes et de séquences.

Chapitre 4 : Réalisation

I. Introduction

Une fois la phase de conception de l'application est terminée, nous pouvons maintenant passer à la réalisation qui fait l'objet de ce chapitre et élaborée minutieusement les diagrammes associés. En effet, nous décrirons l'environnement de travail, en mettant en lumière les technologies et les outils que nous avons sélectionnés. Ensuite, nous présenterons un aperçu des interfaces réalisées tout en expliquant les fonctionnalités développées.

II. Environnement de développement

Dans cette section, nous dévoilons les fondements de notre solution en exposant les frameworks et les logiciels qui ont été mobilisés pour sa réalisation.

II.1 Technologies utilisées

Afin de réaliser notre application, nous avons choisis les technologies suivantes :

Flutter

Flutter est un kit de développement logiciel (SDK) d'interface utilisateur open-source créé par Google. Il est utilisé pour développer des applications pour Android, iOS, Linux, Mac, Windows, Google Fuchsia et le web à partir d'une seule base de code [25].

Node JS

Node.js est un environnement de serveur open source multiplateforme qui peut s'exécuter sur Windows, Linux, Unix, macOS, etc. C'est un environnement d'exécution JavaScript back-end, s'exécute sur le moteur JavaScript V8 et se basant sur du code JavaScript en dehors d'un navigateur Web [26].

Express JS

ExpressJS est un framework qui se veut minimaliste. Très léger, il apporte peu de surcouches pour garder des performances optimales et une exécution rapide [27].

MongoDB

MongoDB est une base de données NoSQL orientée documents, qui stocke les données dans des documents BSON (Binary JSON). Elle permet une grande flexibilité et évolutivité, adaptée aux applications nécessitant une gestion efficace de grandes quantités de données variées. [28].

ReactJS

React (aussi appelé React.js ou ReactJS) est une bibliothèque (library) JavaScript, permettant de développer des applications web qui peuvent être rechargées sans recompiler [29].

Beautiful Soup

Beautiful Soup est une bibliothèque Python utilisée pour extraire des données de fichiers HTML et XML. Elle est particulièrement utile pour le web scraping, permettant de récupérer facilement le contenu des pages web et de le manipuler selon les besoins de l'utilisateur.

II.2 Langage de programmation

Dart

Dart est un langage de programmation optimisé pour les applications sur plusieurs plateformes. Il est développé par Google et est utilisé pour créer des applications mobiles, de bureau, de serveur et web. [30]

Python

Python est un langage de programmation interprété, orienté objet et de haut niveau. Il est largement utilisé pour l'analyse de données, l'intelligence artificielle et bien d'autres domaines, grâce à ses nombreuses bibliothèques et frameworks.

II.3 Logiciels utilisés

Visual Studio Code

Visual Studio Code est un éditeur de code open-source développé par Microsoft supportant un très grand nombre de langages grâce à des extensions. Il supporte l'autocomplétions, la coloration syntaxique, le débogage, et les commandes git [31].

Git

Git est un système de contrôle de version distribué largement utilisé dans le développement de logiciels [33].

Lucidchart

Lucidchart est une plateforme de collaboration en ligne, basée sur le cloud, permettant la création de diagrammes et la visualisation de données, et autres schémas conceptuels [34].

III. Présentation de la solution

Dans cette partie, nous allons exposer quelques interfaces de notre application en essayant de présenter les fonctionnalités les plus importantes dans notre travail.

III.1 Interface d'accueil

L'interface d'accueil de notre application illustré par la figure 22 offre une vision générale de notre solution.

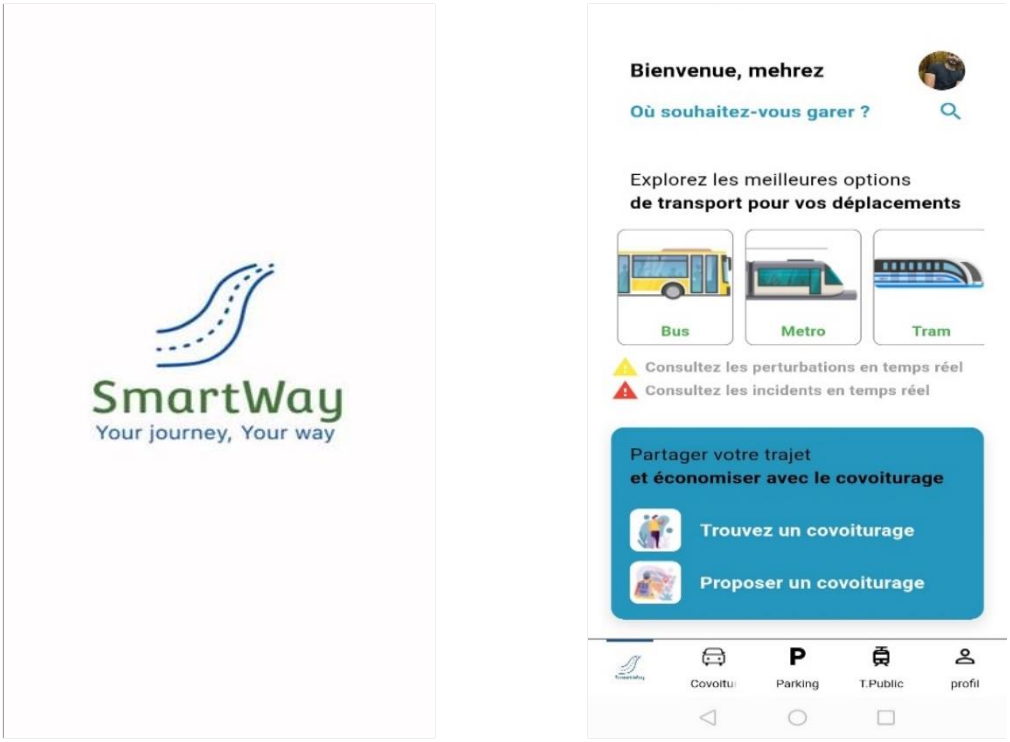


Figure 22 : Interface d'accueil

III.2 Interface d’authentification

La figure 23 illustre l’interface d’authentification où les utilisateurs qui ont déjà un compte peuvent se connecter à notre plateforme en saisissant leurs adresses email et leurs mots de passe sauvegardés déjà dans la base de données.

Figure 23 : Interface authentication

III.3 Interface d’inscription

Tous les utilisateurs peuvent s'inscrire sur la plateforme en fournissant les informations nécessaires, comme indiqué dans la figure 24. Après la saisie des données valides, une page de vérification s'ouvre, comme présenté dans la figure 25.

Un code est envoyé à l'adresse e-mail saisie lors de la première étape. Si le code de vérification est correct, l'inscription est réussie et l'interface de connexion s'ouvre.

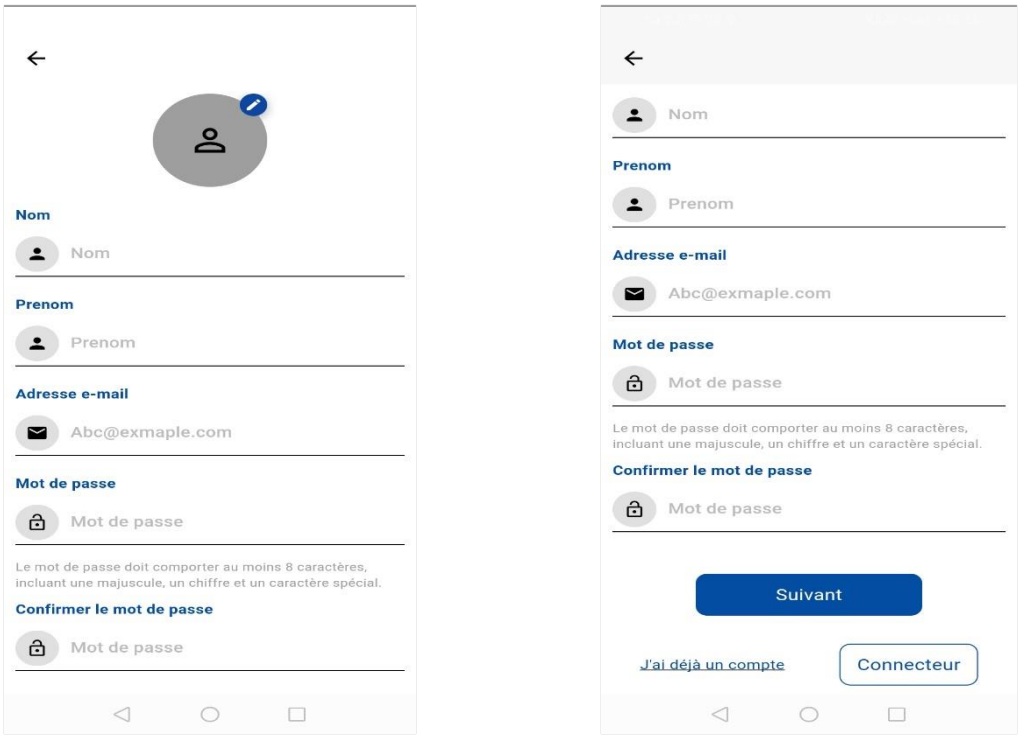


Figure 24 : Interface inscription

Figure 25 : Interface vérification

III.4 Interface de parking

Les utilisateurs inscrits peuvent consulter la liste des parkings disponibles en un seul clic sur l'icône "parking" située en bas de la page dans la barre de navigation. Une fois cette action est effectuée, une interface affichant la liste de tous les parkings apparaît, incluant le nom, l'adresse et la capacité, comme illustré dans la figure 26 De plus, la possibilité de recherche par nom est disponible en haut de la liste.

En cliquant sur un parking, une fenêtre de description s'ouvre, affichant toutes les informations disponibles sur le parking, comme indiqué dans la figure 27.

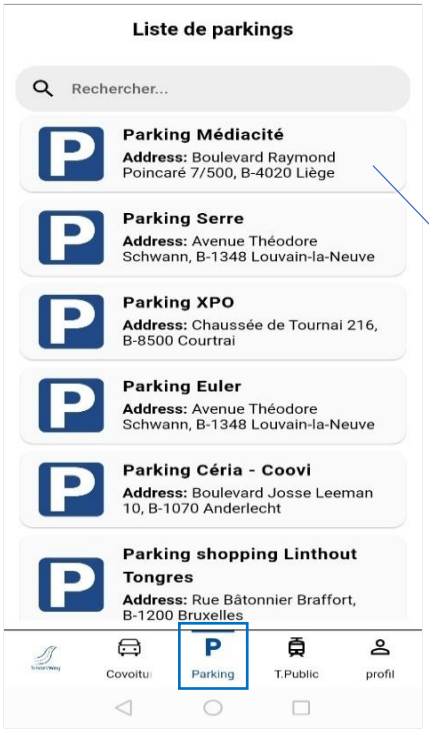


Figure 26 : barre de navigation

Figure 27 : Parking détails

III.5 Interface de réservation

Pour effectuer une réservation, les utilisateurs peuvent simplement sélectionner un parking et appuyer sur celui-ci. Une fois sélectionné, une fenêtre de description s'ouvre, et en bas de cette fenêtre, un bouton intitulé "Réserver" est disponible comme indique la figure 27. Après avoir appuyé sur ce bouton, les utilisateurs doivent suivre deux étapes et saisir des informations valides pour finaliser la réservation avec succès.

Dans la première étape, comme illustré dans la figure 28, ils doivent tout simplement spécifier deux dates d'entrée et de sortie valides.

Pour la deuxième étape, comme indiqué dans la figure 29, il suffit de sélectionner le véhicule qu'ils souhaitent garer, si celui-ci est disponible. Sinon, ils peuvent appuyer sur le bouton « Ajouter un véhicule » (bouton numéro 1 comme illustré dans la figure 29). Une fois sélectionné, une interface d'ajout de voiture s'ouvre, où ils doivent entrer les informations nécessaires et appuyer sur « Ajouter un véhicule », comme indiqué dans la figure 30.

Une fois ces étapes terminées, ils n'ont qu'à cliquer sur le bouton "Réserver". Si tout se passe comme prévu, un message de succès apparaîtra et la réservation sera ajoutée à la liste des activités de l'utilisateur.

Figure 28 : saisie étape 1

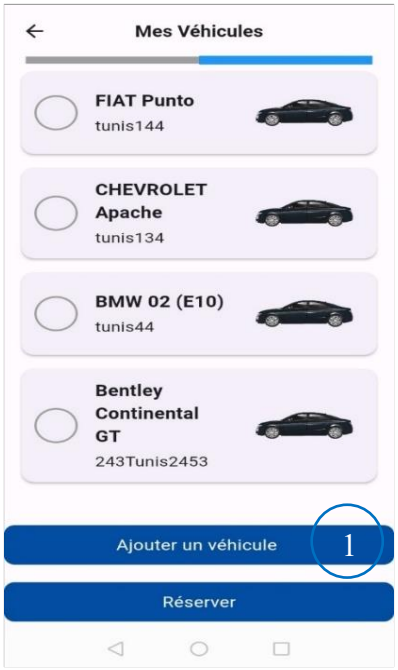


Figure 29 : saisie étape 2

Figure 30 : interface ajouter voiture

III.6 Interface liste des activités

La liste des activités est placée dans le tableau de bord utilisateur comme indiqué dans la figure 31. Elle présente les détails des 5 dernières réservations, notamment leur état, date de début, date de fin, durée, prix et détails du véhicule. Le prix est

automatiquement calculé en fonction de la différence entre la date de début et la date de fin en le multipliant par un tarif horaire standard de 200 cents par heure. Chaque réservation est accompagnée d'un code QR accessible en appuyant sur le bouton numéro 1 comme indiqué dans la figure 32, facilitant ainsi l'entrée au parking lorsqu'il est scanné sur une machine spéciale à l'extérieur. De plus, les utilisateurs ont la possibilité de prolonger la réservation de 15 minutes avant son expiration en appuyant sur le bouton numéro 2 représentant une horloge, comme illustré dans la figure 32.

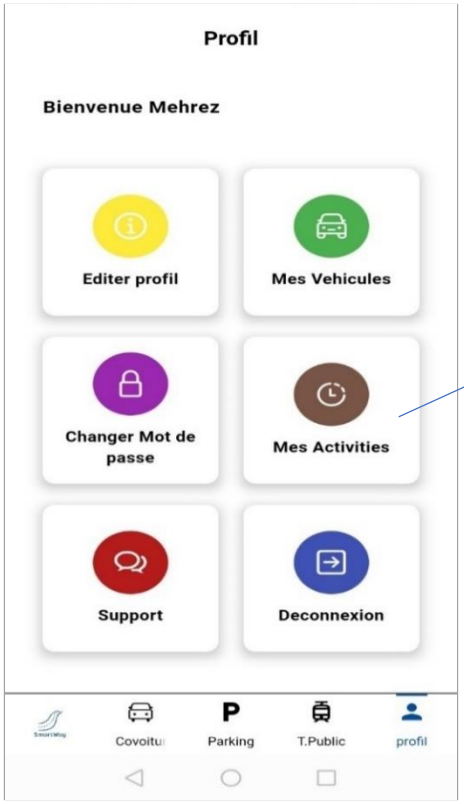


Figure 31 : interface tableau de bord

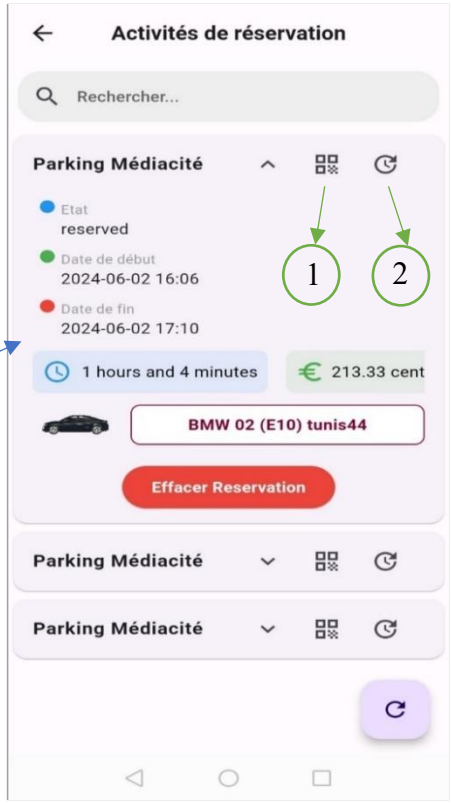
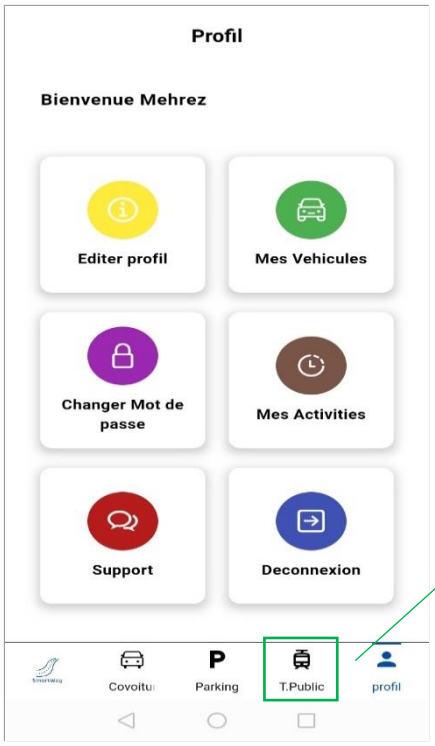


Figure 32 : Interfaces liste des activités

III.7 Interface du transport public

Les utilisateurs inscrits peuvent consulter les horaires du transport public ainsi que les retards en temps réel en un seul clic sur l'icône sous le nom ‘Transport public ‘ située en bas de page dans la barre de navigation, comme indiqué dans la figure 33. Une fois

cette action effectuée, une interface s'affiche proposant deux choix : "Transports publics" et "Perturbations", comme illustré dans la figure 34.



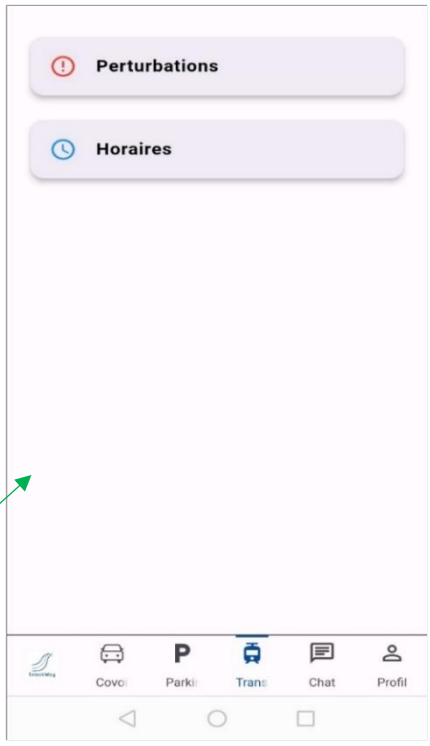


Figure 33 : barre de navigation **Figure 34 : liste de choix**

Si l'utilisateur sélectionne l'option "Perturbations", une liste affichant les perturbations du transport public s'ouvre, comme illustré dans la figure 35. Cette liste se met à jour en temps réel et se charge automatiquement dès qu'il y a de nouvelles perturbations. Elle est réalisée grâce à la technologie de web scraping avec la bibliothèque Beautiful Soup.

En choisissant l'option ‘Transports publics ‘, une liste comprenant trois types de transports s'affiche : "Métro", "Bus", "Tram", comme indiqué dans la figure 36. Une fois l’utilisateur sélectionne un type de transport, une autre liste présentant les numéros spécifiques des lignes de ce transport apparaît, comme illustré dans la figure 37. En appuyant sur un numéro de ligne, un fichier PDF s'ouvre avec les horaires et le schéma de cette ligne, offrant la possibilité de télécharger le fichier sur le stockage externe du téléphone.

Figure 35 : Liste de perturbations

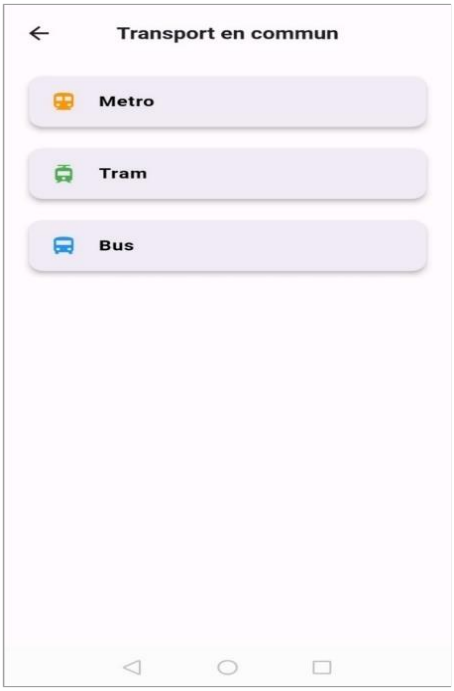


Figure 36 : liste de transport public

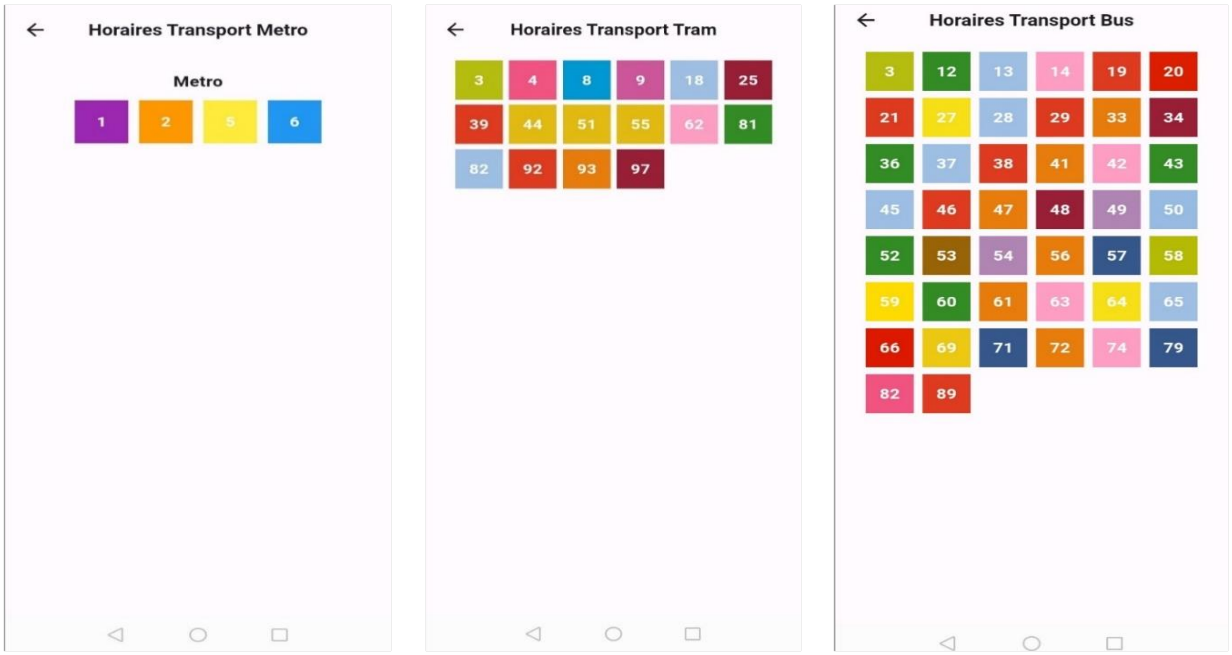


Figure 37 : Spécification de ligne de transport

III.8 Interface tableau de bord utilisateur (Dashboard User)

Les utilisateurs inscrits peuvent consulter un tableau de bord en sélectionnant l'icône du profil dans la barre de navigation. Ce tableau contient en haut le nom de l'utilisateur avec le mot "Bienvenue" et six menus, comme indiqué dans la figure 38 :

1. Edit Profile : C'est une interface permettant de mettre à jour les données de l'utilisateur. Il suffit de remplir les champs nécessaires et d'appuyer sur "Éditer". Si tout se passe bien, une fenêtre contextuelle de succès s'ouvre, comme illustré dans la figure 39.

2. Mes Véhicules : C'est une interface contenant les véhicules enregistrés par l'utilisateur, illustrée dans la figure 40.

3. Changer Mot de Passe : il s'agit d'interface permettant de changer le mot de passe de l'utilisateur si nécessaire. Il suffit de taper l'ancien mot de passe et le nouveau mot de passe. Si tout se passe bien, une fenêtre contextuelle de succès s'ouvre, comme illustré dans la figure 41.

4. Mes Activités : C'est une interface qui contient les cinq dernières réservations de l'utilisateur, déjà évoquée et illustrée dans la figure 32.

5. Support : C'est une interface pour soumettre des réclamations en cas de problème, illustrée dans la figure 42.

6. Déconnexion : si l'utilisateur clique sur l'icône numéro 6, une fenêtre contextuelle apparaît, demandant s'il souhaite se déconnecter. En appuyant sur « OK », l'utilisateur se déconnecte ; sinon, il peut annuler, comme illustré dans la figure 43.

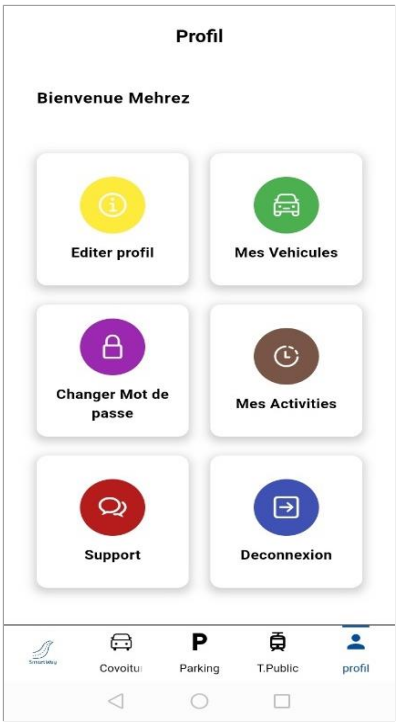


Figure 38 : tableau de bord utilisateur

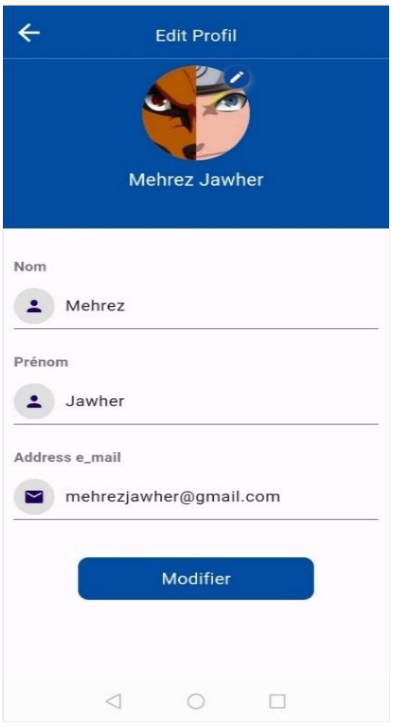


Figure 39 : Edit profile

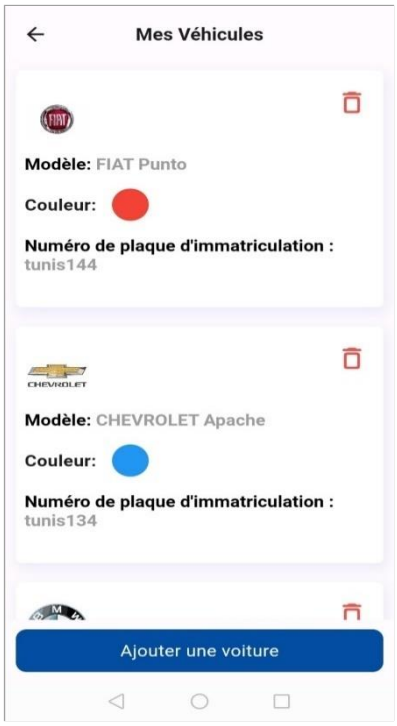


Figure 40 : Mes Véhicules

Figure 41 : Changer Mot de passe

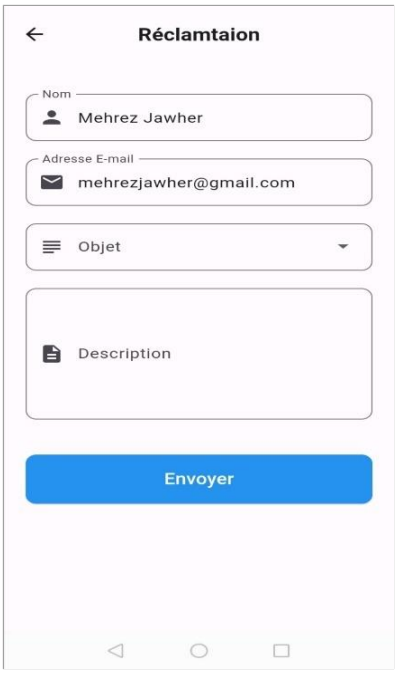


Figure 42 : Support

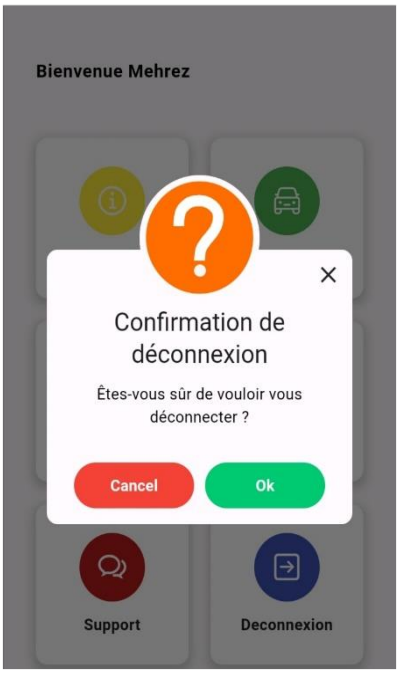


Figure 43 : Déconnexion

III.9 Interface Covoiturage

Cette interface contient deux options : "Ajouter un trajet " et " Trouver un trajet ", comme illustré dans la figure 44. Pour y accéder, il suffit de sélectionner l'icône voiture sous le nom "Covoiturage" dans la barre de navigation. Pour sélectionner une option, il suffit de cliquer sur le choix souhaité et d'appuyer sur le bouton "Sélectionner" pour valider et continuer.

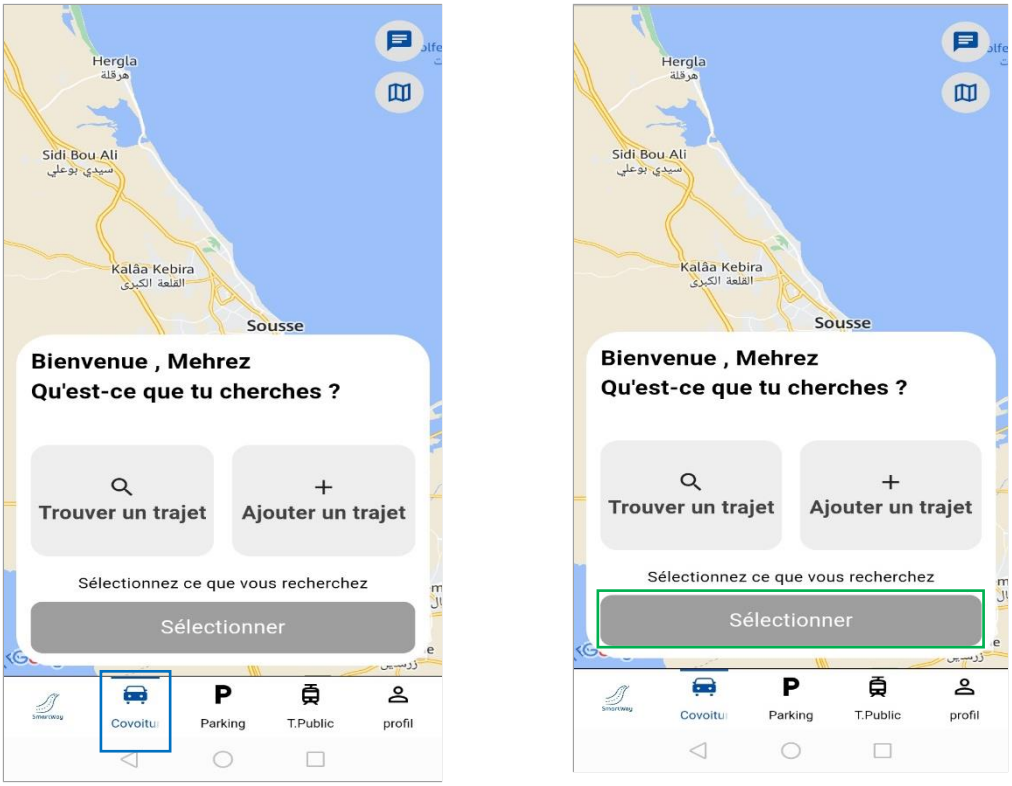


Figure 44 : Interface covoiturage

III.9.1 Interface chercher covoiturage

Si l'option "Trouver un trajet" est sélectionnée, une interface s'ouvre comme indiqué dans la figure 45. Il suffit d'entrer le point de départ et la destination du covoiturage. Si une correspondance est trouvée, une liste contenant tous les covoiturages disponibles avec les détails s'affichera, comme illustré dans la figure 46.

Pour rejoindre un covoiturage, il suffit de cliquer sur "Demande de réservation" comme illustré dans la figure 47. Une demande sera alors envoyée au responsable du covoiturage et sera placée en statut "en attente". Si une place est disponible et que les conditions sont acceptées, la demande sera validée, comme illustré dans la figure 55. L'utilisateur a également la possibilité de négocier le prix et de discuter avec le conducteur lorsqu'il envoie une demande pour rejoindre le covoiturage. Il sera automatiquement ajouté à la liste de discussions, comme illustré dans la figure 48. Ensuite, il sera redirigé vers une interface de messagerie où il pourra discuter des détails .

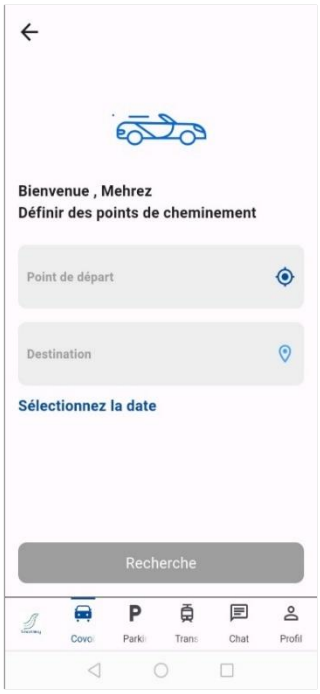


Figure 45 : Interface chercher covoiturage

Figure 46 : liste de covoiturages

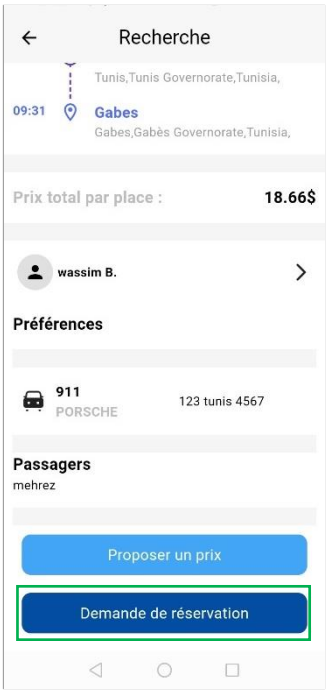


Figure 47 : Interface Détails covoiturage

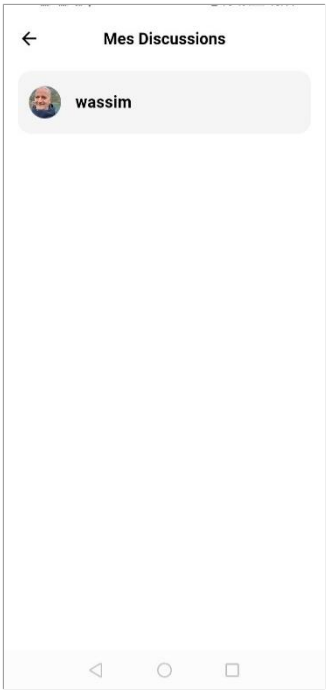


Figure 48: Interface messagerie

III.9.2 Interface Créer covoiturage

Pour créer un covoiturage, il suffit de suivre les étapes illustrées dans les figures 49 à 54 et de saisir des informations valides. Si tout se passe bien, une fenêtre contextuelle de succès apparaîtra et le covoiturage sera créé avec succès

Figure 49 : sélectionner choix ' créer covoiturage'

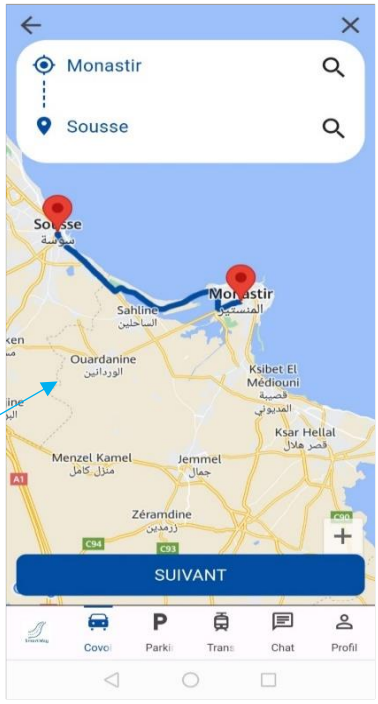


Figure 50 : choisir le point départ et destination

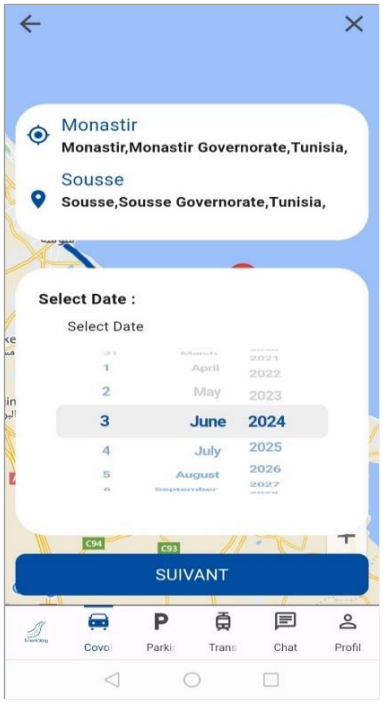


Figure 51 : sélecter date

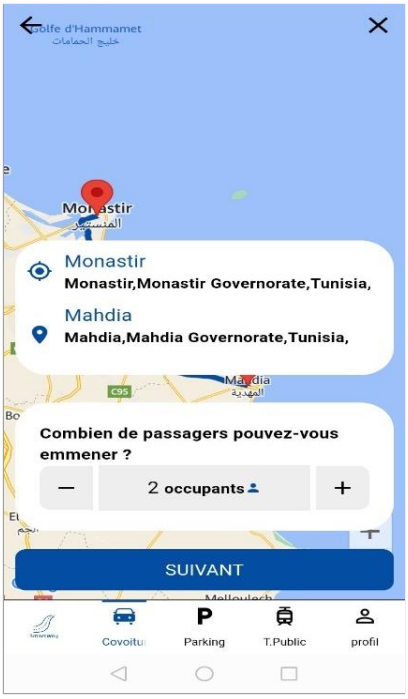


Figure 52 : Sélecter nombre de participant

Figure 53 : sélecter voiture

Figure 54 : sélecter prix

Si tout est validé, il suffit de vérifier les données et de cliquer sur le bouton "Publier". Pour consulter la liste des trajets, l'utilisateur doit cliquer sur l'icône de carte en haut à droite de la figure 49. Une interface s'ouvrira contenant tous les trajets publiés, comme illustré dans la figure 55.

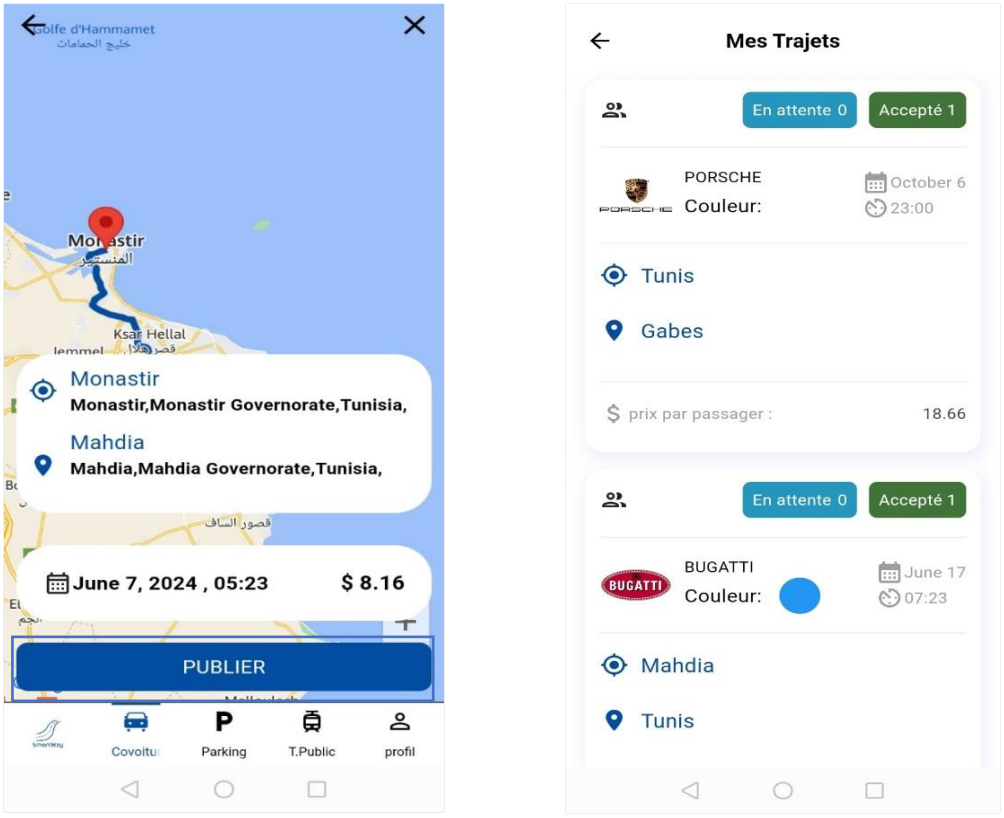


Figure 55 : Vérifications des données covoiturage

II.10 Interface tableau de bord administrateur

Cette interface, destinée à l'administrateur, contient quatre options dans la barre latérale : Gérer les utilisateurs, Gérer les parkings, Envoyer des notifications et Gérer les réclamations des utilisateurs. Ces fonctionnalités permettent à l'administrateur de gérer efficacement les utilisateurs, les parkings, les notifications et les réclamations au sein de l'application. La figure 56 illustre l'interface correspondante.

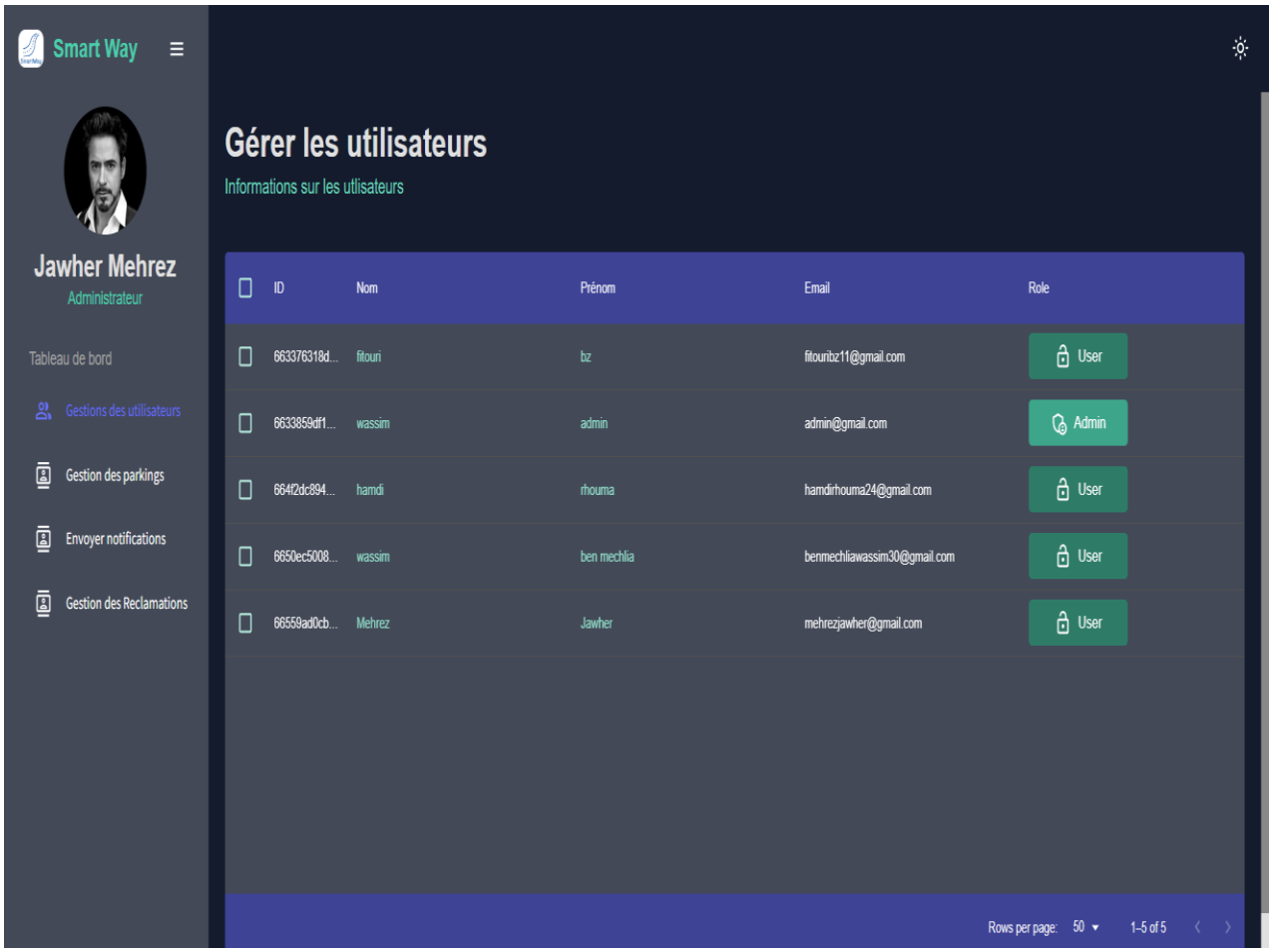


Figure 56 : Interface tableau de bord administrateur

III.11 Interface gérer les utilisateurs

L'administrateur peut consulter tous les utilisateurs et changer le rôle d'un utilisateur si nécessaire comme le montre la figure 57.

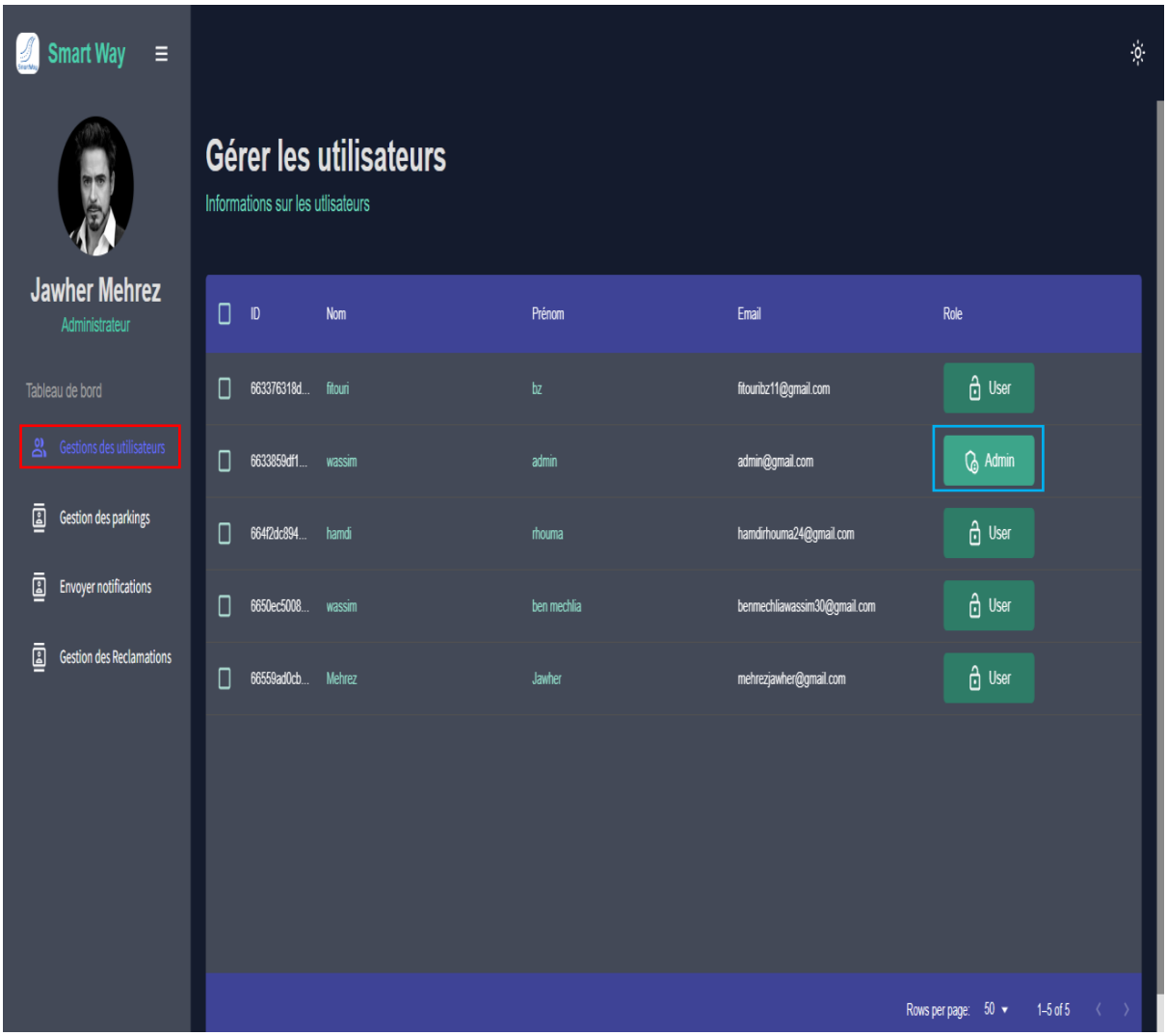


Figure 57 : Gérer les utilisateurs

III.12 Interface gérer les parkings

L'administrateur a la possibilité de visualiser la liste complète des parkings disponibles. De plus, il peut enrichir cette liste en ajoutant un nouveau parking via le bouton "Ajouter parking", en renseignant les informations requises puis en validant. En outre, il est en mesure de supprimer un parking existant en cliquant sur le bouton "Supprimer". Toutes ces fonctionnalités sont démontrées dans la figure 58.

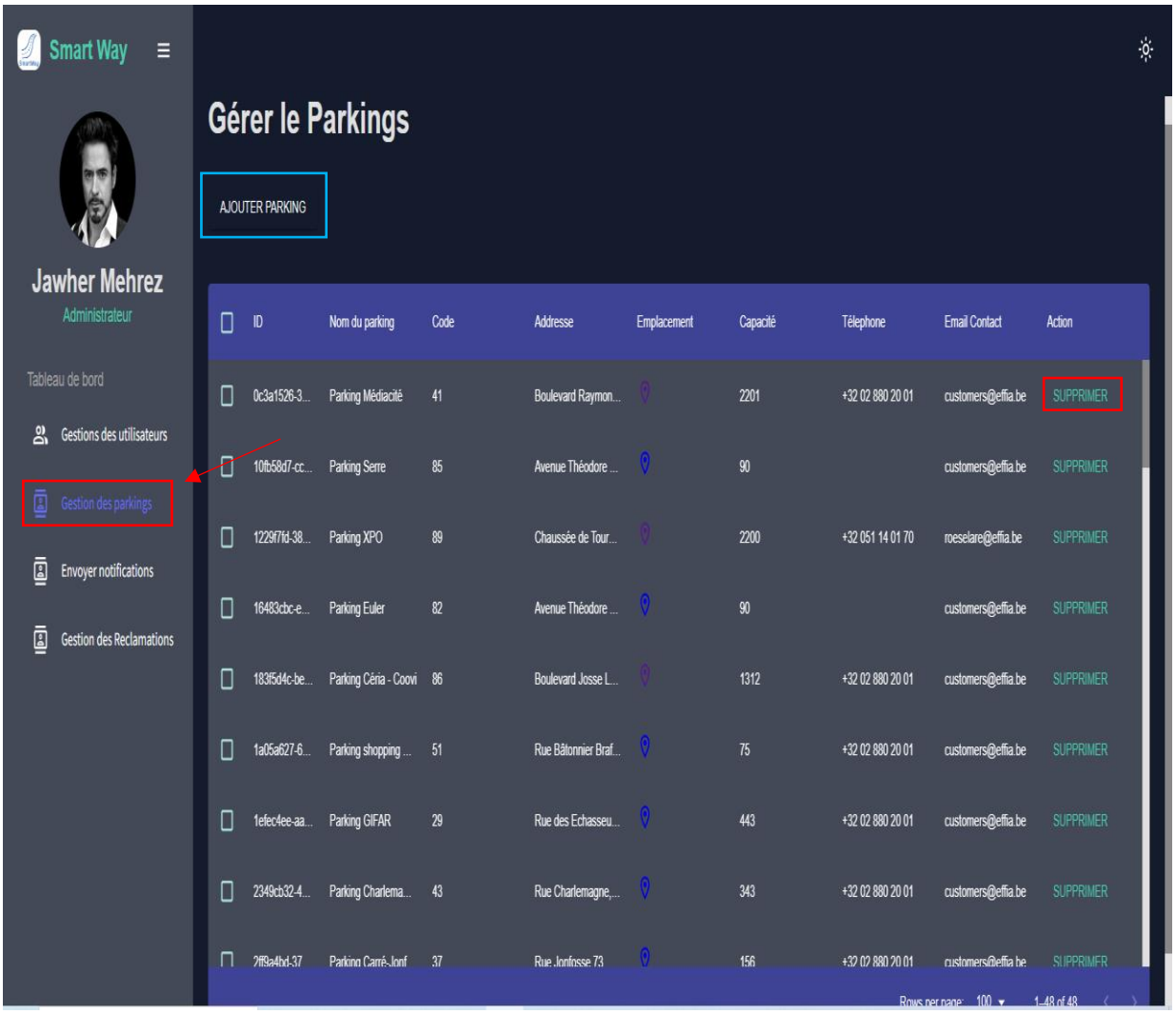


Figure 58 : Interface gérer les parkings

III.13 Interface envoyer des notifications

L'administrateur peut envoyer des notifications en remplissant les champs nécessaires et en cliquant sur "Envoyer", comme illustré dans la figure 59.

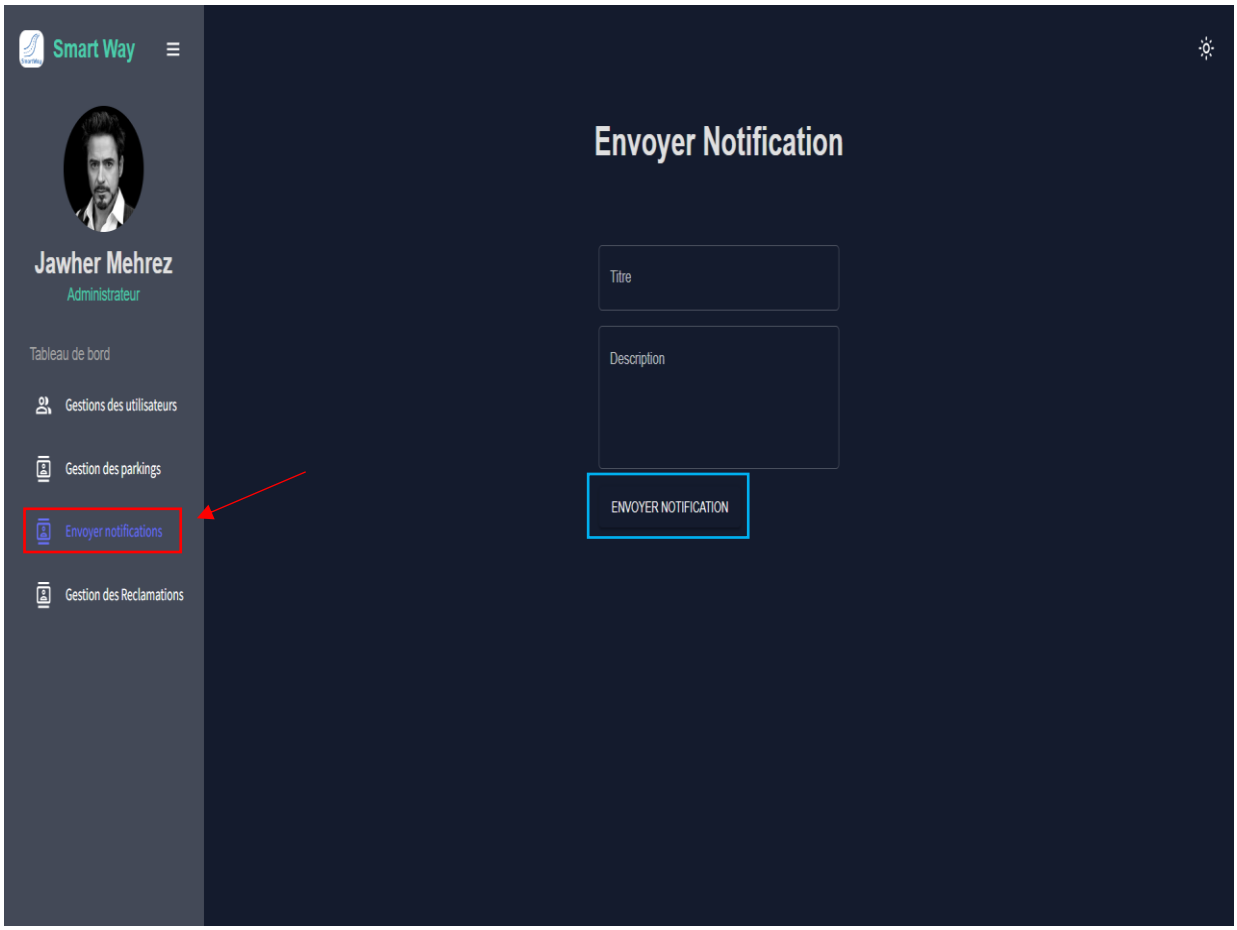


Figure 59 : Interface envoyer des notifications

III.14 Interface gérer les réclamations

L'administrateur peut consulter les réclamations émises par les utilisateurs. De plus, il a le pouvoir de supprimer ces réclamations en cas de nécessité. Toutes ces fonctionnalités sont illustrées dans la figure 60.

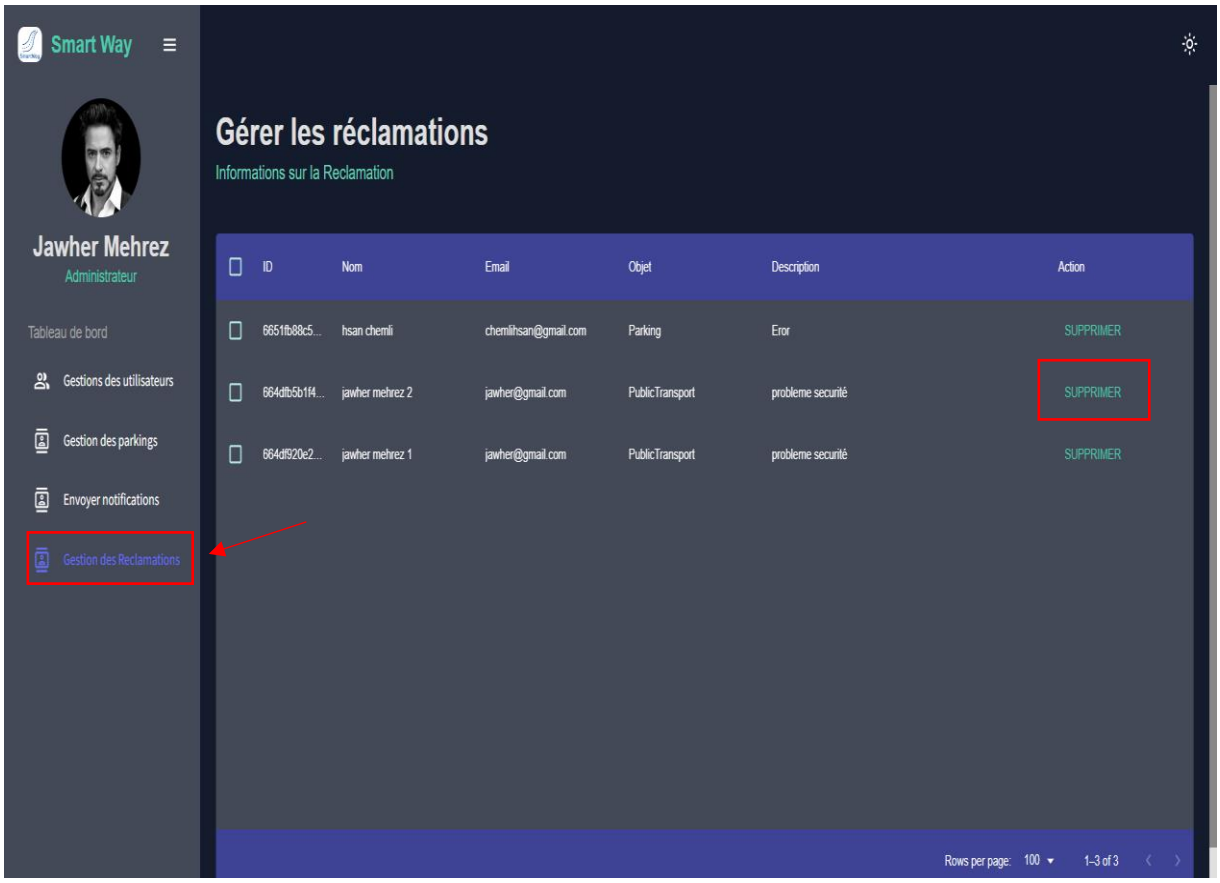


Figure 60 : Interface gérer les réclamations

Conclusion

Dans ce dernier chapitre, nous avons mis en avant l'environnement de travail en mentionnant les technologies ainsi que les outils auxquels nous avons eu recours durant la réalisation de notre projet. En outre, nous avons exposé les différentes interfaces réalisées tout en détaillant toutes les fonctionnalités développées par les utilisateurs de notre plateforme.

Conclusion générale et perspectives

Dans l’urbanisme moderne, la gestion efficace du trafic urbain constitue un enjeu majeur. Les méthodes traditionnelles de surveillance et de gestion de la circulation sont souvent inefficaces, créant des problèmes tant pour les gestionnaires de trafic que pour les citadins. Face à cette réalité, nous avons développé une application mobile dédiée à la gestion du trafic urbain, offrant ainsi une solution plus adaptée et efficiente.

Pour se faire, nous avons tout d’abord défini notre problématique. Ensuite, nous avons fait une étude de l’existant pour pouvoir fixer nos objectifs. Dans une deuxième étape, nous avons identifié nos besoins et les modéliser en se basant sur des diagrammes de cas d’utilisation. Ensuite, nous avons entamé la conception de notre projet et nous avons choisi notre architecture. En effet, nous avons adopté une architecture solide qui est la clean architecture qui assuré la qualité du code et la fiabilité du système. Enfin, nous avons développé notre application en mettant l'accent sur l'expérience utilisateur et la performance.

Notre dévouement envers l'excellence dans le développement mobile reste central dans notre vision pour l'avenir. Nous projetons d'intégrer l'intelligence artificielle pour mieux répondre aux besoins évolutifs de nos utilisateurs. En outre, nous prévoyons d'ajouter des fonctionnalités de paiement et de navigation en temps réel afin d'offrir une expérience utilisateur encore plus immersive. Parallèlement, nous nous engageons à rendre le module de covoiturage plus convivial en ajoutant une liste pour les demandes de covoiturage, tout en renforçant la sécurité des données de nos utilisateurs.

En somme, ce projet incarne notre engagement envers l'amélioration de l'environnement urbain et témoigne de notre vision d'un avenir où la technologie joue un rôle central dans la création de communautés urbaines prospères et harmonieuses.

Annexes

ID	User Story	ID Task	Task	Priority
1	En tant qu'utilisateur, je dois m'authentifier pour pouvoir accéder à mon espace	1.1	Création de l'interface d'authentification	8
		1.2	Création des APIs nécessaires pour l'inscription.	13
		1.3	Intégration et test des API crées.	13
2	En tant qu'utilisateur, je dois m'inscrire à l'application afin d'avoir un compte	2.1	Création de l'interface inscription	8
		2.2	Ajouter Flexible Box Layout pour que l'application s'adapte sur tout type d'écran.	8
		2.3	Création des APIs nécessaires	13
		2.4	Intégration et test des API crées.	13
3	En tant qu'utilisateur, je veux réinitialiser mon mot de passe en cas d'oubli	3.1	Création de l'interface de réinitialisation du mot de passe	8
		3.2	Création des APIs nécessaires pour réinitialiser le mot de passe	13
		3.3	Intégration et test des API crées.	13
4	En tant que qu'utilisateur, je veux gérer mon profil	4.1	Préparer le design de l'interface « Profil »	8
		4.2	Création de l'interface graphique	8
		4.3	Création des APIs nécessaires	13
		4.4	Intégration et test des API crées.	13
		4.5	Implémenter le code nécessaire pour la gestion de profil.	20
5	En tant que qu'utilisateur, je peux	5.1	Préparer le design de l'interface « List des parkings »	8
		5.2	Création de l'interface graphique.	13

	Consulter la liste des parkings	5.3	Ajouter Flexible Box Layout pour que l'application s'adapte sur tout type d'écran.	8
		5.4	Création des APIs nécessaires pour la liste de parkings	13
		5.5	Intégration et test des API crées.	13
		5.6	Implémenter le code nécessaire pour consulter la liste des parkings	20
6	En tant qu'utilisateur, je peux effectuer ou annuler une réservation	6.1	Préparer le design	8
		6.2	Création de l'interface graphique.	13
		6.3	Création des APIs nécessaires	13
		6.4	Intégration et test des API crées.	13
		6.5	Implémenter le code nécessaire	20
7	En tant qu'utilisateur, je peux consulter la liste de réservations effectuées	7.1	Création de l'interface graphique.	8
		7.2	Création des APIs nécessaires	13
		7.3	Intégration et test des API crées.	13
		7.4	Implémenter le code nécessaire pour consulter la liste de réservations effectuées	20
8	En tant qu'utilisateur, je peux créer ou rechercher un covoiturage	8.1	Création de l'interface graphique.	8
		8.2	Création des APIs nécessaires pour créer un covoiturage	13
		8.3	Création des APIs nécessaires pour rechercher un covoiturage	13
		8.4	Intégration et test des API crées.	13
		8.5	Implémenter le code nécessaire	20
9	En tant qu'utilisateur, je peux discuter avec la responsable de covoiturage	9.1	Création de l'interface graphique.	8
		9.2	Création des APIs nécessaires pour la discussion	13
		9.3	Intégration et test des API crées.	13
		9.4	Implémenter le code nécessaire	20

10	En tant qu'utilisateur, je veux consulter ma liste de voitures	10.1	Préparer le design de l'interface « Liste de voitures »	8
		10.2	Création de l'interface graphique.	8
		10.3	Création des APIs nécessaires pour consulter la liste des voitures	13
		10.4	Intégration et test des API crées.	8
		10.5	Implémenter le code nécessaire	13
11	En tant qu'utilisateur, je peux consulter les horaires et les retards des transports public	11.1	Préparer le design de l'interface « consulter les horaires des transports public »	8
		11.2	Préparer le design de l'interface « consulter les retards des transports public »	13
		11.3	Création de l'interface graphique.	13
		11.4	Création des APIs nécessaires	13
		11.5	Intégration et test des API crées.	13
		11.6	Implémenter le code nécessaire	20
12	En tant qu'utilisateur, je veux consulter les alertes el les incidents en temps réels	12.1	Préparer le design de l'interface « Consulter les alertes el les incidents en temps réels »	8
		12.2	Création de l'interface graphique	8
		12.3	Création des APIs nécessaires	8
		12.4	Intégration et test des API crées.	8
		12.5	Implémenter le code nécessaire	13
13	En tant qu'utilisateur, je veux recevoir des notifications et envoyer des réclamations	13.1	Création de l'interface graphique	8
		13.2	Création des APIs nécessaires	8
		13.3	Intégration et test des API crées.	13
		13.4	Implémenter le code nécessaire pour recevoir des notifications	13
		13.5	Implémenter le code nécessaire pour envoyer des réclamations	13
14		14.1	Préparer le design de l'interface	8

	En tant qu'administrateur, je veux voir ou modifier des utilisateurs	14.2	Création de l'interface graphique	8
		14.3	Création des APIs nécessaires	8
		14.4	Intégration et test des API crées.	13
		14.5	Implémenter le code nécessaire pour voir et modifier des utilisateurs	13
15	En tant qu'administrateur, je veux consulter les réclamations envoyées par les utilisateurs	15.1	Préparer le design de l'interface	8
		15.2	Création de l'interface graphique	8
		15.3	Création des APIs nécessaires	13
		15.4	Intégration et test des API crées.	13
		15.5	Implémenter le code nécessaire	13
16	En tant qu'administrateur, je veux consulter ou modifier les horaires des transports public	16.1	Préparer le design de l'interface « Consulter les horaires des transports public »	8
		16.2	Préparer le design de l'interface	8
		16.3	Création de l'interface graphique	8
		16.4	Création des APIs nécessaires	13
		16.5	Intégration et test des API crées.	13
		16.6	Implémenter le code nécessaire	13
17	En tant qu'administrateur, je veux consulter ou modifier la liste des parkings disponibles	17.1	Préparer le design de l'interface	8
		17.2	Création de l'interface graphique	8
		17.3	Création des APIs nécessaires	13
		17.4	Intégration et test des API crées.	13
		17.5	Implémenter le code nécessaire	13
18	En tant qu'administrateur, je veux envoyer des notifications a les utilisateurs	18.1	Préparer le design de l'interface « Envoyer des notifications »	8
		18.2	Création de l'interface graphique	8
		18.3	Création des APIs nécessaires	13
		18.4	Intégration et test des API crées.	13
		18.5	Implémenter le code nécessaire pour envoyer des notifications	13

19	Compréhension & Collecte de données	19.1	En tant que Data scientiste, je dois comprendre les besoins exacts de client	30
		19.2	En tant que Data scientiste, je dois collecter les données initiales et vérifier leurs qualités	30
		19.3	En tant que Data scientiste, je dois explorer des données et vérifier leurs qualités	30

Webographie

[1] https://vivianet.com/ , consulté le 01-02-2024

[2] https://www.linkedin.com/company/vivia-clinical-net/mycompany/verification/ , consulté le 03-02-2024

[3] https://www.parkeasy.co/ , consulté le 04-07-2023

[4] https://split.tn/ , consulté le 05-02-2024

[5] https://triphub.transport.brussel&hl=ln&gl=US, consulté le 06-02-2024

[6] https://blog-gestion-de-projet.com/gestion-de-projet/outils-methodes-gestion-projet/share , consulté le 13-02-2024

[7] https://www.wrike.com/fr/project-management-guide/faq/quest-ce-que-la-methodologie-agile-en-gestion-de-projet/ , consulté le 18-02-2024

[8] https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1443834-scrum maitriser-le-framework-star-des-methodes-agiles/ , consulté le 20-02-2023

[9] https://www.planzone.fr/blog/quest-ce-que-la-methodologie-agile, consulté le 15-07-2023

[10] https://yeswedev.bzh/blog/les-methodes-agiles-un-guide-complet-pour-choisir-la-meilleure-approche-pour-votre-projet , consulté le 16-02-2024

[11] https://www.gantt.com/fr/ , consulté le 19-02-2024

[12] https://www.appvizer.fr/magazine/operations/gestion-de-projet/specifications-fonctionnelles , consulté le 02-03-2024

[13] https://www.advaloris.ch/gestion-de-projet/definir-besoins-gestion-projet, consulté le 12-03-2024

[14] https://creately.com/blog/fr/diagrammes/tutoriel-sur-les-diagrammes-de-cas-dutilisation/ , consulté le 19-03-2024

[15] https://cian.developpez.com/uml2/tutoriel/sequence/ , consulté le 21-03-2024

[16] https://mathieux51.gitbooks.io/howto/content/II.cahier-des-charges/2.expression-des-besoins/2.besoins-non-fonctionnels.html , consulté le 25-03-2024

[17] https://www.ionos.fr/digitalguide/sites-internet/developpement-web/uml-un-langage-de-modelisation-pour-la-programmation-orientee-objet/, consulté le 17-03-2024

[18] https://www.softfluent.fr/blog/architecture-logicielle-pour-application/, consulté le 17-02-2024

[19] https://www.dhiwise.com/post/mastering-the-art-of-clean-architecture-in-flutter/, consulté le 25-02-2024

[20] https://www.softfluent.fr/blog/architecture-logicielle-et-modeles-de-conception/ , consulté le 23-02-2024

[21] https://appmaster.io/fr/blog/modeles-architecturaux-mvc-mvp-et-mvvm , consulté le 23-02-2024

[22] https://www.dhiwise.com/post/mastering-the-art-of-clean-architecture-in-flutter/, consulté le 26-02-2024

[23] https://sparxsystems.fr/resources/tutorials/uml/dynamic-model.html , consulté le 22-03-2024

[23] https://www.lucidchart.com/pages/fr/diagramme-de-deploiement-uml , consulté le 25-05-2024

[25] https://flutter.dev/ , consulté le 10-03-2024

[26] https://www.pngwing.com/en/search?q=node+js, consulté le 13-03-2024

[27] https://welovedevs.com/fr/articles/express-js/ , consulté le 15-03-2024

[28] https://datascientest.com/mongodb, consulté le 17-03-2024

[29] https://fr.legacy.reactjs.org/, consulté le 20-03-2024

[30] https://dart.dev/, consulté le 11-03-2024

[31] https://framalibre.org/content/visual-studio-code , consulté le 28-07-2024

[32] https://blog.webnet.fr/presentation-de-postman-outil-multifonction-pour-api-web, consulté le 14-03-2024

[33] https://www.appvizer.fr/operations/business-process/lucidchart , consulté le 20-03-2024

[34] https://www.lemoniteur.fr/article/l-urbanisme-n-est-pas-la-simple-extension-de-architecture.1045154 , consulté le 01-02-2024