

« TITRE : DWWM

Dossier de projet

Réalisation de l'application web et web mobile



Aïmen TRIFI

Table des matières

1. Introduction.....	4
• 1.1. Présentation personnelle	4
• 1.2. Résumé du projet "Parks".....	4
• 1.3. Compétences du référentiel couvertes.....	4
2. Analyse et Spécifications Fonctionnelles.....	5
• 2.1. Objectifs et Mission du Projet.....	5
• 2.2. Utilisateurs Cibles.....	5
• 2.3. Gestion de projet : Méthodologie Agile avec GitHub Projects.....	5
• 2.4. Moodboard et Identité Visuelle.....	5
• 2.5. Diagramme des Cas d'Utilisation (Use Case)	6
• 2.6. Liste des Fonctionnalités Attendues (MVP)	6
3. Conception Technique et Architecture.....	6
• 3.1. Choix de l'Architecture.....	6
• 3.2. Outils et Stack Technique.....	6
• 3.3. Modèle Conceptuel de Données (MCD)	6
• 3.4. Modèle Physique de Données (MPD)	13
• 3.5. Arborescence du Projet.....	13
4. Développement et Implémentation (Structuré par Blocs RNCP) .	15
• 4.1. Bloc BC01 — Développer la partie front-end d'une application sécurisée.....	15
• 4.2. Bloc BC02 — Développer la partie back-end d'une application sécurisée.....	15

5. Déploiement.....	15
• 5.1. Préparation de l'Application pour la Production.....	15
• 5.2. Procédure de Déploiement sur un Serveur Plesk.....	16
6. Bilan et Conclusion.....	17
• 6.1. Difficultés Rencontrées et Solutions Apportées.....	17
• 6.2. Axes d'Amélioration Possibles.....	19
• 6.3. Conclusion Personnelle.....	19
7. Annexes.....	20

1. Introduction

1.1. Présentation personnelle

Je me nomme Aïmen TRIFI, quarante-quatre ans. Autodidacte de l'informatique, j'ai choisi l'école La Plateforme pour cadrer mes acquis, les actualiser aux bonnes pratiques et me spécialiser en développement web.

Ce domaine m'a convaincu par l'alliance de la rigueur logique, de la créativité et de l'impact concret pour l'utilisateur.

Au fil de mon parcours, d'abord comme Technicien informatique puis Développeur web, j'ai travaillé de l'interface aux bases de données et consolidé des compétences transversales en conception, sécurité, performance et déploiement, avec une pratique d'API REST et des méthodes Agile.

Dans cette dynamique s'inscrit "Parks", une application menée de A à Z sur une stack moderne, conçue pour démontrer ma capacité à concevoir, réaliser et livrer un produit abouti.

1.2. Résumé du projet "Parks"

L'idée de "Parks" est née d'une expérience personnelle récurrente : en me rendant chaque jour en voiture à mon centre de formation situé en centre-ville, je perdais un temps considérable à chercher une place pour me garer. Parallèlement, je disposais d'un garage personnel qui restait inoccupé toute la journée. J'ai réalisé que de nombreuses personnes devaient être dans la même situation.

De ce constat est né "Parks" : une plateforme web qui connecte les propriétaires de places de parking inutilisées avec des automobilistes cherchant à se garer simplement. L'application est une Single Page Application (SPA) développée avec le framework **Laravel 12** pour le back-end et la bibliothèque **React** (avec TypeScript) pour le front-end. La communication entre les deux est assurée par **Inertia.js**, permettant une expérience utilisateur fluide et moderne.

1.3. Compétences du référentiel couvertes

Ce projet couvre l'ensemble des compétences des deux activités-types du référentiel RNCP37674 :

- **Bloc BC01** : Développer la partie front-end d'une application web ou web mobile sécurisée.

- **Bloc BC02 :** Développer la partie back-end d'une application web ou web mobile sécurisée.

2. Analyse et Spécifications Fonctionnelles

2.1. Objectifs et Mission du Projet

- **Pour les automobilistes :** Réduire le stress et le temps perdu à chercher une place.
- **Pour les propriétaires :** Offrir une source de revenus passive en monétisant un espace vacant.

2.2. Utilisateurs Cibles

- **Les propriétaires :** Particuliers ou entreprises disposant de places de parking inoccupées.
- **Les clients :** Travailleurs, résidents ou visiteurs cherchant une solution de stationnement.

2.3. Gestion de projet : Méthodologie Agile avec GitHub Projects

Pour organiser le développement du projet, j'ai adopté une méthodologie **Agile** en utilisant l'outil **GitHub Projects**.

J'ai créé un tableau Kanban pour visualiser l'avancement des tâches, que j'ai organisées en plusieurs sprints. Chaque sprint correspondait à un objectif précis (ex: "Sprint 1 : Mise en place de l'authentification", "Sprint 2 : Développement de la recherche de parking").

Cette méthode m'a permis de prioriser les fonctionnalités, de suivre ma progression de manière structurée et de simuler un environnement de travail collaboratif. (*Voir Annexe 1, Figure 1.4*).

2.4. Moodboard et Identité Visuelle

Avant la phase de maquettage, j'ai réalisé un **moodboard** pour définir l'identité visuelle et l'ambiance de l'application "Parks".

L'objectif était de créer une atmosphère à la fois moderne, rassurante et simple d'utilisation. J'ai sélectionné une palette de couleurs, des typographies et des visuels qui évoquent la confiance, la technologie et la fluidité. (*Voir Annexe 2, Figure 2.1*)

2.5. Diagramme des Cas d'Utilisation (Use Case)

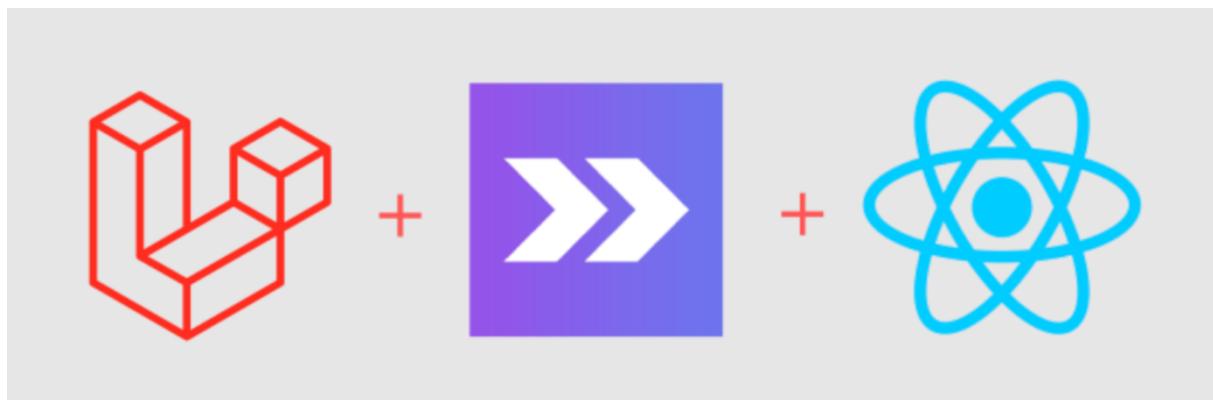
Le diagramme suivant identifie les principaux acteurs et leurs interactions avec le système. (Voir Annexe 1, Figure 1.2)

2.6. Liste des Fonctionnalités Attendues (MVP)

- **Gestion des utilisateurs** : Inscription, connexion, gestion de profil.
- **Pour les clients** : Recherche de parkings, réservation en ligne.
- **Pour les propriétaires** : Mise en ligne d'un parking, gestion des disponibilités.

3. Conception Technique et Architecture

3.1. Choix de l'architecture



J'ai opté pour une architecture monolithique moderne. L'objectif était de combiner la robustesse et la rapidité de développement d'un back-end traditionnel avec la fluidité et l'interactivité d'un front-end de type SPA (Single Page Application).

La bibliothèque Inertia.js a été le choix central pour servir de pont entre ces deux mondes, permettant à Laravel de contrôler le routage tout en offrant une expérience utilisateur sans rechargement de page.

3.2. Outils et stack technique

Chaque outil a été sélectionné pour sa pertinence, sa performance et sa capacité à s'intégrer dans un écosystème de développement moderne et efficace.

- **Langages :**

php 8.2

- **PHP 8.2** : Choisi pour sa maturité et ses performances. En tant que langage principal du framework Laravel, il offre une base solide et fiable pour toute la logique métier côté serveur.



- **TypeScript** : Préféré à JavaScript standard pour le front-end afin d'apporter un typage statique au code. Cela rend l'application plus robuste, prévient de nombreuses erreurs avant même l'exécution, et facilite grandement la maintenance et la collaboration sur des projets de plus grande envergure.



- **SQL** : Langage incontournable pour interagir avec notre base de données relationnelle MySQL, principalement utilisé à travers l'ORM Eloquent.

- **Backend :**



- **Framework Laravel 12 :** J'ai choisi Laravel pour son écosystème complet, sa convention de code claire (MVC) et sa popularité dans le monde professionnel. Sa structure "batteries included" permet de développer rapidement des fonctionnalités complexes et sécurisées.



- **ORM Eloquent :** Intégré à Laravel, Eloquent a été un choix évident pour simplifier et sécuriser les interactions avec la base de données. Il abstrait les requêtes SQL et protège nativement contre les injections SQL.

- **Authentification Laravel Breeze** : Pour la gestion des utilisateurs, j'ai opté pour Breeze qui fournit une implémentation complète et sécurisée de l'authentification (inscription, connexion, réinitialisation de mot de passe) dès le départ, me permettant de me concentrer sur les fonctionnalités métier.



- **Gestion des dépendances Composer** : Outil standard de l'écosystème PHP, Composer a été utilisé pour gérer toutes les bibliothèques et dépendances du projet côté serveur.
 - **Frontend :**



- **Bibliothèque React** : J'ai choisi React pour son approche basée sur les composants, qui permet de construire des interfaces utilisateur modulaires et réutilisables. Sa popularité et son vaste écosystème en font un choix de premier ordre pour les applications web modernes.



- **Framework CSS Tailwind CSS** : Préféré à d'autres frameworks comme Bootstrap, Tailwind CSS a été choisi pour son approche "utility-first" qui offre une liberté de design totale tout en maintenant un système de conception cohérent.



- **Outil de build Vite.js** : Intégré par défaut avec les nouvelles versions de Laravel, Vite.js a été utilisé pour sa rapidité exceptionnelle en développement, notamment grâce à son système de recharge à chaud (Hot Module Replacement) qui améliore considérablement la productivité.

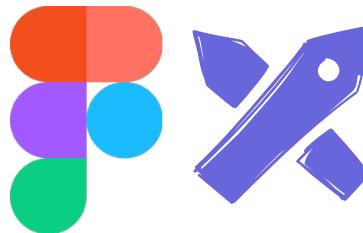


- **Gestion des dépendances NPM** : Outil standard de l'écosystème Node.js, NPM a été utilisé pour gérer toutes les bibliothèques et dépendances du projet côté client.
- **Base de donnée :**



- **SGBD MySQL** : J'ai choisi MySQL pour sa fiabilité, ses performances et sa très large adoption dans l'industrie, ce qui en fait un standard pour les applications web basées sur PHP.

- **Conception & Gestion de projet :**



- **Wireframes (Figma) & Diagrammes (Excalidraw)** : Ces outils ont été choisis pour leur simplicité et leur efficacité à traduire des idées en spécifications visuelles et techniques claires (wireframes, MCD, MLD, cas d'utilisation).

The screenshot shows a project management application with a dark theme. At the top, there's a navigation bar with links for 'Backlog', 'Priority board', 'Team items', 'Roadmap', 'In review', 'My items', 'Sprint2', 'Sprint_current', and 'New view'. Below the navigation is a search bar with placeholder text 'Filter by keyword or by field' and buttons for 'Discard' and 'Save'.

The main area is divided into three columns representing different stages of the project:

- Backlog:** Contains 6 items, 5 of which are in progress. One item is described as 'This item hasn't been started'. Examples include 'parks #11 MLD' and 'parks #20 Pages CRUD'.
- In progress:** Contains 4 items. One item is described as 'This is actively being worked on'. Examples include 'parks #19 Pages Login , Register, Mot de passe oublié' and 'parks #8 Faire un prototype'.
- Done:** Contains 16 items, all of which have been completed. Examples include 'parks #18 Homepage Guest', 'parks #17 Factories', and 'parks #16 Seeders'.

At the bottom of each column, there are buttons for '+ Add item'.

- **Gestion de projet Agile (GitHub Projects)** : J'ai utilisé GitHub Projects pour sa parfaite intégration avec mon dépôt de code. Cela m'a permis de lier directement les tâches de mon tableau Kanban aux commits et aux branches, assurant une traçabilité complète de l'avancement du projet.
- **Environnement de Développement & Déploiement**



- **IDE PHPStorm** : Choisi pour ses fonctionnalités avancées dédiées au développement avec PHP et Laravel, ainsi que pour son excellent support de l'écosystème front-end (TypeScript, React).



- **Gestion de version (Git & GitHub)** : Indispensable pour tout projet de développement, Git a permis de versionner le code, tandis que GitHub a servi de plateforme de sauvegarde et de collaboration.



- **Serveur local MAMP (pour MySQL)** : J'ai utilisé MAMP uniquement pour lancer le service de base de données MySQL de manière simple et rapide sur mon poste de travail.



- **Panneau d'hébergement Plesk :** L'utilisation de Plesk, fourni par le centre de formation, m'a permis de me familiariser avec un environnement d'hébergement web professionnel standard, utilisé par de nombreuses entreprises.

3.3. Modèle Conceptuel de Données (MCD)

Le MCD formalise la structure de la base de données et les relations entre les entités. (Voir Annexe 1, Figure 1.3)

3.4. Modèle Physique de Données (MPD)

Le MPD est la représentation finale de la base de données, spécifiant les types de données exacts (VARCHAR(255), INT, etc.). Ce modèle est le plan direct utilisé pour écrire les fichiers de migration Laravel. (Voir Annexe 1, Figure 1.4)

3.5. Arborescence du Projet

Le projet "Parks" respecte et tire parti de la structure de dossiers standard fournie par Laravel, qui favorise une organisation claire et la séparation des responsabilités (SoC), conformément à l'architecture MVC.

Voici une description des répertoires les plus importants pour le projet :

- **app/** : C'est le cœur de l'application.
 - **app/Http/Controllers/** : Contient les contrôleurs qui gèrent la logique métier, comme UserController.php ou BookingController.php.
 - **app/Models/** : Contient les modèles Eloquent qui interagissent avec la base de données, comme User.php et Parking.php.

- **database/** : Contient tout ce qui est relatif à la base de données.
 - **database/migrations/** : Contient les fichiers de migration qui décrivent la structure des tables (create_users_table, create_parkings_table, etc.).
 - **database/seeds/** : Contient les classes pour peupler la base de données avec des données de test (UserSeeder, ParkingSeeder, etc.).
- **resources/** : Contient les fichiers bruts du front-end.
 - **resources/js/** : C'est le répertoire principal pour le code front-end. Il contient les composants React (.tsx), les pages (Pages/), les layouts et les hooks.
 - **resources/css/** : Contient la feuille de style principale app.css qui configure Tailwind CSS.
- **routes/** : Contient les fichiers de définition des routes.
 - **routes/web.php** : Définit les routes principales de l'application qui sont accessibles via un navigateur.
 - **routes/auth.php** : Définit les routes spécifiques à l'authentification (connexion, inscription), fournies par Laravel Breeze.
- **public/** : C'est le seul dossier accessible publiquement. Il contient le point d'entrée de l'application (index.php) ainsi que les assets compilés (CSS, JS) par Vite.js.

Cette structure standardisée facilite grandement la maintenance, la collaboration et l'évolution du projet.

4. Développement et Implémentation (Blocs RNCP)

4.1. Bloc BC01 — Développer la partie front-end d'une application sécurisée

- **Installer et configurer l'environnement de travail** : Mise en place de l'environnement avec Laravel, Breeze, React, Inertia, Vite.js, et Git.
- **Maquetter des interfaces utilisateur** : Création des Use Cases, User Stories et wireframes.
- **Réaliser des interfaces utilisateur statiques** : Intégration en composants React avec TypeScript et Tailwind CSS.
- **Développer la partie dynamique des interfaces** : Utilisation des hooks de React et d'Inertia pour la gestion des formulaires et de l'interactivité.

4.2. Bloc BC02 — Développer la partie back-end d'une application sécurisée

- **Mettre en place une base de données relationnelle** : Conception (MCD, MPD) et implémentation avec Laravel Migrations.
- **Développer les composants d'accès aux données** : Utilisation de l'ORM Eloquent pour créer les modèles et définir les relations.
- **Développer les composants métier côté serveur** : Implémentation de la logique MVC avec les contrôleurs, la validation des requêtes et l'utilisation de Laravel Breeze.
- **Documenter le déploiement** : Préparation de l'application et rédaction de la procédure de déploiement sur un serveur Plesk.

5. Déploiement

Le déploiement est l'étape finale qui consiste à rendre l'application "Parks" accessible au public sur internet. **À ce stade, le projet n'a pas encore été déployé en production, car le code actuel est un Proof of Concept (POC) qui n'est pas encore finalisé.**

Le déploiement est cependant prévu pour une version future. En préparation de cette étape, l'application a été structurée pour être compatible avec un environnement de production, et la procédure complète de mise en ligne a été documentée. Cette documentation, rédigée dans le README.md du projet, détaille les étapes pour un serveur d'hébergement standard équipé d'un panneau de contrôle Plesk, un environnement fourni par mon centre de formation.

5.1. Préparation de l'Application pour la Production

Avant de transférer le code sur le serveur distant, il est indispensable de préparer l'application pour qu'elle soit optimisée, performante et sécurisée.

- **Compilation des assets front-end :** La première action consiste à compiler l'ensemble des fichiers React/TypeScript et Tailwind CSS. Pour cela, j'ai utilisé la commande npm run build. Cette commande fait appel à **Vite.js** pour transpiler, minifier et packager tous les assets front-end en fichiers statiques (JavaScript et CSS) optimisés. Le résultat est généré dans le répertoire public/build, prêt à être servi par le serveur web.
- **Optimisation des dépendances back-end :** Côté serveur, il est crucial de n'installer que les dépendances nécessaires à la production. Pour cela, j'ai utilisé la commande composer install --optimize-autoloader --no-dev. L'option --no-dev exclut les paquets de développement (comme les outils de débogage et de test), et --optimize-autoloader améliore les performances de Laravel en créant une "map" optimisée des classes du projet.

5.2. Procédure de Déploiement sur un Serveur Plesk

J'ai rédigé un guide de déploiement détaillé dans le fichier README.md du projet pour assurer un processus de mise en ligne fiable et reproductible.

1. Préparation de l'environnement Plesk :

- Création d'un sous-domaine dédié au projet (ex: parks.trifi.fr).
- Création d'une base de données MySQL via l'interface Plesk et récupération des identifiants (nom de la base, utilisateur, mot de passe).

2. Transfert des fichiers :

- Téléversement de l'ensemble des fichiers du projet (à l'exception des dossiers **node_modules** et **vendor**) sur le serveur, via le client FTP FileZilla.

3. Configuration sur le serveur :

- Connexion au serveur via le terminal **SSH** fourni par Plesk.
- Exécution de **composer install --optimize-autoloader --no-dev** pour installer les dépendances back-end directement sur le serveur.
- Copie du fichier **.env.example** en **.env** et configuration avec les informations de la base de données de production, l'URL de l'application (APP_URL), et en s'assurant de passer en mode production avec **APP_ENV=production** et **APP_DEBUG=false**.

4. Finalisation du déploiement via Artisan :

- Exécution de la séquence de commandes artisan via le terminal SSH de Plesk pour finaliser l'installation et optimiser l'application :
 - **php artisan key:generate** : pour générer une clé de chiffrement unique et sécurisée.
 - **php artisan config:cache** : pour mettre en cache la configuration et accélérer le chargement.
 - **php artisan route:cache** : pour mettre en cache les routes.
 - **php artisan migrate --force** : pour exécuter les migrations et créer la structure de la base de données en production.
L'option --force est nécessaire car l'action est potentiellement destructive.

6. Bilan et Conclusion

6.1. Difficultés Rencontrées et Solutions Apportées

Au cours du développement du projet "Parks", j'ai fait face à plusieurs défis techniques et conceptuels qui ont été des opportunités d'apprentissage significatives.

- **Apprentissage et rigueur de TypeScript :**

- **Problème :** Venant d'un arrière-plan plus orienté JavaScript, l'apprentissage de TypeScript a été assez laborieux au début. La

nécessité de typer chaque variable, chaque propriété d'objet et chaque retour de fonction a demandé un temps d'adaptation et a ralenti ma vitesse de développement initiale.

- **Solution :** J'ai surmonté cette difficulté en adoptant une approche méthodique : j'ai pris le temps de créer des interfaces (interface) pour mes objets de données principaux (comme User, Parking, Booking) et de typer systématiquement les props de mes composants React. Bien que difficile au début, j'ai rapidement constaté les bénéfices : moins d'erreurs à l'exécution, un code plus lisible, et une auto-complétion bien plus efficace dans PHPStorm.

- **Le défi de la conception visuelle (Wireframing) :**

- **Problème :** La phase de conception des wireframes et des maquettes sur Figma a été très chronophage pour moi. Ayant une préférence naturelle pour le développement back-end, trouver l'inspiration pour créer un design d'interface à la fois esthétique et ergonomique a été un véritable défi.
- **Solution :** Pour surmonter ce blocage, j'ai consacré du temps à l'analyse de sites web existants dont j'appréciais le design (veille concurrentielle). J'ai également suivi des tutoriels sur les bonnes pratiques de l'UI/UX design. Bien que cette phase ait été exigeante, elle m'a forcé à sortir de ma zone de confort et à mieux comprendre l'importance de l'expérience utilisateur dans la réussite d'un projet.

- **Configuration de l'environnement de développement :**

- **Problème :** Au lancement du projet, j'ai fait face à des erreurs de connexion à la base de données et des problèmes de configuration entre le serveur Laravel et le serveur Vite.js, comme le montrent les erreurs "SQLSTATE[HY000] [2002]".
- **Solution :** J'ai résolu ces problèmes en vérifiant méticuleusement la configuration de mon fichier .env, en m'assurant que le service MySQL de MAMP était bien démarré, et en vérifiant que les ports utilisés par php artisan serve et npm run dev n'étaient pas en conflit. Cela a renforcé mes compétences en débogage d'environnement.

6.2. Axes d'Amélioration Possibles (Roadmap V2)

Le Proof of Concept (POC) actuel a permis de valider la stack technique (Laravel/Inertia/React) et d'implémenter les fonctionnalités vitales de gestion pour les propriétaires de parkings (gestion du CRUD et consultation des statistiques via le dashboard).

La V2 du projet se concentrera sur la finalisation du parcours client, l'optimisation des performances et la mise en conformité.

- **Finalisation du tunnel de réservation client :**
 - **Pages de détail :** Créer la vue React permettant d'afficher un parking spécifique depuis la page de résultats. * **Composant de réservation :** Développer le formulaire de sélection des dates/heures et de confirmation de la réservation, en lien avec le cas d'utilisation "Réserver un parking". * **Intégration du paiement :** Connecter le tunnel de réservation à un prestataire de paiement, en s'appuyant sur la structure de tables `invoices` et `payments` déjà en place (voir MCD, Annexe 1).
- **Implémentation des pages "Mon Compte" Client :** * **Mes Factures :** Créer la vue React pour que l'utilisateur puisse "Lister ses factures", en utilisant la structure backend existante. * **Moyens de paiement :** Développer l'interface pour "Gérer ses moyens de paiements", en s'appuyant sur le modèle `billing_methods`.
- **Amélioration de l'accessibilité (a11y) :**
 - Auditer et mettre à jour les composants React (formulaire, modales, navigation) pour se conformer aux standards d'accessibilité (WCAG/RGAA).
 - Assurer une navigation complète au clavier et la compatibilité avec les lecteurs d'écran pour garantir une application inclusive.
- **Optimisation des performances (Cache) :**
 - Mettre en place un système de cache (ex: Redis ou cache applicatif Laravel) pour les requêtes de base de données fréquemment utilisées.
 - Cibler en priorité la recherche publique de parkings et l'agrégation des statistiques du dashboard pour réduire les temps de réponse.
- **Système de soumission d'avis :**
 - Le POC actuel affiche les avis dans le dashboard. La V2 ajoutera le formulaire permettant au client de "Noter une réservation" après une location, en utilisant le `ReviewController` existant.
- **Fonctionnalités V2+ (déjà listées) :** * Intégrer une recherche par géolocalisation. * Développer une messagerie interne.

6.3. Conclusion Personnelle

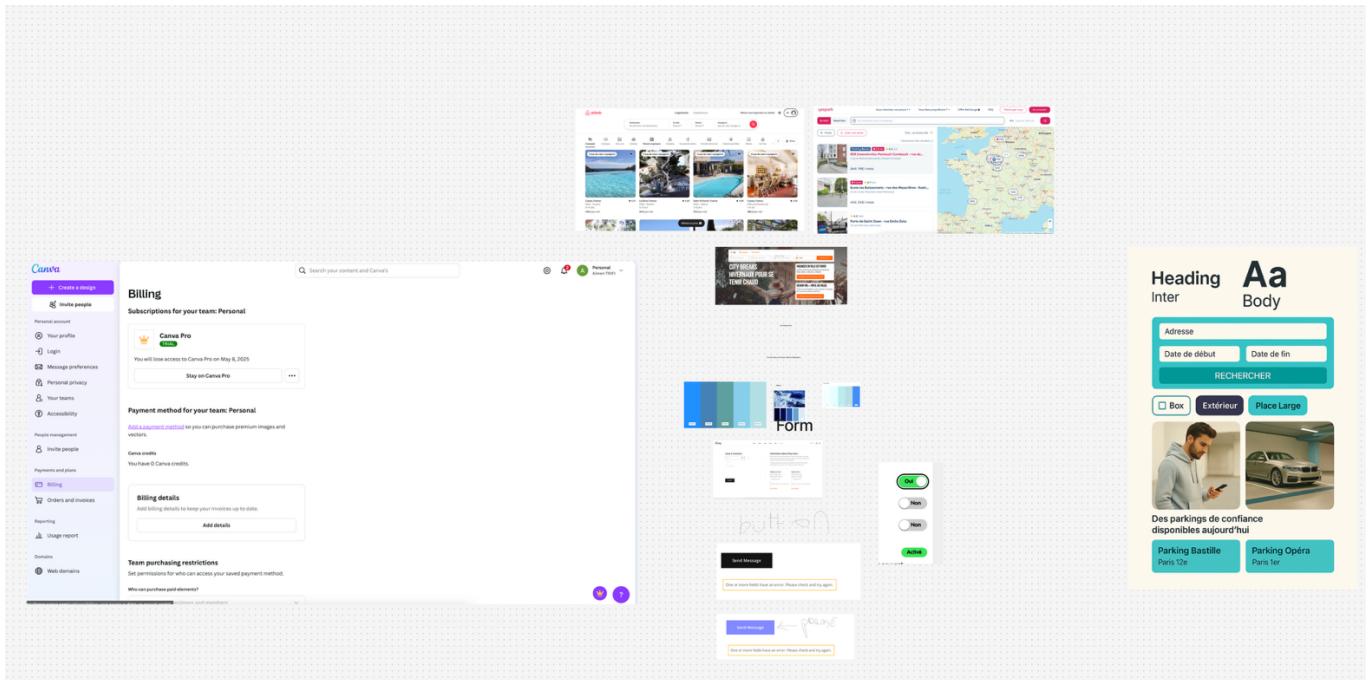
Ce projet a été une expérience extrêmement formatrice qui m'a permis de consolider mes compétences sur l'ensemble de la stack technique, de la conception initiale à la préparation du déploiement. Au-delà des aspects purement techniques, j'ai beaucoup appris sur la nécessité d'avoir une approche structurée et une grande rigueur pour mener un projet de cette envergure à son terme.

Chaque difficulté rencontrée, que ce soit dans la conception des interfaces ou dans la configuration de l'environnement, a été une leçon précieuse. Je tiens à remercier sincèrement toute l'équipe pédagogique de La Plateforme pour sa patience, sa disponibilité et ses conseils avisés qui m'ont été indispensables pour surmonter ces obstacles et pour mener à bien ce projet.

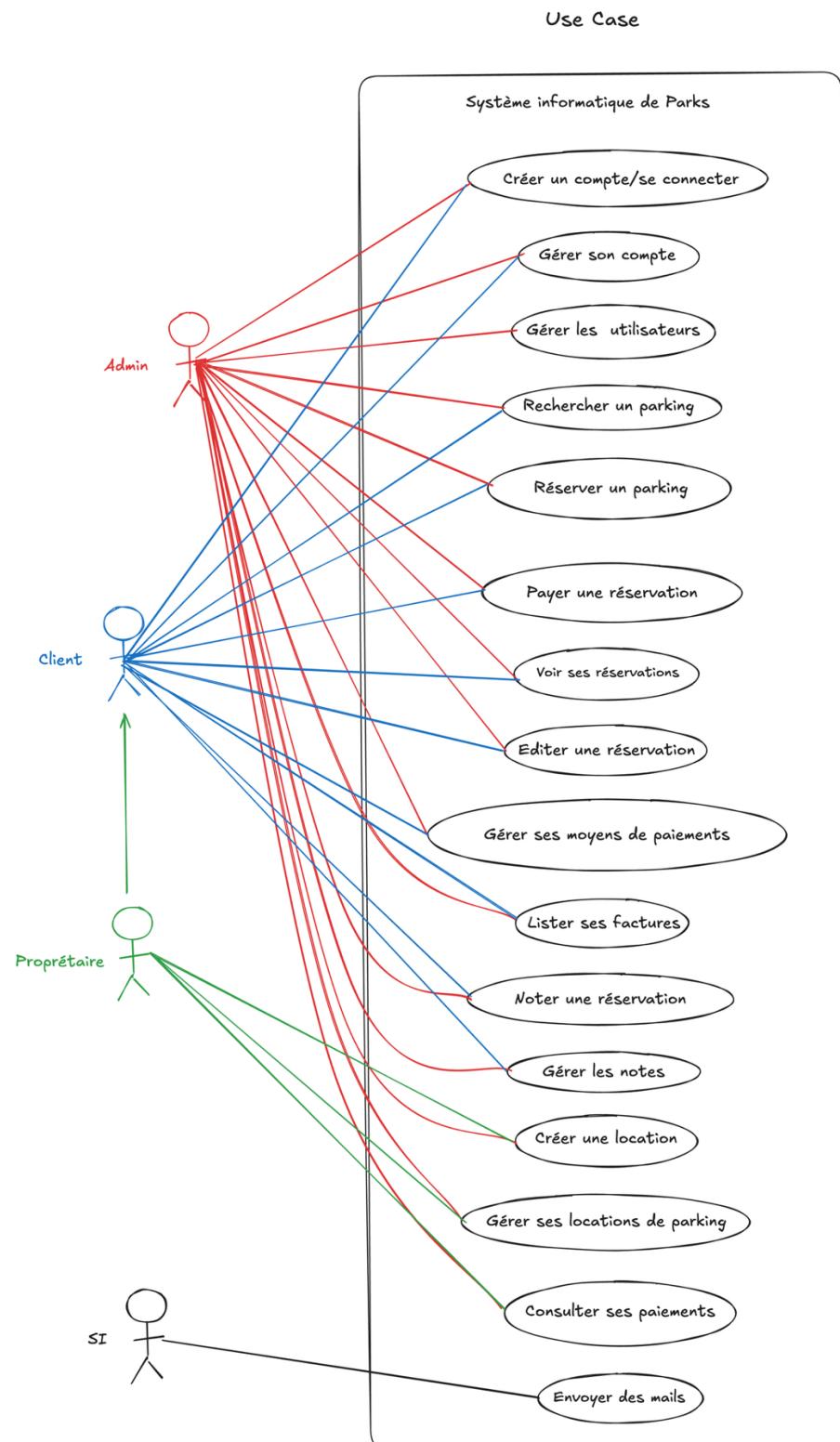
7. Annexe

Annexe 1 : Conception et Gestion de Projet

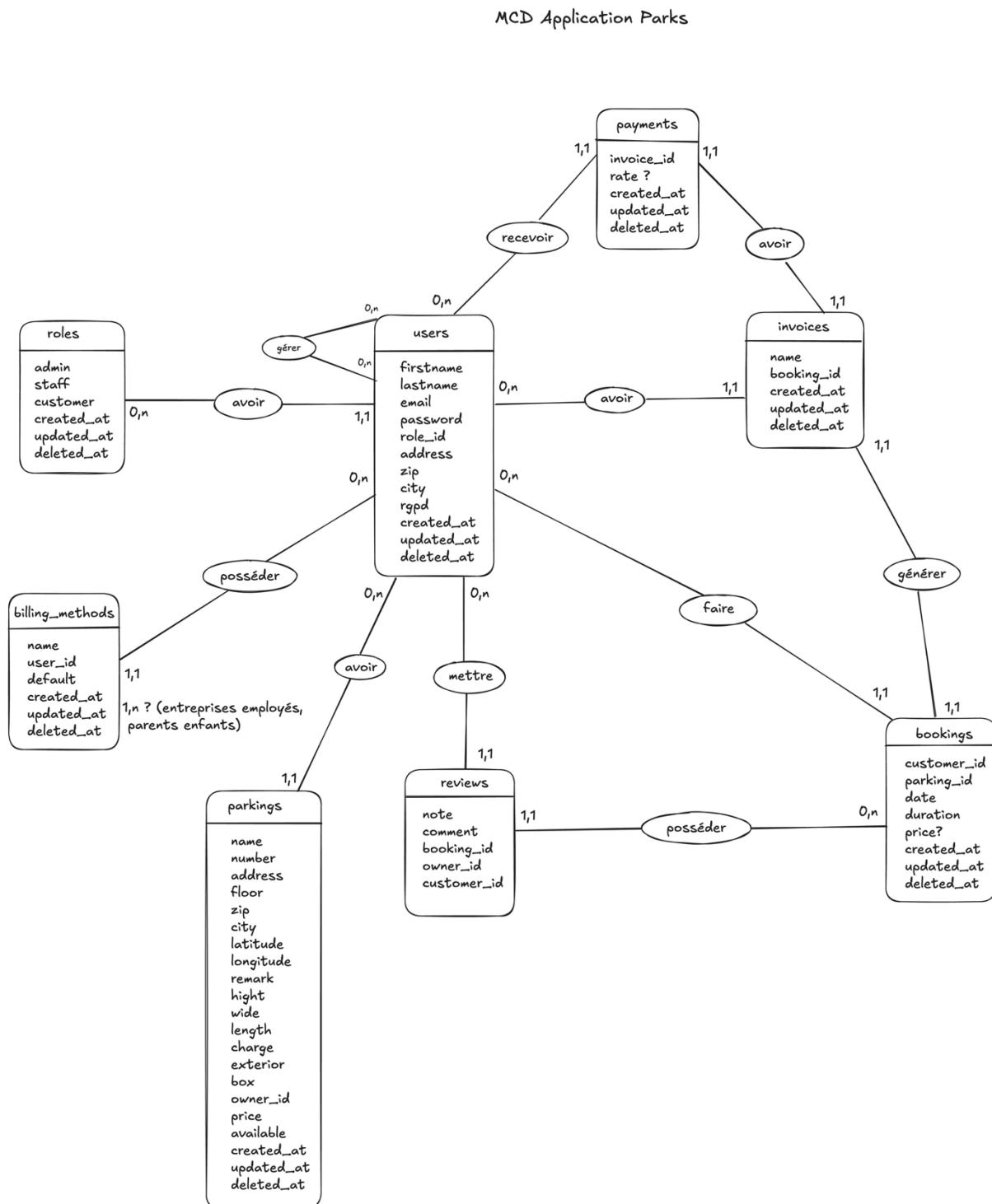
- **Figure 1.1 : Moodboard de l'application "Parks"**
 - **Description :** Ce tableau d'inspiration a défini la direction artistique du projet, en se concentrant sur une identité visuelle moderne, rassurante et simple d'utilisation.



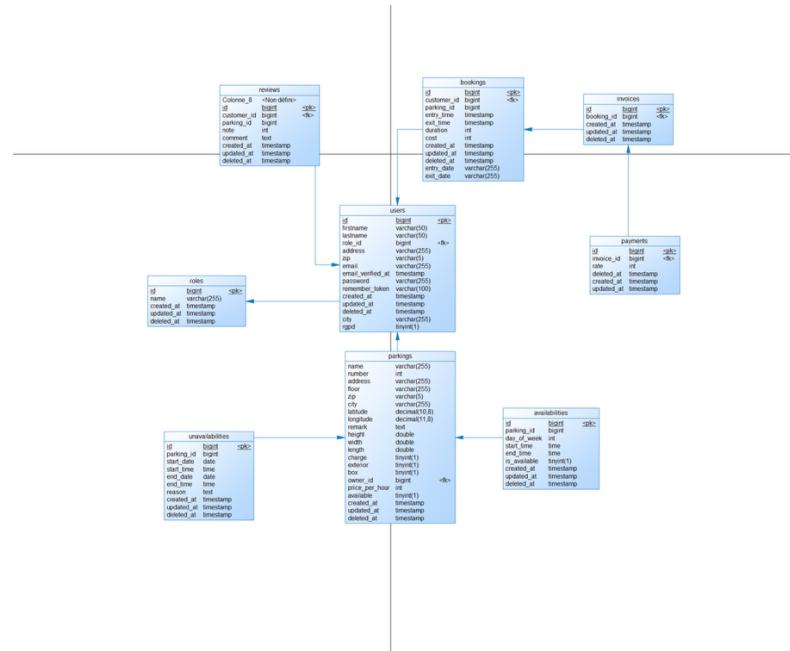
- **1.2 : Diagramme de cas d'utilisation (Use Case)**
 - **Description :** Ce schéma, réalisé avec Excalidraw, modélise les interactions des différents acteurs (Client, Propriétaire, Admin) avec le système, définissant ainsi le périmètre fonctionnel de l'application.



- **Figure 1.3 : Modèle Conceptuel de Données (MCD)**
 - **Description :** Ce schéma, réalisé avec Excalidraw, représente les entités de la base de données (users, parkings, bookings, etc.) et leurs relations, servant de base à la structure de la base de données.



- **Figure 1.4 : MPD**
 - Description : Capture d'écran du MPD



• Figure 1.5 : Gestion de projet Agile avec GitHub Projects

- **Description :** Capture d'écran du tableau Kanban sur GitHub Projects, montrant l'organisation des tâches en sprints et le suivi de leur avancement (À faire, En cours, Terminé).

The screenshot shows the GitHub Projects interface for the 'Parks' project. The board is organized into three columns: Backlog, In progress, and Done.

- Backlog (6 items):**
 - parks #11 MLD
 - parks #20 Pages CRUD
 - parks #21 Securisation routes et pages
 - parks #22 Restriction des pages en fonction des rôles
 - parks #23 Tests des pages
 - parks #24 Préparation déploiement
- In progress (4 items):**
 - parks #19 Pages Login , Register, Mot de passe oublié
 - parks #8 Faire un prototype
 - parks #39 page about guest
 - parks #40 page rent guest
- Done (16 items):**
 - parks #18 Homepage Guest
 - parks #17 Factories
 - parks #14 Controllers
 - parks #16 Seeders
 - parks #9 MCD
 - parks #15 Routes
 - parks #10 USE CASE

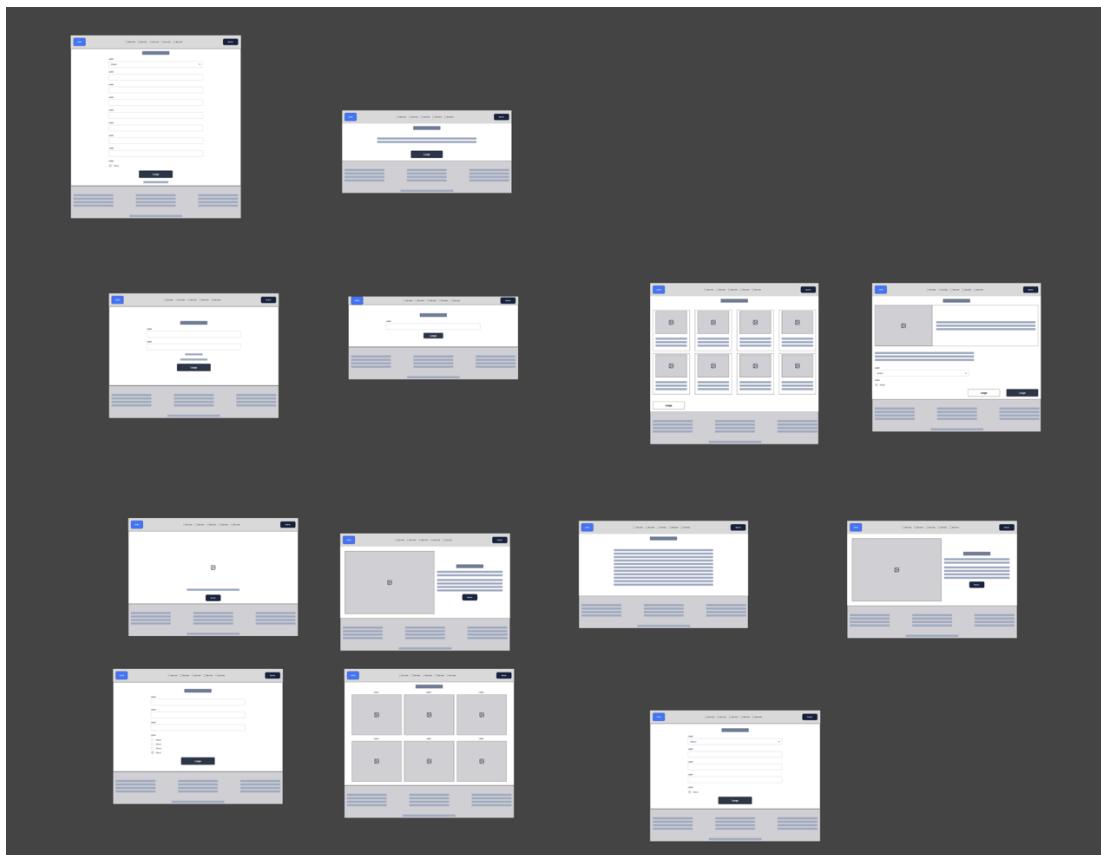
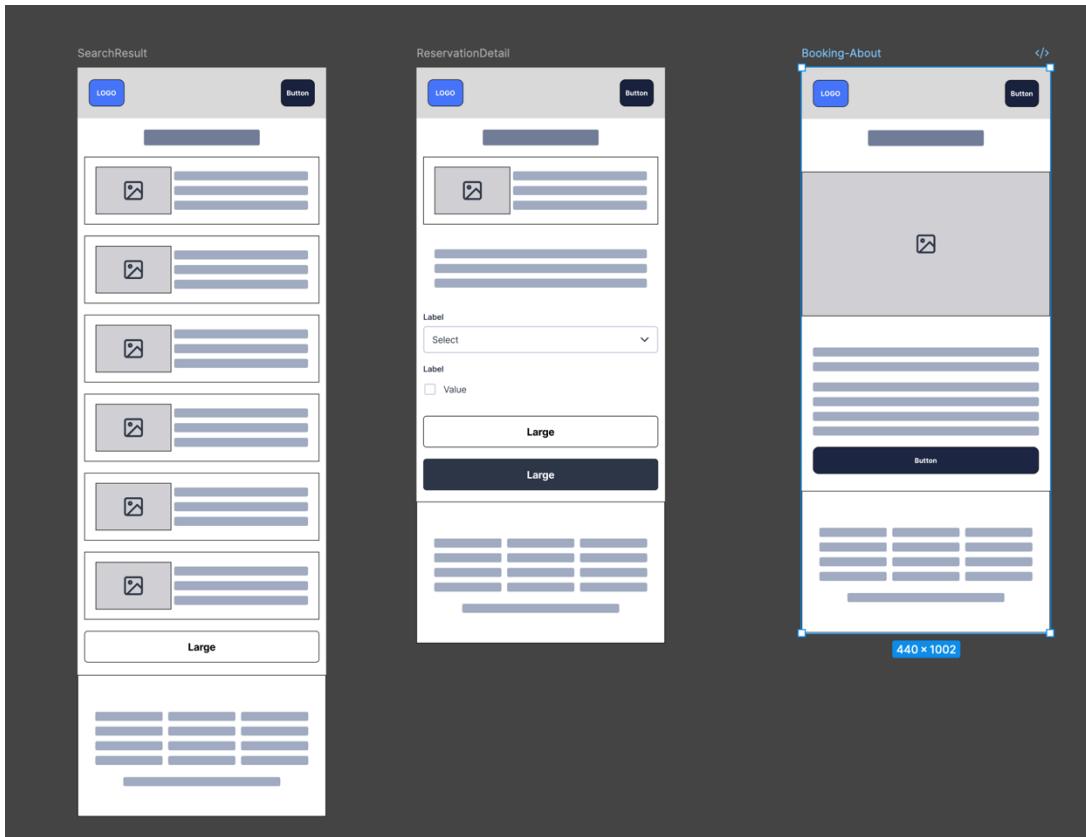
The board includes navigation tabs for Backlog, Priority board, Team items, Roadmap, In review, My items, Sprint2, Sprint_current, and New view. It also features a search bar, status update, insights, workflows, and a filter for keyword or field.

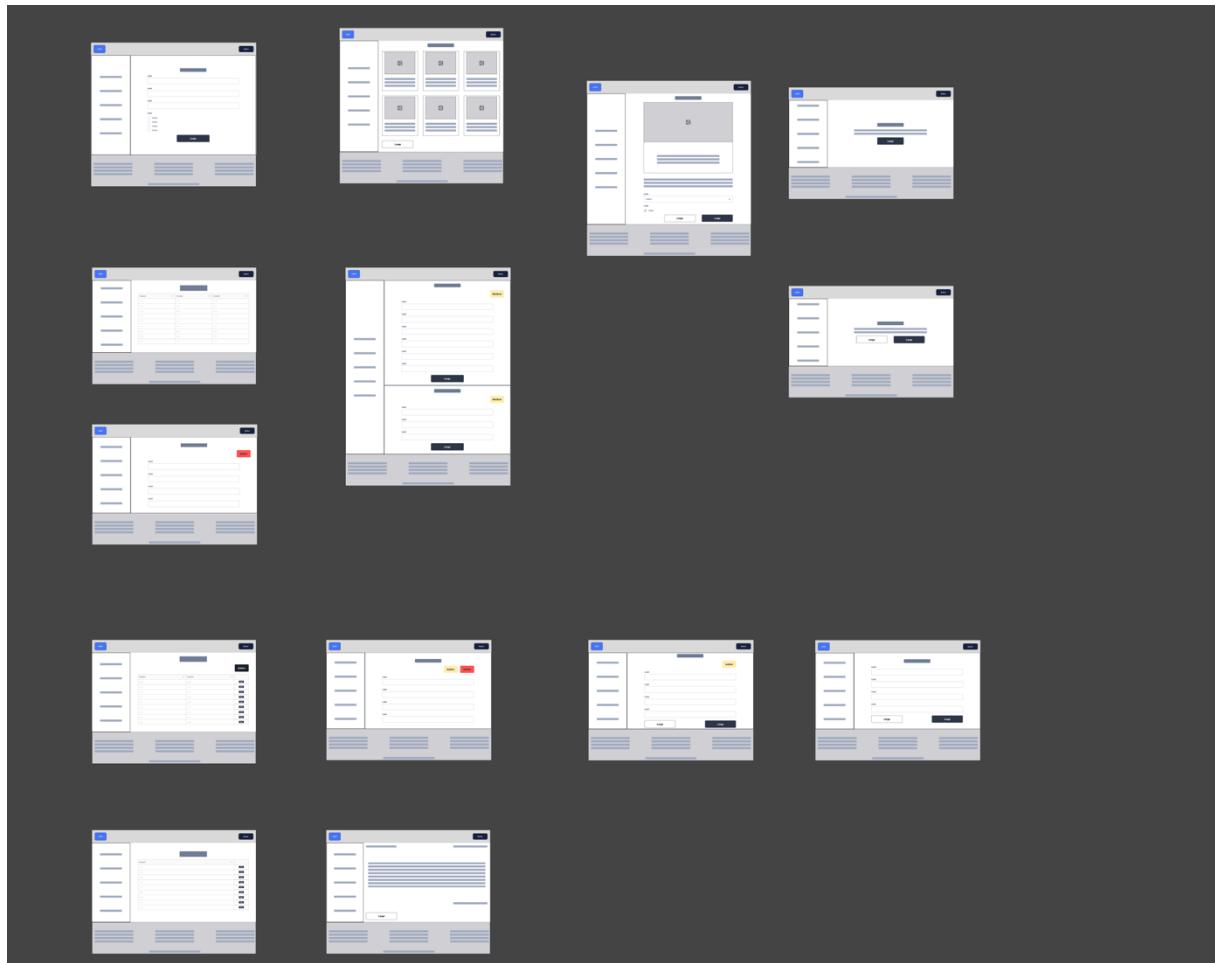
Annexe 2 : Wireframes de l'application (Figma)

• Figure 2.1 : Wireframes des principales pages de l'application

- **Description :** Cet assemblage de wireframes, créés sur Figma, présente la structure des pages clés de l'application en vues mobile et desktop, illustrant l'approche "mobile-first".







Annexe 3 : Environnement de Développement

- **Figure 3.1 : Lancement de l'environnement de développement**

- **Description :** Cette capture du terminal montre l'exécution de la commande composer run dev, qui lance simultanément le serveur **back-end Laravel** et le serveur **front-end Vite.js**.

```
aimen@mac parks % composer run dev
> ComposerConfig::disableProcessTimeout
> npx concurrently -c "#93c5fd,#c4b5fd,#fb7185" "php artisan serve" "php artisan queue:listen --tries=1" "php artisan psal --timeout=0" "npm run dev" --names=server,queue,logs,vite --kill-others
[vite]
[vite] > dev
[vite] > vite
[vite]
[queue]
[queue] [INFO] Processing jobs from the [default] queue.
[queue]
[logs]
[logs] [INFO] Tailing application logs. Press Ctrl+C to exit
[logs] Use -v|-vv to show more details
[server]
[server] [INFO] Server running on [http://127.0.0.1:8000].
[server]
[server] Press Ctrl+C to stop the server
[server]
[vite] Generating .flowbite-react/class-list.json file...
[vite]
[vite] VITE v6.2.0 ready in 763 ms
[vite]
[vite] + Local: http://localhost:5173/
[vite] + Network: use --host to expose
[vite]
[vite] LARAVEL v12.17.0 plugin v1.2.0
[vite]
[vite] + APP_URL: http://localhost
```

- **Figure 3.2 : Configuration de la connexion à la base de données**

- **Description :** Extrait du fichier .env dans PHPStorm, montrant la configuration de la connexion à la base de données MySQL.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=8889
DB_DATABASE=parks
DB_USERNAME=root
DB_PASSWORD=root|
```

Annexe 4 : Exemples de Code Back-end

- **Figure 4.1 : Fichier de migration pour la table parkings**

- **Description :** Extrait du code de la migration `create_parkings_table.php`, illustrant l'approche "code-first" pour créer la structure de la base de données avec Laravel.

```

Aïmen *
public function up(): void
{
    Schema::create('parkings', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->integer('number')->nullable();
        $table->string('address');
        $table->string('floor')->nullable();
        $table->string('zip', 5);
        $table->string('city');
        $table->decimal('latitude', 10, 8)->nullable();
        $table->decimal('longitude', 11, 8)->nullable();
        $table->text('remark')->nullable();
        $table->float('height');
        $table->float('width');
        $table->float('length');
        $table->boolean('charge')->default(false); // Borne de recharge
        $table->boolean('exterior')->default(false); // Extérieur ou non
        $table->boolean('box')->default(false); // Box fermé ou non
        $table->foreignId('owner_id')
            ->constrained('users')
            ->restrictOnDelete()
            ->cascadeOnUpdate();
        $table->integer('price_per_hour'); // Stocké en centimes
        $table->boolean('available')->default(true);
        $table->timestamps();
        $table->softDeletes();
        $table->index(['city']);
        $table->index(['zip']);
    });
}

```

- **Figure 4.2 : Modèle Eloquent Parking.php avec ses relations**
 - **Description :** Extrait du modèle Parking.php, montrant la propriété \$fillable (protection contre le Mass Assignment) et la définition des relations Eloquent.

```
class Parking extends Model
{
    protected $fillable = [
        'name',
        'number',
        'address',
        'floor',
        'zip',
        'city',
        'latitude',
        'longitude',
        'remark',
        'height',
        'width',
        'length',
        'charge',
        'exterior',
        'box',
        'owner_id',
        'price_per_hour',
        'available',
    ];
    no usages
    protected $casts = [...];
    no usages  ↳ Aïmen
    public function owner()
    {
        return $this->belongsTo(related: User::class, foreignKey: 'owner_id');
    }
    no usages  ↳ Aïmen
    public function bookings()
    {
        return $this->hasMany(related: Booking::class);
    }
}
```

- **Figure 4.3 : Logique métier dans le contrôleur UserController.php**
 - **Description :** Extrait de la méthode store du UserController.php, montrant la validation des données, la création d'un utilisateur et le hachage du mot de passe.

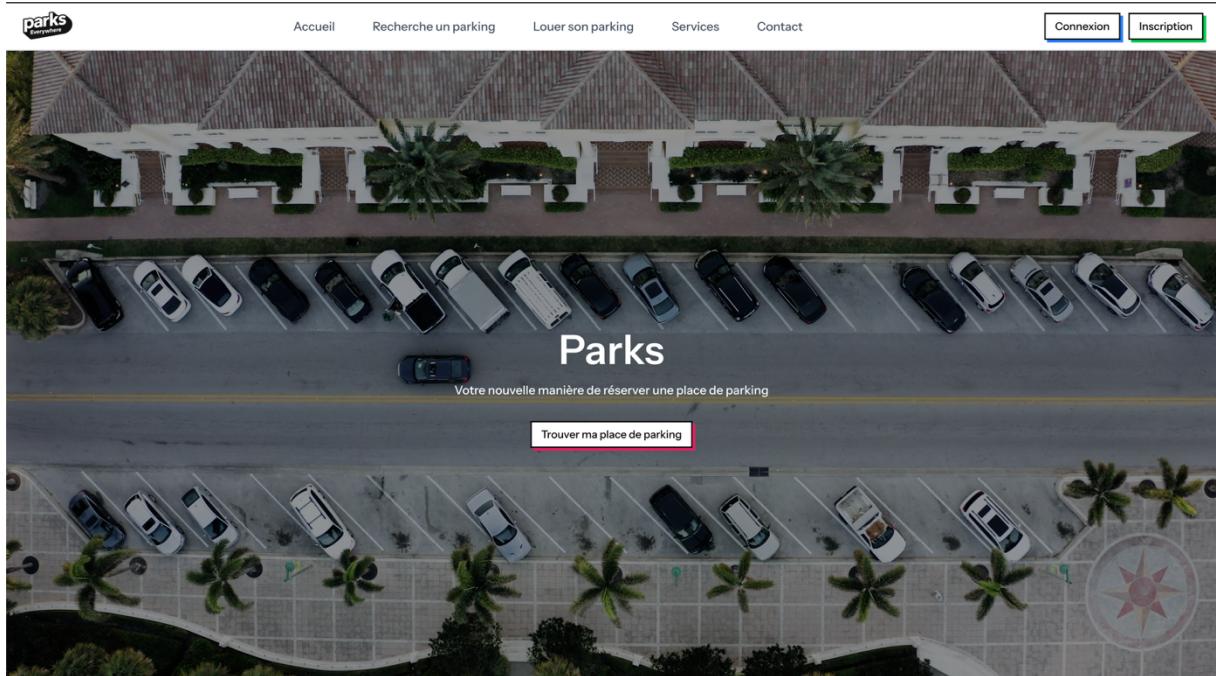
```

10  class UserController extends Controller
11
12      // Aïmen *
13
14      public function store(Request $request)
15  {
16
17          $validatedData = $request->validate([
18              'firstname' => 'required|string|max:255',
19              'lastname' => 'required|string|max:255',
20              'email' => 'required|string|email|max:255|unique:users',
21              'password' => 'required|string|min:8|confirmed',
22              'role_id' => 'required|integer|exists:roles,id',
23              'address' => 'nullable|string|max:255',
24              'zip' => 'nullable|string|max:10',
25              'city' => 'nullable|string|max:255',
26              'rgpd' => 'nullable|boolean',
27          ]);
28
29          try {
30              User::create([
31                  'firstname' => $validatedData['firstname'], // Correction: firstname -> firstname
32                  'lastname' => $validatedData['lastname'], // Correction: lastname -> lastname
33                  'email' => $validatedData['email'],
34                  'password' => Hash::make($validatedData['password']), // Utilisation de Hash::make()
35                  'role_id' => $validatedData['role_id'], // Correction: role_id
36                  'address' => $validatedData['address'],
37                  'zip' => $validatedData['zip'],
38                  'city' => $validatedData['city'],
39                  'rgpd' => $validatedData['rgpd'] ?? false,
40                  'email_verified_at' => null, // Ajout du champ si nécessaire
41              ]);
42
43              return redirect()->route('users.index')->with('success', 'Utilisateur créé avec succès.');
44
45          } catch (\Exception $e) {
46              return back()->withInput()->withErrors(['error' => 'Erreur lors de la création de l\'utilisateur: ' . $e->getMessage()]);
47
48          }
49
50      }
51
52  }
53
54
55
56
57
58
59
60
61
62
63
64

```

Annexe 5 : Rendu de l'Application

- **Figure 5.1 : Page d'accueil de "Parks"**
 - **Description :** Capture d'écran de la homepage de Parks



- **Figure 5.2 : Page de connexion fonctionnelle de "Parks"**
 - **Description :** Capture d'écran de l'interface de connexion de l'application.

The screenshot shows the login page of the 'Parks' website. At the top, there is a navigation bar with links for Accueil, Recherche un parking, Louer son parking, Services, and Contact. On the far right of the navigation bar, there are two buttons: 'Connexion' (blue) and 'Inscription' (green). Below the navigation bar is the 'parks everywhere' logo. The main section is titled 'Mon espace' and contains the text 'Saisir vos informations de connexion pour accéder à votre compte.' Below this, there are two input fields: 'Email' (containing 'email@parks.com') and 'Mot de passe' (containing 'Mot de passe'). To the right of the password field is a link 'Mot de passe oublié?'. Below the password field is a checkbox labeled 'Se rappeler de moi'. At the bottom of the form is a large black button with the text 'Se connecter' in white. At the very bottom of the page, there is a small link 'Vous n'avez pas de compte ? [S'enregistrer](#)'.

- **Figure 5.3 : Page d'inscription fonctionnelle de "Parks"**
 - **Description :** Capture d'écran de l'interface d'inscription de l'application.
-

The screenshot shows a web-based account creation form for 'parks'. At the top right is the 'parks' logo. Below it, the heading 'Create an account' is displayed, followed by the sub-instruction 'Enter your details below to create your account'. The form consists of four input fields: 'Name' (placeholder 'Full name'), 'Email address' (placeholder 'email@example.com'), 'Password' (placeholder 'Password'), and 'Confirm password' (placeholder 'Confirm password'). A large black 'Create account' button is centered below the inputs. At the bottom left, there is a link 'Already have an account? Log in'.

- **Figure 5.3 : Page non connecté**
 - **Description :** Capture d'écran d'une page non connectée

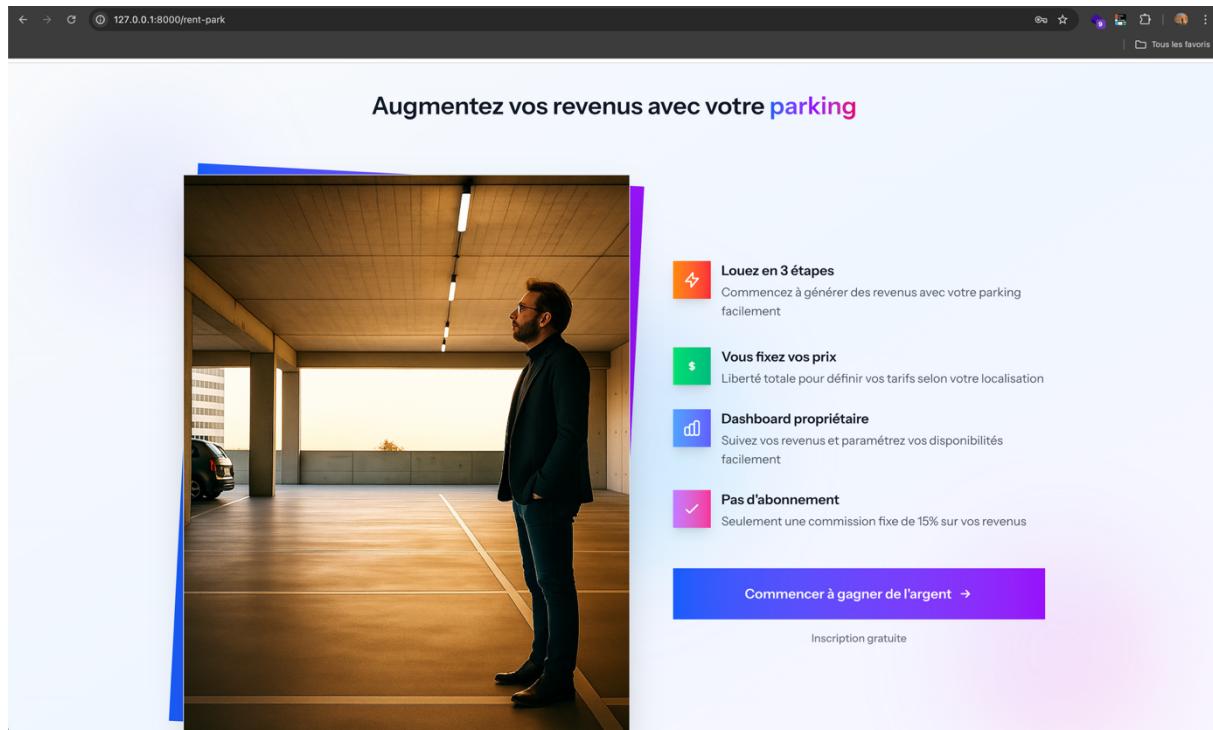


Figure 5.4 : Parcours de recherche publique

Description : Capture d'écran de l'interface de recherche publique et de la page de résultats. Conformément aux spécifications, le visiteur peut lancer une recherche par ville depuis une page dédiée (`Search.tsx`). Le backend (`ParkingController@search`) traite la demande, filtre les parkings disponibles et renvoie les résultats à une vue (`ResultSearch.tsx`) qui affiche la liste des parkings correspondants avec leurs informations principales.

Search.tsx

```

1 import React from 'react';
2 import GuestLayout from '@/layouts/GuestLayout';
3 import AuthLayout from '@/layouts/AuthLayout';
4 import SearchSection from '@/components/guest/parking/SearchSection';
5
6 import { SharedData } from '@/types';
7
8 Show usages ⌘ Aïmen
9 const SearchPage: React.FC = () : Element => {
10     return (
11         <SearchSection />
12     );
13 };
14
15
16 SearchPage.layout = (page: React.ReactElement) : Element => {
17     // 'page.props' contient les props de la page
18     const { auth } = page.props as SharedData;
19
20     if (auth.user) {
21         return <AuthLayout title="Rechercher un parking">{page}</AuthLayout>;
22     }
23
24     return <GuestLayout>{page}</GuestLayout>;
25 };
26
27 no usages ⌘ Aïmen
28 export default SearchPage;

```

Page de recherche de parking .

The screenshot shows a parking search form on a website. The URL in the address bar is 127.0.0.1:8000/search. The page title is "Réserver un parking en quelques clics". The sub-instruction below the title is "Trouvez et réservez votre place de parking idéale".

Ville: Cannes

Arrivée: Date: 13 novembre 2025, Heure: 09:00

Départ: Date: 13 novembre 2025, Heure: 16:00

Options:

- Box fermé:
- Parking extérieur:
- Borne de recharge:

RECHERCHER