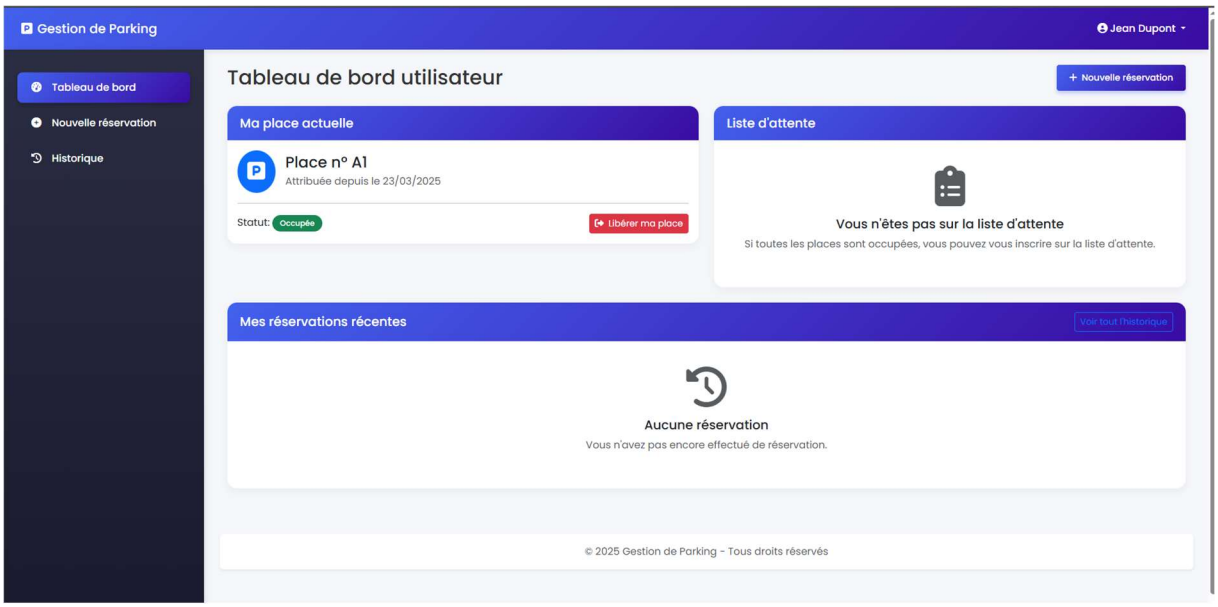
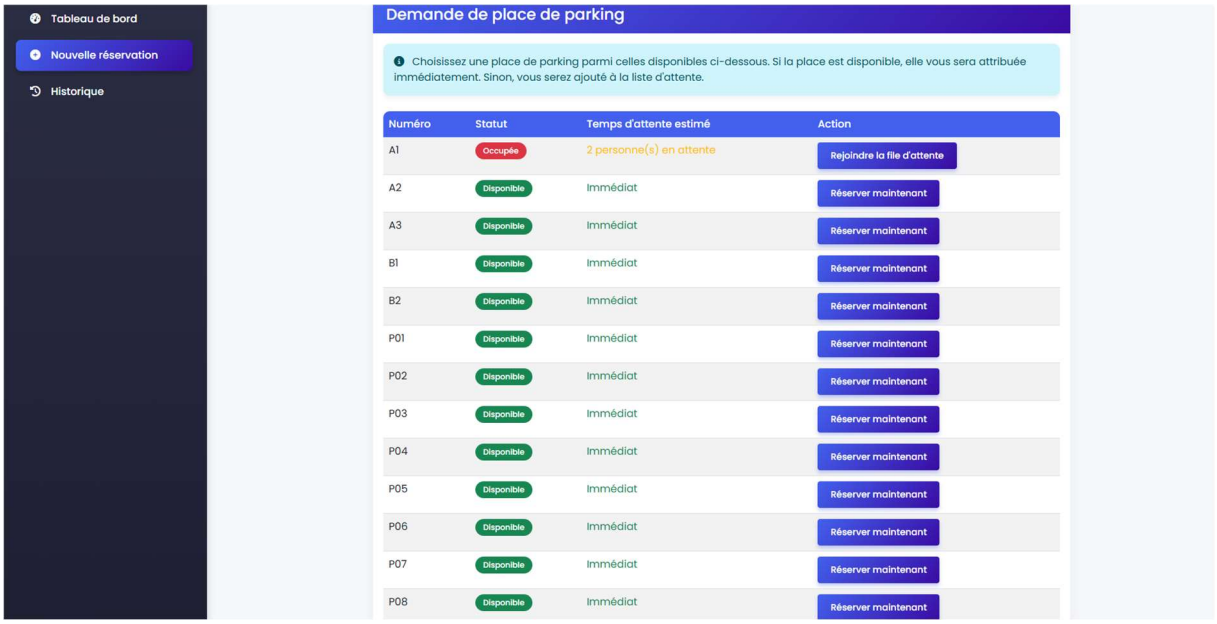
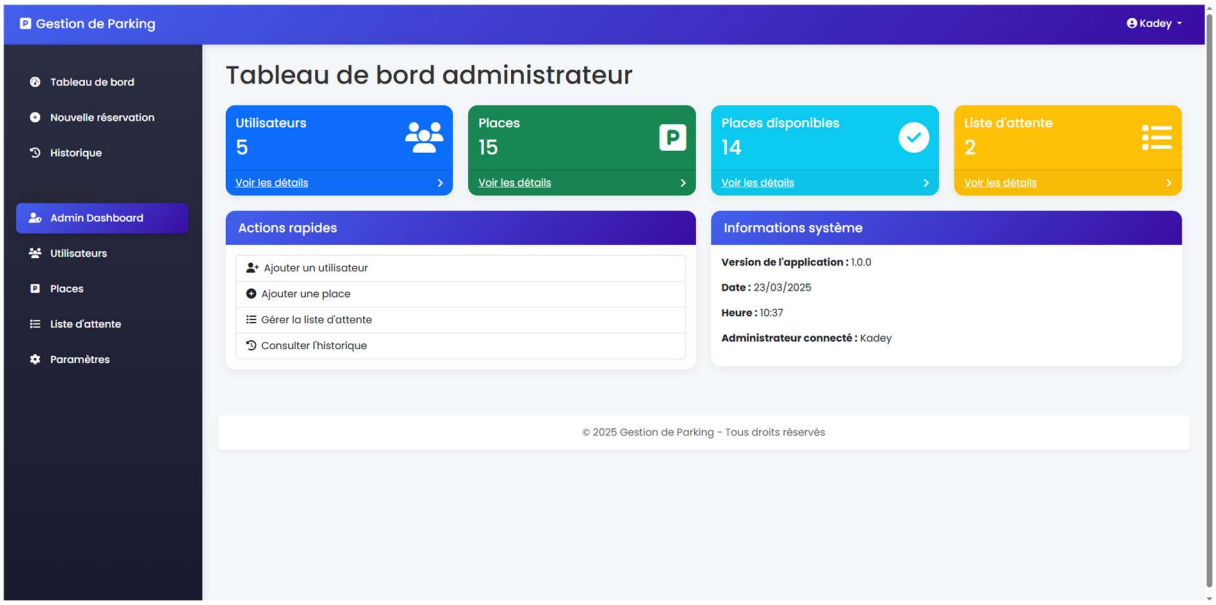
Documentation Technique - Application de Gestion de Parking

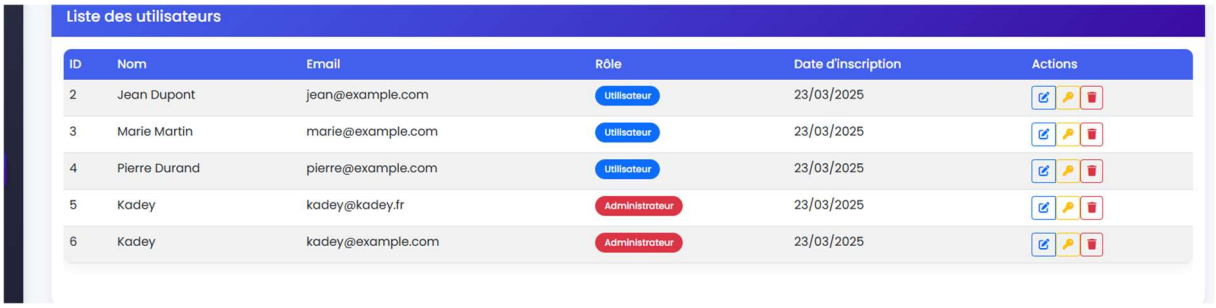
IMAGES UTILISATEURS :

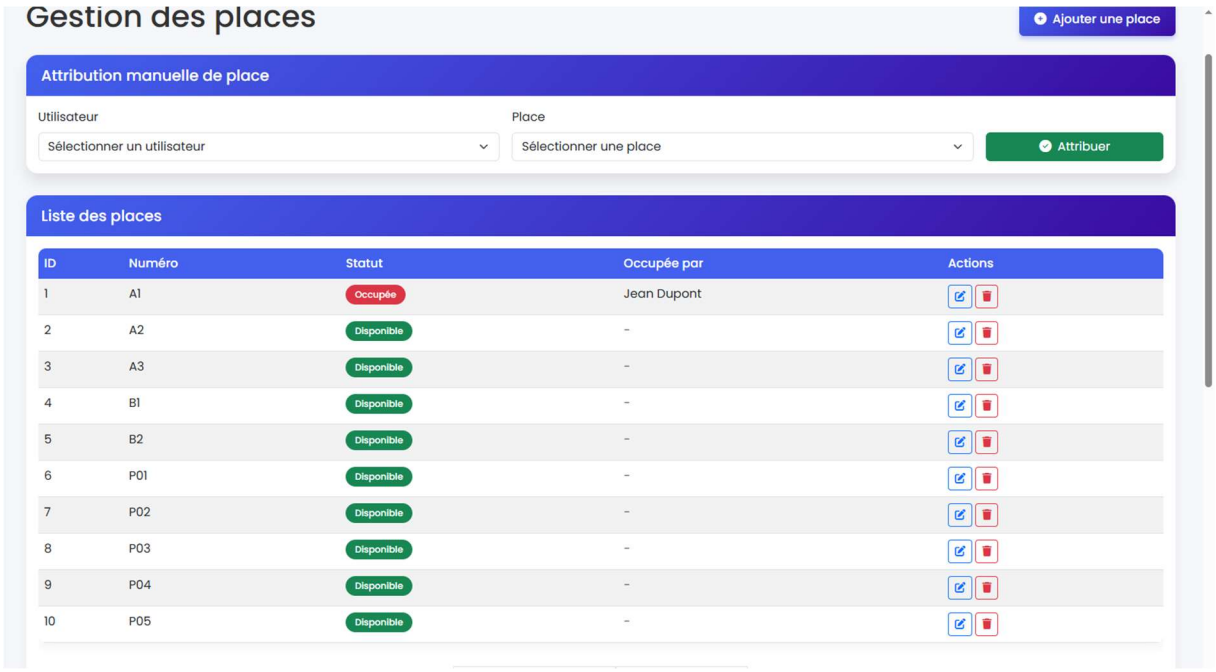


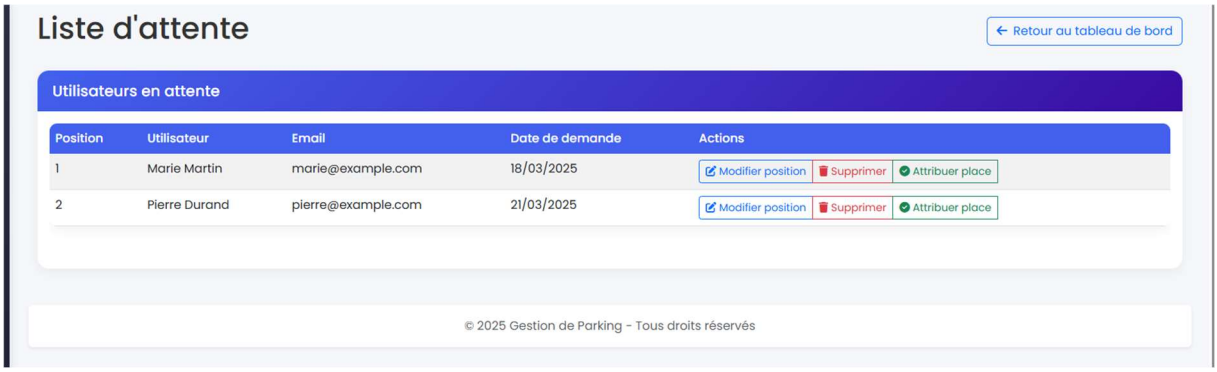


IMAGES ADMINISTRATEUR :


Gestion de Parking
Kadey
Tableau de bord
Nouvelle réservation
Historique
Admin Dashboard
Utilisateurs
Places
Liste d'attente
Paramètres
Tableau de bord administrateur
Utilisateurs
5
Voir les détails
Places
15
Voir les détails
Places disponibles
14
Voir les détails
Liste d'attente
2
Voir les détails
Actions rapides
Ajouter un utilisateur
Ajouter une place
Gérer la liste d'attente
Consulter l'historique
Informations système
Version de l'application : 1.0.0
Date : 23/03/2025
Heure : 10:37
Administrateur connecté : Kadey
© 2025 Gestion de Parking - Tous droits réservés


Liste des utilisateurs
ID Nom Email Rôle Date d'inscription Actions
2 Jean Dupont jean@example.com Utilisateur 23/03/2025
3 Marie Martin marie@example.com Utilisateur 23/03/2025
4 Pierre Durand pierre@example.com Utilisateur 23/03/2025
5 Kadey kadey@kadey.fr Administrateur 23/03/2025
6 Kadey kadey@example.com Administrateur 23/03/2025


Gestion des places
Ajouter une place
Attribution manuelle de place
Utilisateur
Sélectionner un utilisateur
Place
Sélectionner une place
Attribuer
Liste des places
ID Numéro Statut Occupée par Actions
1 A1 Occupée Jean Dupont
2 A2 Disponible -
3 A3 Disponible -
4 B1 Disponible -
5 B2 Disponible -
6 P01 Disponible -
7 P02 Disponible -
8 P03 Disponible -
9 P04 Disponible -
10 P05 Disponible -



1. Vue d'ensemble

Cette application Laravel est conçue pour gérer un système de réservation de places de parking. Elle permet aux utilisateurs de réserver des places, de les libérer, et aux administrateurs de gérer l'ensemble du système.

2. Structure des Dossiers

Dossiers Principaux

- **app/** : Contient la logique métier de l'application
- **Http/** : Gère les requêtes HTTP et les middlewares
- **Models/** : Définit les modèles de données et leurs relations
- **Providers/** : Contient les fournisseurs de services pour l'application
- **Helpers/** : Fonctions utilitaires réutilisables
- **bootstrap/** : Contient les fichiers d'initialisation de l'application
- **config/** : Contient les fichiers de configuration de l'application
- **database/** : Contient les migrations et les seeders
- **migrations/** : Définit la structure de la base de données
- **seeders/** : Crée des données initiales pour les tests
- **MCD/** : Contient le Modèle Conceptuel des Données (diagrammes)
- **public/** : Point d'entrée de l'application, contient les assets accessibles publiquement
- **resources/** : Contient les assets non-compilés comme les vues, les fichiers CSS et JS
- **views/** : Contient les templates Blade de l'application
- **admin/** : Vues pour l'interface administrateur

- **auth/** : Vues pour l'authentification
- **layouts/** : Templates de mise en page réutilisables
- **user/** : Vues pour l'interface utilisateur
- **routes/** : Définit les routes de l'application
- **web.php** : Routes pour l'interface web
- **console.php** : Commandes personnalisées pour Artisan
- **storage/** : Stockage de fichiers générés par l'application
- **tests/** : Contient les tests automatisés
- **vendor/** : Dépendances gérées par Composer

3. Fichiers Clés

Fichiers à la Racine

- **.env** : Variables d'environnement spécifiques à l'installation
- **.env.example** : Modèle pour les variables d'environnement
- **artisan** : Outil en ligne de commande Laravel
- **composer.json** et **composer.lock** : Gestion des dépendances PHP
- **package.json** : Gestion des dépendances JavaScript
- **vite.config.js** : Configuration du bundler Vite
- **phpunit.xml** : Configuration pour les tests PHPUnit

Documents

- **Maquette.png** : Design visuel de l'application
- **README.md** : Documentation générale du projet
- **URLS DU SITE.pdf** : Guide des URL disponibles
- **Compte rendu.docx** : Rapport détaillé sur le projet

4. Modèles (app/Models)

- **User.php** (82 lignes) : Représente un utilisateur du système
- Gère les informations d'authentification
- Contient la méthode isAdmin() pour vérifier les droits d'administration
- Relations avec les réservations et les places

- **Place.php** (47 lignes) : Représente une place de parking
- Contient le numéro et le statut (disponible, occupée)
- Relation avec l'utilisateur qui occupe la place
- Méthode isAvailable() pour vérifier la disponibilité
- **Reservation.php** (59 lignes) : Représente une réservation
- Contient les dates de début/fin et le statut
- Relations avec l'utilisateur et la place
- Méthode isActive() pour vérifier si la réservation est en cours
- **WaitingList.php** (56 lignes) : Gère la liste d'attente
- Contient la position de l'utilisateur dans la file
- Relations avec l'utilisateur et la place demandée
- **Setting.php** (61 lignes) : Gère les paramètres de l'application
- Stocke les configurations comme la durée de réservation

5. Contrôleurs (app/Http/Controllers)

- **Controller.php** (9 lignes) : Classe de base pour tous les contrôleurs
- **AuthController.php** (167 lignes) : Gère l'authentification
- Connexion et déconnexion
- Réinitialisation de mot de passe
- Changement de mot de passe
- **UserController.php** (244 lignes) : Gère les fonctionnalités utilisateur
- Affichage du tableau de bord utilisateur
- Demande et libération de place
- Gestion des réservations
- Gestion du profil utilisateur
- Affichage des notifications
- **AdminController.php** (531 lignes) : Gère les fonctionnalités administrateur
- Tableau de bord administrateur
- Gestion des utilisateurs (CRUD)

- Gestion des places (CRUD)
- Attribution de places
- Gestion de la liste d'attente
- Historique des réservations
- Paramètres de l'application

6. Vues (resources/views)

Layouts

- **layouts/app.blade.php** : Template principal de l'application

Authentification

- **auth/login.blade.php** : Formulaire de connexion
- **auth/forgot-password.blade.php** : Formulaire de récupération de mot de passe
- **auth/reset-password.blade.php** : Formulaire de réinitialisation de mot de passe
- **auth/change-password.blade.php** : Formulaire de changement de mot de passe

Interface Utilisateur

- **user/dashboard.blade.php** (168 lignes) : Tableau de bord utilisateur
- Affiche la place actuelle
- Montre la position dans la liste d'attente
- Liste les réservations récentes
- Bouton pour libérer une place
- **user/reservation-form.blade.php** : Formulaire de demande de place
- **user/reservation-history.blade.php** : Historique des réservations
- **user/profile.blade.php** : Gestion du profil
- **user/notifications.blade.php** : Affichage des notifications

Interface Administrateur

- **admin/dashboard.blade.php** : Tableau de bord administrateur
- **admin/users/** : Gestion des utilisateurs
- **admin/places/** : Gestion des places

- **admin/waiting-list/** : Gestion de la liste d'attente
- **admin/reservations/** : Gestion des réservations
- **admin/settings.blade.php** : Paramètres de l'application

7. Routes (routes/web.php)

Le fichier **web.php** (90 lignes) définit toutes les routes de l'application :

- **Routes publiques** : Accueil, page de connexion
- **Routes d'authentification** : Connexion, déconnexion, récupération de mot de passe
- **Routes utilisateur** : Dashboard, réservations, profil, notifications
- **Routes administrateur** : Gestion des utilisateurs, places, liste d'attente, paramètres

8. Migrations (database/migrations)

Les fichiers de migration définissent la structure de la base de données :

- **create_users_table.php** : Table des utilisateurs
- **create_places_table.php** : Table des places de parking
- **create_reservations_table.php** : Table des réservations
- **create_waiting_list_table.php** : Table de la liste d'attente
- **create_settings_table.php** : Table des paramètres
- **add_role_to_users_table.php** : Ajout du champ rôle aux utilisateurs
- **add_place_id_to_waiting_lists_table.php** : Ajout de la référence à la place demandée

9. Fonctionnalités Clés

Utilisateurs

- Connexion/déconnexion
- Demande de place de parking
- Libération de place
- Consultation de la position dans la liste d'attente
- Consultation de l'historique des réservations
- Gestion du profil

Administrateurs

- Gestion des utilisateurs (ajout, modification, suppression)
- Gestion des places (ajout, modification, suppression)
- Attribution manuelle de places
- Gestion de la liste d'attente
- Consultation de l'historique des réservations
- Configuration des paramètres de l'application

10. Système de Rôles

- **admin** : Accès complet à toutes les fonctionnalités
- **user** : Accès limité aux fonctionnalités utilisateur

11. Workflows Principaux

Réservation d'une place

1. L'utilisateur se connecte
2. Il accède au formulaire de demande de place
3. Si une place est disponible, elle lui est attribuée immédiatement
4. Sinon, il est ajouté à la liste d'attente

Libération d'une place

1. L'utilisateur se connecte à son tableau de bord
2. Il clique sur "Libérer ma place"
3. Le système marque la réservation comme terminée
4. La place est marquée comme disponible

Attribution automatique de place

1. Un administrateur libère une place
2. Le système vérifie la liste d'attente
3. La place est attribuée à l'utilisateur en tête de liste

12. Configuration et déploiement

Pour déployer l'application :

1. Cloner le dépôt

2. Copier .env.example vers .env et configurer les variables
3. Installer les dépendances avec composer install
4. Générer la clé avec php artisan key:generate
5. Exécuter les migrations avec php artisan migrate
6. Exécuter les seeders avec php artisan db:seed
7. Lancer le serveur avec php artisan serve

SQLite :

- Nécessite pas de serveur de base de données séparé
- Stocke toutes les données dans un seul fichier, facilitant les sauvegardes
- Suffisamment performant pour gérer la charge d'un système de gestion de parking de taille moyenne
- Idéal pour les environnements de développement et les petites installations