Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance ▸ *TRIFI*
Nom d'usage ▸ *TRIFI*
Prénom ▸ *Aïmen*
Adresse ▸ *5 rue du grand mur, 06250 Mougins*

Titre professionnel visé

Développeur web et web mobile

MODALITE D'ACCES :

x Parcours de formation
☐ Validation des Acquis de l'Expérience (VAE)

Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel. **Ce titre est délivré par le Ministère chargé de l'emploi.**

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen**.

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.

Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.

2. du **Dossier Professionnel** (DP) dans lequel le candidat a consigné les preuves de sa pratique professionnelle.

3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation

4. de l'entretien final (dans le cadre de la session titre).

[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]

Ce dossier comporte :

- pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- une déclaration sur l'honneur à compléter et à signer ;
- des documents illustrant la pratique professionnelle du candidat (facultatif)
- des annexes, si nécessaire.

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.

http://travail-emploi.gouv.fr/titres-professionnels

Sommaire

Exemples de pratique professionnelle

Activité-type n° 1 Développer la partie front-end d'une application web ou web mobile sécurisée.	**p.**	**5**
► CP n°1 Installer et configurer son environnement de travail en fonction du projet web ou web mobile	p.	5
► CP n° 2 Maquetter des interfaces utilisateur web ou web mobile	p.	8
► CP n° 3 Réaliser des interfaces utilisateur statiques web ou web mobile	p.	10
► CP n° 4 Développer la partie dynamique des interfaces utilisateur web ou web mobile	p.	12
Activité-type type n° 2 Développer la partie back-end d'une application web ou web mobile sécurisée	**p.**	**14**
► CP n°1 Mettre en place une base de données relationnelle	p.	14
► CP n° 2 Développer des composants d'accès aux données SQL et NoSQL	p.	16
► CP n° 3 Développer des composants métier coté serveur	p.	18
► CP n° 3 Documenter le déploiement d'une application dynamique web ou web mobile	p.	20

DOSSIER PROFESSIONNEL (DP)

Titres, diplômes, CQP, attestations de formation *(facultatif)*	**p.**	22
Déclaration sur l'honneur	**p.**	23
Documents illustrant la pratique professionnelle *(facultatif)*	**p.**	24
Annexes *(Si le RC le prévoit)*	**p.**	25

EXEMPLES DE PRATIQUE PROFESSIONNELLE

DOSSIER PROFESSIONNEL (DP)

Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée.

Compétence n°1 ▸ *Installer et configurer son environnement de travail en fonction du projet web ou web mobile*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

- **Initialisation du projet et gestion de version :**
  - Dès le début, j'ai initialisé un dépôt **Git** pour assurer le suivi de toutes les modifications du code.
  - J'ai ensuite utilisé l'installateur **Laravel** pour créer le projet, en choisissant le starter kit **laravel/breeze** avec l'option **React + Inertia**. L'ensemble du projet a été versionné et synchronisé avec un dépôt distant sur **GitHub**.

- **Installation des dépendances :**
  - Après l'installation des dépendances back-end par **Composer**, le script m'a invité à installer les dépendances front-end, ce que j'ai fait avec la commande **npm install**.

- **Configuration de l'IDE et de la base de données :**
  - J'ai ouvert le projet dans mon IDE **PHPStorm**.
  - J'ai démarré **MAMP** pour lancer le service de base de données **MySQL**, puis j'ai configuré la connexion dans le fichier .env du projet.

- **Lancement de l'environnement de développement :**
  - Pour lancer l'ensemble des services nécessaires au développement, j'ai utilisé la commande **composer run dev**. Comme défini dans le fichier composer.json, ce script exécute simultanément :
    - Le serveur back-end Laravel (php artisan serve).
    - Le serveur de développement front-end Vite.js (npm run dev) pour la compilation des assets et le rechargement à chaud.

2. Précisez les moyens utilisés :

- **Frameworks et bibliothèques :** Laravel 12, Breeze, React, Inertia.js, Vite.js.

• **Outils : Git et GitHub pour la gestion de version**, Composer, NPM, PHPStorm, MAMP (pour MySQL).

3. Avec qui avez-vous travaillé ?

J'ai mené ce projet seul. Cependant, j'ai essayé d'avoir une organisation de travail professionnelle, en utilisant
des outils collaboratifs (GitHub + GitHub Projects), afin d'avoir un plan détaillé et un suivi des tâches à effectuer.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *La Plateforme*

Chantier, atelier, service ▸ *En formation*

Période d'exercice ▸ Du : *23/09/2024* au : *15/10/2025*

5. Informations complémentaires *(facultatif)*

DOSSIER PROFESSIONNEL (DP)

Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Compétence n° 2 ▸ *Maquetter des interfaces utilisateur web ou web mobile*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

• **Analyse fonctionnelle :** J'ai modélisé les interactions des acteurs (Client, Propriétaire, Admin) avec le système en créant un **diagramme de cas d'utilisation (Use Case)** avec l'outil en ligne **Excalidraw**.

• **User Stories :** J'ai ensuite traduit ces cas d'utilisation en **"user stories"**, comme : *"En tant que Client, je veux rechercher un parking par ville et par date afin de trouver une place disponible."*

• **Wireframing :** Sur la base de ces besoins, j'ai utilisé Figma pour créer les wireframes (maquettes filaires) des écrans principaux de l'application. J'ai conçu les vues pour desktop et mobile en adoptant une approche "mobile-first" afin de garantir une expérience utilisateur optimale sur tous les appareils.

2. Précisez les moyens utilisés :

• **Outils de conception : Figma** pour les wireframes, **Excalidraw** pour le diagramme de cas d'utilisation.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul pour maquetter les interfaces utilisateur web et web mobile. Pour les uses case et users stories, j'ai d'abord fait valider par mon formateur.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *La Plateforme*

Chantier, atelier, service ▸ *formation*

Période d'exercice ▸ Du : *23/09/2024* au : *15/10/2025*

5. Informations complémentaires *(facultatif)*

Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Exemple n° 3 ▸ *Réaliser des interfaces utilisateur statiques web ou web mobile*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Une fois les maquettes validées, je les ai traduites en code pour construire la base statique de l'application. J'ai utilisé **React (avec TypeScript)** pour créer une interface modulaire et maintenable.

- **Structuration en composants :** J'ai décomposé l'interface en plusieurs composants logiques et réutilisables. Par exemple, GuestLayout.tsx sert de gabarit pour toutes les pages publiques, en important des composants comme Header.tsx et Footer.tsx. Les pages elles-mêmes, comme HomeGuest.tsx, sont composées de sections spécifiques (HeroSection.tsx). Cette approche garantit la cohérence visuelle et facilite les mises à jour.
- **Style avec Tailwind CSS :** L'ensemble du style a été géré avec le framework **Tailwind CSS**, comme configuré dans package.json. J'ai utilisé ses classes utilitaires (ex: flex, bg-white, md:h-[70px]) directement dans mon code JSX pour appliquer le design de manière rapide et cohérente. Pour des éléments plus complexes comme les sélecteurs de date dans le formulaire de recherche, j'ai intégré des bibliothèques de composants pré-stylés comme **Flowbite React**.
- **Responsive Design :** Grâce aux breakpoints de **Tailwind** (ex: md:, lg:), j'ai rendu les composants adaptatifs. Par exemple, le composant Header.tsx affiche une navigation complète sur grand écran et passe à un menu "burger" sur mobile, assurant une expérience utilisateur optimale sur tous les appareils.

2. Précisez les moyens utilisés :

- **Bibliothèques :** React, TypeScript, Tailwind CSS, Flowbite React.
- I**DE :** PHPStorm.

3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé en autonomie, mais j'ai profité des retours et conseils de mon formateur.

DOSSIER PROFESSIONNEL (DP)

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *La Plateforme*

Chantier, atelier, service ▸ *formation*

Période d'exercice ▸ Du : *23/09/2024* au : *15/10/2024*

5. Informations complémentaires *(facultatif)*

Activité-type 1 Réaliser des interfaces utilisateur statiques web ou web mobile

Compétence n° 4 ▸ Développer la partie dynamique des interfaces utilisateur web ou web mobile

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour transformer les interfaces statiques en une expérience utilisateur fluide et réactive, j'ai mis en œuvre la partie dynamique en utilisant les fonctionnalités offertes par l'écosystème React et la bibliothèque **Inertia.js**.

- **Gestion de formulaire avec useForm :** Le hook useForm d'Inertia a été central dans mon approche. Dans le composant SearchSection.tsx, je l'ai utilisé pour gérer l'état complet du formulaire de recherche (ville, dates, options). Ce hook simplifie grandement la communication avec le back-end :
  - Il maintient l'état des champs (data).
  - Il gère l'état de soumission (processing), ce qui me permet de désactiver le bouton de recherche pendant l'envoi pour éviter les requêtes multiples.
  - Il récupère et affiche automatiquement les erreurs de validation renvoyées par Laravel. La soumission se fait via la fonction post(route('parking.search'), ...), qui envoie les données au contrôleur Laravel sans aucun rechargement de page, créant ainsi une expérience de type Single Page Application (SPA).

- **Interactivité des composants React :** J'ai utilisé les hooks natifs de React, comme useState, pour gérer l'état local des composants. Par exemple, dans SearchSection.tsx, l'état des calendriers (startDate, endDate) est géré localement avant d'être formaté et synchronisé avec l'objet data du hook useForm. De même, les boutons pour les options de recherche (Box fermé, Borne de recharge) appellent une fonction toggleOption qui met à jour l'état du formulaire de manière réactive.

- **Gestion de l'authentification dynamique :** La page de connexion (login.tsx) applique le même principe. Le hook useForm gère les identifiants de l'utilisateur. Lors de la soumission, si Laravel renvoie une erreur (ex: mot de passe incorrect), useForm la capture et me permet de l'afficher à l'utilisateur via le composant InputError, sans recharger la page et en conservant les données déjà saisies (sauf le mot de passe, par sécurité).

2. Précisez les moyens utilisés :

- **Bibliothèques :** React (hooks useState, useEffect), Inertia.js (hook useForm).
- **Langage :** TypeScript.
- **Outils :** Outils de développement React pour l'inspection des états et des props des composants.

3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé en autonomie.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *La Plateforme*

Chantier, atelier, service ▸ *formation*

Période d'exercice ▸ Du : *23/09/2024* au : *15/10/2025*

5. Informations complémentaires *(facultatif)*

Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

Compétence n° 1 ▸ *Mettre en place une base de données relationnelle*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour assurer la persistance et la cohérence des données de l'application "Parks", j'ai mis en place une base de données relationnelle en suivant une démarche structurée.

- **Conception avec la méthode Merise :** La première étape a été la modélisation des données. J'ai utilisé l'outil **Excalidraw** pour réaliser un **Modèle Conceptuel de Données (MCD)**. Ce schéma a permis de définir les entités principales de l'application (users, parkings, bookings, roles, reviews, etc.), leurs attributs respectifs, et les relations qui les unissent avec leurs cardinalités. Par exemple, le MCD montre qu'un user peut posséder plusieurs parkings et effectuer plusieurs bookings. Cette étape de conception a été cruciale pour garantir une structure logique et évolutive.

- **Implémentation avec Laravel Migrations :** Une fois le modèle de données validé, je l'ai traduit en structure de base de données physique en utilisant le système de **migrations de Laravel**. Cette approche *code-first* permet de versionner l'état de la base de données avec **Git** et de la recréer de manière fiable sur n'importe quel environnement. J'ai créé un fichier de migration pour chaque table, par exemple :
  - 2025_06_26_121021_create_parkings_table.php : pour définir la table parkings avec ses colonnes (name, address, city, price_per_hour, etc.).
  - 2025_06_26_182059_create_bookings_table.php : pour la table bookings, en y définissant les clés étrangères (customer_id, parking_id) qui assurent l'intégrité référentielle avec les tables users et parkings.

- **Peuplement de la base de données :** Pour disposer de données de test, j'ai également créé des *seeders* (ex: ParkingSeeder.php, UserSeeder.php) qui utilisent des *factories* pour peupler la base de manière automatisée.

2. Précisez les moyens utilisés :

- **Méthodologie de conception :** Merise (MCD, MLD).
- **Outils de modélisation :** Excalidraw.
- **SGBD :** MySQL (lancé via MAMP).
- **Framework :** Laravel (Migrations, Seeders, Factories).
- **Outils de développement :** PHPStorm et son terminal intégré pour les commandes artisan (création de contrôleurs, de modèles, etc.).

3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé en autonomie.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *La Plateforme*

Chantier, atelier, service ▸ *formation*

Période d'exercice ▸ Du : *23/09/2024* au : *15/10/2025*

5. Informations complémentaires *(facultatif)*

Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

Compétence n° 2 ▸ *Développer des composants d'accès aux données SQL et NoSQL*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour que mon application puisse interagir avec la base de données **MySQL**, j'ai développé des composants d'accès aux données en utilisant l'ORM (Object-Relational Mapper) **Eloquent** de Laravel. Cette approche m'a permis de manipuler les données de manière sécurisée et orientée objet, en abstaillant les requêtes SQL brutes.

- **Création des Modèles Eloquent :** Pour chaque table de ma base de données (définie dans mon MCD), j'ai créé un modèle Eloquent correspondant via la commande php artisan make:model. Par exemple, User.php, Parking.php, et Booking.php sont des classes qui héritent des fonctionnalités d'Eloquent pour interagir avec leurs tables respectives.

- **Sécurisation contre le "Mass Assignment" :** Dans chaque modèle, j'ai défini la propriété $fillable pour déclarer explicitement les champs qui peuvent être assignés massivement lors de la création ou de la mise à jour d'un enregistrement. Par exemple, dans Booking.php, seuls des champs comme customer_id, parking_id, entry_date sont autorisés, ce qui protège l'application contre les vulnérabilités où un utilisateur malveillant pourrait tenter de modifier des champs sensibles.

- **Définition des relations entre les données :** J'ai implémenté les relations logiques entre mes différentes entités directement dans les modèles Eloquent. Cela rend les requêtes complexes plus intuitives et lisibles. Par exemple :
  - Dans le modèle Booking.php, la méthode parking() définit une relation belongsTo(Parking::class), ce qui me permet d'accéder facilement aux informations du parking depuis une réservation ($booking->parking).
  - Inversement, dans le modèle Parking.php, la méthode bookings() définit une relation hasMany(Booking::class), me permettant de récupérer toutes les réservations associées à un parking ($parking->bookings).

Cette approche garantit que toutes les interactions avec la base de données sont sécurisées (Eloquent protège nativement contre les injections SQL) et maintenables.

2. Précisez les moyens utilisés :

- **Framework / ORM :** Laravel / Eloquent.
- **Langage :** PHP 8.
- **Outils de développement : PHPStorm** et son **terminal intégré** pour les commandes artisan (ex: php artisan make:model).

3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé en autonomie.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *La Plateforme*

Chantier, atelier, service ▸ *formation*

Période d'exercice ▸ Du : *23/09/2024* au : *15/10/2025*

5. Informations complémentaires *(facultatif)*

Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

Compétence n° 3 ▸ *Développer des composants métier coté serveur*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

J'ai structuré l'ensemble de la logique back-end de l'application "Parks" en suivant l'architecture **Modèle-Vue-Contrôleur (MVC)**, qui est au cœur de Laravel. Cette approche permet de séparer clairement les responsabilités : les modèles gèrent l'accès aux données, les vues (gérées par Inertia/React) s'occupent de l'affichage, et les contrôleurs orchestrent la logique métier.

- **Gestion des routes et des contrôleurs :** J'ai défini toutes les routes de l'application dans le fichier routes/web.php. J'ai utilisé les routes de type Route::resource pour générer automatiquement les routes standards (index, create, store, show, edit, update, destroy) pour chaque entité, comme pour les utilisateurs (UserController::class). Chaque route pointe vers une méthode spécifique d'un contrôleur, qui agit comme le point d'entrée de la logique métier.

- **Implémentation de la logique métier dans les contrôleurs :** Les contrôleurs, tels que UserController.php ou BookingController.php, contiennent le cœur de la logique applicative. Par exemple, dans la méthode store de UserController, j'exécute une séquence d'opérations métier :
  1. **Validation des données :** J'utilise la méthode $request->validate() pour m'assurer que les données reçues du formulaire front-end sont complètes et conformes aux règles définies (ex: l'email doit être unique, le mot de passe confirmé).
  2. **Interaction avec le modèle :** Si la validation réussit, j'interagis avec le modèle User pour créer un nouvel enregistrement en base de données, en prenant soin de crypter le mot de passe avec la façade Hash::make() pour la sécurité.
  3. **Réponse au client :** Enfin, je retourne une redirection vers la page de liste des utilisateurs avec un message de succès. Inertia se charge de transmettre cette réponse au front-end React sans rechargement de page.

- **Développement des composants de sécurité :** Pour la gestion des utilisateurs, je me suis appuyé sur le starter kit **Laravel Breeze**, qui fournit des contrôleurs dédiés et sécurisés pour l'authentification (inscription, connexion, réinitialisation de mot de passe), comme défini dans routes/auth.php. Ces contrôleurs constituent une brique métier essentielle pour la sécurisation de l'application.

2. Précisez les moyens utilisés :

- **Framework :** Laravel (Architecture MVC, Routage, Validation des requêtes).
- **Langage :** PHP 8.
- **Outils de développement : PHPStorm** et son **terminal intégré** pour l'exécution des commandes artisan (ex: php artisan make:controller).

3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé en autonomie.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *La Plateforme*

Chantier, atelier, service ▸ *formation*

Période d'exercice ▸ Du : *23/09/2024* au : *15/10/2025*

5. Informations complémentaires *(facultatif)*

Activité-type 2 — Développer la partie back-end d'une application web ou web mobile sécurisée

Compétence n° 4 ▸ Documenter le déploiement d'une application dynamique web ou web mobile

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Pour finaliser le cycle de vie du projet "Parks", j'ai préparé l'application pour une mise en production et documenté la procédure de déploiement sur le **serveur d'hébergement avec panneau de contrôle Plesk** fourni par mon centre de formation.

A) Préparation de l'application pour la production (actions réalisées) :

Avant de transférer les fichiers sur le serveur, j'ai compilé et optimisé l'application en local pour garantir les meilleures performances.

1. **Compilation des assets front-end :** J'ai exécuté la commande npm run build. Cette action lance **Vite.js** pour transpiler le code TypeScript/React et optimiser les fichiers CSS, générant des assets statiques dans le répertoire public/build.
2. **Optimisation des dépendances back-end :** J'ai utilisé la commande composer install --optimize-autoloader --no-dev pour installer uniquement les dépendances nécessaires à la production et optimiser le chargement des classes de Laravel.

B) Documentation de la procédure de déploiement sur Plesk :

J'ai rédigé un guide de déploiement dans le fichier README.md du projet, détaillant les étapes spécifiques à un environnement Plesk.

1. **Préparation de l'environnement Plesk :**
   - Création d'un sous-domaine dédié au projet (ex: parks.mondomaine.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 (sauf les dossiers node_modules et vendor) sur le serveur, via un client FTP ou le gestionnaire de fichiers de Plesk.
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 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, et en s'assurant que APP_DEBUG=false.
4. **Finalisation du déploiement :**
   - Exécution de la séquence de commandes artisan via le terminal SSH de Plesk pour finaliser l'installation :
     - php artisan key:generate

- php artisan config:cache
- php artisan route:cache
- php artisan migrate --force pour créer les tables dans la base de données de production.

Cette documentation structurée garantit un déploiement fiable et reproductible de l'application "Parks" dans un environnement d'hébergement web standard.

2. Précisez les moyens utilisés :

• **Panneau d'hébergement : Plesk**.

• **Outils de build :** Vite.js (via npm run build), Composer.

• **Gestion de version :** Git et GitHub pour le versionnage et le transfert du code.

• **Documentation :** Fichier README.md au format Markdown.

• **Serveur d'application :** Laravel (via les commandes artisan).

• **Outils de développement : PHPStorm** et son **terminal intégré** pour exécuter les commandes en local et se connecter en SSH.

3. Avec qui avez-vous travaillé ?

Ce projet a été réalisé en autonomie.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *La Plateforme*

Chantier, atelier, service ▸ *formation*

Période d'exercice ▸ Du : *23/09/2024* au : *15/10/2025*

5. Informations complémentaires *(facultatif)*

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

Déclaration sur l'honneur

Je soussigné(e) TRIFI Aïmen ,

déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis l'auteur(e) des réalisations jointes.

Fait à Mougins le 07/10/2025

pour faire valoir ce que de droit.

Signature :

Documents illustrant la pratique professionnelle

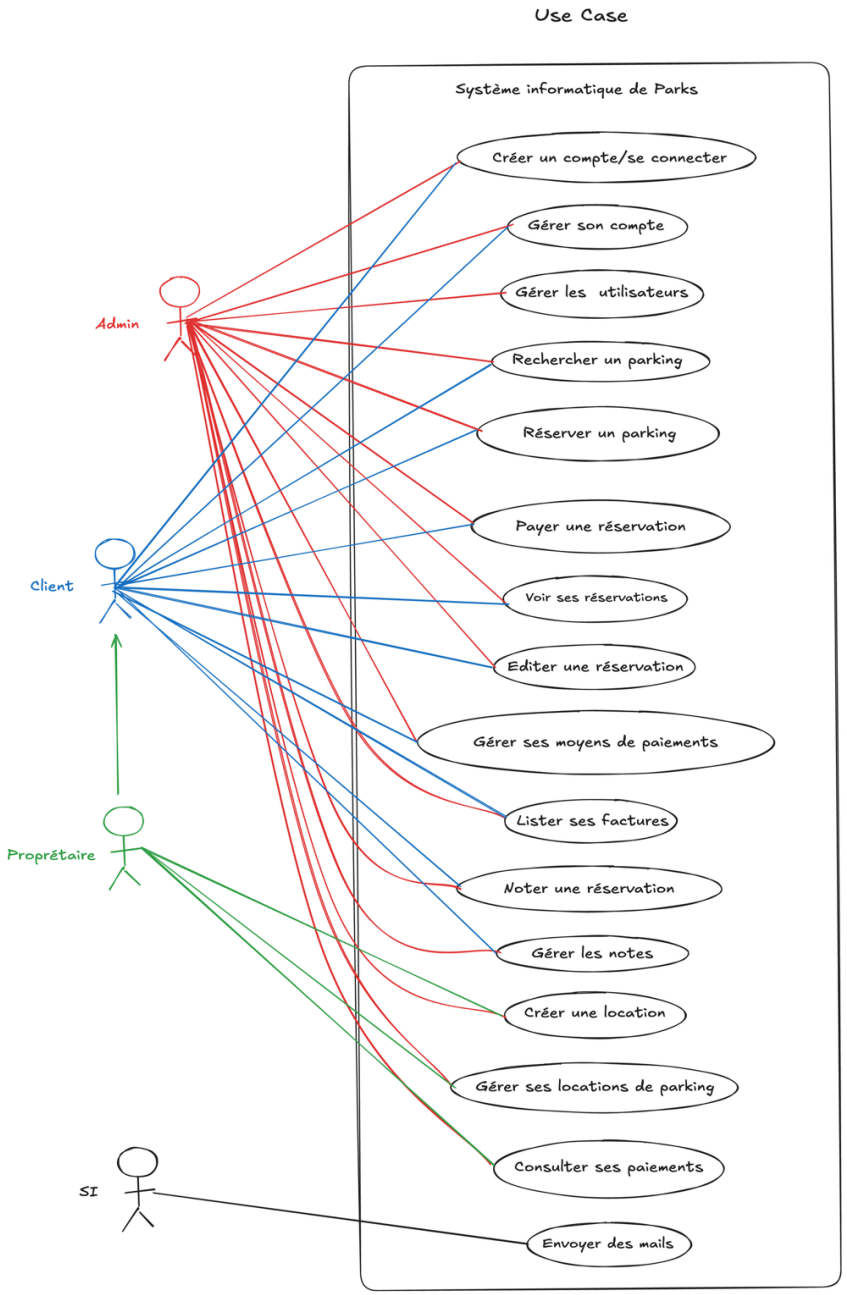
(facultatif)

Intitulé
Cliquez ici pour taper du texte.

ANNEXES

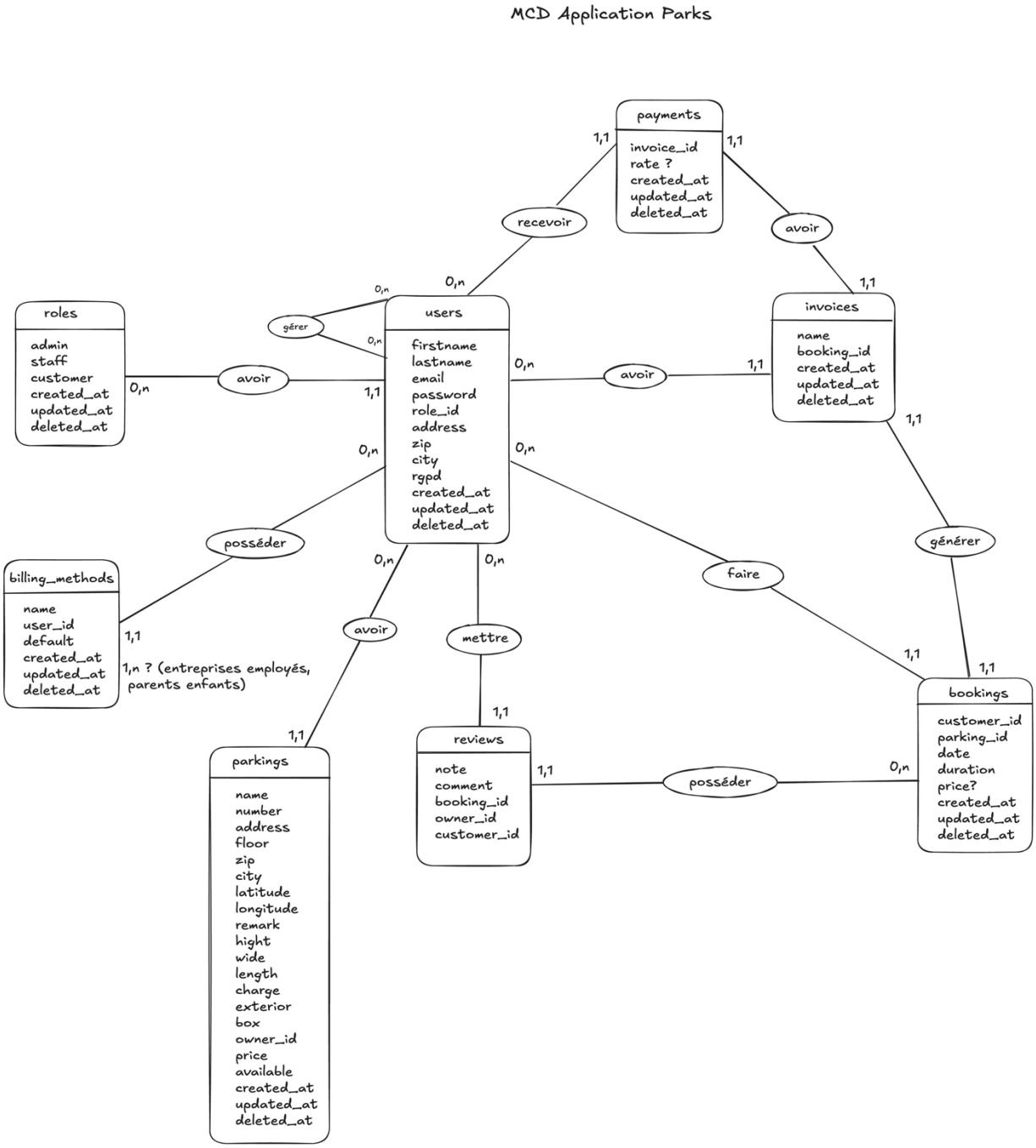
I./ Conception Fonctionnelle et Données :

- Figure 1.1 : Diagramme de cas d'utilisation (Use Case) de l'application Parks



ANNEXES

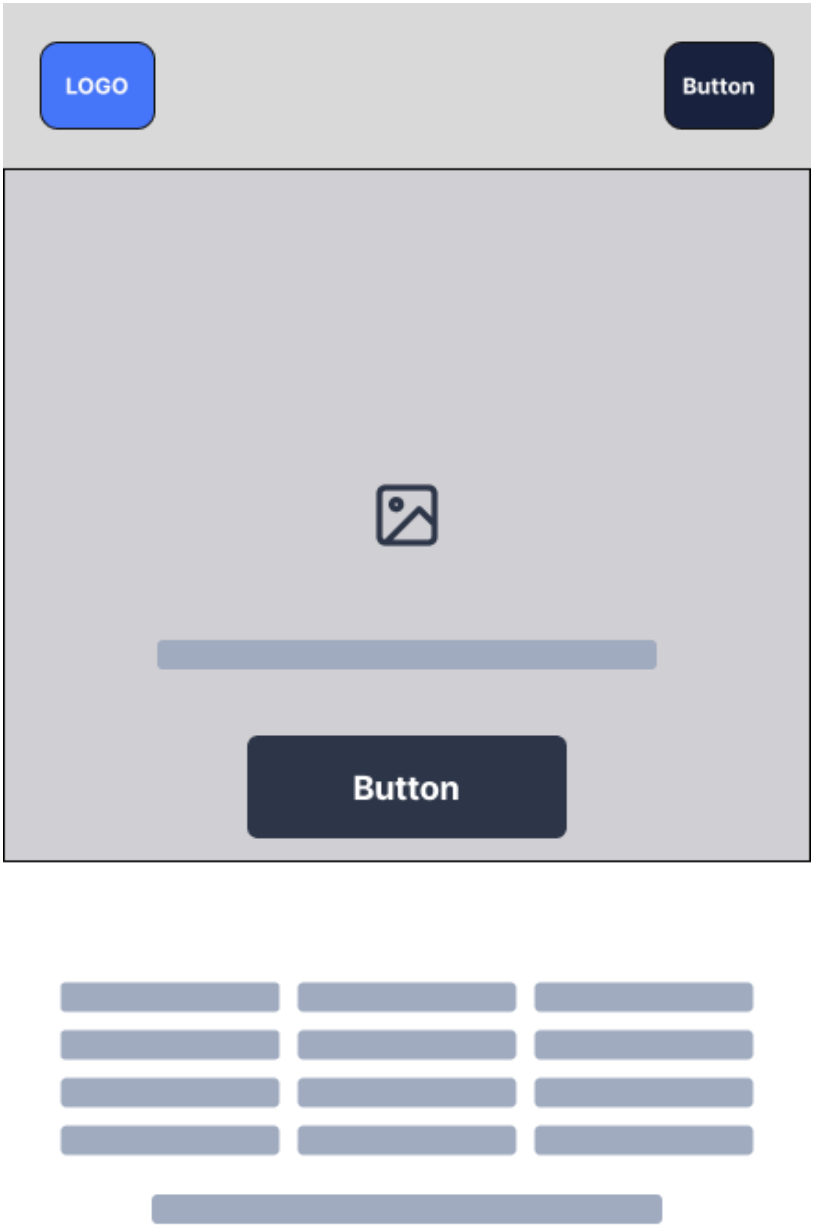
- Figure 1.2 : Modèle Conceptuel de Données (MCD) du projet



ANNEXES

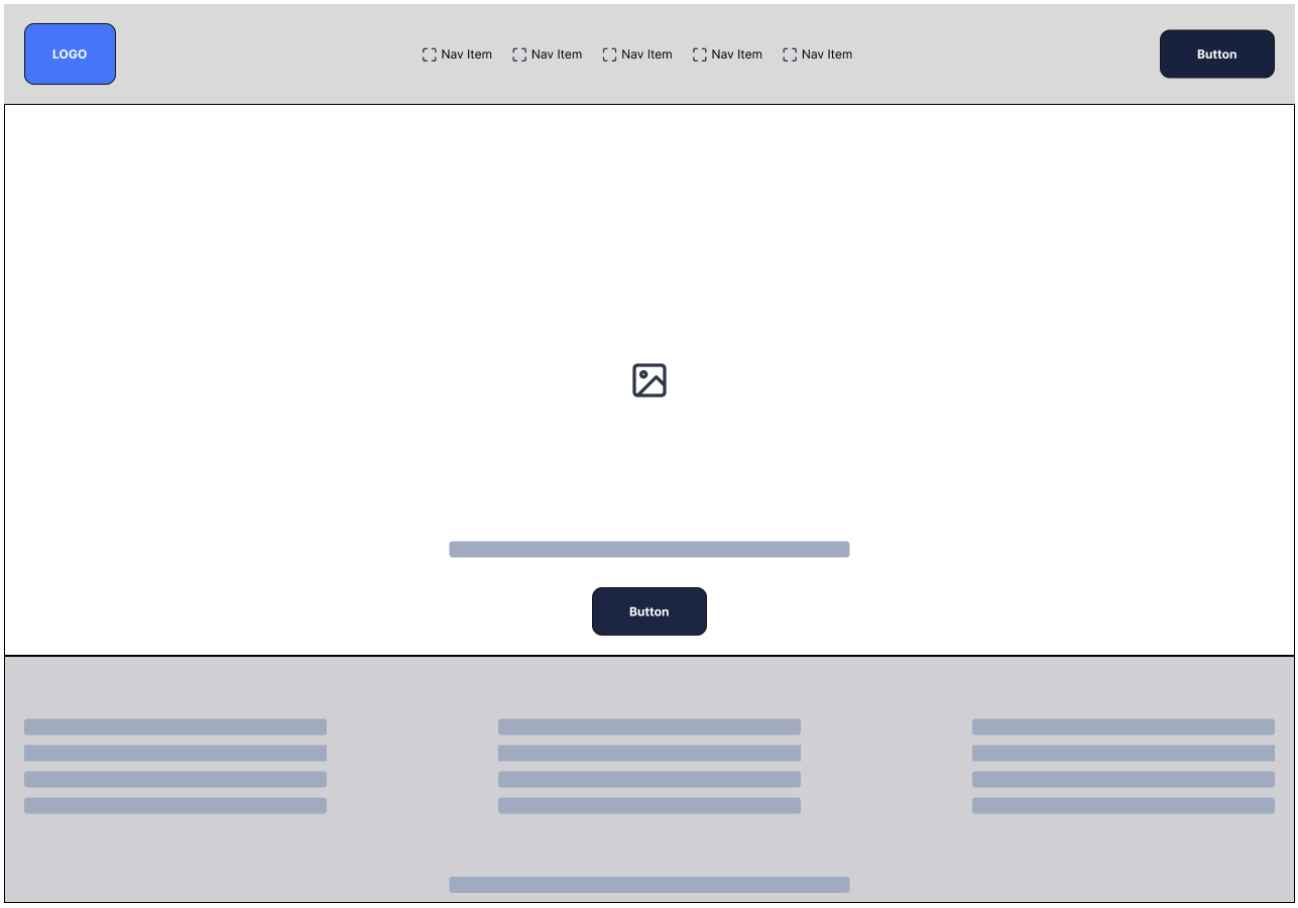
II./ Wireframes (Figma)

- Figure 2.1 : Wireframe de la page d'accueil - Vue Mobile



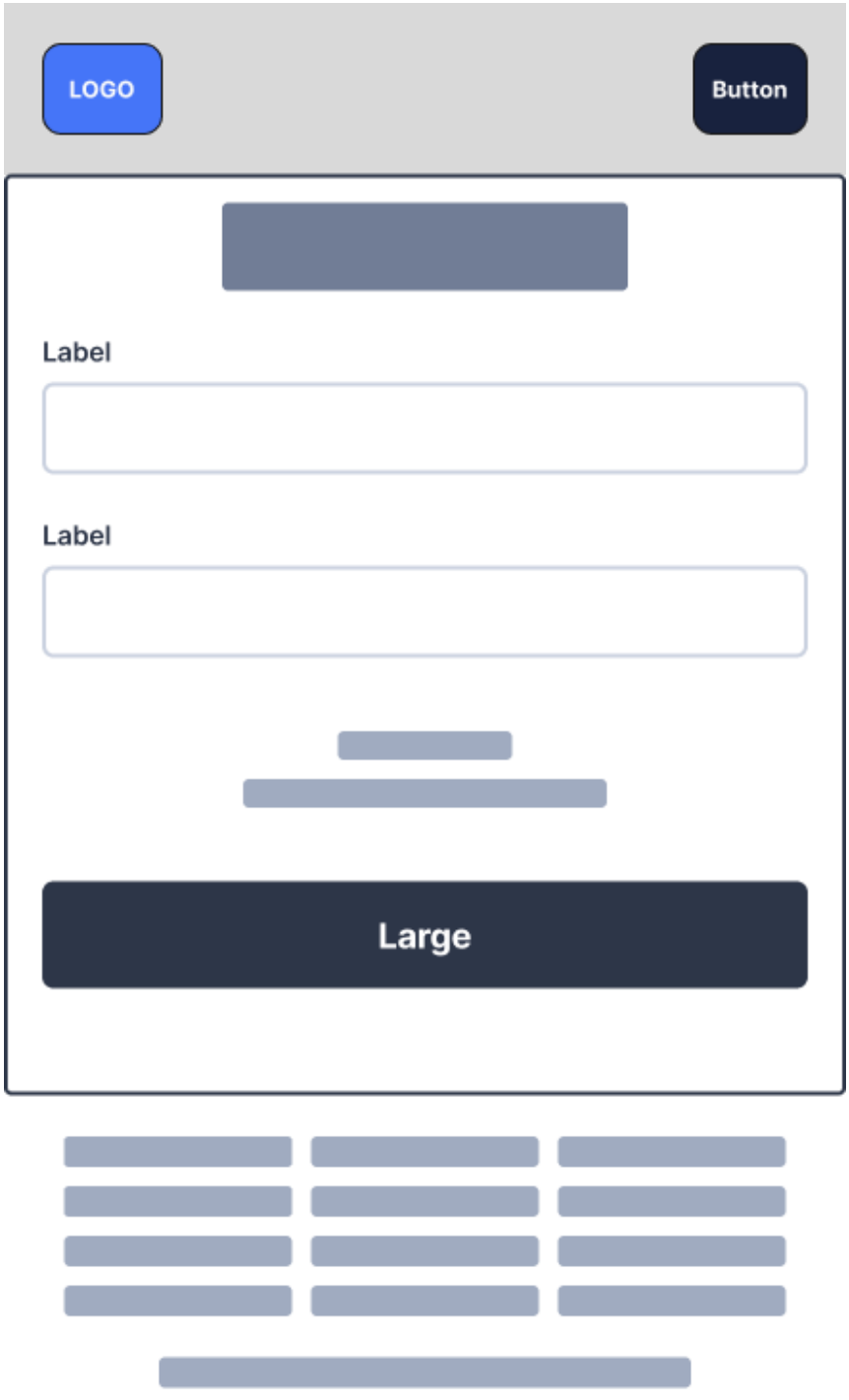
ANNEXES

- Figure 2.2 : Wireframe de la page d'accueil - Vue Desktop



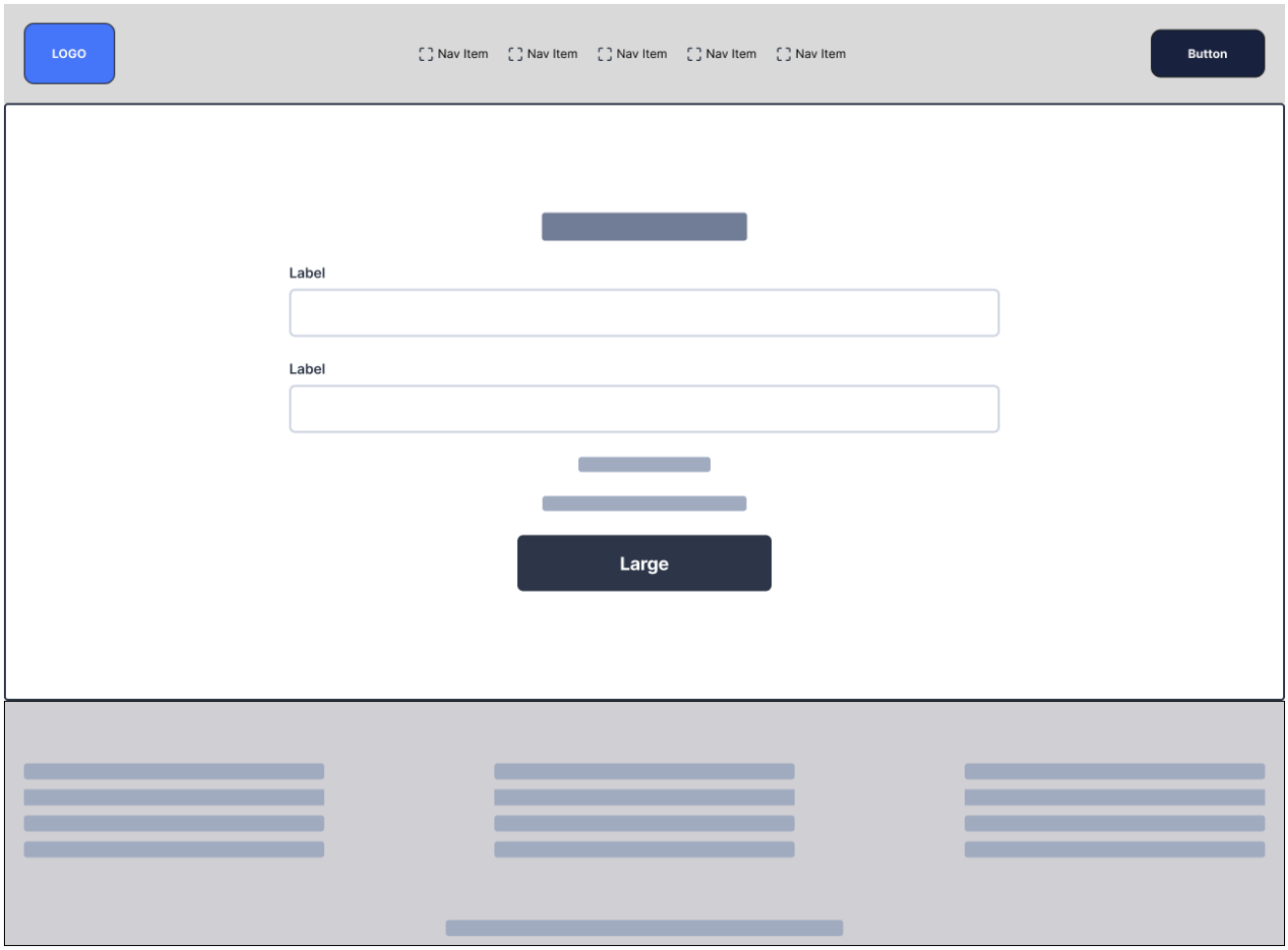
ANNEXES

- Figure 2.3 : Wireframe de la page de login - Vue Mobile



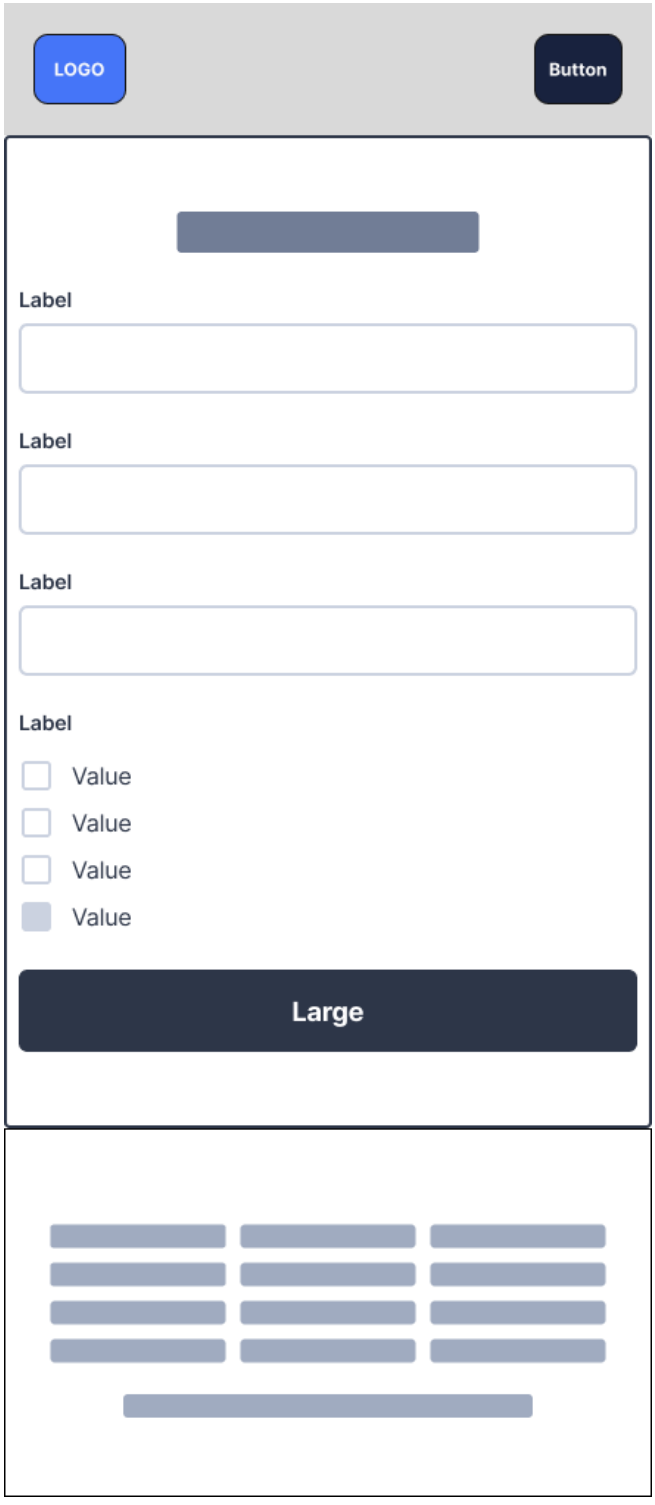
ANNEXES

- Figure 2.4 : Wireframe de la page de login - Vue Desktop



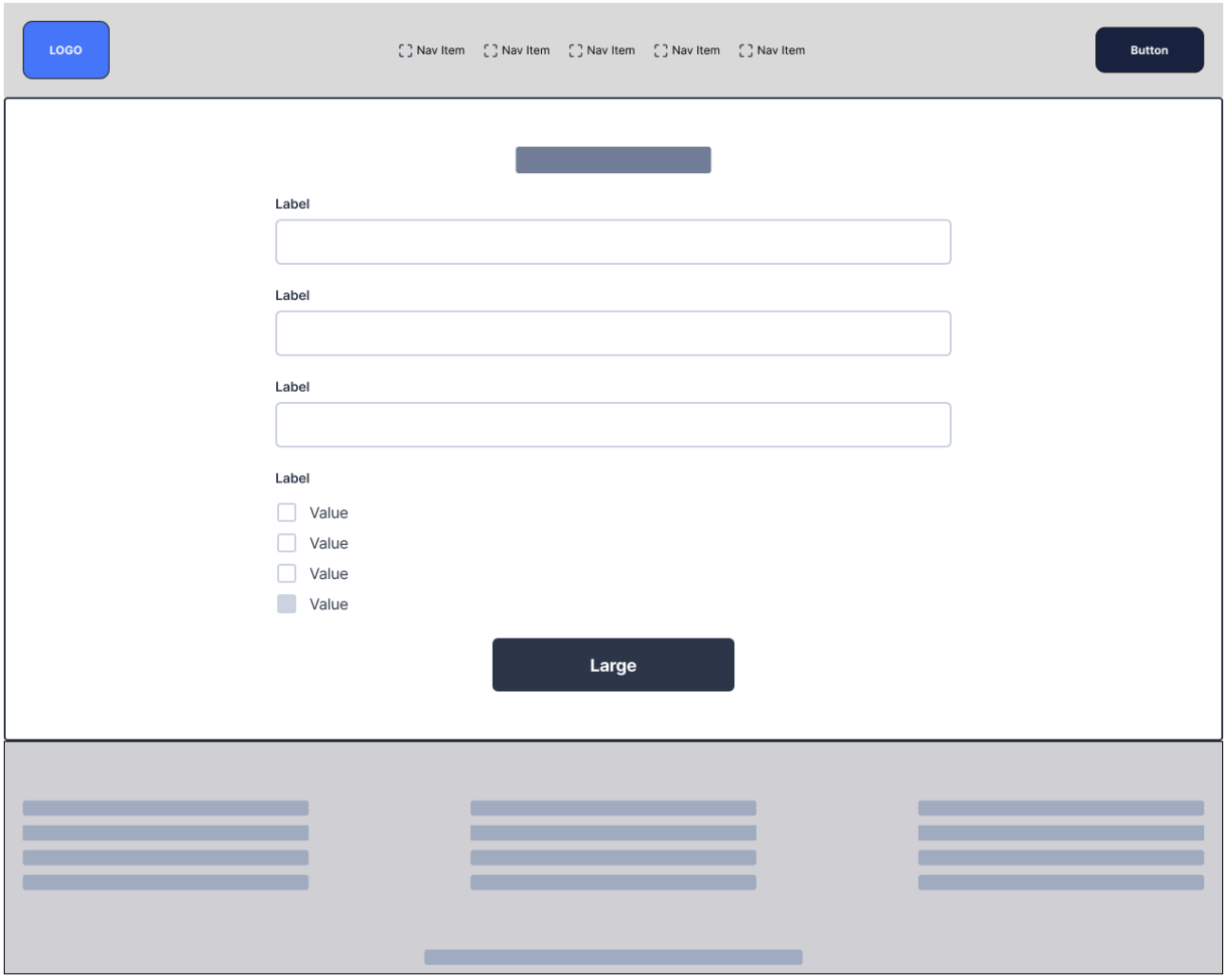
ANNEXES

- Figure 2.5 : Wireframe de la page de recherche - Vue Mobile



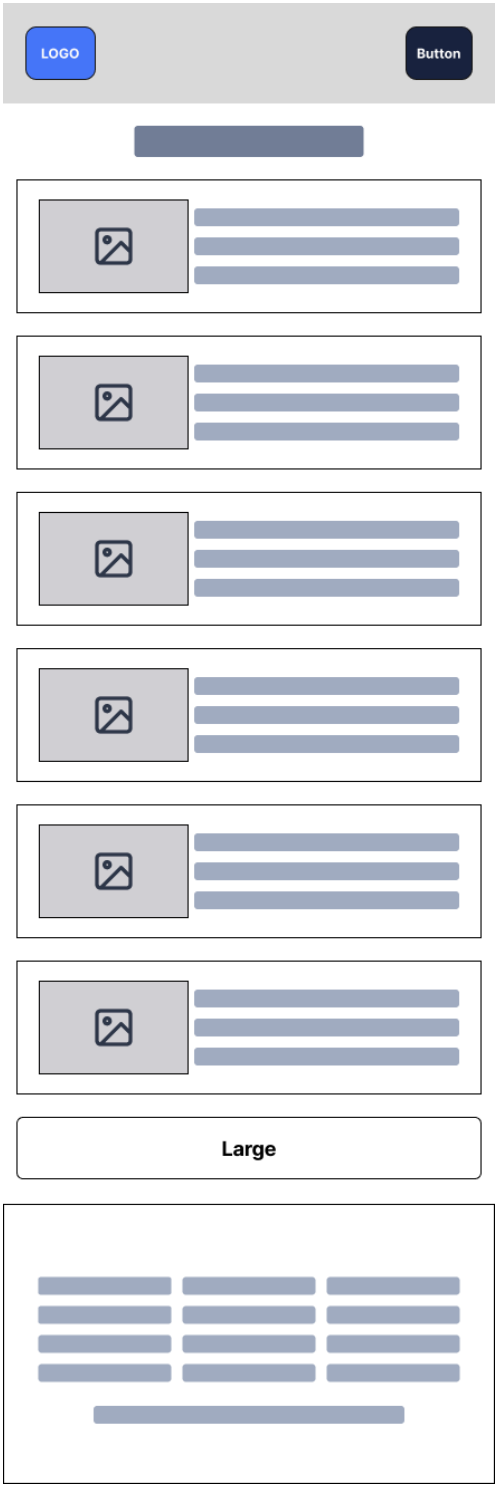
ANNEXES

- Figure 2.5 : Wireframe de la page de recherche - Vue Desktop



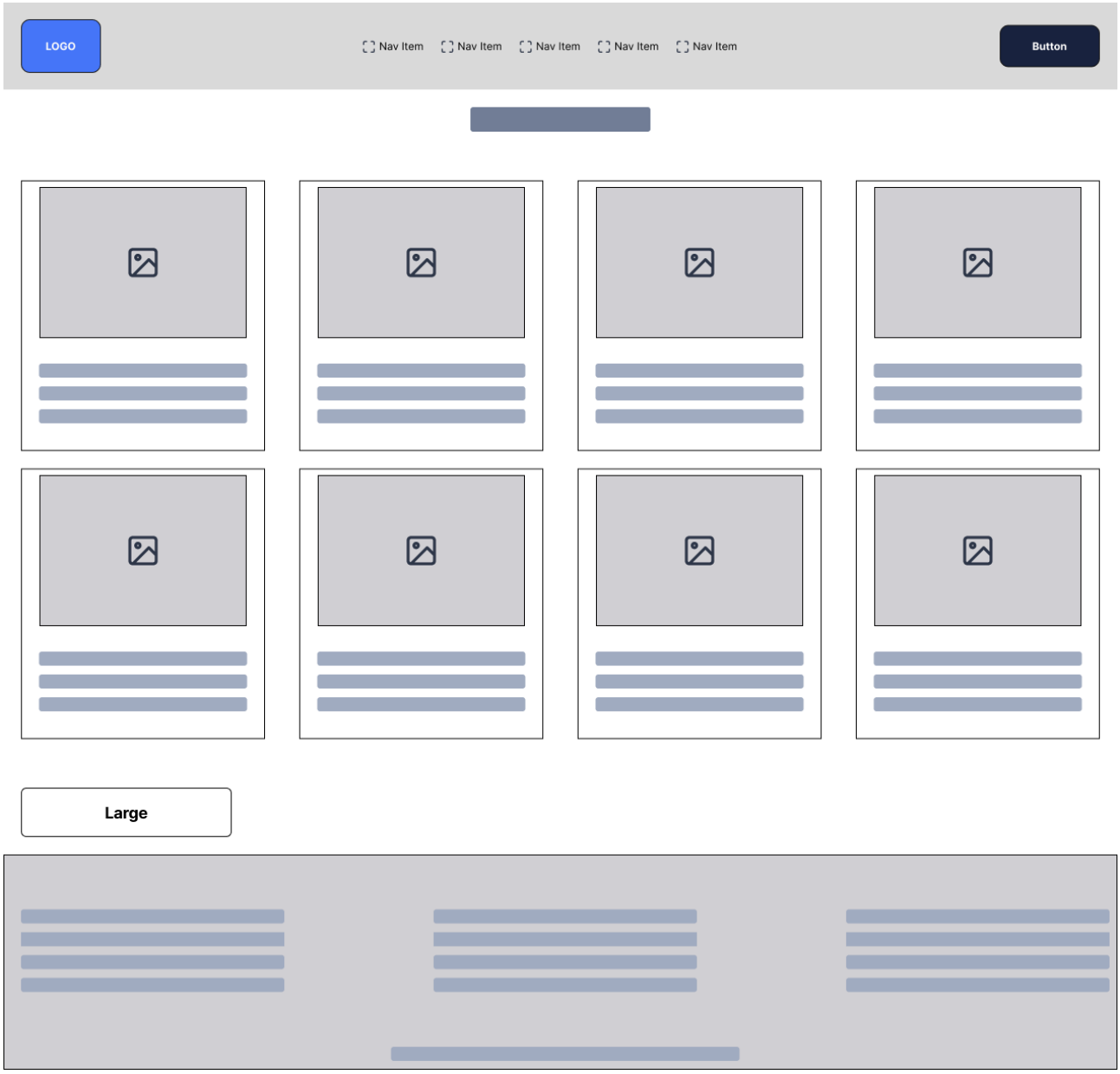
ANNEXES

- Figure 2.6 : Wireframe de la page des résultats de recherche - Vue Mobile



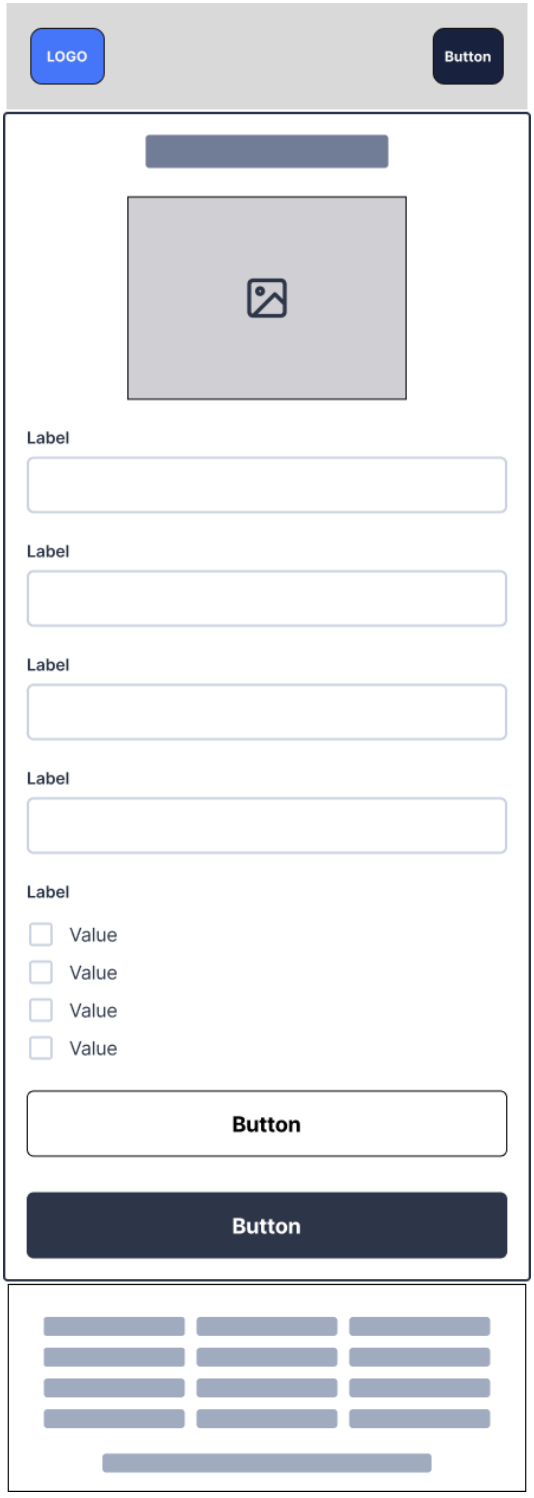
ANNEXES

- Figure 2.7 : Wireframe de la page des résultats de recherche - Vue Desktop



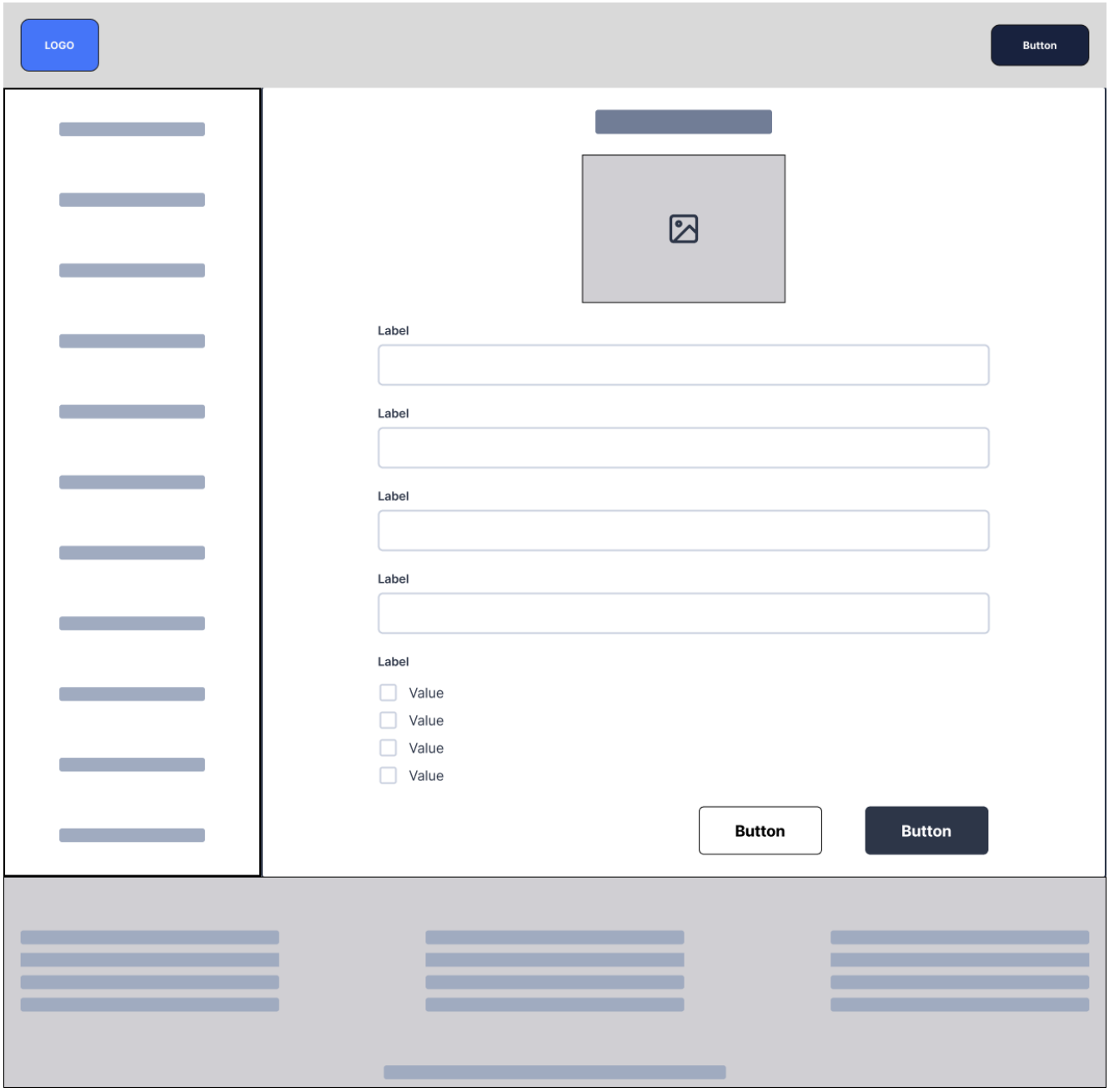
ANNEXES

- Figure 2.8 : Wireframe de la page d'ajout d'un parking - Vue mobile



ANNEXES

- Figure 2.9 : Wireframe de la page d'ajout d'un parking - Vue Desktop



ANNEXES

III./ Environnement et Développement Back-end

- Figure 3.1 : Lancement de l'environnement de développement complet à l'aide de la commande ***composer run dev***.

```
aimen@mac parks % composer run dev
> Composer\Config::disableProcessTimeout
> npx concurrently -c "#93c5fd,#c4b5fd,#fb7185,#fdba74" "php artisan serve" "php artisan queue:listen --tries=1" "php artisan pail --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 781 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 dans le fichier ***.env***

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

ANNEXES

- Figure 3.2 : Exemple de fichier de migration Laravel pour la création de la table parkings

```
Aimen *
public function up(): void
{
    Schema::create( table: 'parkings', function (Blueprint $table) {
        $table->id();
        $table->string( column: 'name');
        $table->integer( column: 'number')->nullable();
        $table->string( column: 'address');
        $table->string( column: 'floor')->nullable();
        $table->string( column: 'zip', length: 5);
        $table->string( column: 'city');
        $table->decimal( column: 'latitude', total: 10, places: 8)->nullable();
        $table->decimal( column: 'longitude', total: 11, places: 8)->nullable();
        $table->text( column: 'remark')->nullable();
        $table->float( column: 'height');
        $table->float( column: 'width');
        $table->float( column: 'length');
        $table->boolean( column: 'charge')->default( value: false);
        $table->boolean( column: 'exterior')->default( value: false);
        $table->boolean( column: 'box')->default( value: false);
        $table->foreignId( column: 'owner_id')
            ->constrained( table: 'users')
            ->restrictOnDelete()
            ->cascadeOnUpdate();
        $table->integer( column: 'price_per_hour');
        $table->boolean( column: 'available')->default( value: true);
        $table->timestamps();
        $table->softDeletes();
        $table->index( columns: 'city');
        $table->index( columns: 'zip');
    });
}
```

ANNEXES

- Figure 3.4 : Exemple de Modèle Eloquent ***Parking.php*** avec ses relations

```php
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);
    }
```

ANNEXES

- Figure 3.5 : Exemple de logique métier dans un contrôleur ***UserController.php*** avec la sauvegarde d'un utilisateur après validation des champs et hashage du mot de passe.

```php
class UserController extends Controller
    Aïmen *
    public function store(Request $request)
    {
        $validatedData = $request->validate([
            'firstname' => 'required|string|max:255',
            'lastname' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:8|confirmed',
            'role_id' => 'required|integer|exists:roles,id',
            'address' => 'nullable|string|max:255',
            'zip' => 'nullable|string|max:10',
            'city' => 'nullable|string|max:255',
            'rgpd' => 'nullable|boolean',
        ]);
        try {
            User::create([
                'firstname' => $validatedData['firstname'], // Correction: firstname -> firstname
                'lastname' => $validatedData['lastname'],   // Correction: lastname -> lastname
                'email' => $validatedData['email'],
                'password' => Hash::make($validatedData['password']), // Utilisation de Hash::make()
                'role_id' => $validatedData['role_id'], // Correction: role_id
                'address' => $validatedData['address'],
                'zip' => $validatedData['zip'],
                'city' => $validatedData['city'],
                'rgpd' => $validatedData['rgpd'] ?? false,
                'email_verified_at' => null, // Ajout du champ si nécessaire
            ]);
            return redirect()->route( route: 'users.index')->with('success', 'Utilisateur créé avec succès.');

        } catch (\Exception $e) {
            return back()->withInput()->withErrors(['error' => 'Erreur lors de la création de l\'utilisateur: ' . $e->getMessage()]);
        }
    }
```