

Chapitre 1 — Introduction générale

1.1. Contexte et objectifs du projet

Le développement du e-commerce connaît une croissance continue, notamment dans des secteurs de niche comme la vente de maillots de football personnalisés.

Le projet « Maillot de Foot » s'inscrit dans le cadre de ma formation Développeur Web et Web Mobile. Il a pour objectif de mettre en pratique l'ensemble des compétences du référentiel REAC à travers la conception et le développement d'une application web moderne.

L'application a été pensée comme une boutique e-commerce spécialisée dans les maillots de football. J'ai effectué ce choix car il m'est apparu en tant qu'utilisateur de ce type de site, qu'il me serait plus aisément à reproduire, tout d'abord en m'inspirant d'eux pour comprendre la logique dans la conception d'un site e-commerce mais également parce que par leur fréquentation j'ai pu cerner les limites de certains d'entre eux ce qui me semblait une promesse d'améliorations possibles.

Ce choix thématique répond aussi à un cas d'usage courant (site marchand), tout en offrant une base pédagogique suffisamment conséquente pour manipuler des concepts variés : catalogue de produits, gestion du panier, comptes utilisateurs, adresses. L'enjeu principal était de créer un site complet permettant aux utilisateurs de :

- consulter un catalogue de clubs et de maillots.
- afficher les détails de chaque produit.,
- ajouter des articles dans un panier.
- gérer un compte utilisateur et ses adresses.
- simuler un processus de commande.

Ce projet nécessite la maîtrise conjointe du front-end et du back-end, ainsi que l'intégration des aspects liés à la sécurité, à l'ergonomie et à la gestion de projet.

1.2. Enjeux utilisateurs et métier

Du point de vue métier, l'application doit offrir une expérience de type e-commerce adaptée à un public ciblé (supporters et/ou amateurs de football, collectionneurs de maillots).

Du point de vue utilisateur, les enjeux étaient :

- Simplicité d'utilisation : navigation intuitive, interface claire, parcours d'achat fluide.
- Rapidité et réactivité : grâce à Inertia et React, qui évitent les rechargements complets de page.
- Accessibilité : compatibilité avec divers supports (desktop et mobile).
- Sécurité : gestion fiable des comptes et sessions utilisateurs, protection des données personnelles.

1.3. Périmètre et limites du projet

Le périmètre de l'application comprend :

- La gestion des clubs et maillots (catalogue et fiches produits).
- L'affichage des fiches produits avec images.
- La gestion du panier (ajout, mise à jour, suppression, vidage).
- La gestion d'un compte utilisateur (authentification, adresses).
- La simulation d'un processus de commande.

En revanche, certaines fonctionnalités n'ont pas pu être implémentées dans cette version par manque de temps :

- La gestion avancée d'un espace administrateur.
- L'intégration d'un véritable système de paiement en ligne.
- La mise en place d'un module de livraison complet.
- La possibilité pour l'utilisateur d'ajouter des maillots dans une wish list.

1.4. Méthode et organisation

Le projet a été mené de manière incrémentale et agile :

- Découpage en fonctionnalités prioritaires (MVP : catalogue, panier, compte utilisateur). J'ai dans un premier temps concentré mon effort à la mise en place des fonctionnalités de base. J'ai débuté par la création du compte utilisateur côté client (enregistrement, connexion, détails du compte, adresses). J'ai ensuite travaillé sur la partie catalogue pour terminer avec le panier.

Le **MVP** (*Minimum Viable Product*, ou « produit minimum viable » en français) est une méthode de développement qui consiste à **définir et réaliser uniquement les fonctionnalités essentielles** pour que l'application soit utilisable par les premiers utilisateurs. L'idée consiste à ne pas tout développer dès le départ mais à se concentrer sur un noyau fonctionnel minimal. Par la suite, on pourra enrichir le projet de fonctionnalités supplémentaires.

Cette façon de procéder permet d'obtenir rapidement une première version utilisable. Les utilisateurs peuvent tester le site et émettre des retours. J'ai ainsi pu valider les choix technologiques et la faisabilité technique de mon projet :

- Mise en place progressive du front-end et du back-end.
- Utilisation de Git pour la gestion des versions (versioning) et la collaboration (avec moi-même il est vrai mais il m'est apparu nécessaire de créer des branches afin de procéder à des essais sans affecter le code initial).
- Tests manuels réguliers pour valider chaque étape du parcours des utilisateurs.

Cette méthodologie m'a permis de conserver une vision produit claire tout en respectant les contraintes pédagogiques.

J'ai également scindé la conception de mon travail en deux parties. Je me suis dans un premier temps concentré sur la partie liée au compte utilisateur côté client. Quand il m'a paru que les fonctionnalités essentielles du compte utilisateur côté client (connexion, enregistrement, détails du compte, adresse) avaient été développées, j'ai estimé que je devais désormais m'occuper de la partie produit du site, avec le développement de la pages listant les maillots des clubs et des équipes nationales, la page référençant le maillot sélectionné par l'utilisateur et la page panier.

Je n'ai pas encore développé la partie administrateur cependant je pense établir un rôle en utilisant un middleware (prévu dans le MCD, MLD, MPD) avec des autorisations étendues par rapport à un utilisateur lambda, lui offrant la possibilité de gérer le site (commandes, stock, ajout/retrait des produits, modification produit, accès aux comptes utilisateurs).

L'utilisation d'un middleware dans le cadre de mon projet contribuera pour ce rôle d'administrateur :

- **à la centralisation de la sécurité** : Un middleware permet d'appliquer un contrôle d'accès à tous les points d'entrée (contrôleurs, routes) de la partie administrateur sans devoir répéter les vérifications dans chaque fonction. Cela constitue la solution la plus sûre et la plus propre.
- **Évolutif et maintenable** : Je pourrai faire évoluer la gestion des rôles facilement (plusieurs rôles ou permissions) sans toucher à la logique métier de chaque page.
- **Bonne pratique Laravel** : Cela suit les conventions et recommandations et offre la possibilité de réutiliser la structure pour d'autres rôles/parties privées ultérieurement.

Chapitre 2 — Vue d'ensemble technique

2.1. Stack technologique

Le projet repose sur une stack moderne et cohérente :

- Laravel 11 (PHP 8.2) : framework¹ back-end, gestion du routage, logique métier, accès BDD via Eloquent ORM.
- Inertia.js : pont entre Laravel et React, permettant de construire une Single Page Application sans API REST explicite.
- React : framework front-end, création d'interfaces dynamiques et réactives.
- Tailwind CSS : framework CSS utilitaire, garantissant un design responsive et homogène.
- Vite : outil de build rapide et moderne pour le front-end.
- MySQL : système de gestion de base de données relationnelle.

2.2. Architecture de l'application

L'architecture suit le modèle MVC (modèle, vue, contrôleur) de Laravel, enrichi par l'intégration Inertia/React :

¹ Framework : un framework est un ensemble de composants logiciels réutilisables qui permettent de développer de nouvelles applications plus efficacement

- Côté serveur (Laravel)

- *Models* : représentent les entités métier (User, Club, Maillot, Cart, Address).
- *Controllers* : assurent la logique métier (ex. CartController pour gérer le panier).
- *Migrations* : définissent la structure des tables et leurs relations.
- *Routes* : exposent les points d'entrée (parcours public, parcours utilisateur connecté).

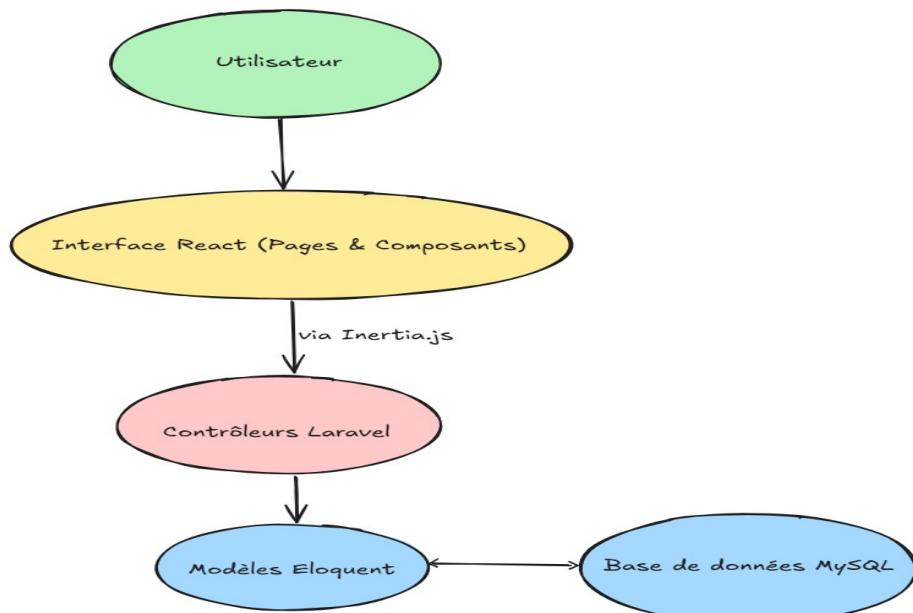
- Côté client (React via Inertia)

- *Pages* : chaque route est liée à une page React (Home, MaillotDetail, Cart, AccountDetails...).
- *Composants* : éléments réutilisables (Header, Footer, WelcomeMessage, PanierLink).
- *Styles* : Tailwind pour la mise en forme et la responsivité.

Cette organisation permet :

- une séparation claire entre logique serveur et interface,
- une expérience fluide côté utilisateur (navigation sans rechargement complet),
- une évolutivité (possibilité d'ajouter de nouvelles pages et composants facilement).

Schéma global d'architecture



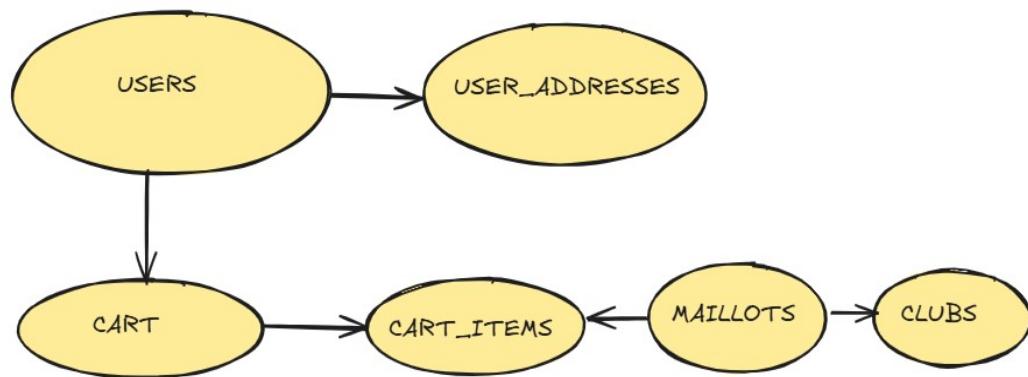
Front-end (React) : gère l'affichage et l'interaction utilisateur.

Inertia.js : transmet les données entre Laravel et React sous forme de props.

Back-end (Laravel) : applique la logique métier, valide les données et interagit avec la base.

Base de données (MySQL) : stocke les informations persistantes (utilisateurs, clubs, maillots, adresses, paniers).

Schéma de base de données (simplifié)



Explication des relations :

- Un User peut avoir plusieurs Adresses (relation N-N via user_addresses).
- Un User possède un Cart (panier).
- Un Cart contient plusieurs CartItems (articles panier), chacun lié à un Maillot.
- Un Maillot appartient à un Club.

2.3. Qualité et sécurité

La sécurité et la robustesse de l'application ont été considérées comme des priorités dès la conception. Plusieurs mécanismes intégrés à Laravel et aux outils front-end assurent la fiabilité du système :

- **Authentification** : le middleware auth protège l'accès aux zones privées (compte, adresses, panier). Seuls les utilisateurs connectés peuvent accéder à ces fonctionnalités.

```
app > Http > Middleware > AuthSessionMiddleware.php > ...
6  use Illuminate\Http\Request;
7  use Symfony\Component\HttpFoundation\Response;
8
9  class AuthSessionMiddleware
10 {
11     /**
12      * Handle an incoming request.
13      *
14      * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
15      */
16     public function handle(Request $request, Closure $next): Response
17     {
18         $sessionId = $request->cookie('session-id');
19
20         if (!$sessionId) {
21             return redirect()->route('Login');
22         }
23
24         $session = \App\Models\UserSession::with('user')
25             ->where('id', $sessionId)
26             ->where('expires_at', '>', now())
27             ->first();
28
29         if (!$session) {
30             return redirect()->route('Login')->withoutCookie('session-id');
31         }

```

- **Sessions utilisateurs** : un suivi des connexions est assuré grâce à la table dédiée user_sessions, permettant de tracer les activités et de renforcer le contrôle d'accès.

```
app > Models > UserSession.php > UserSession
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6  use Illuminate\Database\Eloquent\Relations\BelongsTo;
7
8  class UserSession extends Model
9  {
10     protected $fillable = [
11         'id',
12         'user_id',
13         'expires_at',
14         'ip_address',
15         'user_agent',
16         'last_activity',
17     ];
18
19     protected $casts = [
20         'expires_at' => 'datetime',
21         'last_activity' => 'datetime',
22         'created_at' => 'datetime',
23     ];
24
25     public $incrementing = false;
26     protected $keyType = 'string';

```

- **Validation des données** : les formulaires sont systématiquement validés côté serveur (Laravel) et côté client (React). Cela réduit les risques d'erreurs et empêche l'injection de données malveillantes.
- **Protection CSRF** : activée par défaut dans Laravel, elle sécurise tous les formulaires en générant un jeton unique associé à chaque session.
- **Gestion des relations en base** : les contraintes d'intégrité référentielle (clés primaires et étrangères) sont mises en place via les migrations, garantissant la cohérence des données.

Au-delà de ces mécanismes techniques, le projet applique plusieurs **bonnes pratiques de développement** destinées à renforcer la maintenabilité et la lisibilité du code :

- **Respect du standard PSR-12** : le standard **PSR-12** (PHP Standards Recommendations) fournit des règles de style pour le langage PHP : indentation, noms de variables et méthodes, organisation des classes. L'objectif est d'obtenir un **code homogène et lisible**, facilitant la relecture et la collaboration entre développeurs.
- **Utilisation d'Eloquent ORM** : Plutôt que d'écrire des requêtes SQL brutes, le projet s'appuie sur Eloquent ORM (Object Relational Mapping).

Les Avantages d'Eloquent ORM :

- **Maintenance** : les évolutions de la base (ex. ajout d'un champ) sont facilement prises en charge via les migrations et les modèles.
- **Sécurité** : les injections SQL sont évitées grâce à la protection native de Laravel.
- **Lisibilité** : les relations sont exprimées sous forme d'objets (`$user->useraddresses`, `$club->maillots`, `$cart->items`), ce qui rend le code plus compréhensible.

`$user->addresses`: permet de récupérer toutes les adresses liées à un utilisateur. Dans modèle User.php :

```
app > Models > User.php > User
16  class User extends Authenticatable
39  protected $casts = [
40      'email_verified_at' => 'datetime',
41      'birth_date' => 'date',
42      'is_active' => 'boolean',
43      'created_at' => 'datetime',
44      'updated_at' => 'datetime',
45  ];
46
47  // Relations
48  public function sessions(): HasMany
49  {
50      return $this->hasMany(UserSession::class);
51  }
52
53  public function addresses(): HasMany
54  {
55      return $this->hasMany(UserAddress::class);
56  }
```

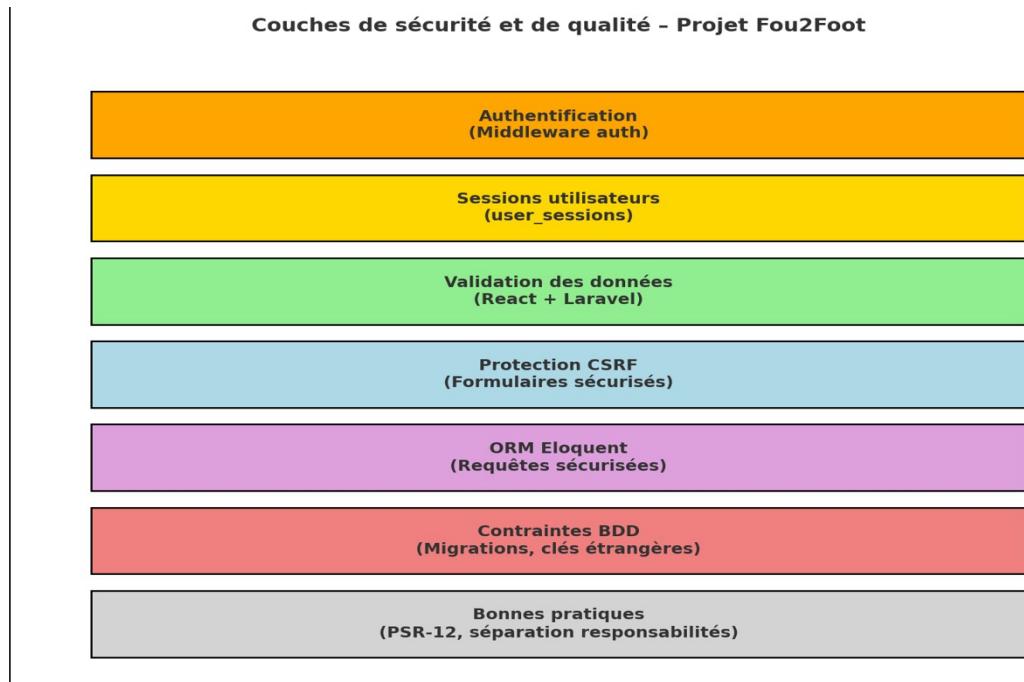
\$club → maillots : permet de récupérer tous les maillots associés à un club.

Dans modèle Club.php :

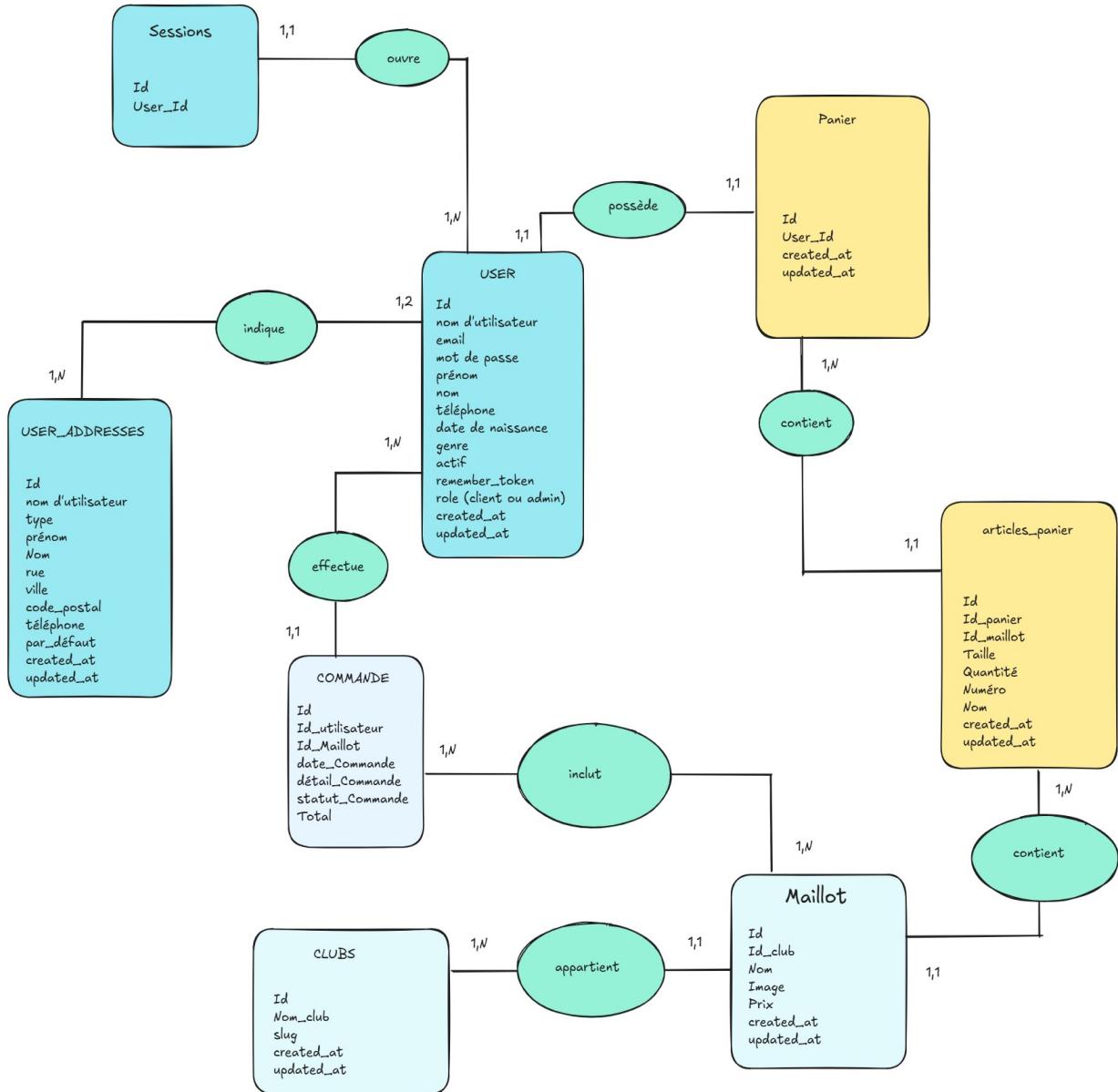
```
app > Models > Club.php > Club
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Club extends Model
8  {
9      public function maillots()
10     {
11         return $this->hasMany(Maillot::class);
12     }
13 }
14
```

- **Séparation des responsabilités** : l'architecture du projet suit le principe *Separation of Concerns*. Les contrôleurs gèrent la logique applicative, les modèles manipulent les données, et les composants React/Tailwind assurent la présentation. Cette structuration rend le projet plus facile à maintenir et à tester.

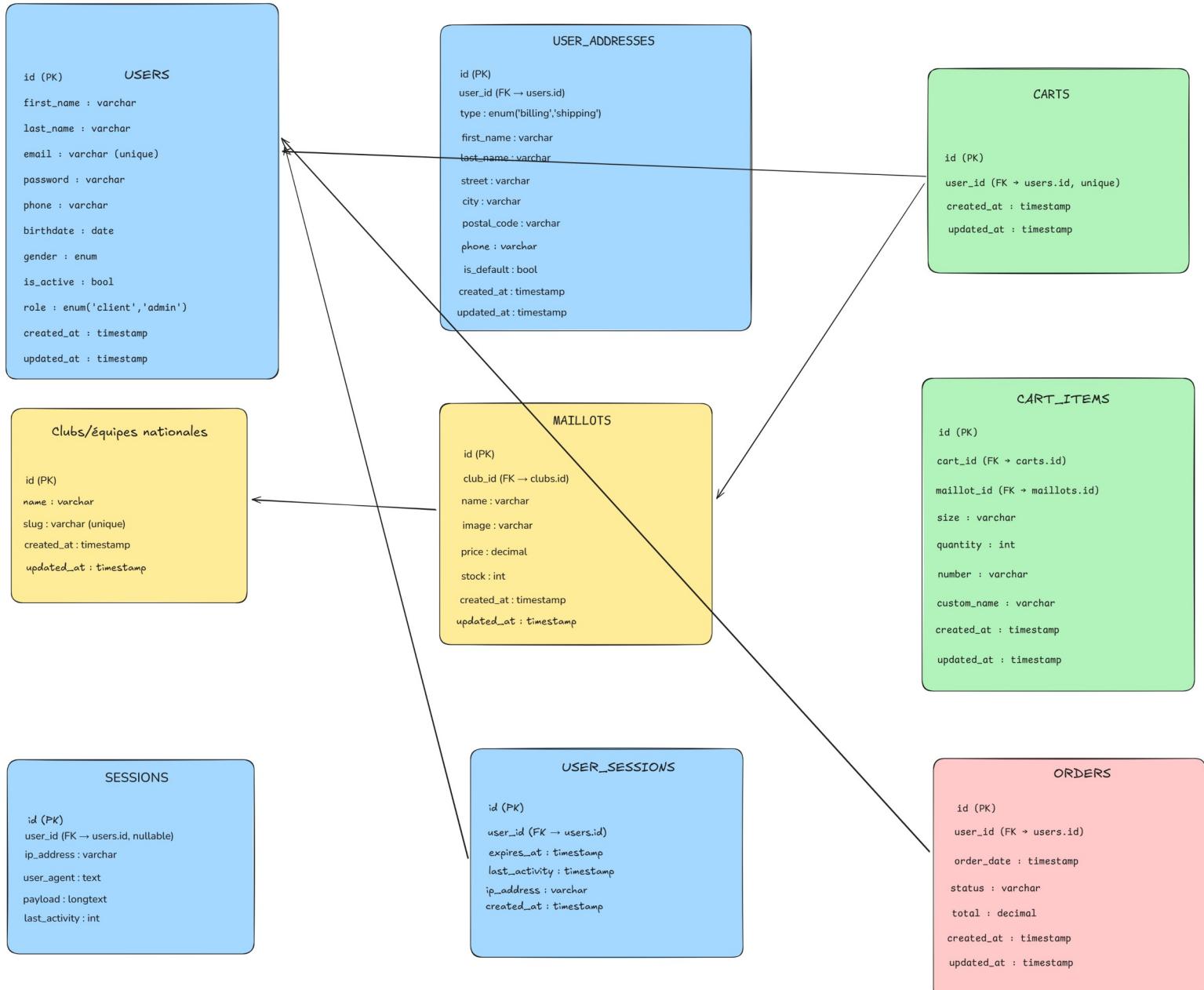
En combinant **mécanismes de sécurité natifs** et **bonnes pratiques de développement**, l'application bénéficie d'un socle fiable, sécurisé et durable, garantissant à la fois la protection des données et la facilité d'évolution du projet.



MCD



MLD



Structure du MPD

Utilisateur (users)

- **Id** : bigint UNSIGNED, PRIMARY KEY, AUTO_INCREMENT
- **username** : varchar(50), UNIQUE
- **email** : varchar(191), UNIQUE
- **password** : varchar(191)
- **first_name** : varchar(100) (nullable)
- **last_name** : varchar(100) (nullable)
- **phone** : varchar(20) (nullable)
- **birth_date** : date (nullable)
- **gender** : enum('male','female','other') (nullable)
- **email_verified_at** : timestamp (nullable)
- **is_active** BOOLEAN DEFAULT TRUE,
- **role** ENUM('client','admin') NOT NULL DEFAULT 'client',
- **remember_token** : varchar(100) (nullable)
- **created_at/updated_at** : timestamp (nullable)
- **Indexes** : email, username, is_active

Adresse utilisateur (user_addresses)

- **id** : bigint UNSIGNED, PRIMARY KEY, AUTO_INCREMENT
- **user_id** : bigint UNSIGNED, FOREIGN KEY (users.id)
- **type** : enum('billing','shipping')
- **first_name/last_name** : varchar(100)
- **street** : text
- **city** : varchar(100)

- **postal_code** : varchar(20)
- **country** : varchar(2) DEFAULT 'FR'
- **phone** : varchar(20) (nullable)
- **is_default** : tinyint(1) DEFAULT 0
- **created_at/updated_at** : timestamp (nullable)
- **Indexes** : user_id, type, is_default

Club (clubs)

- **id** : bigint UNSIGNED, PRIMARY KEY, AUTO_INCREMENT
- **name** : varchar(191)
- **slug** : varchar(191), UNIQUE
- **created_at/updated_at** : timestamp (nullable)

Maillot (maillots)

- **id** : bigint UNSIGNED, PRIMARY KEY, AUTO_INCREMENT
- **club_id** : bigint UNSIGNED, FOREIGN KEY (clubs.id)
- **nom** : varchar(191)
- **image** : varchar(191)
- **price** : decimal(8,2) DEFAULT 0.00
- **created_at/updated_at** : timestamp (nullable)
- **Indexes** : club_id

Panier (carts)

- **id** : bigint UNSIGNED, PRIMARY KEY, AUTO_INCREMENT

- **user_id** : bigint UNSIGNED, UNIQUE, FOREIGN KEY (users.id)
- **created_at/updated_at** : timestamp (nullable)

Article du panier (cart_items)

- **id** : bigint UNSIGNED, PRIMARY KEY, AUTO_INCREMENT
- **cart_id** : bigint UNSIGNED, FOREIGN KEY (carts.id)
- **maillot_id** : bigint UNSIGNED, FOREIGN KEY (maillots.id)
- **size** : varchar(191) (nullable)
- **quantity** : int DEFAULT 1
- **numero** : varchar(191) (nullable)
- **nom** : varchar(191) (nullable)
- **created_at/updated_at** : timestamp (nullable)
- **Indexes** : cart_id, maillot_id

Sessions (sessions)

- **id** : varchar(191), PRIMARY KEY
- **user_id** : bigint UNSIGNED, FOREIGN KEY (users.id)
- **ip_address** : varchar(45) (nullable)
- **user_agent** : text (nullable)
- **payload** : longtext
- **last_activity** : int
- **Indexes** : user_id, last_activity

Relations et contraintes

Origine	Cible	Foreign Key	Cardinalité
user_addresses	users	user_id → users.id	n,1
carts	users	user_id → users.id	n,1
cart_items	carts	cart_id → carts.id	n,1
cart_items	maillots	maillot_id → maillots.id	n,1
maillots	clubs	club_id → clubs.id	n,1
sessions	users	user_id → users.id	n,1

Le MPD ci-dessus correspond à la structure physique des tables SQL : types de champs, clés primaires, étrangères et index sont explicitement listés pour une implémentation optimale en base MySQL.

Chapitre 3 — Développement de la partie front-end sécurisée

3.1. CP1 — Installer et configurer l'environnement de travail

La mise en place de l'environnement a constitué une étape essentielle pour garantir un développement fluide et conforme aux standards.

- **Installation des outils principaux :**

- Framework Laravel version 11.9 avec PHP \geq 8.2 (cf. composer.json)
- Node.js et NPM pour la partie front-end.
- Vite comme bundler moderne.
- Git pour le suivi de version et la collaboration.

- **Configuration du projet :**
 - Génération d'un projet **Laravel 11** via Composer.
 - Installation du preset **Inertia.js + React + TailwindCSS**.
 - Mise en place du fichier **vite.config.js** pour gérer le build et le hot reload.
 - Configuration des routes côté Laravel pour pointer vers les pages React via Inertia.
- **Organisation du code :**
 - Dossier resources/js/Pages pour les pages React.
 - Dossier resources/js/Components pour les composants réutilisables.
 - Dossier resources/css pour les styles centralisés.

Cet environnement garantit une intégration fluide entre Laravel et React, tout en profitant des avantages de Vite (rapidité, HMR).

Démarrage de l'application

Dans le terminal, taper npm run dev pour lancer le front.

Ouvrir une commande supplémentaire et taper php artisan serve pour lancer la back.

3.2. CP2 — Maquetter une application

La maquette a été pensée en amont afin de guider le développement. Elle s'appuie sur les **principaux parcours utilisateurs** :

- **Accueil** : présentation générale du site et accès rapide au catalogue.

- **Catalogue des maillots** : affichage en grille, filtrage par clubs.
- **Fiche produit** : visuels du maillot, description, prix, disponibilité, bouton “ajouter au panier”.
- **Panier** : liste des articles ajoutés, quantités modifiables, total calculé automatiquement.
- **Authentification** : page de connexion et d’inscription.
- **Espace utilisateur/ tableau de bord** : gestion des adresses, suivi des commandes.

Les outils de prototypage utilisés (Figma en l’occurrence) ont permis d’orienter mes **choix ergonomiques** :

- Une navigation simple (header + footer),
- Des cartes produit homogènes (visuel, nom, prix),
- Des formulaires clairs (connexion, adresse).

J’ai fait le choix de réaliser une maquette Figma qui représenterait les emplacements des différents éléments que je comptais en mettre en place lors de l’écriture du code. Les choix des couleurs ne correspondaient pas à cette étape de la conception à des choix définitifs mais avaient pour fonction d’établir un visuel de ma future structure.

Néanmoins en concevant mon site, j’ai procédé à de nombreux réajustements et surtout à de nouvelles orientations esthétiques.

De plus, ainsi que je l’ai souligné précédemment, je me suis inspiré de plusieurs sites de vente en ligne de maillots de football notamment le site footbebe.

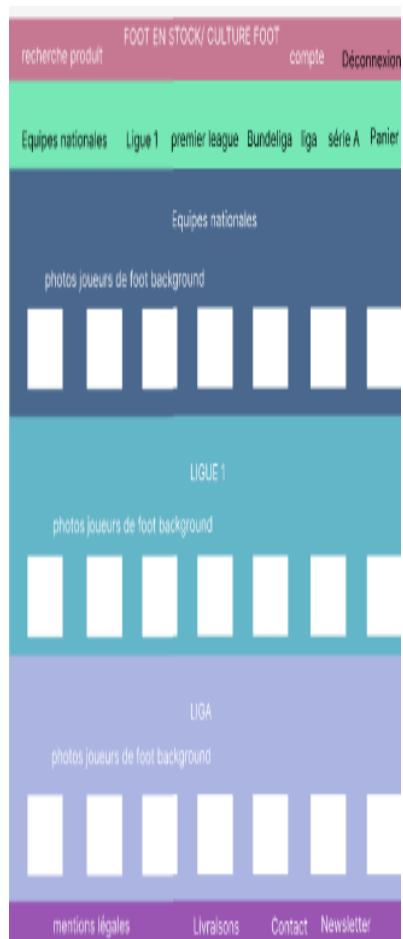
En effet leur site me paraissait plus cohérent que ceux de leurs concurrents, la navigation entre les pages répondait à une logique et se révélait fluide. L’ergonomie de leur site m’a également semblé un garant de leur sérieux, de leur professionnalisation.

L’exhaustivité de leur catalogue m’a également paru un indicateur de fiabilité du site car si les produits proposés sont nombreux, c’est que le site est capable de répondre à la demande et la prend en compte.

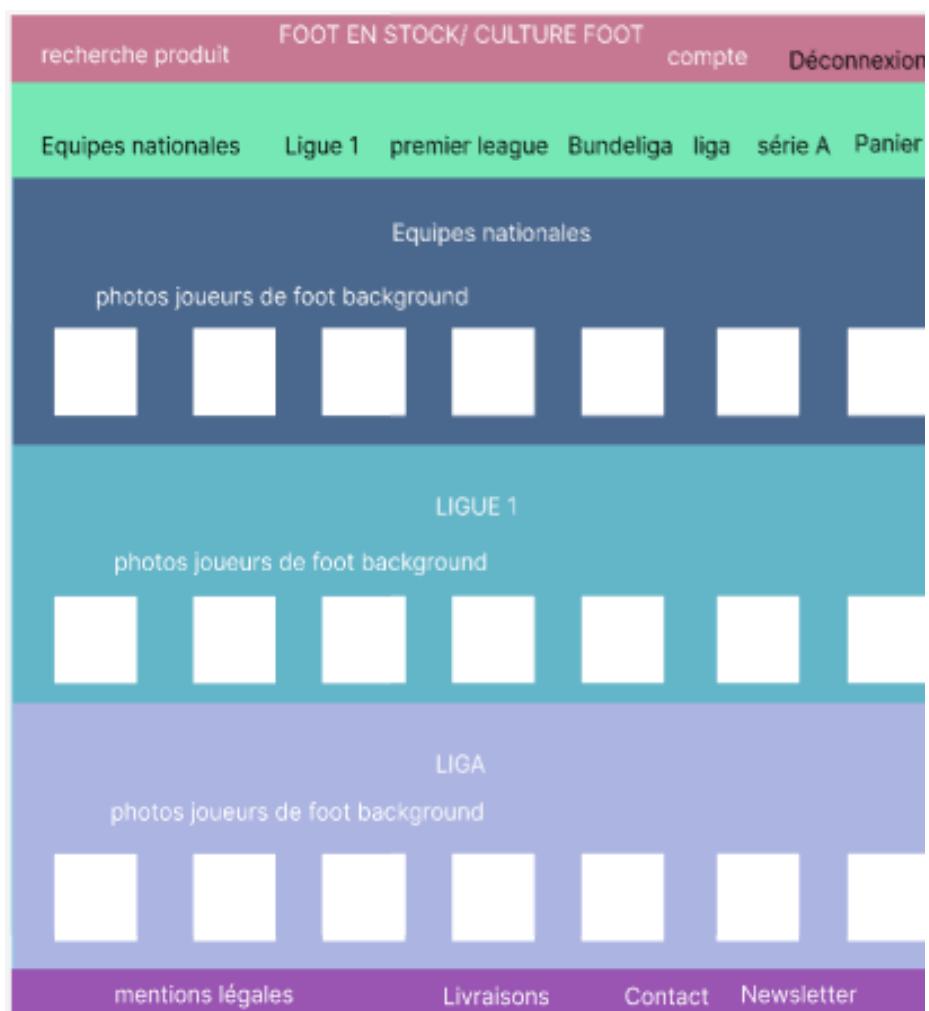
Le nom de mon site n’était pas encore déterminé au moment du maquettage. Je l’ai modifié ultérieurement pour Fou2Foot.

PAGE ACCUEIL

Version mobile



Version desktop



Connexion/ Inscription

Version mobile

Frame 27

recherche produit FOOT EN STOCK/ CULTURE FOOT compte Déconnexion

Equipes nationales Ligue 1 premier league Bundesliga liga série A Panier

S'enregistrer Connexion

email identifiant ou email

identifiant Mdp

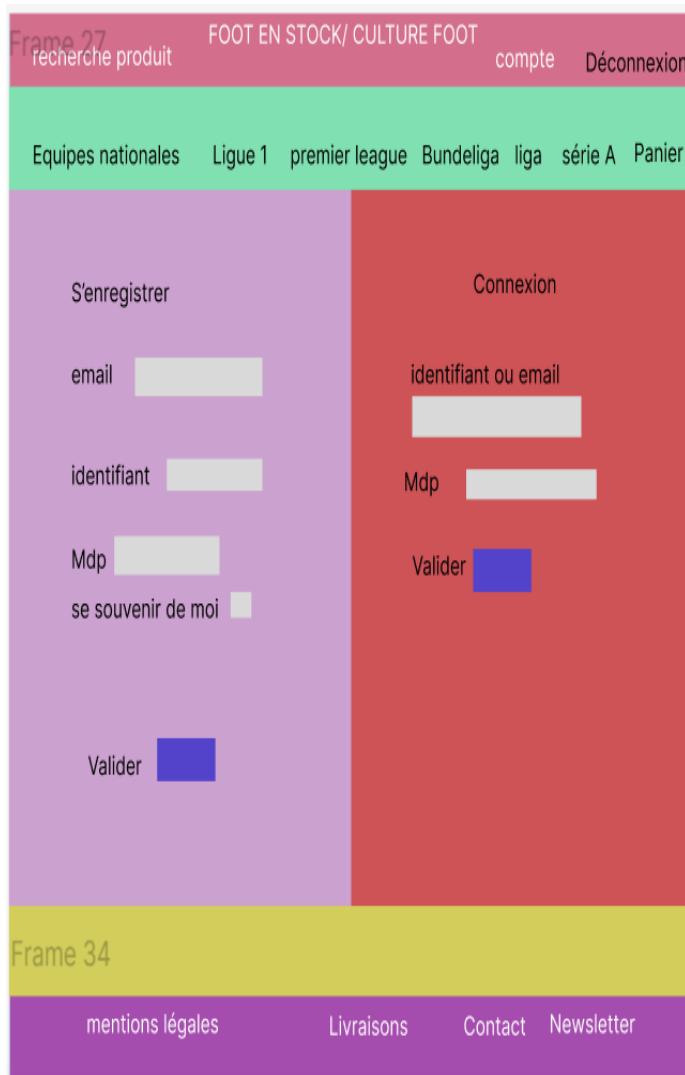
Mdp Valider

se souvenir de moi

Valider

Frame 34

mentions légales Livraisons Contact Newsletter



This image shows a mobile version of a login and registration form. The interface is divided into two main sections: a purple left section for registration and a red right section for login. Both sections feature input fields for email/identifiant and password/Mdp, with a 'Valider' button at the bottom. The purple section includes a 'se souvenir de moi' checkbox. The red section has a 'Connexion' header. At the top, there's a navigation bar with links like 'recherche produit', 'FOOT EN STOCK/ CULTURE FOOT', 'compte', and 'Déconnexion'. Below the navigation is a horizontal menu with 'Equipes nationales', 'Ligue 1', 'premier league', 'Bundesliga', 'liga', 'série A', and 'Panier'. The bottom of the screen features a yellow bar with 'Frame 34' and a purple footer bar with links for 'mentions légales', 'Livraisons', 'Contact', and 'Newsletter'.

Version desktop

The image shows a desktop view of a website with a pink header bar. The header contains the text "Frame 27", "recherche produit", "FOOT EN STOCK/ CULTURE FOOT", "compte", and "Déconnexion". Below the header is a green navigation bar with links: "Equipes nationales", "Ligue 1", "premier league", "Bundesliga", "liga", "série A", and "Panier". The main content area is divided into two sections: a purple section on the left and a red section on the right. The purple section contains the text "S'enregistrer" and fields for "email" (input type="text") and "identifiant" (input type="text"). It also has a checkbox for "se souvenir de moi" and a blue "Valider" button. The red section contains the text "Connexion" and fields for "identifiant ou email" (input type="text") and "Mdp" (input type="text"). It has a checkbox for "se souvenir de moi" and a blue "Valider" button. At the bottom of the page is a yellow bar with the text "Frame 34" and a purple footer bar with links: "mentions légales", "Livraisons", "Contact", and "Newsletter".

Frame 27

recherche produit

FOOT EN STOCK/ CULTURE FOOT

compte Déconnexion

Equipes nationales Ligue 1 premier league Bundesliga liga série A Panier

S'enregistrer

email

identifiant

Mdp

se souvenir de moi

Connexion

identifiant ou email

Mdp

Valider

Valider

Frame 34

mentions légales

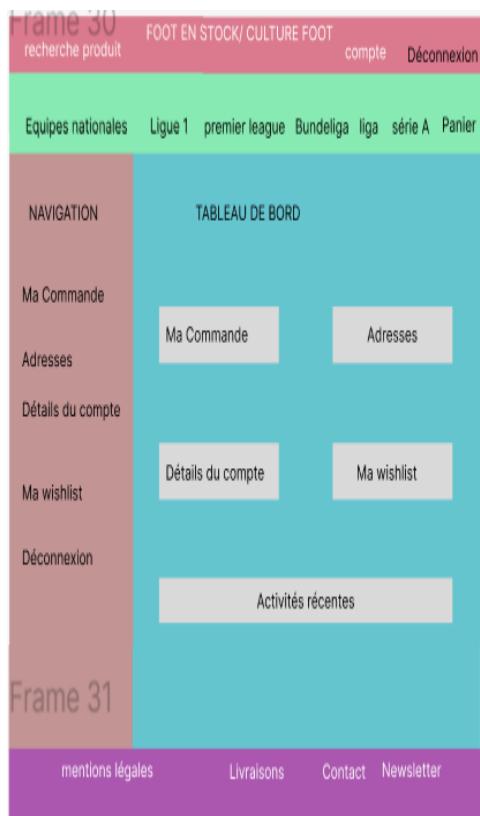
Livraisons

Contact

Newsletter

Tableau de bord

Version mobile



Version desktop

The screenshot shows a desktop application window with the following structure:

- Header:** "Frame 30" logo, "recherche produit" search bar, "FOOT EN STOCK/ CULTURE FOOT" category, "compte" account link, and "Déconnexion" log out link.
- Top Navigation Bar:** "Equipes nationales", "Ligue 1", "premier league", "Bundeliga", "liga", "série A", and "Panier".
- Left Sidebar (NAVIGATION):** "Ma Commande", "Adresses", "Détails du compte", "Ma wishlist", and "Déconnexion".
- Central Content Area (TABLEAU DE BORD):**
 - "Ma Commande" and "Adresses" buttons.
 - "Détails du compte" and "Ma wishlist" buttons.
 - A large button labeled "Activités récentes".
- Bottom Footer:** "Frame 31" watermark, "mentions légales", "Livraisons", "Contact", and "Newsletter" links.

Liste Maillots

Version desktop

The screenshot shows a desktop application interface for a football apparel store. At the top, there's a navigation bar with a logo 'Frame 25' and a search bar labeled 'recherche produit'. To the right of the search bar are links for 'FOOT EN STOCK/ CULTURE FOOT', 'compte', and 'Déconnexion'. Below the navigation bar is a secondary menu with categories: 'Equipes nationales', 'Ligue 1', 'premier league', 'Bundeliga', 'liga', 'série A', and 'Panier'. The main content area features a search input field labeled 'NOM CLUB/ EQUIPE NATIONALE'. Below this, there are three rows of three placeholder images each, representing jersey products. Underneath these images is the text 'descriptions succinctes du produit'. At the bottom of the page is a footer bar with links: 'mentions légales', 'Livraisons', 'Contact', and 'Newsletter'.

Frame 25

recherche produit

FOOT EN STOCK/ CULTURE FOOT

compte Déconnexion

Equipes nationales Ligue 1 premier league Bundeliga liga série A Panier

NOM CLUB/ EQUIPE NATIONALE

descriptions succinctes du produit

Frame 33

mentions légales Livraisons Contact Newsletter

Détail maillot

Version Desktop

The screenshot shows a desktop view of a football apparel website. At the top, there's a navigation bar with a search bar labeled "Frame 26 recherche produit", a category "FOOT EN STOCK/ CULTURE FOOT", and user account links "compte" and "Déconnexion". Below the navigation is a secondary menu with categories like "Equipes nationales", "Ligue 1", "premier league", "Bundeliga", "liga", "série A", and "Panier". The main content area features a large blue background. On the left, there's a placeholder image for a product. To the right, there are input fields for "Taille" (size) with a dropdown menu, "Personnalisation" (personalization) with a dropdown menu, "Quantité" (quantity) with a dropdown menu, and a red "Ajouter au panier" (Add to cart) button. Below these controls, there are two sections: "Description" and "Avis" (Reviews). At the bottom, there's another navigation bar with links for "Frame 32", "mentions légales" (legal mentions), "Livraisons" (Deliveries), "Contact", and "Newsletter".

Commande

J'avais initialement pensé à réaliser une page récapitulant les commandes effectuées par un utilisateur, cependant la conception de mon site s'est avérée plus chronophage que je l'avais escomptée, si bien que je n'ai produit cette page qu'en front.

Je présente néanmoins la page prévue ci-dessous et que je compte réaliser ultérieurement avec le code couleur de mon site e-commerce.

Je n'ai pas également encore réalisé la page WishList car elle ne m'a pas paru faire partie du MVP.

Commande. Version desktop

Frame 28
recherche produit

FOOT EN STOCK/ CULTURE FOOT

compte Déconnexion

Equipes nationales Ligue 1 premier league Bundesliga liga série A Panier

Détails facturations

ID

NOM Prénom

adresse [REDACTED]

email [REDACTED]

téléphone [REDACTED]

N° Carte de paiement [REDACTED]

Expiration [REDACTED]

N° CVC [REDACTED]

Détails commande

articles

total [REDACTED]

Frame 35

mentions légales

Livrailons

Contact Newsletter

3.3. CP3 — Réaliser une interface utilisateur statique et adaptable

Après la phase de maquettage, j'ai décidé de commencer par la partie front du projet suivant les recommandations de notre formateur.

Il a donc fallu transformer ces maquettes en interfaces statiques fonctionnelles, en respectant les principes d'accessibilité, de responsive design. et de cohérence graphique définis lors de la conception (notamment pour l'architecture de la page). Concernant ce dernier point, mes aspirations ont évolué. Alors qu'initialement je projetais d'utiliser des couleurs unies, j'ai finalement opté pour l'utilisation d'un dégradé rouge à bleu pour le header et un dégradé violet à bleu pour l'arrière-plan (background). Ce code couleur me paraissait plus moderne, plus frais, moins monotone qu'un fond blanc.

Cette étape correspond à la mise en œuvre concrète des maquettes, sans intégrer encore la logique dynamique ou les traitements métier.

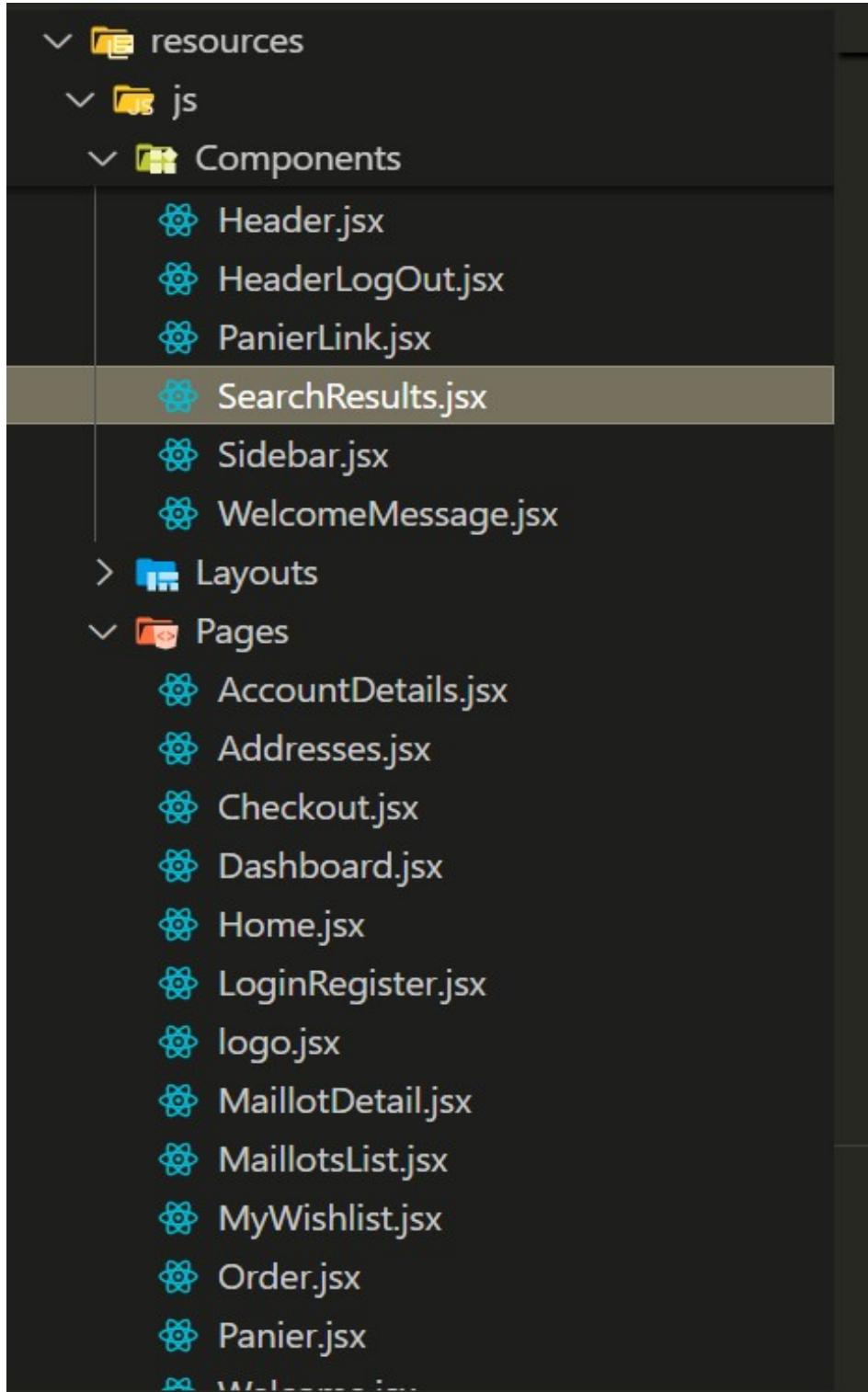
- **Pages statiques :**

- Home.jsx (page d'accueil qui correspond à la vitrine du site, accessible hors connexion).
- LoginRegister.jsx et Register.jsx. (connexion et enregistrement d'un nouvel utilisateur)
- Dashboard.jsx (Tableau de bord).
- Addresses.jsx (facturation + livraison car un utilisateur peut commander et faire livrer un maillot en cadeau à une adresse différente de la sienne).
- AccountDetails.jsx (détails du compte).
- MaillotsList.jsx (catalogue filtré) pour la visualisation du catalogue.
- MaillotDetail.jsx (fiche produit) pour l'affichage des maillots.
- Panier.jsx (produit sélectionné par l'utilisateur).

- **Composants réutilisables :**

- Navbar.jsx et Footer.jsx pour la navigation,

- WelcomeMessage.jsx afin d'afficher le nom et l'email de l'utilisateur dans le tableau de bord, Sidebar.jsx pour avoir accès au dasboard sur les pages de l'utilisateur.
- PanierLink.jsx pour le résumé du panier accessible depuis toutes les pages.
- SearchResult.jsx (pour la barre de recherche),



On peut simuler ici la navigation d'un utilisateur à partir des pages front-end

Page Accueil (ne nécessite pas de connexion car c'est la vitrine du site)

The screenshot shows a web browser window with the URL 127.0.0.1:8000 in the address bar. The page has a large header "Maillots de Football Officiels" with a red-to-blue gradient background. Below it is a sub-header "Retrouvez les tenues des plus grands clubs et des plus grandes sélections". A main text block states: "Nous nous engageons à vous fournir des maillots de football officiels de la meilleure qualité, personnalisables, à partir d'un catalogue complet recensant les meilleures ligues, avec livraison rapide. Si vous ne trouvez pas le maillot de football que vous aimez, n'hésitez pas à nous contacter. Mises à jours régulières du site avec de nouveaux maillots". Below this is a section titled "Nos Maillots Phares" featuring four thumbnail images of football jerseys: a blue and white V-neck jersey, a white and blue V-neck jersey, a grey and red jersey with "Emirates FLY BETTER", and a dark blue and red striped jersey with "Emirates FLY BETTER".

← → C ⓘ 127.0.0.1:8000 ⌂ ⌂ Scolaire ⋮

Maillots de Football Officiels

Retrouvez les tenues des plus grands clubs et des plus grandes sélections

Nous nous engageons à vous fournir des maillots de football officiels de la meilleure qualité, personnalisables, à partir d'un catalogue complet recensant les meilleures ligues, avec livraison rapide. Si vous ne trouvez pas le maillot de football que vous aimez, n'hésitez pas à nous contacter. Mises à jours régulières du site avec de nouveaux maillots

Nos Maillots Phares



Page inscription

Connexion **Inscription**

Nom d'utilisateur

Email

Mot de passe

Confirmer le mot de passe

S'inscrire

page connexion

Connexion **Inscription**

Nom d'utilisateur ou Email

Mot de passe

Se connecter

* fonctionnalité se souvenir de moi en cours

Tableau de bord

The screenshot shows the FOU2FOOT dashboard at 127.0.0.1:8000/dashboard. The top navigation bar includes the FOU2FOOT logo, a search bar, and links for Mon compte, Déconnexion, and a shopping cart with 7 items. The main content area features a welcome message "Bienvenue, mina !" with the email mina@carp.com. Below it is a section titled "Mon Tableau de Bord" with four cards: "Mes Commandes" (track purchases), "Adresse" (manage delivery address), "Détails du compte" (edit personal information), and "Ma wishlist" (view favorite items). A sidebar on the left lists navigation options: Tableau de bord, Commandes, Adresse, Détails du compte, Ma wishlist, and Se déconnecter.

Détails du compte

The screenshot shows the account details page at 127.0.0.1:8000/accountdetails. It features a sidebar with the same navigation options as the dashboard. The main content area is titled "Mon compte" and describes managing personal information and preferences. It displays "Informations personnelles" with fields for Identifiant (minna), Nom complet (MINA BELLA), Email (mina@carp.com), and Téléphone (77 33 11 88 66). A "Sécurité" section includes a "Changer le mot de passe" link. The bottom of the page shows a footer with the URL 127.0.0.1:8000/dashboard.

Adresses

The screenshot shows the FOU2FOOT website interface. The top navigation bar includes a logo, a search bar, and links for 'Mon compte', 'Déconnexion', and a shopping cart with 7 items. A sidebar on the left titled 'Navigation' lists 'Tableau de bord', 'Commandes', 'Adresse' (which is selected and highlighted in blue), 'Détails du compte', 'Ma wishlist', and 'Se déconnecter'. The main content area is titled 'Mes adresses' and contains a section for 'Adresse de facturation' (Billing address) with a 'Par défaut' button. The address listed is 'MINA BELLA' at 'rue gandolphe, 06210 Mandelieu, FR, 77 33 11 88 66'. Buttons for 'Modifier' and 'Supprimer' are also present.

Liste Maillots d'un club

The screenshot shows the FOU2FOOT website interface for the Olympique Lyonnais club. The top navigation bar is identical to the previous screenshot. The main content area is titled 'Maillots de Olympique Lyonnais' and features a message: 'Découvrez notre collection de maillots Olympique Lyonnais - 8 maillots disponibles'. Below this, four jerseys are displayed in a row: 'Domicile 2024' (white jersey with red and blue accents), 'Extérieur 2024' (black jersey with red and blue accents), 'maillot third' (red jersey with white and blue accents), and 'Maillot 75 ans' (light grey jersey with red and blue accents).

Présentation Maillot

The screenshot shows a product page for the Olympique Lyonnais 2024 home jersey. On the left, there is a large image of the white jersey with blue and red stripes on the shoulders and sleeves. The jersey features the 'Emirates FLY BETTER' logo on the front. To the right of the image, the text 'Olympique Lyonnais' and 'Domicile 2024' is displayed. Below this, the price 'Prix : 20 €', type 'Type : Maillot', and available quantity 'Quantité disponible : 1200' are shown. A dropdown menu for 'Taille' (Size) is set to 'S'. The quantity 'Quantité' is set to '1'. There are two optional checkboxes: 'Ajouter un numéro (+2 €)' and 'Ajouter un nom (+3 €)'. A bold 'Total: 20 €' is displayed, followed by a blue 'AJOUTER AU PANIER' button.

Panier de l'utilisateur

The screenshot shows a shopping cart page titled 'Mon panier'. The table lists five items with the following details:

Maillot	Taille	Quantité	Nom	Numéro	Prix maillot	Suppléments	Total ligne	Actions
Japon, Geisha	S	1			20 €	-	20.00 €	<button>Supprimer</button>
Olympique Lyonnais, OL 2025-2026 domicile	S	1	MINA P	88	20 €	5.00 €	25.00 €	<button>Supprimer</button>
Olympique Lyonnais, Maillot 75 ans	S	1	MinA	28	20 €	5.00 €	25.00 €	<button>Supprimer</button>
France, Domicile 2024	S	1	Mina Bell@	28	20 €	5.00 €	25.00 €	<button>Supprimer</button>
Olympique Lyonnais, extérieur 2025/26	S	1	MINA B	69	20 €	5.00 €	25.00 €	<button>Supprimer</button>

Outils et technologies utilisés

- **React** pour la création de composants modulaires et réutilisables.
 - **TailwindCSS** pour appliquer rapidement une mise en forme responsive et homogène. TailwindCSS permet aux classes utilitaires (flex, grid, p-4, md:w-1/2, etc.) de garantir un affichage adapté sur mobiles, tablettes et ordinateurs.
 - **Inertia.js** pour intégrer les vues React au sein du projet Laravel sans rechargement complet de la page.
-
- **Accessibilité :**
 - Utilisation de balises sémantiques (<nav>, <main>, <footer>),
 - Respect du contraste entre texte et fond,
 - Champs de formulaires correctement labellisés avec aria label.

3.4. CP4 — Développer une interface utilisateur dynamique

Après la mise en place des interfaces statiques, l'étape suivante a consisté à rendre ces pages interactives et connectées aux données réelles de l'application. L'objectif était d'offrir une expérience fluide et réactive à l'utilisateur final, en exploitant les possibilités offertes par **Inertia.js** et **React**.

Navigation dynamique

Les pages de l'application sont chargées via Inertia, permettant de conserver un comportement de type **SPA (Single Page Application)** tout en s'appuyant sur Laravel côté serveur. Par exemple, un clic sur un maillot dans MaillotsList.jsx charge instantanément la page MaillotDetail.jsx, avec les données transmises directement par le contrôleur Laravel, sans rechargement complet du navigateur.

Gestion des formulaires

Les formulaires d'authentification (connexion, inscription) et de gestion des adresses utilisateurs ont été mis en place avec une double validation : côté client (React) pour améliorer l'expérience utilisateur, et côté serveur (Laravel) pour assurer la fiabilité et la sécurité des données. Les erreurs éventuelles sont renvoyées par Inertia et affichées de manière ergonomique dans l'interface.

Interaction avec le panier

La page MaillotDetail.jsx permet l'ajout direct d'un produit au panier. Grâce à Inertia, la mise à jour est immédiate : le panier se modifie en temps réel et l'utilisateur peut voir son contenu évoluer sans interruption. Les quantités peuvent être modifiées ou les articles supprimés depuis le panier, avec un rafraîchissement automatique du total. Un composant dédié (PanierLink.jsx) affiche en permanence le nombre d'articles, offrant une vision rapide de l'état du panier sur l'ensemble des pages.

Mises à jour côté client

Les **hooks React** (useState, useEffect) ont été utilisés pour gérer l'état local des composants et synchroniser l'affichage avec les actions de l'utilisateur. Ainsi, chaque modification (par exemple, changer la taille ou le nombre d'articles) entraîne immédiatement la mise à jour du rendu visuel et des totaux affichés.

Sécurité front-end

Afin de protéger les données des utilisateurs :

- les formulaires appliquent une validation stricte côté client (format d'email, longueur du mot de passe, champs obligatoires).
- aucune donnée sensible n'est stockée dans le navigateur.
- les échanges sont sécurisés grâce à Inertia et aux protections natives de Laravel, notamment le **token CSRF**.

Résultat

Ces développements dynamiques garantissent une interface fluide, intuitive et sécurisée, tout en

respectant les bonnes pratiques d'ergonomie et de performance. L'utilisateur peut naviguer, gérer son compte et effectuer ses achats dans un environnement réactif, fidèle à l'expérience attendue d'une boutique en ligne moderne.

3.5. Accessibilité et éco-conception

Dans le cadre de ce projet, une attention particulière a été accordée aux bonnes pratiques d'**accessibilité** et d'**éco-conception**, afin de proposer une application à la fois inclusive et respectueuse de l'environnement numérique.

Accessibilité (conformité RGAA)

- **Navigation au clavier** : toutes les pages et formulaires sont utilisables sans souris, uniquement avec la touche Tab et les raccourcis clavier.
- **Labels et attributs ARIA** : ajout d'aria-labels et de rôles sémantiques sur les icônes et boutons pour assurer une lecture correcte par les technologies d'assistance (lecteurs d'écran).
- **Lisibilité et contrastes** : choix de polices lisibles, respect des seuils de contraste préconisés par le RGAA, tailles de texte adaptatives et design responsive.

Compétences mobilisées :

- Réaliser une interface utilisateur web statique et adaptable (prise en compte des normes d'accessibilité, design responsive).
- Développer une interface utilisateur web dynamique (intégration d'attributs ARIA, gestion de l'interactivité accessible).

Éco-conception

- **Optimisation des médias** : conversion des images en formats légers (WebP) et compression systématique pour limiter le poids des pages.
- **Code CSS optimisé** : grâce à **TailwindCSS**, les classes inutilisées sont supprimées lors de la phase de build (purgeCSS), ce qui réduit la taille du code distribué.

- **Chargement progressif des composants** : avec Inertia.js, seuls les composants nécessaires sont chargés (lazy loading), ce qui limite la consommation de bande passante et améliore les performances côté client.

Mon site de e-commerce profite du lazy chargement pour les images et pour les relations entre entités, ce qui améliore la rapidité et la réactivité.

Compétences mobilisées :

- Développer des composants d'accès aux données (gestion optimisée des ressources et des performances).
- Préparer et exécuter le déploiement d'une application web (intégration d'optimisations d'éco-conception avant la mise en production).

```
resources > js > Pages > ⚡ MaillotsList.jsx > ⚡ MaillotsList > ⚡ maillots.map() callback
6   export default function MaillotsList({ club, maillots }) {
34     {maillots.map((maillot, index) => (
35       <article
36         key={maillot.id}
37         className="bg-white rounded-lg shadow-md overflow-hidden hover:shadow-lg transition"
38         role="gridcell"
39         aria-rowindex={Math.floor(index / 4) + 1}
40         aria-colindex={(index % 4) + 1}
41       >
42         <Link
43           href={`/maillots/${maillot.id}`}
44           className="block group"
45           aria-label={`Voir les détails du maillot ${maillot.nom}`}
46         >
47           <div className="aspect-square overflow-hidden">
48             <img
49               src={`/ ${maillot.image}`}
50               alt={`Maillot ${maillot.nom} - ${club.name}`}
51               className="w-full h-full object-cover group-hover:scale-105 transition-transform"
52               loading="lazy"
53             />
54           </div>
55     )
56   )
57 }
```

Voir ligne 52, lazy loading

Le problème sans lazy loading

Dans une application web classique, il arrive souvent que tous les composants (boutons, formulaires, menus, pages entières...) soient chargés d'un coup dès l'ouverture du site. Les conséquences sont les suivantes :

- temps de chargement initial long.
- beaucoup de données téléchargées inutilement.
- ressources du navigateur consommées même pour des parties du site que l'utilisateur n'ouvrira peut-être jamais.

Le principe du *chargement progressif* (lazy loading)

Avec le **lazy loading**, l'application ne charge que ce qui est nécessaire au moment où cela s'avère utile :

- quand l'utilisateur consulte la page d'accueil, seuls les composants visibles (menu, produits, images utiles) sont chargés.
- si par exemple, l'utilisateur clique sur « Mon panier », le composant *Cart* est chargé à ce moment-là, et pas avant.
- si l'utilisateur n'ouvre jamais son profil , les fichiers liés à ce composant ne seront pas téléchargés.

Comment cela fonctionne avec Inertia.js

Inertia.js agit comme un « pont » entre **Laravel (back-end)** et **React/Vue (front-end)**.

- Chaque route Laravel correspond à un composant front (par exemple /panier → composant Cart.vue).
- Lorsqu'on navigue dans l'application, Inertia envoie seulement les **données nécessaires** et charge le **composant correspondant**.
- Si ce composant n'a jamais été utilisé avant, il est téléchargé à ce moment-là. C'est ça le *lazy loading*.

Les avantages

1. **Moins de bande passante consommée** : au lieu de charger 5 Mo de scripts au départ, on en charge par exemple 1 Mo, puis le reste uniquement si besoin.
2. **Site plus rapide** : la première page s'affiche plus vite.
3. **Éco-conception** : moins de données transférées = impact environnemental réduit (car moins de ressources serveurs et réseaux utilisées).
4. **Expérience utilisateur améliorée** : navigation fluide, sans rechargement complet de page.

Pour conclure le *chargement progressif des composants avec Inertia.js* signifie que l'application ne charge pas l'intégralité du code front-end au lancement, mais seulement ce dont l'utilisateur a besoin au fur et à mesure.

3.6. Bilan du front-end

La phase front-end du projet a permis de mettre en œuvre l'ensemble des compétences professionnelles CP1 à CP4 : installation et configuration de l'environnement de travail, maquettage des interfaces, réalisation statique puis dynamisation des pages utilisateur.

Ce travail s'est traduit par :

- la création d'une interface moderne et responsive, respectant les bonnes pratiques de **conception web et d'ergonomie** ;
- l'intégration de fonctionnalités dynamiques (navigation fluide via Inertia.js, gestion des formulaires, mise à jour du panier en temps réel).
- l'application des recommandations en matière d'**accessibilité (RGAA :Référentiel Général d'Amélioration de l'Accessibilité)** et d'**éco-conception** (optimisation des images, réduction du code superflu, chargement progressif des composants).
- la mise en place de mécanismes garantissant la **sécurité** côté client (validation des entrées, protection CSRF via Laravel).

En résumé, le front-end est désormais en capacité d'interagir efficacement avec la partie back-end. Il offre aux utilisateurs finaux une expérience fluide, intuitive et sécurisée, conforme aux standards attendus dans une boutique en ligne moderne.

Chapitre 4 — Développement Back-end

Après avoir conçu et dynamisé l'interface utilisateur côté front-end, l'étape suivante a consisté à mettre en place la partie back-end de l'application. Cette couche a pour rôle de gérer la logique métier, l'accès aux données et la sécurité globale du système. Elle repose sur le framework **Laravel**, qui fournit un environnement robuste pour interagir avec la base de données, traiter les requêtes des utilisateurs et garantir l'intégrité des informations.

L'objectif de ce chapitre est de présenter la mise en œuvre progressive des compétences professionnelles **CP5 à CP8**, allant de la création de la base de données relationnelle jusqu'à la documentation du déploiement de l'application web.

4.1. CP5 — Mettre en place une base de données relationnelle

Afin de permettre la gestion cohérente et sécurisée des données de l'application **Fou-2-Foot**, une base de données relationnelle a été conçue puis déployée. Cette étape visait à formaliser le modèle conceptuel, à le traduire en schéma relationnel et à l'implémenter dans le projet au moyen des outils intégrés de Laravel.

Outils et technologies utilisés

- **MySQL** comme système de gestion de base de données relationnelle, offrant robustesse et compatibilité avec Laravel.
- **Eloquent ORM** pour manipuler les données sous forme de modèles objets et faciliter les relations entre entités.

- **Migrations Laravel** pour créer, versionner et maintenir la structure de la base de données de manière automatisée.
- **Seeders** pour insérer des données de test (utilisateurs, clubs, maillots) et accélérer les phases de développement et de validation.

Conception et organisation des données

Le modèle conceptuel a été décliné en tables relationnelles dans le respect des principes de normalisation (éviter les redondances, garantir la cohérence) :

- **users** : stocke les informations de compte et d'authentification.
- **user_addresses** : gère les adresses de facturation et de livraison. J'ai jugé essentiel de permettre à un utilisateur de définir deux adresses distinctes, car ce type de produit peut être acheté pour soi ou offert en cadeau.
- **clubs** : référence les clubs de football afin d'associer chaque maillot à son équipe.
- **maillots** : contient le catalogue de produits (nom, image, prix, club associé).
- **carts** et **cart_items** : permettent la gestion du panier (articles sélectionnés, quantités, options de personnalisation).
- **orders** (*prévu en extension*) : table destinée au suivi des commandes et des paiements.

Base de données

localhost/phpmyadmin/index.php?route=/database/structure&db=maillot

phpMyAdmin

Serveur courant : MySQL

Récentes Préférées

Nouvelle base de données

fou2foot

maillot

Nouvelle table

carts

cart_items

clubs

maillots

migrations

sessions

users

user_addresses

Serveur : MySQL:3306 » Base de données : maillot

Structure SQL Rechercher Requête Exporter Importer Opérations Privilèges Plus

Filtres Contenant le mot :

Table	Action	Lignes	Type	Interclassement	Taille	Perte
carts	Parcourir Structure Rechercher Insérer Vider Supprimer	18	MyISAM	utf8mb4_unicode_ci	3,4 kio	-
cart_items	Parcourir Structure Rechercher Insérer Vider Supprimer	59	MyISAM	utf8mb4_unicode_ci	14,6 kio	560 c
clubs	Parcourir Structure Rechercher Insérer Vider Supprimer	87	MyISAM	utf8mb4_unicode_ci	14,8 kio	-
maillots	Parcourir Structure Rechercher Insérer Vider Supprimer	214	MyISAM	utf8mb4_unicode_ci	29,5 kio	-
migrations	Parcourir Structure Rechercher Insérer Vider Supprimer	12	MyISAM	utf8mb4_unicode_ci	2,7 kio	-
sessions	Parcourir Structure Rechercher Insérer Vider Supprimer	2	MyISAM	utf8mb4_unicode_ci	21,0 kio	10,0 kic
users	Parcourir Structure Rechercher Insérer Vider Supprimer	18	MyISAM	utf8mb4_unicode_ci	18,5 kio	20 c
user_addresses	Parcourir Structure Rechercher Insérer Vider Supprimer	21	MyISAM	utf8mb4_unicode_ci	7,1 kio	-
8 tables	Somme	431	MyISAM	utf8mb4_general_ci	111,5 kio	10,5 kio

Tout cocher / Vérifier les tables non optimisées Avec la sélection :

Console de requêtes SQL de données

Table users

localhost/phpmyadmin/index.php?route=/sql&db=maillot&table=users&pos=0

phpMyAdmin

Parcourir Structure SQL Rechercher Insérer Exporter Importer Privilèges Opérations Plus

Affichage des lignes 0 - 17 (total de 18, traitement en 0,0037 seconde(s).)

SELECT * FROM `users`

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

Options supplémentaires

	id	username	email	password	first_name	last_name
<input type="checkbox"/> Éditer Copier Supprimer	1	Albundy	albundy@gmail.com	\$2y\$12\$UvN5QMkelWZ9jKthn5codO.qKMFCFvNc8KwPsMjnRZ...	AL	BU
<input type="checkbox"/> Éditer Copier Supprimer	2	prasanna	prasann@yahoo.in	\$2y\$12\$O2AMY26P67ahq57agiSkXeWHCfnoEdhRbse3QsN.w...	Prasanna Lakshmi	Dai
<input type="checkbox"/> Éditer Copier Supprimer	3	Lolo	lolo@gmail.com	\$2y\$12\$BKwgXMX9dDzQOKZH1nxw.uq24jq8BfQggGMKFnivcxr...	Lolo	Rig
<input type="checkbox"/> Éditer Copier Supprimer	4	Sophie69	sophie@gmail.com	\$2y\$12\$U6nJ4RH09y.TiXoE3rdkgeD6e6.M400Mm5GDogmpkQw...	NULL	NU
<input type="checkbox"/> Éditer Copier Supprimer	5	marion	marion@hotmail.com	\$2y\$12\$des/YkbbsYL9MI/1rYB6eu97Yzd7.LdCmhX.j69F4W...	Marion	LA
<input type="checkbox"/> Éditer Copier Supprimer	6	magnum	magnum@yahoo.us	\$2y\$12\$JGfAGqsS.cugICQviXDLfOraM.waSjCidCnMP6.xsyX...	thomas	Ma
<input type="checkbox"/> Éditer Copier Supprimer	7	raïs	rais@yahoo.fr	\$2y\$12\$S5cQ9VfHc7dG.A0maBwl..WaUcnDCiou/qFTm0qcnZ5...	Raïs	Pac

Console de requêtes SQL

localhost/nhnmyadmin/index.php?route=/server/privilileges&dh=maillot&table=users&checkprivs=dh=maillot&checkprivstable=users&viewing_mode=table

Structure table users

localhost/phpmyadmin/index.php?route=/table/structure&db=maillot&table=users

Scolaire

Parcourir Structure SQL Rechercher Insérer Exporter Importer Priviléges Opérations Plus

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	id	bigint		UNSIGNED	Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
2	username	varchar(50)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
3	email	varchar(191)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
4	password	varchar(191)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
5	first_name	varchar(100)	utf8mb4_unicode_ci		Oui	NULL			Modifier Supprimer Plus
6	last_name	varchar(100)	utf8mb4_unicode_ci		Oui	NULL			Modifier Supprimer Plus
7	phone	varchar(20)	utf8mb4_unicode_ci		Oui	NULL			Modifier Supprimer Plus
8	birth_date	date			Oui	NULL			Modifier Supprimer Plus
9	gender	enum('male', 'female', 'other')	utf8mb4_unicode_ci		Oui	NULL			Modifier Supprimer Plus
10	email_verified_at	timestamp			Oui	NULL			Modifier Supprimer Plus
11	is_active	tinyint(1)			Non	1			Modifier Supprimer Plus
12	remember_token	varchar(100)	utf8mb4_unicode_ci		Oui	NULL			Modifier Supprimer Plus
13	created_at	timestamp			Oui	NULL			Modifier Supprimer Plus
14	updated_at	timestamp			Oui	NULL			Modifier Supprimer Plus

Tout cocher Avec la sélection : Parcourir Modifier Supprimer Primaire Unique Index Spatial

Texte entier

Console de requêtes SQL 3 colonnes Normaliser

Etant encore en phase évolutive, il est vraisemblable que je modifierai à terme les valeurs des types pour que celles-ci soient plus restrictives.

CartItem

localhost/phpmyadmin/index.php?route=/sql&pos=0&db=maillot&table=cart_items

Scolaire

phpMyAdmin

Parcourir Structure SQL Rechercher Insérer Exporter Importer Priviléges Opérations Plus

Affichage des lignes 0 - 24 (total de 59, traitement en 0,0012 seconde(s).)

SELECT * FROM `cart_items`

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

1 > >> Tout afficher Nombre de lignes : 25 Filtrer les lignes : Chercher dans cette table Trier par clé : Aucun(e)

Nouvelle base de données

- fou2foot
- maillot
- Nouvelle table
- carts
- cart_items
- clubs
- maillots
- migrations
- sessions
- users
- user_addresses

<input type="checkbox"/>				Éditer	Copier	Supprimer	138	1	188	XL	3 2025-07-22 11:59:49	2025-07-25 11:17:50	33	ZORRO
<input type="checkbox"/>				Éditer	Copier	Supprimer	114	2	185	S	1 2025-07-20 02:22:00	2025-09-27 00:46:56	NULL	NULL
<input type="checkbox"/>				Éditer	Copier	Supprimer	7	4	177	XL	2 2025-07-15 07:59:45	2025-07-25 09:19:18	NULL	LULOL
<input type="checkbox"/>				Éditer	Copier	Supprimer	8	4	188	XL	3 2025-07-15 08:35:53	2025-07-25 09:19:29	NULL	LUBLEU
<input type="checkbox"/>				Éditer	Copier	Supprimer	140	17	2	XL	3 2025-07-22 13:33:39	2025-07-22 13:55:37	31	ULYSSE
<input type="checkbox"/>				Éditer	Copier	Supprimer	153	3	389	XL	3 2025-07-26 00:32:44	2025-07-26 01:19:55	88	AL BUNDY
<input type="checkbox"/>				Éditer	Copier	Supprimer	154	3	1	XL	1 2025-07-26 00:42:28	2025-07-26 00:42:28	97	BUNDY FRANCE
<input type="checkbox"/>				Éditer	Copier	Supprimer	19	3	188	XL	3 2025-07-15 09:40:00	2025-07-26 01:20:06	0	AL FRANCE
<input type="checkbox"/>				Éditer	Copier	Supprimer	20	3	36	XL	1 2025-07-15 09:41:37	2025-07-26 01:20:47	99	AL OL
<input type="checkbox"/>				Éditer	Copier	Supprimer	129	7	36	S	3 2025-07-21 12:05:35	2025-07-21 13:26:54	11	Marion OL
<input type="checkbox"/>				Éditer	Copier	Supprimer	146	1	371	XL	2 2025-07-25 10:09:09	2025-07-25 10:09:09	00	Zorro le renard
Console de requêtes SQL														

Maillots

localhost/phpmyadmin/index.php?route=/sql&pos=0&db=maillot&table=maillots

phpMyAdmin

Serveur courant : MySQL

Récentes Préférées

Nouvelle base de données
fou2foot
maillot
Nouvelle table
cart
cart_items
clubs
maillots
migrations
sessions
users
user_addresses

Parcourir Structure SQL Rechercher Insérer Exporter Importer Privilèges Opérations Plus

Affichage des lignes 0 - 24 (total de 214, traitement en 0,0008 seconde(s).)

SELECT * FROM `maillots`

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

1 > >> Tout afficher Nombre de lignes : 25 Filtrer les lignes : Chercher dans cette table Trier par clé : Aucun(e)

Options supplémentaires

	id	club_id	nom	image	created_at	updated_at	price
<input type="checkbox"/> Éditer Copier Supprimer	1	1	Domicile 2024	images/maillot/images_maillot/france.jpg	2025-07-11 00:51:44	2025-07-11 00:51:44	20.00
<input type="checkbox"/> Éditer Copier Supprimer	2	1	Extérieur 2024	images/maillot/images_maillot/france_ext.jif	2025-07-11 00:51:44	2025-07-11 00:51:44	20.00
<input type="checkbox"/> Éditer Copier Supprimer	3	2	Domicile 2024	images/maillot/images_maillot/bresil.jif	2025-07-11 00:51:44	2025-07-11 00:51:44	20.00
<input type="checkbox"/> Éditer Copier Supprimer	4	2	Extérieur 2025	images/maillot/images_maillot/bresil26_exterieur.w...	2025-07-11 00:51:44	2025-07-11 00:51:44	20.00
<input type="checkbox"/> Éditer Copier Supprimer	5	3	Domicile 2024	images/maillot/images_maillot/espagne_24_domicile...	2025-07-11 00:51:44	2025-07-11 00:51:44	20.00
<input type="checkbox"/> Éditer Copier Supprimer	6	3	Extérieur 2024	images/maillot/images_maillot/espagne_24_exterieur...	2025-07-11 00:51:44	2025-07-11 00:51:44	20.00
<input type="checkbox"/> Éditer Copier Supprimer	7	4	Domicile 2024	images/maillot/images_maillot/pays_bas_24_domicile...	2025-07-11 00:51:44	2025-07-11 00:51:44	20.00
<input type="checkbox"/> Console de requêtes SQL			Extérieur		2025-07-11	2025-07-11	

UserAddress

Mise en œuvre technique

- Les **migrations** ont permis de générer automatiquement les tables et leurs contraintes (clés primaires (PK), clés étrangères(FK), index).
- Les relations entre modèles ont été définies via Eloquent, par exemple :
 - User → UserAddress (relation *one-to-many*),
 - Club → Maillot (relation *one-to-many*),
 - Cart → CartItem (relation *one-to-many*).
- Les **seeders** ont permis d'initialiser des données de démonstration, facilitant les tests fonctionnels des pages de liste (MaillotsList.jsx) et de détail (MaillotDetail.jsx).

Résultat

La mise en place de la base de données relationnelle offre plusieurs avantages concrets :

- une gestion structurée et centralisée des informations,
- l'assurance de l'intégrité référentielle (par exemple, un maillot appartient toujours à un club existant),
- la traçabilité et la cohérence des données utilisateurs (par exemple, chaque panier est lié à un compte),
- un socle fiable et évolutif pour les étapes suivantes : développement des composants d'accès aux données (CP6) et mise en œuvre de la logique métier côté serveur (CP7).

Migrations

localhost/phpmyadmin/index.php?route=/sql&pos=0&db=maillot&table=migrations

phpMyAdmin

Serveur courant : MySQL

Récentes Préférées

Nouvelle base de données
fou2foot
maillot
Nouvelle table
carts
cart_items
clubs
maillots
migrations
sessions
users
user_addresses

Parcourir Structure SQL Rechercher Insérer Exporter Importer Privilèges Opérations Plus

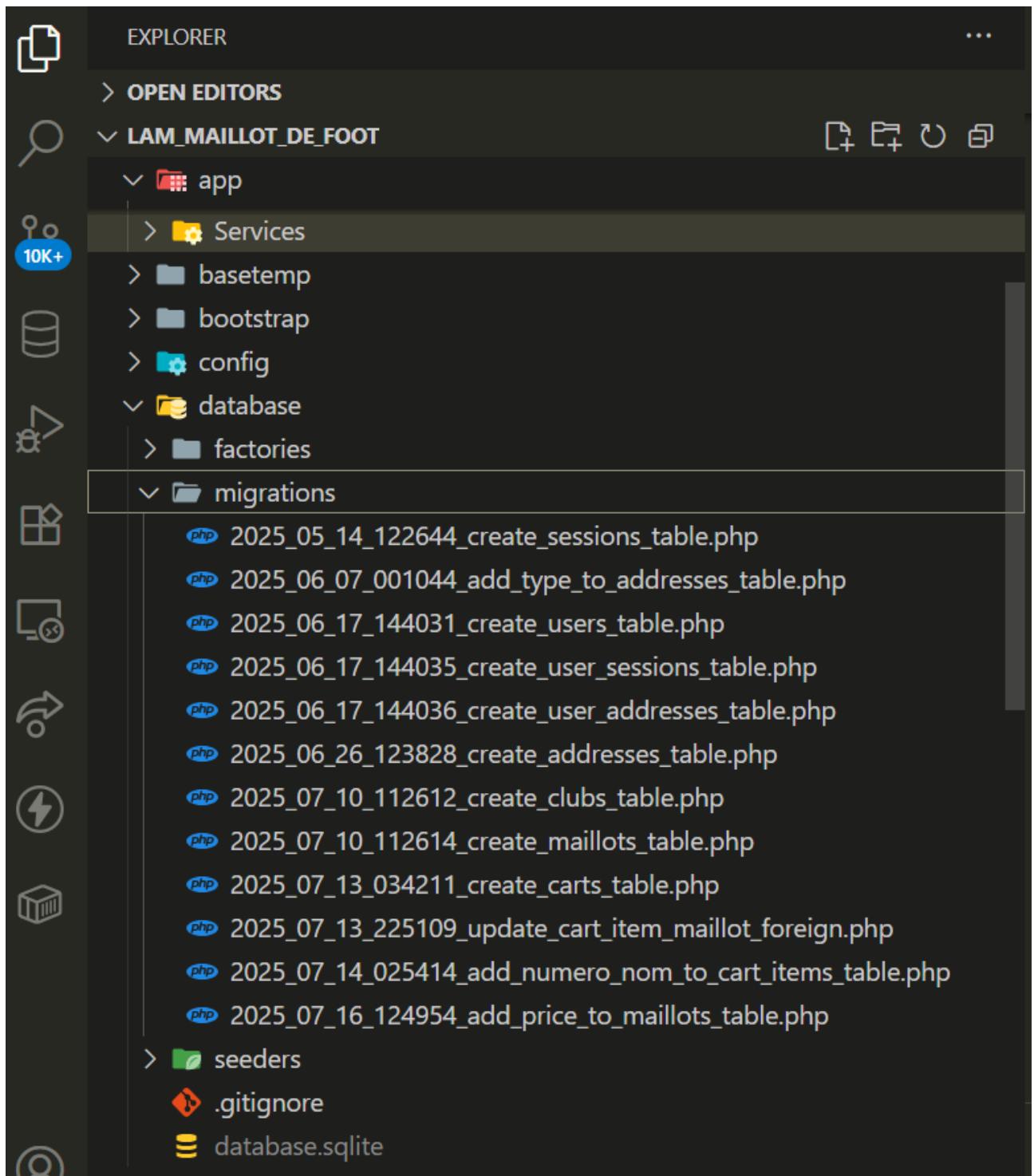
	id	migration	batch
<input type="checkbox"/>	1	2025_05_14_122644_create_sessions_table	1
<input type="checkbox"/>	2	2025_06_07_001044_add_type_to_addresses_table	1
<input type="checkbox"/>	3	2025_06_17_144031_create_users_table	1
<input type="checkbox"/>	4	2025_06_17_144035_create_user_sessions_table	1
<input type="checkbox"/>	5	2025_06_17_144036_create_user_addresses_table	1
<input type="checkbox"/>	6	2025_06_26_123828_create_addresses_table	1
<input type="checkbox"/>	7	2025_07_10_112612_create_clubs_table	1
<input type="checkbox"/>	8	2025_07_10_112614_create_maillots_table	1
<input type="checkbox"/>	9	2025_07_13_034211_create_carts_table	2
<input type="checkbox"/>	10	2025_07_13_225109_update_cart_item_maillot_foreign	3
<input type="checkbox"/>	11	2025_07_14_025414_add_numero_nom_to_cart_items_table	4
<input type="checkbox"/>	12	2025_07_16_124954_add_price_to_maillots_table	5

Tout cocher Avec la sélection : Éditer Copier Supprimer Exporter

Tout afficher Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

Opérations sur les résultats de la requête

Imprimer Copier dans le presse-papiers Exporter Afficher le graphique Crée une vue
Console de requêtes SQL



migrations

4.2. CP6 — Développer des composants d'accès aux données SQL et NoSQL

Alors que le CP5 a consisté à **concevoir et mettre en place la base de données relationnelle** (création des tables, relations et contraintes), le CP6 s'attache à **exploiter cette base au sein de l'application** grâce aux modèles Eloquent et aux requêtes SQL. Autrement dit, le CP5 correspond à la **construction de la structure**, tandis que le CP6 concerne **l'accès et la manipulation des données** pour alimenter les fonctionnalités du projet.

La mise en place des composants d'accès aux données a constitué une étape clé du développement back-end. L'objectif était de permettre à l'application de communiquer de manière fiable et sécurisée avec la base de données relationnelle, afin de fournir au front-end les informations nécessaires (maillots, clubs, utilisateurs, paniers).

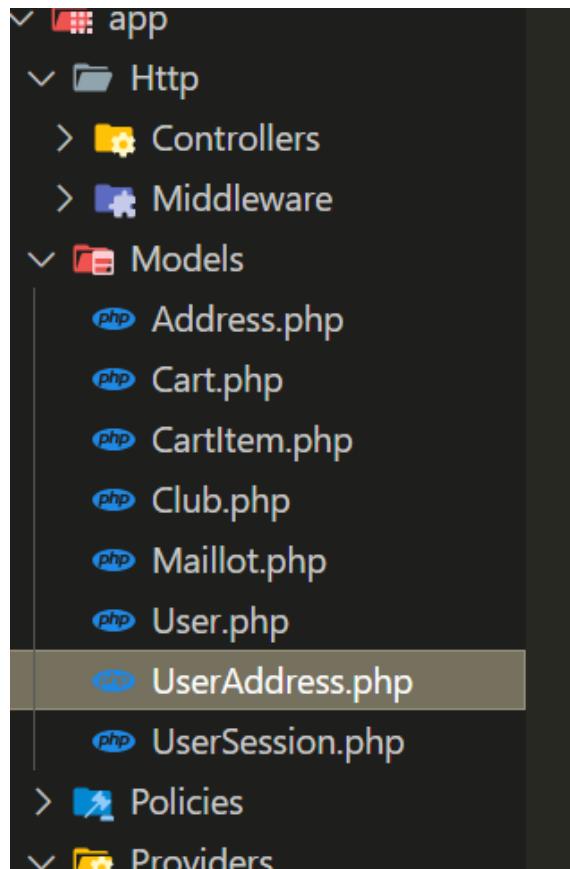
Outils et technologies utilisés

- **Eloquent ORM (Laravel)** pour représenter chaque table de la base sous forme de modèle objet et simplifier les requêtes.
- **Migrations et seeders** pour assurer la cohérence et la reproductibilité des données entre environnements.
- Optionnel : ouverture possible à des solutions NoSQL (par exemple MongoDB) pour des cas spécifiques de stockage non structuré, même si ce projet n'en nécessitait pas.

Mise en œuvre technique

- Création des modèles principaux :
 - User et UserAddress pour la gestion des comptes et des adresses,
 - Club et Maillot pour le catalogue de produits,
 - Cart et CartItem pour la gestion des paniers,

- UserSession : permet d'assurer la gestion et la traçabilité des connexions utilisateurs, renforçant ainsi la sécurité et le suivi des activités. UserSession.php est un **modèle Eloquent** côté back-end, qui s'appuie sur une table (user_session). Dans le chapitre précédent (accès aux données) on valorise les modèles, car ils sont la traduction du schéma SQL en objets manipulables dans Laravel.
- Order pour le suivi des commandes (pas encore conçu mais prévu en extension).



- Définition des relations entre modèles avec Eloquent :
 - User → UserAddress : un utilisateur peut avoir plusieurs adresses,
 - Club → Maillot : un club peut proposer plusieurs maillots,
 - Cart → CartItem : un panier contient plusieurs articles,
 - User → Cart : un utilisateur possède un ou plusieurs paniers.
- Mise en place de requêtes typiques :
 - Récupération des maillots d'un club (dans app/Http/Controllers/ClubController.php, méthode maillots(\$slug)), **ligne 12** :

```
$club = Club::where('slug', $slug)→firstOrFail();
$maillots = $club->maillots()->get();
```

```
app > Http > Controllers > ClubController.php > ClubController > maillots
  3
  4  use App\Models\Club;
  5  use Illuminate\Http\Request;
  6  use Inertia\Inertia;
  7
  8 class ClubController extends Controller
  9 {
10     public function maillots($slug)
11     {
12         $club = Club::where('slug', $slug)->firstOrFail();
13         $maillots = $club->maillots()->get();
14
15         return Inertia::render('MaillotsList', [
16             'club' => $club,
17             'maillots' => $maillots,
18         ]);
19     }
20
21     public function findSlugByName(Request $request)
22     {
23         $term = $request->query('name');
24         $club = \App\Models\Club::whereRaw('LOWER(name) = ?', [strtolower($term)])->first();
25
26         if(!$club) {
27             return response()->json(['error' => 'Club non trouvé'], 404);
28         }
29     }
30 }
```

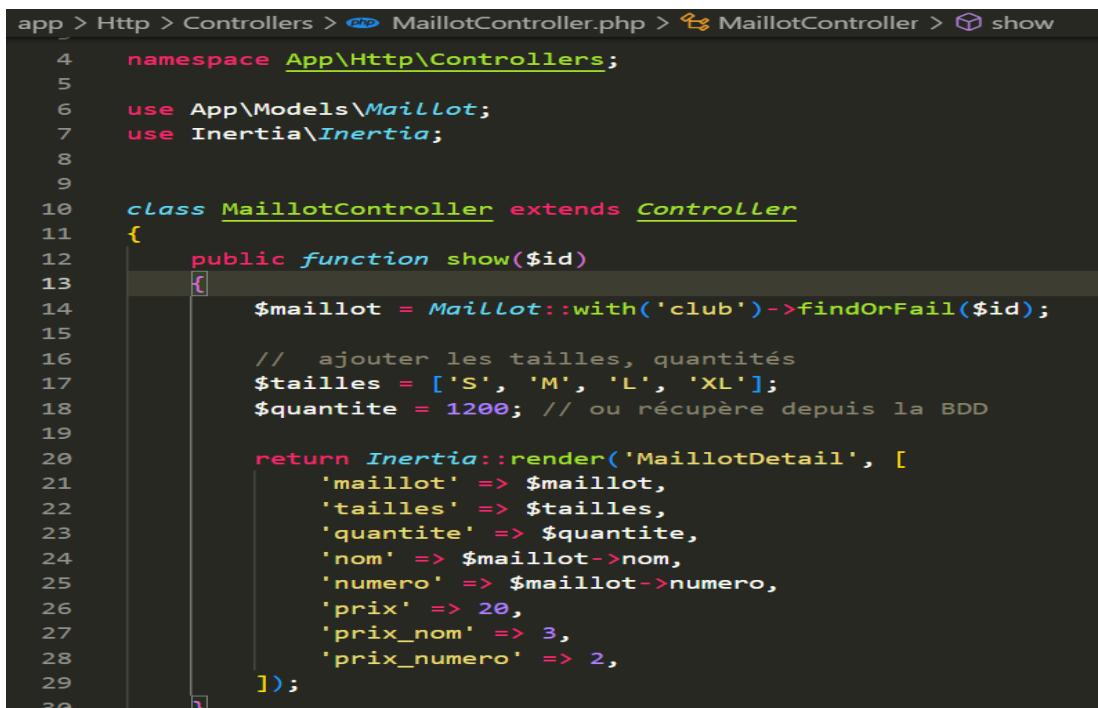
Pour afficher la liste des maillots associés à un club donné, le contrôleur interroge la relation définie dans le modèle Club.

Ici, firstOrFail() garantit que le club existe bien en base de données, et renvoie une erreur 404 dans le cas contraire. La méthode maillots() exploite la relation *one-to-many* entre Club et Maillot afin de récupérer directement tous les maillots associés.

Ce mécanisme illustre l'un des avantages d'Eloquent : transformer une relation SQL en une méthode simple et lisible dans le code.

- Affichage du détail d'un maillot (Dans app/Http/Controllers/MaillotController.php, méthode show(\$id)), **ligne 14** :

```
$maillot = Maillot::with('club')->findOrFail($id);
```



```
app > Http > Controllers > MaillotController.php > MaillotController > show
 4  namespace App\Http\Controllers;
 5
 6  use App\Models\Maillot;
 7  use Inertia\Inertia;
 8
 9
10 class MaillotController extends Controller
11 {
12     public function show($id)
13     {
14         $maillot = Maillot::with('club')->findOrFail($id);
15
16         // ajouter les tailles, quantités
17         $tailles = ['S', 'M', 'L', 'XL'];
18         $quantite = 1200; // ou récupère depuis la BDD
19
20         return Inertia::render('MaillotDetail', [
21             'maillot' => $maillot,
22             'tailles' => $tailles,
23             'quantite' => $quantite,
24             'nom' => $maillot->nom,
25             'numero' => $maillot->numero,
26             'prix' => 20,
27             'prix_nom' => 3,
28             'prix_numero' => 2,
29         ]);
30     }
31 }
```

Lorsqu'un utilisateur consulte la fiche d'un maillot, le contrôleur utilise findOrFail() pour en récupérer les informations.

Dans cet exemple, findOrFail(\$id) permet de charger un maillot précis à partir de son identifiant,

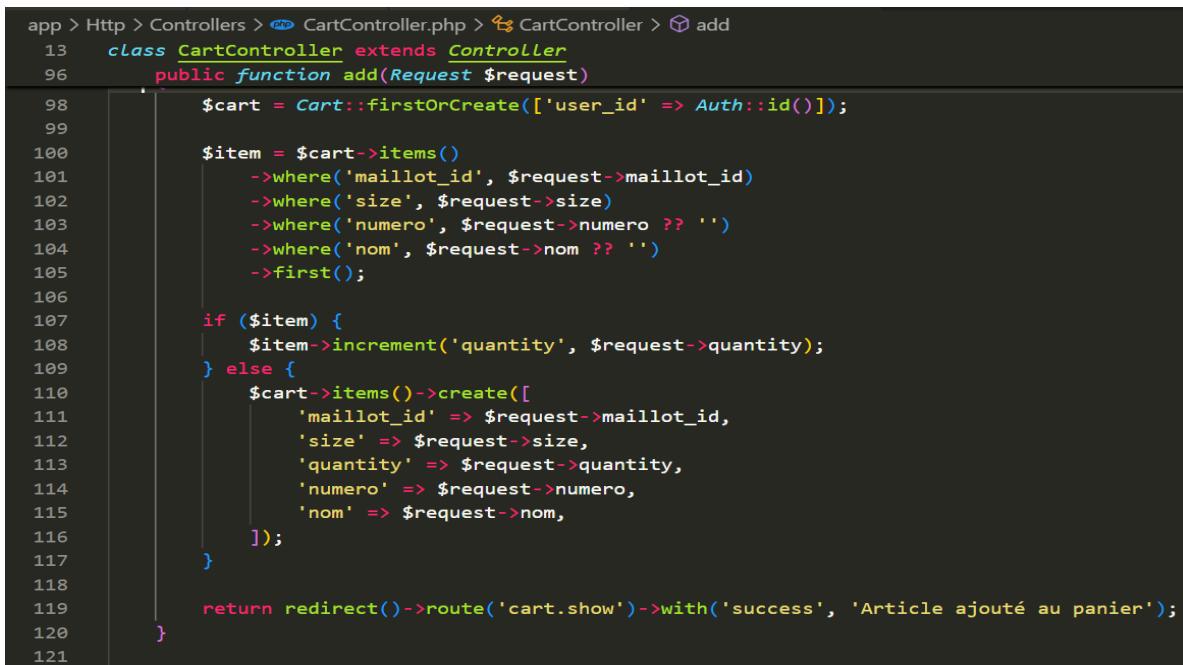
tout en gérant l'éventualité où il n'existe pas (erreur 404). La méthode with('club') charge en même temps les informations du club associé, évitant une requête supplémentaire.

Cette approche assure à la fois **efficacité** (requête optimisée) et **sécurité** (gestion des erreurs), tout en rendant le code plus expressif et compréhensible.

- ajout d'un article au panier (dans app/Http/Controllers/CartController.php, méthode add(Request \$request), :

```
$cart = Cart::firstOrCreate(['user_id' => Auth::id()]);
```

```
$item = $cart->items()  
    ->where('maillot_id', $request->maillot_id)  
    ->where('size', $request->size)  
    ->where('numero', $request->numero ?? '')  
    ->where('nom', $request->nom ?? '')  
    ->first();  
  
if ($item) {  
    $item->increment('quantity', $request->quantity);
```



```
app > Http > Controllers > CartController.php > CartController > add  
13  class CartController extends Controller  
96  public function add(Request $request)  
97      $cart = Cart::firstOrCreate(['user_id' => Auth::id()]);  
98  
99      $item = $cart->items()  
100         ->where('maillot_id', $request->maillot_id)  
101         ->where('size', $request->size)  
102         ->where('numero', $request->numero ?? '')  
103         ->where('nom', $request->nom ?? '')  
104         ->first();  
105  
106     if ($item) {  
107         $item->increment('quantity', $request->quantity);  
108     } else {  
109         $cart->items()->create([  
110             'maillot_id' => $request->maillot_id,  
111             'size' => $request->size,  
112             'quantity' => $request->quantity,  
113             'numero' => $request->numero,  
114             'nom' => $request->nom,  
115         ]);  
116     }  
117  
118     return redirect()->route('cart.show')->with('success', 'Article ajouté au panier');  
119  
120 }
```

Ce code indique deux points importants :

- Création ou récupération du panier : la méthode firstOrCreate() vérifie si l'utilisateur connecté dispose déjà d'un panier. Si ce n'est pas le cas, un nouvel enregistrement est automatiquement créé en base.
- Recherche d'un article existant : grâce à la relation items(), Eloquent interroge la table cart_items pour vérifier si le panier contient déjà un produit avec les mêmes caractéristiques (maillot, taille, numéro, nom).

Selon le résultat, l'application choisit soit d'**incrémenter la quantité** de l'article trouvé, soit de **créer une nouvelle ligne** avec \$cart->items()->create([...]).

Cet exemple illustre bien l'apport du **CP6** : un accès aux données simplifié, sécurisé et expressif, qui masque la complexité des requêtes SQL derrière des méthodes orientées objet.

Afin de rendre le code plus lisible et de simplifier la manipulation des données, les relations entre les différentes entités de la base ont été définies directement dans les modèles Eloquent. Le tableau ci-dessous récapitule l'ensemble de ces relations, leur type et leur utilisation au sein du projet.

Tableau — Relations Eloquent du projet Fou2Foot

Modèle	Méthode relation	Type de relation	Cible	Exemple d'utilisation
User	addresses()	hasMany	UserAddress	\$user->addresses
	carts()	hasMany	Cart	\$user->carts()->latest()->first()
	sessions()	hasMany	UserSession	\$user->sessions
	orders()	hasMany	Order	\$user->orders
UserAddress	user()	belongsTo	User	\$address->user
Club	maillots()	hasMany	Maillot	\$club->maillots
Maillot	club()	belongsTo	Club	\$maillot->club
	cartItems()	hasMany	CartItem	\$maillot->cartItems
Cart	user()	belongsTo	User	\$cart->user
	items()	hasMany	CartItem	\$cart->items
CartItem	cart()	belongsTo	Cart	\$item->cart
	maillot()	belongsTo	Maillot	\$item->maillot
Order (prévu)	user()	belongsTo	User	\$order->user
	items()	hasMany	OrderItem	\$order->items
UserSession	user()	belongsTo	User	\$session->user

Sécurité et performance

- Validation des données saisies grâce aux **Form Requests** de Laravel.
- Protection native contre les injections SQL via les méthodes préparées d'Eloquent.
- Gestion des clés étrangères et des contraintes d'intégrité pour garantir la cohérence des enregistrements.

Résultat

L'implémentation des composants d'accès aux données assure :

- une interaction fiable entre le back-end et la base de données.
- un code plus lisible et maintenable grâce à l'ORM (Object-Relational Mapping).
- une intégration fluide avec le front-end via Inertia.js, permettant de charger et manipuler les données (maillots, panier, adresses) en temps réel,
- une base technique solide pour la mise en œuvre de la logique métier côté serveur (**CP7**).

Conclusion

La mise en œuvre des composants d'accès aux données a permis de relier efficacement l'application à la base MySQL, tout en tirant parti des fonctionnalités offertes par Eloquent ORM. Grâce aux relations entre modèles (Club → Maillot, Cart → CartItem, etc.), il est possible d'écrire un code expressif, lisible et sécurisé pour gérer l'ensemble des opérations : récupération de listes, affichage de détails ou ajout d'articles au panier.

Ces exemples démontrent la capacité de l'application à manipuler les données de manière fiable, en respectant les contraintes d'intégrité et en simplifiant la logique d'accès. Ce socle technique constitue une étape essentielle avant la mise en place de la logique métier propre à l'application, qui sera abordée dans le **CP7**.

4.3. CP7 — Développer des composants métier côté serveur

Une fois les modèles et l'accès aux données mis en place (CP6), l'étape suivante a consisté à intégrer la **logique métier** de l'application. Cette couche traduit les règles fonctionnelles du projet en traitements concrets exécutés côté serveur, garantissant à la fois cohérence, fiabilité et sécurité.

Objectifs

- Implémenter les règles e-commerce spécifiques à l'application : gestion des paniers, personnalisation des maillots, validation des adresses, préparation des commandes.
- Centraliser cette logique dans les modèles et contrôleurs Laravel afin de séparer clairement la partie métier de la simple manipulation de données.

Principales règles métier implémentées

1. Gestion du panier

- Un utilisateur authentifié possède au plus **un panier actif**. Celui-ci est créé automatiquement au premier ajout d'article grâce à :

```
$cart = Cart::firstOrCreate(['user_id' => Auth::id()]);
```

```
app > Http > Controllers > CartController.php > CartController > add
13   class CartController extends Controller
16     public function getCount()
37
38       return response()->json(['count' => 0], 500);
39     }
40   }
41
42   public function show()
43   {
44     $cart = Cart::firstOrCreate(['user_id' => Auth::id()]);
45     $cart->load('items.maillot.club');
46 }
```

- Lors d'un ajout, si le panier contient déjà le même maillot avec la même taille et la même personnalisation (nom et numéro), la ligne est **fusionnée** en incrémentant la quantité. Sinon, un nouvel enregistrement est créé.
- Le calcul du prix tient compte des **suppléments de personnalisation** : +3 € pour un nom et +2 € pour un numéro.

```
app > Http > Controllers > CartController.php > CartController > show > Closure >
13  class CartController extends Controller
14      public function show()
15
16          return inertia('Panier', [
17              'cartItems' => $cart->items->map(function($item) {
18                  $maillot = $item->maillot;
19                  $price = $maillot ? $maillot->price : 0;
20                  $image = $maillot ? $maillot->image : null;
21                  $name = $maillot ? $maillot->name : '????';
22
23                  // Suppléments personnalisation
24                  $suppNom = $item->nom ? 3 : 0;
25                  $suppNumero = $item->numero ? 2 : 0;
26                  $supplement = $suppNom + $suppNumero;
27
28                  // Total ligne
29                  $total = ($price + $supplement) * $item->quantity;
30
31              })
32          ]);
33
34      }
35
36  
```

2. Catalogue (Club/Maillot)

- Chaque maillot est rattaché à un club de football. La relation Club → Maillots garantit qu'un produit ne peut exister qu'au sein d'un catalogue de club valide.
- L'affichage d'un maillot charge également les données du club, afin d'offrir à l'utilisateur un contexte complet.

3. Commandes (prévu en extension)

- Le passage en caisse (checkout) est déjà prévu : transformer le contenu du panier en **commande (Order)** avec ses lignes, puis vider le panier.
- Cette étape devra à terme inclure une **transaction** pour garantir l'intégrité des données (création de l'ordre, enregistrement des articles, calcul des totaux).

4. Adresses et livraison

- Chaque utilisateur peut définir une adresse de facturation et de livraison.
- Une adresse par défaut est associée au panier afin de fluidifier le processus de commande.

5. Sécurité et autorisations

- Des **policies** contrôlent que seul le propriétaire du panier peut modifier ou supprimer ses articles.
- Les sessions utilisateurs (UserSession) sont enregistrées afin de tracer les connexions et renforcer la sécurité.

Exemple : logique métier du panier

```
app > Http > Controllers > CartController.php > CartController > add
13  class CartController extends Controller
42  public function show()
95  }
96  public function add(Request $request)
97  {
98      $cart = Cart::firstOrCreate(['user_id' => Auth::id()]);
99
100     $item = $cart->items()
101     ->where('maillot_id', $request->maillot_id)
102     ->where('size', $request->size)
103     ->where('numero', $request->numero ?? '')
104     ->where('nom', $request->nom ?? '')
105     ->first();
106
107     if ($item) {
108         $item->increment('quantity', $request->quantity);
109     } else {
110         $cart->items()->create([
111             'maillot_id' => $request->maillot_id,
112             'size' => $request->size,
113             'quantity' => $request->quantity,
114             'numero' => $request->numero,
115             'nom' => $request->nom,
116         ]);
117     }
118
119     return redirect()->route('cart.show')->with('success', 'Article ajouté au panier');
120 }
```

Cet extrait illustre la logique métier : fusionner les lignes similaires au lieu de créer des doublons, afin d'assurer la cohérence du panier.

Résultat

L'intégration de ces règles métier a permis :

- de rendre l'application cohérente avec les attentes fonctionnelles d'une boutique en ligne,
- d'automatiser les traitements répétitifs (fusion des articles, calcul des prix),
- de renforcer la sécurité grâce aux policies et aux sessions,
- de préparer l'évolution vers une gestion complète des commandes et paiements (via le modèle Order).

Ainsi, le CP7 démontre la capacité du projet à dépasser la simple gestion de données pour appliquer de vraies **règles de gestion métier**, garantissant une expérience utilisateur fiable et conforme aux objectifs fonctionnels.

4.4. CP8 — Documenter le déploiement d'une application web ou web mobile

Afin de rendre le projet exploitable en conditions réelles, un processus de déploiement a été défini et documenté. L'objectif était de pouvoir installer l'application sur un serveur distant de manière reproductible et sécurisée, en garantissant la disponibilité du front-end et du back-end.

Environnement cible

- **Serveur Linux (Ubuntu 22.04 LTS)** hébergé sur un VPS.
- **Stack LEMP** (Nginx, MySQL, PHP 8.2) adaptée à Laravel.
- **Node.js** pour compiler les assets front-end via Vite.
- **Composer** pour gérer les dépendances back-end.

Procédure de déploiement

1. Préparation du serveur :

- Installation des dépendances système (php-cli, php-mysql, mysql-server, nginx, nodejs, npm, composer).
- Création d'un utilisateur dédié non-root pour sécuriser les opérations.

2. Récupération du projet :

- Clonage du dépôt Git (git clone ...).
- Configuration des droits d'accès sur storage/ et bootstrap/cache/.

3. Configuration de l'application :

- Copie et adaptation du fichier .env pour définir les paramètres (connexion MySQL, clés API, variables d'environnement).
- Génération de la clé applicative Laravel :

```
php artisan key:generate
```

4. Base de données :

- Création de la base MySQL et d'un utilisateur dédié.
- Exécution des migrations et des seeders pour créer les tables et insérer des données initiales :

```
php artisan migrate --seed
```

5. Compilation front-end :

- Installation des dépendances front-end :

```
npm install
```

- Build de production avec Vite :

```
npm run build
```

6. Configuration du serveur web (Nginx) :

- Création d'un virtual host pointant sur le dossier public/ de Laravel.
- Activation de HTTPS avec **Certbot** (Let's Encrypt).

7. Mise en production :

- Lancement de Laravel via **PHP-FPM**.
- Mise en place de **supervision** (fail2ban, UFW firewall) pour renforcer la sécurité.

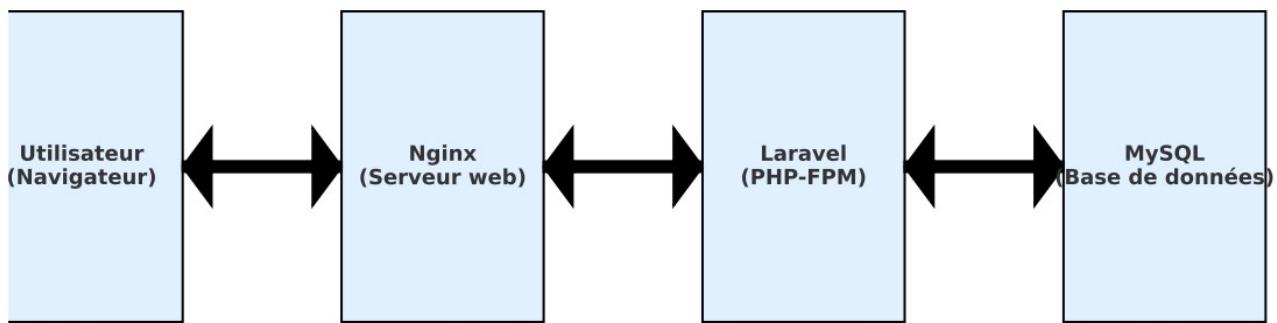
Résultat

Grâce à cette procédure, l'application peut être déployée sur un serveur distant en suivant des étapes claires et reproductibles. La documentation garantit que tout développeur ou administrateur système peut installer et configurer le projet de manière identique, assurant ainsi la portabilité et la maintenabilité du système.

Tableau — Checklist de déploiement

Étape	Commande / Action	Résultat attendu
1. Préparation du serveur	<code>sudo apt update && sudo apt install php-cli php-mysql mysql-server nginx nodejs npm composer</code>	Serveur prêt avec PHP, MySQL, Nginx, Node.js, Composer
2. Récupération du projet	<code>git clone <repo> && cd projet</code>	Code source disponible sur le serveur
3. Permissions	<code>chmod -R 775 storage bootstrap/cache</code>	Laravel peut écrire dans ses répertoires
4. Configuration	<code>cp .env.example .env puis adaptation (DB, APP_KEY, etc.)</code>	Variables d'environnement définies
5. Génération de la clé	<code>php artisan key:generate</code>	Clé d'application générée
6. Base de données	<code>php artisan migrate --seed</code>	Tables et données initiales créées
7. Dépendances front-end	<code>npm install && npm run build</code>	Assets front compilés en mode production
8. Dépendances back-end	<code>composer install --optimize-autoloader --no-dev</code>	Packages PHP installés et optimisés
9. Configuration Nginx	Virtual host pointant vers public/	Site accessible via HTTP
10. Sécurisation HTTPS	<code>sudo certbot --nginx -d domaine.com</code>	Certificat SSL actif
11. Mise en production	Lancer PHP-FPM, configurer UFW et fail2ban	Application sécurisée et disponible en ligne

schéma d'architecture simplifié du déploiement :



Ce schéma illustre le flux de communication lors de l'exécution de l'application : l'utilisateur interagit via son navigateur, les requêtes sont d'abord traitées par le serveur web **Nginx**, qui redirige vers **Laravel (PHP-FPM)** pour la logique applicative. Laravel interroge ensuite **MySQL** pour accéder aux données, et les réponses suivent le chemin inverse afin d'être affichées à l'utilisateur.

4.5. Bilan du back-end

Le développement back-end a permis de doter l'application d'un socle robuste, assurant la cohérence des données et l'application des règles métier. Les différentes étapes ont couvert les compétences professionnelles attendues :

- **CP5** : conception et mise en place d'une base de données relationnelle normalisée, garantissant l'intégrité des informations.
- **CP6** : implémentation de composants d'accès aux données via Eloquent ORM, facilitant la manipulation sécurisée et lisible des tables.

- **CP7** : intégration de la logique métier, avec gestion complète du panier, règles de tarification (suppléments pour la personnalisation), autorisations et préparation de l'extension vers la gestion des commandes.
- **CP8** : documentation du processus de déploiement, permettant la reproductibilité et l'industrialisation du projet dans un environnement serveur sécurisé.

Résultat : le back-end fournit désormais un environnement fiable et évolutif, garantissant la cohérence des interactions entre utilisateurs, interface et base de données. Couplé au front-end, il assure une expérience fluide, sécurisée et conforme aux objectifs initiaux du projet.

Chapitre 5 — Conclusion et perspectives

5.1. Bilan global

Le projet Fou2Foot a permis de mettre en pratique l'ensemble des compétences visées par le **Titre Professionnel Développeur Web et Web Mobile (DWWM)**.

Sur le plan **technique**, le développement a couvert :

- la création d'une interface utilisateur moderne et responsive avec React, TailwindCSS et Inertia.js ;
- la mise en place d'une base de données relationnelle complète et normalisée (MySQL) ;
- l'implémentation de la logique métier côté serveur via Laravel et Eloquent ;
- la préparation d'un processus de déploiement reproductible et sécurisé.

Sur le plan **méthodologique**, ce projet a renforcé des compétences clés : analyse d'un besoin, structuration d'un projet, gestion des versions avec Git, documentation technique et présentation professionnelle.

L'application constitue ainsi une **preuve de concept fonctionnelle** : un site e-commerce spécialisé dans la vente de maillots personnalisés, répondant aux attentes des utilisateurs cibles (choisir un produit, le personnaliser, le mettre au panier et préparer une commande).

5.2. Difficultés rencontrées et solutions apportées

Comme tout projet de développement, plusieurs obstacles ont dû être surmontés :

- **Gestion des dépendances** : la coexistence entre Laravel, React et TailwindCSS a nécessité un temps d'adaptation, résolu grâce à l'utilisation de Vite et à une structuration claire des répertoires.
- **Accès aux données** : la mise en place correcte des relations entre tables (ex. panier ↔ articles) a représenté un défi. L'utilisation d'Eloquent ORM a simplifié l'écriture des requêtes complexes.
- **Logique métier du panier** : fusionner des articles similaires (même maillot, taille et personnalisation) demandait une réflexion particulière. La solution a consisté à encapsuler cette règle dans le contrôleur puis à envisager un déplacement futur dans le modèle.
- **Sécurité et sessions** : la gestion des connexions utilisateurs et la mise en place de politiques d'accès ont été essentielles pour éviter toute manipulation indue des paniers.

Ces difficultés ont permis de progresser en autonomie et de consolider la compréhension des bonnes pratiques de développement.

5.3. Perspectives d'évolution

Bien que le projet soit fonctionnel, plusieurs améliorations et extensions sont envisageables :

- **Finalisation du module Commandes** : implémenter le modèle Order, gérer le passage en caisse et l'historique des commandes.
- **Intégration des paiements en ligne** : connecter l'application à une API tierce (Stripe, PayPal) pour finaliser le processus d'achat.
- **Mise en place de la partie Administrateur** : l'administrateur doit pouvoir accéder à l'ensemble des données utilisateurs, gérer les stocks disponibles et pouvoir ajouter ou supprimer des maillots.
- **Amélioration des performances** : mise en cache des pages catalogue, optimisation SQL et intégration d'un CDN pour les images.

- **Renforcement de la sécurité** : authentification à deux facteurs (2FA), gestion avancée des sessions utilisateurs.
- **Tests automatisés** : mise en place de tests unitaires et fonctionnels avec PHPUnit et Laravel Dusk, afin de valider le bon fonctionnement en continu.
- **Déploiement évolutif** : automatisation via Docker ou CI/CD (GitHub Actions, GitLab CI) pour simplifier la maintenance et la montée en charge.

Conclusion générale

Ce projet a constitué une expérience formatrice, permettant de passer de la conception à la mise en œuvre complète d'une application web. L'association entre **technologies modernes** (Laravel, React, Tailwind, Inertia, Vite) et **bonnes pratiques de développement** (ORM, migrations, sécurité, documentation, déploiement) illustre pleinement les compétences acquises dans le cadre du TP DWWM.

Il offre une base solide pour des évolutions futures, mais aussi un exemple concret de mise en situation professionnelle, préparant à intégrer des projets réels en entreprise.

Résumé – Conclusion

Le projet **Fou-2-Foot** a permis de concevoir et de développer une application e-commerce spécialisée dans la vente de maillots personnalisés. Il s'inscrit dans le cadre du **Titre Professionnel Développeur Web et Web Mobile (DWWM)** et a couvert l'ensemble des compétences attendues, du front-end au back-end jusqu'au déploiement.

Sur le plan **technique**, l'application repose sur une architecture moderne :

- **Front-end** : React, TailwindCSS et Inertia.js pour une interface responsive, dynamique et accessible.
- **Back-end** : Laravel et MySQL pour la gestion des données, Eloquent ORM pour simplifier les requêtes, et une logique métier structurée (gestion du panier, règles de tarification, sécurité).

- **Déploiement** : documentation complète du processus d'installation et de mise en production sur un serveur Linux avec Nginx et MySQL.

Sur le plan **méthodologique**, ce projet a renforcé la capacité à analyser un besoin, à structurer un projet logiciel, à travailler sur Git et à documenter les étapes techniques de manière claire et reproductible.

Plusieurs **difficultés** ont jalonné le développement (gestion des dépendances, relations de données, fusion d'articles similaires dans le panier), mais elles ont été surmontées grâce à la maîtrise progressive de Laravel, d'Eloquent et des bonnes pratiques de sécurité.

Enfin, des **perspectives d'évolution** sont identifiées : finalisation du module Commandes et du paiement en ligne, amélioration des performances, mise en place de tests automatisés et intégration d'outils modernes de déploiement (Docker, CI/CD).

En conclusion, ce projet constitue une **preuve de concept fonctionnelle** et une expérience professionnalisante. Il démontre la capacité à conduire un projet web complet, depuis la conception jusqu'au déploiement, et constitue une base solide pour des évolutions futures en contexte professionnel.