

DOSSIER PROFESSIONNEL (DP)

Nom de naissance ▶ MOREIRA
Nom d'usage ▶
Prénom ▶ Stéphane
Adresse ▶ 2 Place des arcades
06250 Mougins

Titre professionnel visé

Développeur Web et Web Mobile

MODALITÉ D'ACCÈS :

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

DOSSIER PROFESSIONNEL ^(DP)

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.



Sommaire

Exemples de pratique professionnelle

Développer la partie front-end d'une application web ou web mobile sécurisée	p.	5
▶ Installer et configurer son environnement de travail en fonction du projet web ou web mobile	p.	5
▶ Maquetter des interfaces utilisateur web ou web mobile	p.	9
▶ Réaliser des interfaces utilisateur statiques web ou web mobile	p.	12
▶ Développer la partie dynamique des interfaces utilisateur web ou web mobile	p.	14
Développer la partie back-end d'une application web ou web mobile sécurisé	p.	17
▶ Mettre en place une base de données relationnelle	p.	17
▶ Développer des composants d'accès aux données SQL et NoSQL	p.	23
▶ Développer des composants métier coté serveur	p.	27
▶ Documenter le déploiement d'une application dynamique web ou web mobile		30
Titres, diplômes, CQP, attestations de formation <i>(facultatif)</i>	p.	33
Déclaration sur l'honneur	p.	34
Documents illustrant la pratique professionnelle <i>(facultatif)</i>	p.	35
Annexes <i>(Si le RC le prévoit)</i>	p.	36

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1 Développer la partie front-end d'une application web ou web mobile sécurisée

Exemple 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 :

Pour ma soutenance, j'ai choisi de créer un e-commerce fictif qui réunirait toutes les compétences demandés.

2. Précisez les moyens utilisés :

L'IDE (Integrated Development Environment) que j'utilise est **VisualStudioCode**.

Téléchargement de VisualStudioCode: <https://code.visualstudio.com/download>

Voici la liste des technologies que j'ai utilisées pour mon projet:

- ✧ Backend : **Laravel** 12.0 (PHP ^8.2)
- ✧ Administration : **Filament** v3.3
- ✧ Frontend : **Inertia.js** 2.0 (React 19)
- ✧ Styles : **TailwindCSS** 4.0, **DaisyUI** 5.0.28, **Framer Motion** 12.9.2
- ✧ Base de données : **SQLite** (par défaut)
- ✧ Build : **Vite** 6.0

Pour les frameworks Laravel et Inertia, j'ai choisi d'utiliser un starter kit que j'ai trouvé grâce à mes veilles, déjà configuré en JS pour utiliser du JSX au lieu du TSX:

<https://medium.com/@yazidkhalidi/converting-laravels-react-starter-kit-from-typescript-to-javascript-1691d38e5d3d>

Il est à noter que TailwindCSS et Vite sont **inclus** dans le starterkit.

Pour initier mon projet j'ai donc utilisé la commande suivante dans mon terminal:

```
laravel new --using=YazidKHALDI/react-jsx-starter-kit piwee-laravel
```

Enfin j'ai suivi la **documentation officielle** de Laravel pour la suite de l'installation.

Pour importer les bibliothèques DaisyUI et Motion, j'ai ouvert un terminal de commande à la racine de mon projet bien initialisé et j'ai utilisé la commande:

```
npm install daisyui@latest  
npm install framer-motion
```

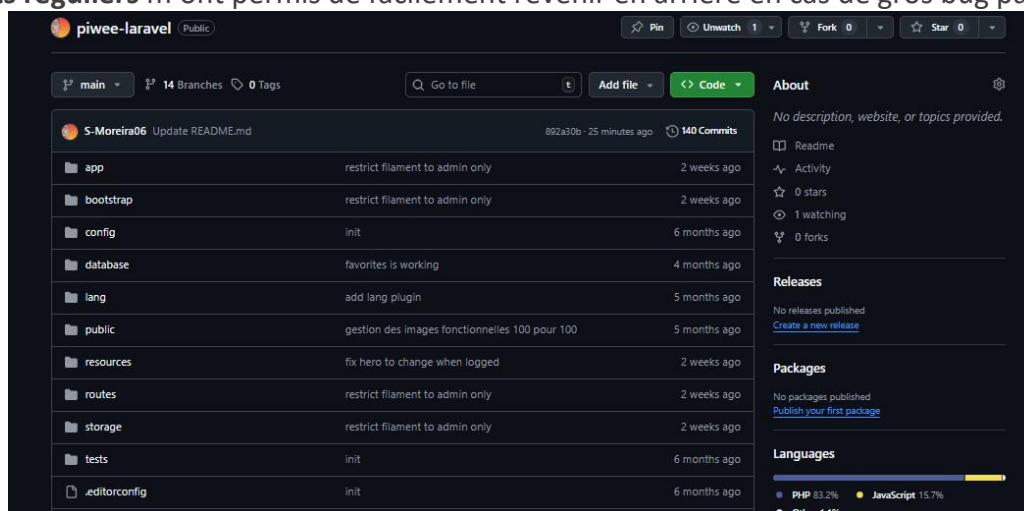
Pour la mise en place du framework Filament, j'ai suivi la documentation du site officiel, et voici la commande pour initier l'installation:

```
composer require filament/filament:"^3.3" -W  
php artisan filament:install --panels
```

DOSSIER PROFESSIONNEL (DP)

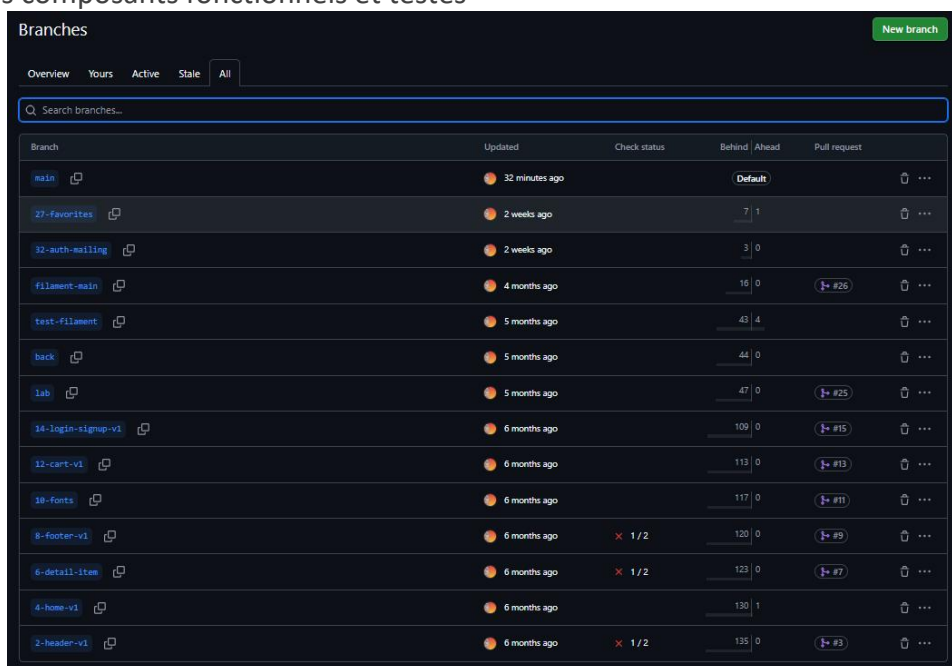
Enfin tout au long de mon projet, j'ai utilisé les fonctionnalités de **Git** et **GitHub** afin d'organiser et de versionner mon projet.

Des **commits réguliers** m'ont permis de facilement revenir en arrière en cas de gros bug par exemple:



Git add , Git commit, Git push!

Et j'ai utilisé **une branche** par fonctionnalité, ce qui m'a permis de merge sur ma branche main uniquement les composants fonctionnels et testés



Voici quelques unes de mes branches

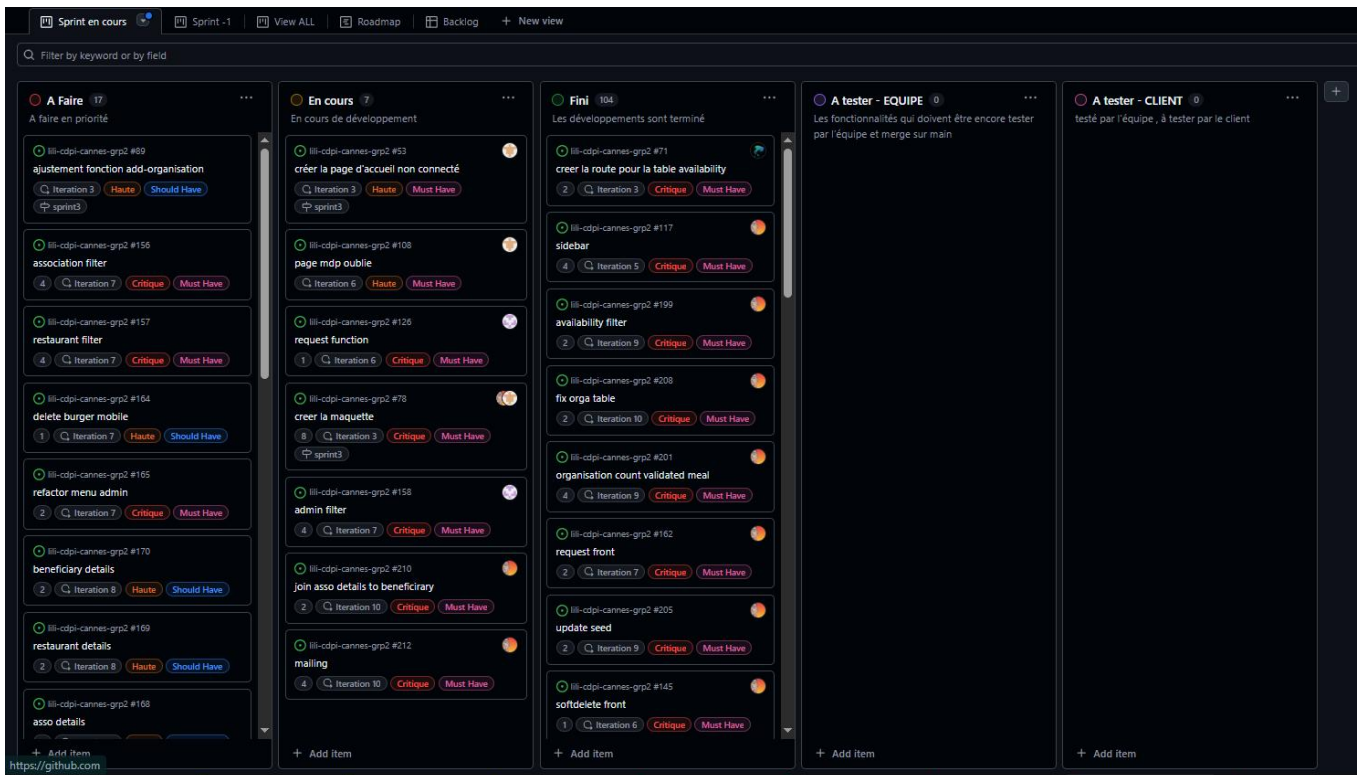
J'ai voulu utiliser githubproject mais je me suis vite rendu compte que c'était une perte de temps pour du développement en solo, mais j'ai pu «gerer» un **projet de groupe** lors d'un **Atelier** (Periode d'entreprise au sein même de l'école) :

Nous étions en groupe de **5 personnes** pour créer un backoffice pour une association: La Petite Lili. Mais nous n'allons pas rentrer plus dans les détails du site en lui même mais plutôt du **githubproject**.

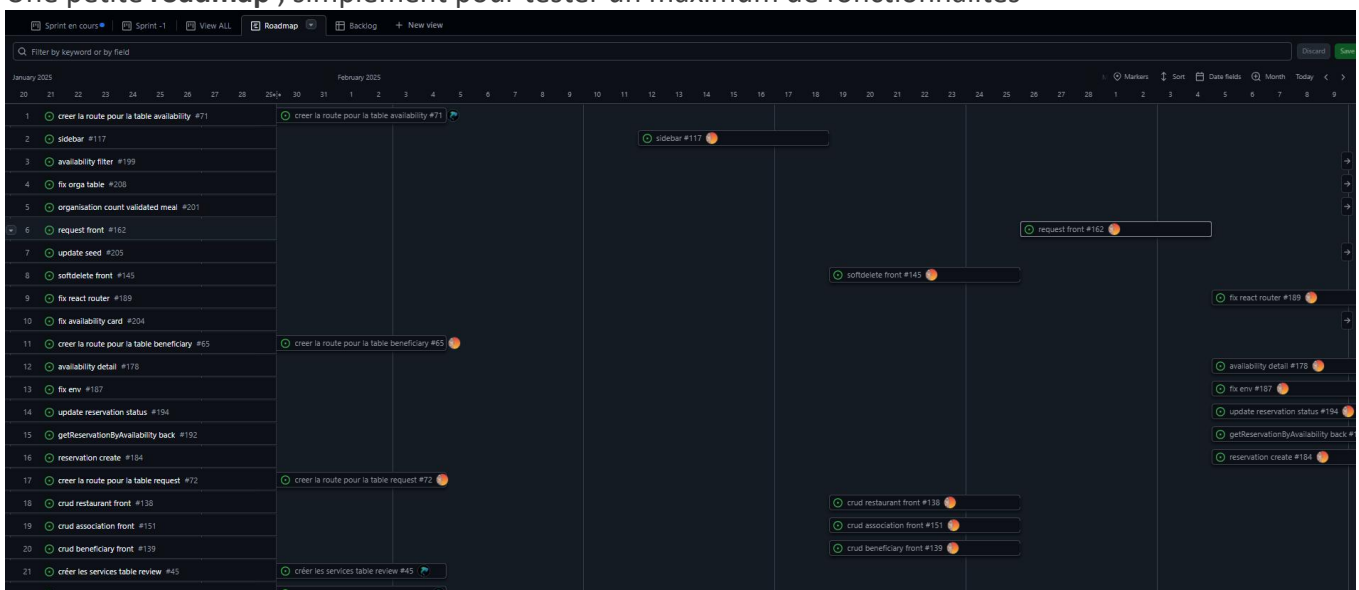
DOSSIER PROFESSIONNEL ^(DP)

J'ai donc organisé le projet en **sprint**, afin que nous puissions étaler notre travail sur toute la durée de l'atelier.

J'ai mis en place un onglet **Sprint en cours** et **Sprint -1** afin d'avoir une vue rapide sur le travail à accomplir et le travail en retard du sprint précédent.



Une petite **roadmap**, simplement pour tester un maximum de fonctionnalités



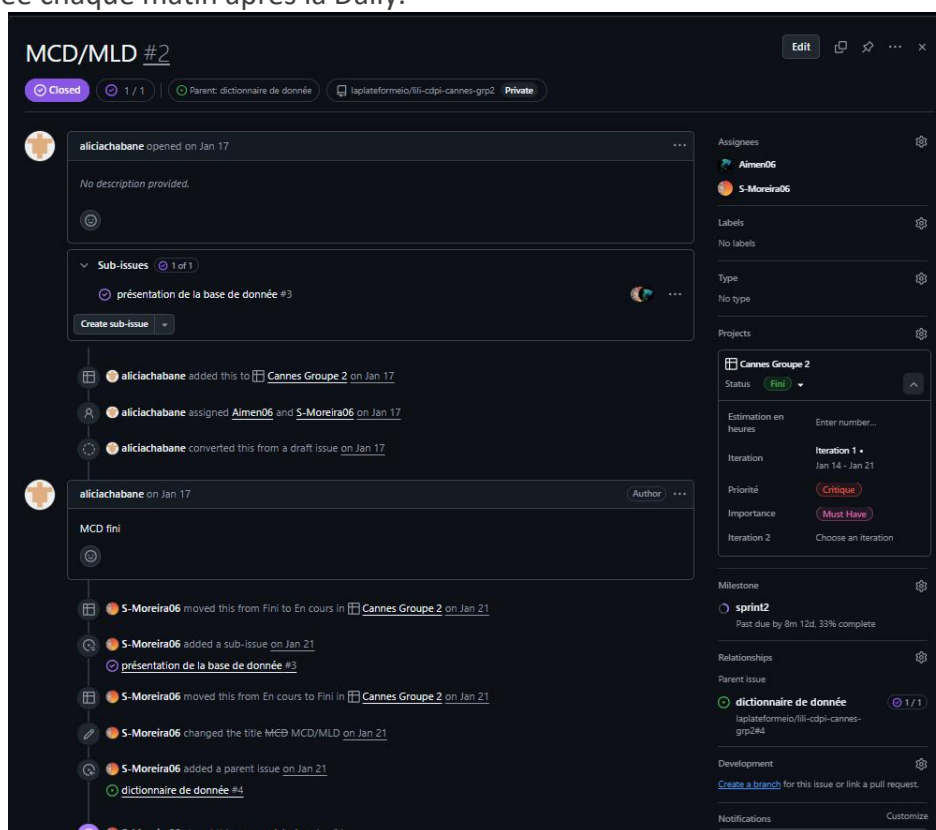
Chaque tâche est définie dans un **ticket** qui est attribué à un ou plusieurs contributeurs.

Chaque ticket est taggué par importance et par priorité et a généré **sa propre** banche pour que le

DOSSIER PROFESSIONNEL (DP)

travail soit organisé correctement , éviter que deux personnes travaillent sur une même fonctionnalité par exemple.

À la validation, une **review** a été demandée à un autre membre du groupe, et pour finir la **gestion des conflits** a été gérée chaque matin après la Daily.



3. Avec qui avez-vous travaillé ?

Seul / En groupe de 5 (Aïmen, Alicia, Hugo, Nouralah et moi)

4. Contexte

Nom de l'entreprise, organisme ou association ► Ecole LaPlateforme.io // L'Atelier de la Plateforme

Chantier, atelier, service ► Projet E-Commerce // Backoffice pour «La Petite Lili»

Période d'exercice ► Du : 01/04/2025 au : 15/07/2025

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL (DP)

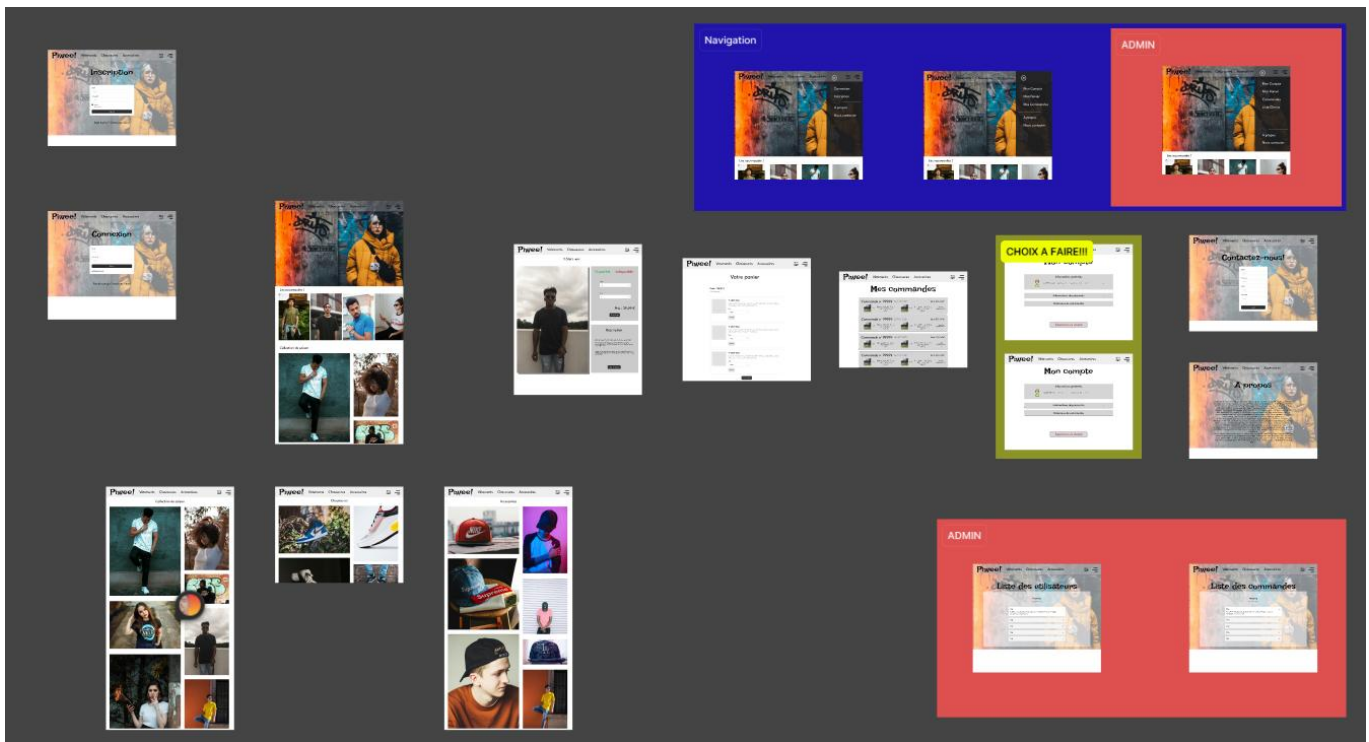
Exemple n°2 ► Maquetter des interfaces utilisateur web ou web mobile

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

J'ai réalisé un moodboard et un wireframe afin de prévisualiser l'apparence de mon site.

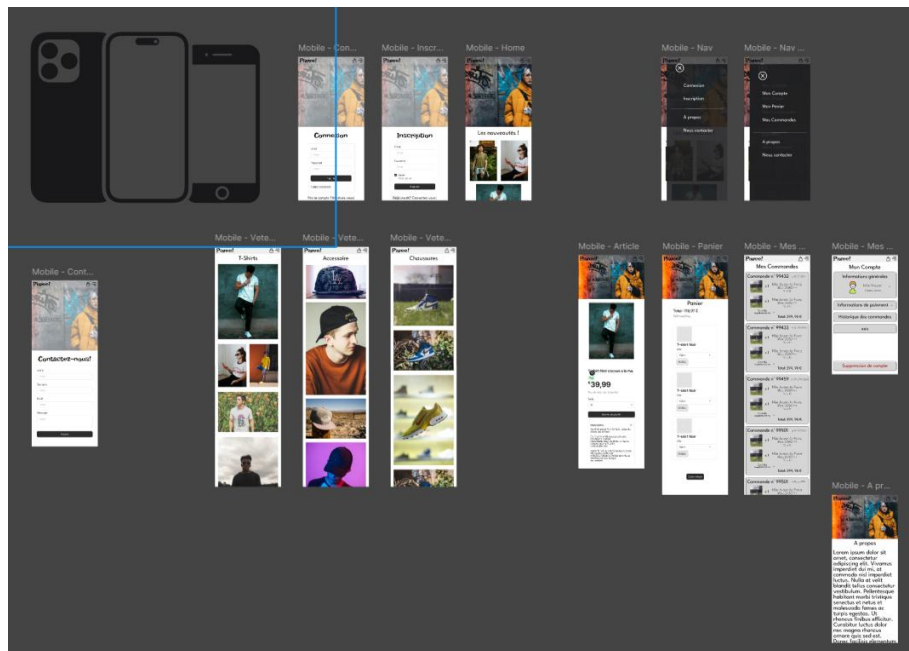
2. Précisez les moyens utilisés :

Pour structurer l'interface, un **wireframe** complet a été réalisé sous **Figma**, puis transformé en **prototype interactif** pour valider les parcours utilisateurs.

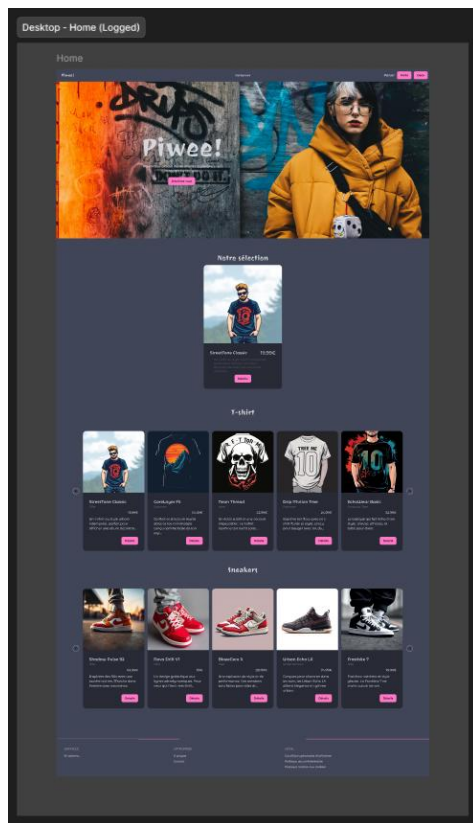


Wireframe desktop

DOSSIER PROFESSIONNEL (DP)



Wireframe mobile



Pour des raisons évidentes de temps, j'ai simplement créé la maquette de la **page d'accueil** avec les composants **DaisyUI**, assurant cohérence et rapidité d'implémentation.

DOSSIER PROFESSIONNEL ^(DP)

3. Avec qui avez-vous travaillé ?

Seul

4. Contexte

Nom de l'entreprise, organisme ou association ► Ecole LaPlateforme.io

Chantier, atelier, service ► Projet E-Commerce

Période d'exercice ► Du : 01/04/2025 au : 15/07/2025

5. Informations complémentaires *(facultatif)*

DOSSIER PROFESSIONNEL (DP)

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 :

J'ai réalisé plusieurs page **statique** pour mon site: CGU, Politique de cookies , page de contacte, etc... Pour cet exemple , j'ai choisi de vous présenter ma page «A propos».

2. Précisez les moyens utilisés :

Via **React** , j'ai d'abord créé une constante 'section' qui est un **tableau d'objets**, afin que mon code soit plus lisible, scalable et maintenable rapidement et facilement.

```
const sections = [
  {
    title: "1. Notre histoire",
    content: (
      <p>
        Piwee! a été créé pour offrir une expérience de shopping unique aux amoureux de mode. Passionnés par l'innovation et le
      </p>
    ),
  },
  {
    title: "2. Notre mission",
    content: (
      <p>
```

Début de mon tableau 'section' pour comprendre sa structure

J'ai ensuite mis en place une **boucle** pour afficher celui-ci:

```
<div className="">
  {sections.map((section) => (
    <div>
      <h2 className="">{section.title}</h2>
      {section.content}
    </div>
  ))}
</div>
```

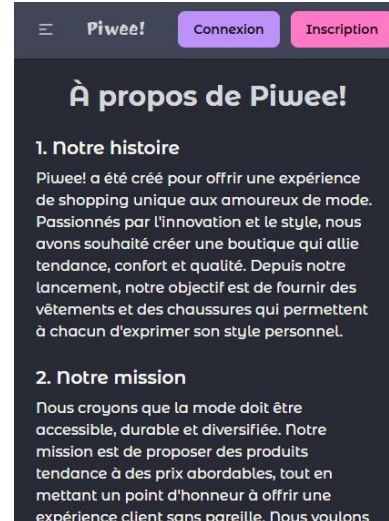
La fonction map() permet l'itération de mes sections avec animation Framer Motion uniforme.

DOSSIER PROFESSIONNEL (DP)

Et enfin : mise en page avec **TailwindCSS** et **DaisyUI** , puis petites animations avec **Framer Motion**:

```
export default function About() {
  return (
    <Layout className="min-h-screen">
      <Head title="À propos"> />
      <motion.section
        className="max-w-4xl mx-auto p-6 bg-base-100 shadow-lg my-10"
        initial={{ opacity: 0, y: 30 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ duration: 0.6, ease: "easeOut" }}
      >
        <h1 className="text-3xl font-bold mb-6 text-center">À propos de Piwee!</h1>

        <div className="space-y-6 text-base-content">
          {sections.map((section, index) => (
            <motion.div
              key={index}
              initial={{ opacity: 0, y: 20 }}
              animate={{ opacity: 1, y: 0 }}
              transition={{ duration: 0.5, delay: 0.2 + index * 0.1 }}
            >
              <h2 className="text-xl font-semibold mb-2">{section.title}</h2>
              <section.content>
            </motion.div>
          ))}
        </div>
      </motion.section>
    </Layout>
  );
}
```



3. Avec qui avez-vous travaillé ?

Seul

4. Contexte

Nom de l'entreprise, organisme ou association ► Ecole LaPlateforme.io

Chantier, atelier, service ► Projet E-Commerce

Période d'exercice ► Du : 01/04/2025 au : 15/07/2025

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL (DP)

Exemple 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 :

Lorsqu'un utilisateur clique sur un article, il est redirigé vers la page des détails de l'article en question. J'ai donc mis en place un système de gestion dynamique des tailles et des quantités à renvoyer au panier en fonction des stocks.

2. Précisez les moyens utilisés :

Initialisation de l'état dynamique avec `useState()`

```
const [selectedSize, setSelectedSize] = useState(item.sizes[0]?.size || "");
const [quantity, setQuantity] = useState(1);
const selectedStock = item.sizes.find(s => s.size === selectedSize)?.stock || 0;
const isQuantityTooHigh = quantity > selectedStock;
```

On instancie deux **états locaux** pour gérer la taille sélectionnée (`selectedSize`) et la quantité (`quantity`).

L'init de `selectedSize` propose **par défaut** la première taille disponible (UX).

`selectedStock` **calcule dynamiquement** le stock disponible pour la taille choisie, via un simple `find()` dans le tableau des tailles.

`isQuantityTooHigh` sert à afficher un **message d'erreur** en cas de dépassement du stock.

Sélection dynamique des tailles : **mapping, animation et accessibilité**

```
{item.sizes.map(({ size }) => (
  <motion.p
    key={size}
    whileHover={{ scale: 1.05 }}
    className={`badge px-2.5 place-self-center cursor-pointer ${
      selectedSize === size ? "badge-primary" : "badge-outline"
    }`}
    variants={{
      hidden: { opacity: 0, y: 10 },
      visible: { opacity: 1, y: 0 },
    }}
    onClick={() => setSelectedSize(size)}
  >
    {size}
  </motion.p>
)}
```

On affiche **dynamiquement** toutes les tailles disponibles pour le produit, sous forme de **badges interactifs**.

On utilise les **capacités d'animation** de **Motion** (`motion.p`, variante visible/hidden et effet au survol) pour améliorer l'UX.

Le visuel du badge reflète la `selectedSize` via la **classe DaisyUI** (`badge-XXX`).

Gestion de la quantité

```
<motion.button
  whileHover={{ scale: 1.1 }}
  className="btn btn-primary"
  onClick={() => setQuantity(q => Math.max(1, q - 1))}
>
-
</motion.button>
<p className="place-self-center">{quantity}</p>
<motion.button
  whileHover={{ scale: 1.1 }}
  className="btn btn-primary"
  onClick={() => setQuantity(q => q + 1)}
>
+
</motion.button>
```

L'utilisateur ajuste la quantité via deux boutons (+/-), la décrémentation ne descend jamais en dessous de 1 (sécurité UX).

Animation sur les boutons pour inviter l'action.

Affichage du nombre sélectionné au centre, mise à jour instantanée.

Système d'alerte et validation du stock

```
{isQuantityTooHigh && (
  <div className="text-error text-center mb-2">
    La quantité demandée dépasse le stock disponible ({selectedStock}).
  </div>
)}
```

Dès que l'utilisateur essaie de dépasser le stock, un message s'affiche en temps réel, évitant de mauvaises surprises au checkout.

Validation immédiate côté front sans attendre une réponse serveur.

3. Avec qui avez-vous travaillé ?

Seul

DOSSIER PROFESSIONNEL ^(DP)

4. Contexte

Nom de l'entreprise, organisme ou association ► Ecole LaPlateforme.io

Chantier, atelier, service ► Projet E-Commerce

Période d'exercice ► Du : 01/04/2025 au : 15/07/2025

5. Informations complémentaires *(facultatif)*

DOSSIER PROFESSIONNEL (DP)

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

Exemple n° 1 ► Mettre en place une base de données relationnelle

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

Conception de la base de données avec **MERISE**. Mise en place de la base de données en SQLite grâce aux **migrations** et **seeders Laravel**

2. Précisez les moyens utilisés :

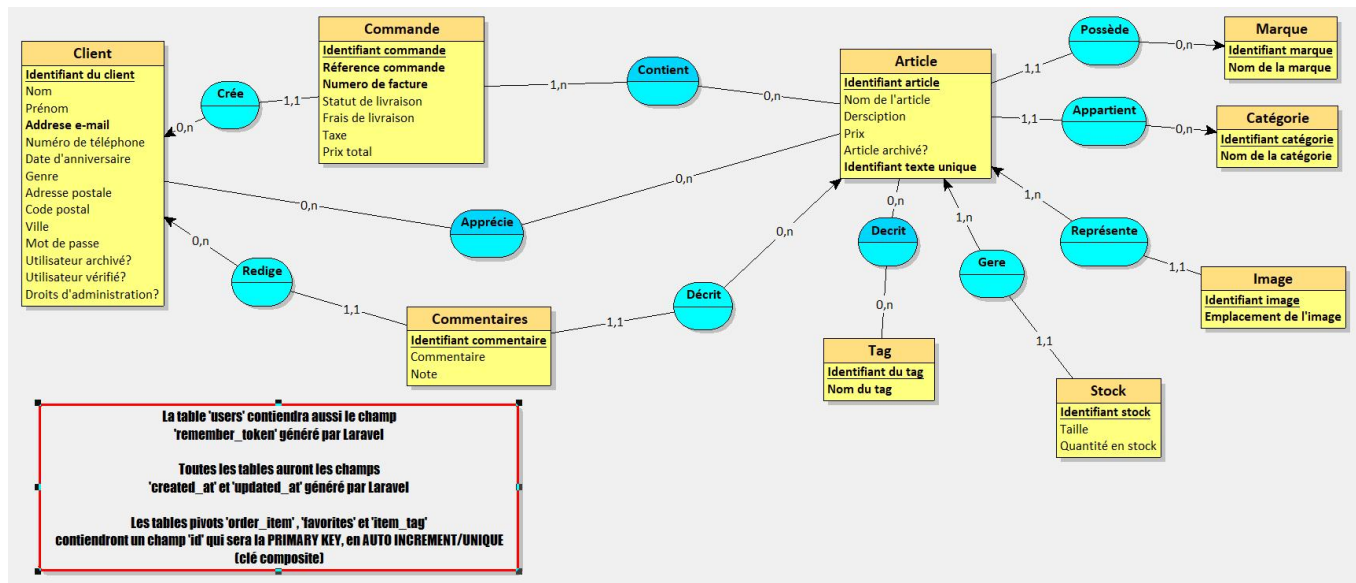
J'ai d'abord **conceptualisé** ma base de données avec la méthode Merise

Dictionnaire de données non-technique:

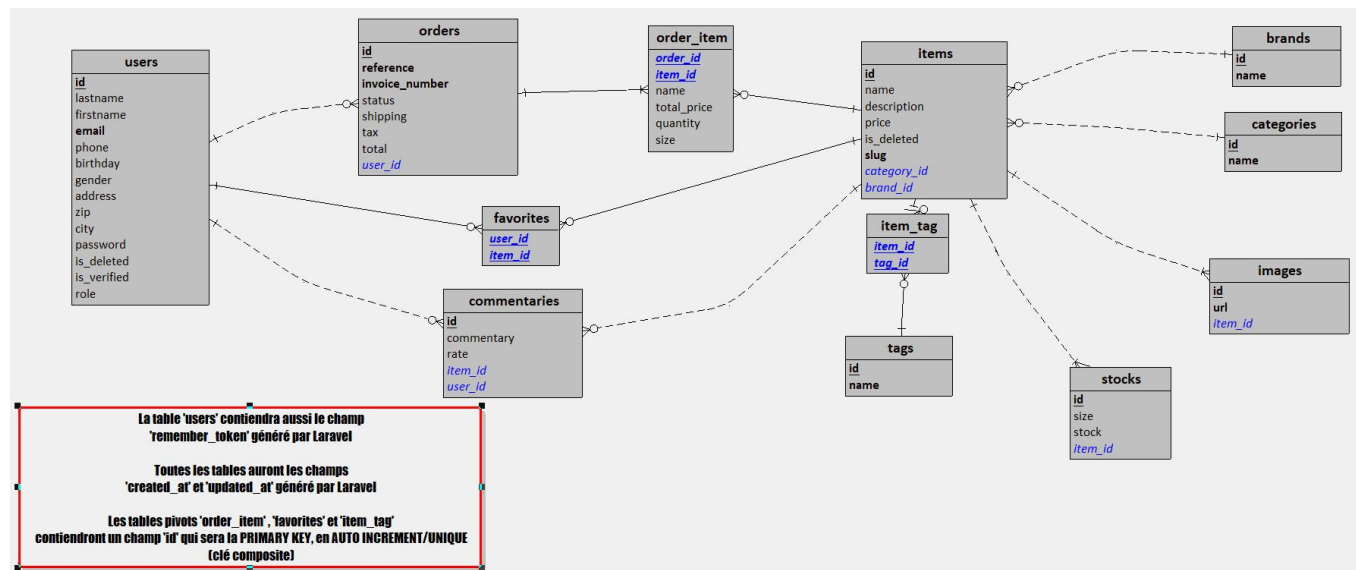
Nom de la données	Format	Longueur	Contraintes
Nom	Alphabétique	30	Obligatoire
Prénom	Alphabétique	30	Obligatoire
Adresse E-mail	Alphanumérique	50	Obligatoire, unique
Date d'anniversaire	Date	-	Obligatoire
Genre	Alphabétique	10	Obligatoire
Adresse	Alphanumérique		Obligatoire
Code postal	Alphanumérique	6	Obligatoire
Ville	Alphabétique	30	Obligatoire
Mot de passe	Alphanumérique	>6	Obligatoire
Document : Client			
Nom de la données	Format	Longueur	Contraintes
Nom de l'article	Alphanumérique	30	Obligatoire
Marque	Alphanumérique	30	Obligatoire
Type de vêtements	Alphabétique	30	Obligatoire, unique
Prix	Numérique	-	Obligatoire
Document : Article			
Nom de la données	Format	Longueur	Contraintes
Référence unique	Alphanumérique	6	Obligatoire, unique
Frais de livraisons	Numérique	-	Obligatoire
Taxe	Numérique	-	Obligatoire, unique
Numero de facture	Alphanumérique	10	Unique
Statut de livraison	Alphabétique	10	Obligatoire
Document : Commande			

DOSSIER PROFESSIONNEL (DP)

MCD (utilisation de Looping.exe)

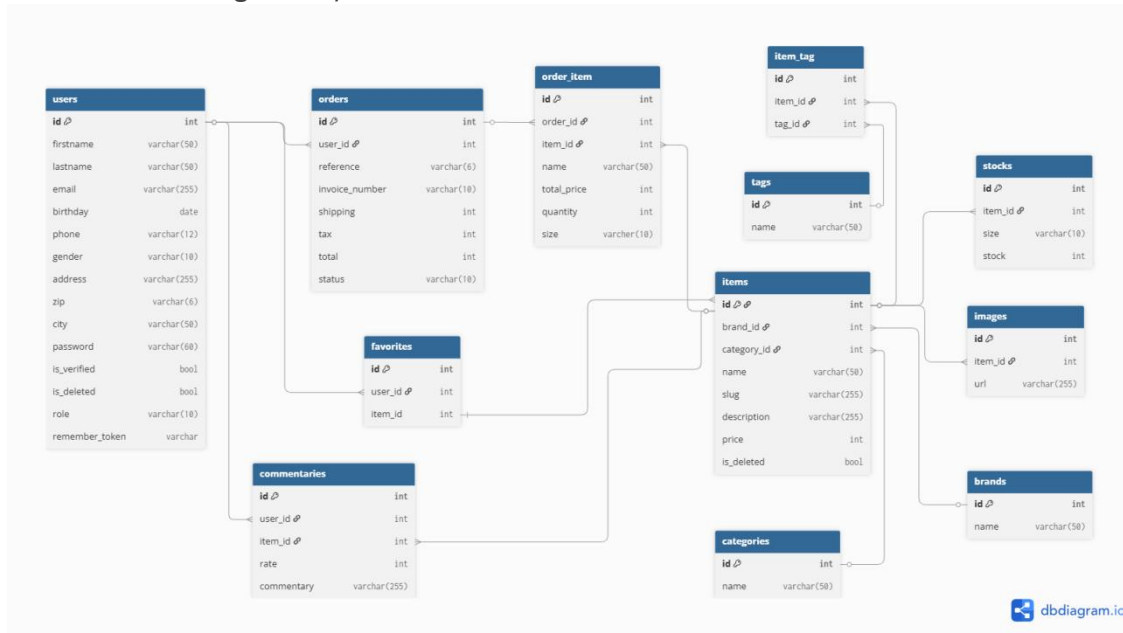


MLD (utilisation de Looping.exe)



DOSSIER PROFESSIONNEL (DP)

MPD (utilisation de DbDiagram.io)



Dictionnaire de données technique (exemples pour users, orders, items)

Nom de la données	Format	Longueur	Contraintes
lastname	Varchar	50	Not null
firstname	varchar	50	Not null
email	varchar	255	Not null, unique
birthday	Date	-	Not null
gender	varchar	10	Not null
address	varchar	255	Not null
zip	varchar	6	Not null
city	varchar	50	Not null
password	varchar	60	Not null
is_deleted	boolean		Not null
is_verified	boolean		Not null
role	varchar	10	Not null
Document : users			

Clé primaire: (id)

DOSSIER PROFESSIONNEL (DP)

Nom de la données	Format	Longueur	Contraintes
id	int	-	Not null, unique
user_id	int	-	Not null
reference	varchar	6	Not null, unique
invoice_number	varchar	10	Not null, unique
shipping	int	-	Not null
tax	int	-	Not null
total	int	-	Not null
status	varchar	10	Not null
Document : orders			

Clé primaire: (id) // Clé(s) secondaire(s): (user_id)

Nom de la données	Format	Longueur	Contraintes
id	int	-	Not null, unique
name	varchar	50	Not null, unique
brand_id	int	-	Not null
category_id	int	-	Not null
description	varchar	255	Not null, unique
slug	varchar	255	Not null, unique
price	int	-	Not null
is_deleted	boolean	-	Not null
Document : items			

Clé primaire: (id) // Clé(s) secondaire(s): (brand_id, category_id)

Ensuite , on met en place notre structure avec les **migrations**. Celles ci nous permettent de **versionner** et **automatiser** les tables de notre base de données.

```
Schema::create('items', function (Blueprint $table) {
    $table->id();
    $table->foreignId('brand_id')->constrained('brands')->cascadeOnDelete();
    $table->foreignId('category_id')->constrained('categories')->cascadeOnDelete();
    $table->string('name');
    $table->string('slug');
    $table->string('description');
    $table->unsignedInteger('price');
    $table->boolean('isDeleted');
    $table->timestamps();
});
```

Création de la table items

'Schema' est la **façade** qui va nous permettre de gérer nos structures de tables. Grâce a la fonction create(), a qui on passe en **paramètre** le nom de la table, puis avec la **syntaxe proposée sur la documentation** nous pouvons spécifier les nom des champs ainsi que leurs type.

Pour les champs brand_id et category_id, j'ai spécifié que ce sont des **clés étrangères**, et ajouté la **contrainte** avec la table concernée ainsi que le fait que si une catégorie ou une marque est supprimée, les articles liés sont aussi supprimés (**suppression en cascade**)

DOSSIER PROFESSIONNEL (DP)

Mise en place de seeders pour les marques , les catégories et les 40 premiers articles

```
<?php

namespace Database\Seeders;

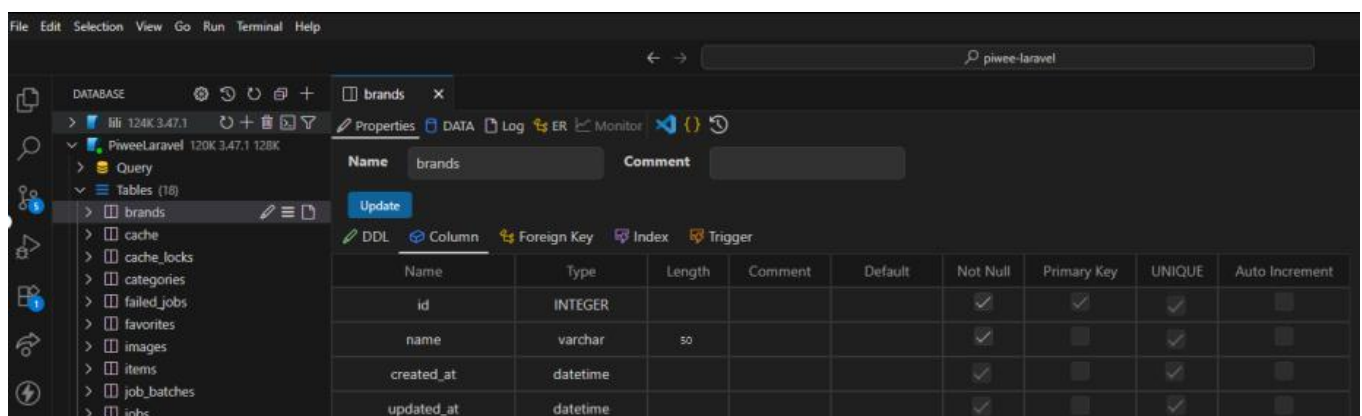
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use App\Models\Brand;
use Illuminate\Database\Seeder;

class BrandSeeder extends Seeder
{
    public function run(): void
    {
        $brands = [
            ['id' => 1, 'name' => 'Nike'],
            ['id' => 2, 'name' => 'Adidas'],
            ['id' => 3, 'name' => 'Puma'],
            ['id' => 4, 'name' => 'Reebok'],
            ['id' => 5, 'name' => 'Under Armour'],
            ['id' => 6, 'name' => 'New Balance'],
            ['id' => 7, 'name' => 'Asics'],
            ['id' => 8, 'name' => 'Converse'],
            ['id' => 9, 'name' => 'Vans'],
            ['id' => 10, 'name' => 'Skechers'],
            ['id' => 11, 'name' => 'Fila'],
            ['id' => 12, 'name' => 'Kappa'],
            ['id' => 13, 'name' => 'Lacoste'],
            ['id' => 14, 'name' => 'Le Coq Sportif'],
            ['id' => 15, 'name' => 'Diadora'],
            ['id' => 16, 'name' => 'K-Swiss'],
            ['id' => 17, 'name' => 'Onitsuka Tiger'],
            ['id' => 18, 'name' => 'Hoka One One'],
            ['id' => 19, 'name' => 'Salomon'],
            ['id' => 20, 'name' => 'Mey'],
        ];

        foreach ($brands as $brand) {
            Brand::updateOrCreate(['id' => $brand['id']], $brand);
        }
    }
}
```

Seeder pour la table Brand

Pour visualiser mon fichier sqlite, j'utilise l'extension **vscode Database Client**:



DOSSIER PROFESSIONNEL ^(DP)

3. Avec qui avez-vous travaillé ?

Seul

4. Contexte

Nom de l'entreprise, organisme ou association ► Ecole LaPlateforme.io

Chantier, atelier, service ► Projet E-Commerce

Période d'exercice ► Du : 01/04/2025 au : 15/07/2025

5. Informations complémentaires *(facultatif)*

DOSSIER PROFESSIONNEL (DP)

Exemple 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 :

Après avoir créé les modèles Eloquent, j'ai pu créer la plupart de mes CRUD grâce à Filament.

2. Précisez les moyens utilisés :

Eloquent ORM

Syntaxe **orientée objet** expressive. Évite les **boilerplates** (sections de code répétées à plusieurs endroits) et les **injections SQL**. J'utilise **Eloquent** lorsque **Filament** n'est pas pertinent (pour l'inscription ou les favoris par exemple).

```
public function destroy($itemId)
{
    $userId = auth()->id();

    Favorite::where('user_id', $userId)
        ->where('item_id', $itemId)
        ->delete();

    return back()->with('success', 'Article retiré des favoris');
}
```

Dans le controller, voici la fonction pour supprimer un favoris avec l'utilisation du modèle Favorite. On restreint notre requête grâce aux conditions where et on supprime

Filament

Permet la création de **panel d'administration complet**. Grâce aux **ressources** Filament, on peut très rapidement créer un **CRUD** sur la **base d'un modèle Eloquent**.

Voici un exemple pour la table Item:

```
class ItemResource extends Resource
{
    protected static ?string $model = Item::class;
```

On crée l'objet «ItemRessource» et on y récupère le modèle Item existant.

Create/Update

```
public static function form(Form $form): Form
```

La fonction form() correctement utilisé initie le formulaire de création et de modification !

DOSSIER PROFESSIONNEL (DP)

On va donc y déclarer les inputs (soit les champs de notre table item)

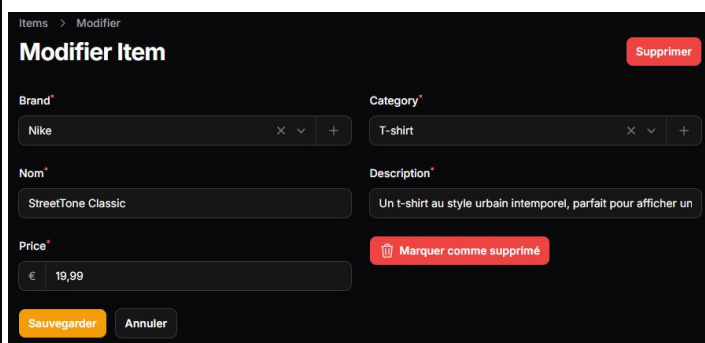
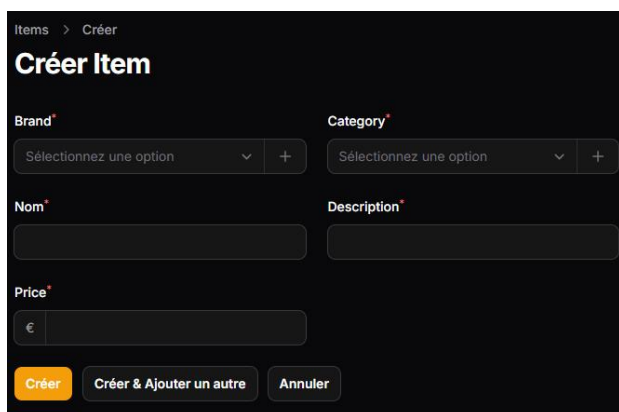
```
Forms\Components\TextInput::make('price')
    ->numeric()
    ->prefix('€')
    ->maxValue(42949672.95)
    ->required(),
```

Voici un exemple concret avec le champ «price». Ce bout de code va créer un **input de type texte**. On peut ajouter des **paramètres** à celui-ci afin de limiter l'utilisateur pour qu'il entre des données **valide** pour notre base de données, rendre le champ **obligatoire**, etc

```
Forms\Components\Select::make('brand_id')
    ->relationship('brand', 'name')
    ->searchable()
    ->preload()
    ->createOptionForm([
        Forms\Components\TextInput::make('name')
            ->label('Nom de la marque')
            ->required()
            ->maxLength(50),
    ])
    ->required(),
```

Ici un exemple d'une **clé étrangère** avec la marque. `->relationship()` avec en **paramètre** la table et le champ en relation avec notre champ «brand_id»
`->searchable()` limite l'administrateur à utiliser une **entrée existante** dans la table «brand», ou à en créer une à la volée grâce à `->createOptionForm`

Voici le rendu :



Formulaire de **création** et de **modification** d'article

DOSSIER PROFESSIONNEL (DP)

Read

```
public static function table(Table $table): Table
{
    return $table
        ->columns([
            Tables\Columns\TextColumn::make('name')
                ->tooltip(fn ($record) => $record->name)
                ->limit(30)
                ->searchable(),
        ])
```




J'utilise la fonction table, pour lire les données, qui seront automatiquement mis en page avec le panel filament

Items > Liste

Items

Créer

Rechercher

Name	Image	Price	Brand	Category	Description	Tailles disponibles	
StreetTone Classic		19.99	Nike	T-shirt	Un t-shirt au style urbain int...	M L	Modifier Supprimer
Drip Motion Tree		24.99	Salomon	T-shirt	Exprime ton flow avec ce t-shi...	L S	Modifier Supprimer
Pulse Fade		29.99	Salomon	T-shirt	Un dégradé moderne qui capte l...	M L XL	Modifier Supprimer

```
->actions([
    Tables\Actions\EditAction::make(),
    Tables\Actions\DeleteAction::make(),
])
```

Le **delete** se met en place automatiquement en appelant le bouton dans `->actions()`, et j'ai implémenté un soft delete afin de pouvoir garder une **traçabilité** dans ma base de données disponible dans le formulaire de modification

3. Avec qui avez-vous travaillé ?

Seul

DOSSIER PROFESSIONNEL ^(DP)

4. Contexte

Nom de l'entreprise, organisme ou association ► Ecole LaPlateforme.io

Chantier, atelier, service ► Projet E-Commerce

Période d'exercice ► Du : 01/04/2025 au : 15/07/2025

5. Informations complémentaires *(facultatif)*

DOSSIER PROFESSIONNEL (DP)

Exemple n° 3 ► Développer des composants métier coté serveur

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

J'ai développé plusieurs composants métier et j'ai choisi de vous présenter **la gestion de mes articles par un administrateur grâce à Filament**. Son rôle sera de:

- ✧ Permettre la **création** et la **modification** rapide des articles via un formulaire ergonomique.
- ✧ **Gérer les liaisons essentielles** : chaque article est relié à une marque, une catégorie, des images et des stocks (par taille).
- ✧ **Superviser le catalogue** : visualiser, filtrer, éditer et supprimer les articles selon les besoins.
- ✧ Assurer la **gestion du cycle de vie de l'article** (activation, suppression logique, réactivation).

2. Précisez les moyens utilisés :

J'ai mis en place le **modèle Eloquant** en paramétrant les **fillable**. C'est aussi ici que j'ai mis en place le système de création de **slug**:

```
protected $fillable = [
    'brand_id',
    'category_id',
    'name',
    'slug',
    'description',
    'price',
    'isDeleted',
];

protected static function booted()
{
    static::saving(function ($item) {
        if (!empty($item->name)) {
            $item->slug = Str::slug($item->name);
        }
    });
}
```

L'événement saving se déclenche avant que l'enregistrement ne soit sauvegardé en base de données, que ce soit lors d'une création (create) ou d'une mise à jour (update).

et défini les relations avec les autres entités:

```
public function brand(): BelongsTo
{
    return $this->belongsTo(Brand::class);
}

public function category(): BelongsTo
{
    return $this->belongsTo(Category::class);
}
```

Un article appartient à une marque et une catégorie

```
public function images(): HasMany
{
    return $this->hasMany(Image::class);
}

public function stocks()
{
    return $this->hasMany(Stock::class);
}
```

Un article peut avoir plusieurs images et plusieurs variantes/tailles en stock

```
public function favorites()
{
    return $this->hasMany(Favorite::class);
}

public function favoritedBy()
{
    return $this->belongsToMany(User::class, 'favorites');
}
```

Relations pour gerer les favoris

belongsToMany(): Chaque entité peut être liée à plusieurs autres entités, et réciproquement.

Puis on a centralisé toute la logique métier dans une **ressource filament** «ItemResource» qui regroupe toute la **logique d'administration des articles** :

- **Formulaire de création/édition** (vu précédemment)
- **Tableau de bord** (listing, tris, filtres)
- **Actions personnalisées** (suppression logique, réactivation)
- **Gestion des relations** (images, stocks)

Gestion des images d'un article

ImageRelationManager.php va nous permettre de customiser notre composant Image:

```
class ImagesRelationManager extends RelationManager
{
    protected static string $relationship = 'images';
}
```

Dans notre nouvelle classe , on appelle la méthode image() de notre ressource

```
public function form(Forms\Form $form): Forms\Form
{
    return $form->schema([
        Forms\Components\FileUpload::make('url')
            ->disk('public')
            ->directory('img')
            ->image()
            ->required(),
    ]);
}
```

On crée un champ d'upload pour le champ url de la base de données, en spécifiant le disk (défini dans config/filesystems.php) et le dossier dans lequel sera enregistré le fichier. On limite le type de fichier a des images et on rend le champ obligatoire

Pour l'affichage:

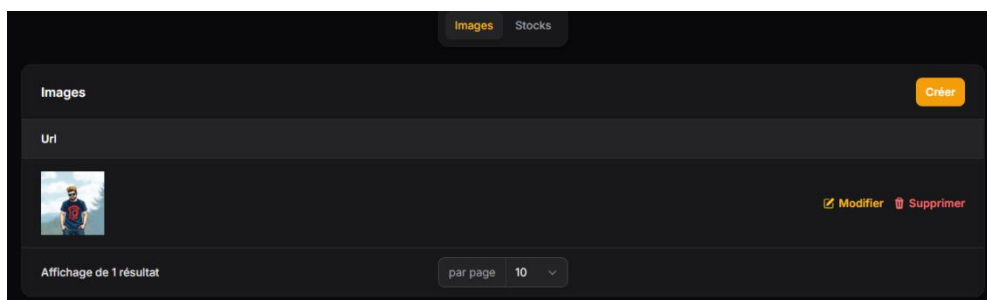
```
public function table(Tables\Table $table): Tables\Table
{
    return $table
        ->columns([Tables\Columns\ImageColumn::make('url')->disk('public')->size(80),])
        ->headerActions([CreateAction::make(),])
        ->actions([Tables\Actions\EditAction::make(), Tables\Actions\DeleteAction::make(),]);
}
```

->columns() : on recuper l'url de notre bdd, on cherche dans le disk public et on donne une taille de 80 a nos miniatures

->headerActions().... Nous permet d'ajouter un bouton de création d'image en haut du composant et -

->actions permet d'ajouter des boutons de modification et de suppression à chaque ligne du composant

DOSSIER PROFESSIONNEL (DP)



3. Avec qui avez-vous travaillé ?

Seul

4. Contexte

Nom de l'entreprise, organisme ou association ► Ecole LaPlateforme.io

Chantier, atelier, service ► Projet E-Commerce

Période d'exercice ► Du : 01/04/2025 au : 15/07/2025

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL ^(DP)

Exemple 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 :

J'ai documenté le déploiement dans un fichier README.

2. Précisez les moyens utilisés :

Tout au long du développement, j'ai pris des notes sur la mise en place de mon environnement , de mes dépendances et de ma base de données.

J'ai donc rédigé un dossier README complet disponible a ce lien:

<https://github.com/S-Moreira06/piwee-laravel/blob/main/README.md>

DOSSIER PROFESSIONNEL (DP)

🔧 Installation

🔗 Prérequis

- PHP ^8.2
- Composer
- Node.js ^20
- SQLite (ou autre base de données)

🔗 1. Cloner le projet

```
git clone https://github.com/S-Moreira06/piwee-laravel.git
cd piwee
```

🔗 2. Installation des dépendances

```
# Dépendances PHP
composer install

# Dépendances Node.js
npm install
```

🔗 3. Configuration de l'environnement

```
# Copier le fichier d'environnement
cp .env.example .env

# Générer la clé d'application
php artisan key:generate

# Créer la base de données SQLite
touch database/database.sqlite
```

🔗 4. Configuration de la base de données

Modifier le fichier `.env` selon vos besoins :

```
APP_NAME=Piwee
APP_ENV=local
APP_DEBUG=true
APP_URL=http://localhost:8000

DB_CONNECTION=sqlite
# Ou pour MySQL :
# DB_CONNECTION=mysql
# DB_HOST=127.0.0.1
# DB_PORT=3306
# DB_DATABASE=piwee
# DB_USERNAME=root
# DB_PASSWORD=
```

🔗 5. Migrations et données

```
# Exécuter les migrations
php artisan migrate

# (Optionnel) Exécuter les seeders
php artisan db:seed
```

🔗 6. Build des assets

```
# Pour le développement
npm run dev

# Pour la production
npm run build
```

🚀 Démarrage

🔗 Mode Développement

```
# Option 1 : Script automatisé (recommandé)
composer run dev
# Lance automatiquement : serveur PHP, queue worker, et Vite

# Option 2 : Manuel
php artisan serve
npm run dev
```

🔗 Mode Production

```
npm run build
php artisan serve --env=production
```

DOSSIER PROFESSIONNEL ^(DP)

3. Avec qui avez-vous travaillé ?

Seul

4. Contexte

Nom de l'entreprise, organisme ou association ► Ecole LaPlateforme.io

Chantier, atelier, service ► Projet E-Commerce

Période d'exercice ► Du : 01/04/2025 au : 15/07/2025

5. Informations complémentaires *(facultatif)*

DOSSIER PROFESSIONNEL ^(DP)

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.

DOSSIER PROFESSIONNEL ^(DP)

Déclaration sur l'honneur

Je soussigné(e) Stéphane MOREIRA ,
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 Cannes le 20/08/2025

pour faire valoir ce que de droit.

Signature :

DOSSIER PROFESSIONNEL ^(DP)

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé
Cliquez ici pour taper du texte.

DOSSIER PROFESSIONNEL ^(DP)

ANNEXES
