Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance - *Razvetskaya*
Nom d'usage - *Razvetskaya*
Prénom - *Yanina*
Adresse - *142 avenue Francis Tonner*

Titre professionnel visé

Développeur web et web mobile

MODALITÉ D'ACCÈS :

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

Présentation du dossier

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

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

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

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

Pour prendre sa décision, le jury dispose :

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

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

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

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

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

Ce dossier comporte :

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

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

Sommaire

Exemples de pratique professionnelle

Développer la partie front-end d'une application web ou web mobile sécurisée	**p.**	**6**
CP1 Installer et configurer son environnement de travail en fonction du projet web ou web mobile.	p.	6
CP2 Maquetter des interfaces utilisateur web ou web mobile	p.	9
CP3 Réaliser des interfaces utilisateur statiques web ou web mobile	p.	14
CP4 Développer la partie dynamique des interfaces utilisateur web ou web mobile	p.	20
Développer la partie back-end d'une application web ou web mobile sécurisée	**p.**	**24**
CP5 Mettre en place une base de données relationnelle	p.	24
CP6 Développer des composants d'accès aux données SQL et NoSQL	p.	33
CP7 Développer des composants métier coté serveur	p.	35
CP 8 Documenter le déploiement d'une application dynamique web ou web mobile		37
Titres, diplômes, CQP, attestations de formation *(facultatif)*	**p.**	39
Déclaration sur l'honneur	**p.**	40
Documents illustrant la pratique professionnelle *(facultatif)*	**p.**	41
Annexes *(Si le RC le prévoit)*	**p.**	42

EXEMPLES DE PRATIQUE PROFESSIONNELLE

DOSSIER PROFESSIONNEL (DP)

Activité-type 1

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

Exemple n°1 ▸ **CP1 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 :

Dans le cadre de ce projet, j'ai participé au développement de la partie front-end d'une application web sécurisée, en utilisant Laravel pour le back-end et Inertia.js avec React pour le front-end.

Tâches réalisées :

Installation et configuration de l'environnement de développement :
J'ai utilisé le Starter Kit React officiel de Laravel, qui offre une base moderne et robuste pour le développement d'applications web. Ce kit repose sur React 19, TypeScript, Tailwind CSS, shadcn/ui et utilise Vite pour la compilation rapide du code front-end.
J'ai installé les dépendances nécessaires avec Composer pour Laravel, ainsi que Node.js et npm pour le front-end. Une base de données MySql a été mise en place pour le développement local.
L'environnement de développement a été configuré sous Visual Studio Code (VS Code), avec les extensions suivantes :

- DB Client pour interagir avec la base de données MySqlite,
- PHP Intelephense pour l'autocomplétion et l'analyse du code PHP,
- Prettier pour le formatage automatique du code,
- Tailwind CSS IntelliSense pour bénéficier de la complétion des classes utilitaires Tailwind.
  Le projet a été versionné via GitHub, avec initialisation du dépôt, gestion des branches et collaboration via pull requests.
- Pour la gestion des données, j'ai utilisé l'ORM Eloquent de Laravel, qui permet une interaction simple et fluide avec la base de données à travers des modèles.

Préparation de l'interface utilisateur :
À partir des maquettes fournies via Figma et FigJam, j'ai développé les interfaces en React avec JavaScript XML, en m'appuyant sur Inertia.js.
J'ai utilisé Tailwind CSS pour le design, et intégré des composants issus de la bibliothèque shadcn/ui, pour garantir une cohérence visuelle, une bonne accessibilité et un rendu responsive.

Gestion des routes :
J'ai configuré la navigation côté client avec Inertia.js, en coordination avec les routes Laravel, afin de permettre une navigation fluide sans rechargement de page tout en conservant la logique serveur de Laravel.

2. Précisez les moyens utilisés :

Pour réaliser le développement front-end de l'application, j'ai utilisé les moyens techniques suivants :

Environnement de développement :

- **Laravel** pour la partie serveur, gestion des routes, sécurité.
- **Inertia.js** pour faire le lien entre le back-end Laravel et le front-end.
- **React** pour le développement des composants front-end dynamiques et interactifs.

Gestion du code source :

- **Git** pour le contrôle de version.
- **GitHub** comme plateforme d'hébergement du code, permettant la collaboration, le suivi des modifications via les branches, les commits et les pull requests.

3. Avec qui avez-vous travaillé ?

J'ai travaillé sur ce projet seule.

4. Contexte

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

Chantier, atelier, service ▸ *Cliquez ici pour taper du texte.*

Période d'exercice ▸ Du : *Cliquez ici* au : *Cliquez ici*

5. Informations complémentaires *(facultatif)*

Activité-type 1

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

Exemple n°1 ▸ **CP2 Maquetter des interfaces utilisateur web ou web mobile**

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

Étant seule responsable du projet, j'ai commencé par rédiger les user stories, dans le but de définir précisément les besoins des utilisateurs finaux et structurer les fonctionnalités principales de l'application.

Ces user stories, rédigées à la fois pour les utilisateurs et les administrateurs, m'ont permis de poser les bases d'une interface claire, fluide et centrée sur l'expérience utilisateur.

User Stories – Utilisateur

En tant que	Je veux	Afin de
Utilisateur non inscrit	Créer un compte	Accéder à l'application et utiliser ses fonctionnalités
Utilisateur	Me connecter à mon compte	Accéder à mon espace personnel et mes données
Utilisateur	Consulter la page d'accueil	Découvrir les produits proposés
Utilisateur	Voir la liste des produits	Choisir ce qui m'intéresse avant de passer à l'action
Utilisateur	Ajouter un produit au panier	Finaliser un achat plus tard
Utilisateur	Naviguer facilement entre les pages	Avoir une expérience fluide et intuitive

User Stories – Administrateur

En tant que	Je veux	Afin de
Administrateur	Me connecter à l'espace admin	Gérer les utilisateurs et les produits
Administrateur	Accéder à la liste des utilisateurs et de leurs achats	Pouvoir consulter et gérer les comptes, et leurs commandes
Administrateur	Modifier ou supprimer un utilisateur	Gérer les droits ou résoudre un problème
Administrateur	Ajouter un nouveau produit	Mettre à jour le contenu visible par les utilisateurs
Administrateur	Modifier ou supprimer un produit	Garder l'offre à jour et pertinente
Administrateur	Accéder à des statistiques (tableau de bord)	Suivre l'activité et ajuster les décisions

Maintenant que les user stories sont bien établies et que les objectifs du projet sont clairs, il est temps de passer à la visualisation concrète du futur site. Pour cela, j'ai créé une série de wireframes. Contrairement aux maquettes, qui représentent une version finale et stylisée du design, les wireframes sont des esquisses simplifiées qui servent à organiser les éléments clés de l'interface utilisateur. Ils permettent de structurer les zones et les composants sans se focaliser sur l'aspect esthétique.

Dans le cadre de ce projet, j'ai conçu deux versions de wireframes pour chaque page : une adaptée aux appareils mobiles et une autre optimisée pour les écrans desktop.

Voici un aperçu des wireframes d'une des pages de mon site (vue d'ensemble) :

Wireframes de la page "Our Collection"

Ces wireframes présentent les versions mobile et desktop de la page "Our Collection" du futur site Skinn. Ils mettent en avant l'organisation des produits et la navigation, avec une interface adaptée à chaque support pour une expérience utilisateur optimale.

Création d'une bibliothèque de composants graphiques:

Pour assurer la cohérence visuelle de l'ensemble du projet et faciliter les évolutions futures, j'ai créé une page dédiée aux composants graphiques (boutons, champs de formulaire, menus, icônes, etc.)

2. Précisez les moyens utilisés :

Pour réaliser ces wireframes j'ai utilisé figma qui m'a permis de les réaliser facilement.

3. Avec qui avez-vous travaillé ?

J'ai travaillé sur ce projet seule.

4. Contexte

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

Chantier, atelier, service ▸ *Projet personnel réalisé en cours de formation.*

Période d'exercice ▸ Du : *Cliquez ici* au : *Cliquez ici*

5. Informations complémentaires *(facultatif)*

Le repo de ce projet est accessible à l'adresse suivante:
La maquette figma est accessible:

DOSSIER PROFESSIONNEL (DP)

Activité-type 1

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

Exemple n°1 ▸ **CP3 Réaliser des interfaces utilisateur statiques web ou web mobile**

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

Pour cette compétence "Réaliser des interfaces utilisateur statiques web ou web mobile", j'ai développé la page d'accueil de mon projet en utilisant une approche modulaire et composable. Voici les étapes clés :

a) Structure modulaire avec des composants réutilisables
J'ai découpé l'interface en composants statiques indépendants, chacun ayant un rôle spécifique :
-HeroSection : Bannière d'accueil avec titre, sous-titre et bouton call-to-action.
- SkinnSection/ IngredientImage : Sections visuelles mettant en avant des produits ou ingrédients.
- PostGrid : Grille de cartes (articles/fiches produits) avec image + texte.
- TextHero : Bloc de contenu textuel enrichi (ex : valeurs de la marque).
- PartnersLogo : Liste responsive de logos de partenaires.
- CategoryGrid : Navigation visuelle par catégories sous forme de grille ou carrousel.

Ces composants sont assemblés dans GuestLayout, qui gère la structure globale (header, footer, styles de base) pour une cohérence sur toutes les pages.

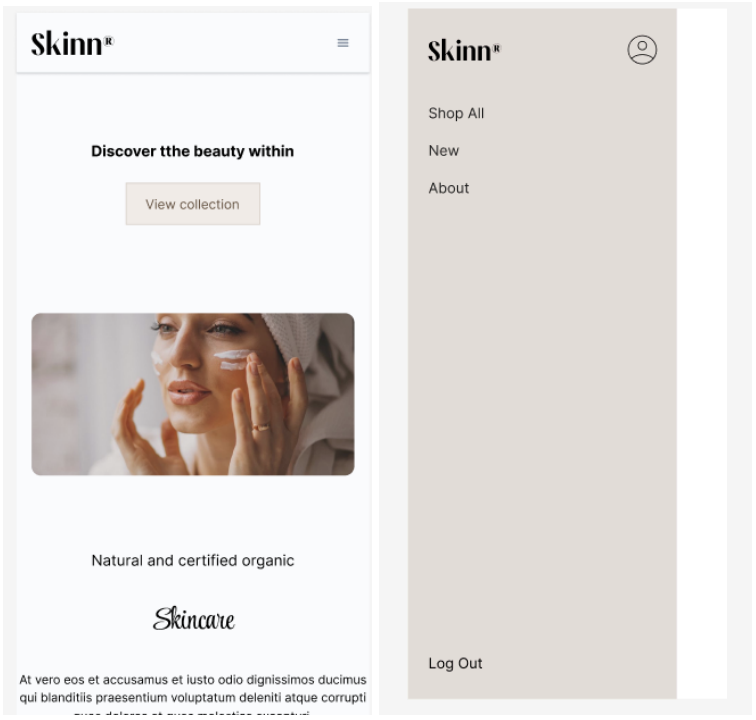
```
import CategoryGrid from '@/components/category-grid';
import IngredientImage from '@/components/ingredient-
image';HeroSection from '@/components/hero-section';
import PartnersLogo from '@/components/partners-logo';
import PostGrid from '@/components/post-grid';
import SkinnSection from '@/components/skinn-section';
import TextHero from '@/components/text-hero';
import GuestLayout from '@/layouts/guest-layout';

export default function Welcome() {
    return (
        <GuestLayout>
            <HeroSection />
            <SkinnSection />
            <IngredientImage />
            <PostGrid />
            <TextHero />
            <PartnersLogo />
            <CategoryGrid />
        </GuestLayout>
    );
}
```

b) Technologies et méthodologie

- Tailwind CSS : Utilisation des utilitaires pour styliser rapidement les composants tout en gardant le code maintenable.
- Approche "Mobile First : Les breakpoints (`sm:`, `md:`, `lg:`) permettent d'adapter chaque composant à tous les écrans.
  Example:
  - Le TextHero réduit sa taille de police et son padding sur mobile.

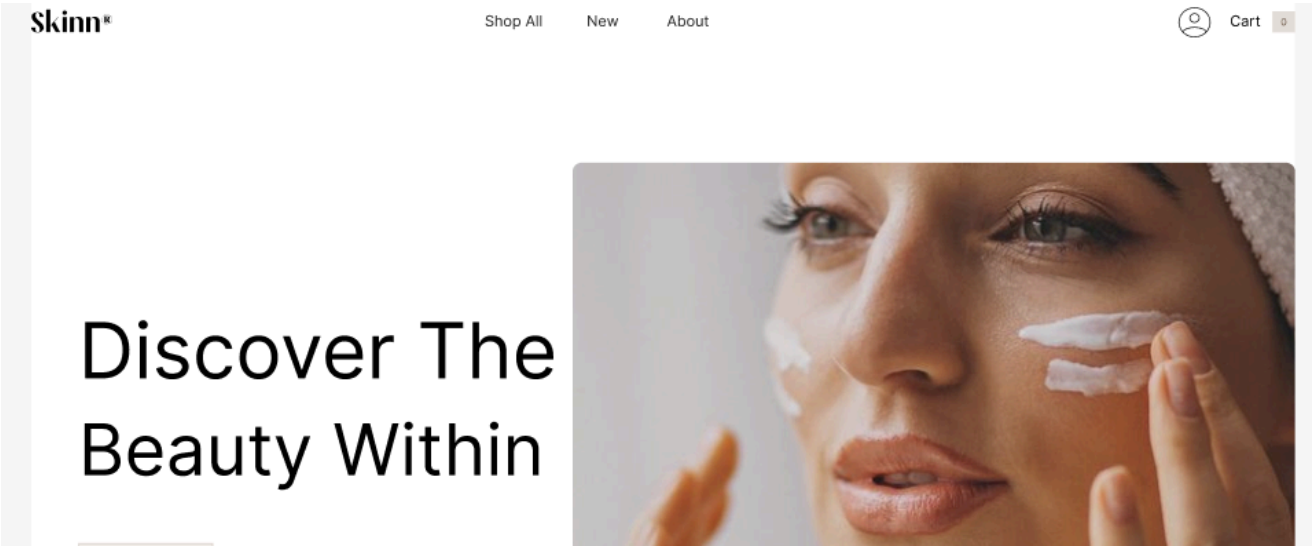
c) Bonnes pratiques appliquées
- Sémantique HTML : Balises appropriées (`<section>`, `<article>`) pour le SEO et l'accessibilité.
- Accessibilité: Contraste des couleurs vérifié, attributs `alt` pour les images, labels pour les interactions.

d) Exemple concret : le menu responsive
- Sur mobile : Menu caché derrière un burger menu (économie d'espace), déployé au clic.



-

- Sur desktop : Navigation toujours visible, avec un hover sur les liens pour améliorer l'UX.



2. Pourquoi cette approche ?
- Maintenabilité: Un composant = un fichier, facile à mettre à jour ou réutiliser.
- Évolutivité : Les composants sont conçus pour accueillir du contenu dynamique (ex : `PostGrid` peut recevoir des données API futures).

Fonctionnement du Responsive avec Tailwind CSS dans le composant HeroSection

1. Mécanisme responsive avec Tailwind

Tailwind utilise des breakpoints préfixés (comme `md:`, `lg:`) pour adapter le style, similaires aux media queries CSS traditionnelles mais plus rapides à implémenter. Voici comment cela s'applique :

```
<div className="relative h-80 overflow-hidden bg-indigo-600 md:absolute md:left-0 md:h-full md:w-1/3
lg:w-1/2">
```

- Par défaut (mobile) :
    - `h-80` : Hauteur fixe (320px)
    - `overflow-hidden` : Cache les débordements

- bg-indigo-600 : Fond coloré (remplace l'image sur petits écrans)

- À partir de md: (768px) :
  - md:absolute : Positionnement absolu
  - md:left-0 : Alignement à gauche
  - md:h-full : Prend toute la hauteur parente
  - md:w-1/3 : Occupe 1/3 de la largeur
- À partir de lg: (1024px) :
  - lg:w-1/2 : La colonne image s'élargit à 50%

3. Gestion de l'image

```
<img className="size-full object-cover" src="..."
alt=""/>
```

- size-full : Remplit l'espace disponible (100% width/height)
- object-cover : Garantit que l'image couvre la zone sans déformation (recadrage intelligent)

Texte et bouton adaptatifs

```
<div className="pr-6 pl-6 md:ml-auto md:w-2/3 md:pl-16 lg:w-1/2 lg:pl-
24">
```

- Mobile :
  - Padding horizontal (px-6) pour éviter les bords collés
- Desktop :
  - md:ml-auto : Alignement à droite
  - md:w-2/3 → lg:w-1/2 : Largeur proportionnelle à l'écran

Padding gauche progressif (`pl-16` → `pl-24`) pour l'espacement

Grâce à cette approche modulaire et responsive avec Tailwind CSS, le composant HeroSection s'adapte parfaitement à tous les appareils, offrant une expérience utilisateur cohérente tout en respectant les principes de performance et d'accessibilité définis dans le projet.

2. Précisez les moyens utilisés :

Pour concevoir ce projet, j'ai travaillé avec Visual Studio Code comme éditeur de code principal, appréciant ses fonctionnalités étendues et ses extensions utiles pour le développement web.

Les tests d'affichage et de compatibilité ont été réalisés via Google Chrome et ses outils de développement intégrés, qui m'ont permis de :
- Vérifier le rendu responsive sur différentes tailles d'écran
- Débugger le CSS et le JavaScript en temps réel
- Optimiser les performances de la page

Afin de garantir un code propre et conforme aux standards, je me suis appuyé sur la documentation MDN (Mozilla Developer Network) pour :
- Maîtriser la sémantique HTML (balises appropriées, structure accessible)
- Comprendre en détail les propriétés CSS et leurs comportements

Pour accélérer la mise en page tout en assurant un design cohérent, j'ai choisi d'utiliser Tailwind CSS(à la place de Bootstrap dans votre exemple), en consultant régulièrement sa documentation officielle pour :
- Appliquer les utilitaires responsive (breakpoints)
- Personnaliser la configuration (couleurs, espacements)
- Implémenter des composants réutilisables

Ressources clés :
- MDN :
https://developer.mozilla.org/fr/docs/Web/HTML
- Tailwind CSS : https://tailwindcss.com/docs/installation

Stack Overflow :
https://stackoverflow.com/

GitHub Discussions :
https://github.com/tailwindlabs/tailwindcss/discussions

Laracasts Forum :

https://laracasts.com/discuss

Cette méthodologie m'a permis de développer efficacement tout en maintenant un code robuste et maintenable.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seule sur ce projet.

4. Contexte

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

Chantier, atelier, service ▸ *Projet personnel réalisé en cours de formation..*

Période d'exercice ▸ Du : *Cliquez ici* au : *Cliquez ici*

5. Informations complémentaires *(facultatif)*

Activité-type 1

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

Exemple n°1 – **CP4 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 :

1. Front-end sécurisé

Bien que Laravel gère nativement des aspects critiques (CSRF, validation back-end), j'ai renforcé la sécurité côté front-end via :

- Protection CSRF : Tokens automatiquement inclus dans les formulaires via Inertia (`<Head>`).
- Validation côté client : Messages d'erreur dynamiques (ex: vérification des champs email/mot de passe avant soumission).
- Gestion des accès :
  - Redirection des utilisateurs non authentifiés (via middleware Laravel).
  - Masquage des éléments UI sensibles (ex: bouton "Admin Dashboard" pour les rôles non autorisés).
- HTTPS : Mise en place en production pour chiffrer les données.

2. Interfaces dynamiques

J'ai implémenté des composants interactifs avec Inertia et React, notamment :

a. Tableau de bord administrateur

J'ai conçu un tableau de bord interactif pour la gestion des commandes, avec des fonctionnalités dynamiques implémentées via Inertia.js et React.

- Transmission des données :
  - Les données sont passées depuis les contrôleurs Laravel aux composants React via `Inertia::render()`.

```php
return Inertia::render('admin/dashboard', [
    'orders' => $orders,
    'stats' => $stats,
    'sales2025' => $completeMonthlySales, // Graphique des ventes mensuelles
]);
```

Graphiques dynamiques (LineChart) :

- Affichage des ventes mensuelles pour 2025 avec Chart.js.
- Les données sont structurées sous forme de tableau [janvier, février, ..., décembre] pour un rendu simplifié côté front.
- Exemple de requête SQL (SQLite) :

```php
$monthlySales = DB::table('orders')
    ->join('order_product', 'orders.id', '=', 'order_product.order_id')
    ->join('products', 'order_product.product_id', '=', 'products.id')
    ->selectRaw("CAST(strftime('%m', orders.created_at) AS INTEGER) as month,
SUM(order_product.quantity * products.sales_price) as total_sales")
    ->whereYear('orders.created_at', 2025)
    ->groupBy('month')
    ->orderBy('month')
    ->pluck('total_sales', 'month');
```

- Tableau des commandes (OrderTable) :
    - Affichage paginé/triable des commandes avec statuts (pending, paid, etc.).
    - Possibilité de filtrer par statut directement côté client (sans rechargement).

b. Espace client

J'ai développé une interface dédiée aux clients avec :

- Liste des commandes personnelles :
    - Chargement via Order::where('client_id', $user->id) pour garantir l'isolation des données.

- Protection des routes avec le middleware `auth` de Laravel.

Détail d'une commande :

- Vérification d'accès pour éviter qu'un utilisateur ne voie les commandes d'un autre :

```php
$order = Order::where('id', $id)
    ->where('client_id', $user->id)->firstOrFail();
```

Conditions de développement

- Stack technique :
  - Laravel (backend) + Inertia.js (bridge frontend) + React (composants).
  - Tailwind CSS pour le styling responsive.
- Gestion d'état :
  - Pas besoin de Redux/Context API : les props sont injectées directement par Inertia.
- Sécurité :
  - Protection des routes avec `middleware(['auth'])`.

2. Précisez les moyens utilisés :

J'ai utilisé Laravel 12 et Inertia.js (React) , avec Tailwind CSS pour l'UI. Les données étaient gérées via Eloquent et validées côté serveur. Git a servi au versionnage, et les middlewares Laravel ont protégé les accès.

3. Avec qui avez-vous travaillé ?

J'ai travaillé sur ce projet seule.

4. Contexte

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

Chantier, atelier, service ▸ *Projet personnel réalisé en cours de formation..*

Période d'exercice ▸ Du : *Cliquez ici* au : *Cliquez ici*

5. Informations complémentaires *(facultatif)*

DOSSIER PROFESSIONNEL (DP)

Activité-type 2

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

Exemple n° 1 ▸ **CP5 Mettre en place une base de données relationnelle**

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

Dans le cadre de ce projet, j'ai participé au développement du back-end d'une application e-commerce sécurisée, développée avec le framework Laravel et la bibliothèque Inertia.js pour l'intégration avec le front-end React.

Tâches réalisées :

Mise en place de la structure back-end :

J'ai utilisé Laravel pour gérer toute la logique côté serveur. Le projet s'appuie sur le Starter Kit React officiel de Laravel, incluant l'intégration d'Inertia.js, Tailwind CSS, TypeScript, React, et la bibliothèque de composants shadcn/ui.

J'ai veillé à respecter des conventions de nommage claires et cohérentes pour toutes les entités et leurs attributs, comme user_id pour identifier un utilisateur ou product_name pour nommer un produit. Cela facilite la lecture, la maintenance et la compréhension du code.

Pour faciliter la gestion et l'interrogation de la base de données SQLite durant le développement, j'ai utilisé l'extension DB Client dans Visual Studio Code, qui permet de visualiser les tables, exécuter des requêtes SQL, et vérifier facilement le contenu des données sans quitter l'éditeur.

Pour l'architecture REST (Representational State Transfer), j'ai créé plusieurs contrôleurs

Dont ProductsController pour les produits.

```
    public function index()
    {
        $products = Product::with('category')-
>get(); $categories = Category::all();

        return Inertia::render('admin/products', [
            'products' => $products,
            'categories' => $categories,
        ]);
    }
```

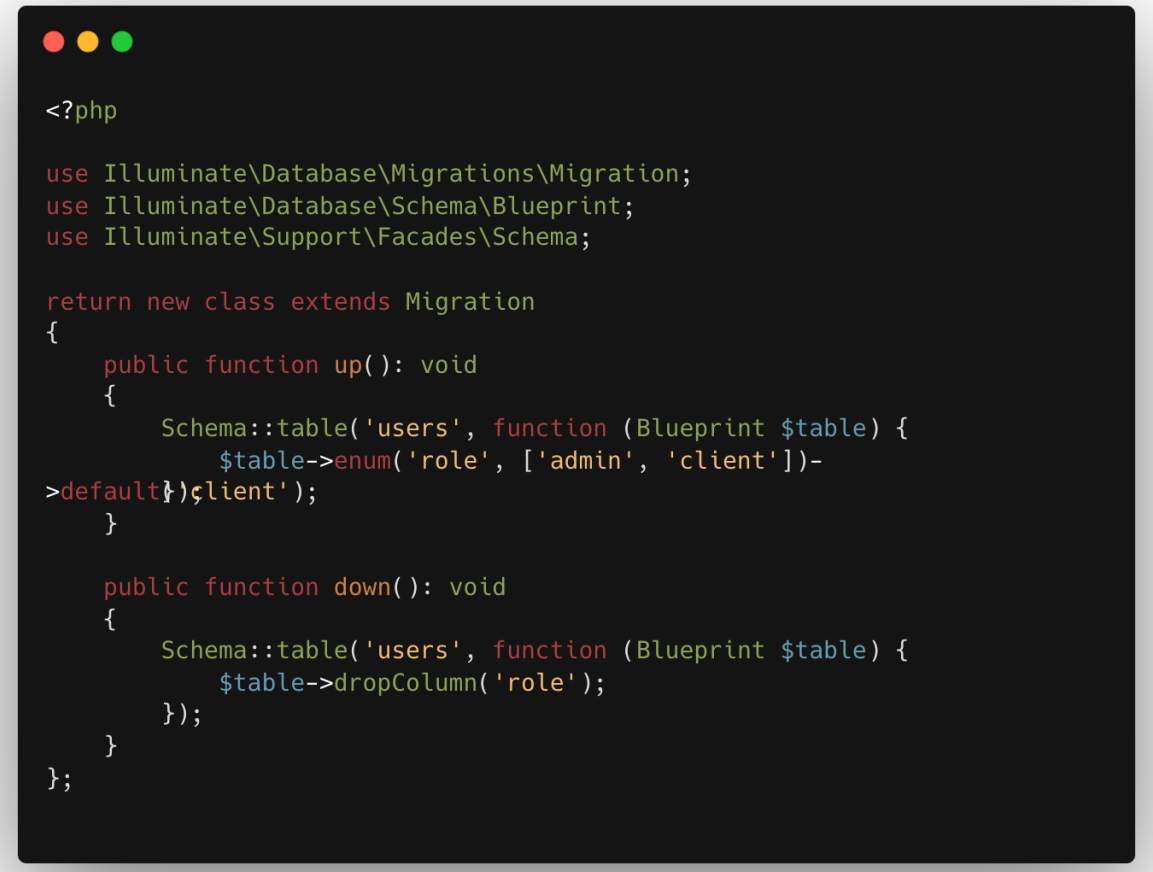
Sécurité et rôles :

L'authentification est fournie par le Starter Kit Laravel. J'ai ajouté une couche de sécurité supplémentaire en utilisant un middleware personnalisé pour restreindre certaines routes aux utilisateurs ayant le rôle admin.

J'ai mis en place un middleware permettant de distinguer deux rôles utilisateur : admin et client, via l'ajout d'un champ **role** dans la table **users**, géré par une migration.

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;

class RoleMiddleware
{

    public function handle(Request $request, Closure $next, $role): Response
    {
        // Check if the current authenticated user has a different role than
$role    if ($request->user()?->role !== $role) {
            abort(403, 'Unauthorized.');
            // abort 403 means it throws an error
        }

        return $next($request);
        // By default, you MUST need this line
    }
}
```

Mise en place de la base de données :
Le système de gestion de base de données utilisé est SQLite pour l'environnement de développement. J'ai mis en place des migrations Laravel afin de créer et modifier les tables. Par exemple, la table users a été modifiée pour intégrer un champ role de type enum, avec les valeurs admin ou client.



```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::table('users', function (Blueprint $table) {
            $table->enum('role', ['admin', 'client'])->default('client');
        });
    }

    public function down(): void
    {
        Schema::table('users', function (Blueprint $table) {
            $table->dropColumn('role');
        });
    }
};
```

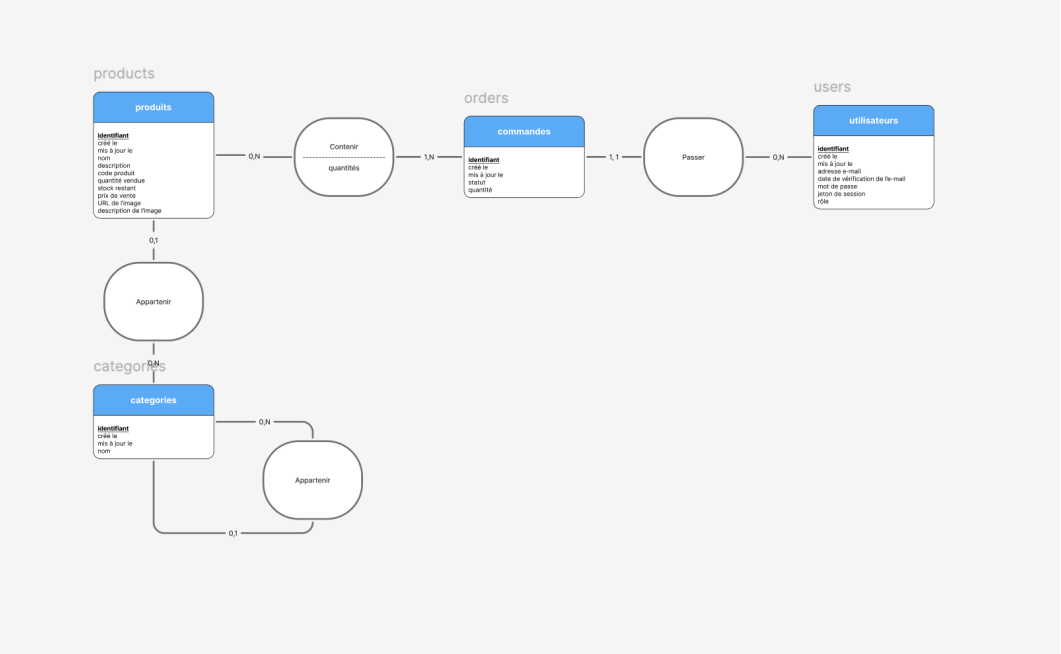
Utilisation d'Eloquent ORM :
L'application utilise Eloquent, l'ORM intégré à Laravel, pour interagir avec la base de données. J'ai défini des modèles et des relations, notamment dans le modèle Category qui gère une hiérarchie de catégories via des relations parent et children. Les commandes (Order) sont liées à des produits avec une relation many-to-many, enrichie d'un champ quantity.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    protected $fillable = ['name', 'parent_id'];

    public function products()
    {
        return $this->hasMany(Product::class);
    }

    public function parent()
    {
        return $this->belongsTo(Category::class,
'parent_id');
    }


    public function children()
    {
        return $this->hasMany(Category::class, 'parent_id');
    }
}
```

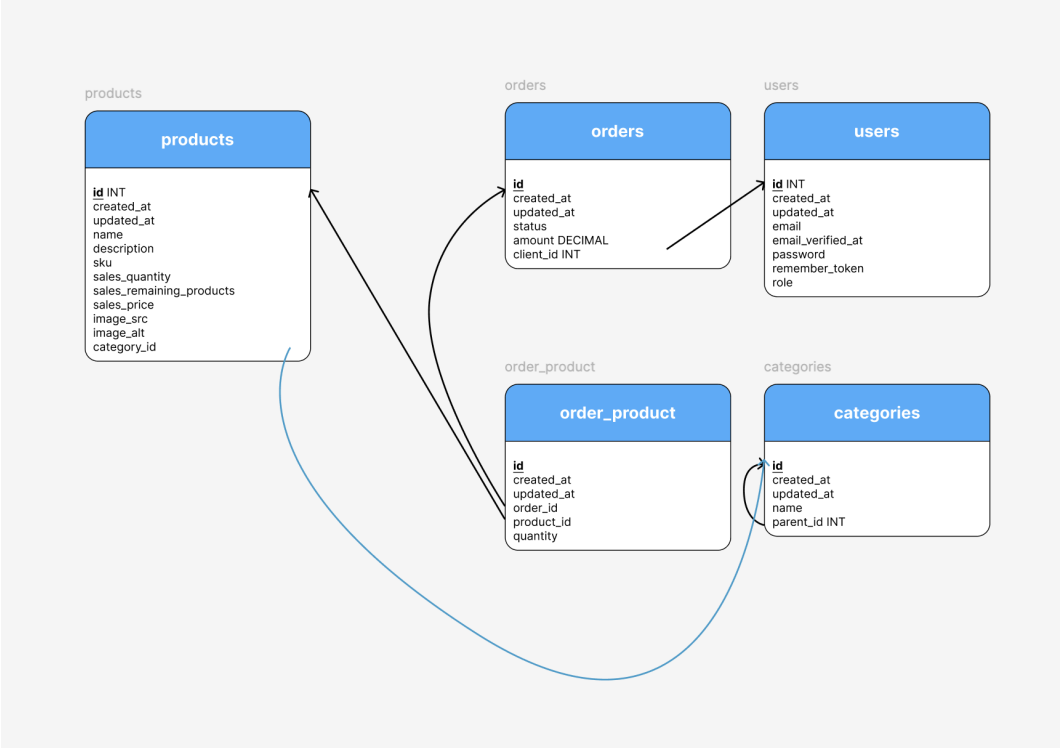
Génération de données de test :
J'ai mis en place des seeders pour insérer des données fictives dans la base. Par exemple, une commande a été créée via un seeder et associée à plusieurs produits à l'aide de la méthode attach() avec des quantités spécifiques.

```php
<?php

namespace Database\Seeders;

use App\Models\Order;
use App\Models\Product;
use App\Models\User;
// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    public function run(): void
    {
        $order = Order::create(['client_id' => 1, 'status' =>
'paid']);
        $product1 = Product::get(1);
        $order->products()->attach([
            1 => ['quantity' => 49],
            5 => ['quantity' => 10],
        ]);
    }
}
```

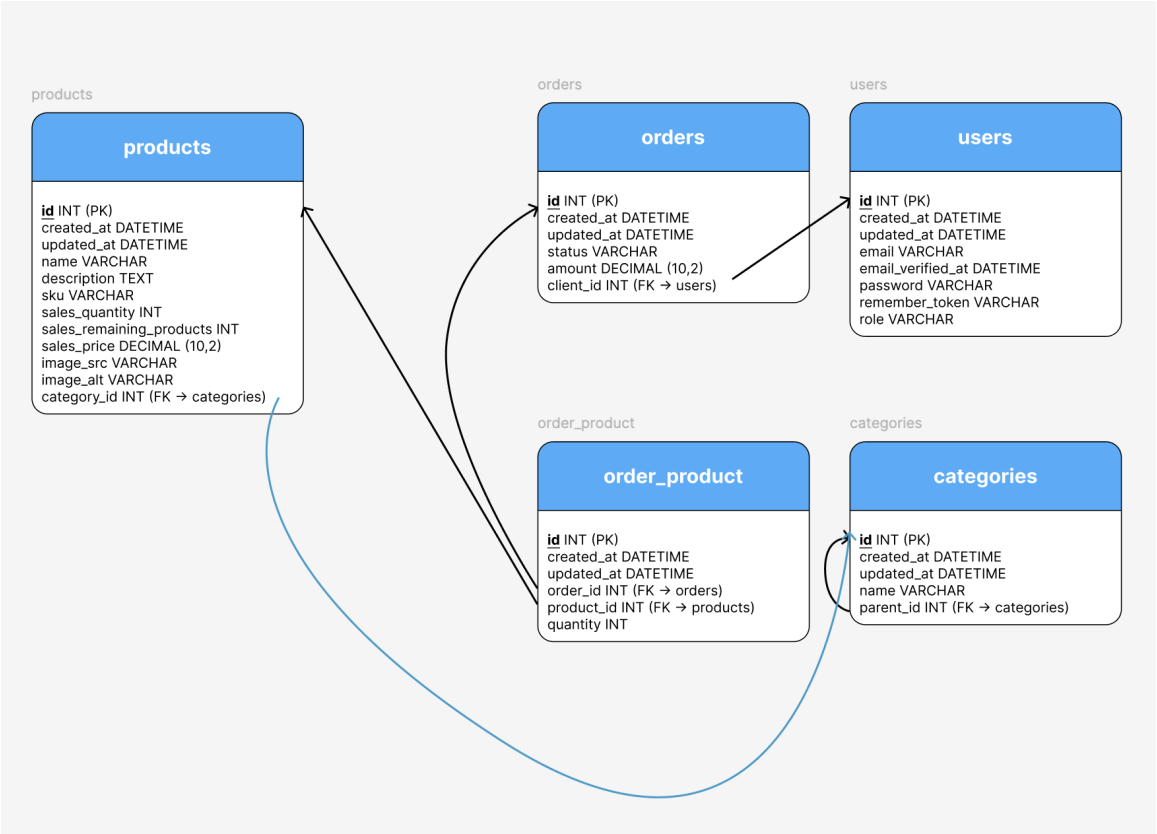
Modélisation de la base de données :
En amont du développement, j'ai conçu un MCD (Modèle Conceptuel de Données) pour définir les entités principales de l'application : utilisateurs, produits, commandes, catégories, etc.



Ce MCD a ensuite été transformé en MLD (Modèle Logique de Données), permettant de structurer précisément les relations entre les entités (clé primaire, clé étrangère, cardinalité…). Cela m'a permis de générer des migrations Laravel cohérentes avec la logique métier.

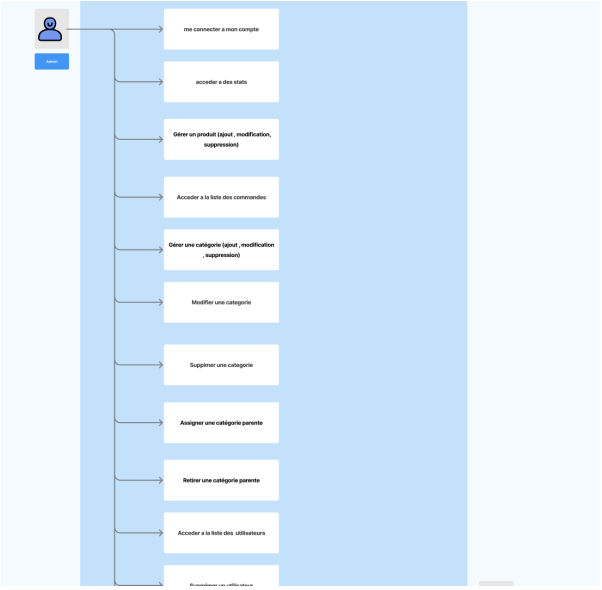


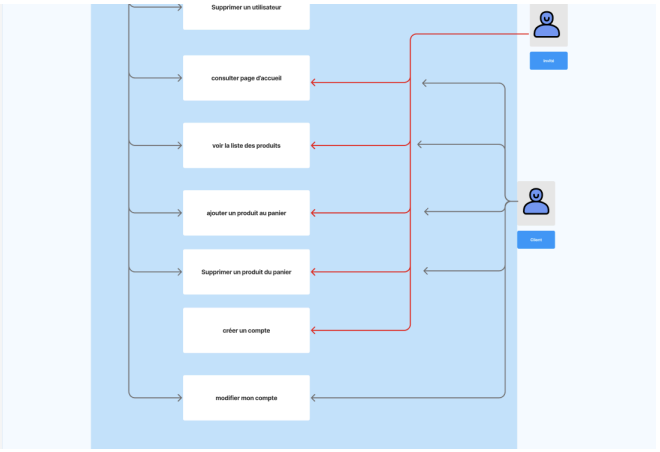
Ensuite, j'ai transformé mon MLD (Modèle Logique de Données) en MPD (Modèle Physique de Données)



Dans le cadre de ce projet e-commerce, j'ai élaboré plusieurs cas d'utilisation pour définir précisément les interactions entre les utilisateurs et le système. Ces cas d'usage décrivent les scénarios principaux,

comme la création d'une commande par un client, la gestion des produits par un administrateur, ou encore la consultation de l'historique des commandes.





2. Précisez les moyens utilisés :

- **Laravel Framework** : utilisé pour développer la partie back-end sécurisée, gérer les routes, contrôleurs, authentification, et sécuriser l'accès via middleware.
- **Eloquent ORM** : pour manipuler la base de données relationnelle de manière simple, avec définition claire des relations entre entités (one-to-many, many-to-many).
- **SQLite** : système de gestion de base de données relationnelle utilisé en environnement de développement, facile à configurer et léger.
- **Migrations Laravel** : pour créer et modifier les structures de tables dans la base de données, versionner ces modifications et garantir la cohérence avec le modèle de données.
- **Seeders Laravel** : pour insérer des données fictives, facilitant ainsi les tests et le développement.
- **DB Client (extension VS Code)** : outil permettant d'interroger et visualiser facilement la base de données SQLite depuis l'éditeur, accélérant le débogage et la validation des données.
- **Middleware personnalisé** : pour gérer les droits d'accès des utilisateurs en fonction de leur rôle (admin ou client), renforçant la sécurité de l'application.
- **Conventions de nommage claires** : pour assurer la lisibilité, la cohérence et la maintenance facile des données dans la base relationnelle (ex. user_id, product_name).

- **Git et GitHub** : gestion du versioning, garantissant un suivi rigoureux des évolutions du back-end et de la base de données.

3. Avec qui avez-vous travaillé ?

J'ai travaillé sur ce projet seule.

4. Contexte

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

Chantier, atelier, service ▸ *Projet personnel réalisé en cours de formation.*

Période d'exercice ▸ Du : *Cliquez ici* au : *Cliquez ici*

5. Informations complémentaires *(facultatif)*

Activité-type 2

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

Exemple n° 1 ▸ **CP6 Développer des composants d'accès aux données SQL et NoSQL**

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

Dans ce projet, j'ai utilisé principalement SQLite comme base de données relationnelle pour le développement local, avec le framework Laravel, qui facilite l'accès aux données grâce à son ORM Eloquent. Plutôt que d'écrire des requêtes SQL brutes, j'ai exploité les fonctionnalités d'Eloquent pour insérer, lire, mettre à jour et supprimer des données de manière sécurisée, fluide et orientée objet.

Eloquent permet de manipuler les tables de la base comme des modèles PHP, ce qui rend le code plus lisible et maintenable. Grâce à ses méthodes intégrées, il est possible de gérer les relations entre tables, de chaîner les requêtes, et de bénéficier d'une protection automatique contre les injections SQL. Ce système m'a permis de développer rapidement les opérations CRUD pour les produits, tout en garantissant un bon niveau d'abstraction.

Pour peupler la base, j'ai mis en place des seeders, qui automatisent l'insertion de données de test dans les différentes tables. Cela m'a grandement facilité le développement et les tests, en recréant rapidement un environnement cohérent à chaque itération.

Concernant la gestion des erreurs, le Starter Kit Laravel + React fournit un système intégré de gestion des exceptions, notamment au niveau de l'authentification, avec des messages d'erreur clairs qui sont affichés dans le front-end via Inertia.js. Par exemple, lors du processus de login, les erreurs de saisie sont capturées et affichées dynamiquement dans le formulaire grâce à la gestion d'état dans React.

Sur le plan de la validation des données, le projet applique des contrôles côté front-end (via les types d'input HTML et la gestion des erreurs dans React) et côté back-end, en utilisant les mécanismes de validation de Laravel (comme FormRequest ou les règles validate() dans les contrôleurs). Cela garantit la cohérence, la sécurité et l'intégrité des données entrantes.

Enfin, pour l'accès et la manipulation directe des données SQLite, j'ai utilisé l'extension DB Client dans Visual Studio Code. Cet outil m'a permis de visualiser les tables, exécuter des requêtes SQL et vérifier les données en temps réel sans quitter mon environnement de développement.

2. Précisez les moyens utilisés :

Pour développer les composants d'accès aux données, j'ai utilisé les outils suivants :

- **Laravel** : Framework PHP qui intègre **Eloquent ORM**, facilitant la manipulation de données en orienté objet.
- **SQLite** : Base de données relationnelle légère, utilisée localement pour le développement et les tests.
- **DB Client** (extension Visual Studio Code) : Pour interroger directement la base SQLite, visualiser les tables, exécuter des requêtes SQL et suivre l'évolution des données.
- **Eloquent ORM** : Outil principal pour le CRUD, les relations entre entités, les filtres, et la protection contre les injections SQL.
- **Seeders Laravel** : Génération automatisée de données de test pour accélérer le développement.
- **React + Inertia.js** : Pour la gestion du front-end, l'affichage des erreurs et la validation des formulaires.
- **Validation Laravel** : Utilisation des règles de validation côté back-end pour contrôler les données envoyées depuis les formulaires.

3. Avec qui avez-vous travaillé ?

J'ai travaillé sur ce projet seule.

4. Contexte

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

Chantier, atelier, service ▸ *Projet personnel réalisé en cours de formation..*

Période d'exercice ▸ Du : *Cliquez ici* au : *Cliquez ici*

5. Informations complémentaires *(facultatif)*

Activité-type 2

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

Exemple n° 1 - **CP7 Développer des composants métier coté serveur**

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

Dans le cadre de mon projet, j'ai développé plusieurs composants métier côté serveur en utilisant le framework Laravel (PHP), qui s'appuie sur l'architecture MVC (Modèle-Vue-Contrôleur). Cette structure m'a permis de bien séparer la logique métier, l'accès aux données et la présentation côté front-end, ce qui rend l'application plus organisée et plus facile à maintenir.

Une partie importante de mon travail a consisté à réaliser du refactoring pour améliorer la qualité du code. J'ai supprimé les redondances, simplifié certaines fonctions trop longues ou complexes, et réorganisé les fichiers pour respecter les conventions de Laravel. Le but était d'avoir un code plus lisible, plus propre et plus performant.

Le projet utilise Inertia.js pour connecter le back-end Laravel au front-end React. Ce système permet de charger les données dynamiquement sans recharger complètement la page, tout en gardant une structure claire. Les données sont transmises depuis les contrôleurs Laravel vers les composants React sous forme de props, ce qui centralise la logique métier côté serveur.

J'ai également appliqué des pratiques de développement propres à Laravel, comme le Repository Pattern, pour séparer la logique métier de la couche d'accès aux données, ce qui facilite la réutilisation du code et améliore sa testabilité.

L'ensemble de ces choix techniques m'a permis de produire un code plus robuste, évolutif et conforme aux bonnes pratiques du développement côté serveur.

2. Précisez les moyens utilisés :

Pour développer les composants métier côté serveur, j'ai utilisé les outils et techniques suivants :

- **Laravel** (PHP) : framework back-end basé sur le modèle MVC, utilisé pour structurer la logique métier au sein des contrôleurs et des services.
- **Inertia.js** : utilisé comme passerelle entre Laravel et React, permettant de transmettre les données du back-end vers le front-end sans rechargement complet de la page.

- **React** : bibliothèque JavaScript côté client, qui reçoit les données depuis Laravel et les affiche dynamiquement via des composants.
- **Refactoring** : nettoyage du code, suppression des redondances, découpage en fonctions claires et réutilisables pour améliorer la qualité et la maintenabilité.
- **Repository Pattern** : utilisé pour séparer la logique métier de l'accès aux données, afin de garder un code plus modulaire et mieux organisé.
- **Contrôleurs Laravel** : cœur du traitement métier, ils gèrent les règles d'application, les appels aux modèles Eloquent, et la transmission des données vers React.
- **Middleware Laravel** : utilisé pour appliquer certaines règles globales (comme la gestion des permissions ou l'accès restreint à certaines routes).

3. Avec qui avez-vous travaillé ?

4. Contexte

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

Chantier, atelier, service ▸ *Projet personnel réalisé en cours de formation.*

Période d'exercice ▸ Du : *Cliquez ici* au : *Cliquez ici*

5. Informations complémentaires *(facultatif)*

Activité-type 2

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

Exemple n° 1 - **CP 8 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 déployé une application Laravel + Inertia.js (React) dans un environnement Docker, afin de m'affranchir des limitations imposées par des solutions comme Plesk.
Le déploiement s'est fait en local, dans un conteneur Docker simulant un environnement de production.
J'ai construit une image contenant PHP 8.3.25, Composer, Node.js et Nginx.
J'ai configuré les services nécessaires via docker-compose.yml (PHP, MySQL, Nginx) et j'ai adapté le projet à cet environnement :

- migration de la base vers MySQL (au lieu de SQLite en local),
- compilation du front avec Vite,
- gestion des sessions en base de données,
- configuration d'un serveur Nginx personnalisé.

J'ai aussi résolu des erreurs liées aux builds Vite, aux fichiers manquants, et aux ports déjà utilisés sur la machine hôte.

2. Précisez les moyens utilisés :

Docker & Docker Compose pour créer un environnement isolé et reproductible

PHP 8.2, Composer, Node.js, Nginx dans l'image Docker

Laravel 12 + Inertia.js (React) pour le projet

MySQL 8 pour la base de données

Terminal & VS Code pour la configuration, le debug et l'exécution

npm pour le build des fichiers frontend via Vite

php artisan migrate pour la gestion des tables et des sessions

Fichiers configurés : .env, Dockerfile, docker-compose.yml, default.conf (Nginx), web.php, app.blade.php

3. Avec qui avez-vous travaillé ?

J'ai travaillé sur ce projet seule.

4. Contexte

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

Chantier, atelier, service ▸ *Projet personnel réalisé en cours de formation.*

Période d'exercice ▸ Du : *Cliquez ici* au : *Cliquez ici*

5. Informations complémentaires *(facultatif)*

Activité-type 3 Cliquez ici pour entrer l'intitulé de l'activité

Exemple n° 1 ► *Cliquez ici pour entrer l'intitulé de l'exemple*

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

2. Précisez les moyens utilisés :

3. Avec qui avez-vous travaillé ?

4. Contexte

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

Chantier, atelier, service ► *Projet personnel réalisé en cours de formation..*

Période d'exercice ► Du : *Cliquez ici* au : *Cliquez ici*

5. Informations complémentaires *(facultatif)*

Titres, diplômes, CQP, attestations de formation

(facultatif)

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

Déclaration sur l'honneur

Je soussignée Yanina Razvetskaya ,

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

Fait à Cliquez ici pour taper du texte. le Cliquez ici pour choisir une date

pour faire valoir ce que de droit.

Signature :

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé
Cliquez ici pour taper du texte.

ANNEXES

(Si le RC le prévoit)