



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)

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.

DOSSIER PROFESSIONNEL^(DP)

► <http://travail-emploi.gouv.fr/titres-professionnels>

Sommaire

Exemples de pratique professionnelle

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

p. 5

- CP1 Installer et configurer son environnement de travail en fonction du projet web ou web mobile.
- CP2 Maquetter des interfaces utilisateur web ou web mobile
- CP3 Réaliser des interfaces utilisateur statiques web ou web mobile
- CP4 Développer la partie dynamique des interfaces utilisateur web ou web mobile

p. 5
p. 7
p. 16
p. 24

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

p. 24

- CP5 Mettre en place une base de données relationnelle
- CP6 Développer des composants d'accès aux données SQL et NoSQL
- CP7 Développer des composants métier côté serveur
- CP 8 Documenter le déploiement d'une application dynamique web ou web mobile

p. 28
p. 38
p. 41
p. 47

Titres, diplômes, CQP, attestations de formation (*facultatif*)

p. 56

Déclaration sur l'honneur

p. 57

Documents illustrant la pratique professionnelle (*facultatif*)

p. 58

Annexes (*Si le RC le prévoit*)

p. 59

DOSSIER PROFESSIONNEL ^(DP)

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 et utilise Vite pour la compilation rapide du code front-end.

J'ai installé les dépendances nécessaires avec Composer pour Laravel 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 MySQL
- 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 en m'appuyant sur Inertia.js.

J'ai utilisé Tailwind CSS pour le design 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 :

DOSSIER PROFESSIONNEL (DP)

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 ➔ *Projet réaliser en cours de formation*

Période d'exercice ➔ Du : *07/04/2025* au : *07/07/2025*

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL (DP)

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

DOSSIER PROFESSIONNEL (DP)

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

Chaque user story a donné lieu à une fonctionnalité concrète, développée et testée au cours du projet. Cette correspondance garantit la traçabilité entre les besoins exprimés et les résultats obtenus.

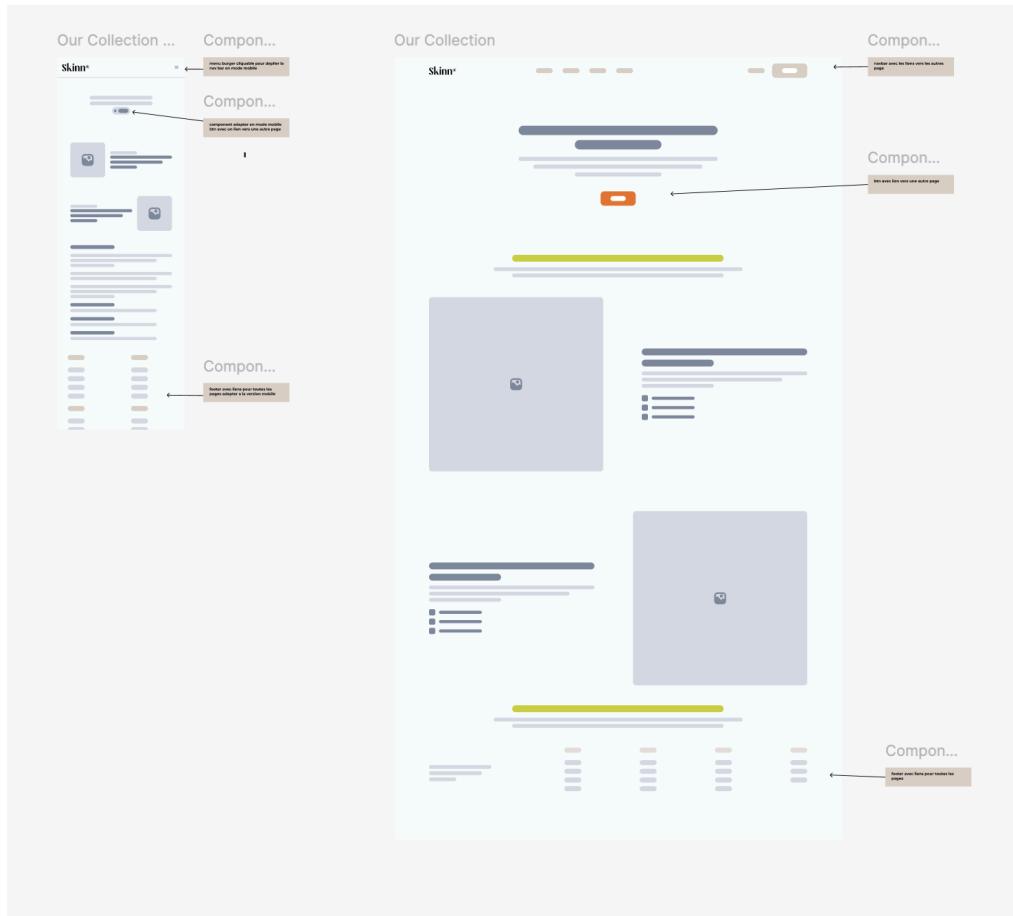
- Créer un compte → Formulaire d'inscription complet, avec validation côté client et serveur
- Se connecter à son espace personnel → Système d'authentification sécurisé, messages d'erreur clairs et redirection après connexion
- Consulter les produits disponibles → Page Our Collection affichant dynamiquement les produits enregistrés dans la base de données
- Ajouter un produit au panier → Bouton d'action interactif relié à la base, mise à jour instantanée du panier
- Gérer les utilisateurs et les produits (administrateur) → Tableau de bord d'administration avec fonctionnalités CRUD (création, modification, suppression)
- Consulter les statistiques (administrateur) → Page de suivi avec graphique dynamique (Chart.js) affichant les ventes et les commandes

DOSSIER PROFESSIONNEL (DP)

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) :



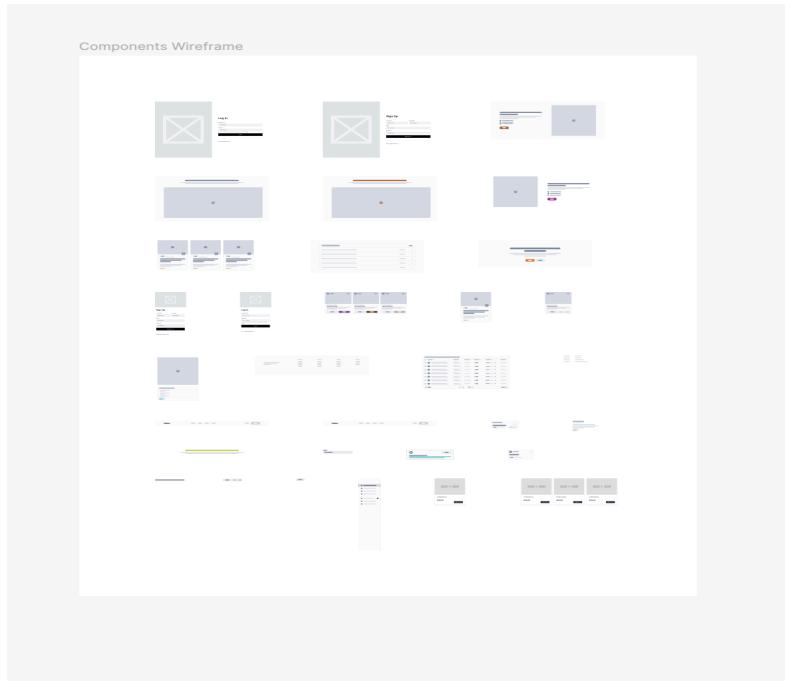
Wireframes de la page "Our Collection"

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.

DOSSIER PROFESSIONNEL (DP)

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.)



Les wireframes et maquettes ont été créés pour offrir une interface claire, lisible et cohérente sur tous les supports.

La palette de couleurs naturelles — beige, blanc cassé et marron doux — évoque la simplicité et la confiance, tout en mettant en valeur les produits.

Les contrastes ont été vérifiés pour assurer une bonne lisibilité, notamment sur mobile

DOSSIER PROFESSIONNEL (DP)

Colors

Color	HEX Code
Black	#000000
Beige 1	#F1EBE7
Beige 2	#DACEC6
White	#ffffff
Error	#DA1E28
Titles	#68513F
Success	#DA1E28
Cards	#68513F

La typographie, sobre et structurée, garantit une lecture fluide avec des titres bien hiérarchisés. Les composants (cartes, boutons, formulaires) conservent les mêmes styles pour maintenir une cohérence visuelle.

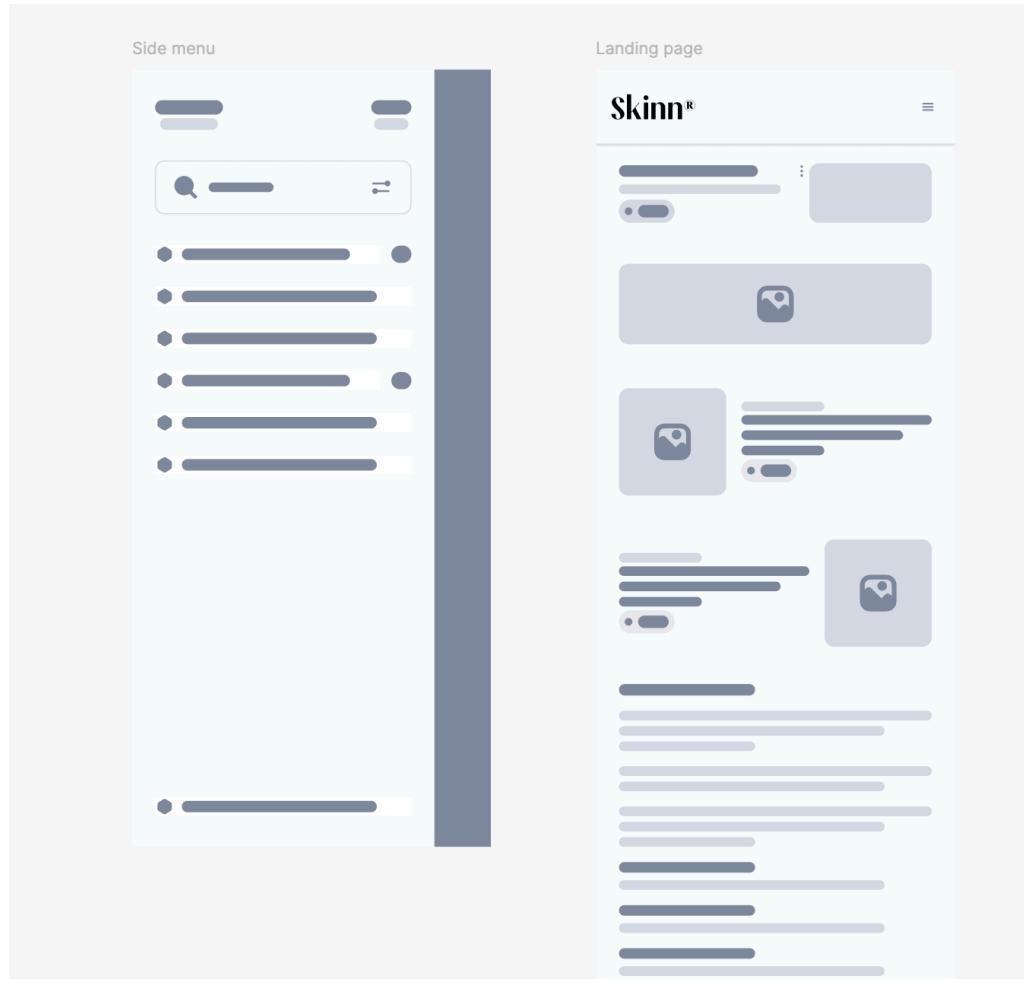
Typography

Font Family: Inter

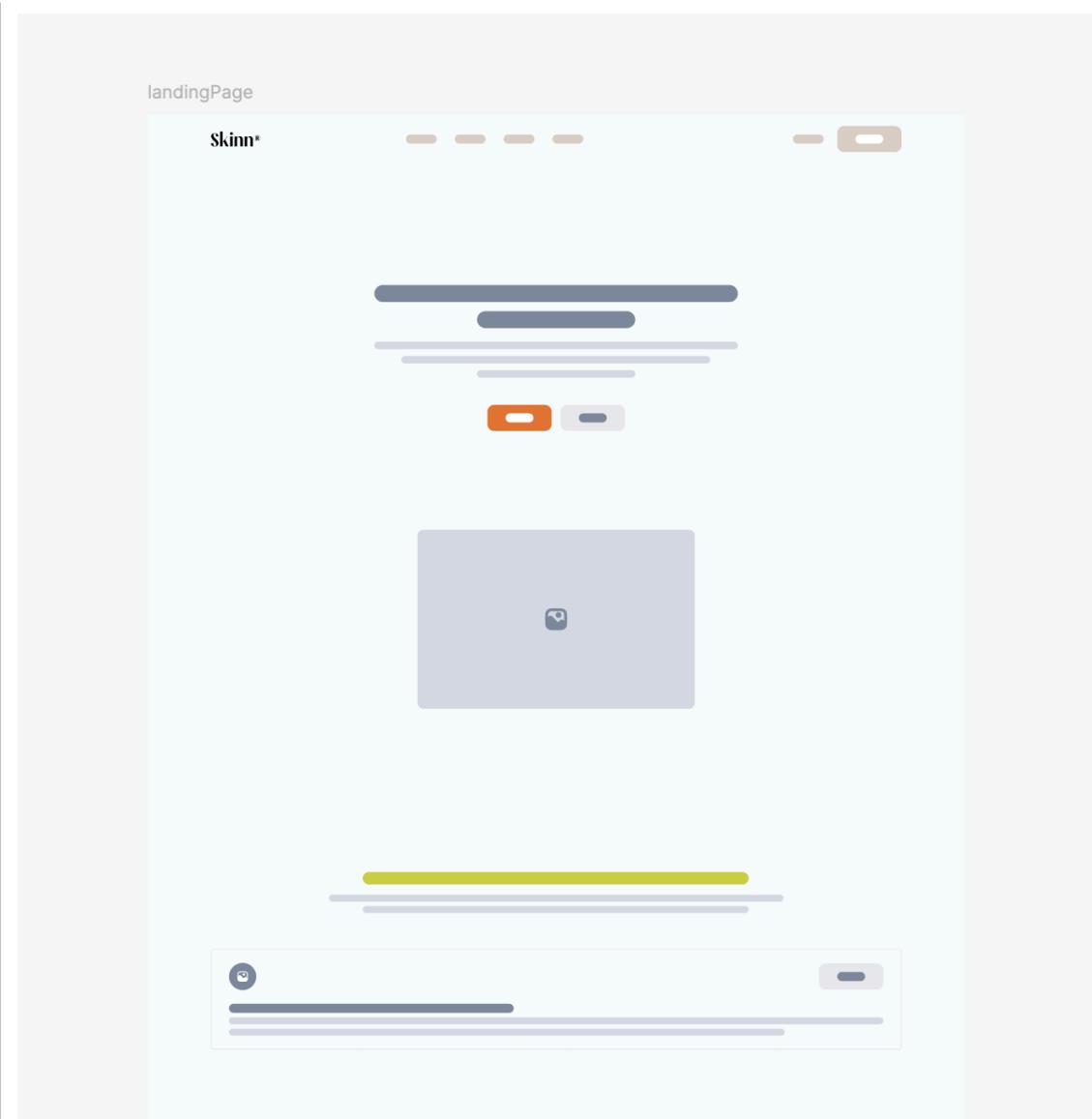
Type	Font Weight	Font Size
Heading 1	Bold / 700	88px
<i>Heading 2</i>	Bold / 700	32px
Heading 3	Bold / 700	65px
Subtitle	Medium / 500	14px
Button	Medium / 500	14px
Footer	Medium / 500	14px

DOSSIER PROFESSIONNEL (DP)

Sur mobile, la navigation reste simple et accessible ; sur desktop, la mise en page exploite l'espace sans alourdir la lecture.



DOSSIER PROFESSIONNEL (DP)



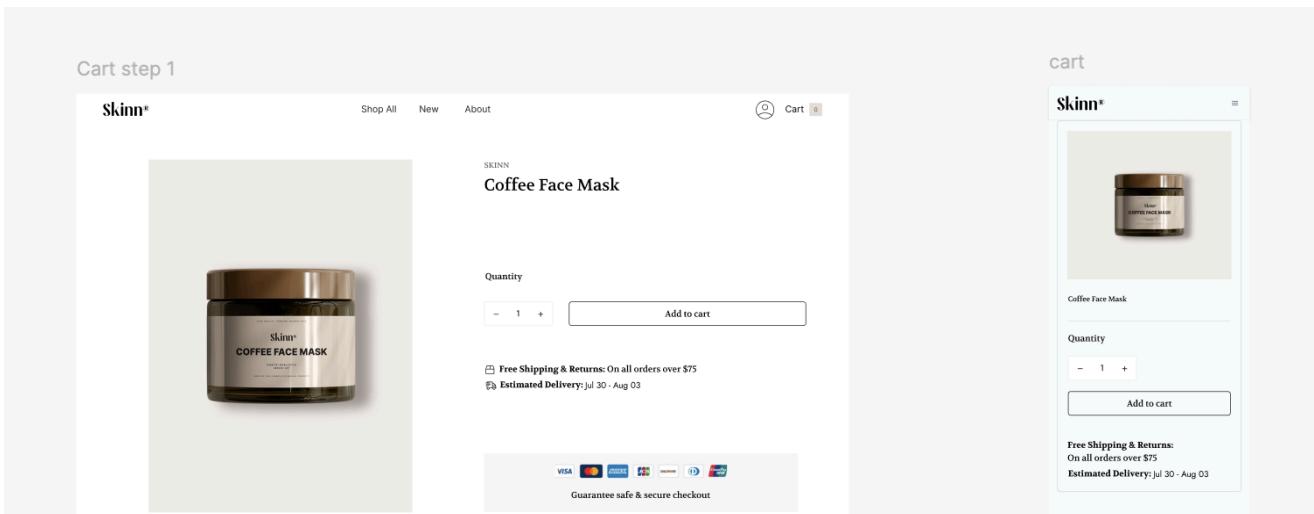
Les maquettes finales reprennent la charte graphique et les principes établis lors de la conception des wireframes, en intégrant les visuels, les couleurs et les typographies définitives.

Elles ont été déclinées en format mobile et desktop afin d'illustrer l'adaptabilité de l'interface et la cohérence du design sur tous les supports.

Sur mobile, les éléments sont empilés et centrés pour une navigation fluide, tandis que sur desktop, la mise en page s'élargit pour mieux valoriser les visuels et offrir plus d'espace aux contenus.

DOSSIER PROFESSIONNEL (DP)

Les deux versions conservent la même hiérarchie visuelle et les mêmes composants, garantissant une expérience homogène et accessible



Après la conception et la finalisation des maquettes, il a été nécessaire de vérifier le bon affichage du site sur différents formats d'écran.

Ces tests ont permis de s'assurer que l'interface restait lisible, fonctionnelle et cohérente aussi bien sur mobile, tablette que sur ordinateur.

Tests multi-supports

Support	Résolution	Résultat
Mobile	360×640	Interface claire, menu fluide
Tablette	768×1024	Grilles équilibrées
Bureau	≥1280	Mise en page stable

2. Précisez les moyens utilisés :

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

DOSSIER PROFESSIONNEL^(DP)

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 : **07/04/2025** au : **07/07/2025**

5. Informations complémentaires (*facultatif*)

Le repo de ce projet est accessible à l'adresse suivante: <https://github.com/YaninaRZ/final-project>

La maquette figma est accessible:

<https://www.figma.com/design/BdtEviT1itZRpe5tAfcPN1/Skinn?node-id=33519-25213&p=f&t=A3JStrz7oF9TfvvH-0>

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 :

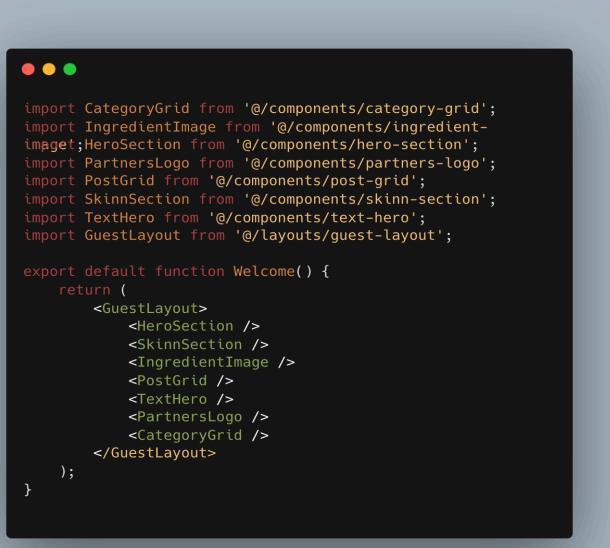
Pour cette compétence "Réaliser des interfaces utilisateur statiques web ou web mobile", j'ai développé la page d'accueil de mon projet en utilisant une approche modulaire et composable. Voici les étapes clés :

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';
import 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>
  );
}
```

DOSSIER PROFESSIONNEL (DP)

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:

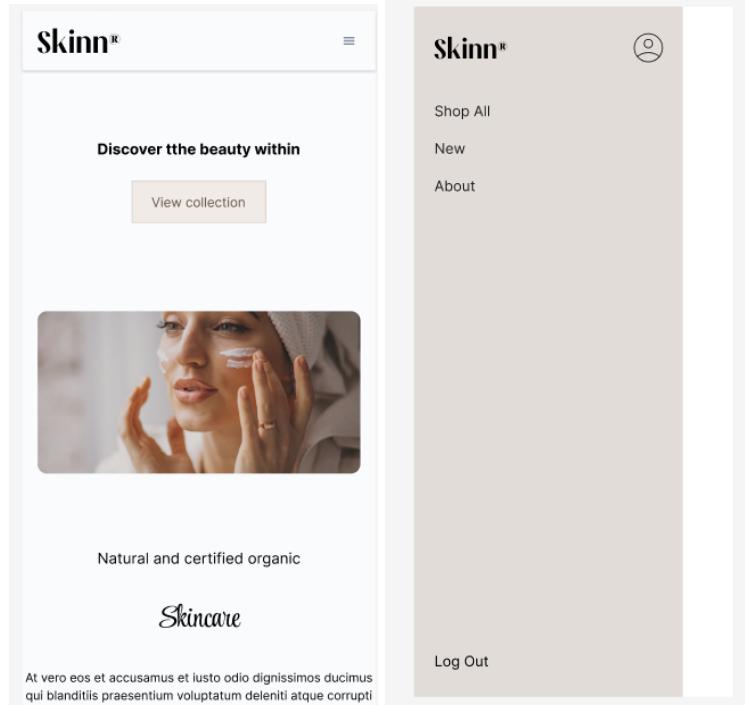
- Le TextHero réduit sa taille de police et son padding sur mobile

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

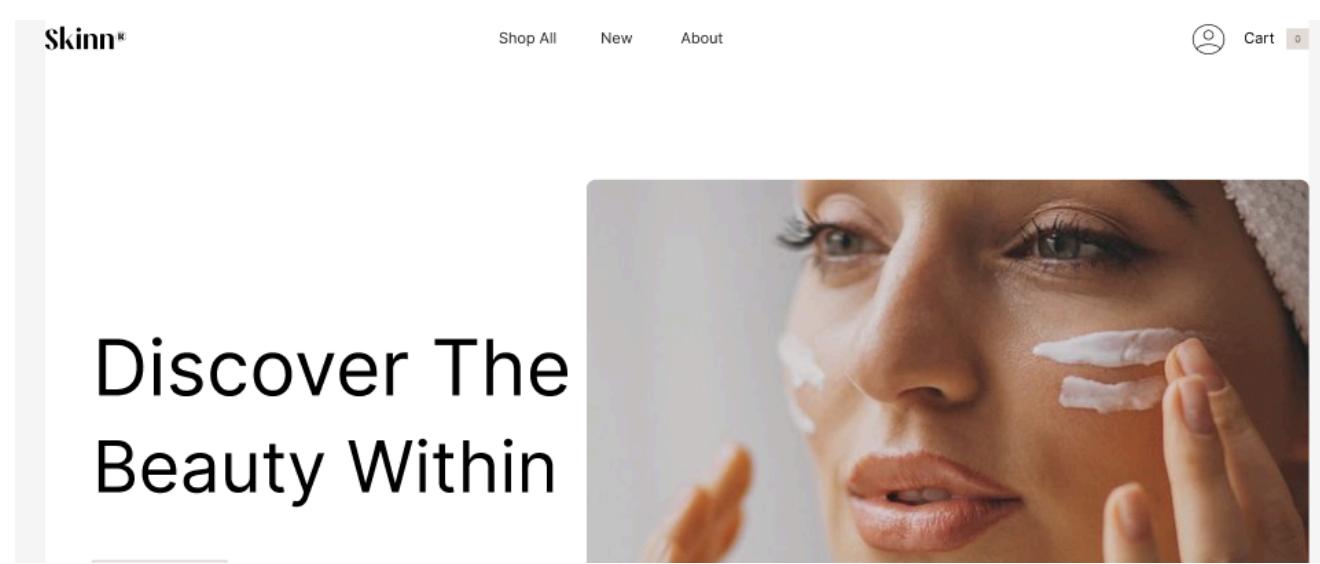
d) Exemple concret : le menu responsive

- 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

DOSSIER PROFESSIONNEL (DP)



The screenshot shows the homepage of the Skinn website. At the top, there's a navigation bar with the brand name "Skinn®" on the left, and "Shop All", "New", and "About" links in the center. On the far right, there's a user icon and a "Cart" button with a "0" next to it. Below the navigation, a large hero section features the text "Discover The Beauty Within" on the left and a close-up photograph of a woman applying cream to her face on the right.

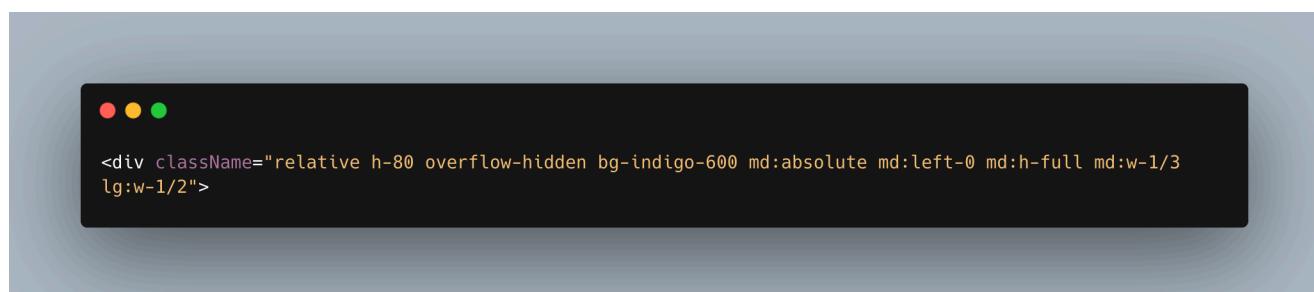
2. Pourquoi cette approche ?

- 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 :



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

DOSSIER PROFESSIONNEL (DP)

- `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



```

```

- `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

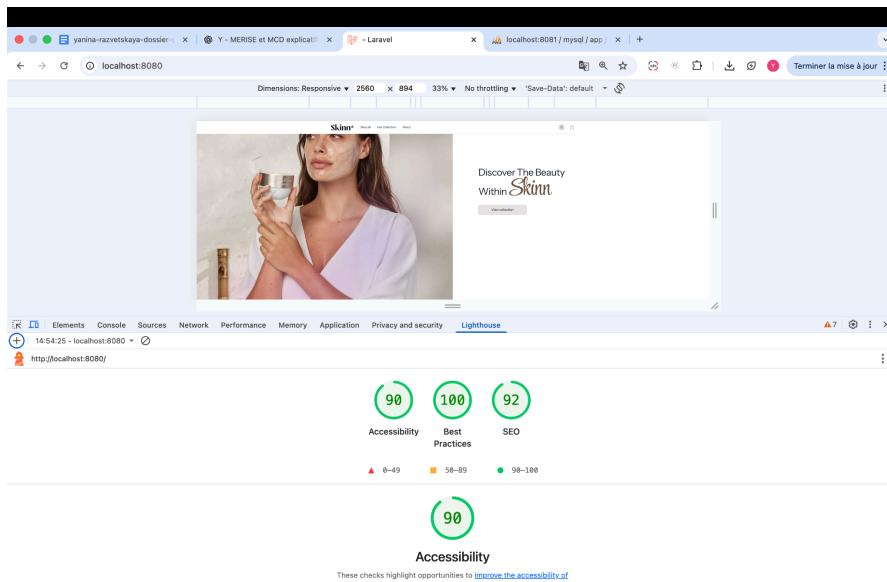
DOSSIER PROFESSIONNEL (DP)

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.

Accessibilité et conformité RGAA :

Afin de vérifier la conformité de l'application aux bonnes pratiques d'accessibilité, j'ai effectué un audit automatisé avec l'outil Lighthouse, intégré à Google Chrome DevTools.



Ces résultats démontrent un niveau d'accessibilité élevé, conforme aux recommandations RGAA et WCAG AA.

L'audit a été réitéré après intégration des correctifs, et les scores se sont maintenus au-dessus de 90 dans toutes les catégories.

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

DOSSIER PROFESSIONNEL (DP)

- 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, en consultant régulièrement sa documentation officielle pour :

- Appliquer les utilitaires responsive (breakpoints)
- Personnaliser la configuration (couleurs, espacements)
- Implémenter des composants réutilisables

Ressources clés :

- MDN : [\[https://developer.mozilla.org/fr/docs/Web/HTML\]](https://developer.mozilla.org/fr/docs/Web/HTML)
- Tailwind CSS : [\[https://tailwindcss.com/docs/installation\]](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.**

DOSSIER PROFESSIONNEL^(DP)

Période d'exercice ▶ Du : 07/04/2025 au : 07/07/2025

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL^(DP)

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().

DOSSIER PROFESSIONNEL (DP)

```
● ● ●  
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 :

```
● ● ●  
$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.

DOSSIER PROFESSIONNEL (DP)

- 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 :

```
$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']).

Afin de garantir la fiabilité de l'application et la cohérence entre les besoins et les fonctionnalités livrées, plusieurs tests ont été réalisés sur les parties front-end, back-end.

Tests front-end (interface utilisateur)

ID	Fonction testée	Étapes	Résultat attendu	Résultat obtenu
F01	Navigation principale	Cliquer sur chaque lien du menu	Accès correct aux pages, sans erreur	✓ Conforme
F02	Formulaire d'inscription	Remplir les champs valides et soumettre	Compte créé et message de confirmation	✓ Conforme
F03	Authentification	Saisir identifiants corrects	Redirection vers espace personnel	✓ Conforme
F04	Ajout au panier	Cliquer "Ajouter au panier"	Produit ajouté avec confirmation visuelle	✓ Conforme
F05	Menu responsive	Ouvrir sur mobile (360px)	Menu fonctionnel et lisible	✓ Conforme

DOSSIER PROFESSIONNEL (DP)

Tests applicatifs

ID	Fonction testée	Étapes	Résultat attendu	Résultat obtenu
A01	Création de commande	Sélectionner produit et valider	Enregistrement en base avec statut "pending"	✓ OK
A02	Rôle administrateur	Accéder au tableau de bord	Accès autorisé uniquement pour admin	✓ OK
A03	Rôle utilisateur	Essayer d'accéder à /admin	Message d'erreur + redirection accueil	✓ OK
A04	Validation formulaire	Laisser champ vide	Message d'erreur clair	✓ OK
A05	Déconnexion	Cliquer "Se déconnecter"	Session fermée, retour à l'accueil	✓ OK

Tests back-end

ID	Fonction testée	Étapes	Résultat attendu	Résultat obtenu
B01	Création d'un utilisateur	Enregistrement via formulaire	L'entrée est bien ajoutée dans la table users	Conforme
B02	Commande avec plusieurs prod	Valider une commande avec 2 articles	Insertion dans orders + table pivot order_product	Conforme
B03	Suppression d'un utilisateur	Supprimer un utilisateur existant	Suppression cascade dans orders	Conforme
B04	Accès rôles et permissions	Connexion admin vs utilisateur	Accès admin au tableau de bord uniquement	Conforme

Tests d'accessibilité (via Lighthouse)

Critère	Description	Résultat
Contraste des couleurs	Boutons, texte, fond vérifiés	Conforme
Navigation clavier	Accès complet sans souris	Conforme
Balises ALT	Présentes sur toutes les images	Conforme
Structure HTML	Titres hiérarchisés, balises sémantiques	Conforme
Score global Lighthouse	Accessibilité : 91/100	Conforme

2. Précisez les moyens utilisés :

J'ai utilisé Laravel 12 et Inertia.js , 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.

DOSSIER PROFESSIONNEL^(DP)

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 : *07/04/2025* au : *07/07/2025*

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL^(DP)

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

Activité-type 2

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 :

DOSSIER PROFESSIONNEL (DP)

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.

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 MySQL 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 contrôleur
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.

DOSSIER PROFESSIONNEL (DP)

Deux rôles principaux sont définis : **Administrateur** et **Utilisateur**.

- **Administrateur** : accès complet au tableau de bord (gestion des utilisateurs, produits, commandes, statistiques)
Exemple : suppression d'un produit ou modification d'un compte utilisateur
- **Utilisateur** : accès uniquement à son espace personnel et à ses propres données.
Exemple : consultation de ses commandes et mise à jour de son profil

```
● ● ●

<?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 MySQL 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.

DOSSIER PROFESSIONNEL (DP)



```
<?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.

DOSSIER PROFESSIONNEL (DP)

```
<?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.

DOSSIER PROFESSIONNEL (DP)

```
<?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.

Le modèle de données repose sur une architecture relationnelle simple et cohérente, adaptée à une application e-commerce.

Il se compose des entités Utilisateurs, Produits, Commandes, Catégories, reliées entre elles pour assurer la traçabilité et la cohérence des informations.

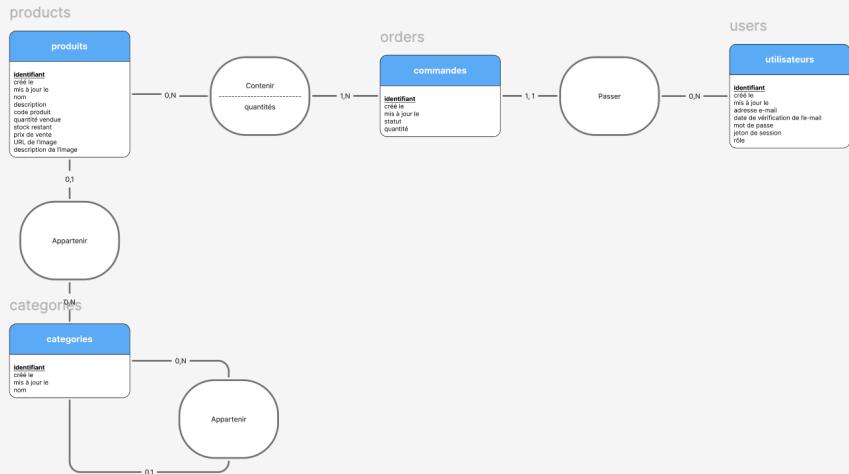
Chaque entité est représentée par une table dans la base MySQL, avec des relations bien définies :

Relation	Type	Détail / clé étrangère	Contrainte
Utilisateur → Commande	1 → N	orders.user_id → users.id	onDelete('cascade')
Commande ↔ Produit	N ↔ N	Table pivot order_product (order_id, product_id)	onDelete('cascade')
Produit → Catégorie	N → 1	products.category_id → categories.id	onUpdate('cascade')
Utilisateur → Rôle	N → 1	users.role_id → roles.id	onDelete('restrict')

Cette modélisation permet :

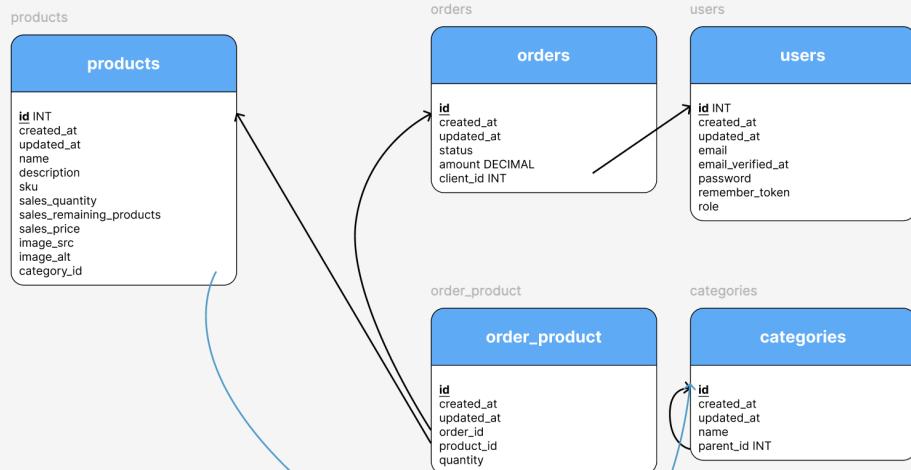
- une navigation simple entre les utilisateurs, commandes et produits ;
- une gestion claire des rôles et des autorisations ;
- une suppression en cascade des données dépendantes ;
- une traçabilité complète des interactions.

DOSSIER PROFESSIONNEL (DP)

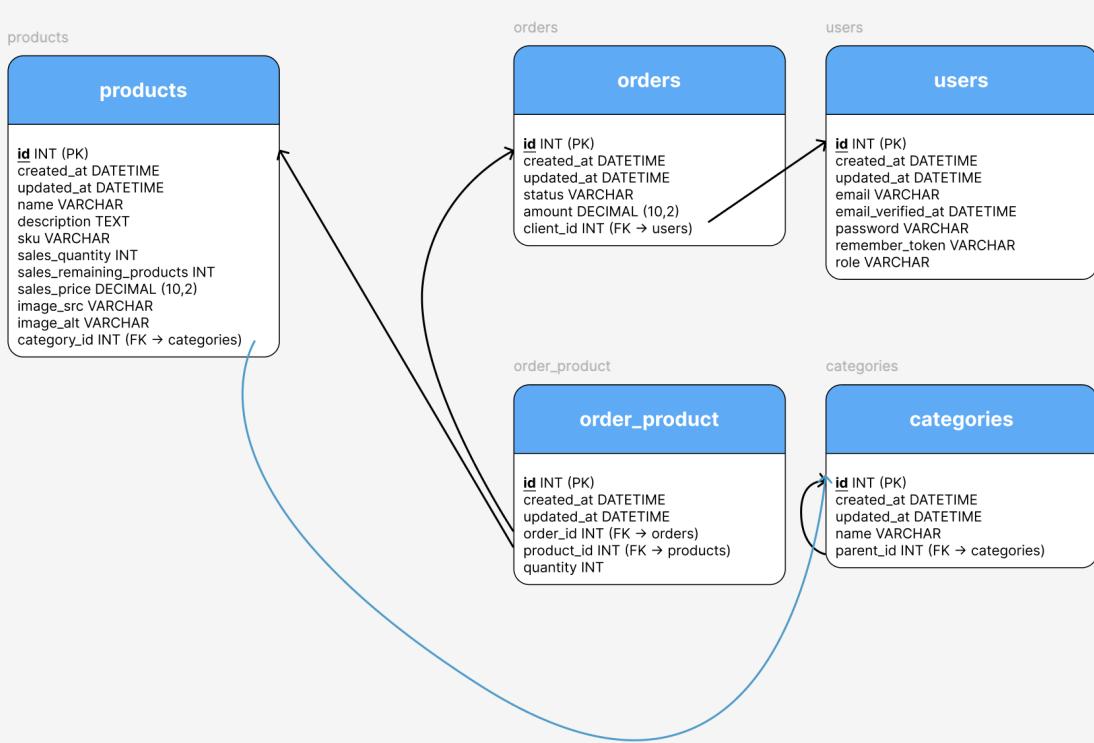


Ce MCD a ensuite été transformé en MLD (Modèle Logique de Données), permettant de structurer précisément les relations entre les entités (clé primaire, clé étrangère, cardinalité...). Cela m'a permis de générer des migrations Laravel cohérentes avec la logique métier.

DOSSIER PROFESSIONNEL (DP)



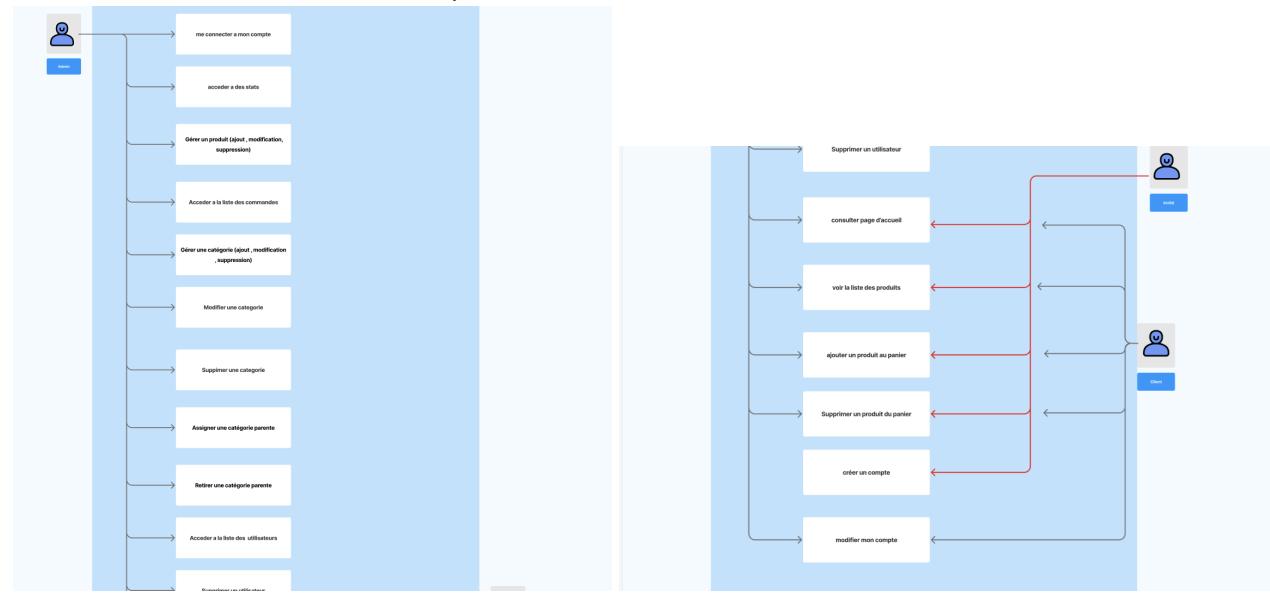
Ensuite, j'ai transformé mon MLD (Modèle Logique de Données) en MPD (Modèle Physique de Données)



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,

DOSSIER PROFESSIONNEL (DP)

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)
- MySQL : 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 MySQL 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)

DOSSIER PROFESSIONNEL^(DP)

- 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 : **07/04/2025** au : **07/07/2025**

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 - 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 MySQL 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.

Des **seeders** ont été utilisés pour insérer automatiquement des données de test et valider les opérations CRUD.

Le tableau ci-dessous présente un exemple de jeu d'essai vérifiant la cohérence de la base et le bon fonctionnement des relations

Cas de test	Entrée / action	Résultat attendu	Résultat obtenu
Création d'un utilisateur	Enregistrement d'un nouvel utilisateur	Données ajoutées dans la table users	<input checked="" type="checkbox"/> Conforme
Commande avec plusieurs produit	Ajout via table pivot order_product	Lignes créées pour chaque produit associé	<input checked="" type="checkbox"/> Conforme
Suppression d'un utilisateur	Suppression dans users	Suppression en cascade des commandes associées	<input checked="" type="checkbox"/> Conforme
Modification d'une catégorie	Changement du nom dans categories	Produits liés mis à jour automatiquement	<input checked="" type="checkbox"/> Conforme

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

DOSSIER PROFESSIONNEL^(DP)

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 MySQL, 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.
- MySQL : Base de données relationnelle légère, utilisée localement pour le développement et les tests.
- DB Client (extension Visual Studio Code) : Pour interroger directement la base MySQL, 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.

DOSSIER PROFESSIONNEL (DP)

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 : **07/04/2025** au : **07/07/2025**

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 - CP7 Développer des composants métier côté serveur

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

DOSSIER PROFESSIONNEL (DP)

Tâches réalisées et conditions d'exécution

Le développement des composants métier a été mené sous Laravel 12 (PHP 8.3), base MySQL 8, avec front React via Inertia.js.

Travail effectué d'abord en local (Docker) pour itération rapide, puis mis en production (o2switch) après validation fonctionnelle.

Composants métier livrés (côté serveur)

- Gestion du catalogue : création/édition/suppression de produits, rattachement à une catégorie, contrôle du stock

```
public function dashboard()
{
    $orders = Order::with('client', 'products')->get();

    $stats = [
        'totalOrders'      => Order::whereIn('status', ['pending', 'paid'])->count(),
        'pendingOrders'   => Order::where('status', 'pending')->count(),
        'paidOrders'       => Order::where('status', 'paid')->count(),
        'activeOrders'     => Order::where('status', 'active')->count(),
        'completedOrders' => Order::where('status', 'completed')->count(),
        'returnOrders'     => Order::where('status', 'returned')->count(),
    ];

    // === Compatibilité MySQL & SQLite ===
    $driver = DB::connection()->getDriverName(); // 'mysql' ou 'sqlite'
    $year  = 2025; // ou (int) date('Y') si tu veux l'année courante

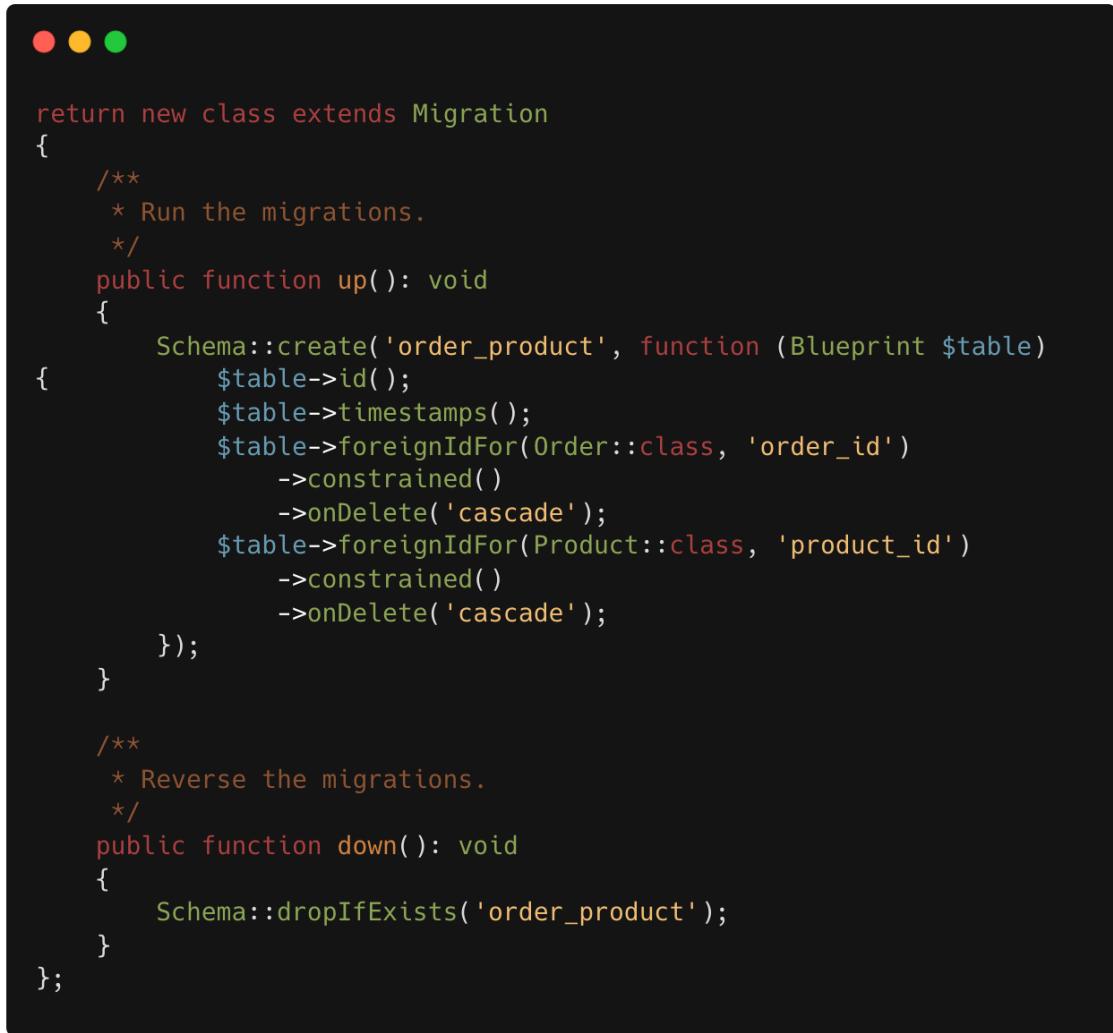
    if ($driver === 'mysql') {
        $monthExpr = 'MONTH(orders.created_at)';

        $monthlySales = DB::table('orders')
            ->join('order_product', 'orders.id', '=', 'order_product.order_id')
            ->join('products', 'order_product.product_id', '=', 'products.id')
            ->selectRaw("$monthExpr AS month, SUM(order_product.quantity * products.sales_price) AS total_sales")
            ->whereYear('orders.created_at', $year)
            ->groupByRaw($monthExpr)
            ->orderByRaw($monthExpr)
            ->pluck('total_sales', 'month');
    } else { // sqlite
        $monthExpr = "CAST(strftime('%m', orders.created_at) AS INTEGER)";

        $monthlySales = DB::table('orders')
            ->join('order_product', 'orders.id', '=', 'order_product.order_id')
            ->join('products', 'order_product.product_id', '=', 'products.id')
            ->selectRaw("$monthExpr AS month, SUM(order_product.quantity * products.sales_price) AS total_sales")
            ->whereRaw("CAST(strftime('%Y', orders.created_at) AS INTEGER) = ?", [$year])
            ->groupByRaw($monthExpr)
            ->orderByRaw($monthExpr)
            ->pluck('total_sales', 'month');
    }
}
```

DOSSIER PROFESSIONNEL (DP)

- Panier & commandes : ajout/suppression d'articles, calcul des totaux, validation de commande avec contrôle du stock et de l'intégrité des lignes.



```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('order_product', function (Blueprint $table)
        {
            $table->id();
            $table->timestamps();
            $table->foreignIdFor(Order::class, 'order_id')
                ->constrained()
                ->onDelete('cascade');
            $table->foreignIdFor(Product::class, 'product_id')
                ->constrained()
                ->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('order_product');
    }
};
```

- Comptes & rôles : profil utilisateur, rôle “admin” pour le back-office, rôle “utilisateur” pour le front, restrictions d'accès centralisées.
- Tableau de bord : agrégations SQL pour statistiques (volume de commandes, produits actifs), mises à disposition au front par Inertia (props).

Règles métier (exemples concrets)

- Commande : impossible de valider si le stock d'au moins un produit est insuffisant (contrôle côté serveur avant insertion des lignes).
- Prix & totaux : le total commande = somme des lignes (prix unitaire × quantité) ; recalcul et verrouillage du montant au moment de la validation

DOSSIER PROFESSIONNEL^(DP)

- Rôles & permissions : seules les routes d'administration permettent le CRUD des produits et des utilisateurs ; les utilisateurs classiques n'y accèdent pas

Architecture IO

- Contrôleurs concentrés sur l'orchestration ; logique métier encapsulée dans des Services et Repertoires pour séparer règles et accès aux données
- Form Requests pour la validation serveur (formats, bornes, messages)
- Ressources/transformers pour ne renvoyer au front que les données utiles (cohérence des payloads Inertia)
- Refactoring réalisé : élimination de redondances, méthodes trop longues scindées, nommage standardisé

Sécurité côté serveur

Validation serveur systématique (Form Request) pour prévenir injections et données incohérentes.

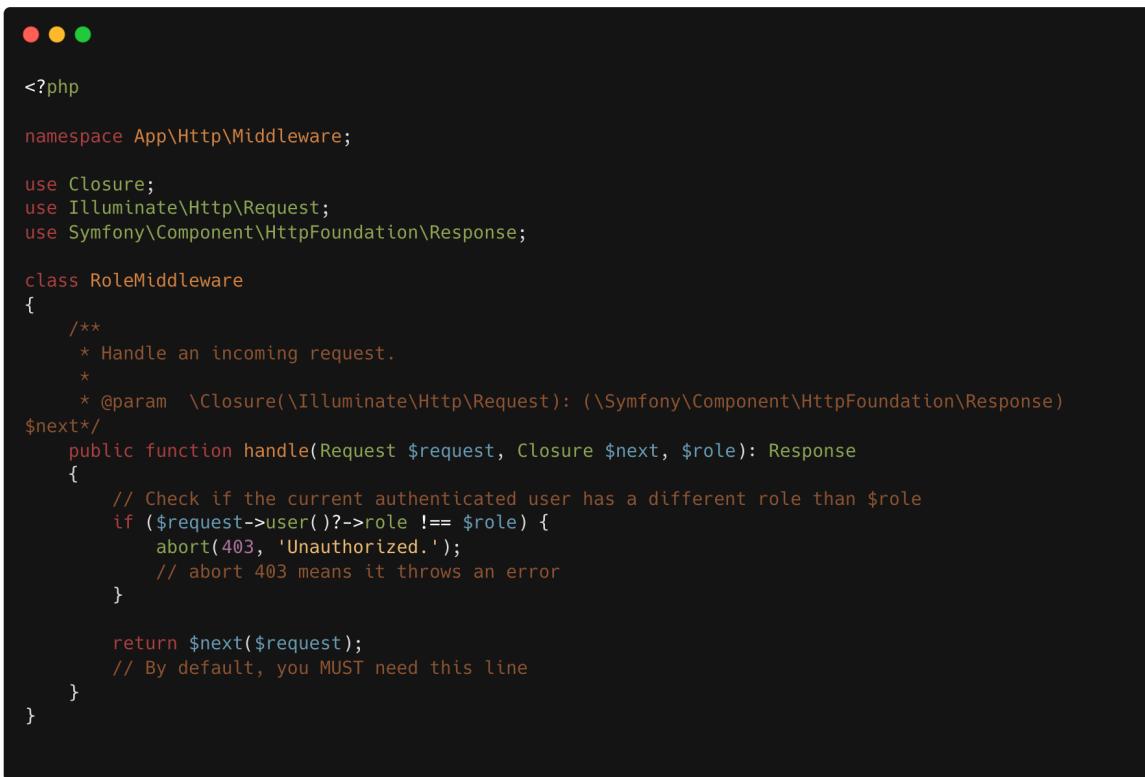
- Rôles & middlewares (auth, contrôle d'accès admin) sur toutes les routes sensibles
- CSRF actif sur les formulaires, hashage des mots de passe
- Messages d'erreur non verbeux : pas d'informations techniques exposées à l'utilisateur

Scénarios d'usage sécurisés (rôles)

- Administrateur : accès au Dashboard, gestion CRUD des produits/utilisateurs, accès aux statistiques

DOSSIER PROFESSIONNEL (DP)

- Utilisateur : gestion de son profil, consultation du catalogue, panier, création de commande ; accès refusé aux routes d'admin



```
<?php

namespace App\Http\Middleware;

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

class RoleMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)
     $next
     */
    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
    }
}
```

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

DOSSIER PROFESSIONNEL (DP)

- 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 ?

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 : *07/04/2025* au : *07/07/2025*

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 - 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 :

Pour simuler un environnement de production, j'ai d'abord utilisé Docker, afin de rendre l'application totalement autonome et portable.

J'ai configuré plusieurs conteneurs pour exécuter PHP 8.3, MySQL 8.0 et Nginx, ce qui m'a permis de reproduire fidèlement les conditions d'un serveur réel.

Grâce à Docker Compose, le lancement de l'ensemble du projet a été automatisé.

J'ai également adapté l'application à cet environnement en :

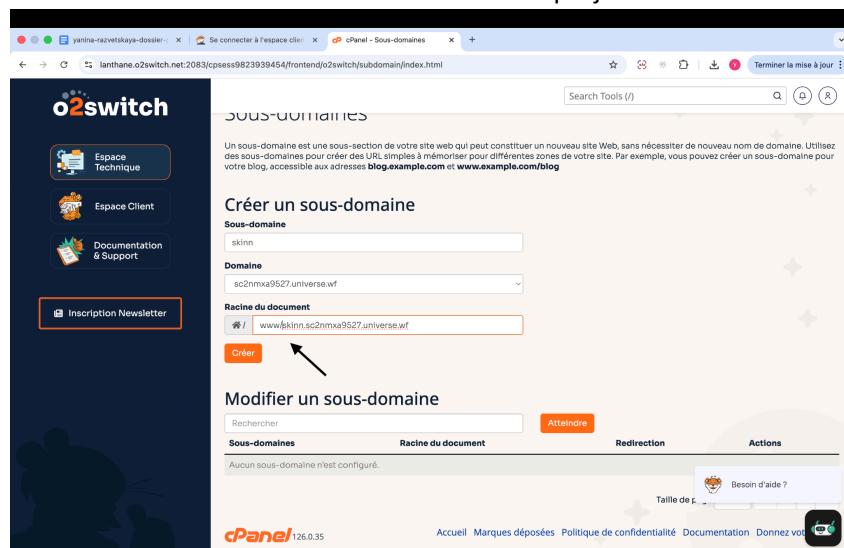
- créant un Dockerfile personnalisé pour le conteneur PHP
- configurant Nginx à l'aide du fichier default.conf
- construisant le front-end avec la commande npm run build
- exécutant les migrations Laravel avec php artisan migrate

Par la suite, j'ai décidé de déployer le projet sur l'hébergeur o2switch afin de le rendre accessible en ligne.

L'application est hébergée sur o2switch. Le déploiement suit un processus simple

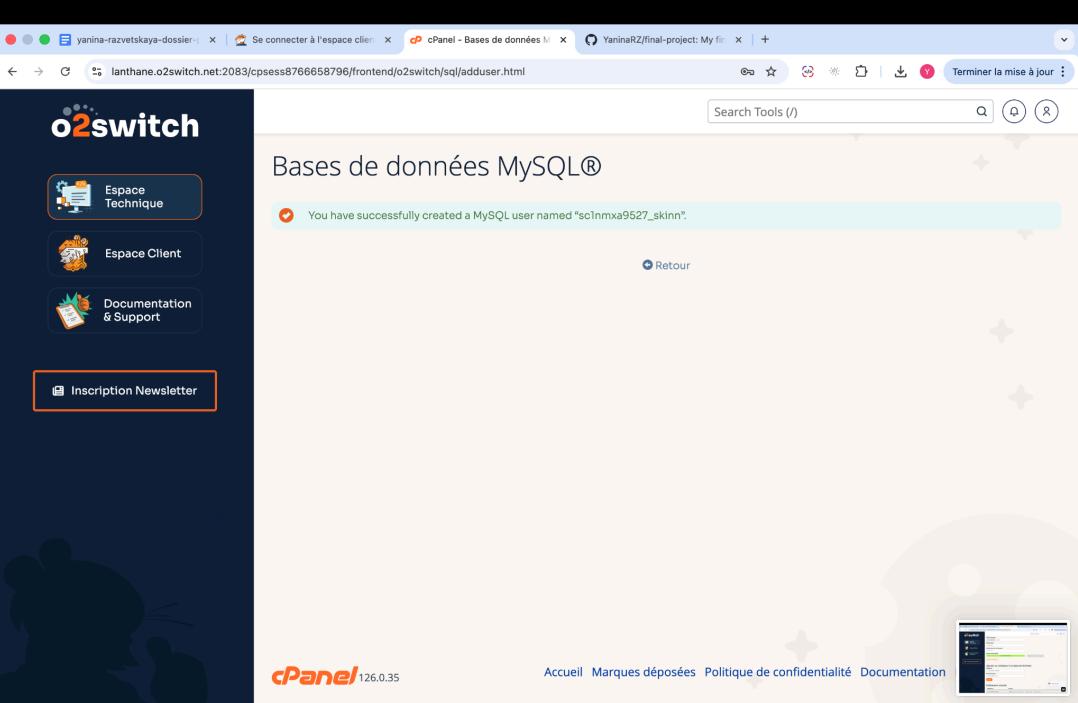
Étapes principales :

- Création d'un sous-domaine dédié au projet

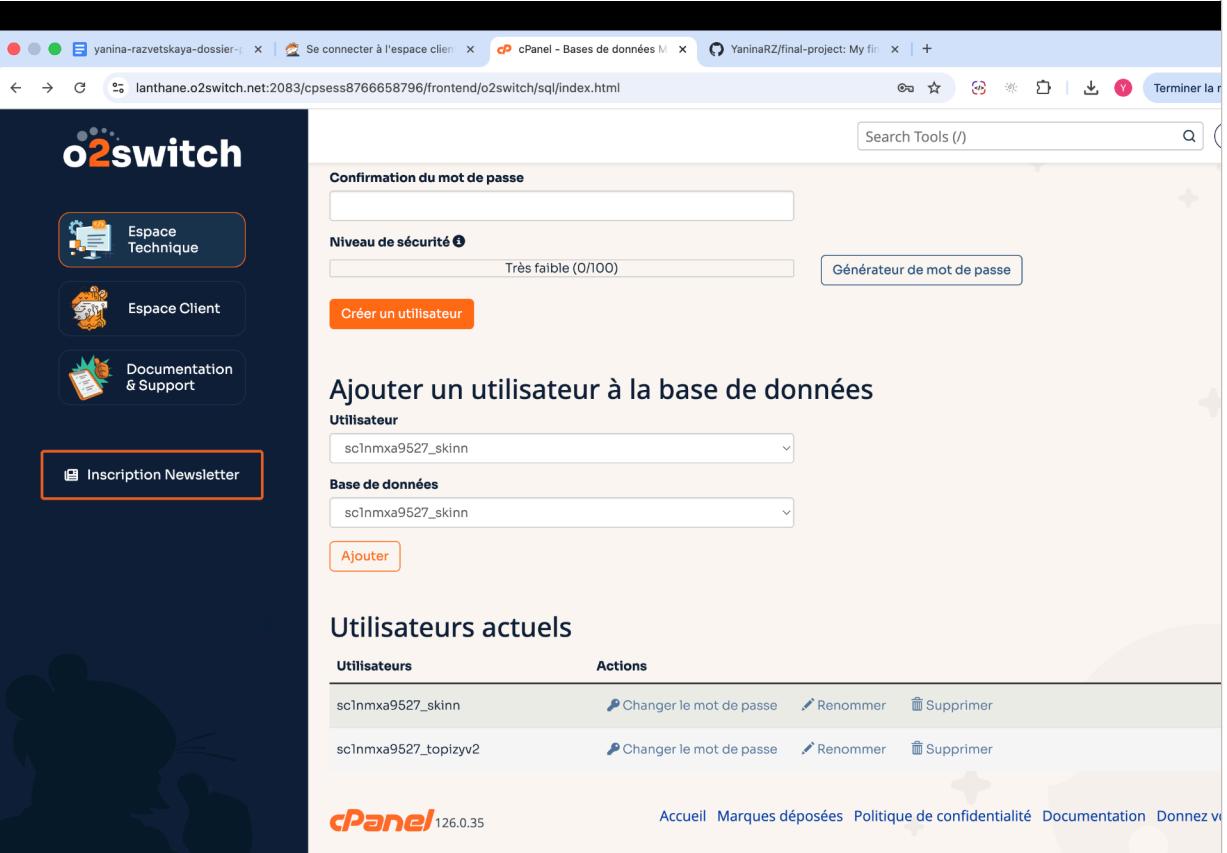


- Création de la base MySQL et d'un utilisateur avec priviléges complets

DOSSIER PROFESSIONNEL (DP)



The screenshot shows the MySQL user creation process. A success message at the top states: "You have successfully created a MySQL user named "sc1nmxa9527_skinn".



The screenshot shows the "Confirmation du mot de passe" (Password confirmation) section where a password has been entered. Below it, the "Niveau de sécurité" (Security level) is shown as "Très faible (0/100)". A "Générateur de mot de passe" (Password generator) button is available. The "Créer un utilisateur" (Create user) button is visible.

Ajouter un utilisateur à la base de données

Utilisateur: sc1nmxa9527_skinn

Base de données: sc1nmxa9527_skinn

Actions:

Utilisateurs	Actions
sc1nmxa9527_skinn	Changer le mot de passe Renommer Supprimer
sc1nmxa9527_topizyv2	Changer le mot de passe Renommer Supprimer

DOSSIER PROFESSIONNEL (DP)

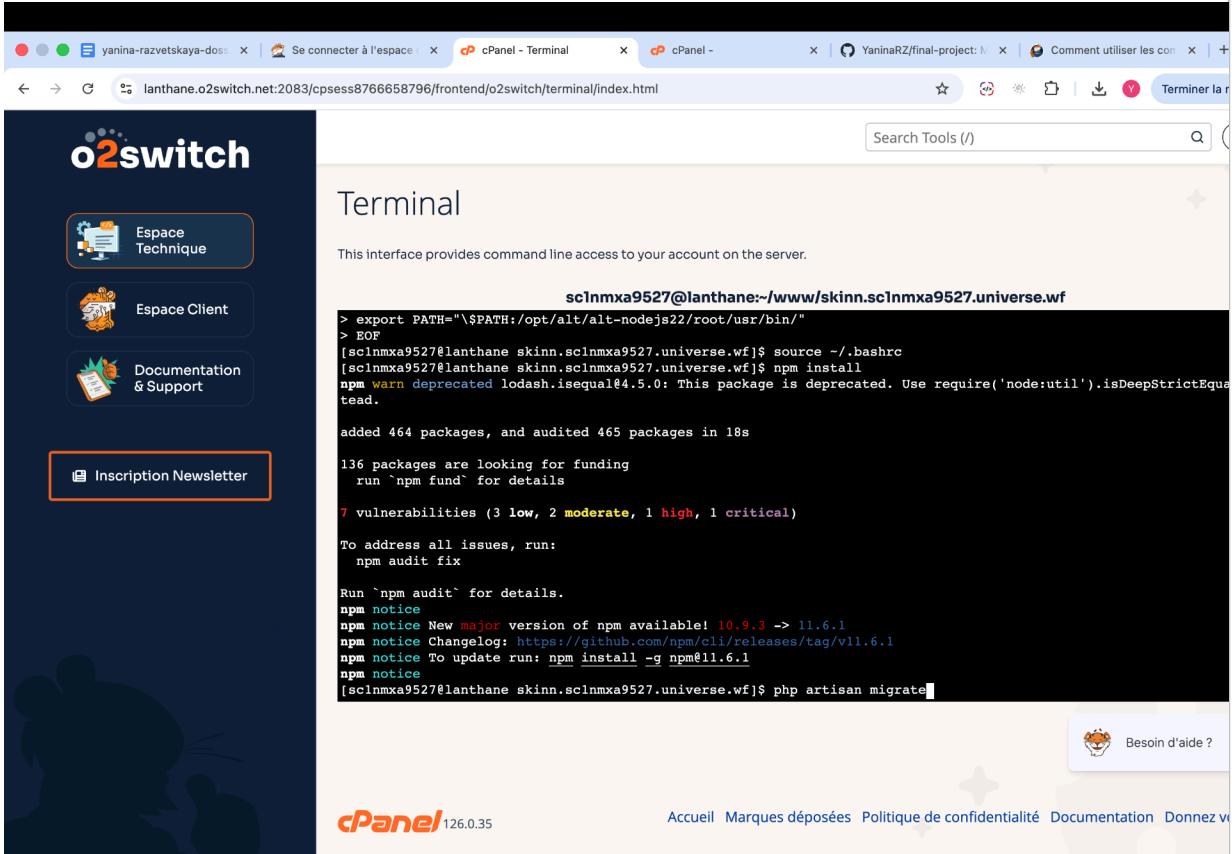
The screenshot shows a web-based MySQL user privilege management tool. On the left, there's a sidebar with the o2switch logo and links for 'Espace Technique', 'Espace Client', 'Documentation & Support', and an 'Inscription Newsletter' button (which is highlighted with an orange border). The main content area is titled 'Gérer les priviléges des utilisateurs'. It shows a user named 'sc1nmx9527_skinn' and a database named 'sc1nmx9527_skinn'. A section titled 'TOUS LES PRIVILÉGES' contains two columns of checkboxes for various MySQL privileges. The checked items are: ALTER, CREATE, CREATE TEMPORARY TABLES, DELETE, EVENT, INDEX, LOCK TABLES, SELECT, and TRIGGER in the first column; and ALTER ROUTINE, CREATE ROUTINE, CREATE VIEW, DROP, EXECUTE, INSERT, REFERENCES, SHOW VIEW, and UPDATE in the second column. At the bottom of this section are two buttons: 'Apporter des modifications' (Apply changes) and 'Réinitialiser' (Reset).

- 3. Clonage du dépôt GitHub dans le dossier du sous-domaine via le terminal cPanel

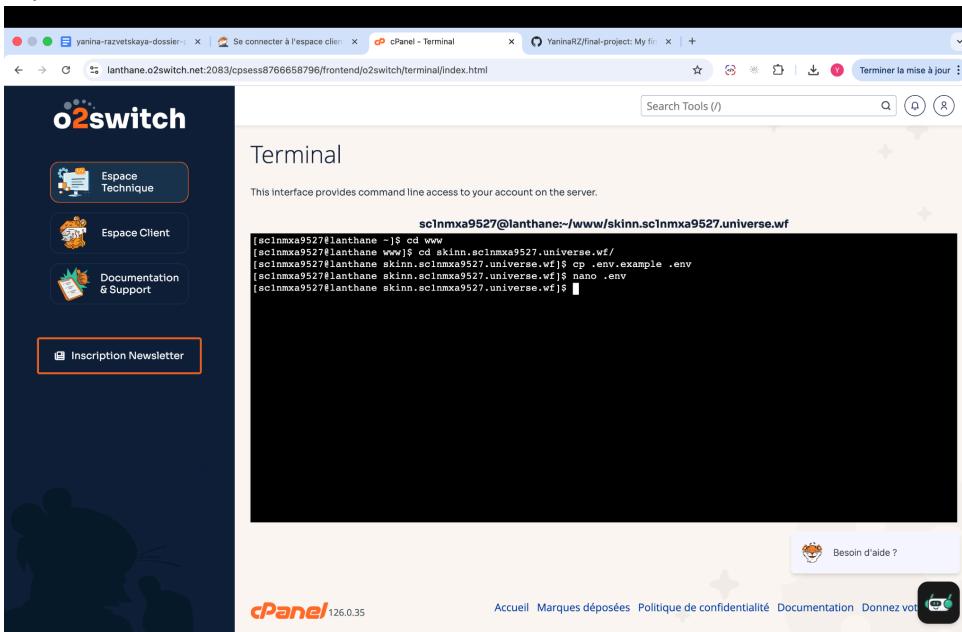
The screenshot shows a GitHub repository page for 'YaninaRZ / final-project'. The repository is public and has 2 branches and 0 tags. The 'Code' tab is selected. On the right side, there's a 'Clone' section with options for 'HTTPS', 'SSH', and 'GitHub CLI'. The 'HTTPS' option is highlighted with an orange border, and its URL is displayed: 'https://github.com/YaninaRZ/final-project'. Below this, there are links for 'Open with GitHub Desktop' and 'Download ZIP'. To the right of the clone section, there are sections for 'About', 'Releases', 'Packages', and 'Languages'. The 'About' section includes a description of the project as 'My final project' and lists 'Readme', 'Activity', 'Stars', 'Watching', and 'Forks'. The 'Languages' section shows a chart where JavaScript is 75.7%, PHP is 23.0%, and Other is 1.3%.

DOSSIER PROFESSIONNEL (DP)

- Installation des dépendances avec npm install



- Copier .env



DOSSIER PROFESSIONNEL (DP)

- Génération de la clé d'application : php artisan key:generate

The screenshot shows a web browser with multiple tabs. The main tab displays the o2switch dashboard with sections for Espace Technique, Espace Client, Documentation & Support, and an Incription Newsletter button. To the right of the dashboard is a terminal window titled "Terminal". The terminal shows the command "php artisan key:generate" being run, with the output indicating the success of the operation.

```
sc1nmxa9527@lanthane:~/www/skinn.sc1nmxa9527.universe.wf
- Installing stripe/stripe-php (v17.3.0): Extracting archive
- Installing tightenco/ziggy (v2.5.2): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

[INFO] Discovering packages...
inertiajs/inertia ...
laravel/pail ...
laravel/sail ...
laravel/tinker ...
nesbot/carbon ...
nunomaduro/collision ...
nunomaduro/termwind ...
tightenco/ziggy ...

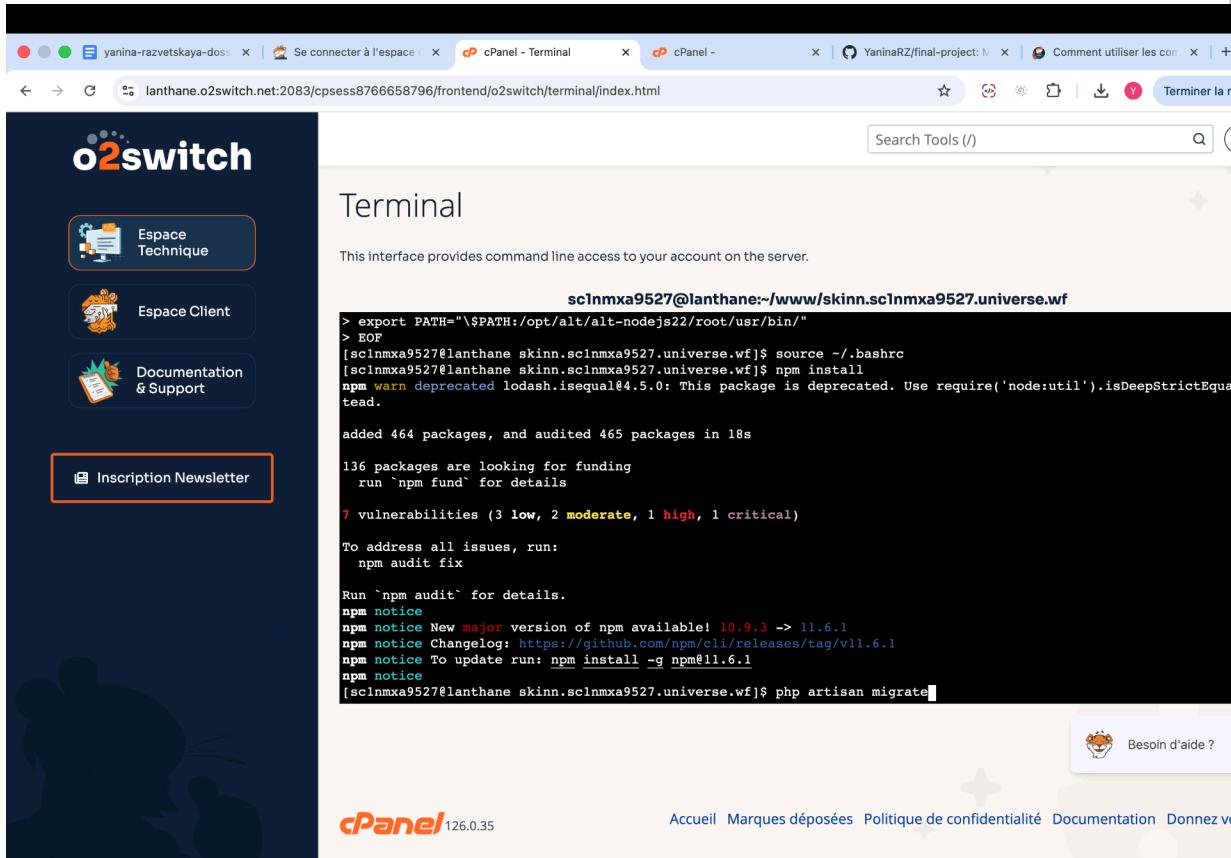
80 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
[sclnmxa9527@lanthane skinn.sclnmxa9527.universe.wf]$ php artisan key:generate

[INFO] Application key set successfully.

[sclnmxa9527@lanthane skinn.sclnmxa9527.universe.wf]$
```

DOSSIER PROFESSIONNEL (DP)

- Migration de la base de données : php artisan migrate.



L'application a été pensée pour offrir une expérience claire, fluide et intuitive.

Côté utilisateur :

- Créer un compte via la page d'inscription
- Se connecter à son espace personnel
- Consulter les produits disponibles dans Our Collection
- Ajouter un ou plusieurs articles au panier
- Valider la commande et consulter l'historique des achats

Côté administrateur :

- Se connecter avec un compte administrateur
- Accéder au tableau de bord
- Gérer les utilisateurs et les produits (ajout, modification, suppression)

Technical overview (EN)

Project: Skinn

Goal: Create a responsive and secure e-commerce web app to manage users, products, and orders.

Stack: Laravel 12 (PHP 8.3), MySQL 8, Inertia.js, React, Tailwind CSS.

Features: Authentication (user/admin), CRUD for products, shopping cart, and dashboard with

DOSSIER PROFESSIONNEL (DP)

Chart.js.

Accessibility: Checked with Lighthouse (91/100).

Performance: Optimized with image compression, and browser caching.

Deployment: Hosted on o2switch (cPanel, PHP/MySQL).

Security: Input validation, CSRF protection, password hashing, and restricted admin routes.

Une veille régulière a été réalisée tout au long du projet pour suivre les évolutions du développement web et les bonnes pratiques liées à Laravel et React.

Sources consultées :

- Laravel News – nouveautés du framework et bonnes pratiques de sécurité
- Smashing Magazine – accessibilité, performance et UX
- MDN Web Docs – référence technique (HTML, CSS, JavaScript)

Apports concrets :

- Application des bonnes pratiques Lighthouse (optimisation des performances)
- Amélioration de l'accessibilité (contraste, ARIA, structure sémantique)
- Sécurisation des formulaires et validation côté serveur

Cette veille a permis d'adopter les standards actuels du développement web et de maintenir un code propre, sécurisé et performant.

L'application est désormais pleinement fonctionnelle, hébergée en ligne et conforme aux critères de performance, d'accessibilité et de sécurité attendus dans un environnement professionnel.

2. Précisez les moyens utilisés :

PHP 8.3, Composer, Node.js, et Nginx étaient intégrés dans l'image Docker afin de simuler un environnement de production complet.

Fichiers configurés :

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

Pour le déploiement de l'application, j'ai utilisé l'hébergeur o2switch..

3. Avec qui avez-vous travaillé ?

J'ai travaillé sur ce projet seule.

DOSSIER PROFESSIONNEL (DP)

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 : **07/04/2025** au : **07/07/2025**

5. Informations complémentaires (*facultatif*)

DOSSIER PROFESSIONNEL^(DP)

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 : 07/04/2025 au : 07/07/2025

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 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 Cannes

le 09/10/2025

pour faire valoir ce que de droit.

Signature :



DOSSIER PROFESSIONNEL (DP)

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé

DOSSIER PROFESSIONNEL ^(DP)

ANNEXES

(Si le RC le prévoit)