



Application E-Commerce

**Projet réalisé dans le cadre de la présentation au
Titre Professionnel Développeur Web et Web Mobile**

Présenté par

Yanina Razvetskaya

REMERCIEMENTS.....	4
INTRODUCTION.....	5
Activité 1 : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité.....	6
A. Installer l'environnement de développement.....	6
B. Maquetter une application.....	6
C. Réaliser une interface utilisateur web statique et adaptable.....	7
D. Développer une interface utilisateur web dynamique.....	7
Activité 2 : Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité.....	7
A. Créer une base de données.....	7
B. Développer les composants d'accès aux données.....	8
C. Développement Back-End avec Laravel, Eloquent et Inertia.js.....	8
D. Tests et déploiement.....	9
RÉSUMÉ DU PROJET – SKINN.....	9
Cahier des charges.....	11
1. Contexte et Objectifs.....	11
1.1 Contexte.....	11
1.2 Objectifs.....	11
2. Fonctionnalités Clés.....	11
2.1 Front-office (Expérience Client).....	11
3. Public Cible et UX.....	12
3.1 Personas.....	12
3.2 Adaptations UX.....	12
4. Spécifications Techniques.....	12
Stack :.....	12
3. Feuille de route.....	14
4. Produit Minimum Viable (MVP).....	16
A. Front-office (Client).....	16
B. Back-office (Admin).....	16
Mise en place de l'application.....	17
Zoning.....	17
Wireframes.....	18
Chartes graphique et logo.....	18
Identité visuelle : couleurs et typographie.....	19
Choix des couleurs.....	19
Spécifications techniques.....	21
Gestion de version.....	21
Choix techniques.....	23
Front-End.....	23
Back-End.....	23

ACCESSEURITÉ.....	24
Types d'appareils compatibles.....	24
Contrôle et validation.....	25
Architecture du projet.....	26
MCD.....	26
MLD.....	26
MPD.....	27
Use case.....	28
Utilisation des Seeders pour l'insertion des données.....	29
Sécurité.....	31
Sécurisation des routes.....	31
Sécurisation et protection CSRF.....	32
Contrôle d'unicité de l'adresse e-mail pour sécuriser l'inscription.....	33
CORS.....	33
Tests dans un projet Laravel.....	34
Test unitaires.....	34
Détail des Échecs.....	35
Test fonctionnels.....	35
Cahier de tests.....	36
Gestion de projet.....	38
Organisation du travail — cycle en V.....	38
Analyse des besoins.....	38
Conception.....	39
Développement.....	39
Déploiement.....	39
Conclusion.....	40

REMERCIEMENTS

Je tiens à remercier tout d'abord mon formateur Sambeau P.n pour son accompagnement tout au long de cette formation. Sa pédagogie, sa patience et sa disponibilité ont été essentielles à ma progression. Grâce à lui, j'ai pu développer mes compétences et mener ce projet à bien avec confiance.

Je remercie également Aïcha, responsable de site, pour son soutien et sa bienveillance tout au long de mon parcours.

Enfin, un grand merci à mon amie Nadia pour son encouragement constant et sa présence précieuse.

INTRODUCTION

Après un parcours universitaire en traduction puis en droit, j'ai ressenti le besoin de me réorienter vers un domaine qui me correspond davantage et dans lequel je pourrais m'épanouir sur le long terme. C'est ainsi que je me suis tourné vers le développement web, avec l'objectif clair d'en faire mon futur métier.

Pour entamer cette reconversion, j'ai intégré l'école La Plateforme, une formation qui me permet de poser des bases solides tout en me préparant à poursuivre mes études à Ynov, dans l'optique d'obtenir un Master. J'ai fait le choix de me former au développement full-stack, car j'apprécie autant les aspects techniques du back-end que les aspects visuels et interactifs du front-end. Pouvoir comprendre et réaliser toutes les parties d'un site m'apporte une vraie satisfaction, et c'est vers ce type de polyvalence que je souhaite construire mon avenir professionnel.

Dans le cadre de cette formation, j'ai développé un projet final nommé Skinn, une boutique en ligne dédiée aux soins corporels naturels. Ce projet a été l'occasion de concevoir un site complet, en partant de zéro, en mettant en œuvre tout ce que j'ai appris jusqu'ici, et en approfondissant ma capacité à construire une application cohérente de bout en bout.

Cette première étape n'est que le début d'un parcours que je souhaite mener jusqu'au bout, avec l'objectif de devenir développeur full-stack à l'issue de mes études.

Activité 1 : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

A. Installer l'environnement de développement

Pour ce projet, j'ai configuré un environnement de développement complet en utilisant Laravel comme framework back-end, Inertia.js pour la liaison entre le back-end et le front-end, et React pour la partie interface utilisateur.

J'ai démarré en installant Laravel via Composer, puis j'ai intégré le starter kit Laravel avec le support Inertia.js et React, ce qui m'a permis d'avoir une base fonctionnelle avec une architecture moderne et réactive.

B. Maquetter une application

Dans le cadre de ce projet, j'ai réalisé une maquette complète d'une application web en suivant une méthodologie rigoureuse et en utilisant Figma comme outil principal. Mon processus s'est articulé autour de trois étapes clés :

1. Zoning

J'ai commencé par établir un zoning pour structurer l'espace de l'application. Cette étape m'a permis de définir les zones principales (header, contenu, sidebar, footer, etc.) et d'organiser les éléments de manière logique, en tenant compte de l'expérience utilisateur. Le zoning a servi de base pour la suite de la conception.

2. Wireframe

Ensuite, j'ai créé des wireframes détaillés en version desktop et mobile. Ces schémas, réalisés en noir et blanc dans Figma, m'ont permis de :

- Positionner précisément les composants (boutons, menus, images, formulaires).
- M'assurer que l'interface était intuitive et accessible.

3. Maquette haute fidélité

Une fois les wireframes validés, j'ai développé une maquette interactive et responsive en intégrant :

- Une charte graphique cohérente (couleurs, typographies, icônes).
- Des composants réutilisables pour garantir la cohérence du design.
- Des micro-interactions pour améliorer l'expérience utilisateur.

C. Réaliser une interface utilisateur web statique et adaptable

Pour cette partie du projet, j'ai développé une interface web responsive en utilisant Tailwind CSS, un framework utilitaire qui m'a permis de concevoir rapidement des composants modernes et adaptatifs.

D. Développer une interface utilisateur web dynamique

Au vu des users stories réalisées en amont de la réalisation du projet, il était impératif que mon E-Commerce dispose d'une interface dynamique. Dès lors, je me suis très rapidement tourné vers le starter kit de Laravel (avec Inertia.js et React) pour bénéficier de ses fonctionnalités intégrées et de son écosystème complet, tout en assurant une intégration fluide entre le front-end et le back-end.

Activité 2 : Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

A. Créer une base de données

Pour concevoir la base de données de mon application, j'ai choisi d'utiliser la méthode MERISE, une approche d'analyse et de conception de systèmes d'information qui permet de structurer la réflexion autour des données et des traitements.

Elle se base sur une séparation claire entre trois niveaux :

- le Modèle Conceptuel de Données (MCD)
- le Modèle Logique de Données (MLD)
- et le Modèle Physique de Données (MPD)

Le MCD m'a permis d'identifier les entités principales (*Utilisateur, Produit, Commande, Catégorie*), leurs attributs et les relations entre elles (par exemple : *un utilisateur passe plusieurs commandes*). Une fois le MCD validé, je l'ai transformé en MLD en précisant les clés primaires, les clés étrangères et les cardinalités, puis en MPD, adapté au système de gestion de base de données MySQL.

B. Développer les composants d'accès aux données

J'ai structuré la persistance des données avec Laravel/Eloquent, combiné à Inertia.js pour le front. Les migrations versionnées ont permis d'évoluer le schéma de base de données de manière contrôlée. Cette stack intégrée a offert un bon équilibre entre productivité et maintenabilité, tout en garantissant la cohérence des données à travers les différentes couches de l'application.

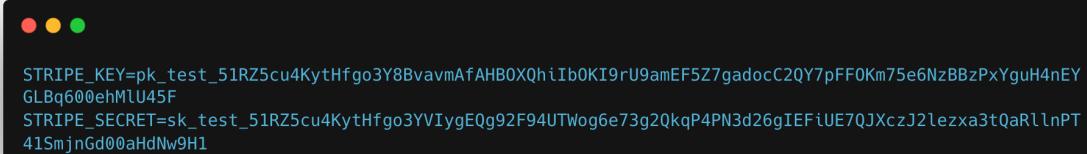
C. Développement Back-End avec Laravel, Eloquent et Inertia.js

Le back-end est développé en Laravel dans une architecture monolithique. Les données sont gérées avec Eloquent (ORM) et exposées aux vues via Inertia.js, qui sert de pont entre Laravel et les composants React. La navigation est de type SPA côté client (transitions sans rechargement complet), tout en conservant le rendu et la logique côté serveur.

Intégration du paiement avec Stripe

Afin d'intégrer un système de paiement sécurisé dans l'application e-commerce, j'ai utilisé Stripe, une solution conforme aux normes de sécurité **PCI DSS**.

L'API de Stripe est utilisée pour créer une session de paiement à partir de la dernière commande de l'utilisateur connecté. Les clés API sont stockées dans le fichier d'environnement .env pour garantir la confidentialité.



```
STRIPER_KEY=pk_test_51RZ5cu4KytHfg03Y8BvavmAfAHB0XQhiIb0KI9rU9amEF5Z7gadocC2QY7pFF0Km75e6NzBBzPxYguH4nEY
GLBq600ehMLU45F
STRIPER_SECRET=sk_test_51RZ5cu4KytHfg03YVIygEqg92F94UTWog6e73g2QkqP4PN3d26gIEFiUE7QJXczJ2lezxa3tQaRllnPT
41SmjnGd00aHdNw9H1
```

D. Tests et déploiement

Une fois le développement achevé, j'ai réalisé des tests fonctionnels pour valider les différentes fonctionnalités du projet, notamment les formulaires, l'authentification, et la gestion des données utilisateurs.

Côté front-end, j'ai vérifié l'interactivité des composants React et la bonne intégration avec Inertia.js lors des échanges avec le serveur.

Pour le déploiement, j'ai préparé une procédure simplifiée permettant de migrer la base de données, compiler les assets front-end en mode production, et configurer les variables d'environnement. Le projet est ainsi prêt à être déployé sur un serveur de production ou une plateforme cloud avec un minimum d'efforts.

RÉSUMÉ DU PROJET – SKINN

Skinn est une boutique en ligne dédiée aux soins corporels naturels et innovants, pensée pour les personnes soucieuses de leur bien-être. L'objectif principal du projet était de proposer une solution à la fois épurée, intuitive et accessible, tant pour les utilisateurs que pour les administrateurs.

L'interface utilisateur a été conçue de manière minimaliste, afin d'offrir une expérience d'achat fluide, sans distraction. Chaque étape du parcours est simplifiée pour permettre à l'utilisateur de naviguer et de finaliser ses achats en toute sérénité. Du côté administration, un espace back-office permet de gérer facilement les produits et les catégories, avec des actions claires de création, modification ou suppression, même sans compétences techniques avancées.

Pour assurer la fiabilité des transactions, le système de paiement a été intégré avec Stripe, garantissant des paiements sécurisés (Stripe, SCA 3-D Secure) à chaque étape du processus.

Sur le plan technique, le projet repose sur Laravel pour le back-end, avec Eloquent pour la gestion des données. Le front-end est géré avec Inertia.js, permettant une navigation fluide et réactive, sans complexité excessive. Ce choix technologique a permis un développement rapide, maintenable et performant.

En résumé, Skinn propose une solution simple et fonctionnelle, où chaque élément est pensé pour aller à l'essentiel : acheter ou gérer des produits de soins du corps en toute simplicité, sans superflu.

Cahier des charges

1. Contexte et Objectifs

1.1 Contexte

Le marché des produits de soin est saturé, mais l'offre actuelle reste peu lisible pour les consommateurs à besoins spécifiques (peau sensible, allergies, préférences bio ou véganes). La navigation sur les sites concurrents est souvent complexe et générique.

1.2 Objectifs

- Offrir une expérience d'achat personnalisée et simplifiée
- Proposer une navigation claire avec des catégories hiérarchisées
- Permettre un achat rapide (Guest checkout) sans création de compte obligatoire

2. Fonctionnalités Clés

2.1 Front-office (Expérience Client)

- **Parcours d'achat fluide :**
- Navigation par catégories parents/enfants (ex. *Soins Visage* → *Crèmes hydratantes, Sérum*)
- **Guest checkout :**
 - Achat sans compte
 - Possibilité de créer un compte après commande.
- **Paiement en ligne :**
 - Intégration Stripe

2.2 Back-office (Administrateur)

- **Gestion des produits et stocks** (via Laravel + Inertia) :
 - CRUD produits avec variantes
 - Suivi des stocks
 - Organisation des catégories via relations Eloquent (parent/enfant)
- **Dashboard dynamique :**
 - Suivi du chiffre d'affaires
 - Commandes en cours de traitement

3. Public Cible et UX

3.1 Personas

The image shows a persona card for the brand Skinn®. On the left, there is a sidebar with three dark grey buttons labeled "Nom", "Age", and "Recherche". To the right, there are two user profiles. The first profile, "Emma", features a photo of a young woman with curly hair, a white lab coat, and a stethoscope around her neck. Below the photo, it says "Emma" and "28 ans", followed by the text "exige des produits naturels et bio". The second profile, "Luc", features a photo of a man in a suit and tie, holding a phone to his ear. Below the photo, it says "Luc" and "45 ans", followed by the text "Soucieux de sa routine de soin complète".

3.2 Adaptations UX

- Interface **responsive** (mobile et desktop)
- UI développée avec Tailwind CSS : rapide, épurée, personnalisable.
- Temps de chargement optimisé

4. Spécifications Techniques

Stack :

- Backend : Laravel (version utilisée : 12)
- Frontend : Inertia.js avec React
- ORM : Eloquent
- CSS : Tailwind CSS

2. 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
----------------	--	--

3. Feuille de route

Voici un extrait de ma feuille de route, centré sur les étapes clés :



Brief

Réalisation d'un e-commerce de vente de produits pour l'hygiène.

À qui ça s'adresse

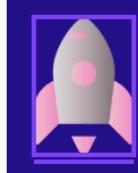
À tous ce qui veulent prendre soin de soi.

Questions	Réponses
Qui veut découvrir les produits ?	Les personnes soucieuses de leur hygiène et de leur bien-être, ainsi que celles cherchant des solutions naturelles ou innovantes pour prendre soin d'elles.
Comment faire découvrir les produits?	En proposant des visuels attractifs et professionnels, en offrant des échantillons gratuits, et en utilisant les réseaux sociaux pour des démonstrations.
Où peut-on les découvrir ?	Principalement en ligne (site e-commerce, réseaux sociaux), ou partenariats avec des boutiques physiques à l'avenir.
Qu'est-ce qu'une skincare?	Une routine de soins pour la peau visant à nettoyer, hydrater, protéger ou traiter les problèmes cutanés.



Inspirations

Exemples d'inspiration sur le même thème



Plan du site

Espace non connecté

Page d'accueil

Commande

Liste des produits par catégorie

Page produits

Page produit

Enregistrement

Paiement

Connexion

Politique de confidentialité

Mentions légales

4. Produit Minimum Viable (MVP)

Mon Produit Minimum Viable se concentre sur l'expérience d'achat essentielle et la gestion des produits, avec une interface intuitive pour les clients et un back office simplifié pour l'administrateur.

Fonctionnalités de base incluses dans le MVP :

- **Authentification**
 - Inscription et connexion pour les utilisateurs (clients et administrateurs)
 - Option Guest checkout permettant d'acheter sans créer de compte
- **Pages légales**
 - Accès aux mentions légales, aux conditions générales de vente (CGV), et à la politique de confidentialité

A. Front-office (Client)

- **Page d'accueil**
 - Affichage des catégories parentes (ex. : *Soins visage, Corps*) et des sous-catégories (ex. : *Crèmes hydratantes, Gommages*)
- **Fiche produit**
 - Affichage clair du prix et bouton "Ajouter au panier"
- **Panier et Checkout**
 - Résumé des articles, possibilité de modifier les quantités
 - Paiement sécurisé via Stripe, sans obligation de créer un compte

B. Back-office (Admin)

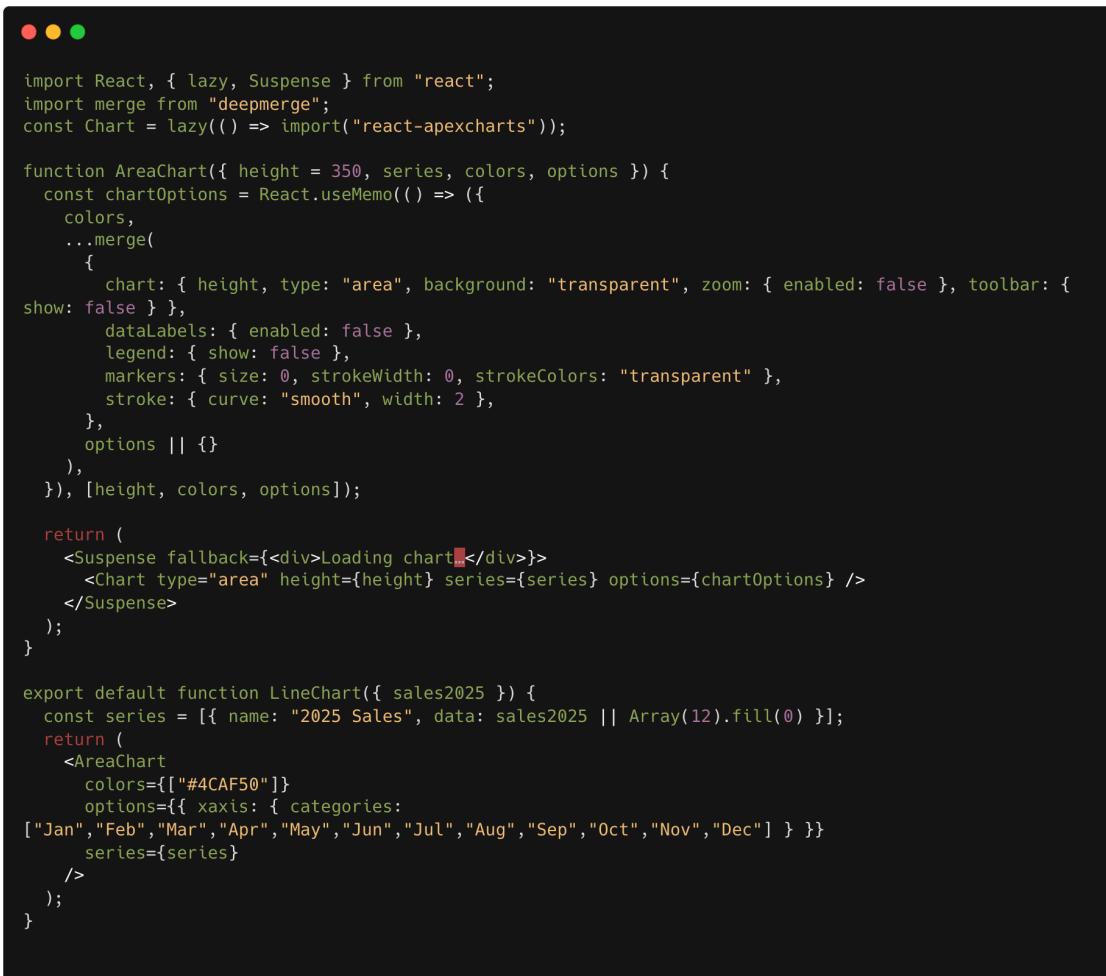
- **Gestion des produits**
 - Ajout et modification des produits, catégories et stocks via une interface Laravel
 - Suivi des commandes avec gestion des statuts (*en attente, expédiée*)
- **Dashboard**
 - Vue du chiffre d'affaires et des produits vendus

Dashboard

Le tableau de bord présente une courbe des ventes mensuelles (année en cours) et une table des commandes. Les montants agrégés (commandes au statut paid) sont calculés côté Laravel, convertis en euros et transmis à React via Inertia.

Figure 1 — LineChart « Ventes mensuelles »

Somme des montants des commandes payées, agrégée par mois (année en cours). Unité : euros.



```
import React, { lazy, Suspense } from "react";
import merge from "deepmerge";
const Chart = lazy(() => import("react-apexcharts"));

function AreaChart({ height = 350, series, colors, options }) {
  const chartOptions = React.useMemo(() => ({
    colors,
    ...merge(
      {
        chart: { height, type: "area", background: "transparent", zoom: { enabled: false }, toolbar: { show: false } },
        dataLabels: { enabled: false },
        legend: { show: false },
        markers: { size: 0, strokeWidth: 0, strokeColors: "transparent" },
        stroke: { curve: "smooth", width: 2 },
      },
      options || {}
    ),
  ), [height, colors, options]);
  return (
    <Suspense fallback={<div>Loading chart</div>}>
      <Chart type="area" height={height} series={series} options={chartOptions} />
    </Suspense>
  );
}

export default function LineChart({ sales2025 }) {
  const series = [{ name: "2025 Sales", data: sales2025 || Array(12).fill(0) }];
  return (
    <AreaChart
      colors={[ "#4CAF50" ]}
      options={{ xaxis: { categories: ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"] } }}
      series={series}
    />
  );
}
```

Figure 2 — OrderTable « Commandes »

Liste paginée des commandes avec : ID, client, montant (€), statut (pending/paid/canceled), date de création. Fonctions : tri, filtre par statut, pagination.

```
'use client';
import React, { useMemo, useState } from 'react';

export default function OrderTable({ orders = [] }) {
  const orderList = Array.isArray(orders) ? orders : (orders.orders || []);

  const [currentPage, setCurrentPage] = useState(1);
  const itemsPerPage = 15;
  const totalItems = orderList.length;
  const indexOfLastItem = currentPage * itemsPerPage;
  const indexOfFirstItem = indexOfLastItem - itemsPerPage;
  const currentOrders = orderList.slice(indexOfFirstItem, indexOfLastItem);

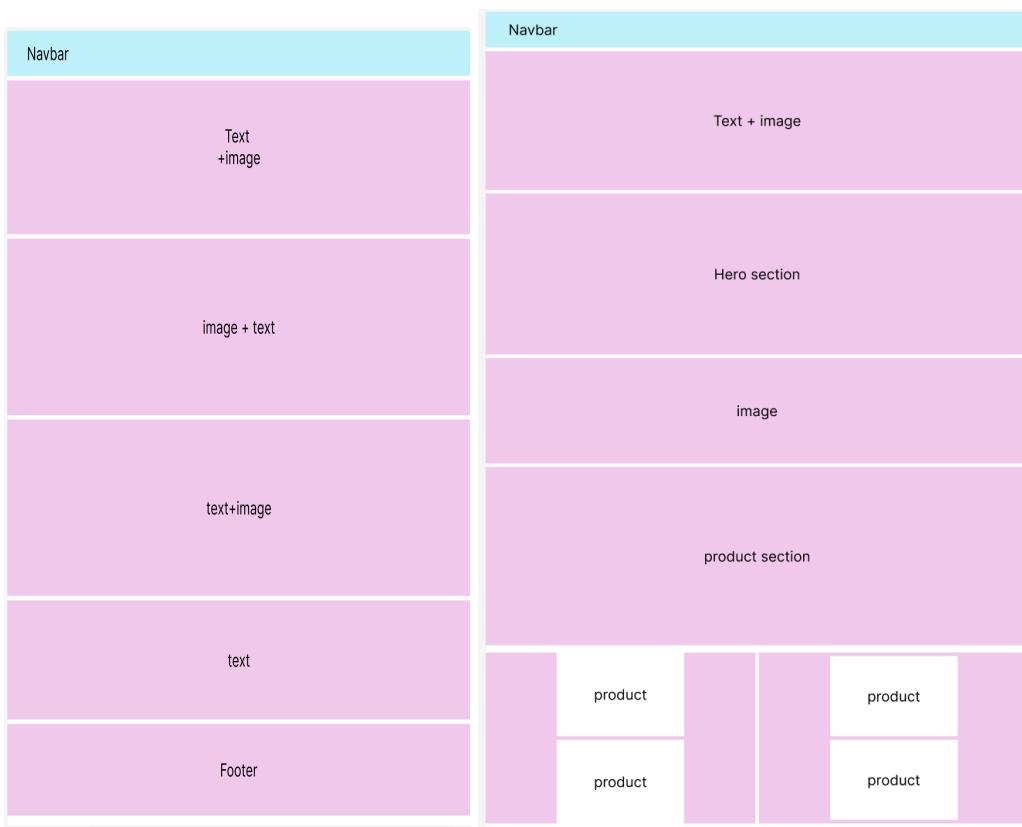
  return (
    <>
      <table className="min-w-full table-fixed divide-y divide-gray-300">
        <thead>
          <tr>
            <th className="px-3 py-3.5 text-left text-sm font-semibold">ID</th>
            <th className="px-3 py-3.5 text-left text-sm font-semibold">Client</th>
            <th className="px-3 py-3.5 text-left text-sm font-semibold">Montant (€)</th>
            <th className="px-3 py-3.5 text-left text-sm font-semibold">Statut</th>
            <th className="px-3 py-3.5 text-left text-sm font-semibold">Créée le</th>
          </tr>
        </thead>
        <tbody className="divide-y divide-gray-200 bg-white">
          {currentOrders.map(o => (
            <tr key={o.id}>
              <td className="px-3 py-4 text-sm text-gray-600">{o.id}</td>
              <td className="px-3 py-4 text-sm text-gray-600">{o.client?.name || o.customer_name || "-"}</td>
              <td className="px-3 py-4 text-sm text-gray-600">{o.amount}</td>
              <td className="px-3 py-4 text-sm text-gray-600">{o.status}</td>
              <td className="px-3 py-4 text-sm text-gray-600">{o.created_at}</td>
            </tr>
          ))}
        </tbody>
      </table>

      <nav className="mt-4 flex items-center gap-3" aria-label="Pagination">
        <button onClick={() => setCurrentPage(p => Math.max(p - 1, 1))} disabled={currentPage === 1}>
          Previous
        </button>
        <span>
          {totalItems ? indexOfFirstItem + 1 : 0}-{Math.min(indexOfLastItem, totalItems)} / {totalItems}
        </span>
        <button
          onClick={() => setCurrentPage(p => (indexOfLastItem >= totalItems ? p : p + 1))} disabled={indexOfLastItem >= totalItems}
        >
          Next
        </button>
      </nav>
    </>
  );
}
```

Mise en place de l'application

Zoning

Le zoning est une méthode d'organisation de l'espace qui consiste à diviser une surface en différentes zones selon leur fonction, leur usage. Il permet de structurer clairement un projet en attribuant à chaque partie un rôle précis. Dans ce projet, le zoning a été réalisé section par section, afin d'assurer une organisation rigoureuse et cohérente de l'ensemble du site e-commerce.



Wireframes



Chartes graphique et logo

Pour ce projet de site e-commerce dédié aux produits d'hygiène et de soin de la peau, j'ai choisi une direction graphique douce, naturelle et rassurante. L'objectif était de refléter des valeurs de bien-être, de pureté et de confiance, tout en proposant une expérience claire et agréable pour l'utilisateur.

La palette de couleurs s'articule autour de tons beiges et bruns, évoquant les matières naturelles, la douceur des soins et l'authenticité. La couleur d'accent principale, #E7DED8, apporte de la lumière et contribue à créer une ambiance épurée, proche de l'univers cosmétique.

Avant de passer à la conception, j'ai réalisé un moodboard réunissant des visuels inspirants (textures naturelles, ambiance spa, tons neutres, packaging sobre), ce qui m'a permis de guider mes choix graphiques dès les premières étapes du projet.



Identité visuelle : couleurs et typographie

Dans le cadre de ce site e-commerce spécialisé dans les produits pour la peau, l'identité visuelle a été pensée pour inspirer confiance, douceur et naturel, tout en mettant en valeur la qualité des produits proposés.

Choix des couleurs

La palette de couleurs a été soigneusement sélectionnée pour évoquer la pureté, la naturalité et le bien-être :

Les teintes naturelles, telles que les beiges doux et le marron profond, ont été choisies pour créer une ambiance chaleureuse, apaisante et organique. Elles évoquent à la fois les couleurs de la peau, les ingrédients bruts et les éléments naturels comme la terre, renforçant ainsi l'identité d'un site ancré dans le bien-être, la nature et l'authenticité des soins proposés.

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

La typographie mêle structure et sensibilité pour refléter l'univers du soin et de la beauté. Les titres en gras assurent une hiérarchie claire et une lecture fluide, tandis que l'usage ponctuel d'une écriture cursive apporte une touche douce, élégante et humaine, évoquant l'artisanat, la délicatesse des gestes et la proximité avec les clients.

Ce contraste crée un équilibre visuel qui renforce l'identité chaleureuse et personnalisée du site.

Typography
Font Family: Inter

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

Colors

Les images utilisées sur le site ont également été soigneusement sélectionnées pour s'harmoniser avec la palette de couleurs. Qu'il s'agisse des tons de peau, des textures naturelles ou des décors neutres et apaisants, chaque visuel vient renforcer la cohérence esthétique du site.

Images

Images



Spécifications techniques

Gestion de version

Pour assurer un suivi clair et structuré du développement, j'ai mis en place une gestion de version simple et efficace basée sur Git.

Le projet repose sur deux branches principales :

- **main** : branche stable utilisée pour les versions prêtes à être mises en production.
- **dev** : branche de développement sur laquelle je code exclusivement. Toutes les fonctionnalités y sont d'abord développées et testées avant d'être intégrées à la version finale.

Les commits sont effectués régulièrement, dès qu'une fonctionnalité est terminée et fonctionnelle. Les messages sont toujours rédigés en français et de manière descriptive,

The screenshot shows a GitHub commit history for the 'dev' branch. At the top, there are filters for 'All users' and 'All time'. Below the header, a dropdown shows 'dev'. The commits are listed in chronological order from June 16, 2025, at the top to June 11, 2025, at the bottom. Each commit includes a message, the author (YaninaRZ), the date (committed last week), the commit hash, and copy/paste icons.

Commit Message	Author	Date	Hash
affichage des données du back dashboard	YaninaRZ	committed last week	6b6f1e3
affichage de la quantité ventes dashboard	YaninaRZ	committed last week	e9c4219
update des noms des routes	YaninaRZ	committed last week	c48c5a0
breadcrumbs mis en place	YaninaRZ	committed last week	2b65dba
fix de la nav mobile	YaninaRZ	committed last week	9bfdd36
affichage quantité des produits commandés	YaninaRZ	committed last week	8f69eaf
stripe payment mise en place en cours	YaninaRZ	committed last week	caab190

Une fois le travail réalisé sur la branche dev terminé et validé, j'ouvre une Pull Request (PR) pour fusionner les modifications vers la branche main.

Cette étape intermédiaire permet de :

- vérifier le code avant intégration (revue et tests manuels ou automatisés)
- garder une trace claire de toutes les évolutions majeures
- éviter les erreurs ou conflits en validant les changements avant leur fusion

Après validation de la pull request, les modifications sont mergées dans main, garantissant ainsi une version stable et à jour du projet.

Ce mode de fonctionnement présente plusieurs avantages :

- Il reste simple à gérer, sans multiplier inutilement les branches
- Il améliore la qualité du code grâce au contrôle préalable des PR
- Il assure la stabilité et la traçabilité du projet tout au long du développement

Choix techniques

Le choix des technologies utilisées s'est fait dans une logique de productivité, de maintenabilité et de simplicité d'intégration entre le front-end et le back-end. L'objectif était de construire un site e-commerce fluide, moderne et bien structuré, sans complexifier inutilement le développement.

Front-End

Pour la partie front-end, j'ai opté pour Inertia.js associé à React. Ce duo me permet de créer une expérience type SPA via Inertia.js, avec une navigation fluide et sans recharge, tout en continuant à bénéficier de la puissance de Laravel côté serveur. Ce choix m'a évité d'avoir à développer une API REST séparée, ce qui a simplifié l'architecture du projet.

Le design est géré avec Tailwind CSS, un framework basé sur des classes utilitaires. J'ai préféré Tailwind à Bootstrap ou SASS pour plusieurs raisons :

- Il permet une intégration rapide grâce à l'usage direct des classes (px-4, mt-6, etc.)
- Il offre une grande flexibilité de personnalisation via le fichier app.css depuis la nouvelle version de Tailwind notamment dans le cadre de l'intégration avec Inertia
- La responsivité est simple à mettre en place grâce aux breakpoints intégrés (md :, lg :, etc.)

J'ai fait en sorte de ne pas surcharger le CSS : au lieu de réécrire beaucoup de styles, j'ai adapté les composants Tailwind pour correspondre au design défini en amont.

Back-End

Pour la partie serveur, j'ai choisi Laravel, un framework PHP robuste, moderne et bien documenté. Ce choix s'est imposé face à Symfony notamment pour les raisons suivantes:

- Son ORM Eloquent facilite énormément la gestion des relations entre entités, notamment entre les produits et leurs catégories
- Son écosystème riche m'a permis de gagner du temps, avec des outils intégrés comme :

- Le Starter Kit Laravel + React, qui offre une base fonctionnelle avec front intégré dès le départ

Enfin, j'ai utilisé les migrations et les seeders pour structurer et remplir facilement ma base de données, notamment pour la création des premières catégories et produits.

ACCESSIBILITÉ

Types d'appareils compatibles

Le site e-commerce Skinn offre une expérience utilisateur optimale sur tous les types d'écrans, du smartphone le plus compact jusqu'aux très grands moniteurs en 4K.

L'objectif est clair : permettre à chaque visiteur, peu importe son appareil, de naviguer confortablement et efficacement.

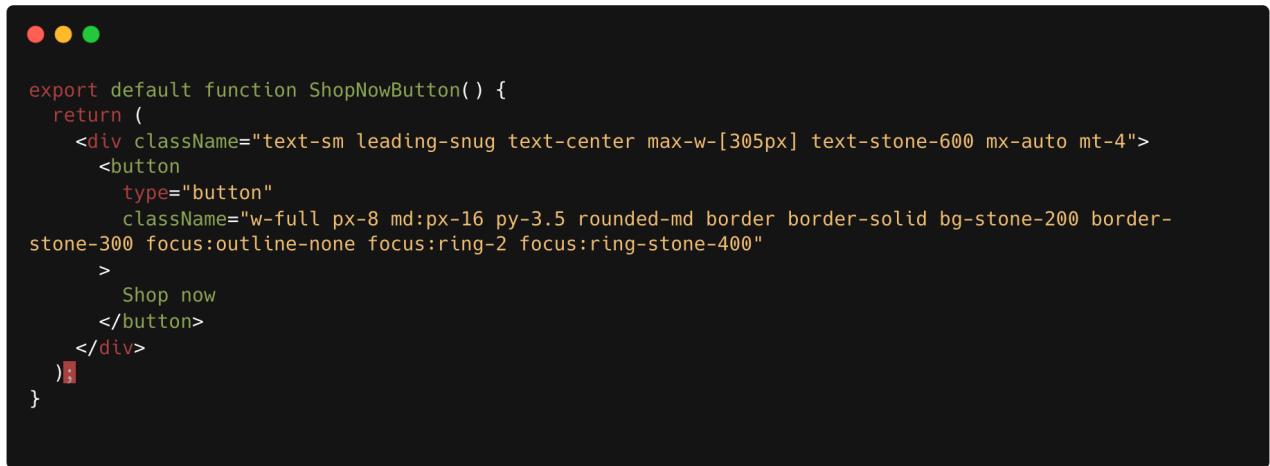
Une stratégie responsive maîtrisée

J'ai utilisé le framework CSS Tailwind, qui facilite la gestion des points de rupture via des classes utilitaires. Par exemple, la classe md : hidden cache un élément sur les écrans d'une largeur minimale de 768 pixels, ce qui correspond en CSS natif à la règle @media (min-width : 768px) { display : none; }. Cette méthode permet d'adapter précisément la présentation sur :

- **Smartphones (320px à 767px)** : navigation simplifiée avec un menu réduit et des zones tactiles élargies pour une ergonomie mobile améliorée
- **Tablettes (768px à 1023px)** : interface équilibrée avec des grilles flexibles et une typographie optimisée pour une lecture confortable
- **Ordinateurs (1024px et plus)** : mise en page complète, sections aérées et contenu hiérarchisé pour une expérience riche et claire

Le site respecte les bonnes pratiques d'accessibilité selon les recommandations WAI et WCAG. J'ai utilisé des balises sémantiques HTML5 telles que `<header>`, `<nav>`, `<main>`, et `<footer>` pour structurer le contenu de manière claire, facilitant ainsi la navigation aux lecteurs d'écran et améliorant le référencement naturel.

De plus, des attributs ARIA ont été intégrés pour décrire les composants interactifs complexes, garantissant que le site reste utilisable via le clavier et accessible aux personnes en situation de handicap.



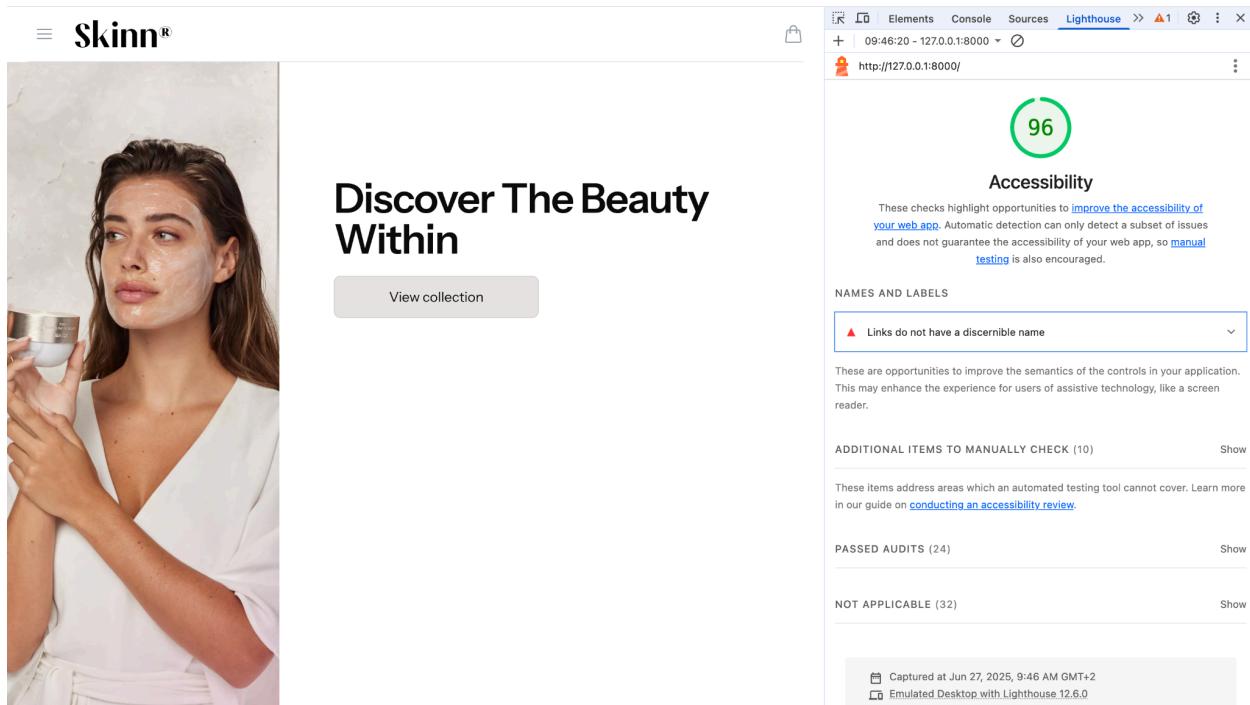
```

export default function ShopNowButton() {
  return (
    <div className="text-sm leading-snug text-center max-w-[305px] text-stone-600 mx-auto mt-4">
      <button type="button" className="w-full px-8 md:px-16 py-3.5 rounded-md border border-solid bg-stone-200 border-stone-300 focus:outline-none focus:ring-2 focus:ring-stone-400">
        >
        Shop now
      </button>
    </div>
  )
}

```

Contrôle et validation

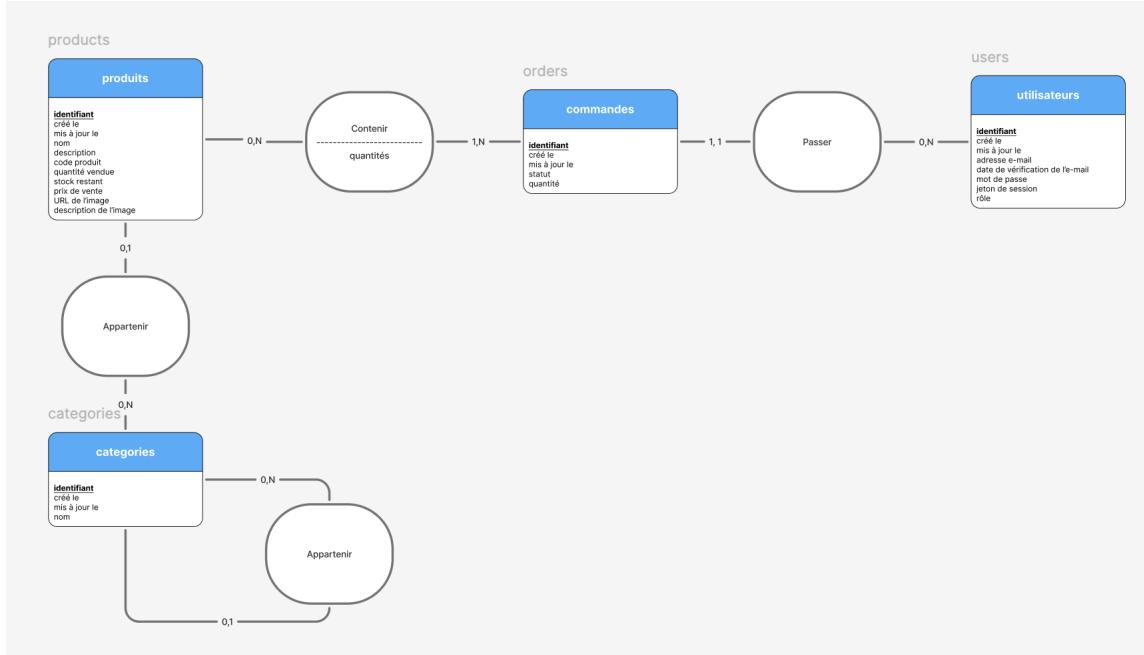
Enfin, j'ai validé l'accessibilité du site à l'aide de l'outil Google Lighthouse, intégré aux outils de développement des navigateurs modernes. Cet audit a permis d'identifier et de corriger les éventuels problèmes, assurant une conformité élevée aux standards d'accessibilité.



The screenshot shows the Skinn website homepage on the left, featuring a woman applying cream to her face. The headline reads "Discover The Beauty Within". On the right, the Google Lighthouse audit results are displayed. The overall score is 96. The "Accessibility" section shows a green circle with the number 96. Below it, a note says: "These checks highlight opportunities to [improve the accessibility of your web app](#). Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so [manual testing](#) is also encouraged." Other sections include "Names and Labels" (warning: "Links do not have a discernible name"), "Additional items to manually check (10)", "Passed audits (24)", and "Not applicable (32)". A footer note at the bottom right of the audit report states: "Captured at Jun 27, 2025, 9:46 AM GMT+2 Emulated Desktop with Lighthouse 12.6.0".

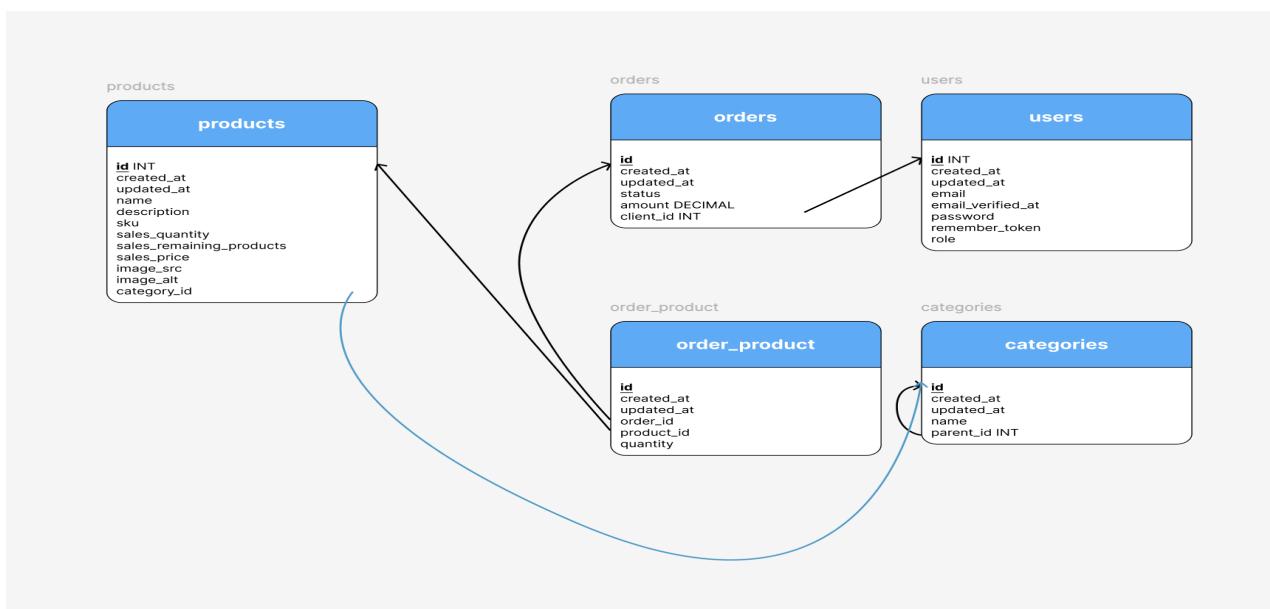
Architecture du projet

MCD



Ce MCD servira de base pour la conception de la base de données relationnelle qui implémentera ce modèle. La prochaine étape serait de convertir ce modèle conceptuel en modèle logique (MLD) avec des tables précises, clés primaires et étrangères.

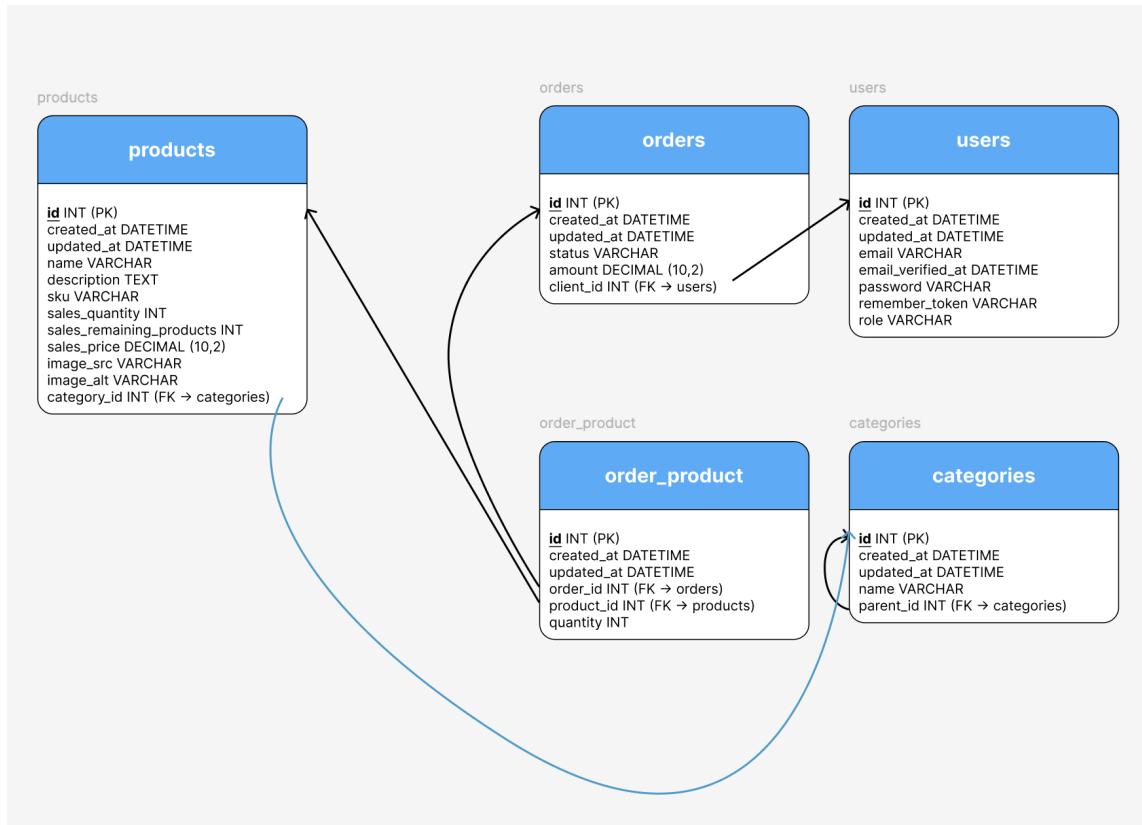
MLD



Le Modèle Logique de Données (MLD) présenté ici a été généré à partir du MCD précédemment analysé. Il représente la structure concrète de la base de données relationnelle avec les tables, leurs champs et les relations entre elles.

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

MPD



Ce MPD est l'aboutissement de la transformation du MCD (Modèle Conceptuel de Données) en MLD (Modèle Logique de Données), puis en structure physique optimisée pour un SGBD (système de gestion de base de données) relationnel. J'ai également intégré les types de données appropriés (VARCHAR, INT, DATETIME, DECIMAL) et les clés primaires/étrangères afin d'assurer l'intégrité référentielle.

Pour passer du MCD/MLD au Modèle Physique de Données (MPD), les relations ont été matérialisées en MySQL via des migrations Laravel et exposées dans les modèles Eloquent. Les extraits ci-dessous suffisent à prouver les cardinalités (1–N, N–N) et l'existence des tables pivots avec attributs métier.

Relations clés implémentées

- User (1) — (N) Order via orders.client_id : un client peut avoir plusieurs commandes.
- Order (N) — (N) Product via le pivot order_product (avec quantity, unit_price) : une commande contient plusieurs produits et inversement.
Category (1) — (N) Product et Category (1) — (N) Category (hiérarchie) : catégories parents/enfants + rattachement des produits.

Eloquent (extraits)

```
User::class  ->hasMany(Order::class, 'client_id');
Order::class -> belongsTo(User::class,
'client_id');
```

```
Category::class -> belongsTo(Category::class, 'parent_id');
Category::class ->hasMany(Category::class, 'parent_id');
Product::class  -> belongsTo(Category::class);
```

```
Order::class  -> belongsToMany(Product::class)->withPivot(['quantity','unit_price'])->withTimestamps();
Product::class -> belongsToMany(Order::class)->withPivot(['quantity','unit_price'])->withTimestamps();
```

Pivot (migration – extrait)

```

Schema::create('order_product', function (Blueprint $t) {
    $t->id();
    $t->foreignId('order_id')->constrained()->cascadeOnDelete();
    $t->foreignId('product_id')->constrained()->restrictOnDelete();
    $t->unsignedInteger('quantity')->default(1);
    $t->unsignedInteger('unit_price'); // centimes
    $t->timestamps();
    $t->unique(['order_id','product_id']); // évite les doublons dans une même
});  
mande

```

Ces éléments montrent la cohérence MCD → MLD → MPD : clés étrangères, contraintes d'unicité, pivot N-N avec attributs fonctionnels, et hiérarchie de catégories. Les captures “migrate:status” + schema dump (CREATE TABLE) complètent la preuve d'implémentation réelle.

Pour attester que le MCD/MLD/MPD a bien été implémenté et appliqué, j'ai exécuté la commande suivante dans l'environnement Docker : docker compose exec app php artisan migrate:status

La sortie affiche le statut “Ran/Yes” pour l'ensemble des fichiers de migration (ex. : `create_order_product_table`, `add_client_id_to_orders_table`, `add_parent_id_to_categories_table`, etc.), démontrant que la structure de base de données a été créée et versionnée avec Laravel Migrations.

```

razvetka@MacBook-Air-de-Razvetskaya final-project % docker compose exec app php artisan migrate:status

Migration name ..... Batch / Status
0001_01_01_00000_create_users_table ..... [1] Ran
0001_01_01_000001_create_cache_table ..... [1] Ran
0001_01_01_000002_create_jobs_table ..... [1] Ran
2025_05_21_134950_alter_users_table ..... [1] Ran
2025_05_23_083720_create_categories_table ..... [1] Ran
2025_05_23_114311_create_products_table ..... [1] Ran
2025_05_26_121411_create_orders_table ..... [1] Ran
2025_05_27_080151_add_category_id_to_all_products_table ..... [1] Ran
2025_05_27_140430_add_client_id_to_orders_table ..... [1] Ran
2025_05_30_085114_create_order_product_table ..... [1] Ran
2025_05_30_123356_add_quantity_to_order_product_table ..... [1] Ran
2025_05_30_142641_make_amount_nullable_in_orders_table ..... [1] Ran
2025_06_05_082904_add_parent_id_to_categories_table ..... [1] Ran
2025_09_17_113210_add_slug_to_categories_table ..... [2] Ran

```

Afin de montrer la structure réelle MySQL générée par les migrations, j'ai exporté le schéma (sans données) depuis Docker et versionné le fichier `database/schema/mysql-schema.sql`. Extrait représentatif (tables `orders` et `order_product` avec clés étrangères) :

```

CREATE TABLE `orders` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  `status` enum('pending','shipping','paid','cancelled','delivered') COLLATE utf8mb4_unicode_ci NOT
NULL,
  `amount` decimal(10,2) DEFAULT NULL,
  `client_id` bigint unsigned DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `orders_client_id_foreign` (`client_id`),
  CONSTRAINT `orders_client_id_foreign` FOREIGN KEY (`client_id`) REFERENCES `users` (`id`) ON DELETE
SET NULL
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
CREATE TABLE `order_product` (
  `id` bigint unsigned NOT NULL AUTO_INCREMENT,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  `order_id` bigint unsigned NOT NULL,
  `product_id` bigint unsigned NOT NULL,
  `quantity` int NOT NULL DEFAULT '1',
  PRIMARY KEY (`id`),
  KEY `order_product_order_id_foreign` (`order_id`),
  KEY `order_product_product_id_foreign` (`product_id`),
  CONSTRAINT `order_product_order_id_foreign` FOREIGN KEY (`order_id`) REFERENCES `orders` (`id`) ON
DELETE CASCADE,
  CONSTRAINT `order_product_product_id_foreign` FOREIGN KEY (`product_id`) REFERENCES `products` (`id`)
ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

```

Le fichier mysql-schema.sql a été généré depuis Docker avec :

```

mkdir -p database/schema
docker compose exec app sh -lc '
mariadb-dump -h"${DB_HOST}" -P"${DB_PORT:-3306}" -u"${DB_USERNAME}" -p"${DB_PASSWORD}" \
--no-data --routines --single-transaction --skip-comments --skip-ssl "${DB_DATABASE}"
' | tee database/schema/mysql-schema.sql > /dev/null

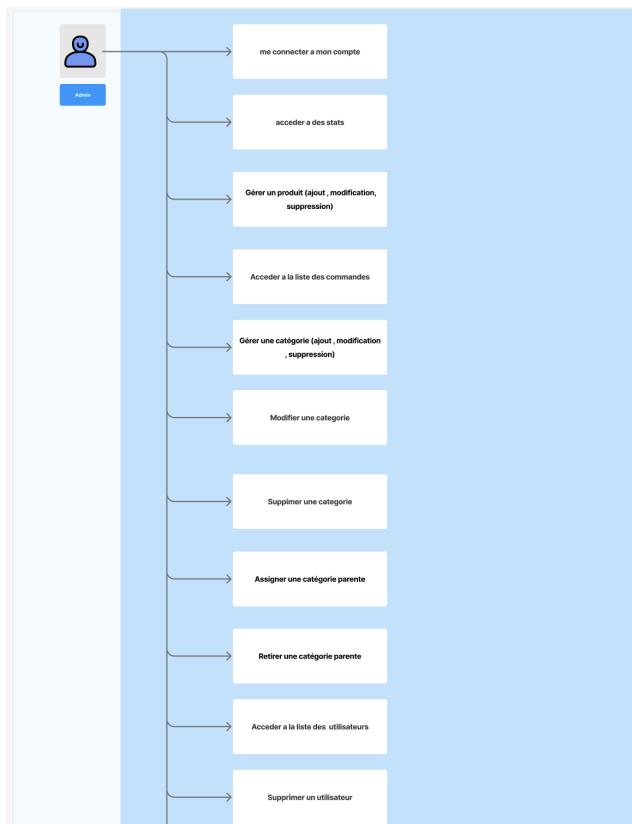
```

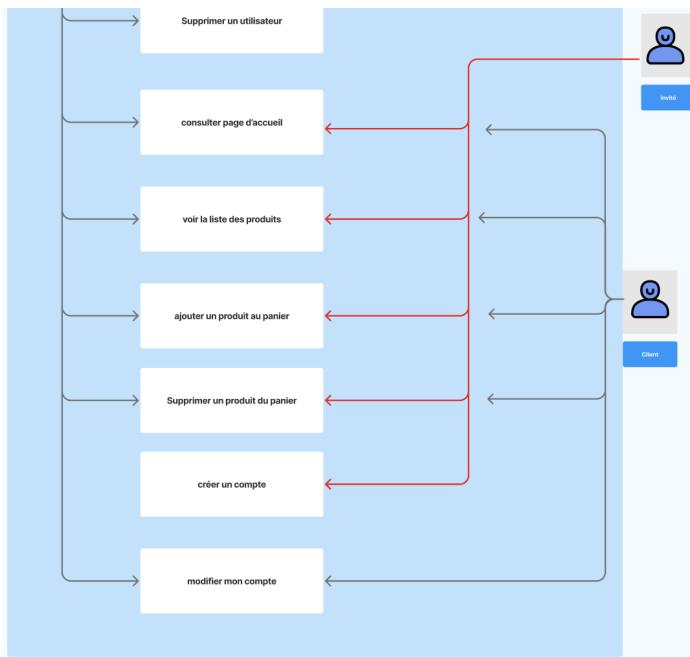
Use case

Ce cas d'utilisation décrit les principales interactions des utilisateurs avec le système, à savoir les clients et l'administrateur. Chaque acteur dispose d'un ensemble de fonctionnalités spécifiques permettant une gestion fluide et intuitive du site.

Acteurs

- **Client invité** : utilisateur qui visite le site sans être connecté
- **Client enregistré** : utilisateur disposant d'un compte
- **Administrateur** : utilisateur chargé de la gestion globale du site





Utilisation des Seeders pour l'insertion des données

Dans le cadre de mon projet, j'ai utilisé des seeders afin d'insérer automatiquement des données de test dans la base de données. Cela permet de peupler rapidement les tables avec des informations réalistes, facilitant ainsi le développement et les tests fonctionnels.

```

<?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],
        ]);
    }
}

```

Dans cet exemple, une commande est créée pour un client spécifique, puis plusieurs produits lui sont associés avec des quantités précises. Cette méthode garantit une base de données cohérente et prête à l'usage dès l'installation du projet.

Documentation des Routes - Projet Skinn

Non connecté

Méthode	Route	Description
GET	/	Page d'accueil
GET	/our-collection	Notre collection
GET	/about	À propos
GET	/contact	Contact
GET	/terms-of-services	Conditions d'utilisation
GET	/privacy-policy	Politique de confidentialité
GET	/licence	Licence

Cart

Méthode	Route	Description
GET	/cart	Voir le panier
POST	/cart/add	Ajouter au panier
POST	/cart/remove	Retirer du panier
POST	/cart/clear	Vider le panier

Commandes

Méthode	Route	Description
POST	/orders-store	Créer une commande
GET	/client-orders	Voir mes commandes
GET	/merci	Page merci

Stripe / Checkout

Méthode	Route	Description
POST	/checkout	Créer un paiement Stripe
GET	/checkout/success	Paiement réussi
GET	/checkout/cancel	Paiement annulé

Client

Méthode	Route	Description
GET	/user-account	Mon compte
GET	/user-password	Changer mot de passe
GET	/user-billing	Facturation client
GET	/client/view-order/{id}	Voir commande client

Produits

Méthode	Route	Description
GET	/products/category/{category}	Lister produits d'une catégorie
GET	/products	Lister tous les produits (client)
GET	/products/{id}	Voir un produit par ID
GET	/post-detail/{id}	Voir un post détail

Sécurité

Sécurisation des routes

Dans le développement de cette application Laravel, j'ai accordé une attention particulière à la sécurisation des routes. Les routes définissent les points d'entrée de l'application et permettent aux utilisateurs d'accéder à diverses fonctionnalités.

Certaines routes sont accessibles librement à tous les utilisateurs, même non connectés. Il s'agit principalement des pages d'accueil, de présentation, de contact ou encore de navigation parmi les produits. Ces routes ne contiennent aucune donnée sensible ni opération critique, ce qui justifie l'absence de restriction d'accès.

D'autres routes donnent accès à des opérations sensibles : création de commandes, gestion de produits, affichage de factures ou données personnelles. Pour ces routes, j'ai mis en place plusieurs niveaux de protection :

- Authentification (auth) : permet de restreindre l'accès uniquement aux utilisateurs connectés

- Vérification d'adresse e-mail (verified) : ajoute une sécurité supplémentaire en exigeant une adresse e-mail vérifiée
- Gestion des rôles : L'application utilise Spatie Laravel-Permission pour gérer les rôles et les permissions. Les routes sensibles sont protégées par le middleware role :admin ou role :client

Sécurisation et protection CSRF

Laravel gère automatiquement la protection CSRF en utilisant un cookie XSRF-TOKEN envoyé avec chaque réponse. Ce cookie est automatiquement lu par Axios, la bibliothèque HTTP utilisée par Inertia.js, qui l'ajoute dans les en-têtes (X-XSRF-TOKEN) lors des requêtes POST, PUT, PATCH ou DELETE. Cela permet à Laravel de vérifier la légitimité de chaque requête sans configuration supplémentaire.

Le middleware HandleInertiaRequests, installé par défaut avec Inertia, permet aussi d'ajouter manuellement le jeton CSRF aux props partagées si nécessaire. Toutefois, dans la majorité des cas, cette étape est inutile puisque le mécanisme basé sur le cookie assure déjà une protection fiable et transparente.

```
const { data, setData, post, processing, errors, reset } =
useForm({
  email: '',
  password: '',
  remember: false,
});

const submit = (e) => {
  e.preventDefault();
  post(route('login'), {
    onFinish: () => reset('password'),
  });
};
```

Le formulaire de connexion utilise la méthode useForm fournie par Inertia.js, qui repose sur Axios. Axios lit automatiquement le cookie XSRF-TOKEN généré par Laravel, garantissant la protection CSRF sans configuration supplémentaire. Le token est envoyé dans l'en-tête X-XSRF-TOKEN pour chaque requête POST, ce qui permet à Laravel de valider la légitimité de la requête côté serveur.

Contrôle d'unicité de l'adresse e-mail pour sécuriser l'inscription

Dans une application Laravel utilisant Inertia.js, la validation des données côté serveur est essentielle pour garantir la cohérence des informations, notamment pour éviter qu'un utilisateur soit créé plusieurs fois avec la même adresse email.

Dans le contrôleur UserController, la méthode store (qui gère la création d'un nouvel utilisateur) utilise la fonction validate() de Laravel. Cette fonction applique plusieurs règles sur les données reçues, dont la règle unique sur le champ d'adresse e-mail.

Cette règle vérifie dans la base de données que l'adresse email soumise n'existe pas déjà pour un autre utilisateur. Si un doublon est détecté, Laravel renvoie automatiquement une erreur de validation. Inertia.js, qui fait la passerelle entre le backend Laravel et le frontend React, transmet cette erreur au composant React. Ainsi, le formulaire peut afficher un message d'erreur précis, demandant à l'utilisateur de corriger sa saisie.

```
public function store(Request $request)
{
    $request->validate([
        'name' => 'required|string|max:255',
        'email' => 'required|string|lowercase|email|max:255|unique:' .
User::class.'password' => 'string',
    ]);

    $user = User::create([
        'name' => $request->name,
        'email' => $request->email,
        'password' => Hash::make($request->password),
    ]);

    return to_route('client');
}
```

CORS

En parallèle, lorsqu'une application Laravel expose une API destinée à être consommée par une application front-end, la configuration du CORS (Cross-Origin Resource Sharing) devient indispensable. CORS est un mécanisme de sécurité des navigateurs qui contrôle

les échanges entre des domaines différents. Par défaut, un navigateur bloque les requêtes effectuées depuis un domaine vers un autre si le serveur cible ne l'autorise pas explicitement. Laravel gère cela via un fichier de configuration config/cors.php, où l'on peut définir les origines autorisées, les types de requêtes acceptées, et si les cookies ou en-têtes doivent être partagés.

Tests dans un projet Laravel

Test unitaires

Laravel propose une structure organisée et claire pour la gestion des tests, ce qui facilite le développement d'applications robustes et fiables. À la racine de chaque projet Laravel, on trouve un dossier tests, divisé principalement en deux sous-répertoires : Unit et Feature. Cette séparation permet de distinguer les types de tests selon leur portée et leur objectif.

Les tests unitaires, situés dans le dossier tests/Unit, ont pour but de tester des composants isolés de l'application, comme des classes, des méthodes ou des fonctions. Ils sont exécutés sans démarrer le noyau de Laravel, ce qui signifie qu'ils n'ont pas accès à la base de données ni aux services fournis par le framework. Cela permet des tests très rapides, axés sur la logique interne du code. L'objectif principal des tests unitaires est de s'assurer que chaque unité de code fonctionne correctement de manière indépendante, sans dépendance à l'environnement global de l'application.

Dans ce projet, l'exécution d'un des tests unitaires a révélé que 8 suites de tests passent complètement, tandis que 2 suites rencontrent des échecs partiels.

```
razvetka@MacBook-Air-de-Razvetskaya skinn2 % php artisan test

  PASS Tests\Unit\ExampleTest
    ✓ that true is true

  FAIL Tests\Feature\Auth\AuthenticationTest
    ✓ login screen can be rendered
    ✘ users can authenticate using the login screen
    ✓ users can not authenticate with invalid password
    ✓ users can logout

  PASS Tests\Feature\Auth\EmailVerificationTest
    ✓ email verification screen can be rendered
    ✓ email can be verified
    ✓ email is not verified with invalid hash

  PASS Tests\Feature\Auth>PasswordConfirmationTest
    ✓ confirm password screen can be rendered
    ✓ password can be confirmed
    ✓ password is not confirmed with invalid password

  PASS Tests\Feature\Auth>PasswordResetTest
    ✓ reset password link screen can be rendered
    ✓ reset password link can be requested
    ✓ reset password screen can be rendered
    ✓ password can be reset with valid token

  PASS Tests\Feature\Auth\RegistrationTest
    ✓ registration screen can be rendered
    ✓ new users can register

  FAIL Tests\Feature\DashboardTest
    ✘ guests are redirected to the login page
    ✘ authenticated users can visit the dashboard

  PASS Tests\Feature\RegisterUserTest
    ✓ user can be created
    ✓ email must be unique

  PASS Tests\Feature\Settings\PasswordUpdateTest
    ✓ password can be updated
    ✓ correct password must be provided to update password

  PASS Tests\Feature\Settings\ProfileUpdateTest
    ✓ profile page is displayed
    ✓ profile information can be updated
    ✓ email verification status is unchanged when the email address is unchanged
    ✓ user can delete their account
    ✓ correct password must be provided to delete account
```

Détail des Échecs

La première suite en difficulté est *Tests\Feature\Auth\AuthenticationTest*. Elle échoue sur un test spécifique qui vérifie que les utilisateurs peuvent s'authentifier via l'écran de login. Cependant, les autres tests de cette suite - comme le rendu de la page de connexion, la gestion d'un mot de passe invalide, ou la déconnexion - réussissent correctement.

La deuxième suite problématique est *Tests\Feature\DashboardTest*, qui présente deux échecs : les invités ne sont pas redirigés vers la page de connexion comme attendu, et les utilisateurs authentifiés ne peuvent pas accéder au tableau de bord.

Test fonctionnels

À l'inverse, les tests fonctionnels (ou tests de fonctionnalités), regroupés dans le dossier `tests/Feature`, interagissent pleinement avec l'application Laravel. Ces tests permettent de vérifier le comportement de l'application dans un contexte réaliste, en testant les

routes, les contrôleurs, les requêtes HTTP, les middlewares, ainsi que les interactions avec la base de données.

Ils sont utiles pour s'assurer que les différentes couches de l'application (routes, contrôleurs, modèles, etc.) collaborent correctement. Par exemple, un test fonctionnel peut vérifier qu'un utilisateur peut s'inscrire via un formulaire, que les données sont bien enregistrées en base, et que la redirection vers la page d'accueil fonctionne comme prévu.

Cahier de tests

Afin de structurer et formaliser la phase de validation, un cahier de tests a été mis en place. Il permet de suivre l'ensemble des fonctionnalités testées, en définissant pour chaque scénario les préconditions, les étapes à suivre, les données de test, les résultats attendus et le statut final. Ce document est un support essentiel pour s'assurer que le projet respecte les exigences fonctionnelles et techniques définies en amont.

Ce cahier de tests facilite également la traçabilité des vérifications effectuées, tout en offrant un cadre clair pour les phases de test de non-régression. Il a servi de base pour valider les développements avant mise en production, en garantissant la cohérence et la fiabilité de l'application.

ID Cas de Test	Scénario	Description	Pré-condition	Étapes de Test	Données de Test	Résultat Attendu	Post-condition	Statut
TC_AUTH_001	Inscription utilisateur	Inscription avec données valides	Application Laravel installée avec Starter Kit	1. Accéder à register2. Remplir formulaire3. Cliquer sur "S'inscrire"	Email: "test@example.com"Mot de passe: "SecurePassword123"	Compte créé, redirection vers dashboard	Utilisateur connecté	PASS
TC_AUTH_002	Connexion utilisateur	Connexion avec identifiants valides	Utilisateur déjà enregistré	1. Accéder à login2. Entrer email et mot de passe3. Cliquer sur "Se connecter"	Email: "test@example.com"Mot de passe: "SecurePassword123"	Connexion réussie, redirection vers dashboard	Utilisateur connecté	PASS
TC_AUTH_003	Connexion utilisateur	Connexion avec mot de passe invalide	Utilisateur déjà enregistré	1. Accéder à login2. Entrer email valide3. Entrer mot de passe invalidé. Cliquer sur "Se connecter"	Email: "test@example.com"Mot de passe: "wrongpass"	Message d'erreur "Les identifiants ne correspondent pas à nos enregistrements"	Utilisateur connecté	PASS
TC_AUTH_004	Réinitialisation de mot de passe	Demande de réinitialisation	Utilisateur déjà enregistré	1. Accéder à forgot-password2. Entrer email sur "Envoyer le lien"	Email: "test@example.com"	Email envoyé avec lienMessage de confirmation attaché	Lien reçu dans le boîtier mail	FAIL
TC_AUTH_005	Connexion utilisateur	Affichage page de connexion	Application Laravel installée	1. Accéder à login		Page de connexion affichée avec champ email/mot de passe		PASS
TC_AUTH_006	Connexion client	Connexion avec rôle client existant	Utilisateur avec rôle client existant	1. Accéder à login2. Entrer identifiants valides. Soumettre la formulaire	Email: client@example.comMot de passe: validPassword123	Redirection vers la page home*	Session authentifiée	PASS
TC_AUTH_007	Connexion admin	Connexion avec rôle admin existant	Utilisateur avec rôle admin existant	1. Accéder à login2. Entrer identifiants admin3. Soumettre la formulaire	Email: admin@example.comMot de passe: validPassword123	Redirection vers le "dashboard"	Session authentifiée	PASS
TC_AUTH_008	Déconnexion	Terminer une session	Utilisateur authentifié	1. Cliquer sur bouton déconnexion		Session terminéeRedirection vers page d'accueil	Session invalidée	PASS
TC_AUTH_009	Confirmation mot de passe	Affichage page confirmation	Utilisateur authentifié	1. Accéder à confirm-password		Page de confirmation mot de passe affichée		PASS
TC_AUTH_010	Validation confirmation	Confirmation mot de passe réussi	Utilisateur authentifié	1. Accéder à /confirm-password	Mot de passe: validPassword123	Redirection vers dashboard	Confirmation timestamp enregistrée	PASS
TC_AUTH_011	Échec confirmation	Confirmation mot de passe échouée	Utilisateur authentifié	1. Accéder à /confirm-password2. Entrer mot de passe invalide3. Soumettre	Mot de passe: wrongPassword	Message d'erreur "auth password"	Pas de redirection	FAIL
TC_AUTH_012	Vérification email	Affichage prompt vérification	Utilisateur non vérifié	1. Accéder à une page préalable	Email non vérifié	Page de vérification email affichée		PASS
TC_AUTH_013	Envoi lien vérification	Demande nouveau lien vérification	Utilisateur non vérifié	1. Cliquer sur "Envoyer le lien"		Message "verification-link-sent" affiché	Email envoyé	PASS
TC_AUTH_014	Vérification inutile	Accès prompt quand déjà vérifié	Utilisateur vérifié	1. Accéder à /verify-email		Redirection vers dashboard		PASS
TC_AUTH_015	Demande réinitialisation	Affichage page demande lien	Application installée	1. Accéder à forgot-password		Page de demande de lien affichée		PASS
TC_AUTH_016	Envoi lien réinitialisation	Demande valide de réinitialisation	Utilisateur existant	1. Accéder à forgot-password2. Entrer email valide3. Soumettre	Email: existant@example.com	Message de confirmation affiché	Lien envoyé par email	PASS
TC_AUTH_017	Demande lien email inconnu	Demande avec email inconnu	Aucun utilisateur avec cet email	1. Accéder à forgot-password2. Entrer email inconnu3. Soumettre	Email: inconnu@example.com	Message générique affiché (pour sécurité)	Aucun email envoyé	PASS
TC_AUTH_018	Affichage formulaire reset	Affichage page réinitialisation	Lien valide reçu	1. Cliquer sur lien de reset2. Entrer nouveau mot de passe3. Soumettre	Token: valideEmail: user@example.com	Formulaire de réinitialisation affiché avec email pré-rempli		PASS
TC_AUTH_019	Réinitialisation mot de passe	Réinitialisation réussie	Lien valide reçu	1. Accéder au lien de reset2. Entrer nouveau mot de passe3. Soumettre	Nouveau mot de passe: NewSecure123!	Redirection vers login avec message de succès	Mot de passe mis à jour	PASS
TC_AUTH_020	Réinitialisation échouée	Token invalide/expiré	Lien expiré/invalide	1. Accéder au lien de reset2. Entrer nouveau mot de passe3. Soumettre	Token: invalidEmail: user@example.comMot de passe: NewSecure123!	Message d'erreur affiché	Mot de passe non modifié	FAIL
TC_AUTH_021	Inscription utilisateur	Affichage page inscription	Application installée	1. Accéder à register		Formulaire d'inscription affiché		PASS
TC_AUTH_022	Inscription valide	Création nouveau compte	Email non utilisé	1. Accéder à register2. Remplir formulaire3. Soumettre	Email: newUserEmail: new@example.comMot de passe: Secure123!Confirmation:	Compte crééRedirection vers dashboard	Utilisateur connecté	PASS

TC_AUTH_023	Inscription email existant	Tentative avec email existant	Utilisateur existe déjà	1. Accéder à register2. Entrer email existant3. Soumettre	Email: existant@example.com	Message d'erreur "email déjà pris"	Compte non créé	FAIL
TC_AUTH_024	Vérification email	Clique sur lien vérification	Lien valide reçu	1. Cliquer sur lien de vérification	Email: user@example.com	Redirection vers dashboard avec paramètre ?verified=1	Email marqué comme vérifié	PASS
TC_AUTH_025	Vérification inutile	Lien cliqué quand déjà vérifié	Email déjà vérifié	1. Cliquer sur lien de vérification		Redirection vers dashboard	Aucun changement	PASS
TC_SETTINGS_001	Affichage profil	Page paramètres profil	Utilisateur connecté	1. Accéder à /settings/profile		Page de profil affichée avec données utilisateur		PASS
TC_SETTINGS_002	Mise à jour profil	Modification infos profil	Utilisateur connecté	1.Modifier les informations2. Soumettre le formulaire	Nom: NouveauNomEmail: nouveau@example.com	Données mises à jourRedirection vers page profil	Profil mis à jour	PASS
TC_SETTINGS_003	Changement email	Modification email	Utilisateur connecté	1. Changer l'email2. Soumettre	Nouvel email: newemail@example.com	Email marqué comme non vérifiéRedirection vers page	Email non vérifié	PASS
TC_SETTINGS_004	Suppression compte	Suppression compte utilisateur	Utilisateur connecté	1. Accéder à paramètres2. Confirmer suppression3. Entrer mot de passe	Mot de passe: correct123	Compte suppriméRedirection vers accueil	Session terminée	PASS
TC_SETTINGS_005	Affichage mot de passe	Page changement mot de passe	Utilisateur connecté	1. Accéder à /settings/password		Formulaire changement mot de passe affiché		PASS
TC_SETTINGS_006	Changement mot de passe	Modification mot de passe	Utilisateur connecté	1. Entrer mot de passe actuel2. Nouveau mot de passe3. Confirmer	Actuel: oldPass123Nouveau: NewSecure123!	Mot de passe mis à jourRedirection vers page paramètres	Mot de passe changé	PASS
TC_SETTINGS_007	Échec changement	Mauvais mot de passe actuel	Utilisateur connecté	1. Entrer mauvais mot de passe2. Nouveau mot de passe3. Confirmer	Actuel: wrongPassNouveau: NewSecure123!	Message d'erreur affiché	Mot de passe inchangé	FAIL
TC_PRODUCT_001	Liste produits admin	Affichage liste produits	Admin connecté	1. Accéder à /admin/products		Liste produits avec catégories affichée		PASS
TC_PRODUCT_002	Liste produits client	Affichage boutique	Utilisateur quelconque	1. Accéder à /products		Produits et catégories affichés		PASS
TC_PRODUCT_003	Création produit	Ajout nouveau produit	Admin connecté	1. Remplir formulaire2. Soumettre	Nom: NouveauProduitPrix: 19.99Catégorie: 1	Produit crééRedirection vers liste	Produit en base	PASS
TC_PRODUCT_004	Affichage détail	Page détail produit	Produit existant	1. Cliquer sur produit ID 1		Détails produit + catégories affichés		PASS
TC_PRODUCT_005	Mise à jour produit	Modification produit	Admin connecté	1.Modifier champs2. Soumettre	Nom: ProduitModifiéPrix: 24.99	Produit mis à jourRedirection vers détail	Données mises à jour	PASS
TC_PRODUCT_006	Suppression produit	Retrait produit	Admin connecté	1. Cliquer supprimer sur produit ID 1		Produit suppriméMessage de succès	Produit retiré de la base	PASS
TC_PRODUCT_007	Filtre par catégorie	Affichage produits filtrés	Catégories existantes	1. Sélectionner catégorie 12. Appliquer filtre		Seuls produits catégorie 1 affichés		PASS
TC_PRODUCT_008	Liste catégories admin	Affichage liste catégories	Admin connecté	1. Accéder à /admin/categories		Liste des catégories affichée		PASS
TC_PRODUCT_009	Création catégorie	Ajout catégorie	Admin connecté	1. Remplir formulaire2. Soumettre	Nom: Accessoires	Catégorie crééeRedirection vers liste	Catégorie en base	PASS
TC_PRODUCT_010	Modification catégorie	Modifier catégorie existante	Admin connecté	1. Modifier nom2. Soumettre	Nom: Electronique	Catégorie mise à jour	Données mises à jour	PASS
TC_PRODUCT_011	Suppression catégorie	Supprimer une catégorie	Admin connecté	1. Cliquer supprimer sur catégorie ID 1		Catégorie suppriméeMessage de succès	Catégorie retirée de la base	PASS

TC_CATEGORY_001	Liste catégories admin	Affichage liste catégories	Admin connecté	1. Accéder à /admin/categories		Liste des catégories affichée		PASS
TC_CATEGORY_002	Hierarchie catégories	Affichage hiérarchique	Admin connecté	1. Accéder à /admin/categories-hierarchy		Arbre des catégories affiché		PASS
TC_CATEGORY_003	Création catégorie	Ajout catégorie	Admin connecté	1. Remplir formulaire2. Soumettre	Nom: Accessoires	Catégorie crééeRedirection vers liste	Catégorie en base	PASS
TC_CATEGORY_004	Modification catégorie	Modifier catégorie existante	Admin connecté	1. Modifier nom2. Soumettre	Nom: Electronique	Catégorie mise à jour	Données mises à jour	PASS
TC_CATEGORY_005	Suppression catégorie	Supprimer une catégorie	Admin connecté	1. Cliquer supprimer sur catégorie ID 1		Catégorie suppriméeMessage de succès	Catégorie retirée de la base	PASS
TC_ORDER_001	Liste commandes admin	Affichage liste commandes	Admin connecté	1. Accéder à /admin/ordres		Liste des commandes affichée		PASS
TC_ORDER_002	Statistiques dashboard	Voir statistiques commandes	Admin connecté	1. Accéder à /admin/dashboard		Statistiques de commandes affichées		PASS
TC_ORDER_003	Liste commandes client	Afficher commandes du client	Client connecté	1. Accéder à /client/my-orders		Liste des commandes personnelles affichée		PASS
TC_ORDER_004	Détail commande client	Affichage détail commande	Client connecté + Commande existante	1. Accéder à /client/view-order/{id}		Détails de la commande affichés		PASS
TC_CHECKOUT_001	Paiement réussite	Paiement via Stripe	Client connecté + Commande existante	1. Accéder à /checkout/create2. Redirection Stripe3. Payer		Redirection vers /checkout/success	Commande marquée comme payée	PASS
TC_CHECKOUT_002	Annulation paiement	Annuler le paiement	Client connecté	1. Accéder à /checkout/create2. Redirection Stripe3. Annuler paiement		Redirection vers /checkout/cancel		PASS
TC_USER_001	Liste clients admin	Affichage liste clients	Admin connecté	1. Accéder à /admin/client		Liste des clients affichée		PASS
TC_USER_002	Création client	Ajout d'un nouveau client	Admin connecté	1. Remplir formulaire2. Soumettre	Nom: Jean DupontEmail: jean@example.com Password: secret	Client créé et redirigé vers liste	Client en base	PASS
TC_USER_003	Suppression client	Supprimer un client	Admin connecté + Client existant	1. Cliquer supprimer sur client ID 1		Client suppriméMessage de succès	Client retiré de la base	PASS

Gestion de projet

Pour ce projet, j'ai assumé l'intégralité du travail en autonomie, prenant en charge la planification, la conception, le développement ainsi que les phases de tests et de validation. Cette approche m'a permis de maîtriser chaque étape du processus, d'assurer la cohérence technique et fonctionnelle du projet, et de garantir une qualité optimale dans la livraison finale.

Organisation du travail — cycle en V

Pour organiser mon travail sur le projet Laravel, j'ai suivi une méthode appelée cycle en V. C'est une façon de travailler qui avance étape par étape. Chaque étape de développement est suivie d'un moment de vérification pour s'assurer que tout fonctionne bien. Cela permet de créer une application de qualité, sans sauter d'étapes importantes.

Analyse des besoins

La première étape a été de bien comprendre ce que devait faire l'application. J'ai listé toutes les fonctionnalités importantes comme l'inscription, la connexion, le tableau de

bord, la gestion des utilisateurs, etc. Cette phase m'a permis d'avoir une vision claire de ce que je devais développer.

Conception

À ce stade, j'ai préparé la structure de l'application. J'ai créé la base de données avec les bonnes tables et relations. J'ai aussi pensé à l'organisation du code (modèles, vues, contrôleurs) et à la sécurité (middleware, protections CSRF, etc.).

Développement

J'ai ensuite commencé à coder l'application. J'ai utilisé Laravel pour créer les fonctionnalités prévues : gestion des utilisateurs, authentification, affichage des pages, enregistrement des données, etc. J'ai aussi utilisé Git pour suivre l'évolution du code et garder des sauvegardes.

Déploiement

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

J'ai configuré plusieurs conteneurs pour faire tourner PHP 8.3.25, MySQL, Nginx et Node.js.

Grâce à Docker Compose, j'ai pu automatiser le lancement de l'ensemble du projet.

J'ai aussi adapté le projet à ce nouvel environnement :

- Mettre en place un Dockerfile personnalisé
- Configurer Nginx avec le fichier default.conf
- Construire le front avec `npm run build`
- Migrer les tables avec `php artisan migrate`

Ce choix m'a permis de contourner les limites de Plesk et d'avoir un environnement proche d'un hébergement réel.

Conclusion

Le développement du projet Skinn m'a permis de mettre en pratique l'ensemble des compétences acquises au cours de ma formation, tant sur les aspects front-end que back-end. Cette boutique en ligne, centrée sur les soins corporels naturels, a été conçue dans une optique de simplicité, d'accessibilité et de sécurité, avec des technologies modernes telles que Laravel, Inertia.js et React.

Ce projet a été une véritable opportunité de me confronter à toutes les étapes du cycle de vie d'une application web, de l'analyse des besoins à la mise en production, en passant par la conception de la base de données, l'implémentation des interfaces, la gestion des utilisateurs, ou encore l'intégration d'un système de paiement sécurisé avec Stripe.

Au-delà de l'aspect technique, ce travail m'a permis de renforcer mon autonomie, ma rigueur et ma capacité à mener un projet complet de A à Z. Il marque une étape importante dans mon parcours de reconversion professionnelle et constitue une base solide pour la suite de mes études et mon ambition de devenir développeuse full-stack.