

PixelPlush

Boutique en ligne de vente de peluche

Projet réalisé dans le cadre de la présentation au

Titre Professionnel Développeur Web et Web Mobile

Présenté par

Youssef Ghollamallah

La Plateforme_ - CDPI

SOMMAIRE

SOMMAIRE

INTRODUCTION

LISTE DES COMPÉTENCES COUVERTES PAR LE PROJET

- 1. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité**
 - a. Installer et configurer son environnement de travail**
 - b. Maquetter des interfaces utilisateur web ou web mobile**
 - c. Réaliser des interfaces utilisateur statiques web ou web mobile.**
 - d. Développer la partie dynamique des interfaces utilisateur web ou web mobiles**
- 2. Développer la partie back-end d'une application web ou web mobile**
 - a. Mettre en place d'une base de données relationnelle**
 - b. Développer des composants d'accès aux données SQL et NoSQL.**
 - c. Développer des composants métier côté serveur.**
 - d. Documenter le déploiement d'une application dynamique web ou web mobile.**

RÉSUMÉ DU PROJET

CAHIER DES CHARGES

- 1. Besoins et objectifs de l'application**
 - a. Besoins**
 - b. Objectifs**
 - c. Cibles**
- 2. Users Stories**
- 3. MVP**
 - a. Users**
 - b. Admin**
- 4. Fonctionnalités détaillées des pages**
- 5. Evolutions potentielles**
 - a. Users**
 - b. Admin**
- 6. Wireframes**
- 7. Charte graphique et logo**
- 8. Exemple de maquettes**

SPÉCIFICATIONS TECHNIQUES

- 1. Technologies**
- 2. Navigateurs compatibles**
- 3. Possibilité de déploiement**
- 4. Création de la base de données**
 - a. MCD**
 - b. MLD**
 - c. MPD**
- 5. Routes front et back**
 - a. Back-end**
 - b. Front-end**

RÉALISATIONS PERSONNELLES

Ajout de produit

- a. Back**
- b. Front**

VULNÉRABILITÉS DE SÉCURITÉ ET VEILLE

- 1. Veille technologique**
 - a. Nodemailer et serveur SMTP**
 - b. Astro**
- 2. Veille de sécurité**
 - a. JWT et cryptage du mot de passe utilisateur**
 - b. Sécurisation des requêtes**
- 3. Processus de recherche et traduction**

CONCLUSION

ANNEXE

INTRODUCTION

Depuis mon jeune âge, je suis passionné par l'informatique, et plus particulièrement par la programmation. Bien que mon parcours m'ait éloigné un temps de cet univers, j'ai continué à m'intéresser activement aux sujets liés au développement. Grâce à cet apprentissage autodidacte, j'ai pu pratiquer régulièrement le code et suivre l'actualité du secteur.

Après une lassitude liée à mon précédent domaine d'activité et un accident de travail, j'ai décidé de me réorienter vers une voie me permettant de valoriser mon expérience acquise. Après quelques recherches sur les moyens d'y parvenir, j'en ai discuté avec ma conseillère France Travail et me suis engagé dans un premier bootcamp chez Webforce3.

À la suite de cette première expérience de cinq mois, j'ai intégré une formation Bachelor IT (B1) à La Plateforme_, où j'ai eu l'opportunité de poursuivre mon apprentissage avec un programme complet en développement web et mobile.

Dans le cadre de cette formation, pour mettre en pratique nos acquis, La Plateforme_ a mis en place des projets de groupe. Pendant un mois, avec quatre collègues, nous avons travaillé sur **PixelPlush**, une boutique en ligne.

Au-delà de l'acquisition de compétences techniques, je souhaitais aussi apprendre à collaborer efficacement en équipe. Jusqu'alors, j'avais principalement travaillé sur des projets personnels, de manière autonome. Ce projet m'a donc permis de découvrir les bonnes pratiques du travail en groupe et de mieux comprendre l'organisation et le fonctionnement d'une équipe projet.

En somme, la réalisation de PixelPlush m'a permis de développer de nouvelles compétences tout en collaborant avec une équipe sur un projet concret, qui aurait probablement nécessité plus de temps si chacun avait travaillé seul.

LISTE DES COMPÉTENCES COUVERTES PAR LE PROJET

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

a. Installer et configurer son environnement de travail

Pour le développement de **PixelPlush**, nous avons mis en place un environnement local basé sur **Laragon** et **Visual Studio Code**.

Nous avons opté pour **PHP** et **MySQL** côté back-end, en organisant le projet selon une architecture **MVC simple**, incluant :

- Un **autoloader personnalisé** pour charger dynamiquement les classes,
- Une **gestion des routes** via un routeur interne et un fichier `.htaccess`,
- Une **séparation claire** entre la logique métier, les vues et les interactions avec la base de données.

Nous avons également utilisé des dépendances externes, notamment **PHPMailer**, pour la gestion de l'envoi d'e-mails lors de l'inscription et du processus de mot de passe oublié.

b. Maquetter des interfaces utilisateur web ou web mobile

En amont de la réalisation à proprement parler de notre application, nous avons choisi dans un premier temps de travailler sur la réalisation des documents de conception. Dans ce but, nous avons donc réalisé en premier lieu des users stories pour définir très clairement ce que pourront faire les utilisateurs sur notre application. Puis nous avons réalisé les wireframes du projet, aux formats mobile et desktop pour imaginer la structuration de nos pages, puis enfin nous avons réalisé une charte graphique et des maquettes pour voir concrètement à quoi allait ressembler notre application.

c. Réaliser des interfaces utilisateur statiques web ou web mobile

Dans le cadre du projet **PixelPlush**, une boutique en ligne fictive spécialisée dans la vente de peluches, nous avons conçu une interface utilisateur à la fois **statique, responsive et accessible**.

L'objectif était de proposer une **expérience fluide et agréable**, quel que soit le terminal utilisé (ordinateur, tablette ou smartphone), afin de reproduire les standards attendus d'un véritable site e-commerce.

L'interface a été développée en **HTML, CSS et JavaScript**, avec une attention particulière portée à la **structure des pages**, à la **lisibilité des contenus**, à la **navigation intuitive** ainsi qu'à l'**adaptabilité du design** grâce à l'utilisation de **flexbox**, de **media queries**, et de composants réutilisables.

Cette démarche nous a permis de mettre en pratique les bonnes pratiques du responsive design, tout en veillant à garantir une cohérence graphique et fonctionnelle sur l'ensemble du site.

d. Développer la partie dynamique des interfaces utilisateur web ou web mobile

À partir des user stories définies lors de la phase de préparation du projet **PixelPlush**, il est rapidement apparu essentiel de proposer une **interface dynamique** permettant une interaction fluide entre l'utilisateur et le site.

Bien que le projet ait été développé en **PHP** côté serveur, nous avons intégré une **logique dynamique** via des **interactions JavaScript** côté client : mise à jour du panier en temps réel, gestion des erreurs de formulaire sans rechargement de page, ou encore affichage conditionnel de certains éléments selon l'état de l'utilisateur (connecté, admin).

Ces fonctionnalités ont été pensées pour **améliorer l'expérience utilisateur**, en rendant le site plus réactif et interactif, tout en conservant une base technique simple et accessible à l'ensemble de l'équipe.

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

a. Mettre en place une base données relationnelle

Tout comme la réalisation du maquettage de l'application, nous avons voulu dès le début organiser la façon dont allaient être structurés nos données. Nous avons donc commencé par réfléchir aux données que nous souhaitions intégrer à l'application, puis nous avons créé un MCD, MLD, ainsi qu'un dictionnaire des données. Pour gérer la base de données de notre application, nous avons choisi d'utiliser MySQL

b. Développer des composants d'accès aux données SQL et NoSQL.

Dans le cadre du projet PixelPlush, une boutique en ligne développée en équipe, nous avons mis en œuvre une couche d'accès aux données en PHP, afin de gérer les interactions entre notre application et la base de données MySQL.

L'objectif était de permettre une récupération et une manipulation efficaces des données (produits, utilisateurs, commandes...) tout en respectant une structure claire et sécurisée, conforme à une architecture MVC.

Nous avons utilisé PDO (PHP Data Objects) pour établir la connexion, chaque table de la base est associée à un fichier modèle PHP. Les modèles contiennent les fonctions CRUD nécessaires

Ils communiquent directement avec la base via PDO, en encapsulant les requêtes SQL.

c. Développer des composants métier côté serveur

Le développement de la logique métier côté serveur était essentiel pour gérer les fonctionnalités centrales de l'application : inscription et connexion des utilisateurs, gestion du panier

Implémentation de la création de compte avec envoi d'email de confirmation via PHPMailer ;

Vérification des identifiants lors de la connexion, avec hashage des mots de passe

d. Documenter le déploiement d'une application dynamique web et web mobile

Dans le cadre de notre application web dynamique enfin de rendre accessible notre application et être en conditions réelles. Nous avons décidé d'héberger notre Projet sur Hostinger, j'ai adapté mon code sur l'hébergeur en modifiant le système d'URL et Documenté dans un fichier README le processus de mise en ligne étape par étape (fichiers, base, configuration).

RÉSUMÉ DU PROJET

Chaque jour, des passionnés de peluches recherchent des produits uniques, réconfortants et originaux à offrir ou à collectionner. Face à une offre fragmentée, nous avons souhaité centraliser cette passion dans un espace simple, chaleureux et intuitif.

Notre site e-commerce nommé **PixelPlush** a été conçu pour répondre à ce besoin en proposant une plateforme dédiée à la vente de peluches pour petits et grands. Il s'adresse à la fois aux particuliers à la recherche du cadeau parfait et aux collectionneurs en quête de nouveautés.

Pensé avant tout pour être fonctionnel et accessible, PixelPlush offre une navigation fluide, un affichage dynamique des produits, et une gestion efficace du panier. Grâce à une interface d'administration, les gestionnaires du site peuvent facilement ajouter, modifier ou supprimer des articles, tandis que les utilisateurs peuvent consulter les nouveautés, filtrer les produits par catégorie, ou encore finaliser leur commande en toute simplicité.

Le site a également été conçu avec une vision à long terme : permettre une évolutivité vers des fonctionnalités plus avancées comme les comptes utilisateurs, les avis produits ou les offres promotionnelles. L'objectif est d'offrir une expérience aussi agréable qu'efficace, aussi bien pour les clients que pour les administrateurs du site.

En résumé, **PixelPlush** vise à devenir une boutique en ligne de référence pour tous les amoureux de peluches, avec une interface soignée, une gestion claire et une expérience utilisateur optimisée.

CAHIER DES CHARGES

1. Besoins et objectifs de l'application

- **Besoins**

Le projet PixelPlush répond à un besoin clair : centraliser l'achat de peluches dans un espace numérique dédié, chaleureux et facile à utiliser, tant pour les utilisateurs que pour les gestionnaires du site.

Pour comprendre ces besoins, nous avons analysé les attentes des utilisateurs en ligne, les habitudes d'achat sur les plateformes e-commerce, ainsi que les contraintes de gestion côté administrateur.

Les utilisateurs recherchent :

- Une navigation simple et fluide pour explorer les peluches disponibles ;
- Des informations claires sur chaque produit (prix, description, disponibilité) ;
- Un accès rapide aux nouveautés, aux meilleures ventes et aux offres spéciales ;

Les gestionnaires du site, quant à eux, ont besoin :

- D'une interface d'administration efficace pour ajouter, modifier ou supprimer des produits ;
- D'un aperçu clair des stocks et des ventes ;
- De la possibilité de mettre en avant certains articles

- **Objectifs**

Le site PixelPlush a été pensé pour :

- Offrir une **expérience utilisateur fluide et agréable**, tant sur ordinateur que sur mobile ;
- **Simplifier la gestion du catalogue** produit grâce à un back-office dédié ;
- **Valoriser les produits** via une présentation soignée et des fiches détaillées ;
- **Renforcer l'identité de la marque** PixelPlush avec un design cohérent et accueillant ;
- Permettre, à terme, d'ajouter des **fonctionnalités évolutives** : comptes clients, avis, wishlist, suivi des commandes...

L'objectif final est de proposer **une solution e-commerce simple, moderne et évolutive**, adaptée à un marché de niche en pleine croissance : celui des passionnés de peluches.

- **Cibles**

Pour mieux concevoir notre plateforme, nous avons identifié plusieurs profils d'utilisateurs :

Maria Hernandez, 27 ans, Bordeaux – Jeune maman

- Équipements : Smartphone Android, tablette
- Ce qu'elle recherche : Une peluche douce et originale pour sa fille
- Navigation : Page "nouveau" puis "peluches animaux"
- Attentes : Navigation fluide, visuels attractifs, livraison rapide
- Prérequis : Site responsive et rassurant

Lucas Brunet, 34 ans, Nantes – Collectionneur

- Équipements : PC portable, smartphone iOS
- Ce qu'il recherche : Des modèles exclusifs ou en édition limitée
- Navigation : Recherche par catégorie, filtre par rareté
- Attentes : Détails produits précis, système de favoris
- Prérequis : Interface claire et base de données bien structurée

Sarah Lambert, 42 ans, Lille – Responsable boutique cadeaux

- Équipements : Desktop pro, iPad
- Ce qu'elle recherche : Une source d'approvisionnement pour ses clients
- Navigation : Page des catégories, possibilité de commander en quantité
- Attentes : Gestion de compte pro, facturation
- Prérequis : Système évolutif pouvant accueillir une offre B2B

Benoit Duroc, 30 ans, Paris – Administrateur PixelPlush

- Équipements : Desktop, tablette
- Ce qu'il recherche : Une interface d'administration simple et fonctionnelle
- Navigation : Tableau de bord admin
- Attentes : Gérer rapidement les produits, voir les ventes en temps réel
- Prérequis : Back-office sécurisé, fiable et ergonomique

2. User Stories

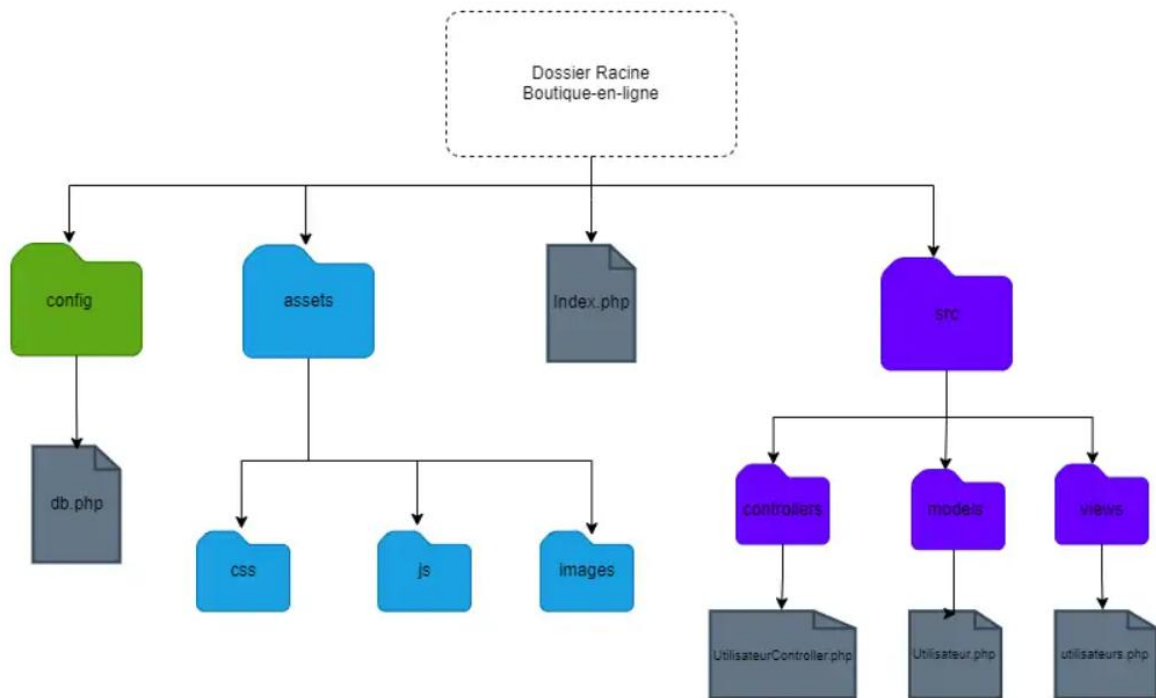
Légende :

MVP

Versions à venir

En Tant que	Je souhaite	Afin de
Visiteur	Consulter la liste des produits	Découvrir les produits proposer
Visiteur	Voir les détails d'un produit	Connaitre ses caractéristique
Visiteur	M'inscrire au site	Avoir accès au fonctionnalité
Visiteur	Ajouter un produit au panier	Préparer ma commande
Visiteur	Visualiser mon panier	Vérifier les articles que j'ai sélectionnés.
Utilisateur connecté	Accéder à mon profil	Modifier mes informations personnelles
Utilisateur Connecté	Valider ma commande	Finaliser mon achat
Utilisateur Connecté	Voir mon historique de commande	Avoir la liste des commandes passé
Utilisateur connecté	Avoir un suivi de commande	Pour vérifier ou en ai ma commande
Utilisateur connecté	Mettre un produit avis	Donner mon avis sur les produits
Administrateur	Ajouter un nouveau produit	Enrichir le catalogue de la boutique
Administrateur	Modifier un produit existant	Afin de corriger ou mettre à jour ses information
Administrateur	Supprimer un produit	Retirer les articles non disponible ou obsolètes
Administrateur	Voir la liste des commandes passées	De suivre les ventes réalisées
Administrateur	Ajouter de nouveau employé	Qu'il puisse utiliser l'application

3. Arborescence



4. MVP – Minimum Viable Product

Notre Produit Minimum Viable s'est naturellement construit autour de l'expérience d'achat en ligne de peluches. L'objectif principal est de permettre aux utilisateurs de consulter les produits, gérer un panier et passer commande, tout en offrant à l'administrateur un espace de gestion simplifié pour piloter le catalogue.

Pour cela, une structure claire, une interface intuitive, et une authentification sécurisée côté administrateur sont primordiales. Ces éléments permettent de garantir la fiabilité des informations visibles par les utilisateurs.

Fonctionnalités présentes dans notre MVP :

- L'utilisateur peut consulter les mentions légales et la politique de confidentialité du site
- L'administrateur peut s'authentifier via un email et un mot de passe pour accéder à l'interface de gestion

a. User (Client)

À son arrivée sur le site :

- Il accède à une page d'accueil avec les nouveautés, les catégories mises en avant et une présentation claire des produits



Navigation :

- Il peut consulter la liste des peluches, avec un système de filtres (catégories)
- Il peut cliquer sur un produit pour accéder à sa fiche détaillée (image, description, prix, quantité).
- Il peut ajouter une peluche à son panier, avec un message de confirmation.
- Il peut valider sa commande (dans cette version MVP, sans paiement réel sandbox PayPal)

b. Administrateur

À son arrivée après connexion :

- Il accède à une interface d'administration avec un tableau de bord simplifié.

Fonctionnalités disponibles :

- Voir la liste des produits et des utilisateurs avec possibilité de modification
- Ajouter un nouveau produit (nom, image, description, prix, catégorie.)
- Modifier une fiche produit existante.
- Supprimer un produit du catalogue si nécessaire.

Ce MVP se concentre sur l'essentiel : permettre une navigation fluide, une présentation claire des produits, et une gestion minimale mais efficace du contenu. Il constitue la base sur laquelle des fonctionnalités plus avancées pourront être développées, comme la gestion des comptes clients, les avis produits, ou encore un système fonctionnel.

5. Fonctionnalités détaillées des pages

L'ensemble des pages administratives est sécurisé : elles sont accessibles uniquement après authentification. Seule la page publique du site (accueil, détail et panier) est accessible à tous. L'espace administrateur est restreint à l'équipe de gestion via un email et un mot de passe.

Page d'accueil

Accessible par tous les utilisateurs.

Cette page présente :

- Une bannière de présentation de la boutique
- Les nouveautés mises en avant
- Les catégories de peluches
- Un accès direct aux mentions légales et la politique de confidentialité

Depuis cette page, les utilisateurs peuvent accéder aux autres parties du site comme leur panier, la page détail d'un produit ou le système d'inscription ou de login

Page détail produit

Affiche les informations détaillées d'une peluche

- Image
- Nom
- Prix
- Description
- Quantité

L'utilisateur peut ajouter l'article au panier depuis cette page

Page profil (User)

Accessible pour les utilisateurs uniquement

Affiche les informations personnelles de l'utilisateur et son historique de commande

- Nom
- Prénom
- E-mail
- Adresse

Page profil (Admin)

Accessible pour les administrateurs.

Même information que les utilisateurs classiques avec une redirection vers les options suivante :

- Gestion produit
- Gestion utilisateur

Page panier

Permet à l'utilisateur de :

- Visualiser les produits ajoutés
- Modifier les quantités
- Supprimer un article du panier
- Valider la commande (dans le MVP : validation sans paiement réel)

6. Evolution potentielles

Il est prévu d'améliorer l'application par des fonctionnalités complémentaires pour une meilleure utilisation et une meilleure expérience utilisateur. Les améliorations seront apportées sur les 2 parties utilisateur

A. User (client)

- Pourra s'abonner à la newsletter pour recevoir les actualités, offres spéciales ou lancements de collections.
- Pourra laisser un avis/commentaire sur une peluche achetée afin de partager son expérience avec les autres clients
- Pourra ajouter des produits en favoris afin de les retrouver plus facilement et être notifié si le produit revient en stock.

B. Administrateur

- Pourra gérer les comptes clients : voir leurs commandes, modifier ou désactiver un compte.
- Pourra ajouter plusieurs images par produit et intégrer des vidéos de présentation.
- Pourra activer un système d'évaluation des produits (notes et avis clients).
- Pourra recevoir une alerte sur les produits en rupture de stock pour les réapprovisionner à temps.

7. Wireframes

8. Charte graphique et logo

L'application étant destinée à un public varié (familles, passionnés de peluches, collectionneurs.), l'objectif était de créer une identité visuelle douce, rassurante et ludique, en adéquation avec l'univers des peluches.

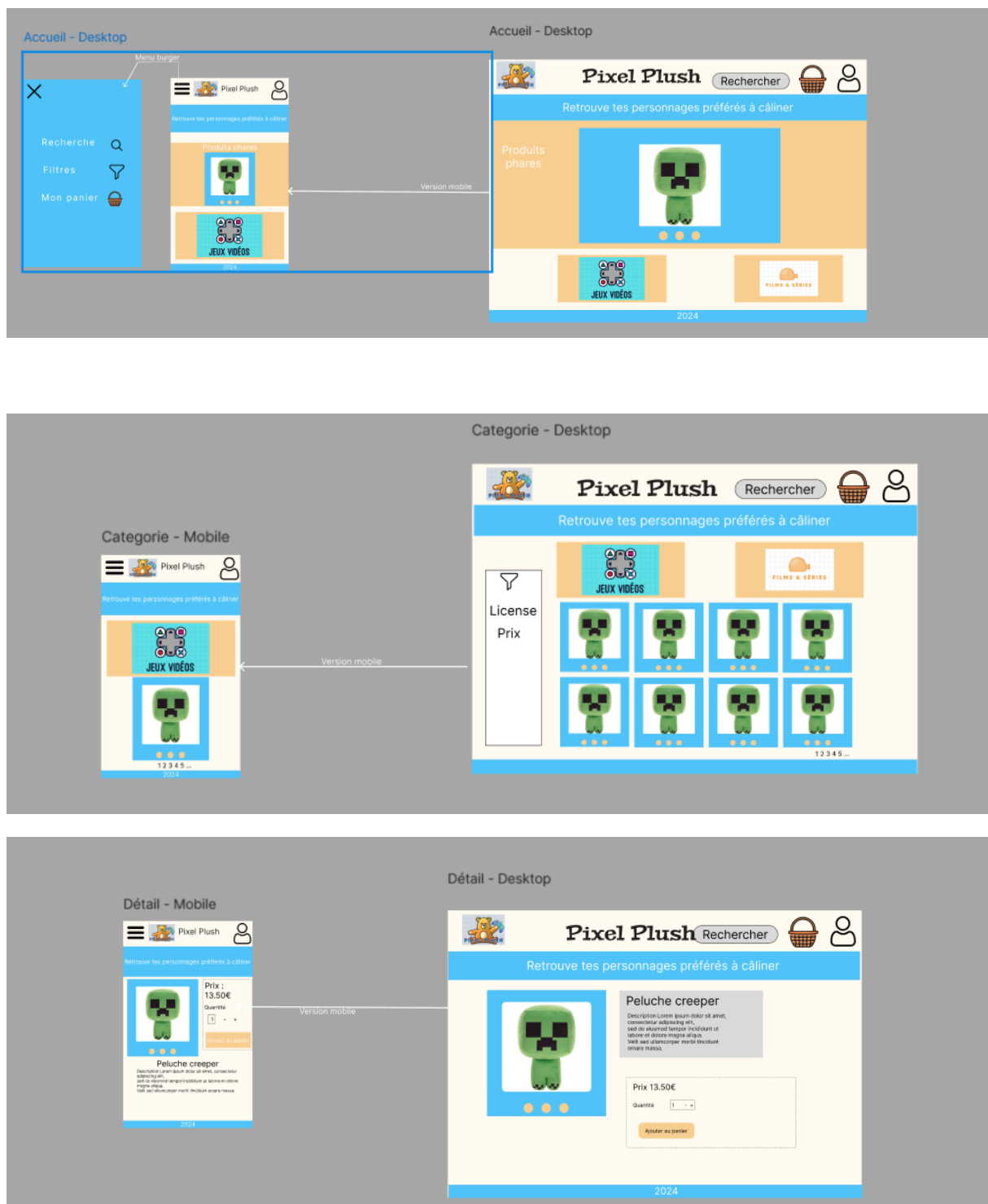
Nous nous sommes inspirés des codes graphiques de l'enfance et de l'univers kawaii.

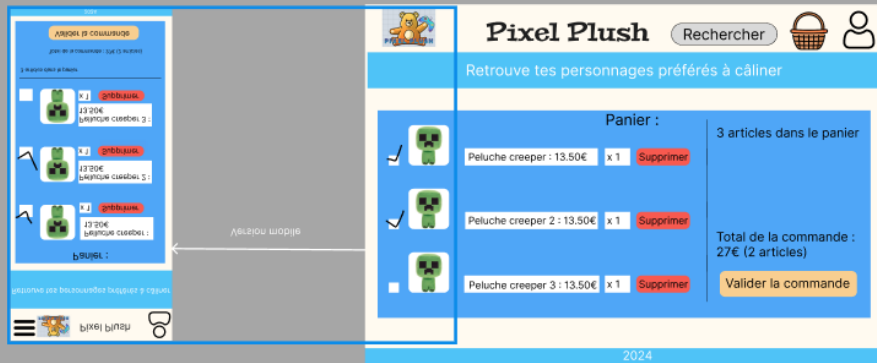


Pour le logo a été pensé dans un esprit minimaliste et attachant. Une peluche d'ours qui reprend une typographie arrondie avec une intégration subtile d'un symbole de peluche.



9. Exemple de maquettes





SPÉCIFICATIONS TECHNIQUES

Pour réaliser le site, nous avons commencé par définir les technologies adaptées à nos objectifs et compétences acquis, tout en gardant à l'esprit la maintenabilité du projet et la simplicité d'utilisation. L'accessibilité multi-supports ainsi que la compatibilité avec les principaux navigateurs ont également été des critères déterminants.

Enfin, nous avons structuré l'architecture technique autour de la base de données, du code serveur (back-end PHP) et de l'affichage (JavaScript).

1. Technologies

- Front-end
 - HTML5
 - CSS3
 - JavaScript
- Backend
 - PHP (MVC Personnalisé)
 - .htaccess
 - PHPMailer pour l'envoi d'email avec google en fournisseur SMTP
- Environnement de développement
 - VSCode
 - Laragon
- Outils complémentaires
 - Prettier

2. Navigateurs compatibles

Pour assurer une bonne accessibilité, l'application est conçue pour être compatible avec les navigateurs les plus utilisés, aussi bien sur desktop que sur mobile.

Selon une étude de marché de 2024, les navigateurs les plus utilisés sont :

- **Desktop**
 - Chrome : 66.88%
 - Safari : 13.2%
 - Edge : 8.48%
- **Mobile :**
 - Chrome : 69.79%
 - Safari : 21.44%

Source : <https://www.blogdumoderateur.com/navigateurs-web-plus-utilises-france-monde-2025/>

3. Possibilités de déploiement

Plusieurs possibilités de déploiement s'offrent à nous avec des solutions plus ou moins faciles d'utilisation, et plus ou moins coûteuses tant au niveau humain que financier.

Nous avons étudié 2 cas :

1. **Le déploiement sur Plesk, l'hébergement fourni par l'école**

Avantages	Inconvénients
<ul style="list-style-type: none">○ Déploiement simple : Gestionnaire de fichiers intégré.○ Support natif de PHP, MySQL○ Gratuit	<ul style="list-style-type: none">○ Non maîtrise des performances fournies de l'hébergeur○ Moins de souplesse qu'une infrastructure personnelle

Estimation du coût mensuel : quasiment nul, fourni par la plateforme

2. **Le déploiement sur Hostinger, Mon hébergeur personnel**

Avantages	Inconvénients
<ul style="list-style-type: none">○ Contrôle total sur les données○ Support natif de PHP, MySQL○ Accès facile avec GIT, gestionnaire de fichiers intégré.○ Certificats SSL pour sécuriser le site○ Accès à un email personnalisé	<ul style="list-style-type: none">○ Un peu plus cher que le déploiement sur Plesk

Estimation du coût mensuel :

- **Nom de domaine : 2 euros la première année puis 9.99**
- **Hébergement Hostinger que j'utilise pour tous mes projets 15€ par mois.**

Au vu des éléments étudiés précédemment, et des limitations rencontrées avec Plesk, Nous avons finalement décidé d'opter pour un hébergement via Hostinger.

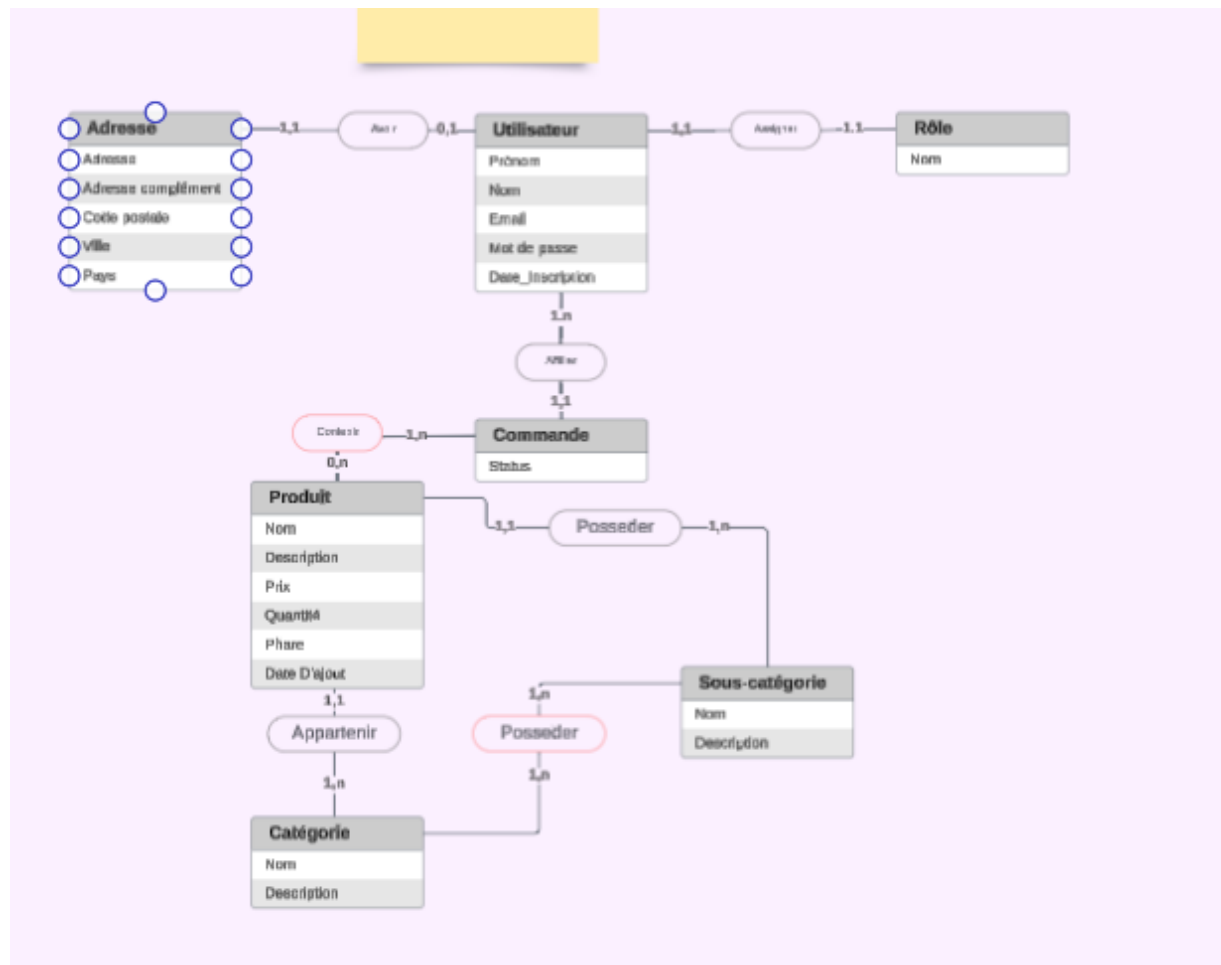
Ce choix s'est imposé pour plusieurs raisons :

- Je disposais déjà d'un compte hostinger avec possibilité d'héberger jusqu'à 100 sites.
- Il propose un accès direct aux bases de données MySQL, à la gestion DNS, aux certificats SSL gratuits.
- Cette expérience m'a permis d'apprendre à déployer au site sur un autre hébergeur, d'en gérer les différents paramètres, et de m'adapter à un environnement de production réel.

Ce changement m'a ainsi offert plus de flexibilité, tout en gardant un bon rapport qualité/prix, et en me formant concrètement à la gestion d'un hébergement web professionnel.

4. Création de la base de données

A. MCD



B. MLD

User (id, prenom, nom, email, mot_de_passe, date_inscription, id_adresse, rôle_id)

Rôle (id_role, nom_role)

Adresse (id, adresse, adresse_complement, code_postal, ville, pays, id_utilisateur)

Produit (id, nom, description, prix, quantite, image, data_ajout, id_souscatégorie, id_catégories)

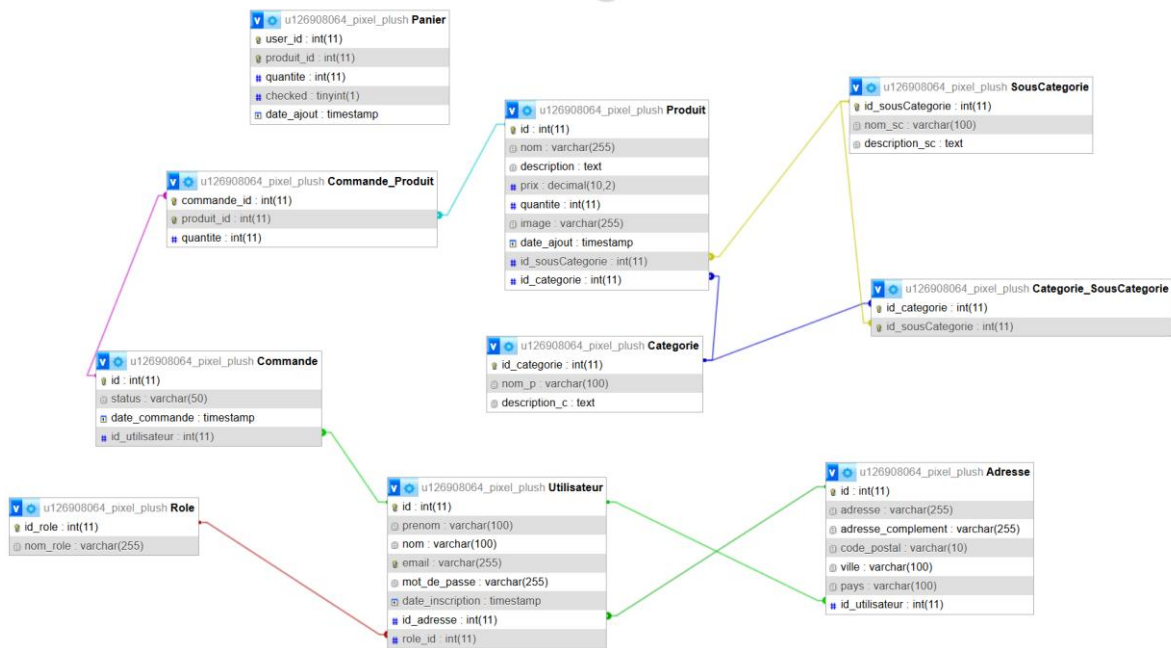
Commande (id, status, date_commande, id_utilisateur)

Panier (user_id, produit_id, quantité, checked, date_ajout)

Catégorie (id_catégorie, nom_p, description)

SousCatégorie (id_souscatégorie, nom_sc, description_sc)

C. MPD



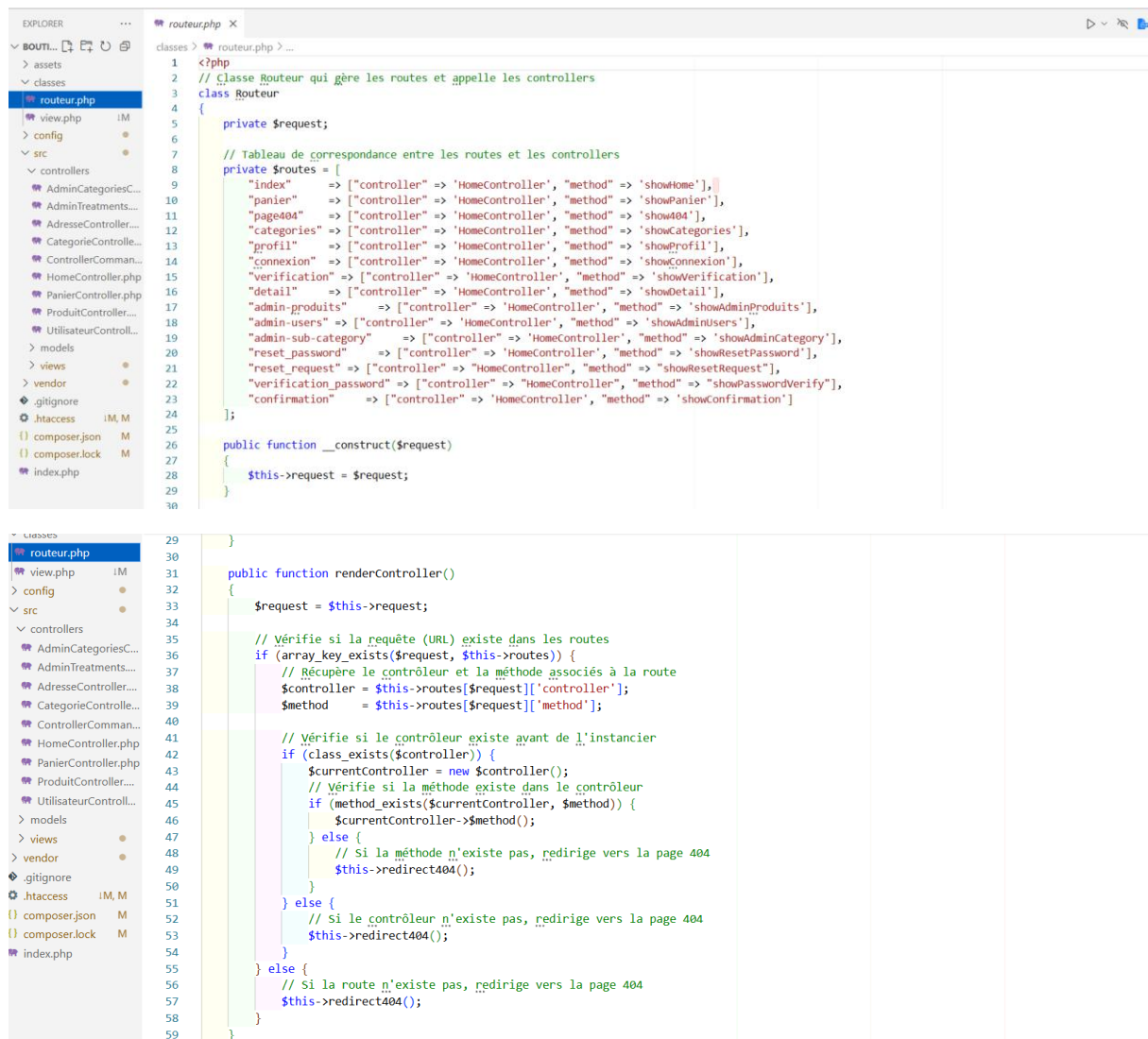
Lien MCD/ MLD : https://lucid.app/lucidchart/59a78560-2c60-440a-be2c-7c79a97970af/edit?viewport_loc=-3764%2C701%2C3837%2C1776%2C0_0&invitationId=inv_412a86a6-3271-49f4-ab9c-aea75d1e947a

5. Routes front et back

A. Backend

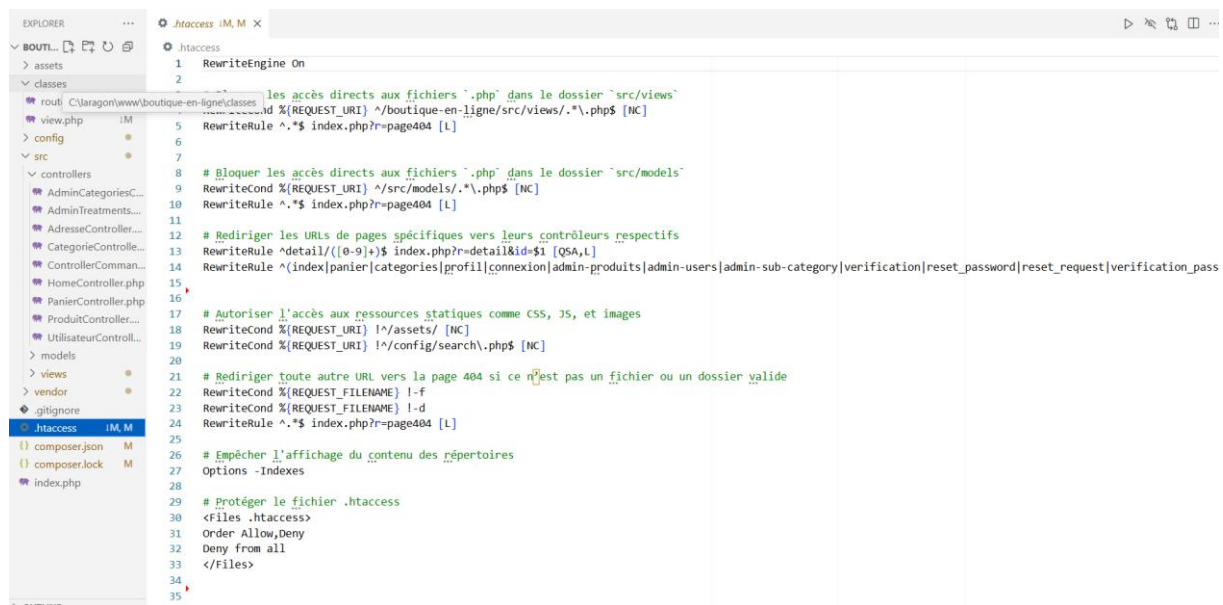
Pour le backend, nous avons décidé de créer un système de route personnalisé avec un fichier routeur.php qui a servi à la redirection des vues en fonction de la vue demandé et une réécriture des URL via .htaccess

Routeur.php :



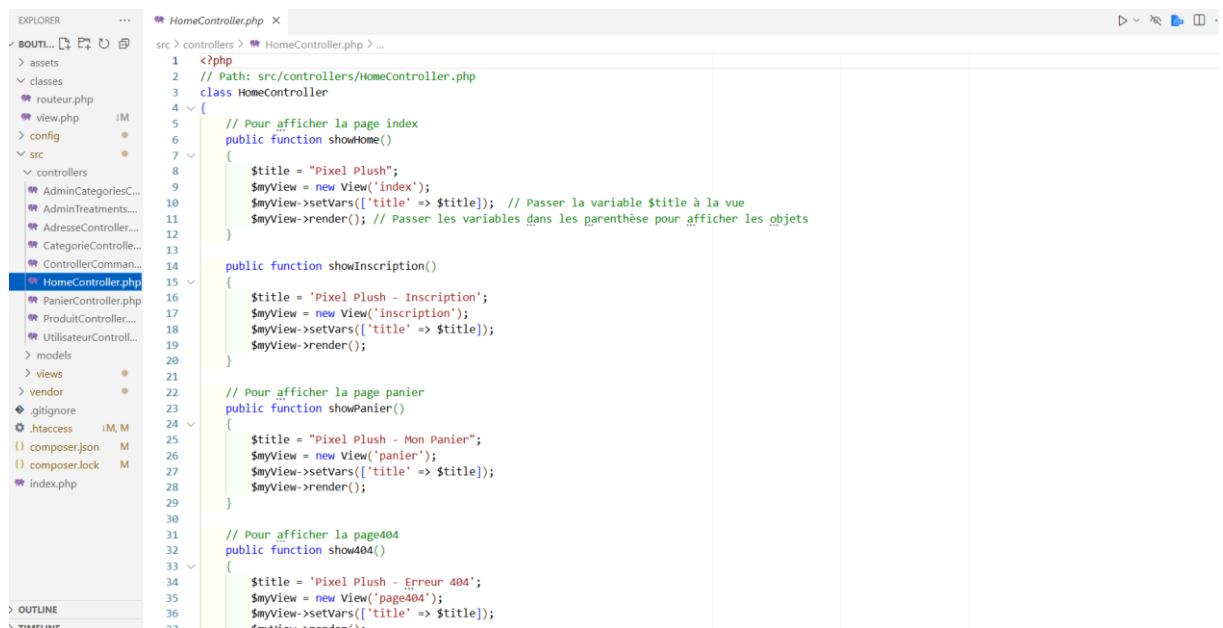
```
1 <?php
2 // classe Routeur qui gère les routes et appelle les controllers
3 class Routeur
4 {
5     private $request;
6
7     // Tableau de correspondance entre les routes et les controllers
8     private $routes = [
9         "index" => ["controller" => 'HomeController', "method" => 'showHome'],
10        "panier" => ["controller" => 'HomeController', "method" => 'showPanier'],
11        "page404" => ["controller" => 'HomeController', "method" => 'show404'],
12        "categories" => ["controller" => 'HomeController', "method" => 'showCategories'],
13        "profil" => ["controller" => 'HomeController', "method" => 'showProfil'],
14        "connexion" => ["controller" => 'HomeController', "method" => 'showConnexion'],
15        "verification" => ["controller" => 'HomeController', "method" => 'showVerification'],
16        "detail" => ["controller" => 'HomeController', "method" => 'showDetail'],
17        "admin-produits" => ["controller" => 'HomeController', "method" => 'showAdminProduits'],
18        "admin-users" => ["controller" => 'HomeController', "method" => 'showAdminUsers'],
19        "admin-sub-category" => ["controller" => 'HomeController', "method" => 'showAdminCategory'],
20        "reset_password" => ["controller" => 'HomeController', "method" => 'showResetPassword'],
21        "reset_request" => ["controller" => 'HomeController', "method" => 'showResetRequest'],
22        "verification_password" => ["controller" => 'HomeController', "method" => 'showPasswordVerify'],
23        "confirmation" => ["controller" => 'HomeController', "method" => 'showConfirmation']
24    ];
25
26     public function __construct($request)
27     {
28         $this->request = $request;
29     }
30
31     public function renderController()
32     {
33         $request = $this->request;
34
35         // Vérifie si la requête (URL) existe dans les routes
36         if (array_key_exists($request, $this->routes)) {
37             // Récupère le contrôleur et la méthode associés à la route
38             $controller = $this->routes[$request]['controller'];
39             $method = $this->routes[$request]['method'];
40
41             // Vérifie si le contrôleur existe avant de l'instancier
42             if (class_exists($controller)) {
43                 $currentController = new $controller();
44                 // Vérifie si la méthode existe dans le contrôleur
45                 if (method_exists($currentController, $method)) {
46                     $currentController->$method();
47                 } else {
48                     // Si la méthode n'existe pas, redirige vers la page 404
49                     $this->redirect404();
50                 }
51             } else {
52                 // Si le contrôleur n'existe pas, redirige vers la page 404
53                 $this->redirect404();
54             }
55         } else {
56             // Si la route n'existe pas, redirige vers la page 404
57             $this->redirect404();
58         }
59     }
60 }
```

.htaccess :



```
1 RewriteEngine On
2
3 # Les accès directs aux fichiers '.php' dans le dossier 'src/views'
4 RewriteCond %{REQUEST_URI} ^/boutique-en-ligne/src/views/.*\.php$ [NC]
5 RewriteRule ^.*$ index.php?r=page404 [L]
6
7
8 # Bloquer les accès directs aux fichiers '.php' dans le dossier 'src/models'
9 RewriteCond %{REQUEST_URI} ^/src/models/.*\.php$ [NC]
10 RewriteRule ^.*$ index.php?r=page404 [L]
11
12 # Rediriger les URLs de pages spécifiques vers leurs contrôleurs respectifs
13 RewriteRule ^detail/([0-9]+)$ index.php?r=detail&id=$1 [QSA,L]
14 RewriteRule ^([index|panier|categories|profil|connexion|admin-produits|admin-users|admin-sub-category|verification|reset_password|reset_request|verification_pass])$ index.php?r=$1 [QSA,L]
15
16
17 # Autoriser l'accès aux ressources statiques comme CSS, JS, et images
18 RewriteCond %{REQUEST_URI} !^/assets/ [NC]
19 RewriteCond %{REQUEST_URI} !^/config/search\.php$ [NC]
20
21 # Rediriger toute autre URL vers la page 404 si ce n'est pas un fichier ou un dossier valide
22 RewriteCond %{REQUEST_FILENAME} !-f
23 RewriteCond %{REQUEST_FILENAME} !-d
24 RewriteRule ^.*$ index.php?r=page404 [L]
25
26 # Empêcher l'affichage du contenu des répertoires
27 Options -Indexes
28
29 # Protéger le fichier .htaccess
30 <Files .htaccess>
31 Order Allow,Deny
32 Deny from all
33 </Files>
34
35
```

HomeController : fichier qui fait le lien entre le titre de la page l'url et le rendu



```
1 <?php
2 // Path: src/controllers/HomeController.php
3 class HomeController
4 {
5     // Pour afficher la page index
6     public function showHome()
7     {
8         $title = "Pixel Plush";
9         $myView = new View('index');
10        $myView->setVars(['title' => $title]); // Passer la variable $title à la vue
11        $myView->render(); // Passer les variables dans les parenthèse pour afficher les objets
12    }
13
14    public function showInscription()
15    {
16        $title = 'Pixel Plush - Inscription';
17        $myView = new View('inscription');
18        $myView->setVars(['title' => $title]);
19        $myView->render();
20    }
21
22    // Pour afficher la page panier
23    public function showPanier()
24    {
25        $title = "Pixel Plush - Mon Panier";
26        $myView = new View('panier');
27        $myView->setVars(['title' => $title]);
28        $myView->render();
29    }
30
31    // Pour afficher la page 404
32    public function show404()
33    {
34        $title = 'Pixel Plush - Erreur 404';
35        $myView = new View('page404');
36        $myView->setVars(['title' => $title]);
37        $myView->render();
38    }
39 }
```

B. Frontend

Pour le frontend lors de la redirection des pages dans les links, nous avons juste réécrit les noms des vues nécessaires.

Dans cette exemple nous redirigeons l'utilisateur vers sa page profil, une fois connecté :

```
}  
} elseif ($action === 'login') {  
    if (!empty($email) && !empty($password)) {  
        $userData = $user->userConnexion($email, $password);  
        if ($userData) {  
            $_SESSION['user'] = $userData;  
            header('Location: profil');  
            exit();  
        } else {  
            $error = "Email ou mot de passe incorrect.";  
        }  
    } else {  
        $error = "Tous les champs sont obligatoires.";  
    }  
}
```


RÉALISATIONS PERSONELLES

Au cours de ce projet. J'ai eu l'occasion de travailler sur plusieurs domaines. J'ai évidemment travaillé sur les parties back et front de l'application, mais j'ai eu l'occasion de découvrir la gestion de projet avec GIT, je me suis également occupé du déploiement de la base de données distante commune qui a servi au bon déroulement du projet.

Je me suis en partie occupé du backend pour aider à la création d'une partie des fonctionnalités qui allez servir à la réalisation du projet, la fonction principale sur lequel j'ai travaillé c'est l'ajout de nouveau produit avec l'ajout d'image dans la base de données :

1. Ajout de produit

➤ BACK

a. Création du modèle :

J'ai commencé par créer un modèle chargé de préparer et de sécuriser les données avant leur insertion dans la base de données. Cela inclut le nettoyage des champs ainsi que la gestion de la liaison entre le produit et l'image

```
public function addProduct($nom, $description, $prix, $quantite, $image, $categorie, $sous_categorie)
{
    try {
        $requete = $this->connexion->prepare("INSERT INTO Produit
        (nom, description, prix, quantite, image, date_ajout, id_categorie, id_sous_categorie)
        VALUES (?, ?, ?, ?, ?, NOW(), ?, ?)");
        $requete->execute([$nom, $description, $prix, $quantite, $image, $categorie, $sous_categorie]);
    } catch (Exception $e) {
        throw new Exception("Erreur lors de l'ajout du produit : " . $e->getMessage());
    }
}
```

b. Mise en place du contrôleur :

Ensuite, j'ai développé un contrôleur dédié à la gestion des requêtes liées à l'ajout de produit. Ce contrôleur contient la méthode responsable de l'appel au modèle pour l'insertion dans la base de données.

```
public function addProduct($nom, $description, $prix, $quantite, $image, $categorie, $sous_categorie)
{
    $this->modelProduit->addProduct($nom, $description, $prix, $quantite, $image, $categorie, $sous_categorie);
}
```

c. Validation côté vue (admin-produit) :

Dans la vue dédiée à l'administration des produits, j'ai intégré une fonction de validation du fichier image uploadé. Cette vérification permet de s'assurer :

- Que le fichier est bien une image,
- Qu'il respecte les formats autorisés
- Que sa taille ne dépasse pas la limite définie.

```
25 // Fonction de validation pour l'upload de fichiers
26 function validation($file)
27 {
28     $error = false;
29     $errorMessage = '';
30     $filename = '';
31
32     if (!empty($file["name"])) {
33         $filename = $file['name'];
34         $dossier_temporaire = $file['tmp_name'];
35         $dossier_upload = __DIR__ . "/../../assets/images/" . $filename;
36
37         $extension_fichier = strtolower(pathinfo($filename, PATHINFO_EXTENSION));
38         $extensions_autorisees = array("jpg", "jpeg", "png", "JPG", "JPEG", "PNG", "webp", "WEBP");
39
40         if (in_array($extension_fichier, $extensions_autorisees)) {
41             $errorMessage = "L'extension n'est pas autorisée. Utilisez JPG, JPEG ou PNG.";
42             $error = true;
43         }
44
45         if (!$error && !move_uploaded_file($dossier_temporaire, $dossier_upload)) {
46             $errorMessage = "Une erreur est survenue pendant l'upload du fichier";
47             $error = true;
48         }
49     }
50
51     return ['success' => !$error, 'message' => $errorMessage, 'filename' => $filename];
52 }
53
54 // Traitement de l'ajout de produit
55 if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['action']) && $_POST['action'] === 'addProduct') {
56     $data = [
57         'nom' => $_POST['nom'],
58         'description' => $_POST['description'],
59         'prix' => $_POST['prix'],
60         'quantite' => $_POST['quantite'],
61         'categorie' => $_POST['categorie'],
62         'sous_categorie' => $_POST['sous_categorie']
63     ];
64
65     // Validation de l'image
66     $imageResult = ['success' => true];
67     if (isset($_FILES['image'])) { ...
68     }
69
70     if ($imageResult['success']) {
71         try { ...
72         } catch (Exception $e) {
73             $_SESSION['message'] = 'Erreur : ' . addslashes($e->getMessage());
74             header('Location: ' . $_SERVER['PHP_SELF']); // Redirection pour éviter le rechargement
75             exit();
76         }
77     } else {
78         $_SESSION['message'] = 'Erreur : ' . addslashes($imageResult['message']);
79         header('Location: ' . $_SERVER['PHP_SELF']); // Redirection pour éviter le rechargement
80         exit();
81     }
82 }
83
```

Ainsi, l'ensemble du processus du traitement en base de données à la validation du fichier a été pensé pour garantir la sécurité et l'intégrité des données lors de l'ajout d'un nouveau produit sur la plateforme

- FRONT

1. Ajout d'un bouton d'action :

Un bouton « Ajouter un produit » a été intégré à la page. Celui-ci déclenche l'ouverture d'une modale permettant de saisir les informations du nouveau produit.

```
<!-- Bouton pour ajouter un produit, qui ouvre une modal -->
<button id="btn-add-product" class="btn btn-ajouter">Ajouter un produit</button>
```

2. Affichage dynamique de la modale :

Lors du clic sur le bouton, une modale ouvrante s'affiche automatiquement grâce à un script JavaScript. Cette modale contient les éléments nécessaires au produit

```
<!-- Modal pour ajouter un produit -->
<div id="addProductModal" class="modal" style="display: none;">
  <div class="modal-content">
    <span class="close" onclick="closeModal('addProductModal')">&times;</span>
    <h3>Ajouter un nouveau produit</h3>
    <form id="addProductForm" class="flex column vertical-center" action="" method="POST" enctype="multipart/form-data">
      <input type="hidden" name="action" value="addProduct">
      <label for="productName">Nom du produit :</label>
      <input type="text" name="nom" required>

      <label for="productDescription">Description :</label>
      <textarea name="description" required></textarea>

      <label for="productPrice">Prix :</label>
      <input type="number" id="prix" name="prix" step="0.01" required>

      <label for="productQuantity">Quantité :</label>
      <input type="number" id="quantite" name="quantite" required>

      <label for="productImage">Image :</label>
      <input type="file" name="image" required accept=".jpg,.jpeg,.png,.webp">

      <label for="productCategory">Catégorie :</label>
      <select name="categorie" required>
        <?php foreach ($categories as $categorie): ?>
          <option value="<?php echo $categorie['id_categorie']; ?>"><?php echo htmlspecialchars($categorie['nom_p']); ?></option>
        <?php endforeach; ?>
      </select>

      <label for="productSubCategory">Sous-catégorie :</label>
      <select name="sous_categorie" required>
        <?php foreach ($sousCategories as $sousCategorie): ?>
          <option value="<?php echo $sousCategorie['id_sousCategorie']; ?>"><?php echo htmlspecialchars($sousCategorie['nom_sc']); ?></option>
        <?php endforeach; ?>
      </select>
    </form>
  </div>
</div>
```

3. Gestion du comportement avec Javascript ;

Le script permet de gérer l'apparition et la fermeture de la modale, réinitialiser les champs après soumission.

```
<script>
  // Fonction pour ouvrir et fermer les modals
  function openModal(modalId) {
    document.getElementById(modalId).style.display = "block";
  }

  function closeModal(modalId) {
    document.getElementById(modalId).style.display = "none";
  }

  // Ouvrir la modal pour ajouter un produit
  document.getElementById('btn-add-product').onclick = function() {
    openModal('addProductModal');
  };
</script>
```

VULNÉRABILITÉS DE SÉCURITÉ ET VEILLE

1. Veille technologique

A. Nodemailer et SMTP

Nodemailer est une bibliothèque pour Node.js qui permet d'envoyer des emails en utilisant le protocole SMTP. Parmi ses fonctionnalités clés, on retrouve l'envoi d'emails de confirmation, la récupération de mot de passe, les notifications, et les newsletters.

Dans le cadre de mon projet, j'ai utilisé cet outil pour permettre aux utilisateurs de me contacter directement via mon portfolio.

Je me suis aidé de ce site pour m'aider dans cette réalisation :

<https://mailtrap.io/fr/blog/sending-emails-with-nodemailer/>

B. Astro

Astro est un framework JavaScript open-source conçu pour la création de sites web rapides, modulaires et axés sur le contenu.

Principales caractéristiques :

- Compatibilité avec divers frameworks : Astro permet d'utiliser des composants provenant de différents frameworks tels que React, Vue, Solid et bien d'autres
- Optimisation des performances : Grâce à sa structure légère, Astro permet de créer des sites web performants, adaptés au SEO et à l'expérience utilisateur.

Dans le cadre de mon portfolio, j'ai choisi d'utiliser Astro pour sa capacité à générer des sites rapides et modulaires. Cette approche permet d'offrir une expérience utilisateur fluide tout en facilitant la maintenance et l'évolution du site

Documentation officielle d'Astro : <https://astro.build/>

2. Veille de sécurité

A. JWT et cryptage du mot de passe utilisateur

Dans le cadre de la réalisation de mon blog avec hono, j'ai choisi d'utiliser jsonwebtoken JWT, me posant la question du chiffrement du mot de passe et de sécurité de l'utilisateur dans la base de données.

Après avoir effectué des recherches, j'ai trouvé une façon de stocker les mots de passe en utilisant la librairie bcrypt

<https://blog.logrocket.com/password-hashing-node-js-bcrypt/>

B. Sécurisation des requêtes

Pour éviter les attaques par injection SQL, j'ai utilisé les requêtes préparées avec PDO,

J'ai mis en place dans les formulaires htmlspecialchars() pour me protéger des attaques XSS

3. Processus de recherche et traduction

Pour effectuer mes recherches, j'adopte en grande partie le processus de recherche suivant :

- Documentation officielle du langage et de l'outils que je vous utilisé
- Recherche de vidéo Youtube, Linkedin, Openclassroom, Grafikart ou autres
- Consultation de forum Stack overflow, blog

Il arrive que certaines des recherches que j'effectue soit en anglais car la plus part des site comme stackoverflow sont uniquement en anglais et je traduis les commentaires et les questions avec des outils comme google traduction quand je ne comprends pas et demande l'avis d'autre personne pour m'aider à comprendre.

CONCLUSION

En conclusion, je tiens à exprimer ma gratitude envers Lucy, Ryan, Lucas et Antony, mes collègues de ce projet. L'échange d'idées et de techniques, enrichi par nos expériences passées, a été déterminant pour définir rapidement et efficacement la méthodologie de travail qui nous a guidés tout au long de ce mois de création. Nous avons pu répondre aux attentes de notre Product Owner en nous appuyant sur ces bases solides de gestion. L'établissement de ces bonnes pratiques dès le début a favorisé la réflexion collective et la co-construction de solutions pertinentes.

ANNEXE

Les repos du projet sont accessibles à cette adresse :

PixelPlush : <https://github.com/YoussefGhollamallah/boutique-en-ligne>

BlogNoSQL : <https://github.com/YoussefGhollamallah/reactethono>

PixelPlush a été mis en ligne sur Hostinger sur : <https://pixelplush.shop>