

DOSSIER PROFESSIONNEL

Juillet 2023

Présentation du projet Boutique en ligne.

Titre RNCP Niveau 5 : Développeur web et mobile

1

I. Introduction

Le présent dossier a pour objectif de présenter en détail le projet de boutique en ligne **Théophile**. Ce projet, réalisé dans le cadre de ma formation en tant que développeur web et mobile, consiste en la création d'une boutique en ligne spécialisée dans la vente de thé en vrac en utilisant le modèle de dropshipping.

II. Principes

Le projet Théophile vise à valider mes compétences en développant une boutique en ligne complète, dotée de fonctionnalités de qualité. En tant que développeur web et mobile, il est essentiel de maîtriser les techniques de développement nécessaires pour créer une expérience utilisateur optimale et une gestion efficace des produits.

III. Planification

Afin de mener à bien ce projet, j'ai établi une planification détaillée sur une période de 2 mois. Cette planification inclut les différentes étapes du développement, les objectifs à atteindre et les délais à respecter. Elle me permettra de travailler de manière organisée et efficiente.

IV. Organisation (Structurer un projet et penser son architecture)

La première étape de ce projet consiste à organiser de manière efficace les différents éléments de développement. J'ai opté pour l'utilisation de l'application **Trello** qui me permet de définir les grands blocs de travail, de créer des tableaux et des tickets pour chaque sous-partie ou sprint. Cette approche m'aide à suivre une organisation chronologique logique et à garantir que les étapes du projet se succèdent de manière cohérente.

V. Utilisation de wireframes (Maquettage de projet / application)

Pour avoir une vision claire de l'interface utilisateur, j'ai utilisé l'application Figma pour créer des wireframes. J'ai développé deux prototypes : un pour la version desktop et un autre pour la version mobile du site. J'ai commencé par créer des wireframes à basse fidélité, puis j'ai affiné le design avec des wireframes à haute fidélité.

VI. Les technos utilisées

1. PHP (Hypertext Preprocessor) :

- PHP est un langage de programmation côté serveur très populaire, principalement utilisé pour développer des applications web dynamiques.
- Il m'a permis de créer des pages web générées dynamiquement en se connectant à une base de données, en manipulant des données et en effectuant des opérations logiques.

2. JavaScript :

- JavaScript est un langage de programmation côté client qui permet d'ajouter des fonctionnalités interactives et dynamiques aux pages web.
- Il m'a permis de créer des fonctionnalités telles que des effets visuels, des validations de formulaire en temps réel et des interactions avec l'utilisateur.

3. HTML (Hypertext Markup Language) :

- HTML est le langage de balisage utilisé pour structurer et créer le contenu d'une page web.
- Il définit la structure de base de la page en utilisant des balises et permet d'inclure des images, des liens, des formulaires et bien plus encore.

4. CSS (Cascading Style Sheets) :

- CSS est utilisé pour styliser et mettre en forme les éléments HTML d'une page web.

- Il me permet de définir la mise en page, les couleurs, les polices, les marges et autres propriétés visuelles des éléments de la page.

5. Bootstrap :

- Bootstrap est un framework CSS et JavaScript open-source qui facilite la conception réactive et le développement de sites web adaptatifs.
- Il offre une collection de composants prêts à l'emploi, de mises en page et de styles pour créer rapidement des sites web esthétiques et réactifs.

6. PHPMyAdmin :

- PHPMyAdmin est une application web qui me permet de gérer facilement ma base de données MySQL à l'aide d'une interface graphique conviviale.
- j'ai effectué des opérations de gestion de base de données telles que la création de tables, l'insertion de données, les requêtes SQL et la gestion des utilisateurs.

7. SQL (Structured Query Language) :

- SQL est un langage de requête utilisé pour communiquer avec une base de données relationnelle.
- Il me permet de récupérer, insérer, mettre à jour et supprimer des données dans ma base de données.

8. Trello :

- Trello est un outil de gestion de projet en ligne basé sur des tableaux, des listes et des cartes.
- je l'ai utilisé pour organiser mes tâches, suivre l'avancement du projet, attribuer des responsabilités et collaborer avec d'autres membres de l'équipe.

9. Git et GitHub :

- Git est un système de contrôle de version distribué qui me permet de gérer et de suivre les changements dans mon code source.
- GitHub est une plateforme d'hébergement de code qui prend en charge le stockage et la collaboration autour de projets Git.

En utilisant cette combinaison de technologies, j'ai pu construire une application web dynamique et réactive avec des fonctionnalités avancées pour gérer ma boutique en ligne de thé en vrac. j'ai exploité la puissance du langage PHP côté serveur, le dynamisme du JavaScript côté client, la structure du HTML et les styles du CSS pour offrir une expérience utilisateur améliorée. La gestion de la base de données a été assurée par PHPMyAdmin et SQL, et j'ai utilisé des outils tels que Trello pour m'organiser et Git/GitHub pour gérer mon code.

VII. Développement front-end(Programmation Orientée Objet:utilisation des classes)

Dans cette partie, j'ai défini une architecture pour le développement front-end de mon projet. J'ai choisi d'adopter le modèle MVC (Modèle-Vue-Contrôleur) et j'ai utilisé Composer pour gérer les dépendances et Alto-Routeur pour le système de routage avec la méthode match. J'ai expliqué en détail ces deux composants essentiels de mon architecture et leur utilité dans le développement front-end.

1. L'architecture MVC

a. Le Modèle (Model) :

Il représente la logique métier et les données de l'application. Il gère la manipulation, le stockage et la récupération des données. Le modèle ne se soucie pas de l'interface utilisateur ou de l'affichage, il se concentre uniquement sur la gestion des données.

b. La Vue (View) :

C'est la partie responsable de l'interface utilisateur. Elle affiche les données au format approprié pour les utilisateurs. La vue se contente d'afficher les informations fournies par le modèle et ne gère pas la logique métier.

c. Le Contrôleur (Controller) :

C'est l'intermédiaire entre le modèle et la vue. Il reçoit les demandes de l'utilisateur et interagit avec le modèle en récupérant les données nécessaires. Ensuite, il transmet les données au bon format à la vue pour qu'elle les affiche à l'utilisateur. Le contrôleur gère également les actions de l'utilisateur et met à jour le modèle en conséquence.

Explication du fonctionnement :

Lorsqu'un utilisateur interagit avec une application basée sur l'architecture MVC, le processus de fonctionnement est le suivant :

- L'utilisateur effectue une action sur l'interface utilisateur (par exemple, clique sur un bouton).
- L'action est interceptée par le contrôleur qui comprend la demande de

l'utilisateur.

- Le contrôleur traite la demande en récupérant les données nécessaires du modèle.

d. Le modèle exécute les opérations métier appropriées pour récupérer ou mettre à jour les données dans la base de données ou tout autre support de stockage.

e. Le modèle renvoie les données demandées au contrôleur.

f. Le contrôleur transmet les données au format approprié à la vue.

g. La vue affiche les données à l'utilisateur.

h. L'utilisateur voit le résultat de son action sur l'interface utilisateur.

L'architecture MVC offre plusieurs avantages :

- Elle facilite la séparation des préoccupations (Separation of Concerns), ce qui signifie que chaque composant a son propre rôle et ne se mélange pas avec les autres.

- Elle rend l'application plus facile à maintenir, car les modifications peuvent être apportées à un composant sans affecter les autres.

- Elle facilite la collaboration entre les développeurs, car ils peuvent travailler indépendamment sur chaque composant.

- Elle favorise une meilleure organisation du code, ce qui améliore la lisibilité et la qualité du logiciel.

En utilisant l'architecture MVC, les développeurs peuvent créer des applications robustes, évolutives et bien structurées, tout en assurant une meilleure expérience utilisateur et une maintenance plus aisée de l'application

2. Composeur

Composer est un gestionnaire de dépendances pour PHP, et son utilité dans mon application est de faciliter la gestion des bibliothèques externes et des packages dont mon projet a besoin pour fonctionner correctement.

a. Gestion des dépendances :

Lorsque je développe une application PHP, il est courant que j'utilise des bibliothèques tierces ou des packages open-source pour ajouter des fonctionnalités spécifiques à mon projet. Cependant, gérer manuellement ces dépendances peut être fastidieux et complexe. C'est là qu'intervient Composer. Il me permet de spécifier les bibliothèques dont mon application dépend et gère automatiquement leur installation, leur mise à jour et leur suppression.

b. Gestion des versions :

Composer me permet de spécifier des contraintes de version pour chaque dépendance dans mes fichiers `composer.json`. Je peux définir des versions spécifiques ou utiliser des expressions comme `"^1.0"` pour indiquer que mon application dépend d'une version supérieure à 1.0, mais inférieure à 2.0. Composer gère intelligemment les conflits de versions entre les dépendances pour s'assurer que tout fonctionne correctement ensemble.

3. Alto-Routeur Intégration facile avec l'autoload

Composer génère automatiquement un fichier d'autoload pour mon application, ce qui signifie que vous n'avez pas besoin d'inclure manuellement chaque fichier de classe des bibliothèques dans mon code. Je peux simplement utiliser `require 'vendor/autoload.php'`, et Composer se charge de charger automatiquement les classes nécessaires au fur et à mesure de leur utilisation dans mon application.

En résumé, Composer simplifie grandement la gestion des dépendances de mon application PHP en automatisant le processus d'installation, de mise à jour et de suppression des bibliothèques tierces. Il facilite également la collaboration avec d'autres développeurs et contribue à maintenir un code propre et bien organisé. Utiliser Composer dans mon projet est donc une pratique essentielle pour développer efficacement et maintenir mon application à jour et fonctionnelle.

VIII. Réalisation de l'interface utilisateur (Faire de l'asynchrone avec JS)

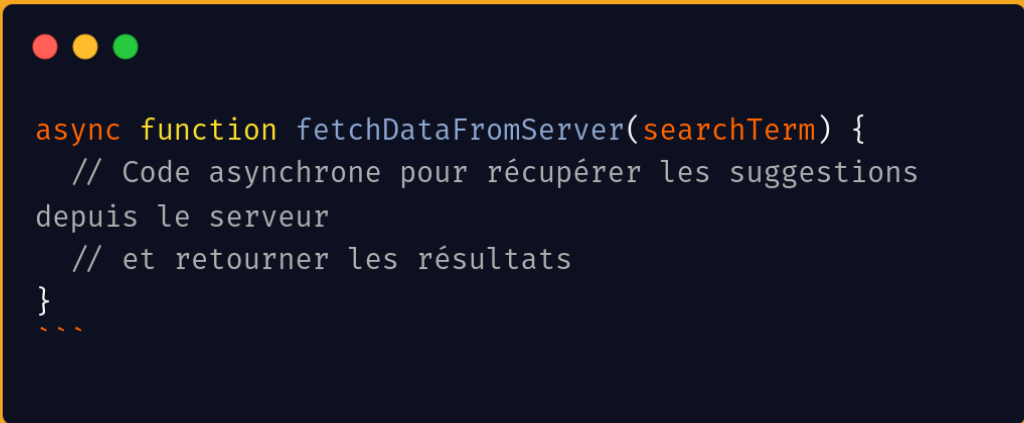
L'interface utilisateur de la boutique en ligne Théophile a été réalisée en mettant l'accent sur les fonctionnalités asynchrones grâce à JavaScript. J'ai mis en place des composants tels qu'une barre de recherche avec auto-complétion, l'ajout de produits au panier, la gestion des comptes utilisateurs, etc. J'ai utilisé le CSS et le JS pour améliorer l'expérience utilisateur et créer un rendu attrayant côté front-end.

Le code asynchrone en JavaScript est utilisé pour effectuer des opérations non bloquantes, ce qui signifie que le reste du code peut continuer à s'exécuter pendant que l'opération asynchrone est en cours. Cela est particulièrement utile pour des tâches qui peuvent prendre du temps, telles que les requêtes réseau ou les opérations d'E/S (entrée/sortie) qui pourraient bloquer l'exécution du code de manière synchrone.

Lorsque j'implémente l'autocomplétion dans une barre de recherche, je peux utiliser des appels asynchrones pour récupérer les suggestions de résultats en temps réel depuis le serveur tout en permettant à l'utilisateur de continuer à interagir avec l'interface utilisateur de manière fluide.

1. Utilisation d'une fonction asynchrone :

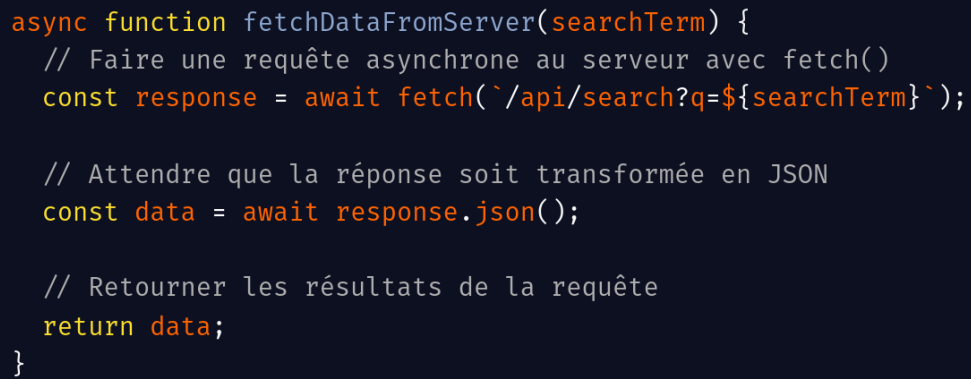
Pour rendre une fonction asynchrone, vous pouvez utiliser le mot-clé `async` avant le mot-clé `function`. Cela indique que la fonction contient du code asynchrone et qu'elle peut contenir des expressions `await`. Lorsque vous utilisez `async`, la fonction retourne automatiquement une promesse qui sera résolue avec la valeur renvoyée par la fonction ou rejetée avec une erreur le cas échéant.

A code snippet is displayed within a dark-themed editor window with three colored window control buttons (red, yellow, green) in the top-left corner. The code defines an asynchronous function named `fetchDataFromServer` that takes a `searchTerm` parameter. The function body contains two comments: `// Code asynchrone pour récupérer les suggestions depuis le serveur` and `// et retourner les résultats`. The function is enclosed in curly braces, and there are three dots at the end of the line.

```
async function fetchDataFromServer(searchTerm) {  
  // Code asynchrone pour récupérer les suggestions  
  depuis le serveur  
  // et retourner les résultats  
}
```

2. Utilisation de l'expression `await` :

L'expression `await` est utilisée à l'intérieur d'une fonction asynchrone pour attendre la résolution d'une promesse. Cela signifie que le code qui suit l'expression `await` ne sera exécuté qu'une fois que la promesse sera résolue (ou rejetée en cas d'erreur). Cela permet d'attendre que les données soient récupérées avant de continuer à traiter les résultats.



```
async function fetchDataFromServer(searchTerm) {  
  // Faire une requête asynchrone au serveur avec fetch()  
  const response = await fetch(`/api/search?q=${searchTerm}`);  
  
  // Attendre que la réponse soit transformée en JSON  
  const data = await response.json();  
  
  // Retourner les résultats de la requête  
  return data;  
}
```

3. Utilisation de l'autocomplétion avec la barre de recherche :

Lorsque l'utilisateur entre des caractères dans la barre de recherche, il peut appeler la fonction asynchrone `fetchDataFromServer` pour récupérer les suggestions de résultats en fonction du terme de recherche.

```
const searchBar = document.getElementById('searchBar');

searchBar.addEventListener('input', async (event) => {
  const searchTerm = event.target.value;

  try {
    // Appeler la fonction asynchrone pour récupérer les suggestions depuis le serveur
    const suggestions = await fetchDataFromServer(searchTerm);

    // Mettre à jour l'interface utilisateur avec les suggestions
    displaySuggestions(suggestions);
  } catch (error) {
    console.error('Erreur lors de la récupération des suggestions :', error);
  }
})
```

Dans cet exemple, lorsque l'utilisateur saisit du texte dans la barre de recherche (`searchBar`), un événement d'entrée (`input`) est déclenché. À ce moment, il appelle la fonction asynchrone `fetchDataFromServer` avec le terme de recherche entré par l'utilisateur. Une fois que la fonction asynchrone se résout avec les résultats, il affiche les suggestions à l'utilisateur en utilisant la fonction `displaySuggestions`.

Le code asynchrone dans cet exemple permet de récupérer les données depuis le serveur de manière non bloquante, offrant ainsi une expérience utilisateur plus fluide lors de l'utilisation de l'autocomplétion dans la barre de recherche.

4. L'ajout de produits dans le panier :

C'est une fonctionnalité courante dans les boutiques en ligne. Cela permet à l'utilisateur de sélectionner des articles qu'il souhaite acheter et de les stocker temporairement

dans son panier avant de passer à la phase de paiement.

Sur l'interface utilisateur du site, j'ai inclus un bouton "Ajouter au panier" ou un icône de panier à côté de chaque produit. Lorsque l'utilisateur clique sur ce bouton, il doit capturer l'action et effectuer l'ajout du produit au panier.

Chaque produit doit avoir des informations essentielles telles que son ID, son nom, son prix, sa quantité disponible, son image, etc. Lorsque l'utilisateur clique sur Ajouter au panier, il doit récupérer ces informations du produit sélectionné.

Une fois les informations du produit récupérées, elles doivent être stockées dans la session de l'utilisateur ou dans un cookie pour garder une trace des produits ajoutés au panier. Une autre option consiste à stocker les données du panier côté serveur dans la base de données, associées à l'ID de l'utilisateur.

Sur les pages pertinentes du site, je dois afficher le contenu du panier de l'utilisateur, lui montrant les articles qu'il a sélectionnés et leur quantité. Je peux également afficher le total des prix des articles dans le panier.

Si l'utilisateur ajoute le même produit plusieurs fois, il doit mettre à jour la quantité du produit dans le panier au lieu de le dupliquer. Cela signifie que je dois gérer les quantités pour chaque article dans le panier.

L'utilisateur doit avoir la possibilité de modifier le contenu de son panier en augmentant ou diminuant la quantité des articles. J'ai également inclus une option pour supprimer complètement un produit du panier si l'utilisateur souhaite l'enlever.

Lorsque l'utilisateur a terminé d'ajouter les produits souhaités dans son panier, j'ai inclus un bouton "Paiement" ou "Passer à la caisse" qui permettra à l'utilisateur de passer à la phase de paiement et de finaliser sa commande.

5. La gestion des comptes utilisateur :

La gestion des comptes utilisateur est une fonctionnalité essentielle dans une boutique en ligne. Elle permet aux utilisateurs de créer un compte, de se connecter et d'accéder à

des fonctionnalités spécifiques, telles que l'ajout de produits au panier, la gestion des informations personnelles, la consultation de l'historique des commandes, etc. Voici comment cette fonctionnalité peut être définie :

Les utilisateurs doivent avoir la possibilité de créer un compte sur le site en fournissant des informations telles que leur nom, leur adresse e-mail et un mot de passe sécurisé.

Une fois que les utilisateurs ont créé un compte, ils doivent pouvoir se connecter en utilisant leur adresse e-mail et leur mot de passe. Je dois mettre en œuvre un système d'authentification sécurisé pour vérifier les informations d'identification de l'utilisateur et lui permettre d'accéder à son compte.

Une fois connecté, l'utilisateur doit avoir accès à un profil où il peut consulter et modifier ses informations personnelles, telles que son nom, son adresse, son numéro de téléphone, etc. J'ai prévu également d'inclure une option permettant à l'utilisateur de télécharger une photo de profil.

J'ai assuré de stocker les informations sensibles, telles que les mots de passe, de manière sécurisée en utilisant des techniques de hachage et de salage pour protéger les données des utilisateurs contre toute intrusion.

Enfin j'ai inclus une option de déconnexion qui permet à l'utilisateur de se déconnecter de son compte en toute sécurité lorsqu'il a terminé ses activités sur le site.

NB: LES FONCTIONNALITÉS QUE J'AI PRÉVUS D'IMPLÉMENTER PLUS TARD:

Il est important d'inclure une politique de confidentialité détaillée pour informer les utilisateurs de la manière dont leurs données personnelles sont collectées, stockées et utilisées sur le site. Je me suis assuré de respecter les réglementations en matière de protection des données, telles que le RGPD (Règlement général sur la protection des données).

Mettre en œuvre des mesures de sécurité telles que la limitation des tentatives de connexion infructueuses, la vérification en deux étapes (2FA) pour renforcer la sécurité

des comptes utilisateurs.

Suppression de compte : Fournir une option pour que les utilisateurs puissent supprimer leur compte s'ils le souhaitent. Assurez-vous de supprimer toutes les données personnelles de manière appropriée conformément aux politiques de confidentialité.

La gestion des comptes utilisateurs est un aspect crucial de toute boutique en ligne, et la mise en œuvre d'une gestion sécurisée et conviviale des comptes garantit une meilleure expérience utilisateur et une meilleure confiance dans le site.

6. JavaScript (JS) et CSS :

JavaScript et CSS sont deux technologies essentielles pour améliorer l'expérience utilisateur d'un site web. Ils permettent d'ajouter des fonctionnalités interactives, des animations, des améliorations visuelles et de rendre le site plus réactif.

JS permet d'ajouter des interactions dynamiques sur mon site. Par exemple, j'ai mis en œuvre une autocomplétion dans la barre de recherche. Ces interactions facilitent la navigation et l'engagement des utilisateurs.

Grâce à JS, j'ai pu valider les formulaires côté client avant de les envoyer au serveur. Cela permet de fournir un retour instantané aux utilisateurs sur d'éventuelles erreurs de saisie avant qu'ils soumettent le formulaire, ce qui réduit les risques d'erreurs et améliore l'efficacité du processus de saisie.

CSS permet d'ajouter des effets visuels et des animations qui rendent le site plus attractif et engageant pour les utilisateurs. Les transitions fluides, les survols d'images, les animations de défilement, etc., ajoutent une touche esthétique au site et renforcent son attractivité.

Avec CSS j'ai aussi rendu mon site responsive, c'est-à-dire qu'il s'adapte automatiquement à différents écrans et appareils (ordinateurs de bureau, tablettes, smartphones). Un design responsive assure une meilleure expérience utilisateur sur tous les types de dispositifs, ce qui est essentiel dans un monde où la navigation mobile est de plus en plus répandue.

En combinant judicieusement les fonctionnalités interactives de JavaScript avec les styles élégants et réactifs de CSS, j'ai pu offrir aux utilisateurs une expérience utilisateur fluide, attrayante et efficace, ce qui contribuera à augmenter la satisfaction des utilisateurs et à améliorer les performances globales de mon site.

IX. Développement de la partie back-end

La partie back-end de la boutique en ligne a été développée en utilisant PHP, MySQL et phpMyAdmin comme SGBD pour la gestion de la base de données. J'ai également implémenté une fonctionnalité de pagination pour limiter l'affichage à 16 produits par page. J'ai détaillé les composants du code nécessaires pour mettre en œuvre cette pagination.

Pagination

le composant qui implémente le système de pagination pour les produits provenant de la table "products". J'ai utilisé PHP pour récupérer les données depuis la base de données (MySQL) et JavaScript pour gérer l'affichage et l'interaction avec la pagination.

1. Récupération des données depuis la base de données :

Tout d'abord, je dois écrire une requête SQL pour récupérer les produits depuis la table "products". je vais utiliser la clause LIMIT pour paginer les résultats en fonction du numéro de page et du nombre d'éléments à afficher par page.


```
// Paramètres pour la pagination
$perPage = 16; // Nombre d'éléments à afficher par page
$page = isset($_GET['page']) ? (int)$_GET['page'] : 1; // Numéro de la page actuelle

// Calcul de l'offset pour la requête SQL
$offset = ($page - 1) * $perPage;

// Requête SQL pour récupérer les produits paginés
$sql = "SELECT * FROM products LIMIT $offset, $perPage";
$stmt = $pdo->query($sql);
$products = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

2. Affichage des produits paginés :

Maintenant que j'ai récupéré les produits depuis la base de données, je dois les afficher dans la page HTML en utilisant PHP.

```
<!-- Affichage des produits -->
<ul>
  <?php foreach ($products as $product): ?>
    <li>
      <h2><?php echo $product['name']; ?></h2>
      <p><?php echo $product['description']; ?></p>
      <p>Prix : <?php echo $product['price']; ?></p>
      ">
    </li>
  <?php endforeach; ?>
</ul>
```

3. Affichage de la pagination :

Pour que les utilisateurs puissent naviguer entre les différentes pages de produits, je dois afficher les liens de pagination. Je vais utiliser JavaScript pour rendre cette pagination interactive et éviter de recharger la page lors de la navigation entre les pages.

```

<!-- Affichage de la pagination -->
<div class="pagination">
  <?php
    // Requête SQL pour compter le nombre total de produits
    $countSql = "SELECT COUNT(*) as total FROM products";
    $countStmt = $pdo->query($countSql);
    $total = $countStmt->fetchColumn();

    // Calcul du nombre total de pages
    $pages = ceil($total / $perPage);

    // Affichage des liens de pagination
    for ($i = 1; $i <= $pages; $i++) {
      echo "<a href='?page=$i'>$i</a> ";
    }
  ?>
</div>

```

4. Gestion de la pagination côté JavaScript :

Pour rendre la pagination interactive, j'ai ajouté du code JavaScript pour gérer le chargement des nouvelles pages de produits sans recharger la page.

```

// Sélection des liens de pagination
const paginationLinks = document.querySelectorAll('.pagination a');

// Ajout d'un gestionnaire d'événements à chaque lien
paginationLinks.forEach(link => {
  link.addEventListener('click', async (event) => {
    event.preventDefault(); // Empêche le comportement par défaut du lien

    // Récupération du numéro de page à partir de l'attribut href
    const page = parseInt(link.getAttribute('href').split('=')[1]);

    // Requête fetch pour récupérer les produits de la nouvelle page
    const response = await fetch(`/get-products.php?page=${page}`);
    const data = await response.json();

    // Affichage des produits de la nouvelle page
    // (vous pouvez personnaliser cette partie en fonction de votre structure HTML)
    const productsContainer = document.querySelector('#productsContainer');
    productsContainer.innerHTML = ''; // Réinitialisation du contenu
    data.forEach(product => {
      const productHTML = `
        <li>
          <h2>${product.name}</h2>
          <p>${product.description}</p>
          <p>Prix : ${product.price}</p>
          
        </li>
      `;
      productsContainer.insertAdjacentHTML('beforeend', productHTML);
    });
  });
});

```

Avec cette implémentation, le composant de pagination permettra aux utilisateurs de naviguer entre les différentes pages de produits sans recharger la page. Cela améliore considérablement l'expérience utilisateur en offrant une navigation plus fluide et réactive sur votre site.

X. Conception de base de données MCD / MLD

Pour concevoir la base de données de la boutique en ligne, j'ai expliqué les notions de MCD (Modèle Conceptuel de Données), MPD (Modèle Physique de Données) et MLD (Modèle Logique de Données). J'ai ensuite réalisé une conception de la base de données "boutique" qui comprend les tables users, admin, products, carts, likes, comments et orders. J'ai fourni les attributs pertinents pour chaque table et j'ai adapté la structure en fonction des besoins spécifiques du projet.

1. MCD (Modèle Conceptuel de Données) :

Le Modèle Conceptuel de Données est la première étape dans la conception de ma base de données. C'est une représentation abstraite et indépendante du système d'information, qui se concentre sur les entités et leurs relations. Le MCD est réalisé avant de penser à la structure physique de la base de données. Les principaux éléments du MCD sont :

- **Entités** : Ce sont les objets du monde réel que je souhaite modéliser dans mon système d'information. Par exemple, dans ma base de donnée, j'ai des entités comme "Utilisateur", "Produit", "Panier", "Commande", etc.

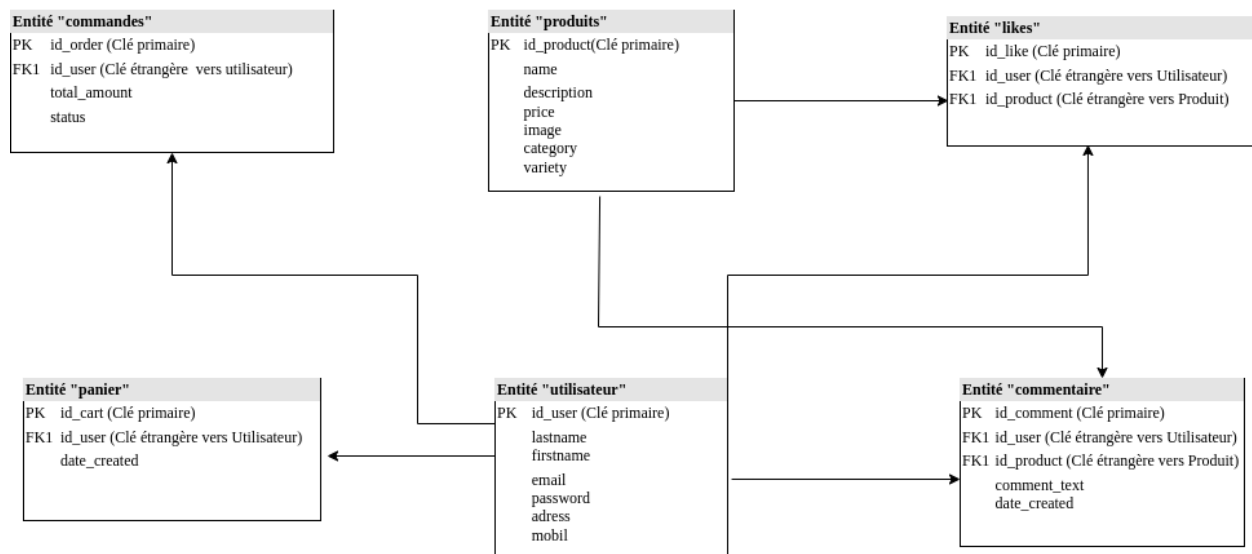
- **Attributs** : Chaque entité a des attributs qui définissent ses caractéristiques. Par exemple, l'entité "Produit" aura des attributs tels que "id_product", "name", "description", "price", "image", "category", "variety", etc.

- **Relations** : Les entités sont liées entre elles par des relations. Par exemple, il y aurait une relation entre "Utilisateur" et "Panier" car un utilisateur peut avoir un panier. Les relations peuvent être de différents types : 1:1 (un à un), 1:N (un à plusieurs), N:N

(plusieurs à plusieurs).

- **Cardinalité** : La cardinalité définit le nombre d'entités liées dans une relation. Par exemple, une relation 1:N entre "Utilisateur" et "Panier" signifie qu'un utilisateur peut avoir plusieurs paniers, mais chaque panier appartient à un seul utilisateur.

Modèle Conceptuel de Données (MCD) :



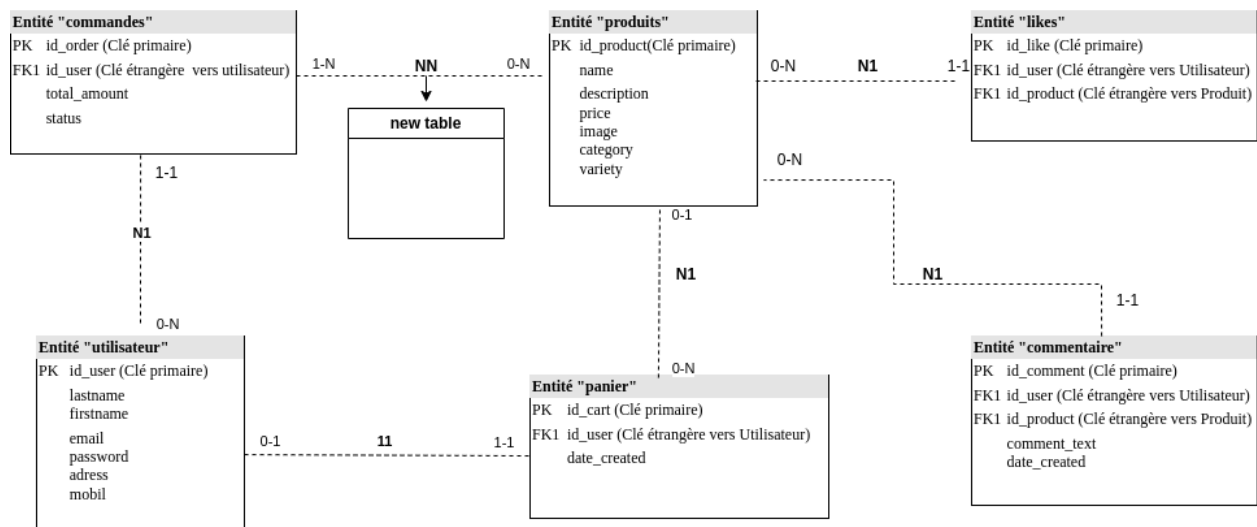
2. MLD (Modèle Logique de Données) :

Le Modèle Logique de Données est la deuxième étape de la conception de ma base de données. Il s'agit d'une représentation intermédiaire entre le MCD et le MPD. Le MLD se concentre sur les détails de mise en œuvre du MPD, mais sans dépendre d'un Système de Gestion de Base de Données (SGBD) spécifique. Les principaux éléments du MLD sont :

- **Types de données** : j'ai spécifié les types de données pour chaque attribut des tables du MPD. Par exemple, les colonnes "name" et "description" sont de type VARCHAR, "price" est de type DÉCIMAL, etc.

- **Contraintes d'intégrité** : j'ai défini les contraintes d'intégrité pour garantir l'intégrité des données. Par exemple, une contrainte d'intégrité pourrait être que le prix d'un produit doit être supérieur à zéro.
- **Autres détails de mise en œuvre** : Le MLD spécifie d'autres détails de mise en œuvre tels que les valeurs par défaut des attributs, les règles de validation, etc.

Modèle Logique de Données (MLD) :



3. MPD (Modèle Physique de Données) :

Le Modèle Physique de Données est la troisième étape de la conception de ma base de données. C'est la représentation concrète du MCD, où les entités et les relations sont traduites en tables et en clés primaires et étrangères. Les principaux éléments du MPD sont :

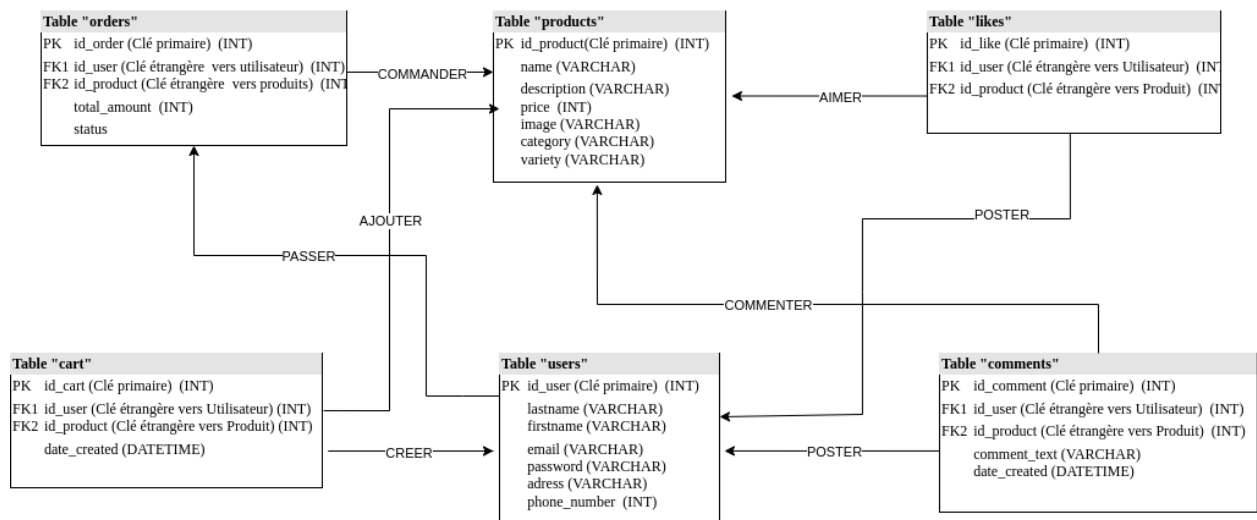
- **Tables** : Chaque entité du MCD devient une table dans le MPD. Par exemple, l'entité "Utilisateur" deviendrait une table "users", l'entité "Produit" deviendrait une table "products", etc.

- **Clés primaires** : Chaque table doit avoir une clé primaire qui identifie de manière unique chaque enregistrement de la table. Par exemple, pour la table "users", la clé primaire pourrait être "id_user".

- **Clés étrangères** : Les relations du MCD sont traduites en clés étrangères dans le MPD. Une clé étrangère est une référence à la clé primaire d'une autre table. Par exemple, dans la table "Panier", vous pourriez avoir une colonne "id_user" qui est une clé étrangère faisant référence à la clé primaire "id_user" dans la table "Utilisateur".

- **Index** : Vous pouvez ajouter des index aux colonnes fréquemment utilisées pour accélérer les opérations de recherche dans la base de données.

Modèle Physique de Données (MPD) :



XI. Veille et sécurité

La sécurité du site "Théophile" est une priorité. J'ai abordé différents aspects de la sécurité, tels que le hachage des mots de passe, la prévention des injections SQL et la sécurisation de l'architecture MVC. J'ai expliqué comment j'ai implémenté ces mesures de sécurité pour garantir la protection des données des utilisateurs.

1. Hachage des mots de passe :

Le hachage des mots de passe est une pratique essentielle pour sécuriser les informations d'identification des utilisateurs. Lorsqu'un utilisateur crée un compte ou met à jour son mot de passe, le mot de passe ne doit jamais être stocké en texte brut dans la base de données. Au lieu de cela, il doit être haché à l'aide d'une fonction de hachage sécurisée avant d'être stocké.

- **Hachage** : Le hachage est un processus qui prend une chaîne de caractères (le mot de passe de l'utilisateur) et la transforme en une séquence de caractères alphanumériques de longueur fixe, appelée "haché". Les algorithmes de hachage couramment utilisés sont bcrypt, Argon2 et SHA-256.

- **Salt (Sel)** : Pour renforcer la sécurité, un "sel" (salt) est généralement ajouté au mot de passe avant le hachage. Le sel est une chaîne de caractères aléatoire qui est différente pour chaque utilisateur. Cela garantit que même si deux utilisateurs ont le même mot de passe, leurs hachages seront différents en raison du sel unique.

- **Vérification des mots de passe** : Lorsque l'utilisateur tente de se connecter, le mot de passe fourni est haché avec le même sel que celui stocké dans la base de données. Le hachage résultant est ensuite comparé au hachage stocké pour vérifier si le mot de passe est correct.

2. Traitement des formulaires :

Le traitement des formulaires est une autre zone critique pour la sécurité. il est important de réaliser quelques pratiques importantes pour éviter les vulnérabilités :

- Validation côté client : j'ai utilisé JavaScript pour effectuer une validation côté client afin de détecter les erreurs de saisie et d'éviter de soumettre des données incorrectes au serveur. Cependant, la validation côté client ne suffit pas, car elle peut être contournée.

```
form.addEventListener('submit', (e) => {  
  let messages = []  
  if(lastname.value === '' || lastname.value = null){  
    messages.push('veillez renseigner votre nom svp')  
  }  
  if (lastname.value.length <= 2){  
    messages.push('la taille de votre nom est trop petite')  
  }  
  if(messages.length > 0){  
    e.preventDefault()  
    error1.innerText = messages.join(', ')  
  }  
})
```

- Validation côté serveur : j'ai effectué une validation côté serveur pour chaque formulaire soumis. Vérifiez que toutes les données attendues sont présentes et correctes.

```
// Vérification de l'adresse e-mail (non vide et format valide)
if (empty($email) || !filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $errors['email'] = 'Veuillez saisir une adresse e-mail valide';
}

// Vérification du mot de passe (non vide et longueur minimale)
if (empty($password) || strlen($password) < 8) {
    $errors['password'] = 'Le mot de passe doit contenir au moins 8 caractères';
}

// Vérification de la confirmation du mot de pass
if ($password !== $confirmPassword) {
    $errors['confirm_password'] = 'Les mots de passe ne correspondent pas';
}
```

- Protection contre les attaques XSS (Cross-Site Scripting) : j'ai mis en place un système de filtrage et échappé correctement toutes les données saisies par les utilisateurs avec la fonction "htmlspecialchars()" pour échapper les caractères spéciaux qui pourraient être interprétés comme du code HTML ou JavaScript. pour éviter l'exécution de scripts malveillants sur les pages du site.


- Protection contre les attaques : j'utilise des jetons CSRF pour empêcher les requêtes forgées venant de domaines externes.

3. Système de routage avec l'architecture MVC :

Dans mon architecture MVC, le système de routage est responsable de diriger les requêtes HTTP vers les bonnes actions du contrôleur en fonction de l'URL demandée

par l'utilisateur. il contribue à la sécurité grâce :

- au Contrôle d'accès : Le système de routage peut être utilisé pour restreindre l'accès à certaines URL ou actions en fonction du rôle de l'utilisateur connecté. Par exemple, seuls les administrateurs peuvent accéder aux fonctionnalités d'administration.
- à la Réécriture d'URL : Un fichier .htaccess bien configuré peut être utilisé pour réécrire les URL et masquer les paramètres de requête. Cela rend l'URL plus conviviale pour les utilisateurs et rend difficile pour les attaquants de deviner les URL sensibles.



```
rewriteCond %{REQUEST_FILENAME} !-f
rewriteCond %{REQUEST_FILENAME} !-d
rewriteRule ^(.*)$ index.php?page=$1
```

- à la Validation d'URL : Le système de routage doit effectuer une validation des URL pour éviter les injections de code et les URL malveillantes.

XII. Exemple d'un composant

Pour illustrer un composant spécifique du site, j'ai détaillé le parcours utilisateur lors de

l'ajout d'un produit au panier. J'ai expliqué comment utiliser ce composant, en mettant en évidence les étapes nécessaires pour finaliser l'ajout au panier. J'ai également décrit le fonctionnement interne de ce composant, en mettant l'accent sur les interactions entre le front-end et le back-end.

Composant : Ajouter un produit au panier

1. L'utilisateur accède à la page d'accueil de la boutique en ligne.
2. Il parcourt les produits disponibles sur la page d'accueil ou explore différentes catégories de produits à l'aide de la navigation.
3. L'utilisateur trouve un produit qui l'intéresse et clique sur son image ou son nom pour accéder à la page de détails du produit.
4. Sur la page de détails du produit, l'utilisateur peut voir toutes les informations sur le produit, telles que son nom, sa description, son prix, son image, etc.
5. L'utilisateur a également la possibilité d'ajouter le produit à son panier en cliquant sur le bouton "Ajouter au panier".
6. Lorsque l'utilisateur clique sur le bouton "Ajouter au panier", une requête asynchrone (Ajax) est envoyée au serveur pour enregistrer l'ajout du produit au panier.
7. Le serveur reçoit la requête, vérifie l'authenticité de l'utilisateur et ajoute le produit à son panier en utilisant la session de l'utilisateur.
8. Une fois que le produit est ajouté au panier, le serveur renvoie une réponse au format JSON indiquant que l'opération a réussi.
9. Sur la page du produit, un message de confirmation apparaît pour informer l'utilisateur que le produit a été ajouté au panier avec succès.
10. L'utilisateur peut continuer à parcourir d'autres produits ou accéder à son panier pour voir les produits qu'il a ajoutés.

11. L'utilisateur peut également modifier la quantité des produits dans le panier, supprimer des produits du panier ou passer à l'étape suivante du processus de paiement.
12. Lorsque l'utilisateur est prêt à passer à l'étape de paiement, il peut accéder à la page de paiement pour finaliser sa commande.

XIII. Difficultés rencontrées

Dans le cadre du projet, j'ai été confronté à certaines difficultés, notamment en ce qui concerne l'écriture de requêtes SQL, en particulier pour la méthode de pagination avec les clauses LIMIT et OFFSET. J'ai éprouvé ces difficultés en détail, soulignant les problèmes spécifiques rencontrés lors du développement du projet.

XIV. Solutions mises en œuvre

Pour surmonter les difficultés mentionnées précédemment, j'ai pris différentes mesures. J'ai consulté la documentation de PHP.net et j'ai approfondi mes connaissances sur les requêtes SQL pour résoudre les problèmes spécifiques rencontrés. J'ai expliqué les solutions que j'ai mises en œuvre pour garantir le bon fonctionnement du projet.

XV. Conclusion

En conclusion, le projet de boutique en ligne "Théophile" a été une expérience enrichissante qui m'a permis de mettre en pratique mes compétences en développement web et mobile. J'ai acquis une expérience précieuse dans la conception et la réalisation d'une boutique en ligne complète. Ce projet m'a également permis de renforcer ma maîtrise des langages front-end et back-end ainsi que ma compréhension

des concepts clés liés à la sécurité et à la gestion des bases de données.