Titre : Développeur web et web mobile

Dossier de projet

Par Jules Jean-Louis

Compétences du référentiel couvertes par le projet

Le projet couvre les compétences énoncées ci-dessous.
Pour l'activité 1, **"Développer la partie front-end d'une application web et web mobile en intégrant les recommandations de sécurité" :**

- Maquetter une application
- Réaliser une interface utilisateur web ou mobile statique et adaptable
- Développer une interface utilisateur web dynamique
- Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

Pour l'activité 2, v**"Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité" :**

- Créer une base de données
- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile
- Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce.

Présentation du projet

"WellGames" est une boutique en ligne de vente de jeux-vidéos, cette boutique a pour but de proposer tous les derniers jeux sorti ainsi que de renforcer l'aspect communautaire, grâce à l'avis et commentaires des clients. Le site s'inspirait des autres sites populaires du genre tel qu'Instant gaming.

Spécifications fonctionnelles

Périmètre du projet

Le site sera réalisé en français et ce dernier devra être accessible sur différents supports, à savoir mobile, tablette et ordinateur.

Cible visée par le site internet

Ce site vise les particuliers qui souhaitent se procure des jeux-vidéos.

Arborescence du site

L'arborescence du site se décline comme suit :

Page d'accueil

Page catalogue
Page recherche
Page descriptif du produit
Page profil
Page panier
Page récapitulatif du panier
Page paiement
Page administration

Description des fonctionnalités

1. Inscription est authentification

Pour améliorer l'expérience utilisateur, j'ai décidé de faire que l'utilisateur pouvait s'inscrire est se connecter sur toutes les pages du site qui n'exige pas d'être connecté.

2. Catalogue de produit et filtre

Une page doit permettre d'afficher l'ensemble des produits disponibles. Cet affichage devra comprendre la photo du produit, le nom du produit ainsi que son prix.
Un filtrage doit être aussi implémenté afin de trier les produits en fonction de leur catégorie, mais aussi en fonction de l'année de sortie. Un système de pagination doit être aussi présent dans le but de facilité la navigation.

3. Autocomplétion

Dans le header une barre de recherche va être implémenté qui permet de chercher les produits par nom.

4. Page descriptif du produit

L'utilisateur devra être en mesure d'accéder à une fiche produit. Cette dernière devra comprendre :

- Une ou plusieurs photos du produit
- Le nom du produit
- Le prix
- La description du produit
- La note du produit
- Les commentaires clients

5. Évaluation des produits et commentaires clients.

L'utilisation devra être en mesure dévaluer le produit grâce à un système de notation sur 5 étoiles. 5 étoiles signifiant "Très satisfait du produit" et 1 étoile "Pas du tout satisfait du produit".
L'utilisateur pourra également laisser un commentaire sur le produit afin de faire part de ses appréciations sur ce dernier. L'utilisateur n'a pas besoin d'acheter le produit pour pouvoir commenter et laisse une note.

6. Page panier

L'utilisateur ne doit pas avoir besoin d'être connecté ou inscrit pour ajouter des produits à son panier.
Ce panier devra afficher les éléments suivants :

- Une photo de l'article
- Le prix de l'article
- La quantité demandée par le client
- Le total des articles sélectionnés
- Un bouton de validation du panier débouchant sur le processus de commande.

Le client devra être en mesure d'augmenter ou diminuer la quantité demandée. Il aura aussi la possibilité de supprimer un article du panier, voire l'intégralité du panier.

7. Espace client

L'utilisateur aura accès à un espace client dans lequel il lui sera possible de consulter est modifié ses informations. À savoir son nom, son prénom, son adresse email, son mot de passe, son code postal et sa ville.
Il aura également la capacité de consulter ses précédentes commandes est de voir leur statut (annulée, en cours de traitement, expédiée).
L'utilisateur peut aussi changer d'image de profil.

8. Espace administrateur

L'administrateur du site devra avoir accès à un espace sécurisé lui permettant d'administrer le site.

8.1 Ajoute des catégories ou sous-catégories

L'administrateur sera en mesure de gérer les catégories et sous-catégories de produit, c'est-à-dire créer des catégories et les supprimer.

8.2 Gestion des produits

L’administrateur aura la possibilité de la capacité d’ajouter, modifier ou supprimer des produits.
Lors de la création du produit, ce dernier sera capable de :

- Donner un titre au produit
- Définir le prix du produit
- Décrire le produit
- Uploader une image du produit
- Établir les stocks du produit

8.3 Suivi des commandes

L’administration aura la possibilité de visualiser les commandes qui ont été effectuées sur le site. Ceci signifie que l’administrateur aura accès aux informations suivantes :

- La date de la commande
- Le nom et le prénom de la personne ayant passé commande
- Son adresse
- Le détail de la commande.

L’administrateur pourra également changer le statut de la commande, par exemple passer la commande en commande expédiée lorsque celle-ci a été remise au prestataire de livraison.

9. Solution de paiement

Une solution de paiement fictive a été mise en place dans le futur, elle sera remplacée par Stripe.

Spécifications techniques

Choix techniques et environnement de travail

Outil utilisé pour la conception de cette boutique :

Behance et Dribbble sont deux plateformes en ligne très populaires qui servent d'inspiration pour le design et l'UX/UI. Elles sont largement utilisées par les designers, les artistes et les professionnels créatifs du monde entier pour partager leur travail, découvrir de nouvelles tendances, trouver des idées et s'inspirer mutuellement.

Figma est en effet un outil de design graphique collaboratif basé sur le cloud. Il permet aux concepteurs, aux développeurs et aux équipes de travailler ensemble sur la conception d'interfaces utilisateur, de maquettes et de prototypes interactifs. Figma offre une interface intuitive et des fonctionnalités puissantes qui facilitent la création et la collaboration sur des projets de conception.

GitHub est une plateforme d'hébergement de dépôts Git. C'est un service en ligne qui permet aux développeurs de stocker leurs projets, de gérer les versions du code, de collaborer avec d'autres développeurs et de suivre les problèmes et les demandes de fonctionnalités (issues et pull requests). GitHub facilite également la contribution et le partage de code entre les développeurs.

Les langages de programmation utilisés en back-end :

- PHP est un langage de script côté serveur populaire, utilisé pour développer des applications web dynamiques et interactives. Il permet de traiter les données côté serveur, de se connecter à des bases de données, de gérer des sessions utilisateur, d'envoyer des requêtes HTTP, etc.
- POO avec l'utilisation de class.

Système de gestions de base de données :

- Base de données SQL.

Les langages de programmation utilisés en front-end :

- Le projet sera réalisé avec du HTML et CSS.
- TailwindsCSS qui est framework CSS.
- JavaScript est un langage de programmation côté client largement utilisé pour rendre les sites web interactifs et dynamiques. Il permet d'ajouter des fonctionnalités telles que des animations, des interactions utilisateur, des validations de formulaires et bien plus encore.

L'environnement de développement est le suivant :

- Éditeur de code : PHP Storm
- Serveur apache

Réalisations

1. Charte graphique

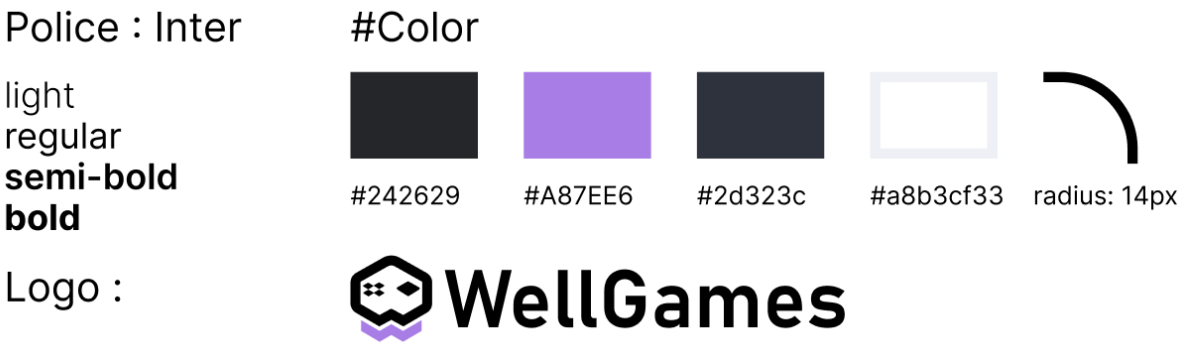
La police d'écriture est : Inter

2. Maquette

La maquette a été réalisée sur Figma, avec une maquette basse définition, ainsi qu'une version mobile du site. Le but était de pouvoir conceptualiser le site web, pouvoir se rendre contre du résultat final vers lequel tendre.

2.1 Palette de couleurs

Je me suis grandement inspiré des autres sites d'e-commerce de vente de jeux-vidéos qui ont pour unes grandes majorités adoptées un thème sombre avec une couleur dominante qui ajoute du contraste.
J'ai décidé partir sur une palette de couleurs sombre, avec des variantes de gris ainsi que des bordures plus claires, afin qu'elle s'apparaisse plus distinctement. Pour le radius des bordures de tous les boutons ainsi que les inputs, j'utiliserais du 14 px.
J'ai ajouté beaucoup d'accents avec un violet que l'on retrouvera sur le logo, j'utiliserais cette couleur pour améliorer le contraste, et attirer le client vers les actions qu'il peut réaliser.



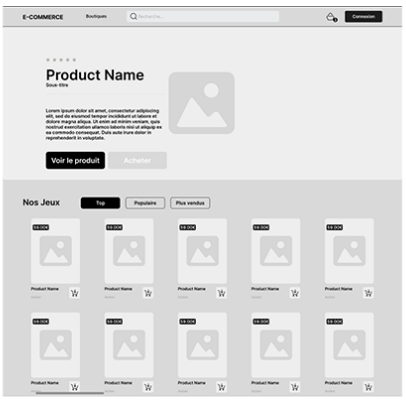
2.2 Maquette basse fidélité (Wireframe)

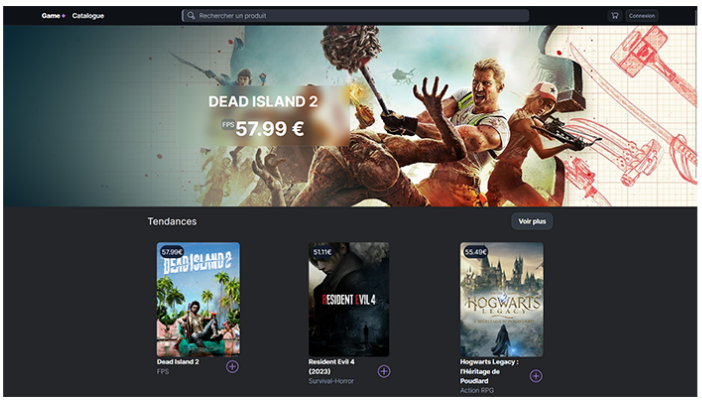
J'ai d'abord réalisé une maquette du site ne basses fidélités, c'est-à-dire que je ne fais que placent les éléments afin d'obtenir un résultat qui me semble cohérents et qui respecte ce que j'avais imaginé. Elle permet de visualiser l'organisation générale des éléments sur l'écran, sans entrer dans les détails de l'apparence de ces éléments. Elle est souvent utilisée pour valider la structure et la hiérarchie de l'information avant de passer à des étapes plus avancées de la conception.

2.3 Maquette haute fidélité

Une maquette haute fidélité est une représentation plus détaillée et plus proche du produit final. Elle inclut des éléments de design tels que les couleurs, les images et les polices, ainsi que des interactions avec l'interface. Elle permet de visualiser à quoi ressemblera le produit final et de valider les choix de design avant de passer à l'étape de développement.

2.4 Représentation de la basse et haute fidélité



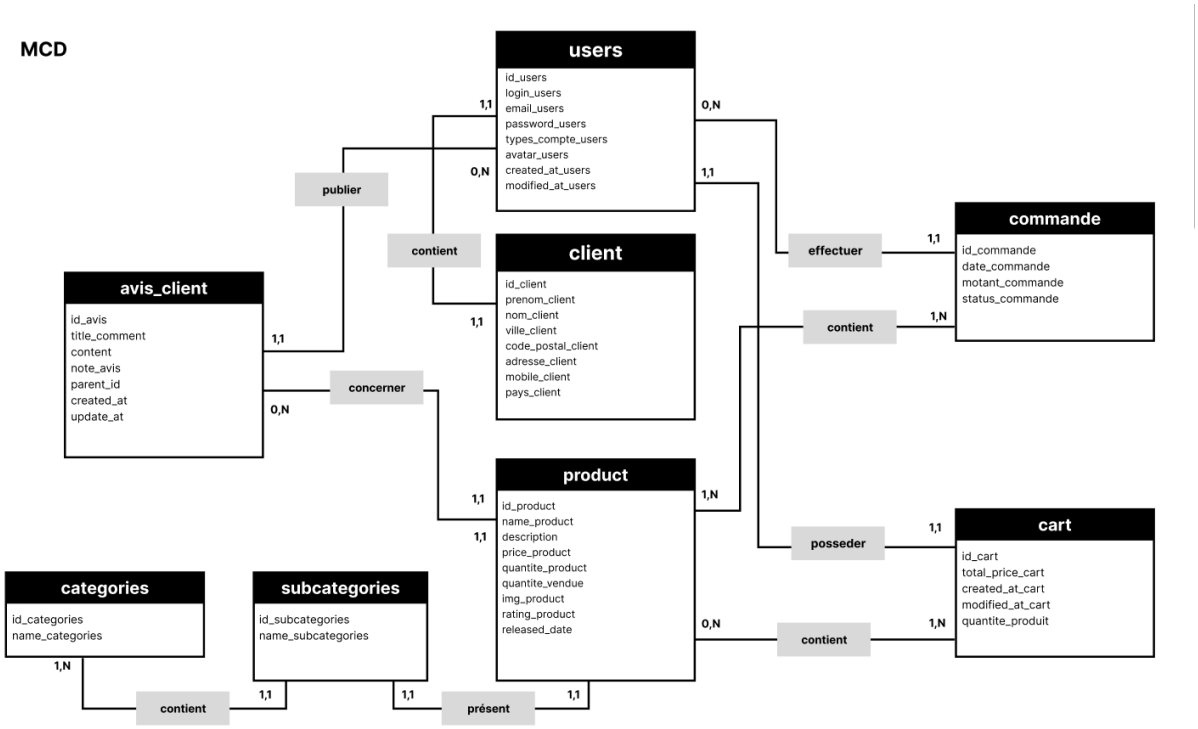


Une petite comparaison entre la version basse fidélité (gauche) et haute fidélité (droite).

3. Conception de la base de données

Après avoir déterminé les fonctionnalités à implémenter sur le site, j'ai pu développer la basse de données suivante :

MCD :



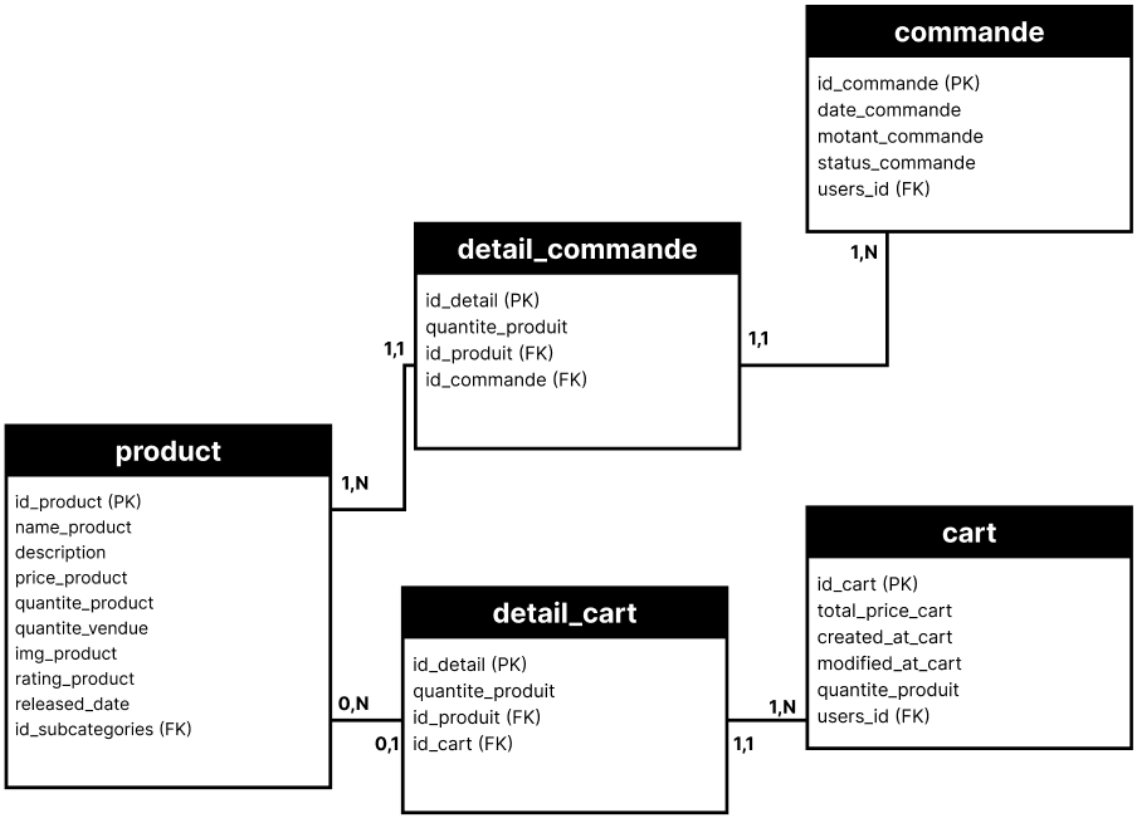
Vous trouverez en annexes le modèle logique de données (MLD) ainsi que le modèle physique de données (MPD).

Comme présenter ci-dessus, la base de données s'articule autour de deux entités principales. J'ai décidé par souci de sécurité de créer deux entités, une "users" et un autre "client", la première contiendra les informations, lorsque le client s'inscrit, comme son login (nom d'utilisateurs), mot de passe, la date de création de son compte ainsi que le type de compte. La deuxième contiendra toutes les données personnelles du client, comme son adresse, la ville, etc… qui devront être remplis si le client veut passer commande. La deuxième entité principale et l'entité "produit" avec tous les attributs relatifs au produit, comme le prix, le nom, la description, la date de parutions, etc….

L'entité "users" est liée à l'entité "commande", "cart", "client" ainsi qu'à l'entité "avis_client". Toutes ces liaisons vont permettre de stocker est géré toutes les informations du client, de l'inscription, connexion a la gestion du panier et des commandes effectuées par le client. J'ai décidé que lorsqu'un client se connecte pour la première fois, il se verra attribué un panier qui sera réinitialisée à chaque fois que le client passe commande, d'où la relation un-à-un entre "users" et "cart".
Le client pourra voir toutes les commandes passées, écrire des commentaires.
L'entité "product" est liée à l'entité "subcategories", "cart", "commande" et "avis_client". Avec ces liaisons, je peux facilement voir, la sous-catégorie d'un produit, et par extension sa catégorie, le produit pourra aussi être ajouté dans un panier, ou une commande effectuée, les commentaires des clients seront aussi liés au produit.

Pour l'ajout de commentaires contenu dans la table "avis_client", j'insère donc une nouvelle ligne et pour ajouter une réponse à un avis, je récupère l'id de l'avis sur le quelle la réponse doit être ajouté et je le place dans la colonne parent_id.

Lors du passager du MCD au MLD il faut transformer les liaisons paires à pair donc N:N en table de liaison, faire apparaître les clés étrangère et supprimer les verbes.



Lors de l'ajout d'un produit dans un panier, le détail du produit sera ajouté dans la table details_cart avec la quantité et l'id_cart pour qui soit reliée à la table cart, je fais la même chose pour la relation entre product et commande.

4. Extraits de code significatifs

4.1 Autocomplétion

J'ai ajouté une barre de recherche qui affiche les résultats, selon ce qui est entré par l'utilisateur.
J'ai tout d'abord créé un fichier de fonction JavaScript dans lequel se trouve une fonction que j'export pour pouvoir la réutiliser sur plusieurs pages du site avec import.
Dans cette fonction, je crée ma fonction "searchHeader" qui va dans un premier temps récupère la valeur de l'input du formulaire qui se trouve dans le Header, après

cela, je vérifie à l'aide de la propriété length est utilisée pour obtenir la taille ou la longueur d'un objet. Son comportement dépend du type d'objet auquel il est appliqué, cela peut être une chaîne de caractères, un tableau (array) ou encore un objet JavaScript. Je vérifie que la longueur de la chaine caractères soit supérieure à 0.
Si c'est le cas, j'effectue un fetch, qui une fonction de Javascript qui permet d'effectuer des requêtes réseaux, pour récupérer des ressources à partir d'un serveur et permet de traiter les réponses de manière asynchrone. JavaScript est un langage de programmation synchrone qui peut être utilisé de manière asynchrone, cela signifie que l'on peut continuer à effectuer des opérations en arrière-plan sans bloquer l'exécution du reste du code.

```
async function searchHeader() {
    let query = SearchBarHeader.value;
    if (query.length > 0) {
        await fetch (`src/php/fetch/produit/searchBarProduct.php?query=${query}`)
            .then(response => response.json())
            .then(data => {
                const containerHeaderSearch = document.getElementById("containerSearchBarResultHeader");
                if (data.status === 'error') {
                    containerHeaderSearch.innerHTML = `<p class="text-center text-[#a8b3cf] py-2">Aucun résultat</p>`;
                } else {
                  containerHeaderSearch.innerHTML = '';
                    for (const product of data.data) {
                        containerHeaderSearch.innerHTML += `
                    <div class="absolute">
                        <a href="produit.php?id=${product.id_product}">
                            <div class="flex items-center">
                                <p class="text-white">${product.name_product}</p>
                            </div>
                            <p class="text-sm text-[#a8b3cf]">${product.price_product} €</p>
                        </a>
                    </div>
                    `;
                }
            });
    } else {
        const containerHeaderSearch = document.getElementById("containerSearchBarResultHeader");
        containerHeaderSearch.innerHTML = '';
    }
}
```

Dans le fichier p searchBarProduct.php, j'ai une condition qui vérifier que la superglobale $_GET['query'] est bien une valeur non null, est trim() me permet

d’enlever les espaces.

```php
if (isset($_GET['query'])) {
    $search = htmlspecialchars(trim($_GET['query']));
    $product = new Product();
    $displayProduct = $product->searchProduct($search);
    if (count($displayProduct) > 0) {
        header("Content-Type: application/json");
        echo json_encode(["status" => "success", "data" => $displayProduct]);
    } else {
        header("Content-Type: application/json");
        echo json_encode(["status" => "error"]);
    }
}
```

Dans ma class Product, j'utilise la méthode searchProduct, j’effectue une requête préparée vers la base de données qui chercher les noms des produits ou les sous-catégories.

```php
public function searchProduct(string $search) : array
{
    $bdd = $this->getBdd();
    $req = $bdd->prepare("SELECT p.id_product, p.name_product, p.price_product, p.rating_product,
                          p.img_product, p.subcategories_id, s.name_subcategories
                          FROM product p
                          JOIN subcategories s ON p.subcategories_id = s.id_subcategories
                          WHERE p.name_product LIKE :search
                          OR s.name_subcategories LIKE :search");
    $req->execute(["search" => "%" . $search . "%"]);
    $result = $req->fetchAll(PDO::FETCH_ASSOC);
    return $result;
}
```

J’ai aussi ajouté un page search.php, page sur laquelle on sera redirigée si l’on clique sur entrée dans laquelle on retrouvera tous les produits correspondants à la recherche. J’ai utilisé une requête $_GET que je récupère et l’insert dans ma méthode searchProduct et je les afficherai directement grâce à un foreach en PHP.

4.2 Paginations

J’ai aussi ajouté un système de pagination pour faciliter la navigation est amélioré l’UX, c'est-à-dire l’expérience utilisateur. Pour ce faire, j’ai créé dans ma class Catalogue une fonction qui compte le nombre de pages. Dans les paramètres, j'ai ajouté la date, l’ordre, les catégories ou sous-catégories afin que le nombre de pages s’adapte au nombre de produits récupère lors du filtrage des produits.

J’effectue une première requête qui récupèrera tous les produits, puis si l’un des quatre paramètres sont différents d’une chaîne de caractères vides, je concatènerai à la première requête avec ce du paramètre non vide. Puis j’exécute et j’utilise la fonction ciel () sur le nombre de produits récupère et je divise par le nombre de produits désiré par page.

4.3 Gestion du panier pour un utilisateur déconnecté

Pour pouvoir sauvegarder les produits ajoutés par un utilisateur non inscrit ou identifié, j’ai utilisé la superglobales $_COOKIE. Un cookie est un fichier texte qui ne peut contenir qu’une quantité limitée de données, ils sont stockés sur l’ordinateur du client, ce qui indique qu’ils peuvent être modifiés par l’utilisateur, il vaut mieux éviter d’y enregistrer des données sensibles comme des mots de passe par exemple. Pour palier au fait que l’on ne peut enregistrer que des chaînes de caractères, je vais utiliser la fonction json_encode de php, ce qui me permettra de sauvegarder un tableau dans ce $_COOKIE[‘cart’].
Lorsque l’utilisateur va effectue l’action d’ajouter un produit au panier, je vais vérifier s'il existe déjà un $_COOKIE[‘cart’], si c’est le cas alors, je le décode le JSON sinon je crée un tableau.

```php
If (isset($_SESSION['id'])) {
 // Code si l'utilisateur est connecté
} else {
// Vérifie si le cookie 'cart' existe et le décode en tableau associatif
    $cart = isset($_COOKIE['cart']) ? json_decode($_COOKIE['cart'], true) : [];
```

J’effectue une recherche dans le tableau si le cookie existe pour vérifier si le produit qui doit être ajouté est déjà présent à l’aide de la fonction array_key_exists() qui prend en paramètre la clé est le tableau. Si c’est le cas, j'ajoute la quantité du produit, sinon je crée une nouvelle ligne avec le produit ajouté.
Une fois toutes les opérations réalisées, j'utilise la fonction setcookie() pour créer mon cookie, je lui donne le ‘cart’, la valeur du cookie dans mon cas le tableau en JSON, la date d’expiration sous forme de timestamp, ici un 30 jours. Le chemin sur le serveur sur lequel le cookie sera disponible, ici tout le site. Pour les deux true a la fin, le premier indique si le cookie doit uniquement être transmis à travers une connexion sécurisée HTTPS depuis le client. Le second indique si le cookie ne doit être accessible que par le protocole HTTP. Cela permet d’interdire l’accès au cookie aux langages de scripts comme le JavaScript par exemple, pour se protéger potentiellement d’une attaque de type XSS.

```php
If (isset($_SESSION['id'])) {
  // Si l'utilisateur est connecté
} else {
   $cart = isset($_COOKIE['cart']) ? json_decode($_COOKIE['cart'], true) : [];

   if (array_key_exists($id_product, $cart)) {

        $cart[$id_product]['quantity'] += $quantity_product;
    } else {

        $cart[$id_product] = ['id' => $id_product,
                              'name' => $name_product,
                              'quantity' => $quantity_product];
    }

    setcookie('cart', json_encode($cart), time() + (86400 * 30), "/", "", true, true);
    header("Content-Type: application/json");
    echo json_encode(["status" => "success"]);
    }
}
```

4.4 Gestion des utilisateurs dans le panel administrateur

Un panel administrateur a été implémenté sur le site, permettant aux administrateurs de gérer les produits, catégories, sous-catégories, commandes et utilisateurs. Le panel intègre le CRUD qui est un acronyme qui désigne les opérations de base sur les données : Create, Read, Update, Delete (Créer, Lire, Modifier, Supprimer). La page administrateur est protégée par une condition if (isset($_SESSION[‘type_compte’] === administrateur’) alors l’utilisateur peut se rendre sur la page sinon il redirige vers l’accueil.

Je récupère tous d’abord tous les utilisateurs que j’affiche ensuite avec un foreach grâce à JavaScript. L'administrateur peut modifier le rôle d’un utilisateur, le supprimer ou consulter ses informations comme le nombre de commandes et d’avis poster ainsi que ça date d’inscription. J’ai aussi ajouté un superadminstrateur qui ne peut être supprimé afin de toujours avoir un accès au panel administrateur.

```php
$errors = [];
if (isset($_GET['id'])) {
    $id = htmlspecialchars($_GET['id']);
    if ($_SESSION['type_compte'] !== 'administrateur') {
      $errors['error'] = 'Vous n\'avez pas les droits pour supprimer un utilisateur';
    } if ($_SESSION['id'] === $id) {
      $errors['error'] = 'Vous ne pouvez pas supprimer votre propre compte';
    } if ($id === '1') {
      $errors['error'] = 'Vous ne pouvez pas supprimer le compte super-administrateur';
    } else {
        $user = new Client();
        $userExist = $user->verifieIfUserExist($id);
        if (!$userExist) {
          $errors['error'] = 'Utilisateur introuvable';
        }
        $deleteUser = $user->deleteUser($id);
            $errors['error'] = 'Utilisateur supprimé avec succès';
    }
}
if (!empty($errors)) {
    header("Content-Type: application/json");
    echo json_encode($errors);
}
```

5. Veille sur les vulnérabilités de sécurité.

Je me suis attaché à me prémunir contre les failles les plus courantes selon l'OWASP (Open Web Application Security Project) qui identifie les principales vulnérabilités de sécurité dans les applications web.

5.1 Injection SQL

Les attaques par injection SQL se produisent lorsque des données non fiables sont intégrées dans une requête SQL envoyée à un interpréteur de base de données vulnérable. Cela peut permettre à un attaquant d'exécuter des commandes malveillantes, de manipuler les données ou d'accéder à des informations sensibles. Ce type d'attaque peut être effectué en exploitant des vulnérabilités dans les champs de saisie des formulaires ou les paramètres des requêtes HTTP, tels que les superglobales $_GET et $_POST en PHP. Par exemple, lorsqu'un utilisateur entre des données dans un formulaire de connexion, au lieu d'entrer son nom d'utilisateur et son mot de passe, il peut entrer une chaîne de caractères contenant du code SQL malveillant. Si l'application ne valide pas correctement les données entrantes, cette chaîne de caractères peut être intégrée dans une requête SQL et exécutée par le

moteur de base de données, ce qui permet à l'attaquant d'altérer le comportement de l'application.

Comment s'en prémunir ?

Pour éviter les injections SQL, il est important de valider correctement les données entrantes et d'utiliser des techniques telles que les requêtes préparées pour séparer les données des instructions SQL

Validez les entrées

Utiliser les fonctions natives de PHP telles que htmlspecialchars() qui va transformer certains caractères spéciaux comme les esperluettes, simples et doubles guillemets et les chevrons en entités HTML, ce qui empêche qu'une sous requête soit passée dans un argument lors d'une requête vers la base de données. Utiliser aussi la fonction trim() pour supprimer les espaces au début est à la chaine de caractères. On peut aussi utiliser des regex est utilisé la fonction preg_match() qui renvoie vrai ou faux et qui va me permettre de valide si le motif décrit dans le regex et valide ou non.

```php
if (preg_match("/^(?=.*[A-Za-z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-z\d@$!%*#?&]{8,16}$/", $password)) {
    return true;
} else {
    return false;
}
```

Préparez les requêtes SQL

J'utilise l'extension PDO (PHP Data Objects) pour interagir avec une base de données en PHP, pour préparer est exécuté des requêtes SQL de manières sécurisée et efficace. Pour préparer une requête avec PDO, je crée un objet PDOStatement en appelant la méthode prepare de cet objet. Une première requête est envoyée au serveur lors du prepare sans les valeurs des arguments. Puis, lors de l'exécution, les arguments sont envoyées, c'est-à-dire que l'on va lier des variables PHP aux paramètres de la requête, donc les deux requêtes sont indépendantes. De cette manière, il est impossible d'effectuer des injections.

J’ai aussi ajouté du typage sur mes fonctions pour que les données correspondent aux types attendus.

```php
public function checkLogin(string $login) : bool
{
    $bdd = $this->getBdd();
    $req = $bdd->prepare("SELECT COUNT(*) as total FROM users WHERE login_users = :login_users");
    $req->execute(array(
        "login_users" => $login
    ));
    $result = $req->fetch();
    if ($result['total'] > 0) {
        return true;
    } else {
        return false;
    }
}
```

Failles XSS

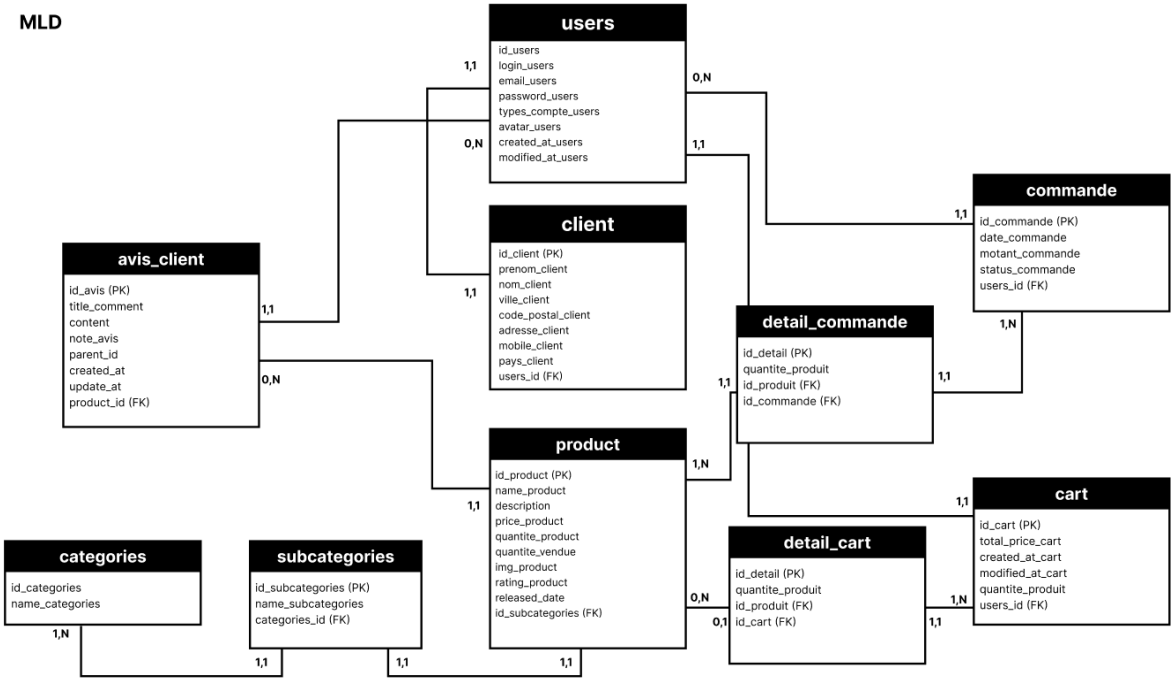
Cross-Site Scripting (XSS) : Les attaques XSS se produisent lorsque des données non fiables sont incluses dans une page web sans validation, ce qui permet à un attaquant d'exécuter du code JavaScript malveillant dans le navigateur des utilisateurs. C’est pour contre cela que j’aie utilisé htmlspecialchars() qui va donc échapper les chevrons, ce qui requit lors de l’utilisation de balise <script>.

Protection page administrateur

La page administrateur est protégée par la superglobales $_SESSION[‘types_compte] qui redirige l’utilisateur vers la page d’accueil s’il n’a pas l’accès à cette page. De même, il y a un super administrateur qui ne peut pas être supprimé par les autres administrateurs.

Annexe

MLD



MPD