

Dossier Projet

Titre : Développeur Web et Web mobile

Boutique en ligne

||| Ibni-yamine Chabane

Table des matières

Compétences du référentiel couvertes par le projet	4
Résumé	4
Spécifications fonctionnelles	5
Arborescence du site	5
Description des fonctionnalités	5
1. Authentification.....	5
2. Page boutique.....	5
3. Page produit.....	5
4. Page profil.....	5
5. Page admin (Back-office).....	6
5.1 Gestion des produits.....	6
5.2 Gestion des catégories Produit.....	6
6. Page panier.....	7
7. Page adresse de facturation.....	7
8. Page paiement.....	7
Spécifications techniques	8
Choix techniques et environnement de travail	8
Architecture du projet	8
Réalisations.....	8
1. Charte graphique	8
2. Maquette	9
3. Conception de la base de données	9
4. Extraits de code significatifs.....	10
4.1. Ajout de produit dans le panier.....	10
4.2. Page Panier.....	15
4.3. Page tableau de bord admin (back-office).....	17
4.3.1. Ajout d'un produit dans la boutique	17
4.3.2. Modifier un produit dans la boutique	19
Accessibilité	19
Sécurité.....	20
1. Sécurisation du module de connexion.....	20
2. Les failles XSS (Cross site scripting).....	21
2.1. Failles XSS stockée (Stored XSS)	22
2.2. XSS réfléchi (Reflected XSS)	22

3. Les injections SQL.....	22
Annexes	24
Maquette	24
Modèle conceptuel des données	25
Modèle logique de données	26
Modèle physique de données.....	27

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, "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.

Résumé

Le projet de boutique en ligne consiste à créer un site fonctionnel dédié à la vente de livres. L'objectif principal de ce projet est de mettre en pratique nos connaissances acquises lors de notre formation. J'ai conçu ce site en collaboration avec mes camarades de projet Samuel Durand et Joris Landaret. Nous avons nommé notre site "Book House".

Nous avons décidé d'adopter le thème de la vente de livres, car cela correspond à notre intérêt pour la lecture de bandes dessinées et à notre volonté de promouvoir la culture littéraire. Nous prévoyons d'inclure une variété de genres de bandes dessinées, des best-sellers aux classiques, afin de satisfaire les différents goûts des lecteurs.

Le site comprendra les fonctionnalités essentielles d'une boutique en ligne, telles que la recherche de produits, les descriptions détaillées des livres et un panier d'achat permettant aux utilisateurs de sélectionner et d'acheter des livres.

Il y a la partie gestion du site qui se fait par la page admin.

Nous avons également simulé le processus d'achat de produits sur le site afin de démontrer son fonctionnement. Cela inclura la sélection d'un livre, l'ajout au panier, le remplissage des informations de paiement et la finalisation de l'achat.

Ce projet nous a permis de mettre en pratique nos compétences en développement web, en conception d'interfaces utilisateur et en gestion de boutique en ligne.

Spécifications fonctionnelles

Arborescence du site

- Page d'accueil
- Page connexion
- Page inscription
- Page boutique
- Page produit
- Page Profil
- Page panier
- Page livraison
- Page confirmation de commande

Une Page Admin est également prévue pour la gestion du site.

Description des fonctionnalités

1. Authentification

C'est une authentification classique, l'utilisateur pourra de s'inscrire et se connecter grâce à un formulaire dans la page présent dans la page correspondant, la connexion se fera avec un e-mail et un mot passe via un formulaire, et l'inscription se fera via un formulaire dans lequel l'utilisateur entrera son email, prénom, nom, mot de passe et une confirmation de mot de passe pour confirmer pour pouvoir valider son inscription.

2. Page boutique

C'est une page qui affiche tous les produits disponibles dans le site, le sera afficher dans une card et à l'intérieur il y aura son image son nom et son prix, Il est prévu d'implémenter un filtre afin de trier les article en fonction de leurs catégorie.

3. Page produit

L'utilisateur qui aura sélectionné un produit sera redirigé vers la page produit qui devra afficher :

- Le nom du produit
- Une image du produit
- Son prix
- Et la description du produit

4. Page profil

L'utilisateur connecté avec son compte pourra avoir acces a sa page profil dans lequel il sera en mesure de changer ses informations, il aura la possibilité de changer :

- Son e-mail (qui sert de moyen de connexion pour se logger dans le site)

- Son prénom
- Son nom
- Son mot de passe

L'utilisateur pourra voir les produits qui aura mis dans son panier en temps réel depuis son profil.

5. Page admin (Back-office)

Cette page est exclusivement accessible au gérant du site qui possède un compte avec le rôle admin, tous les changements dans le site devra se faire à partir de cette page, l'admin peut voir tous les produits présents dans le site.

5.1 Gestion des produits

L'administrateur depuis la page admin peut voir tous les produits présents dans le site, Les produits seront affichés dans une card avec à l'intérieur son image, son nom, son prix et sa catégorie et sous-catégorie et dans chaque card il y aura les boutons suivants :

- Modifier le produit
- Supprimer le produit

En tête de la page admin il y aura aussi un bouton « ajouter un produit » qui redirige vers une page dédiée à l'ajout de produit, dans cette page l'administrateur devra :

- Donner un titre au produit
- Définir le prix du produit
- Faire une description du produit
- Uploader une image du produit
- Définir la catégorie et les sous-catégories du produit
- Définir les stocks disponibles pour le produit

Le bouton lui redirigera vers la page une page dédiée à la modification du produit, Le contenu de cette page sera exactement la même que la page d'ajout mais les champs les champs seront préremplis avec les informations actuelles du produit. Il sera alors possible d'ajouter quelques éléments ou de changer le contenu du produit.

5.2 Gestion des catégories Produit

L'administrateur depuis la page admin pourra créer des catégories et des sous-catégories.

6. Page panier

Dans cette page L'utilisateur pourra voir les produits qu'il a fait ajouté dans le panier depuis la page produit, il lui sera possible de voir pour chaque produits les informations suivantes :

- Une image du produit
- Nom du produit
- Son prix
- Sa quantité
- Son prix total
- Le prix total produit (qui est égale au prix total)
- Un bouton qui permet de supprimer le produit du panier

Le prix total de tous les produits sera comptabilisé et affiché, il sera aussi possible de vider entièrement son panier avec un bouton « supprimer le panier », un bouton « valider le panier »

Quand l'utilisateur sera sûr de son achat il pourra passer à l'étape suivante avec le bouton « valider le panier » qui le redirigera vers la page adresse de facturation.

7. Page adresse de facturation

Depuis cette page l'utilisateur devra entrer ses coordonnées postale et numéros de téléphone dans un formulaire, il pourra passer cette étape une fois le formulaire remplis avec validation sur le bouton « ajouter l'adresse et passe au paiement »

8. Page paiement

Dans cette page l'utilisateur entre dans un champ les informations de son moyen de paiement et pourra valider sa commande

Spécifications techniques

Choix techniques et environnement de travail

Technologies utilisées pour la partie back-end :

- Le projet a été réalisé avec le langage PHP et aussi en POO (Programmation orienté objet)
- Base de données SQL

Technologies utilisées pour la partie front -end :

- Le projet a été réalisé avec du HTML et CSS.
- Javascript pour dynamiser le site et améliorer l'expérience utilisateur

L'environnement de développement est le suivant :

- Editeur de code : Visual Studio Code
- Outil de versioning GIT, GitHub.
- Maquettage : Figma

Pour l'organisation du projet, nous avons utilisé Trello afin de découper le projet en une multitude de tâches à réaliser et de définir leurs ordres de priorité, grâce a cela nous avons pu nous répartir les tâches pour la réalisation du projet.

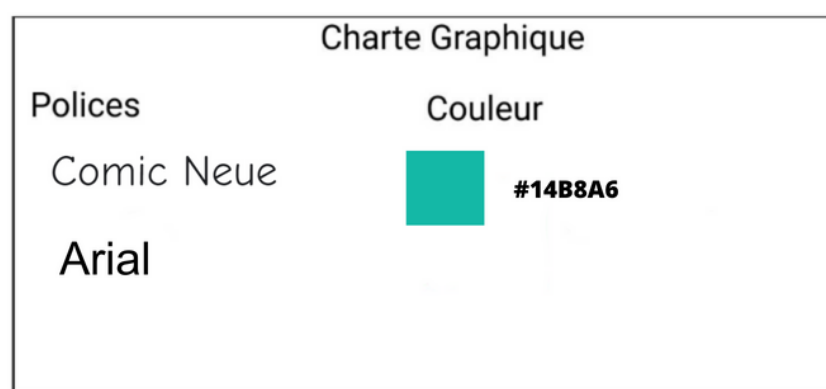
Architecture du projet

Pour la gestion des requêtes et des interactions avec la base de données, j'ai développé des classe dédiée que j'ai nommée "user" , "cart", "shop". Cette classe encapsule les fonctionnalités nécessaires pour effectuer des opérations courantes en requête SQL telles que l'insertion, la modification (update) et la récupération de données depuis la base de données.

Réalisations

1. Charte graphique

Les polices d'écriture sont les suivantes : Comic Neue et Arial.
La couleur dominante est la suivante #14B8A6 :



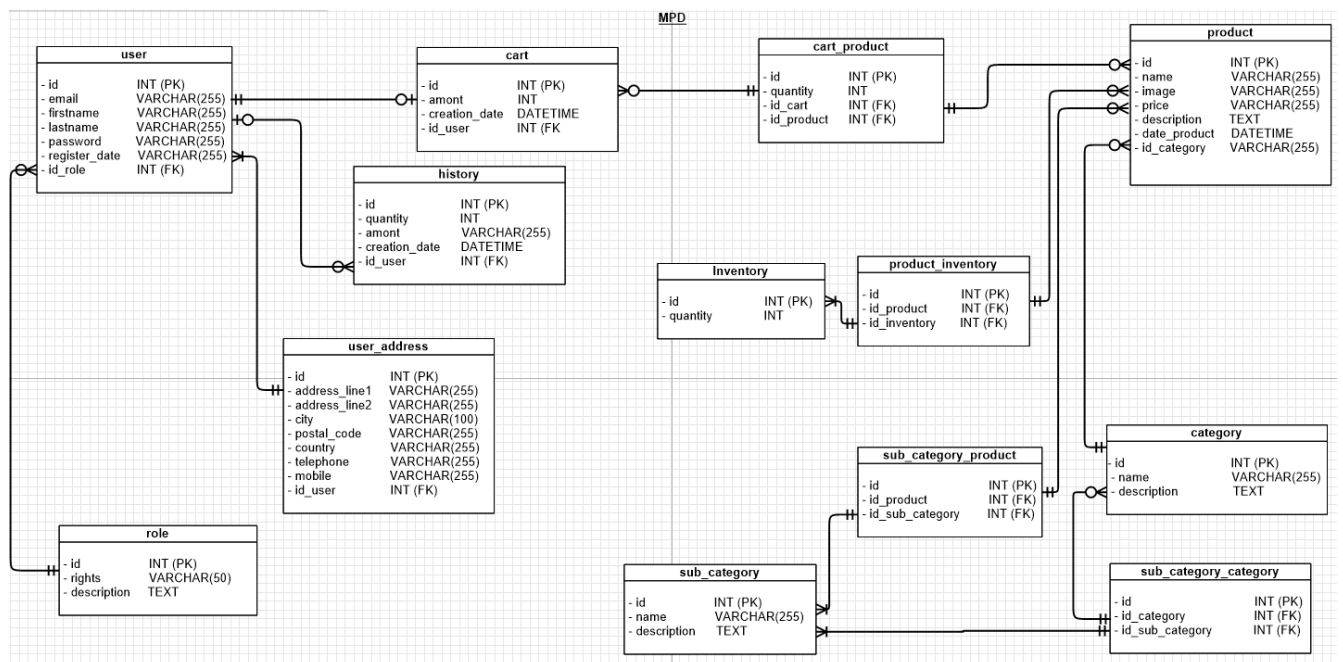
2. Maquette

Pour la conception visuelle de notre boutique en ligne, nous avons utilisé l'outil de conception et de prototypage Figma. Étant donné que nous n'avons pas de designer dans notre groupe, nous avons opté pour une maquette simple et fonctionnelle qui met l'accent sur l'organisation et la convivialité de l'interface utilisateur.

3. Conception de la base de données

Pour la conception de la base de données de notre boutique en ligne, nous avons utilisé la méthode de Merise. Cette approche nous a permis de structurer et d'organiser efficacement les données nécessaires au bon fonctionnement de notre application.

Voici notre MPD (modele physique de données)



Le MCD (Modèle conceptuel de données) et le MLD (Modèle logique de données) sont disponible dans les annexes du dossier.

Dans le schéma illustré ci-dessus, la table user est liée avec la table "user_address", ce qui permettra à un user d'enregistrer son adresse pour la fin d'une de ses commande, "user" est aussi liée à la table "cart" qui est son « panier », il y a une table de liaison "cart_product" qui permet de lier la table "Product" et la table "cart", de cette façon il sera possible d'enregistrer un produit dans le panier ainsi que la quantité.

La table "Product" est liée par une table de liaison "product_inventory" à "inventory" car de cette façon on pourra avoir le nombre de produit disponible dans l'inventaire.

La table "product" est liée par la table "category" pour classer le livre a un type ex : (BD, Comics, Manga, etc . . .)

Ensuite, nous avons une table "category" qui est liée à la table "sub_category" par une table de liaison "sub_category_category" et qui permet de définir les sous-catégories spécifiques à chaque catégorie. Cela nous aide à organiser et à classer les livres manière plus précise.

4. Extraits de code significatifs

4.1. Ajout de produit dans le panier

Depuis la page produit, l'utilisateur inscrit et connecté aura la possibilité d'ajouter un produit dans le panier grâce à un formulaire dans lequel il devra choisir la quantité qu'il voudra commander en fonction de sa disponibilité en stock.

Avec la class « shop » qui me sert à faire des actions en rapport avec les produits de la boutique, j'ai utilisé une méthode 'getProduct'

```
public function getProduct() {
    $request = $this->getDatabase()->prepare('SELECT product.id , `name` , `image` , `description` ,
        `price` , id_category, inventory.id , quantity
        FROM product
        INNER JOIN product_inventory
        ON product_inventory.id_product = product.id
        INNER JOIN inventory
        ON product_inventory.id_inventory = inventory.id
        WHERE product.id = (?)'
    );
    $request->execute(array($_GET['id']));
    return $productDatabase = $request->fetchAll(PDO::FETCH_ASSOC);
}
```

Cette méthode me permet de récupérer les informations en rapport avec le produit sélectionné en \$_GET cliqué à la selection. dans notre cas c'est le produit lui-même qui correspond a l'id \$_GET['id'], les donnée son retourné sous forme de tableau associatif

Depuis la page product j'instancie la class pour pouvoir afficher le contenu du produit

```
session_start();
require_once("src/class/shopClass.php");
require_once("src/class/cartClass.php");

$message = "";
$products = new shop;
$productDatabase = $products->getProduct();

$addProductCart = new cart;
```

Ce qui nous intéresse, c'est que dans les données récupérées, j'ai accès à la quantité disponible en stock du produit.

Grâce à cela je vais pouvoir imposer une limite de quantité dans mon formulaire qui ne pourra pas être dépassé dans le HTML :

```
<input type="number" id="quantity" name="quantity" value="1" maxlength="4" size="3"
min="1" max="<?= $productDatabase[0]['quantity'] ?>">
```

Ce code est fonctionnel mais il y a justement une faille qui pourrait permettre à un utilisateur de contourner la limite de produit en stock et de l'ajouter dans le panier. Il lui suffirait juste de :

- Supprimer la valeur de 'max' pour dépasser la limite
- Modifier le type de l'input et essayer de mettre du texte au lieu d'un nombre

Pour remédier à ce problème, j'ai créé une condition au début de la page pour ce formulaire.

```

if (isset($_POST['submit'])) {
    if ($_POST['quantity']) {
        if($_POST['quantity'] > $productDatabase[0]['quantity']) {
            $message = 'vous ne pouvez pas dépasser la quantité disponible pour ce produit';
        } else if ($_POST['quantity'] == 0) {
            $message = 'vous ne pouvez pas choisir la valeur 0';
        } else if ($_POST['quantity'] < 0) {
            $message = 'vous ne pouvez pas choisir une valeur négative';
        } else {
            $quantity = (int) strip_tags($_POST['quantity']);
            $addProductCart->addProductInCart($quantity);
            $message = $addProductCart->getMessage();
        }
    } else {
        $message = "veuillez choisir une quantité";
    }
}
}

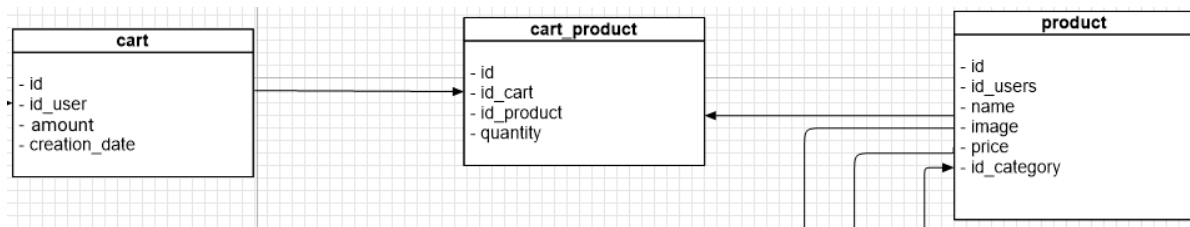
```

Cette condition empêchera l'utilisateur de contourner les limitations, car elle met aussi des limites dans le code PHP par mesure de sécurité, toute tentative de contournement afficheront un message d'erreur en dessous du formulaire. Même le changement de type ne peut être contourné car la variable \$quantity est typé pour être un (INT) . Si toutes les conditions sont remplies l'utilisateur peut ajouter le produit dans son panier.

Cependant l'ajout du produit dans le panier est complexe car à premiers vu, dans le formulaire on envoie seulement la quantité du produit à ajouter dans le panier.

Mais toutes les étapes d'ajout dans le panier se passe dans la méthode addProductInCart() de la class 'cart' , la class 'cart' qui est dédiée à tout ce qui est en rapport avec le panier.

Quand l'utilisateur se logue dans le site avec son compte, à la connexion je récupérer son id d'utilisateur et son rôle avec \$_SESSION pour les réutiliser dans une autre page.



Tous cela devra être envoyé dans les 3 tables si dessus donc il y aura de multiples requêtes dans ma méthode qui vont me permettre d'ajouter le produit dans le panier avec les données que je dispose qui sont `$_SESSION["id_user"]` et `$quantity`

Etape 1 :

Je devrais réserver un panier `cart`, je commence donc avec une requête SQL préparée avec PDO, la requête `INSERT` pour insérer dans la table `cart` l'id de l'utilisateur `$_SESSION["id_user"]`, et le montant sera pour l'instant inséré temporairement avec la valeur 0.

Etape 2 :

Je fais une requête `select *` dans la table `cart` quand l'id de l'utilisateur correspond à notre `$_SESSION["id_user"]`, et je transforme la donnée récupérée en tableau associatif. Cette action a pour but de récupérer l'id de notre table `cart` qui correspond à notre `$_SESSION["id_user"]` pour l'étape 3

Etape 3 :

Je fais une requête vers la table de liaison '`cart_product`' pour y insérer avec la requête `"INSERT INTO"` l'id de notre table '`cart`' récupéré dans l'étape 2, l'id de notre produit qu'on possède déjà en `$_GET['id']` et la `quantity` qui correspond à la quantité du produit qu'on a choisi à l'envoi du formulaire, toutes les infos d'une ligne de la table `cart_product` sont bien remplies.

Etape 4 :

Je fais une requête `SELECT` en `INNER JOIN` sur 4 tables : '`user`', '`cart`', '`cart_product`', '`product`'.

- En liaison de `cart` et `user` par l'id de `user`,
- En liaison de `cart` et `cart_product` par l'id de `cart`
- En liaison de `cart_product` et `product` par l'id du `product`

Tous ceci dans le but de pouvoir récupérer le prix du produit et la quantité

On va pouvoir faire le calcul : `$amount = $quantity * $price`

Dernière étape :

Il me reste à seulement à mettre à jour la colonne `amount` de la table `cart` car je l'avais mis à la valeur 0 dans l'étape 1, avec une requête `"UPDATE"` de la table `cart`.

Notre produit est désormais ajouté dans le panier, voici le code dans son intégralité :

```

public function addProductInCart(int $quantity) {

    //étape 1 on insert l'id de l'user dans la table cart
    $request = $this->getDatabase()->prepare('INSERT INTO cart(id_user , amount, creation_date)
                                           VALUES (?, ?, NOW())'
                                           );

    $request->execute(array($_SESSION['id_user'], 0));
    //étape 2 on select la table cart ou la colonne id user correspond a notre user pour récupérer
    l'id de cart
    $request2 = $this->getDatabase()->prepare('SELECT id
                                           FROM cart
                                           WHERE id_user = (?)
                                           ORDER BY id DESC'
                                           );

    $request2->execute(array($_SESSION['id_user']));
    $cartDb = $request2->fetchAll(PDO::FETCH_ASSOC);
    $id_cart = $cartDb[0]['id'];

    //étape 3 on insert l'id_cart , l'id_product ,et la quantité entré dans le champ par l'user
    $request3 = $this->getDatabase()->prepare('INSERT INTO cart_product(id_cart , id_product ,
                                           quantity)
                                           VALUES (?, ?, ?)'
                                           );

    $request3->execute(array($id_cart, $_GET['id'], $quantity));

    //étape 4 on fait un select des table pour récupérer le prix par rapport a la quantité sur le bon
    produit
    $request4 = $this->getDatabase()->prepare('SELECT user.id , product.id , price , amount ,
                                           quantity, cart.id
                                           FROM user
                                           INNER JOIN cart
                                           ON user.id = cart.id_user
                                           INNER JOIN cart_product
                                           ON cart_product.id_cart = cart.id
                                           INNER JOIN product
                                           ON cart_product.id_product = product.id
                                           WHERE id_user = (?) AND cart.id = (?)'
                                           );

    $request4->execute(array($_SESSION['id_user'], $id_cart));
    $displaySelect = $request4->fetchAll(PDO::FETCH_ASSOC);
    $priceDb = $displaySelect[0]['price'];
    $amount = $priceDb * $quantity;

    //derniere étape ajouter le montant dans notre table cart qui correspond a notre id.
    $update = $this->getDatabase()->prepare('UPDATE cart
                                           SET `amount` = (?)
                                           WHERE id_user = (?) AND cart.id = (?)'
                                           );

    $update->execute(array($amount , $_SESSION['id_user'], $id_cart));

    $this->message = "votre produit à bien été ajouté dans votre panier";

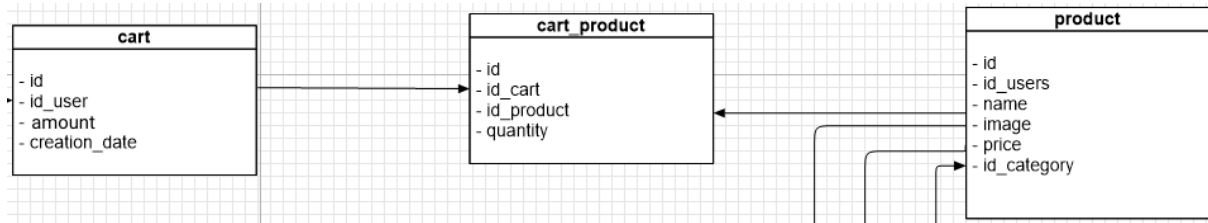
}

```

4.2. Page Panier

Dans la page panier, l'utilisateur sera en mesure de voir les produits qu'il aura ajouté dans le panier en amont depuis la page "product".

Voici les tables concernées pour la récupération des données :



Dans la class "cart" j'ai une méthode "getCartProductsByUserId()" dans lequel on doit entrer en paramètre l'id de l'utilisateur, cette méthode permet de récupérer les données du panier de l'utilisateur connecté sous forme de tableau associatif avec une requête préparé "SELECT" en "INNER JOIN" avec les 3 tables. l'id de l'utilisateur est récupéré quand l'utilisateur se connecte à son compte dans un `$_SESSION['id_user']` , nous avons donc déjà accès à l'id de l'utilisateur :

Voici la méthode de la class "getCartProductsByUserId()" :

```
public function getCartProductsByUserId($user_id) {
    // Requête SQL pour récupérer les produits du panier de l'utilisateur
    $stmt = $this->getDatabase()->prepare('SELECT product.id
                                           as id_product, product.name,
                                           product.price,
                                           product.image,
                                           cart_product.quantity, cart.amount,
                                           cart.id
                                           as id_cart
                                           FROM product
                                           INNER JOIN cart_product
                                           ON product.id =
                                           cart_product.id_product
                                           INNER JOIN cart
                                           ON cart.id = cart_product.id_cart
                                           WHERE cart.id_user = (?)'
                                           );

    $stmt->execute(array($user_id));
    $products = $stmt->fetchAll(PDO::FETCH_ASSOC);
    return $products;
}
```

Une fois les données récupérées, je les affiche avec une boucle "foreach" pour afficher tous les produits dans le html. Voici un aperçu

Image	Nom	Prix unitaire	Quantité	Total	total produit	Action
	Solo Leveling - Tome 01	14 €	2	28 €	28 €	supprimer du panier
	Demon Slayer - Tome 01	10 €	5	50 €	50 €	supprimer du panier
Total :				78 €		

[Supprimer le panier](#) [Valider le Panier](#)

Il y a la possibilité de supprimer un produit du panier grâce à un bouton "supprimer du panier", ce bouton redirige vers une page de traitement "delete_cart_product.php" qui contient la requête préparé "DELETE" qui permettra la suppression du produit qui est dans un \$_GET, dans la boucle foreach j'ai récupéré l'id du "produit" et l'id du "panier" qui correspond au produit que j'affiche pour le mettre dans l'url.

boutique/delete_cart_product.php?id_c=116&id_p=6

Je déclare les variable \$_GET en ajoutant "?id_c=" qui correspond à \$_GET["id_c"] (id cart) et \$_GET["id_p"] (id product) à la fin de l'url de la page de traitement le symbole "&" permet de déclarer 2 variable à la suite. Les valeurs des \$_GET["id_c"] et \$_GET["id_p"] correspondront aux données qui seront affiché grâce à la boucle "foreach" pour l'id du cart et du product.

```
<a href="delete_cart_product.php?id_c=<?= $product['id_cart'] ?>&id_p=<?= $product['id_product'] ?>" class="login-button">supprimer du panier</a>
```

Ces données seront récupéré dans la page de traitement de suppression, les données récupérées en \$_GET seront exécuté dans 2 requête préparé "DELETE" , je supprime le panier de l'utilisateur de la table "cart" et "cart_product"


```

$id_cart = $_GET["id_c"];
$id_product = $_GET["id_p"];

$request = $user->getData()->prepare('DELETE FROM cart WHERE id = (?) AND id_user = (?)');
$request->execute(array($id_cart, $_SESSION['id_user']));
$request = $user->getData()->prepare('DELETE FROM cart_product WHERE id_product = (?) AND id_cart = (?)');
$request->execute(array($id_product, $id_cart));
header("Location: cart.php");

```

Une fois la requête terminée, une redirection est faite vers la page panier. Le produit est bien supprimé car il n'est plus affiché.

Le bouton "supprimer le panier" ne supprime pas avec l'aide d'une page de traitement, mais seulement avec un formulaire.

4.3. Page tableau de bord admin (back-office)

La page de gestion de la boutique est accessible seulement à un utilisateur qui aura le rôle d'admin. Depuis cette page il pourra faire la gestion de tous les produits dans la boutique, comme créer des catégories et sous-catégories. Tous les produit y seront affiché.

4.3.1. Ajout d'un produit dans la boutique

Il y a un bouton en début de page "ajouter un produit" qui redirige vers une page dédiée à l'ajout d'un produit. Cette page contient un formulaire avec des champs :

- Nom du produit
- Image du produit
- Prix
- Quantité
- Catégorie
- Sous-catégorie
- Description

La requête qui envoie les données se trouve dans une méthode de la class product.

```
public function addProduct($title, $price, $description)
```

A l'intérieur toutes les vérifications sur les champs sont faites, il y a une vérification particulière sur l'upload de l'image du produit en voici le code :

```

if ($_FILES['image'] && $_FILES['image']['error'] === 0) {

    // liste des extension autorisé
    $allowed = [
        'jpg' => 'image/jpeg',
        'jpeg' => 'image/jpeg',
        'png' => 'image/png',
    ];

    $this->image = $_FILES['image']['name'];
    $filetype = $_FILES['image']['type'];
    $filesize = $_FILES['image']['size'];

    $extension = pathinfo($this->image, PATHINFO_EXTENSION);

    // on vérifie si l'extension de l'image est autorisé
    if (!array_key_exists($extension, $allowed) || !in_array($filetype, $allowed)) {
        die('ERREUR : format de fichier incorrect');
    }

    // on impose une limite a la taille de l'image 1 MB
    if ($filesize > 1024 * 1024) {
        die('Le fichier dépasse 1 Mo');
    }

    // we generate a complete path
    // on génère le chemin vers le dossier
    $newfilename = __DIR__ . '/../upload/" . $this->image;

    // on upload le l'image dans le dossier upload
    if (!move_uploaded_file($_FILES['image']['tmp_name'], $newfilename)) {
        die("L'upload a échoué");
    }
}

```

Dans ce code on sécurise l'upload de fichier image, les extensions autorisées sont :

- Jpeg
- Jpg
- Png

De cette façon il ne sera pas possible d'injecter des fichiers autre que ceux autorisés, Il y a aussi une condition de qui impose une limite de taille de fichier à 1 Mo pour éviter d'envoyer des fichiers volumineux qui pourraient occuper toute l'espace de stockage.

La dernière condition vérifie si le fichier a bien été uploadé dans le dossier upload, dans le cas contraire un message d'erreur s'affiche.

4.3.2. Modifier un produit dans la boutique

Dans la page tableau de bord admin, tous les produits disponibles sont affichés avec leurs image et les détails,

Un produit aura 2 boutons :

- "modifier le produit" ce bouton redirige vers une page "product_change" dans lequel on peut modifier les information du produit.
- "supprimer le produit" qui supprime le produit en passant par une page de traitement de suppression.

La page "product_change" est identique à la page d'ajout de produit à la différence ou on a tous les infos du produit préremplis dans les champs, et l'image du produit actuel est prévisualisé.

Accessibilité

Il y a certaine page qu'il faut rendre inaccessible aux utilisateurs comme la page admin, qui contient des informations sur les utilisateurs.

Le premier moyen que j'ai mis en œuvre est de créer une condition qui permet l'accès a certaine page selon son rôle, par exemple pour le header de la boutique il y aura :

Les utilisateurs non-inscrits :



Ils auront seulement accès au pages nécessaire pour voir les produit du site, il leurs sera proposer de s'inscrire ou de se connecter.

Les utilisateurs inscrits :



Ils auront accès cette fois au panier et a leurs profils et pourront se déconnecter.

L'administrateur de la boutique :



L'admin pourra avoir accès au même page que l'utilisateur inscrit, il a un accès exclusive a la page admin dans lequel il pourra gérer le site.

Tous cela a été possible grâce au condition faite dans le code.

```
<ul class="links">
  <?php if(isset($_SESSION['rights']) && $_SESSION["rights"] == "administrator") :?>
    <li><a href="index.php">Accueil</a></li>
    <li><a href="shop.php">Boutique</a></li>
    <li><a href="admin_dashboard.php">Admin</a></li>
  <?php else: ?>
    <li><a href="index.php">Accueil</a></li>
    <li><a href="shop.php">Boutique</a></li>
  <?php endif; ?>
</ul>
```

Dans ce code je gère l'accès au lien de navigation, selon le rôle de l'utilisateur, cette condition rend la page admin accessible que si l'utilisateur connecté a le rôle "administrator".

Des conditions de ce type seront appliquées à toutes les pages de l'admin, et à la page panier et profil pour l'utilisateur inscrit. C'est un premier moyen de sécurité.

Sécurité

1. Sécurisation du module de connexion

Nous avons mis en place des mesures de sécurité pour sécuriser la boutique. En commençant par notre module de connexion, pour être sûr que l'utilisateur s'inscrit avec une adresse valide, la fonction "filter_var" a été utilisé :

```
if (filter_var($this->email, FILTER_VALIDATE_EMAIL) == false) {
    die("<h1>vous n'avez pas entrer une adresse email valide</h1>");
}
```

Pour vérifier si l'entrée est un email valide, j'ai utilisé la fonction filter_var avec le filtre "FILTER_VALIDATE_EMAIL". Si l'email entrée pas une adresse email valide, filter_var nous retourne false. De cette façon j'empêche l'inscription même s'il y a un changement du type du champ dans l'input de l'email.

J'ai utilisé un moyen de protection du mot de passe pour les utilisateurs qui s'inscrivent dans la boutique :

```
$password = password_hash($password, PASSWORD_DEFAULT);
```

"password_hash()" nous permet de hashé le mot de passe pour protéger la confidentialité des utilisateurs, renforcer la sécurité et réduire les risques en cas de compromission de la base de données.

A la connexion le mot de passe crypté devra être vérifié grâce à la fonction "password_verify()"

```
foreach ($userDatabase as $user) {  
    if ($email === $user['email'] && password_verify($password, $user['password'])) {
```

"password_verify()" est utilisée pour vérifier si le mot de passe correspond à un hachage stocké précédemment. Elle compare un mot de passe non hashé fourni par l'utilisateur avec une valeur de hachage stockée en utilisant l'algorithme de hachage approprié.

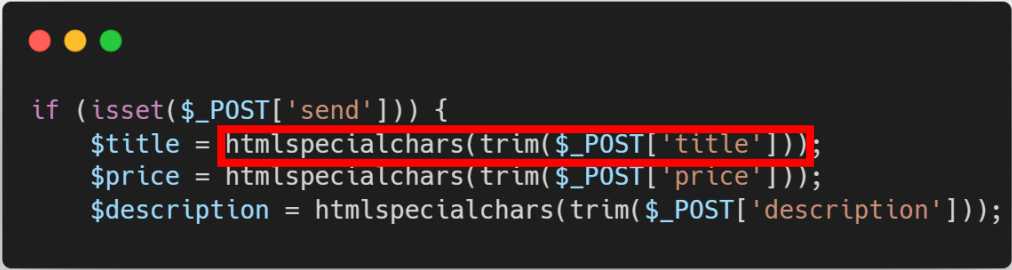
2. Les failles XSS (Cross site scripting)

Une faille XSS (Cross-Site Scripting) est une vulnérabilité de sécurité courante dans les applications web. Elle permet à un attaquant d'injecter du code malveillant (généralement du code JavaScript) dans des pages web consultées par d'autres utilisateurs. Lorsque les utilisateurs accèdent à ces pages infectées, le code malveillant s'exécute dans le navigateur, ce qui peut entraîner diverses conséquences néfastes.

2.1. Failles XSS stokée (Stored XSS)

Dans notre boutique il y a du texte qui est stocké dans la base de données qui devront être affichés dans d'autres pages, il serait possible d'injecter un script malveillant qui serait exécuté lorsqu'une page devrait afficher les textes stockés dans la BDD.

Pour nous prémunir de ces failles dans la boutique, j'ai utilisé la fonction "htmlspecialchars()"



```
if (isset($_POST['send'])) {  
    $title = htmlspecialchars(trim($_POST['title']));  
    $price = htmlspecialchars(trim($_POST['price']));  
    $description = htmlspecialchars(trim($_POST['description']));
```

Cette fonction permet d'échapper les données d'entrée, pour convertir les caractères spéciaux en entités HTML.

2.2. XSS réfléchi (Reflected XSS)

Le code malveillant est intégré dans un lien ou un formulaire qui est envoyé à un serveur web. Le serveur renvoie ensuite le code malveillant dans la réponse, qui est alors exécuté par le navigateur de l'utilisateur qui a cliqué sur le lien ou soumis le formulaire.

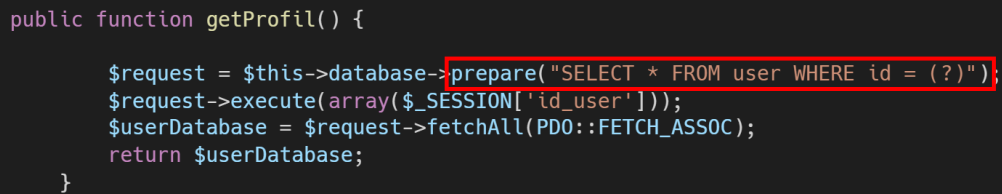
Dans notre cas c'est dans notre barre de recherche qui permet à l'utilisateur de rechercher un produit, les mots tapés s'affichent dans une page pour le résultat de la recherche, il serait donc possible de faire une injection de script directement depuis l'input de recherche, le script s'exécuterait dans la page de résultat de la recherche.

Pour m'en prémunir, "htmlspecialchars()" a été utilisé dans la page de traitement php de l'autocomplétion, sur la variable \$_GET qui récupère les entrées de l'input de l'utilisateur.

3. Les injections SQL

Les injections SQL sont des attaques de sécurité où des données non fiables sont insérées dans des requêtes SQL, permettant ainsi à un attaquant d'exécuter du code SQL malveillant. L'objectif de l'attaquant est de manipuler la requête SQL pour obtenir des informations confidentielles, modifier les données ou compromettre la sécurité du système.

Pour s'en prémunir, nous avons fait des requêtes préparées sur tous les requêtes de la boutique.



```
public function getProfil() {  
    $request = $this->database->prepare("SELECT * FROM user WHERE id = (?)");  
    $request->execute(array($_SESSION['id_user']));  
    $userDatabase = $request->fetchAll(PDO::FETCH_ASSOC);  
    return $userDatabase;  
}
```

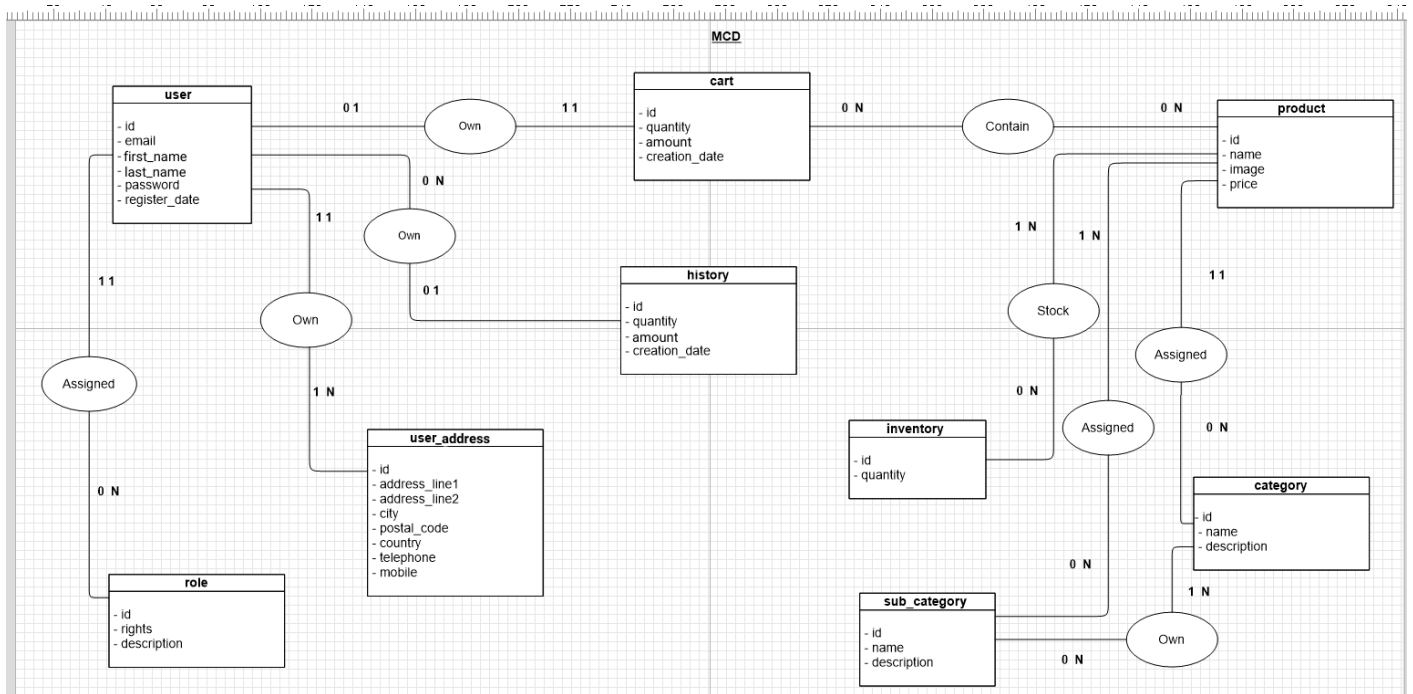
Préparer une requête permet l'échappement des données, ça signifie que les caractères spéciaux ou potentiellement dangereux sont traités de manière sécurisée en tant que valeurs littérales, plutôt que comme du code SQL exécutable.

Annexes

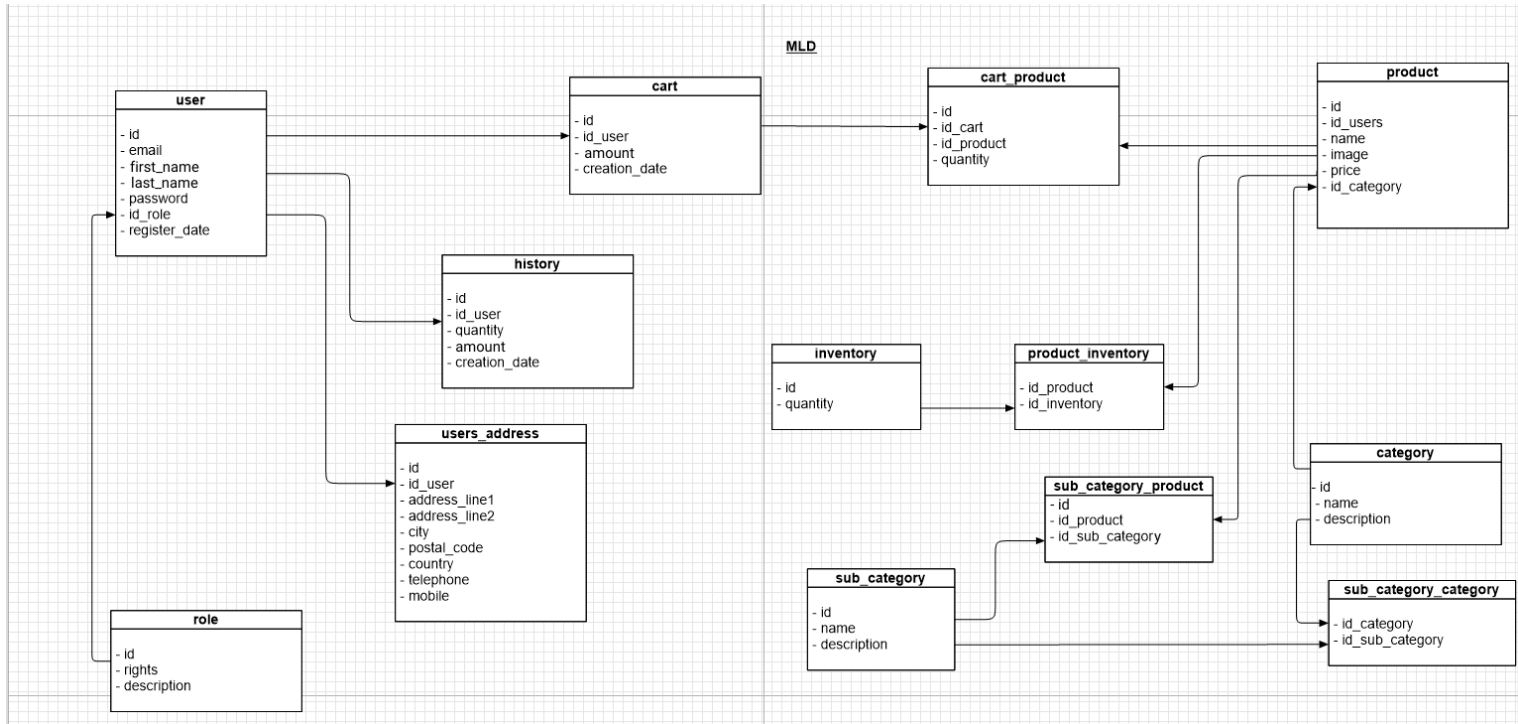
Maquette



Modèle conceptuel des données



Modèle logique de données



Modèle physique de données

