Dossier Projet

Titre : Développeur Web et Web mobile

Boutique en ligne

Ibni-yamine Chabane

Table des matières

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.

Nous avons décidé d'adopter le thème de la vente de livres, car cela correspond à notre intérêt pour la lecture et à notre volonté de promouvoir la culture littéraire. Nous prévoyons d'inclure une variété de genres littéraires, 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 navigation par catégories, la recherche de produits, les descriptions détaillées des livres, les avis des clients, et un panier d'achat permettant aux utilisateurs de sélectionner et d'acheter des livres.

Nous prévoyons également de simuler 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 permettra de mettre en pratique nos compétences en développement web, en conception d'interfaces utilisateur et en gestion de boutique en ligne. Nous

espérons que notre boutique en ligne de vente de livres sera un succès et offrira une expérience agréable aux utilisateurs passionnés de lecture.

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 loger 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 dédier 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 dédier à la modification du produit,
Le contenu de cette page sera exactement la même que la page d'ajout mais les champs les champ seront préremplis avec les informations actuel du produit. Il sera alors possible ajouté 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 sont 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 coordonnée 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 information 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

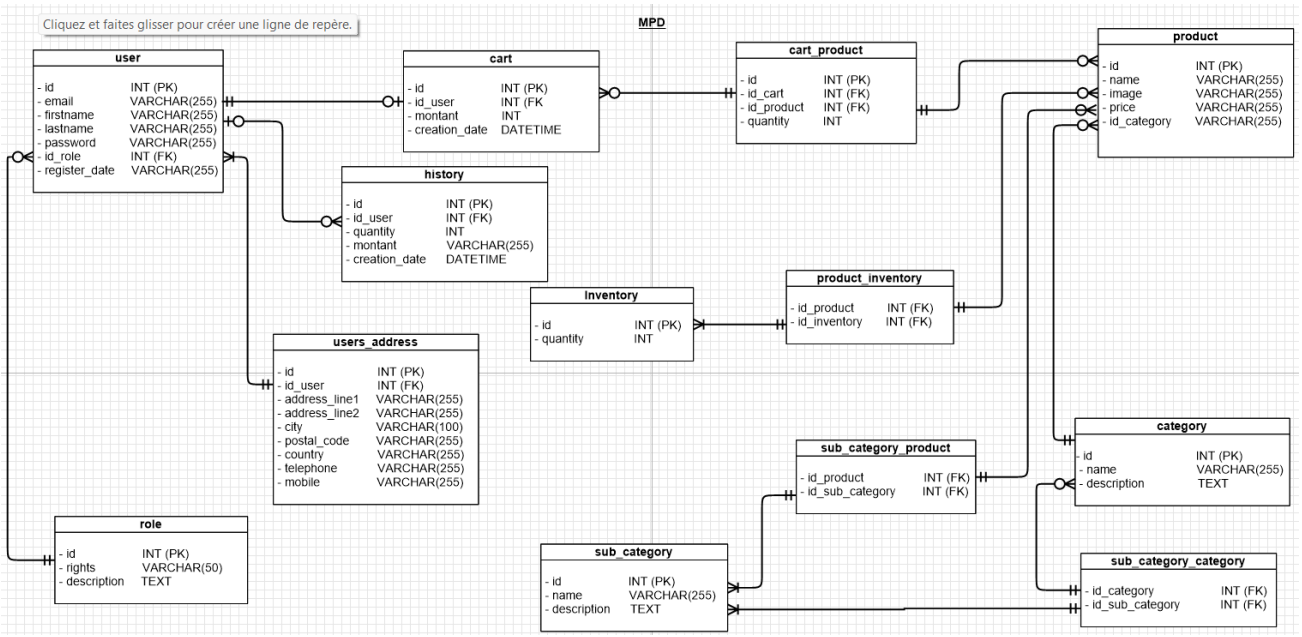
2. Maquette

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'avions 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 sont 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 liée ta 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" et 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 classifier 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 produit de la boutique, j'ai utilisé une méthode 'getProduct'

```php
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 selectionné en $_GET cliqué a 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

```php
  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 récupéré, j'ai accès a 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 lieux d'un nombre

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

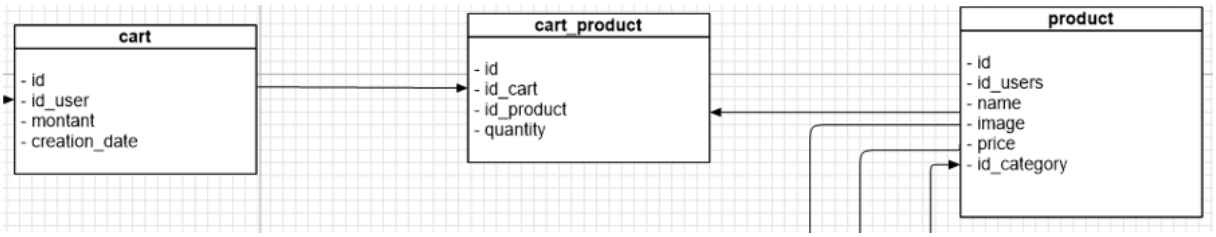
```
if (isset($_POST['submit'])) {

    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 a l'utilisateur de contourner les limitations, car 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 remplis l'utilisateur peut ajouter les produit dans son panier.

Cependant l'ajout du produit dans le panier est complexe car a 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 dédier a tous ce qui est en rapport avec le panier.

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



Tous cela devra être envoyer dans les 3 table si dessus donc il y aura de multiple requête 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

Etape1 :
Je devrais dans un réserver un panier cart , je commence donc avec une requête SQL préparé avec PDO , la requête INSERT pour insérer dans la table cart l'id de utilisateur $_SESSION[id_user] , et le montant sera pour l'instant inséré temporairement avec la valeur 0.

Etape 2 :
Je fait une requête select * dans la table cart quand l'id de l'utlisateur correspond a notre $_SESSION[id_user], et je transforme la donnée récupéré en tableau associatif. Cette action a pour but de récupérer l'id de notre table cart qui correspond a notre $_SESSION[id_user] pour l'étape 3

Etape 3 :
Je fait une requête vers la table liaison 'cart_product' pour y insérerave cla requête INSERT 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 a la quantité du produit qu'on a choisis à l'envoie du formulaire , toutes les infos d'une ligne de la table cart_product sont bien remplis.

Etape 4 :
Je fait une requête SELECT en INNER JOINT sur 4 table : '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 pourvoir faire le calule : $amout = $quantity * $price

Dernière étape :

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

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

```php
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
                                                  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 = (?)'
                                                 );

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

        //derniere étape ajouter le montant dans notre table cart qui correspont 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

4.3. Page ajout de produit (back-office)