CDC PROTOTYPE E-COMMERCE

RESUME

Cahier des charge ayant pour objectif l'étude et l'accompagnement dans la réalisation du projet de site web E-Commerce

Léo LAROU-CHALOT, Florent BERNIER, Teiki BABET

SLAM - DM

EXPRESSION DES BESOINS

Pour ce projet, nous allons devoir réaliser un prototype de site e-commerce en HTML/CSS, JS, PHP et MySQL.

Ce projet aura a pour objectif de nous évaluer sur nos capacités à travailler en groupe et à s'organiser.

Ce site devra permettre aux utilisateurs de se connecter à leur espace.

Le catalogue du site sera accessible en étant connecté ou non, mais l'option d'achat ne sera proposée qu'aux utilisateurs inscrits.

L'administrateur pourra ajouter des articles depuis une section dédiée.

L'utilisateur pourra décider de modifier les informations de son profil ou de visualiser son historique de commandes.

L'utilisateur pourra également accéder à son panier afin de valider une commande.

LES ROLES OBLIGATOIRES

- **Admin** : Il peut ajouter, modifier ou supprimer un produit (**C**reate **R**ead **U**pdate **D**elete)

- **User** (client) : Il peut acheter un produit, modifier son profil et voir son historique de commandes.

DEVELOPPEMENT

UTILISATEUR

CE QU'IL PEUT FAIRE

- Créer un compte
- Se connecter à son espace personnel
- Accéder à son panier
- Acheter les articles du panier
- Accéder à son historique de commandes

UTILISATION

Chaque cas est associé à une fonction, toutes les fonctions user seront dans un fichier **./controler/ft_user.php**. Nous utiliserons un **switch** sur un paramètre **ft** qui sera transmis au travers de l'URL ou d'un formulaire.

ADMINISTRATEUR

CE QU'IL EUT FAIRE

- Gérer les comptes utilisateur
- Ajouter des articles depuis son dashboard
- Modifier des articles depuis son dashboard
- Supprimer des articles depuis son dashboard

UTILISATION

Chaque cas est associé à une fonction, toutes les fonctions user seront dans un fichier **./controler/ft_admin.php**. Nous utiliserons un **switch** sur un paramètre **ft** qui sera transmis au travers de l'URL ou d'un formulaire.

LES CLASSES

CLASS PERSON

Les personnes pouvant se connecter sur le site seront gérées avec des classes de la façon suivante :

- <int> id
- <str> f_name
- <str> l_name
- <str> mail
- <str> password
- <object> pdo

L'attribut $pdo permettra à chaque classe de béneficier d'une connexion à la BDD.

CLASS USER

Les utilisateur seront représentés par une **classe User** qui **héritera** de la **class Person** et possèdera les attributs suivants :

- <str> pseudo
- <str> address
- <int> phone
- <str> profil_picture
- <str> role = user

Les méthodes classiques de getters et de setters seront à définir par la suite (pour la modification de la fiche profil nottament).

CLASS ADMIN

Les administrateurs seront représentés par une **classe Admin** qui **héritera** de la **class Person** et possèdera les attributs suivants :

- <str> role = admin

Les méthodes classiques de getters et de setters seront à définir par la suite (pour la modification des fiches produit nottament).

RESSOURCES ET CODE :

LA CONNEXION

```php
include(__DIR__ . "/../env/env.php");

// Connexion à la base de données
try {
  $conn = new PDO(dsn, dbuser, dbpassword);
  //On définit le mode d'erreur de PDO sur Exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  echo 'Connexion réussie';
  $conn->beginTransaction();
}

// On capture les exceptions si une exception est lancée et on affiche les informations
relatives à celle-ci
catch (PDOException $e) {
  $conn->rollBack();
  echo "Erreur : " . $e->getMessage();
}
```

Pour des raison de sécurité, les constantes de connexion seront dans un dossier **/env.**

Les méthodes **beginTransaction()** désactive l'auto commit tandis que la méthode **rollBack()** annule la transaction en cas d'erreur.

SECURITE ET MOT DE PASSE

```php
// Fonction de hashage du mot de passe retourné par le formulaire d'inscription
$password = password_hash($password, PASSWORD_DEFAULT);

// Fonction de comparaison du mot de passe et d'un hash associé lors de la conneion
password_verify($password, $hash);
```

L'utilisation des méthodes **password_hash()** et **password_verify()** seront nécessaires pour garantir la sécurité des informations recueillies lors de l'inscription et de la connexion.

AJOUT D'UN UTILISATEUR

```php
$email = $_POST["email"];
$password = $_POST["password"];

$password = password_hash($password, PASSWORD_DEFAULT);

$query = $conn->prepare("INSERT INTO users(email, password) VALUES(:email, :password)");
$query->bindParam(':email', $email);
$query->bindParam(':password', $password);
$query->execute();
```

L'utilisation de requête préparée permet la garantie de l'intégrité de l'information ajoutée à notre base de données.

VERIFICATION DE L'EXISTANCE D'UN UTILISATEUR

```php
$query2 = $conn->prepare("SELECT id, email, password FROM users WHERE email = :email");
$query2->execute(['email' => $email]);
$user_exist = $query2->fetch();
if($user_exist){
    echo "L'utilisateur existe !";
} else {
    echo "L'utilisateur n'existe pas !";
}
```

Pour chaque création de compte utilisateur, il faudra vérifier que l'@mail utilisée ne l'est pas déjà. L'utilisation d'une requête préparée pourra permettre cette vérification.