Titre : Développeur Web et Web Mobile

Création d’une boutique en ligne

Samuel Durand

# Table des matières

Compétences du référentiel couvertes par le projet 4

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

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.

Ce projet contient :

Page d’accueil attractive avec plusieurs sections dont une mise en avant des

produits phares sur la page d’accueil / derniers produits mis en ligne

Barre de recherche de produits avec une autocomplétions en javascript asynchrone

Accès à la boutique présentant tous les produits avec la possibilité de les filtrer

par catégorie / sous-catégories sans rechargement de page

Au clic sur chaque produit : une page “détail” complète générée dynamiquement

(nom, image, prix, description, bouton ajouter au panier...)

Un système de création de comptes d’utilisateurs

Module Inscription / Connexion (Utilisation de Javascript et Asynchrone

obligatoire dans cette partie)

Page de gestion du profil utilisateur (Récapitulatif et possibilité de

modifier ses informations, voir son historique d’achat, consultation de panier...)

Espace tableau de bord Administrateur:

Gestion des produits à l’aide de back office (Ajout / Suppression /

Modifications de produits, stocks...)

Gestion des catégories et des sous catégories de produits (Ajout / Suppression / Modifications...)

Système de validation du panier

Spécifications fonctionnelles

Description de l’existant

Arborescence du site

L’arborescence du site se décline comme suit :

- Page d’accueil
- Page d’inscription
- Page de connexion
- Page de profil
- page administration
- Page de produit
- Page pour afficher les produits de barre de recherche

- Page de panier
- Page de validation de panier
- Page de confirmation de panier

## Description des fonctionnalités

### 1.Inscription via un module de connexion

Pour l’inscription, j’ai fait un formulaire qui contient le nom, prénom, adresse email et mot de passe.

En ce qui concerne la sécurité, j'ai mis en place plusieurs mesures pour protéger les informations des utilisateurs. Tout d'abord, j'ai implémenté des contrôles pour garantir que chaque pseudonyme est unique, évitant ainsi les conflits dans la base de données. De plus, j'ai veillé à ce que les mots de passe soient stockés de manière sécurisée en utilisant des techniques de hachage robustes.

Le hachage est un processus qui transforme le mot de passe en une chaîne de caractères illisible, de sorte que même en cas d'accès non autorisé à la base de données, les mots de passe réels restent confidentiels. Lorsqu'un utilisateur se connecte ultérieurement, le mot de passe saisi est haché à nouveau et comparé au hachage enregistré dans la base de données pour vérifier l'authenticité.

### 2.page d’accueil

Derniers produits ajoutés : Cette section présente les produits les plus récents ajoutés à notre catalogue. Chaque produit est accompagné d'une image attrayante, d'une description et du prix.

Produits les plus réputés : Dans cette section, je mais en évidence les produits les plus populaires.

### 3 . Page de profil

La page de profil vous permet de mettre à jour vos informations personnelles telles que votre nom, prénom, adresse email et mot de passe. Pour chaque modification, vous devrez confirmer votre mot de passe actuel.

Pour valider chaque modification, il faut confirmer votre mot de passe actuel. Cela garantit que l’user est bien le propriétaire légitime du compte et évite toute modification non autorisée.

## 4. page d’administration

Sur la page d'administration, il est possible de visualiser tous les produits ajoutés, les modifier ou les supprimer selon les besoins de l’admin. De plus, vous avez la possibilité d'ajouter de nouveaux produits en spécifiant l'image, la description, le prix et le stock.

Voici comment la page d'administration est organisée :

Liste des produits :  une liste complète de tous les produits présents dans la base de donner. Chaque produit est affiché avec son nom, son image, sa description, son prix et son stock actuel. Cette vue d'ensemble permet de prendre rapidement connaissance de l'ensemble des produits.

Modification des produits : il est possible de sélectionner un produit spécifique dans la liste pour le modifier. Une fois sélectionné, on peut avoir accès à toutes les informations du produit, y compris l'image, la

description, le prix et le stock. Il est possible de modifier si nécessaires et les enregistrer pour mettre à jour les détails du produit.

Suppression des produits :  retirer un produit de la base de donner, suppression directement depuis la page d'administration. Une fois confirmée, la suppression sera effectuée et le produit sera retiré de la liste.

Ajout de nouveaux produits : il est possible d'ajouter de nouveaux produits à la base de donner. Pour cela, il faut renseigner les informations requises telles que l'image du produit, sa description, son prix et son stock disponible. Une fois les détails saisis, vous pouvez enregistrer le nouveau produit et il sera ajouté à la liste.

### 5. page de produit

La page du produit est une page simple qui récupère les données du produit à partir de la base de données. Elle affiche les informations spécifiques d'un produit sélectionné.

Une fois les données récupérées, elles sont affichées de manière claire et organisée sur la page du produit. On peut voir une image du produit, lire sa description, connaître son prix actuel et vérifier le niveau de stock disponible. Cette page vous permet d'obtenir toutes les informations essentielles sur un produit spécifique.

### 6. Page pour afficher les produits de barre de recherche

Dans ce projet, j'ai créé une fonctionnalité de barre de recherche avec une fonction d'autocomplétion. Lorsque les utilisateurs saisissent des mots-clés dans la barre de recherche, une liste déroulante s'affiche avec des suggestions contenant l'image, le nom du produit et le prix correspondant.

De plus, j'ai mis en place une page dédiée pour afficher les résultats de recherche. Cette page récupère les termes saisis dans le formulaire de recherche et les inclut dans l'URL. Ensuite, elle effectue une requête dans la base de données pour trouver les produits correspondants aux termes de recherche.

Sur la page des résultats, plusieurs scénarios sont possibles :

1. Si des produits correspondants sont trouvés, ils sont affichés avec leur image, leur nom et leur prix. Les utilisateurs peuvent cliquer sur chaque résultat pour accéder à la page détaillée du produit.

2. Si aucun produit correspondant n'est trouvé, un message indiquant "Aucun produit disponible avec ce nom" est affiché pour informer les utilisateurs qu'aucun résultat n'a été trouvé.

Dans les deux cas, j'ai veillé à ce que les résultats de recherche soient affichés de manière claire et conviviale, en mettant en valeur les informations essentielles telles que l'image, le nom du produit et le prix.

Cela permet aux utilisateurs de rechercher des produits spécifiques, d'obtenir des suggestions grâce à l'autocomplétion et de voir les résultats correspondants sur une page dédiée. Cette fonctionnalité améliore l'expérience de recherche et facilite la découverte des produits disponibles dans la base de données.

7. Page de validation de panier

Pour valider le panier, l'utilisateur doit remplir un formulaire contenant les informations nécessaires, telles que l'adresse de facturation, l'adresse principale et le numéro de téléphone. Ensuite, pour effectuer le paiement, un formulaire simple est proposé, comprenant le numéro de carte, la date d'expiration et le code de sécurité (CVV).

Une fois que l'utilisateur a validé le panier, les informations du panier sont stockées dans la session pour une récupération plus facile. L'identifiant du panier est ensuite inclus dans l'URL pour faciliter le suivi et l'accès aux détails de la commande ultérieurement.

Lorsque l'utilisateur soumet le formulaire de validation du panier, toutes les informations fournies sont vérifiées pour s'assurer qu'elles sont correctes et complètes. Si toutes les données sont valides, l'utilisateur est redirigé vers une page de confirmation de commande.

8.page de confirmation de panier.

Pour la confirmation du panier, tel qu'expliqué sur la page du panier, j'utilise l'ID du panier récupéré dans l'URL pour rediriger l'utilisateur vers la page de confirmation. En utilisant un script JavaScript, je récupère cet ID de section pour effectuer certaines actions.

Une fois que l'utilisateur est redirigé vers la page de confirmation, j'affiche un message de validation de commande pour confirmer que la commande a été passée avec succès. Ce message peut inclure des informations telles que "Commande validée avec succès".

En outre, pour offrir une meilleure expérience utilisateur, j'ai mis en place un mécanisme pour stocker le numéro de commande dans le stockage local (localStorage) du navigateur. Cela permet de garder une trace du numéro de commande même si l'utilisateur actualise la page ou ferme le navigateur. À chaque validation de commande, j'incrémente le numéro de commande afin qu'il augmente en fonction du nombre d'achats effectués.

Lors de l'affichage de la page de confirmation, je récupère le numéro de commande à partir du stockage local et l'affiche à l'utilisateur. Cela lui donne une référence unique pour sa commande et facilite toute correspondance ou suivi futur.

En résumé, après avoir récupéré l'ID du panier depuis l'URL, l'utilisateur est redirigé vers la page de confirmation où un message de validation de commande est affiché. J'utilise un script JavaScript pour récupérer l'ID de section, stocker le numéro de commande dans le localStorage et afficher le numéro de commande à l'utilisateur. Cela améliore l'expérience utilisateur en fournissant une confirmation claire et une référence pratique pour la commande passée.

Voici les spécifications techniques et les choix techniques pour le projet :

Langages utilisés pour la partie back-end :

Le projet a été réalisé en utilisant le langage PHP pour la logique back-end et le traitement des données.

La base de données utilisée est une base de données SQL pour stocker et récupérer les informations nécessaires.

Technologies utilisées pour la partie front-end :

Le projet a été réalisé en utilisant HTML et CSS pour la structure et la présentation de l'interface utilisateur.

JavaScript a été utilisé pour ajouter des fonctionnalités dynamiques, récupérer et afficher les données côté client.

Environnement de développement :

Éditeur de code : Visual Studio Code (ou un autre éditeur de code de votre choix).

Outils de versioning : Git pour la gestion du code source et GitHub pour le stockage et la collaboration.

Maquettage : Figma (ou un autre outil de conception) pour créer des maquettes et des prototypes de l'interface utilisateur.

Ces choix techniques offrent une base solide pour le développement du projet. PHP est un langage populaire et polyvalent pour la logique back-end, tandis que HTML, CSS et JavaScript sont les technologies de base

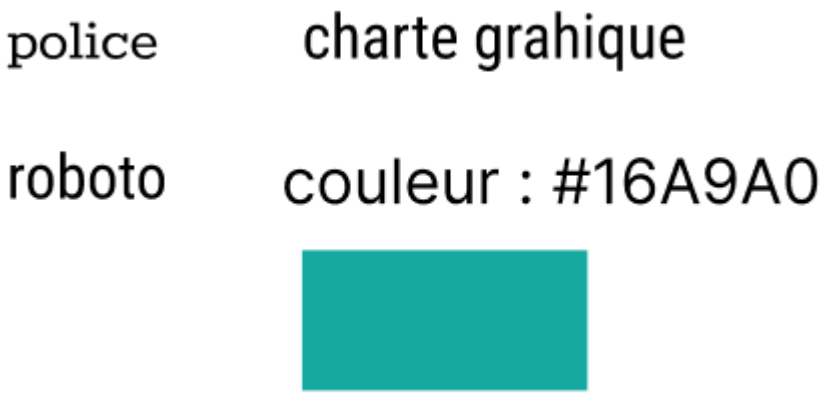
pour la création de l'interface utilisateur et l'interaction avec les utilisateurs. Utiliser Git et GitHub facilite la gestion du code source et la collaboration entre les membres de l'équipe de développement. Enfin, Figma permet de créer des maquettes et des prototypes pour mieux visualiser et concevoir l'interface utilisateur.

Réalisations

1.Charte graphique

Les polices d’écriture sont les suivantes: Roboto et Pacifico.

La couleur dominante est la suivante : #16A9A0



2.maquette

J'ai commencé par réaliser la maquette de l'interface à l'aide de Figma. Pour m'assurer d'obtenir un design épuré et esthétiquement agréable, j'ai choisi de travailler dans des tons de noir et de gris en premier lieu. Cette approche me permettait de me concentrer sur la structure et la disposition des éléments de l'interface sans être influencé par les couleurs.

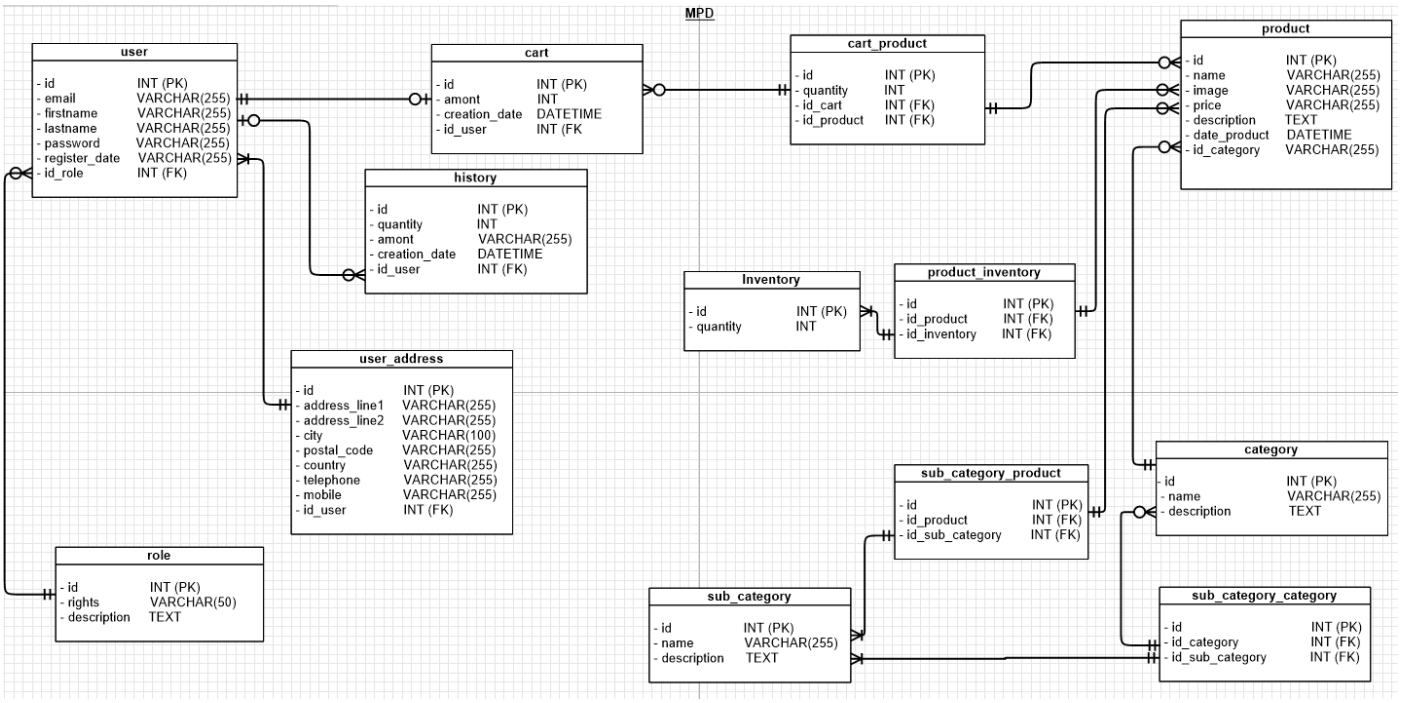
Une fois que j'étais satisfait du design en noir et gris, j'ai progressivement ajouté les couleurs appropriées pour donner vie à l'interface. L'ajout de couleurs a été soigneusement pensé pour créer des contrastes et guider l'attention des utilisateurs vers les éléments importants

En parallèle, j'ai également créé une version mobile de l'interface à côté de la version pour ordinateur de bureau. Cela me permettait d'adapter le design et l'agencement des éléments pour répondre aux contraintes spécifiques des appareils mobiles. En travaillant simultanément sur les deux versions, j'ai pu m'assurer que l'interface était fonctionnelle et attrayante, quelle que soit la plateforme utilisée.

En résumé, en commençant par un design en noir et gris, j'ai pu évaluer la structure et la disposition des éléments de l'interface de manière neutre. Ensuite, j'ai ajouté les couleurs appropriées pour créer une expérience visuelle attrayante. De plus, en créant une version mobile en parallèle, j'ai pu adapter l'interface aux différents formats d'écran. Cette approche m'a permis de concevoir une interface esthétique et adaptée à une variété de dispositifs.

## 3.Conception de la base de données

Au regard des fonctionnalités demandées pour le projet, j’ai développé la base de données suivante :




Vous trouverez également dans les annexes  le modèle conceptuel de données ainsi que le modèle logique de données.

Dans notre base de données, nous avons une table principale appelée "user" qui contient les informations des utilisateurs, telles que leur nom, prénom et adresse e-mail. Cette table est reliée à plusieurs autres tables qui enrichissent notre système.

Une de ces tables est "cart", qui stocke les informations relatives au panier d'un utilisateur. Chaque utilisateur possède un identifiant unique dans la table "user", et cet identifiant est utilisé comme clé étrangère dans la table "cart" pour associer un panier à un utilisateur spécifique. Ainsi, nous pouvons retrouver les produits du panier d'un utilisateur en fonction de son identifiant.

En plus de la table "cart", nous avons la table "cart_product" qui joue le rôle d'une table de liaison. Elle regroupe les informations relatives aux

produits présents dans un panier spécifique. La table "cart_product" contient des colonnes telles que l'identifiant du panier, la quantité des produits et l'identifiant du produit lui-même. Cela nous permet de connaître quels produits sont présents dans chaque panier.

De plus, la table "cart_product" est liée à la table "product" qui contient des informations détaillées sur les produits. Cette table inclut des colonnes telles que l'identifiant du produit, son nom, son prix, sa description, sa date de création et son identifiant de catégorie. Grâce à la table "cart", nous pouvons relier ces tables précédentes afin de procéder au paiement du panier spécifique d'un client et garantir que seul son propre panier est traité.

Cette structure de base de données nous permet de gérer efficacement les informations des utilisateurs, les paniers et les produits associés.

## 4.Extraits de code

### 1.inscription module de connexion

Pour ce module, j'ai créé des fonctions pour l'inscription et la connexion. Dans la fonction d'inscription, j'ai inclus une condition qui vérifie si l'utilisateur est déjà inscrit en consultant la base de données à l'aide de l'adresse e-mail fournie. Si l'e-mail est déjà présent dans la base de données, la requête est arrêtée. Sinon,

si l'e-mail n'est pas trouvé dans la base de données, l'inscription est effectuée et l'utilisateur est redirigé vers la page de connexion.

De plus, j'ai implémenté un système de rôles pour les utilisateurs. Par défaut, un utilisateur normal est attribué le numéro de rôle '2', ce qui lui accorde des privilèges standard sur le site. Le super administrateur, en revanche, est attribué le numéro de rôle '1', ce qui lui confère tous les privilèges et droits sur le site.

```php
    public function register($email, $firstname, $lastname, $password) {



        $request = $this->database->prepare('SELECT * FROM user');
        $request->execute(array());
        $userDatabase = $request->fetchAll(PDO::FETCH_ASSOC);
        $this->email = $email;
        $password = password_hash($password, PASSWORD_DEFAULT);
        $firstname;
        $emailOk = false;
        $right = "";
        $role_id = 2; // le role id 2 correspond a un utilisateur normale.


        foreach ($userDatabase as $user) {

            if ( $this->email == $user['email']){
                $this->message = "cette utilisateur existe déjà";
                $emailOk = false;
                break;
            } else {
                $emailOk = true;
            }
        }


        if ($emailOk == true){
        //on créer l'utilisateur.
          $request = $this->database->prepare("INSERT INTO user(email, first_name,
last_name, password, id_role, register_date) VALUES (?, ?, ?, ?, ?,NOW())");
          $request->execute(array($this->email, $firstname, $lastname, $password,
$role_id));
          $this->message = "inscription réussi";
          header('location : login.php');
        }


    }
```

Asynchrone de la fonction register

```
const registerForm = document.getElementById("register-form");

registerForm.addEventListener("submit", async function (e) {
  e.preventDefault();

  const formData = new FormData(registerForm);
  formData.append("submit", "true");

  try {
    const response = await fetch("register.php", {
      method: "POST",
      body: formData,
    });

    if (response.ok) {
      const result = await response.json();
      console.log(result);

      if (result.success) {
        alert("Vous êtes inscrit !");
        window.location.replace("login.php");
      } else {
        alert(result.message);
      }
    } else {
      console.error("Erreur lors de l'envoi du formulaire:", response.statusText);
    }
  } catch (error) {
    console.error("Erreur lors de l'envoi du formulaire:", error);
  }
});
```

pour ce script l'ID du formulaire d'inscription, "register-form", est récupéré et stocké dans une variable appelée "registerForm". Cela me permet de cibler spécifiquement ce formulaire pour y ajouter des fonctions.

Ensuite, j’ai ajouté un écouteur  au bouton de soumission du formulaire. Lorsque l'utilisateur clique sur ce bouton, une fonction asynchrone est déclenchée pour gérer l'envoi des données du formulaire.

Dans cette fonction, j’utilise la méthode "preventDefault()" pour empêcher le comportement par défaut du formulaire, qui est le rechargement de la page. Cela permet d'effectuer une soumission asynchrone sans interruption de la navigation de l'utilisateur.

Je crée ensuite un objet FormData pour collecter toutes les données du formulaire. Cela me permet de récupérer facilement les valeurs des champs du formulaire.

Ensuite, j’utilise la méthode Fetch pour envoyer une requête asynchrone au fichier PHP "register.php" via la méthode POST. Les données du formulaire sont incluses dans le corps de la requête.

Une fois la requête envoyée, je vérifie si la réponse est "ok" (statut 200) en utilisant la propriété "ok" de l'objet réponse. Si la réponse est positive, cela signifie que le formulaire a été envoyé avec succès.

Si la réponse est positive, j’utilise la méthode "json()" pour extraire les données JSON de la réponse. Ces données JSON contiennent des informations renvoyées par le fichier PHP, telles que le statut de succès et les messages d'erreur ou de confirmation.

Si la propriété "success" de l'objet résultat est définie sur "true", cela signifie que l'inscription a été réussie. Dans ce cas, une alerte s'affiche pour informer l'utilisateur de son inscription réussie. Ensuite, la page est redirigée vers la page de connexion "login.php" pour permettre à l'utilisateur de se connecter.

Si la propriété "success" de l'objet résultat n'est pas définie sur "true", cela indique qu'il y a eu une erreur lors de l'envoi du formulaire. Dans ce cas, un message d'erreur est affiché dans la console du navigateur pour faciliter le débogag

2.connexion

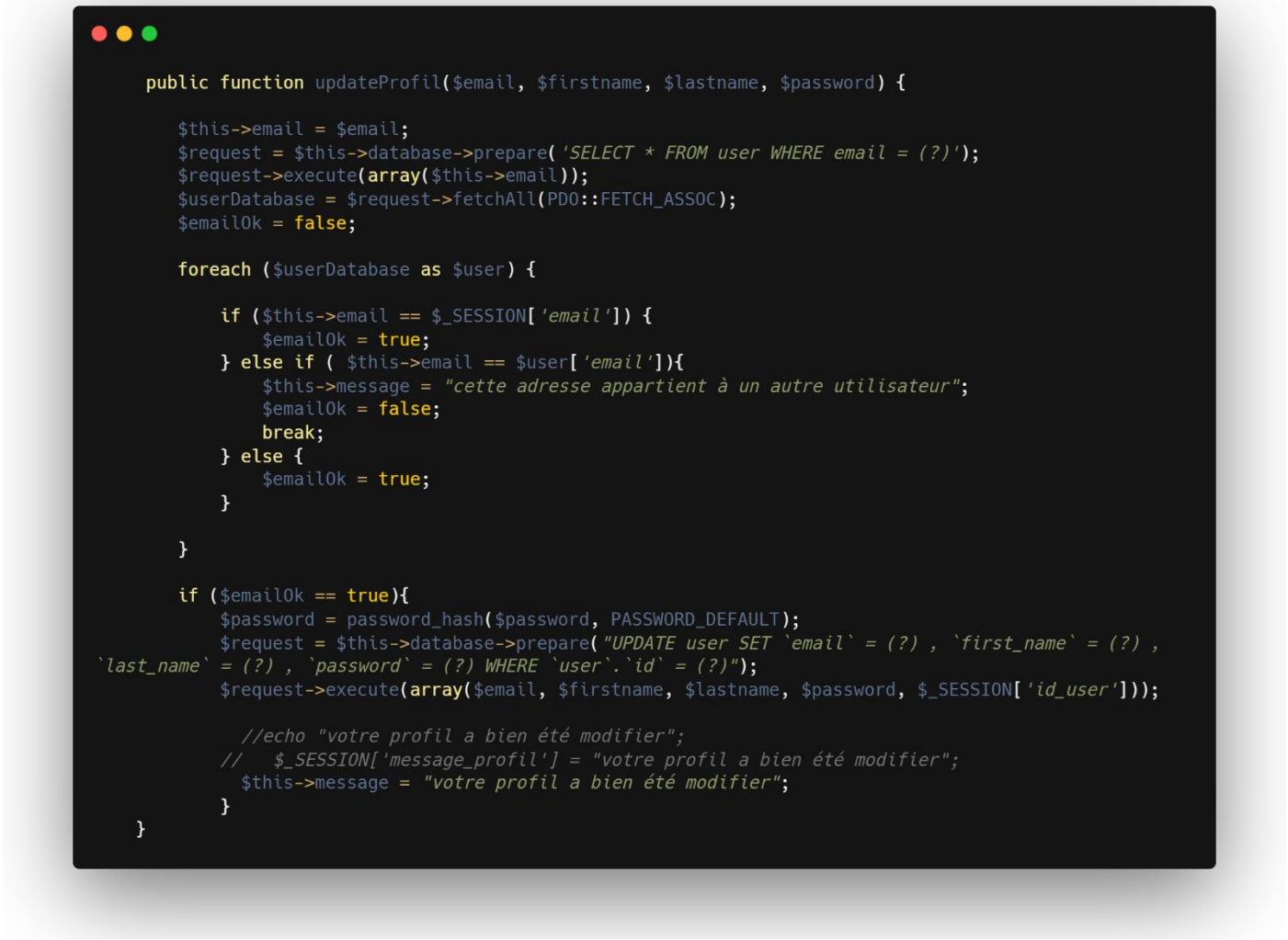
```
    public function connection($email, $password) {
        $request = $this->database->prepare('SELECT u.`id` , `email` , `first_name` , `last_name` ,
`password` , `rights`
                                            FROM user u
                                            INNER JOIN role
                                            ON role.id = u.id_role
                                            WHERE `email` = (?)');
        $request->execute(array($email));
        $userDatabase = $request->fetchAll(PDO::FETCH_ASSOC);

        $this->email = $email;
        $password;
        $logged = false;

        foreach ($userDatabase as $user) {

            if ($email === $user['email'] && password_verify($password, $user['password'])) {
                $_SESSION['email'] = $email;
                $id = $user['id'];
                $_SESSION['id_user'] = $id;
                $_SESSION["firstname"] = $user["first_name"];
                $_SESSION["rights"] = $user["rights"];
                $logged = true;
                break;

            } else {
                $logged = false;
            }
        }

        if( $logged ) {
            header('Location: index.php');
        } else {
            $this->message = "erreur dans l'email ou le password";
        }

    }}
  return go(f, seed, [])
}
```

Pour la fonction de connexion, j'ai créé une fonction appelée "connexion". Dans cette fonction, j'ai utilisé une requête pour sélectionner l'e-mail, le prénom et le nom de la table "user". Ensuite, j'ai effectué une jointure avec la table "role_id" pour récupérer les droits de l'utilisateur.

Ensuite, j'ai inclus des conditions pour vérifier si toutes les conditions requises pour la connexion sont remplies. Si l'une de ces conditions n'est pas valide, j'ai arrêté la connexion en utilisant la commande "break" pour sortir de la boucle.

Si toutes les conditions sont satisfaites, j'ai continué la requête et redirigé l'utilisateur vers la page d'accueil.

3.page de profil



```php
public function updateProfil($email, $firstname, $lastname, $password) {

    $this->email = $email;
    $request = $this->database->prepare('SELECT * FROM user WHERE email = (?)');
    $request->execute(array($this->email));
    $userDatabase = $request->fetchAll(PDO::FETCH_ASSOC);
    $emailOk = false;

    foreach ($userDatabase as $user) {

        if ($this->email == $_SESSION['email']) {
            $emailOk = true;
        } else if ( $this->email == $user['email']){
            $this->message = "cette adresse appartient à un autre utilisateur";
            $emailOk = false;
            break;
        } else {
            $emailOk = true;
        }

    }

    if ($emailOk == true){
        $password = password_hash($password, PASSWORD_DEFAULT);
        $request = $this->database->prepare("UPDATE user SET `email` = (?) , `first_name` = (?) ,
`last_name` = (?) , `password` = (?) WHERE `user`.`id` = (?)");
        $request->execute(array($email, $firstname, $lastname, $password, $_SESSION['id_user']));

          //echo "votre profil a bien été modifier";
        //   $_SESSION['message_profil'] = "votre profil a bien été modifier";
          $this->message = "votre profil a bien été modifier";
        }
}
```

Tout d'abord, la fonction reçoit en paramètres les nouvelles valeurs de l'e-mail, du prénom, du nom et du mot de passe de l'utilisateur. Ces valeurs sont essentielles pour mettre à jour correctement le profil.

Ensuite, la fonction met à jour la variable $this->email avec la nouvelle valeur de l'e-mail. Cela permet de travailler avec la dernière adresse e-mail fournie par l'utilisateur.

Pour assurer l'intégrité des données, la fonction exécute une requête SQL pour sélectionner toutes les lignes de la table "user" où l'e-mail correspond à la nouvelle valeur de $this->email. Les résultats de cette requête sont stockés dans la variable $userDatabase pour une utilisation ultérieure.

Pour garantir la validité de l'e-mail, j’utilise une boucle foreach pour parcourir les résultats de la requête précédente. Dans cette boucle, nous effectuons plusieurs vérifications.

Tout d'abord, je vérifie si l'e-mail correspond à celui stocké dans la session ($_SESSION['email']). Si c'est le cas, cela signifie que l'utilisateur est autorisé à utiliser cet e-mail, et je définie la variable $emailOk sur true.

Ensuite, si l'e-mail correspond à l'e-mail d'un autre utilisateur dans la base de données, j’affiche un message d'erreur approprié et je définie $emailOk sur false. J’utilise également l'instruction break pour sortir de la boucle, car j’ai identifié un problème.

Si aucune des conditions précédentes n'est remplie, cela signifie que l'e-mail est unique et valide. Je définie donc $emailOk sur true.

Une fois que j’ai vérifié la validité de l'e-mail, je hash le mot de passe fourni par l'utilisateur à l'aide de la fonction password_hash.

Cela garantit que le mot de passe est stocké de manière sécurisée dans notre base de données.

Ensuite, je prépare une requête de mise à jour pour mettre à jour l'e-mail, le prénom, le nom et le mot de passe de l'utilisateur dans la base de données. Cette requête est exécutée en utilisant les nouvelles valeurs fournies et l'identifiant de l'utilisateur stocké dans $_SESSION['id_user'].

Si la mise à jour réussit, je définie un message de confirmation approprié dans $this->message pour indiquer que le profil a été modifié avec succès.

4. admin

```php
public function getAllProducts() {

    $request = $this->getDatabase()->prepare('SELECT image, price, quantity, product.name
                                              AS productName, category.name
                                              AS categoryName,
                                              product.id
                                              AS productId, inventory.id
                                              AS inventoryId, product_inventory.id
                                              AS prod_inv_id,
                                              date_product
                                              FROM product
                                              INNER JOIN product_inventory
                                              on product.id = product_inventory.id_product
                                              INNER JOIN inventory
                                              on inventory.id = product_inventory.id_inventory
                                              INNER JOIN category
                                              on product.id_category = category.id
                                              ORDER BY date_product DESC');

                                              $request->execute(array());
                                              return $request->fetchAll(PDO::FETCH_ASSOC);

}
```

Tout d'abord, je prépare une requête SQL pour sélectionner les colonnes spécifiques que je souhaite récupérer de la base de données. Ces colonnes incluent des informations telles que l'image du produit, son prix, sa quantité, son nom, le nom de la catégorie à laquelle il appartient, ainsi que divers identifiants liés à la structure de notre base de données.

Ensuite, j’utilise plusieurs jointures pour relier les tables appropriées. La première jointure est réalisée entre les tables "product" et "product_inventory", en utilisant la colonne "id" du produit. Cette jointure me permet de récupérer les informations relatives aux stocks et aux inventaires de chaque produit.

Ensuite, j’effectue une jointure entre les tables "inventory" et "product_inventory" en utilisant la colonne "id" de l'inventaire. Cela me permet d'associer chaque produit à son inventaire correspondant.

Enfin, j’effectue une dernière jointure entre les tables "product" et "category" en utilisant la colonne "id_category" du produit. Cette jointure me permet de récupérer le nom de la catégorie à laquelle appartient chaque produit.

Pour obtenir les résultats dans l'ordre souhaité, j’utilise la clause ORDER BY avec la colonne "date_product". Cela me permet d'afficher les produits les plus récents en premier.

Une fois que la requête est prête, je l'exécute en utilisant la méthode execute(array()). Cela me permet d'exécuter la requête sans aucun paramètre supplémentaire. Les résultats de la requête sont ensuite récupérés à l'aide de fetchAll(PDO::FETCH_ASSOC).

5.panier

```php
    public function addProductInCart(int $quantity) {

        $request = $this->getDatabase()->prepare('INSERT INTO cart(id_user , amount, creation_date)
                                                  VALUES (?, ?, NOW())'
                                                  );

        $request->execute(array($_SESSION['id_user'], 0));
        $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'];

        $request3 = $this->getDatabase()->prepare('INSERT INTO cart_product(id_cart , id_product ,
quantity)
                                                 VALUES (?, ?, ?)'
                                                 );

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

        $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;

        $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";

    }
```

la fonction addProductInCart  reçoit en paramètre la quantité du produit à ajouter au panier. Cette quantité est fournie par l'utilisateur lors de l'interaction avec le formulaire.

Ensuite, je prépare une requête SQL pour insérer une nouvelle ligne dans la table "cart" de notre base de données. Cette nouvelle ligne représente le panier de l'utilisateur. J'utilise la valeur de $_SESSION['id_user'] pour associer le panier à l'utilisateur connecté. La quantité est initialisée à 0 et la date de création est automatiquement définie à la date et heure actuelles avec la fonction MySQL NOW().

Après avoir exécuté la première requête, je prépare une autre requête pour récupérer l'identifiant du panier qui vient d'être créé. J'utilise la clause ORDER BY id DESC pour obtenir le dernier panier créé par l'utilisateur, car il peut avoir plusieurs paniers dans l'historique. Je stock l'identifiant du panier dans la variable $id_cart pour une utilisation ultérieure.

Ensuite, je prepare une requête pour insérer une nouvelle ligne dans la table "cart_product". Cette ligne représente le lien entre le panier et le produit spécifié par l'utilisateur. J'utilise les valeurs de $id_cart, $_GET['id'] (l'identifiant du produit) et la quantité spécifiée pour cette opération.

Après avoir ajouté le produit dans le panier, je prépare une requête pour récupérer des informations spécifiques sur l'utilisateur, le panier, le produit et la quantité associée. J'utilise plusieurs jointures entre les tables "user", "cart", "cart_product" et "product" pour obtenir les informations nécessaires. Nous filtrons les résultats en utilisant les valeurs de $_SESSION['id_user'] et $id_cart.

Ensuite, je récupère le prix du produit à partir des résultats de la requête précédente et je calcule le montant total en multipliant le prix par la quantité spécifiée.

Je mais ensuite à jour le montant total du panier dans la table "cart" en utilisant une requête de mise à jour. J'utilise les valeurs de $amount, $_SESSION['id_user'] et $id_cart pour effectuer la mise à jour.

Enfin, je définie le message de retour $this->message pour indiquer à l'utilisateur que le produit a été ajouté avec succès dans son panier.

## 6.confirmer la commande

```php
    public function __construct(PDO $connexion) {
        $this->connexion = $connexion;
        $this->testCreditCards = array(
            array(
                'number' => '4242424242424242',
                'expiration' => '12/2024',
                'cvv' => '123'
            )
        );
    }

    public function addAddress($id_user, $adresse_line1, $adresse_line2, $city, $postal_code, $country,
$telephone, $mobile, $credit_card_number, $expiration_date, $cvv) {
        if (!$this->isTestCreditCardValid($credit_card_number, $expiration_date, $cvv)) {
            exit();
        }

        try {
            $query = "INSERT INTO `users_address`(`adresse_line1`, `adresse_line2`, `city`,
`postal_code`, `country`, `telephone`, `mobile`, `id_user`) VALUES (:adresse_line1, :adresse_line2,
:city, :postal_code, :country, :telephone, :mobile, :id_user)";

            $stmt = $this->connexion->prepare($query);
            $stmt->bindParam(":adresse_line1", $adresse_line1);
            $stmt->bindParam(":adresse_line2", $adresse_line2);
            $stmt->bindParam(":city", $city);
            $stmt->bindParam(":postal_code", $postal_code);
            $stmt->bindParam(":country", $country);
            $stmt->bindParam(":telephone", $telephone);
            $stmt->bindParam(":mobile", $mobile);
            $stmt->bindParam(":id_user", $id_user);

            if ($stmt->execute()) {
                echo "L'adresse a été ajoutée avec succès !";

                $deleteCartQuery = "DELETE FROM `cart` WHERE id_user = :id_user";
                $deleteCartStmt = $this->connexion->prepare($deleteCartQuery);
                $deleteCartStmt->bindParam(":id_user", $id_user);

                if ($deleteCartStmt->execute()) {
                    echo "Le panier a été supprimé.";
                } else {
                    echo "Erreur lors de la suppression du panier.";
                }

                $_SESSION['cart_id'] = $_SESSION['cart']['id'];
                header("location: succes.php?cart_id=" . $_SESSION['cart_id']);
                exit();
            } else {
                echo "Erreur lors de l'ajout de l'adresse.";
            }
        } catch (PDOException $e) {
            echo "Erreur : " . $e->getMessage();
        }
    }

    private function isTestCreditCardValid($number, $expiration, $cvv) {
        foreach ($this->testCreditCards as $card) {
            if ($number === $card['number'] && $expiration === $card['expiration'] && $cvv ===
$card['cvv']) {
                return true;
            }
        }
        return false;
    }
}
```

J’utilise la method addAddress pour ajouter une adresse à la table users_address dans la base de données. Elle reçoit en paramètres toutes les informations nécessaires pour une adresse (id utilisateur, ligne d'adresse 1, ligne d'adresse 2, ville, code postal, pays, téléphone, mobile, numéro de carte de crédit, date d'expiration et code de vérification).

Avant d'effectuer l'insertion de l'adresse, la méthode vérifie si la carte de crédit fournie correspond à une carte de crédit de test valide en appelant la méthode isTestCreditCardValid. Si la carte de crédit n'est pas valide, le script s'arrête avec la commande exit().

Ensuite, la méthode prépare une requête SQL pour insérer les informations de l'adresse dans la table users_address. Les valeurs des paramètres sont liées aux paramètres de la requête à l'aide de la méthode bindParam.

Si l'insertion de l'adresse est réussie, un message de succès est affiché. Ensuite, une autre requête est préparée pour supprimer le panier de l'utilisateur de la table cart. Si la suppression du panier est réussie, un message de succès est affiché, sinon un message d'erreur est affiché.

Enfin, l'identifiant du panier est stocké dans $_SESSION['cart_id'], et l'utilisateur est redirigé vers la page "succes.php" avec l'identifiant du panier en paramètre dans l'URL.

La méthode isTestCreditCardValid est une méthode privée utilisée pour vérifier si une carte de crédit fournie correspond à l'une des cartes de crédit de test valides stockées dans le tableau $testCreditCards. Elle parcourt le tableau et compare le numéro de carte, la date d'expiration et le code de vérification avec les valeurs fournies. Si une correspondance est trouvée, la méthode renvoie true, sinon elle renvoie false.

Résumer

Page d'accueil attrayante : La page d'accueil présente les produits ajoutés récemment ainsi qu'une section mettant en avant les produits les plus populaires. Cela permet aux visiteurs de découvrir rapidement les nouveautés et les produits populaires.

Page produit détaillée : Chaque produit dispose d'une page dédiée affichant son image, son nom, son prix et sa quantité disponible. Cela permet aux utilisateurs d'obtenir des informations détaillées sur chaque produit avant de prendre une décision d'achat.

Autocompletion : j'ai également mis en place une fonctionnalité d'autocomplétion et de recherche par mots-clés sur le site. Cela permet aux utilisateurs de rechercher des produits spécifiques en temps réel et d'obtenir des suggestions automatiques pendant la saisie de leur recherche.

Lorsque l'utilisateur commence à saisir des mots-clés dans la barre de recherche, JavaScript intervient en utilisant une approche asynchrone. Il envoie une requête au serveur PHP qui interroge la base de données pour récupérer les produits correspondant aux mots-clés entrés. Le serveur renvoie ensuite les résultats au format JSON.

Grâce à la réponse JSON reçue, les suggestions de produits correspondants s'affichent instant

Module de connexion asynchrone : J'ai mis en place un module de connexion qui utilise JavaScript et PHP de manière asynchrone. Le formulaire d'inscription demande l'adresse e-mail, le nom, le prénom et le mot de passe. Pour la connexion, seuls l'e-mail et le mot de passe sont requis. L'utilisation de JavaScript avec la fonction Fetch permet de récupérer les données d'un fichier PHP et d'attendre une réponse au format JSON pour effectuer la requête.

Partie administration : Dans l'espace administrateur, il est possible de gérer les produits en ajoutant, modifiant ou supprimant des produits de la base de données. De plus, il est possible d'ajouter des catégories et des sous-catégories à l'aide d'un formulaire dédié.

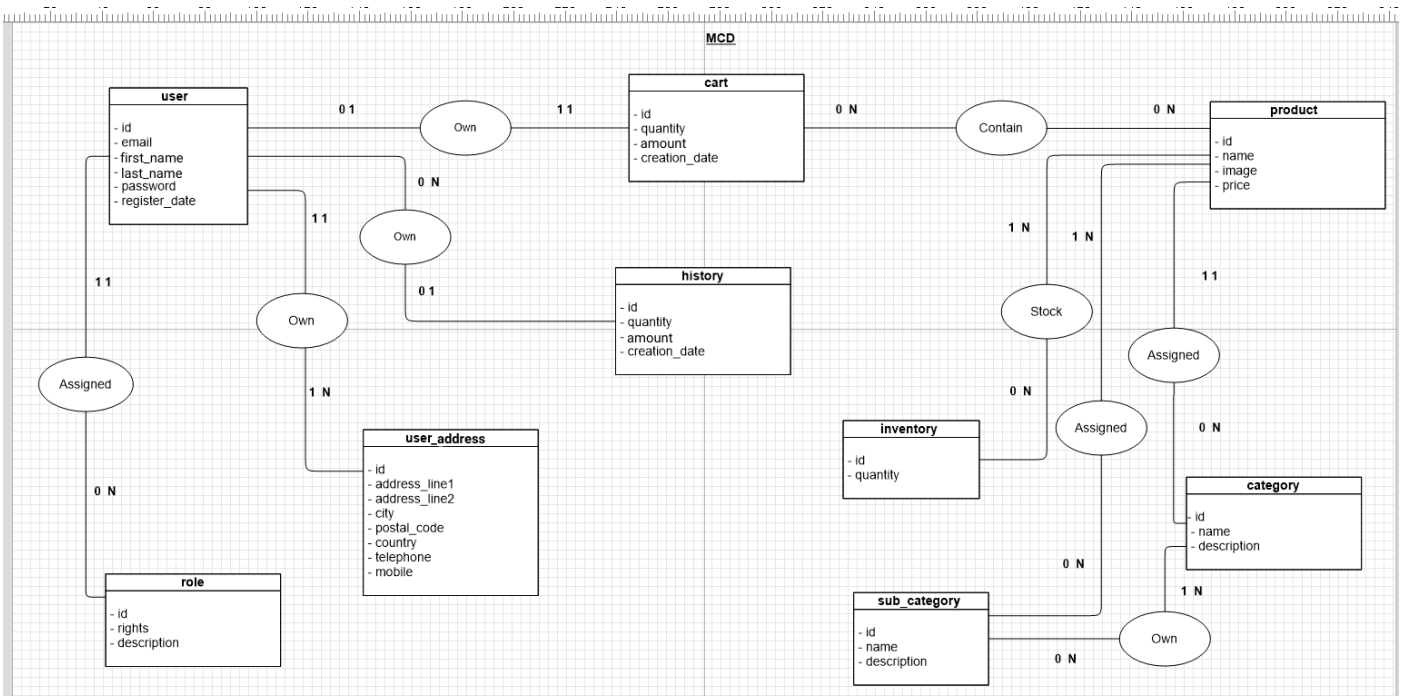
Gestion du panier : Le système de panier permet de récupérer le panier spécifique à chaque utilisateur afin que plusieurs utilisateurs ne voient pas le même panier. Le contenu du panier est affiché, incluant le nom des produits, leur prix individuel et le prix total. Les utilisateurs peuvent supprimer un produit spécifique ou vider entièrement leur panier. Un bouton de validation du panier redirige vers une page de confirmation de commande.

Confirmation de commande : Pour confirmer une commande, les utilisateurs doivent remplir un formulaire avec leurs informations personnelles, y compris les détails de leur carte bancaire. Une fois que toutes les informations sont validées, la commande est passée et l'utilisateur est redirigé vers une page de succès.

Page de succès : La page de succès récupère l'identifiant du panier stocké dans la session précédemment, puis affiche un message de confirmation de commande. Un script JavaScript stocke également le numéro de commande dans le localstorage du navigateur.

Annexes

# Modèle conceptuel de donner




# Modèle Logique de Donner