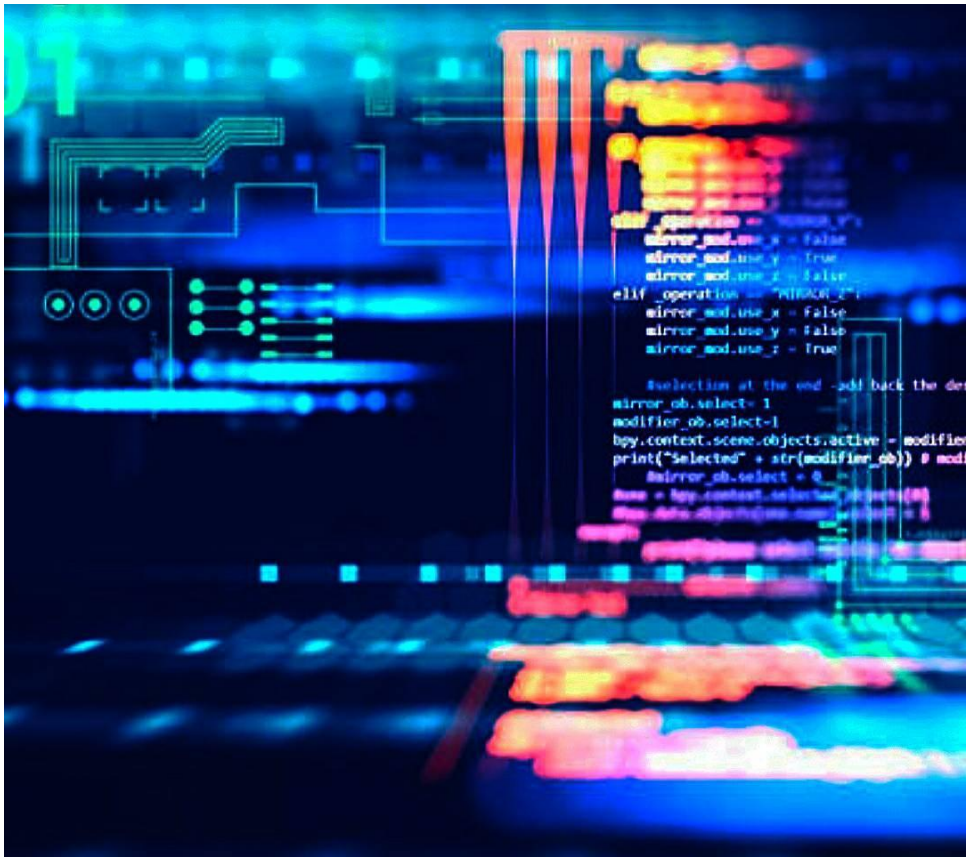
Dossier de projet pour le titre professionnel De développeur WEB et WEB mobile.

Thomas Dellacase



Tables des matières

Compétences du référentiel couvertes par le projet

Activité type 1 « Développer la partie front-end d'une application web ou web mobile

en intégrant les recommandations de sécurité »

- Maquetter une application
- Réaliser une interface utilisateur web statique et adaptable
- Développer une interface utilisateur web dynamique

Activité type 2 « Développer la partie back-end d'une application web ou web mobile

en intégrant les recommandations de sécurité »

- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile

Résumé

Dans le cadre de la formation nous avons dû crée une boutique en ligne en projet de fin de unit. Nous avons donc créé une boutique de jeux de société "Effing dice" Ce projet a comme fonctionnalité une gestion des utilisateurs et de leur commentaire, une gestion des produits et un back office. Un utilisateur doit pouvoir acheter des produits postés des commentaire sur les produits et avoir accès à son historique de commande. Le back office permet à un administrateur de modifier les droits de n'importe quel utilisateur ainsi que de supprimé et ajouter des produits, utilisateurs, et commentaires. Le panier est geré en variable de session ce qui permet de ne pas avoir de tables panier dans la base de données. Nous avons aussi dû passer la base de données de MariaDB en InnoDB pour les effacements en cascade. Ce projet fut principalement développé en PHP en POO, le CSS a été fait entièrement à la main. Il y a une peu de java script pour crée un pop-up au niveau du panier. La maquette a été réaliser avec figma.

Présentation

-Présentation personnelle

Je m'appelle Thomas Dellacase, j'habite à Marseille. Je suis actuellement une Formation dans le développement web et web mobile. Avant de me lancer dans le développement j'étais technicien d'assistance informatique.

-Présentation de la plateforme

La plateforme est une nouvelle école dédiée au numérique sur Marseille. Elle se démarque par sa pédagogie par projet et son ouverture a tout type de profils.

Spécification Fonctionnel du projet

- Un visiteur peut accéder aux sites et ses produits mais devras s'inscrire pour acheter un produit et laisser un commentaire.
- L'utilisateur peut sélectionner un produit l'ajouter à son panier l'acheter et laisser un commentaire sur le produit de son choix. Aussi consulter son panier ainsi que l'historique de commande.
- L'admin peut ajouter ou supprimer des produits, des catégories, des utilisateurs. Il peut aussi update les droits d'un utilisateur pour le passer en modérateur ou admin.

Le paradigme de programmation POO

(Programmation orientée objet)

Qu’est-ce qu'un paradigme de programmation ?

Un paradigme est une approche logique qu’un développeur vas adopter pour résoudre son problème.

Le langage principal qui a été utiliser sur ce projet est le PHP en Orienté objet. La programmation orientée objet est un paradigme de programmation qui permet de modéliser son code sous forme d’objet ayant des propriétés et des méthodes qui interagissent plutôt qu’une séquence d’instructions.

Chaque classe contient un objet qui lui est propre, cet objet lui-même a plusieurs variables et la classe contient plusieurs méthodes permettant de manipuler l’objet.

La classe user

Les variables de classe

L’objet User prend plusieurs variables qui servirons a manipuler

l'objet dans les plusieurs méthodes de la classe.

```php
<?php

class User
{
    private $id;
    public $login;
    public $password;
    public $email;
    public $db;
    private $id_droits;
```

Les Méthodes/Fonctions

Une classe peut contenir plusieurs méthodes/Fonction permettant de manipuler l’objet.

```php
$insert = $this->db->prepare("INSERT INTO user (login, password, email, id_droits ) VALUES (:login, :cryptedpass ,:email, 1)");
$insert->bindValue(":login", $login, PDO::PARAM_STR);
$insert->bindValue(":cryptedpass", $cryptedpass, PDO::PARAM_STR);
$insert->bindValue(":email", $email, PDO::PARAM_STR);
```

La plupart des fonctions intègres du SQL pour communiquer avec la Base de données.

Cette partit de la méthode sert à ajouter un utilisateur et ses variables en base de données.

Il y a plusieurs principes importants pour la programmation orientée objet dont.

-L'encapsulation qui permet de lier de la donnée au code qui la manipule tout en protégeant des actions pouvant altérer cette donnée.

-L'héritage est le principe de la POO qui permet à un objet d'acquérir les propriétés d'un autre objet Souvent on parle d'héritage de classe pour faire référence à cette notion.

Le Back

1.Utilisateurs

Développer les composants d'accès aux données

Ajouter des éléments en base de données.

J'ai dû mettre en place un module de connexion complet pour ce projet (Inscription, connexion, modification de profils). Que j'ai réalisée en POO, J'ai donc du crée une classe user pour pouvoir gérer ces fonctionnalités.

La classe prend comme variables :

- Un id.
- Un login.
- Un password
- Un email
- Un id droits
- Une variable de connexion à la base de données.

La fonction inscription.

Cette fonction va ajouter un utilisateur en base de données.

```
// ------------------------- INSCRIPTION ---------------------------------
public function register ($login, $email, $password, $confirmPW){

    $error_log = null;
    $login =  htmlspecialchars(trim($login));
    $email = htmlspecialchars(trim($email));
    $password = htmlspecialchars(trim($password));
    $confirmPW = htmlspecialchars(trim($confirmPW));
    if (!empty($login) && !empty($password) && !empty($confirmPW) && !empty($email)) {
```

On donne comme variables a cette fonction le login le mail le mots de passe et une confirmation du mot de passe. On utilise en suite htmlspecialchart pour éviter les injection SQL avec des caractère spéciaux .

```php
$logLength = strlen($login);
$passLength = strlen($password);
$confirmLength = strlen($confirmPW);
$mailLength = strlen($email);

    if (($logLength >= 2) && ($passLength >= 2) && ($confirmLength >= 2) && ($mailLength >=2)) {
        $checkLength = $this->db->prepare("SELECT login FROM user WHERE login=:login");
        $checkLength->bindValue(":login", $login, PDO::PARAM_STR);
        $checkLength->execute();
        $count = $checkLength->fetch();
```

Par la suite on vérifie la longueur des strings entrées dans les inputs.

```php
if (!$count) {

 if ( $password == $confirmPW) {



    $cryptedpass = password_hash($password, PASSWORD_BCRYPT); // CRYPTED
//    $this->Login = $login ;
    $insert = $this->db->prepare("INSERT INTO user (login, password, email, id_droits ) VALUES (:login, :cryptedpass ,:email, 1)");
    $insert->bindValue(":login", $login, PDO::PARAM_STR);
    $insert->bindValue(":cryptedpass", $cryptedpass, PDO::PARAM_STR);
    $insert->bindValue(":email", $email, PDO::PARAM_STR);
//    $insert->bindValue();
    $insert->execute();
    header('location:../pages/profil.php');
```

Ensuite si la longueur des strings est bonne et que la confirmation du mot de passe est exacte. On hash le mot de passe (diffèrent de crypter). On prépare ensuite la requête SQL qui vas ajouter l'utilisateur en base de données. A noté que l'idée droits est en dur dans la requête pour qu'un nouvelle utilisateur n'est d'office que les droits d'un utilisateur lambda. On exécute ensuite la requête. SI tout les conditions sont bonnes l'utilisateur est renvoyé vers sa page profils.

Voici la liste des condition mise en place dans cette fonction.

```php
                }
                else {
                    $error_log = "confirmation du mot de passe incorrect";
                }
            }
            else {
                $error_log = "l'identifiant existe déjà";
            }
        }
        else {
            $error_log = "2 caractères minimum doivent être insérés dans chaques champs" ;
        }
    }
    else {
        $error_log = "Champs non remplis" ;
    }
    echo $error_log;
}
```

2. Produits

Développer les composants d'accès aux données

Utiliser des éléments de la base de données.

Nous avons dû crée une barre de recherche pour ce projet.

```php
public function searchBar(){

    if(isset($_POST['submit-search'])){
        $search = $_POST['search'];
        $sql = "SELECT * FROM produits WHERE nom LIKE '%$search%' OR
        description LIKE '%$search%' OR prix LIKE '%$search%'";
        $stmt=$this->db->prepare($sql);
        $stmt->execute();
        $queryResult = $stmt->fetchAll();
        // var_dump($queryResult);

        if(COUNT($queryResult) > 0){
            foreach ($queryResult as $key) {
                echo $key['nom'];
                echo $key['description'];
                echo $key['prix'];
            }
        }
        else{
            echo "no results";
        }
    }
}
```

Pour cela la requête SQL va chercher dans la table produite les colonnes nom description et prix tous le résultat similaire à la recherche rentrée dans l'input. Ensuite si il y a des résultats concluants on stock ces résultats dans une variable $key et avec un foreach on affiche ces résultats.

```php
public function commandes($id){
    $sql = "SELECT * FROM facoprod INNER JOIN produits p ON id_produits = p.id
    INNER JOIN commandes c ON id_commande = c.id
    WHERE c.id_user = :id ORDER BY c.date DESC";
    $requete = $this->db->prepare($sql);
    $requete->bindValue(':id', $id, PDO::PARAM_INT);
    $requete->execute();
    $test = $requete->fetchAll(PDO::FETCH_ASSOC);
    $_SESSION['commandes'] = $test;
}
```

Cette fonction commandes permettant un utilisateur de consulter son historique de commande. La requête utilise un double INNER JOIN car la requête s'exécute sur la table facoprod qui est une table de liaison entre la table produit et la table commande. Ce qui permet d'éviter d'avoir une relation N a N . Donc on doit récupérer des éléments sur ces tables.

Inner join me permet de joindre des tables entres le paramètre on me permet d'indiquer les références des clefs étrangère (il est possible chainer des jointure).

3. Le back office

- Développer la partie back-end d'une application web ou web mobile

Pour ce projet nous avons dû développer un back office permettant à un administrateur de gérer le site. Ce back office a pour fonctionnalité la création et suppression d'utilisateurs, la suppression de commentaire, la modification des droits d'un utilisateur pour le passer modérateur ou admin. Mais aussi tout ce qui touche au produit comme l'ajout et suppression de produits, la création ou suppression de catégorie et consulté l'historique de toutes les commandes.

Pour accéder au back office l'utilisateur doit forcément être admin (id_droits = 1337 sur ce projet). Une vérification se fait au chargement de cette page si l'id n'est pas le bon la page ne chargera pas.

```php
if(!isset($_SESSION['id_droits']) OR $_SESSION['id_droits'] != 1337){
    echo "Only admin can accest this page";
}
else{
```

Cette page est gérer en GET [id] pour les 4 partit du back office.

```php
<nav id="navigation">
    <a href="../index.php">Acceuil</a>
    <a href="admin.php?id=categorie">Cagtegorie</a>
    <a href="admin.php?id=produits">Produits</a>
    <a href="admin.php?id=user">Utilisateurs</a>
    <a href="admin.php?id=commandes">Commandes</a>
</nav>
```

```php
<?php elseif($_GET['id'] == 'produits'): ?>

    <section>

        <h2>AJOUT DE PRODUIT</h2>

        <form id="add_product" action="" method="post" enctype="multipart/form-data">

            <div class="product">
                <input type="text" name="nom" placeholder="Product Name...">
                <input type="text" name="desc" placeholder="Description...">
                <input type="number" name="price" placeholder="Prix...">
                <input type="number" step=".01" name="stock" placeholder="Stock...">

                <label>Select Categorie</label>
                    <select name="nom_cat">
                        <option  selected disabled hidden>Select</option>
                            <?php
                                $cat = new Categorie();
                                $cat->displayChoice();
                            ?>
                    </select>
            </div>
```

Pour modifié les droits d'un utilisateur l'adimin doit avoir accès a tout les utilisateur nous avons donc mis en place un fonction qui recupere tout les utilisateurs de la base de donnée et les revoie a l'admin.

```php
public function getUser(){
    $i = 0;
    $get = $this->db->prepare("SELECT * FROM user");
    $get->execute();
    while($fetch = $get->fetch(PDO::FETCH_ASSOC)){
        $tableau[$i][] = $fetch['id'];
        $tableau[$i][] = $fetch['login'];
        $i++;
    }
    return $tableau;
}
```

Cette fonction prend tout l'id et le login de tout les utilisateurs et les retourne sous forme de tableau.

```php
public function getDisplay(){
    $display = new User();
    $tableau = $display->getUser();
    foreach($tableau as $value){
        echo '<option values"' .$value[0] . '">' . $value[1] . '</option>';
    }
}
```

La fonction qui suis recupere ce tableau et echo les 2 valeur de chaque ligne du tableau dans une option d'un formulaire select (menu deroulant)

```php
public function updateDroit($login, $id_droits){
    $update = $this->db->prepare("UPDATE user SET id_droits = :id_droits WHERE id = :login");
    $update->bindValue(":login", $login, PDO::PARAM_STR);
    $update->bindValue(":id_droits", $id_droits, PDO::PARAM_INT);
    $update->execute();

}
```

Cette fonction permet à l'admin de modifier les droits d'un utilisateur.

4. La PDO

PDO (PHP Data Objects) est une extension PHP qui définit une interface d'accès à une base de données.

Son principal avantage est qu'il permet une abstraction pour l'accès aux données. C'est-à-dire que les fonctions pour exécuter des requêtes et pour récupérer des données sont les mêmes, quelque-soit le serveur SQL utilisé (MySQL, PostgreSQL, ...). Mais aussi c'est une sur couche de sécurité qui protège des injection SQL.

Pour utiliser la PDO j'ai d'abord créé une connexion à la db dans un fichier séparé.

```php
<?php
session_start();
function connect(){

    $database = new \PDO('mysql:host=localhost; dbname=boutique; charset=utf8', 'root', '' );

    $database->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $database->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);

    return $database;
}
?>
```

Pour l'utiliser on utilise un require_once sur les autres fichiers.

```php
<?php
require_once('../function/db.php');
```

La PDO nous permet d'utiliser les statement prepare et execute.

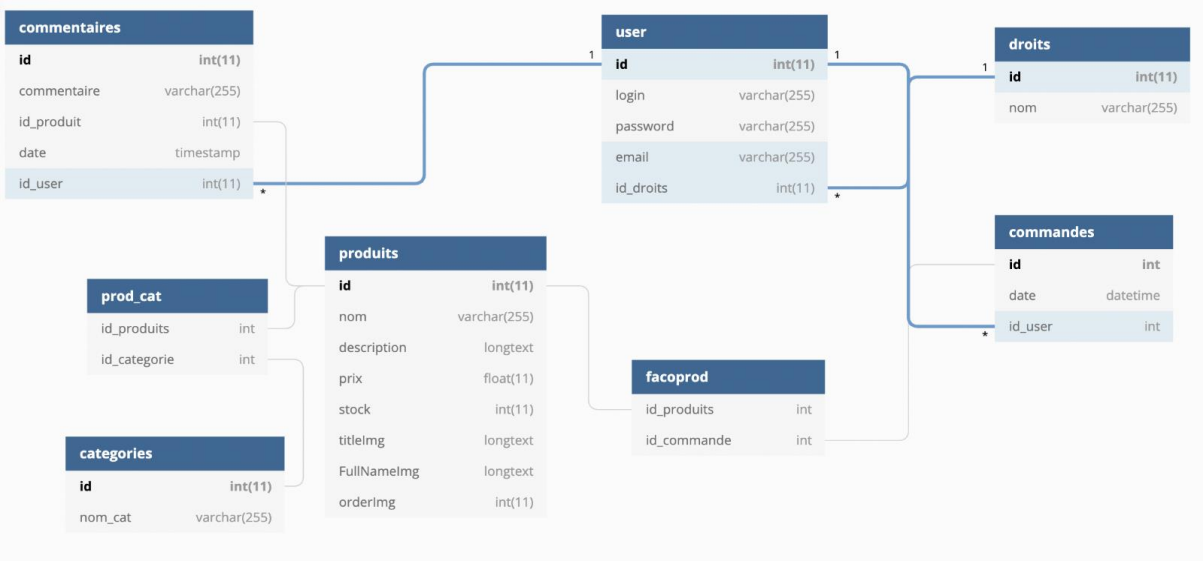
Dans un prepare une requête peut contenir zéro ou plusieurs marqueurs de paramètre nommés (:name) ou point d'interrogation (?) auxquels des valeurs réelles seront substituées lors de l'exécution de l'instruction. Les marqueurs de paramètres nommés et de point d'interrogation ne peuvent pas être utilisés dans le même modèle d'instruction.

Si on a inclus des marqueur dans le statement prepare on doit les inclure pour chaque valeur que l'on souhaite transmettre à l'instruction lorsque que l'on appelle un statement execute.

5. Conception de bas de donnée

Pour les projet conséquent la conception de la base de donnée doit se faire en amont du projet.

J'ai utilisé le model MLD pour la conception de mes bases de données.



Sur cette base de données nous avons 2table de liaison la table pod_cat qui lie la table produits et catégories et la table facoprod qui lie la table produit et la table commande.

On a besoin de ces tables pour éviter les relation N a N (plusieurs a plusieurs).

Pour les relations la table Prod cat. La table catégories auras pour clé primaire sa colonne id et comme clef étrangère id_catgorie de la table prod cat. La table produits a comme clef primaire sa colone id et comme clef étrangère id produit de la table prodcat .

Pour les relations de la tables facoprod. La table commande auras pour clé primaire sa colone id et comme clef étrangère id_commande de la table facoprod. La table produits a comme clef primaire sa colonne id et comme clef étrangère id produit de la table prodcat .

Il peut y avoir plusieurs types de relation :

Relation 1 -> 1

Je n’ai pas eu l’utilité d’une relation 1 a 1 dans ce model . Pour donner un exemple:

Une entreprise gère des équipes et des projets

Chaque équipe ne travaille que sur un seul projet

Chaque projet est géré par zéro ou une équipe

Relation 1 -> n

Une relation 1 a plusieurs

Un utilisateur peut avoir plusieurs commentaires.

Un commentaire ne peut avoir qu'un seul utilisateur

Relation n->n

Pour la relation plusieurs a plusieurs on doit utiliser une table de liaison.

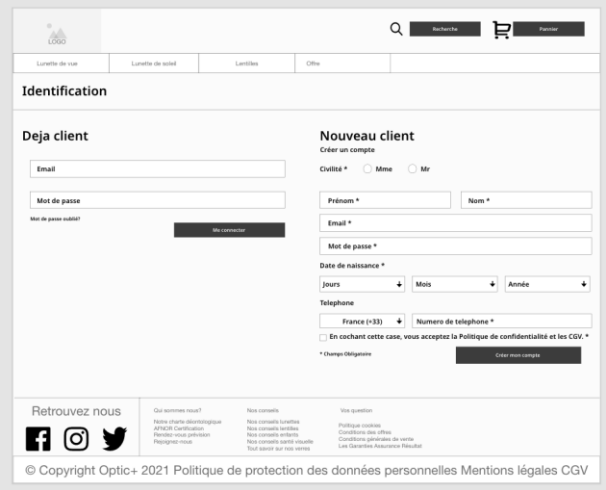
Un produit peut être dans plusieurs commande et une commande peu avoir plusieurs produits.

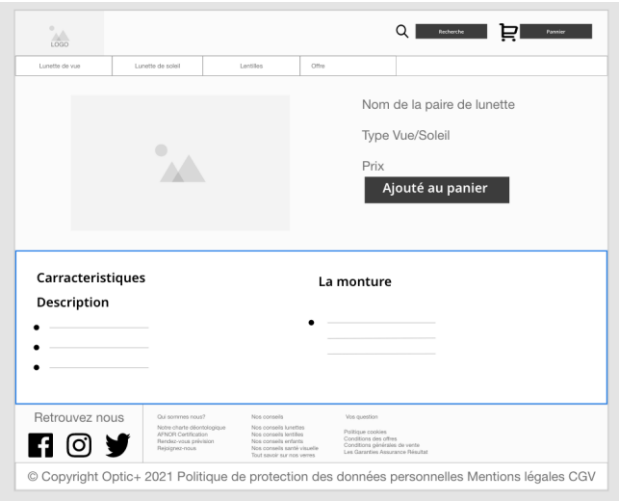
Le front

1. Le maquettage

Une fois arriver sur des projets plus volumineux on a dû faire des maquettes pour prévisualiser les projets.

La maquette servant de d'exemple avoir le rendu final du site n’est pas nécessaire. La plupart des maquettes que j’ai réalisée ont été faites en Wire frame.





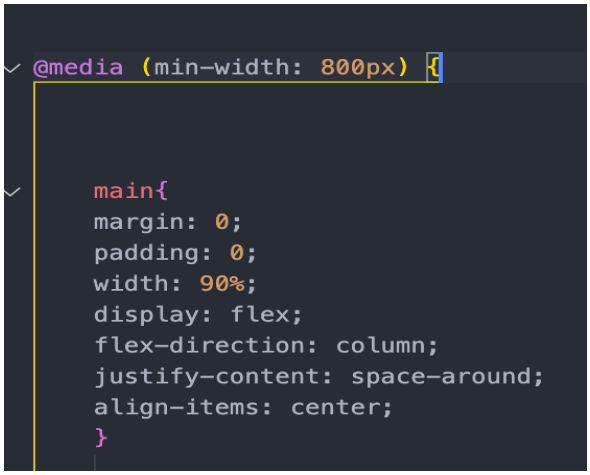
Pour la 1ère maquette que j’ai réalisée j’ai utilisé figma . Par la suite je suis passé à adobe XD qui permet plus de possibilité et une acquisition de plugin beaucoup plus simple permettant de crée des maquettes de meilleure qualité.

2. Le CSS

- Réaliser une interface utilisateur web statique et adaptable

CSS (Cascading Style Sheets ou en français "Feuilles de style en cascade") est utilisé pour styliser et mettre en page des pages Web - par exemple, pour modifier la police, la couleur, la taille et l'espacement de votre contenu, le scinder en plusieurs colonnes ou ajouter des animations et autres éléments décoratifs.

L'utilisation du CSS au-delà de styliser une page web permet avec les media query de crée un site responsif (qui s'adapte a différentes tailles d'écran) pour rester visuellement cohérent sur un grand écran aussi bien que sur un écran de téléphone.



Par exemple cette media query vas utiliser son CSS si la taille de l'écran passe en dessous de 800 px.

3 . Java Script

- Développer une interface utilisateur web dynamique.

Pour ce projet nous avons développé un pop-up en Java script quand un utilisateur ajoute un produit le pop-up lui demande soi il veux consulter son panier ou continuer ses achats.

```
(function($){

    $('.addPanier').click(function(event){
        event.preventDefault();
        $.get($(this).attr('href'),{},function(data){
            if(data.error){
                alert(data.message);
            }else{
                if(confirm(data.message + ". Voulez vous consulez votre panier ?")){
                    location.href = 'panier.php';
                }else{

                }
            }
        },'json');
        return false;
    });


})(jQuery);
```

Quand l'input addpanier est cliquer sur la page produit le script vas créer un évènement. Si il n'y a aucune erreur le pop-up apparaitra avec un lien vers le panier.

La Sécurité.

Comme sécurité nous avons utilisé 3 techniques htmlspecialchars, password hash et la PDO.

Htmlspecialchars : Certains caractères ont des significations spéciales en HTML, et doivent être remplacés par des entités HTML pour être affichés. **htmlspecialchars()** remplace tous ces caractères par leur équivalent dans la chaîne *string*. **htmlspecialchars()** est pratique pour éviter que des données fournies par les utilisateurs contiennent des balises HTML, comme pour un forum ou un chat.

```
$login =  htmlspecialchars(trim($login));
$email = htmlspecialchars(trim($email));
$password = htmlspecialchars(trim($password));
$confirmPW = htmlspecialchars(trim($confirmPW));
```

Password Hash : La fonction password_hash() crée un nouveau hachage en utilisant un algorithme de hachage fort et irréversible. Password Hash est different d'un cryptage de password du fait qu'on ne puisse pas revenir en arrière.

```
$cryptedpass = password_hash($password, PASSWORD_BCRYPT);
```

En **PHP** l'objet **PDO** est une classe qui permet de s'interfacer avec une base de données ici en **MySQL** pour communiquer avec elle via des requêtes SQL. Une bonne pratique dans sa mise en œuvre protégera efficacement vos applications contre les risques d'**injection SQL**.

Thomas