

[Titre : Développeur Web et Web mobile](#)

Dossier de projet : Création d'un e-commerce



Camille CLAPPAZ

Table des matières

Compétences du référentiel couvertes par le projet	3
Résumé	3
Spécifications fonctionnelles	4
1. Description de l'existant	4
2. Périmètre du projet	4
3. Cible adressée par le site internet	4
4. Arborescence du site	4
5. Description des fonctionnalités	4
5.1 Authentification	4
5.2 Catalogue produit et filtre	5
5.3 Fiche produit	5
5.4 Panier client	5
5.5 Fonctionnalité de recherche produit	5
5.6 Espace client	6
5.7 Panel Admin	6
5.7.1 Ajout des catégories produit	6
5.7.2 Gestion des produits	6
5.7.3 Système de promotion	6
5.8 Solution de paiement	6
Spécifications techniques	7
1. Choix techniques et environnement de travail	7
2. Réalisations	7
2.1 Charte graphique	7
2.2 Maquette	8
2.3 Conception de la base de données	9
2.4 Extraits de code significatifs	10
2.4.1 Authentification	10
2.4.2 Autocomplétion	15
2.4.3 Page détail	15
2.4.4 Panel Admin	18
2.4.5 Paiement : Intégration Stripe	23
2.5 Veille sur les vulnérabilités de sécurité	24
2.5.1 Injection SQL	24
2.5.1 HTMLSpecialChars et passwordHash	24
3. Media Queries	25
4. Recherche effectuées à partir d'un site anglophone	26
ANNEXES	26
Wireframe	27
Maquette	28
Modèle conceptuel des données	29
Modèle logique des données	29

REMERCIEMENTS

Compétences du référentiel couvertes par le projet

Le projet couvre les compétences suivantes :

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

Dans le cadre de notre formation en Développement Web et Web mobile, nous avons crée une boutique en ligne de vente de bijoux : CAMYAS. En 28 jours, nous avons crée un site dynamique et responsive. La page d'accueil est attractive avec plusieurs sections dont une qui met en avant les nouveautés. Du header, il est possible d'accéder à tous les produits par catégories ou sous-catégories, sans recharge de la page. Et chaque produit possède une page détail unique générée dynamiquement. Une barre d'auto complétion créée en javascript asynchrone permet une recherche rapide des produits. Une page panier, visible uniquement à la connexion de l'utilisateur permet de voir les articles choisis par l'utilisateur, avec une gestion de stock en temps réel, et contient une solution de paiement (STRIPE). Y sont également intégré la gestion de la TVA et la possibilité d'utiliser un code promo. Pour la partie utilisateur, un module d'inscription/connexion a été intégré au site, ainsi qu'une page permettant la gestion du profil utilisateur où sont visibles les informations personnelles et l'historique d'achat. Et enfin, un tableau de bord administrateur permet la gestion des articles, catégories et sous-catégories (Ajout/suppression/modifications). Pour la création de la partie front-end nous avons utilisé html, css, et javascript, et pour la partie back-end : php.

Spécifications fonctionnelles

1. Description de l'existant

Pour CAMYAS, l'enjeu est de créer une boutique en ligne dynamique, facile d'utilisation et avec un design contemporain pour cibler une large clientèle.

2. Périmètre du projet

Le site sera réalisé en français et ce dernier devra être accessible sur différents supports, à savoir mobile, tablette et ordinateur.

3. Cible adressée du site internet

Le site CAMYAS s'adresse à des particuliers et essentiellement des femmes qui veulent offrir ou s'offrir des bijoux.

4. Arborescence du site

L'arborescence du site se décline comme suit :

- Page d'accueil
- Page connexion
- Page inscription
- Page tous les produits ou trié par catégorie/sous-catégorie
- Page détail
- Page profil
- Page modification du profil
- Page adresse
- Page panier
- Page confirmation de commande
- Page admin

5. Description des fonctionnalités

5.1. Authentification

L'utilisateur a la possibilité de choisir entre s'inscrire ou se connecter, en fonction de s'il a déjà créé un compte.

Pour s'inscrire, il doit compléter un formulaire qui comprend les champs suivants : prénom, nom, email, mot de passe et confirmation du mot de passe.

Une fois que l'utilisateur a rempli le formulaire, il doit cliquer sur le bouton "Envoyer". À ce moment, les champs sont vérifiés afin de s'assurer qu'ils correspondent au format requis. Par exemple, le prénom et le nom ne doivent pas être vides, et les champs "mot de passe" et

"confirmation du mot de passe" doivent être identiques.

Pour se connecter, l'utilisateur doit fournir son email et son mot de passe dans les champs appropriés. Après avoir soumis le formulaire de connexion en cliquant sur "Envoyer", les informations saisies sont vérifiées pour déterminer si elles correspondent à un compte existant. Si tel est le cas, l'utilisateur est alors connecté à son compte. En revanche, s'il n'existe aucun compte correspondant aux informations fournies, un message d'erreur s'affiche pour informer l'utilisateur de cette situation.

5.2 Catalogue produit et filtre

Une page doit permettre d'afficher l'ensemble des produits disponibles. Cet affichage devra comprendre la photo du produit, le nom du produit ainsi que son prix.

Un filtre doit être implémenté afin de trier les articles en fonction de leur catégorie mais également en fonction de leur sous-catégorie.

5.3 Fiche produit

L'utilisateur devra être en mesure d'accéder à une fiche produit. Cette dernière devra comprendre:

- Une ou plusieurs photos du produit
- Le nom du produit avec sa catégorie et sous-catégorie
- Le prix
- La description du produit
- Un bouton « ajouter au panier »

5.4 Panier client

L'utilisateur devra être en mesure de sélectionner un article et de le mettre dans son panier dans le but final de passer commande sur le site.

Ce panier devra afficher les éléments suivants:

- Une photo de l'article
- Le prix de l'article TTC et HT
- La quantité demandée par le client
- Le total des articles sélectionnés
- Les informations de livraison et numéro de téléphone avec un bouton « modifier » ou « ajouter » si ce n'est pas encore renseigné.
- Un formulaire d'information de carte bancaire généré par Stripe.
- Un bouton de validation du panier débouchant sur le processus de commande.

Le client devra être en mesure d'augmenter ou diminuer la quantité demandée. Il aura aussi la possibilité de supprimer un article du panier.

5.5 Fonctionnalité de recherche produit

Le client devra être en mesure d'effectuer une recherche de produit ou de catégorie dans un champ prévu à cet effet. Afin de faciliter la recherche, un système de recherche avec autocomplétion sera mis en place dans la barre de navigation.

5.6 Espace client

L'utilisateur aura accès à un espace client dans lequel il lui sera possible de consulter et modifier ses informations. A savoir son nom, son prénom, son adresse email, son mot de passe mais également son adresse postale et sa photo de profil.

Il aura également la possibilité de consulter ses précédentes commandes, il pourra y voir le prix total de chaque commande, les articles commandés avec leurs quantités respectives, ainsi que le numéro de téléphone et l'adresse de livraison choisie pour la commande.

5.7 Panel Admin

5.7.1 Ajout des catégories produit

L'administrateur sera en mesure de gérer les catégories et les sous-catégories de produit c'est-à-dire créer des catégories et des sous-catégories et les supprimer.

5.7.2 Gestion des produits

L'administrateur aura également la capacité de créer des produits et de les supprimer.

Lors de la création du produit, ce dernier sera en mesure de:

- Donner un titre au produit
- Définir le prix du produit et même un prix promo (avec une date de péremption)
- Décrire le produit
- Uploader une image du produit

5.7.3 Système de promotion

L'administrateur devra être en mesure de mettre en place un système de promotion de type code promo. Ce code devra être renseigné par le client juste avant la validation de son panier.

5.8 Solution de paiement

La solution de paiement choisie est celle proposée par Stripe. Cette solution permettra au client de procéder au paiement de manière sécurisée par carte bancaire.

Une fois la commande passée, si Stripe valide le paiement, le panier se supprime et on peut voir la commande sur l'historique de commande passées, si le paiement est refusé la commande est effacé dans la base de données et le panier n'est pas supprimé.

Spécifications techniques

1. Choix techniques et environnement de travail

Technologies utilisées pour la partie back-end :

- Le projet est réalisé en PHP (Hypertext Preprocessor)
- Base de données SQL
- Système de Gestion de Base de Données : PhpMyAdmin

Technologies utilisées pour la partie front-end:

- Le projet est réalisé avec HTML et CSS.
- Les librairies Bootstrap et Fontawesome.
- Et enfin Javascript afin de dynamiser le site et d'améliorer l'expérience utilisateur.

L'environnement de développement est le suivant:

- Editeur de code: Visual Studio Code
- Maquettage: Figma
- Logo : Canva

Du point de vue de l'organisation :

- nous avons utilisé Trello afin de découper le projet en une multitude de tâches à réaliser et de définir leur ordre de priorité.
- Nous avons utilisé Git avec GitHub pour effectuer le versionning. Nous avons choisi de faire une branche par personne étant donné que nous avions chacune des pages différentes à faire.

2. Réalisations

2.1 Charte graphique

Les polices d'écriture sont :

- Indie Flower pour la nav
- « Helvetica Neue », Helvetica, Arial, sans-serif pour le reste des pages

Les couleurs sont les suivantes :



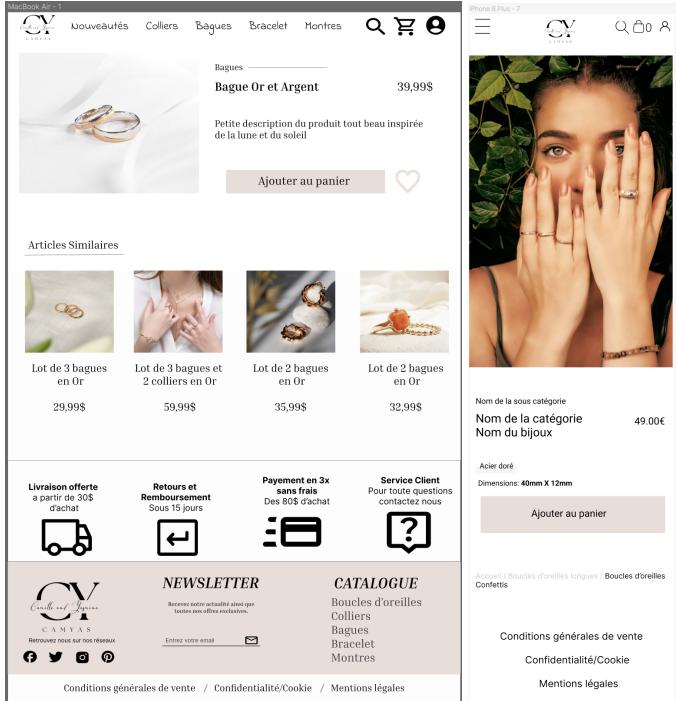
2.2 Maquette

Nous avons réalisé le wireframe et la maquette sur figma, c'est un site gratuit et assez simple d'utilisation qui permet un rendu professionnel rapidement.

Nous avons d'abord créé le wireframe qui permet de voir grossièrement la structure du site et la mise en page.



Ensuite nous avons créé la maquette qui nous montre le rendu final. On y trouve les icônes, le contenu, les photos et la couleur.

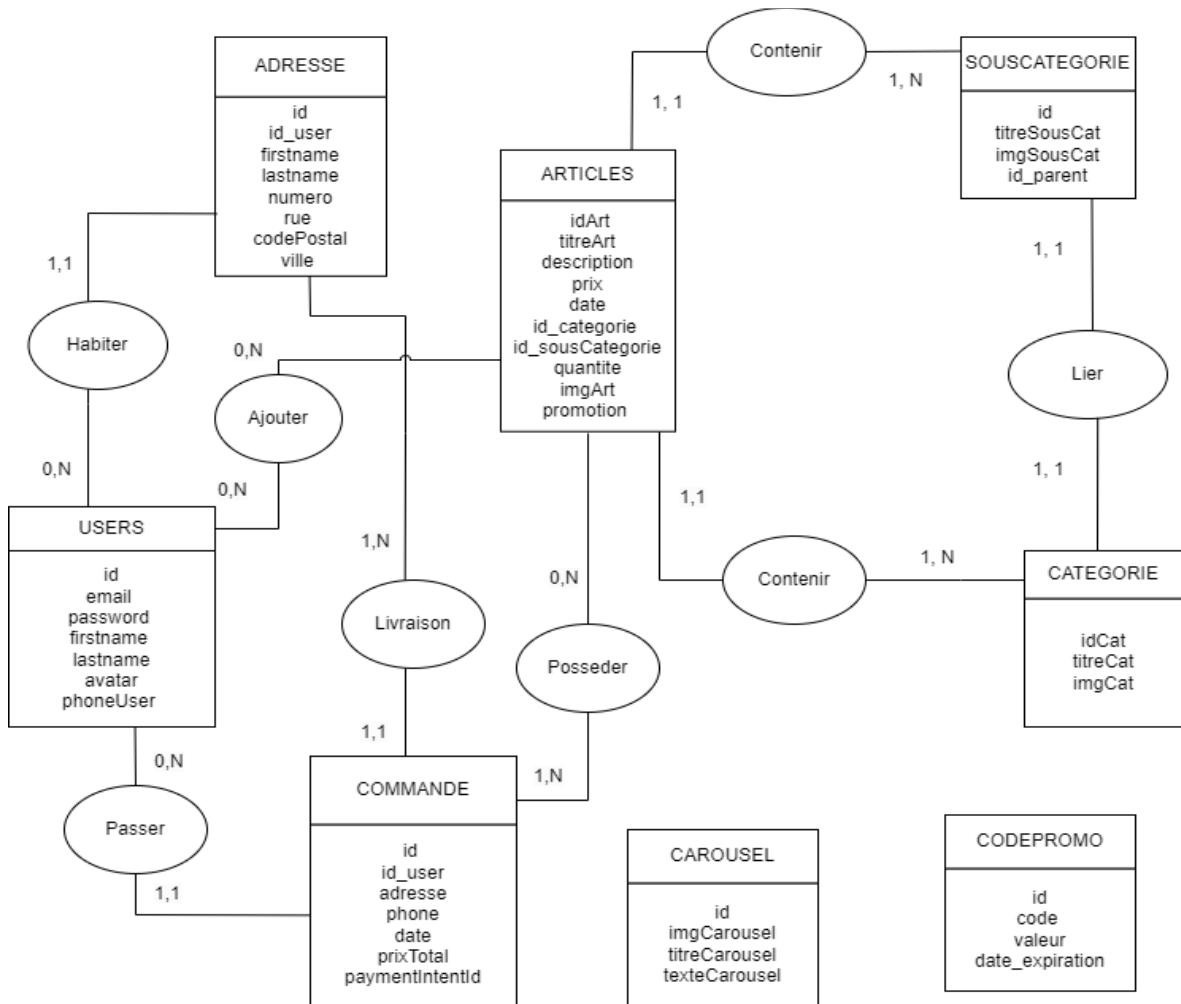


Vous pourrez retrouver l'intégralité des maquettes dans la partie « Annexes »

2.3 Conception de la base de données

Cette étape est très importante car elle permet d'avoir un visuel complet et précis sur notre base de données et à vérifier les relations entre les différentes tables (verbes et cardinalités). Vous pouvez retrouver le MLD dans les “Annexes”.

Modèle Conceptuel de données (MCD) :



Comme illustré ci dessus, on peut voir que la base de données s'articule autour de 3 tables principales.

Tout d'abord , la table **users** qui va permettre d'identifier les clients. Cette table est liée à différentes tables dont la table adresse, la table commande et également la table articles.

Il y a ensuite la table **articles** qui va être reliée à la table souscategorie, categorie, commande et users. Cela permet de lier une commande à un client spécifique.

Enfin, il y a la table **commande** qui permet de recueillir les informations relatives aux commandes passées. Cette dernière est reliée à la table users de manière à pouvoir identifier qui a passé commande. Et également liée à la table articles. Mais en sachant que plusieurs produits peuvent être dans une même commande j'ai créé une table de "liaison" ce qui me permet de stocker plusieurs produits pour une même commande. Elle apparaît dans le MLD.

2.4 Extraits de code significatifs

2.4.1 Authentification

L'inscription est réalisée de manière asynchrone en utilisant l'API fetch avec la méthode POST. Cette approche permet d'afficher les messages d'erreur dynamiquement, sans nécessiter le

rechargement de page.

Pour commencer nous créons un formulaire :

```
<form method="post" id="signup">

    <label for="firstName">Prenom</label><br>
    <input class="inputtext" type="text" id="firstName" name="firstName" /><br>

    <label for="lastName">Nom</label><br>
    <input class="inputtext" type="text" id="lastName" name="lastName" /><br>

    <label for="email">Email</label><br>
    <input class="inputtext" type="email" id="email" name="email" /><br>

    <label for="password">Mot de passe</label><br>
    <input class="inputtext" type="password" id="password" name="password" /><br>

    <label for="password2">Confirmez le mot de passe</label><br>
    <input class="inputtext" type="password" id="password2" name="password2" /><br>

    <p id="message"><?= $msg ?></p>

    <input class="inputsubmit" type="submit" name="Envoyer">
</form>
```

Nous créons un nouvel objet formData pour récupérer les valeurs des champs du formulaire. Ensuite, nous le convertissons en un objet JavaScript à l'aide de la méthode statique Object.fromEntries().

Les données sont ensuite envoyées vers la page de traitement en format JSON. Une fois que les données sont envoyées dans le corps (body) de la requête, nous les récupérons et les décodons pour pouvoir les manipuler. En utilisant ces données, nous créons une instance de la classe "User" en initialisant ses propriétés avec les champs du formulaire. Avant de procéder à l'inscription, différentes vérifications sont effectuées, notamment :

- S'assurer que les champs ne sont pas vides.
- Vérifier si l'adresse email est valide.
- Vérifier que les mots de passe correspondent.
- Vérifier que le mot de passe à bien au minimum 6 caractères, une majuscule et un chiffre
- Vérifier si l'adresse email n'est pas déjà utilisée par un autre utilisateur.

En cas d'erreur lors des vérifications, l'erreur est encodée en JSON pour être affichée côté JavaScript.

```
const formEl = document.querySelector("#signup");
const message = document.querySelector("#message");

formEl.addEventListener("submit", (event) => {
    event.preventDefault();

    const formData = new FormData(formEl);
    const data = Object.fromEntries(formData);

    fetch("./traitement_signup.php", {
        method: "POST",
        headers: {
            "Content-Type": "application/json;charset=UTF-8",
        },
        body: JSON.stringify(data),
    })
    .then((response) => [
        return response.json(),
    ])
    .then((data) => {
        console.log(data);
        message.style.color = "";
        if (data.erreur) {
            // message.style.display = (parameter) data: any
            message.innerHTML = data.erreur;
        } else {
            window.location.href = `http://localhost/boutique-en-ligne/php/connexion.php`;
            // message.style.display = "flex";
            message.style.color = "green";
            message.innerHTML = data.succes;
            formEl.reset();
        }
    })
});
```

```

$data = json_decode(file_get_contents('php://input'), true); // $data ===== $_post

function isCompatible($a,$b)
{
}

if (isset($data)) {
    $email = $data['email'];
    $firstname = $data['firstName'];
    $lastname = $data['lastName'];
    $password = $data['password'];
    $confirm_password = $data['password2'];

    $user = new User('', $email, $password, $firstname, $lastname, 'default.png', '');

    if (empty($email)) {
        $message['erreur'] = '<i class="fa-solid fa-circle-exclamation"></i>Le champ Email est vide.';
    } elseif (empty($firstname)) {
        $message['erreur'] = '<i class="fa-solid fa-circle-exclamation"></i>Le champ Firstname est vide';
    } elseif (empty($lastname)) {
        $message['erreur'] = '<i class="fa-solid fa-circle-exclamation"></i>Le champ Lastname est vide';
    } elseif (empty($password)) {
        $message['erreur'] = '<i class="fa-solid fa-circle-exclamation"></i>Le champ Password est vide';
    } elseif (empty($confirm_password)) {
        $message['erreur'] = '<i class="fa-solid fa-circle-exclamation"></i>Le champ Confirm Password est vide';
    } elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $message['erreur'] = '<i class="fa-solid fa-circle-exclamation"></i>\\'adresse mail n\'est pas valide.';
    } elseif (!isCompatible($password, $confirm_password)) {
        $message['erreur'] = '<i class="fa-solid fa-circle-exclamation"></i>Les champs password sont différents.';
    } elseif (!preg_match('/^([a-z])([a-zA-Z]{5})$/i', $password)){
        $message['erreur'] = '<i class="fa-solid fa-circle-exclamation"></i>Le mot de passe doit faire plus de 6 caractères et doit contenir au moins un chiffre, une lettre en majuscule et une en minuscule. ' . $password;
    }
}

} else {
    if ($user->loginUnique($bdd)) {
        $message['erreur'] = '<i class="fa-solid fa-circle-exclamation"></i>Cette email est déjà utilisé';
    } else {
        $user->register($bdd);
        $message['succes'] = "Données enregistrées avec succès";
    }
}
} else {
    $message['erreur'] = "Données manquantes";
}

header('Content-Type: application/json');
echo json_encode($message);
exit;

```

Enfin, nous ajoutons le nouvel utilisateur à notre base de données en appelant la méthode "register" de la classe User. Dans laquelle on utilise "password_hash()" pour hacher le mot de passe. La fonction trim() est utilisée pour supprimer les espaces en début et en fin d'une chaîne de caractères. Ensuite on insert tout dans notre table user avec la requête SQL "insert into".

```

function register($bdd)
{
    $email = trim($this->email);
    $firstname = trim($this->firstname);
    $lastname = trim($this->lastname);
    $password = password_hash(trim($this->password), PASSWORD_DEFAULT);
    $request = $bdd->prepare('INSERT INTO `users`(`email`, `password`, `firstname`, `lastname`, `avatar`) VALUES (?, ?, ?, ?, ?)');
    $request->execute([$email, $password, $firstname, $lastname, $this->avatar]);
}

```

Pour la connexion, tout se passe comme pour l'inscription sauf pour les vérifications.
On crée également un formulaire :

```

<form method="post">

    <label for="email">Email</label><br>
    <input class="inputtext" type="email" id="email" name="email" /><br>

    <label for="password">Mot de passe</label><br>
    <input class="inputtext" type="password" id="password" name="password" />

    <p id="message"><?= $msg ?></p>

    <input class="inputsubmit" type="submit" name="Envoyer" id="login">

</form>

```

```

if (isset($_POST["Envoyer"])) {
    $email = htmlspecialchars($_POST['email']);
    $prenom = '';
    $nom = '';
    $password = $_POST['password'];
    if (!empty($email) && !empty($password)) {
        $user = new User('', $email, $password, $prenom, $nom, '', '');
        $user->connect($bdd);
        if ($user->isConnected()) {
            header("Location: ../index.php");
        } else {
            $msg = "l'email et le mot de passe ne correspondent pas.";
        }
    } else {
        $msg = "Veuillez remplir tous les champs.";
    }
}

```

Ensuite, on fait des vérifications en php pour plus de sécurité, la méthode connect fait une requête SELECT à la table user et retourne toutes les infos de l'utilisateur. Si les informations correspondent aux infos rentrées dans les inputs, on connecte l'utilisateur. La fonction isConnected vérifie si la session existe ou pas. Donc si l'utilisateur a pu se connecter ou non.

```
formEl.addEventListener("submit", (event) => {
  event.preventDefault();

  const formData = new FormData(formEl);
  const data = Object.fromEntries(formData);

  fetch("traitement_connect.php", {
    method: "POST",
    headers: {
      "Content-Type": "application/json;charset=UTF-8",
    },
    body: JSON.stringify(data),
  })
    .then((response) => {
      return response.json();
    })
    .then((data) => {
      console.log(data);
      if (data.erreur) {
        message.innerHTML = data.erreur;
      } else {
        //window.location.reload();
        window.location.href = `http://localhost/boutique-en-ligne/index.php`;
        message.style.color = "green";
        message.innerHTML = data.succes;
        formEl.reset();
      }
    })
    .catch((error) => console.log(error));
}
```

2.4.2 Autocomplétion

```

const search = document.getElementById("search-bar");
const result = document.getElementById("result");
if (search) {
    //on choisit de travailler avec le keyup car le keydown a un tour de retard comparé au keyup
    search.addEventListener("keyup", () => {
        result.innerHTML = "";
        if (search.value != "") {
            fetch(`./${getPage()}autocomplete.php?search=` + search.value) //on recherche avec la
            value de ce qui a été rentré
            .then((response) => {
                return response.json();
            })
            .then((data) => {
                data.forEach((element) => {
                    let e = document.createElement("p"); // on affiche les résultats dans des balises "p"
                    // on stock l'id pour pouvoir cliquer dessus et aller vers la page détail de l'article
                    e.innerHTML =
                        `<a href= "./${getPage()}detail.php?article_id=` +
                        element.idArt +
                        `">` +
                        element.titreArt;
                    result.appendChild(e);
                });
            });
        }
    });
}

```

Dans le fichier JavaScript, nous effectuons des vérifications pour détecter si le curseur se trouve dans la barre recherche. Si tel est le cas, nous mettons en place un écouteur d'événements pour capturer les saisies de l'utilisateur lorsqu'il relâche une touche du clavier, utilisant l'événement "keyup".

Ensuite, nous utilisons la méthode `fetch`, comme expliqué précédemment, pour effectuer une requête et obtenir les résultats d'autocomplétion. La fonction `getPage()` est utilisée pour déterminer si nous sommes sur la page d'accueil ou une autre page, car les chemins d'accès diffèrent pour les autres fichiers.

Nous utilisons une requête avec la méthode GET et des pourcentages "%". Ces caractères spéciaux signifient "n'importe quel caractère", ce qui nous permet d'obtenir la valeur du paramètre GET indépendamment de ce qui précède ou suit dans l'URL. Donc lorsque l'utilisateur commence à taper « co », il pourra obtenir collier, école, ou eco. La valeur de ce qu'il tape est inclus dans les résultats obtenu.

```

if (isset($_GET['search'])) {
    $req = $bdd->prepare("SELECT `idArt`, `titreArt` FROM `articles` WHERE titreArt LIKE ?");
    $req->execute(['%' . $_GET['search'] . '%']);
    $res = $req->fetchAll(PDO::FETCH_ASSOC);
    $json = json_encode($res);
    echo $json;
}

```

Après avoir obtenu les résultats d'autocomplétion sous forme de réponse JSON, nous traitons ces données dans une fonction. À l'aide de la méthode `forEach()`, nous parcourons chaque élément des données. Pour chacun d'entre eux, nous créons une balise `<p>` et définissons son contenu pour afficher un lien vers la page de détail de l'article correspondant. Ces balises `<p>` sont ensuite ajoutées à la div "result" qui se

trouve en-dessous de la barre de recherche existante. Cela permet d'afficher les résultats d'autocomplétion à l'utilisateur, lui proposant ainsi des suggestions pertinentes lors de la saisie dans la barre de recherche.

2.4.3 Page détail

Cette page est générée en JavaScript à l'aide d'une requête fetch qui récupère les informations de l'article spécifique via une page de traitement, comme expliqué précédemment, utilisant une requête SQL.

```

if (isset($_GET['id'])) {
    $id = $_GET['id'];
    $requete = $bdd->prepare('SELECT * FROM articles
    INNER JOIN souscategorie ON souscategorie.id = articles.id_sousCategorie
    INNER JOIN categorie ON categorie.idCat=souscategorie.id_parent WHERE articles.idArt=:id');
    $requete->execute(array('id' => $id));
    $result = $requete->fetch(PDO::FETCH_ASSOC);
}

if (isset($_GET["all"])) {
    if ($_GET["all"] == 1) {
        $requete = $bdd->prepare('SELECT * FROM articles
        INNER JOIN souscategorie ON souscategorie.id = articles.id_sousCategorie
        INNER JOIN categorie ON categorie.idCat=souscategorie.id_parent');
        $requete->execute();
        $result = $requete->fetchAll(PDO::FETCH_ASSOC); // IMPORTANT fetchAll pour afficher
    }
}

```

```

if ($_SESSION["user"]["email"] == "admin@admin.fr") {
    if (password_verify("Admin1902", $_SESSION["user"]["password"])) {

```

Ensuite, en utilisant une requête SELECT, nous récupérons les informations de l'article ainsi que les informations des catégories et des sous-catégories associées en utilisant INNER JOIN pour lier les clés étrangères. Cette requête nous permet de récupérer toutes les données nécessaires en une seule fois.

Une fois que les données sont obtenues, nous procédons à la création de tous les éléments HTML nécessaires. Nous utilisons la méthode "createElement" pour créer différents éléments tels qu'une div pour contenir les informations de l'article, l'image du produit, son titre, etc. De plus, nous ajoutons également un bouton "ajouter au panier".

```

<h2>Gestions des articles</h2><br>
<div id="articles">
    <div id="formArtPromo">
        <div id="formArt">
            <form action="" method="POST">
                <h4>Création d'article:</h4>
                <br>
                <input type="text" name="titreArt" id="" placeholder="Nom de l'article">
                <textarea name="description" id="" placeholder="Description"></textarea>
                <input type="text" name="prix" id="" placeholder="Prix de l'article €">
                <input type="text" name="promotion" id="" placeholder="Promotion %>">
                <select name="categories" id="categories-select" value="categorie">
                    <option selected disabled>Catégorie</option>
                </select>
                <select name="sousCategories" id="sousCategories-select" value="sousCategorie">
                    <option selected disabled>Sous-catégorie</option>
                </select>
                <input type="text" name="quantite" id="" placeholder="Quantité">
                <input type="text" name="imgArt" id="" placeholder="URL de l'image">
                <button id="creerArt" type="submit" name="creerArt">Créer l'article</button>
            </form>
        </div><br>
    </div>

```

```
let id = window.location.href.split('=');
fetch('recherche.php?id=' + id[1]).then(response => {
    return response.json();
}).then(data => {
    let main = document.getElementById("mainDetail");
    main.setAttribute("class", "mainDetail")
    let detail = document.createElement("div");
    let img = document.createElement("img");
    let infos = document.createElement("div");
    let textInfos= document.createElement("div");
    let cat = document.createElement("h2");
    cat.className="titreDetail";
    cat.innerHTML = data.titreCat + "/" + data.titreSousCat;
    let titrePrix = document.createElement("div");
    titrePrix.setAttribute("id", "titrePrix");
    let titre = document.createElement("h1");
    titre.className="nomDetail";
    titre.textContent = data.titreArt;
    let prix = document.createElement("p");
    prix.className="prixDetail";
    prix.textContent = data.prix + "€";
    let description = document.createElement("p");
    description.className="descripDetail";
    description.textContent = data.description;
    let form = document.createElement("form");
    form.method = "POST";
    let button = document.createElement("button");
    button.setAttribute("id", "ajouterPanier");
    button.setAttribute("name", "ajouterPanier");
    button.setAttribute("value", data.idArt);
```

```

if (isset($_SESSION["user"])) {
    if (isset($_POST["ajouterPanier"])) {
        $req2 = $bdd->prepare("SELECT `quantite_art` FROM `panier` WHERE id_article = ?");
        $req2->execute([$_POST["ajouterPanier"]]);
        $res2 = $req2->fetch(PDO::FETCH_ASSOC);
        if ($req2->rowCount() > 0) {
            $req3 = $bdd->prepare("UPDATE `panier` SET `quantite_art` = ? WHERE id_article = ?");
            $req3->execute([$res2["quantite_art"] + 1, $_POST["ajouterPanier"]]);
            echo '<i class="fa-solid fa-circle-check" style="color: #0cad00;"></i> Article ajouté au panier.';
        } else {
            $req = $bdd->prepare("INSERT INTO `panier`(`id_user`, `id_article`, `quantite_art`) VALUES (?, ?, ?)");
            $req->execute([$_SESSION['user']['id'], $_POST["ajouterPanier"], 1]);
            // $_POST["ajouterPanier"] == id de l'article (jsp ce qu'il fout la)
            echo '<i class="fa-solid fa-circle-check" style="color: #0cad00;"></i> Article ajouté au panier.';
        }
    }
} else {
    echo "Veuillez vous connecter pour ajouter des articles à vos paniers";
}

```

Nous effectuons une vérification pour déterminer si l'utilisateur est connecté, car sans connexion, il ne peut pas ajouter d'articles dans son panier. Nous utilisons la valeur (`$_POST["ajouterPanier"]`) pour récupérer l'identifiant de l'article via la méthode POST. Si cette valeur existe, cela signifie que le bouton "ajouter au panier" a été cliqué.

Ensuite, nous vérifions si l'article existe déjà dans le panier. Si c'est le cas, nous augmentons simplement la quantité de l'article dans le panier. Sinon, nous insérons l'article dans le panier, puis nous affichons un message de confirmation à l'utilisateur pour lui indiquer que l'article a bien été ajouté.

2.4.4 Panel Admin

Le panel Admin n'est accessible que par l'administrateur, la vérification se fait de la manière suivante :

Cette méthode est perfectible, cela fait partie des axes d'amélioration. Nous pourrions plutôt insérer dans la table users un statut admin ou user, et accéder à la page panel admin en filtrant par rapport à ce statut.

L'admin peut créer, supprimer, modifier les articles, catégories et sous-catégories.

Voici un exemple avec les articles :

On crée un formulaire html,

On sécurise les input avec la fonction `htmlspecialchars()` qui évite les injections SQL. Puis on instancie un objet article grâce à la classe article, avec les données récupérées dans le formulaire. Et on utilise la méthode `addArticle()` pour l'ajouter en base de données. La requête utilisée est « `INSERT INTO` ».

```

if (isset($_POST["creerArt"])) {
    $titreArt = htmlspecialchars($_POST['titreArt']);
    $description = htmlspecialchars($_POST['description']);
    $promo = htmlspecialchars($_POST['promotion']);
    $prix = htmlspecialchars($_POST['prix']);
    if ($promo != 0) {
        $prix = $prix - (($promo * $prix) / 100);
    }
    $date = date('Y/m/d');
    $categories = htmlspecialchars($_POST['categories']);
    $quantite = htmlspecialchars($_POST['quantite']);
    $sousCategories = htmlspecialchars($_POST['sousCategories']);
    $imgArt = htmlspecialchars($_POST['imgArt']);
    if (($sousCategories != "Sous-catégorie") && ($categories != "Catégorie")) {
        $article = new Article($titreArt, $description, $prix, $date,
        $categories, $sousCategories, $quantite, $imgArt, $promo);
        $article->addArticle($bdd);
        header("Location:panelAdmin.php"); // Evite qu'en rechargeant la page on
        recrée la même cat.
    }
}

```

Pour afficher l'article créé, on utilise la méthode fetch pour récupérer des données de la page de traitement recherche : Le requête SQL est une requête « SELECT ».

```

fetch('./recherche.php?all=1').then(response => {
    return response.json();
}).then(data => [
    let articles = document.getElementById("articles");
    let arts = document.createElement("div");
    arts.setAttribute("id", "arts");
    articles.append(arts);

    data.filter(function (resultArt) {
        let divArt = document.createElement('div');
        divArt.className="divArt";
        let deleteArt = document.createElement('div');
        deleteArt.className="deleteArt";
        let updateArt = document.createElement('div');
        updateArt.className="updateArt";

        deleteArt.innerHTML = '<div><h4> ' + resultArt.titreArt + '</h4><p>' + resultArt.
        description + '</p><p>' + resultArt.prix + '</p><p> Quantité:' + resultArt.quantite + '</p><p>' + resultArt.titreCat + '/' + resultArt.
        titreSousCat + '</p> <button class="deleteArt" name="deleteArt" data-idArt ="' + resultArt.idArt + '" id="deleteArt" + resultArt.idArt +
        '">i class="fa-solid fa-trash-can fa-lg"</i></button></div>';
        updateArt.innerHTML = '<input id="imgArt" + resultArt.idArt + '' value="' + resultArt.imgArt + '"><input id="titreArt" + resultArt.
        idArt + '' value="' + resultArt.titreArt + '"><input id="description" + resultArt.idArt + '' value="' + resultArt.description +
        '"><input id="prixArt" + resultArt.idArt + '' value="' + resultArt.prix + '<input id="promotion" + resultArt.promotion + '' value="Promotion: ' + resultArt.promotion + '%><button class="editArt" name="editArt" data-idArt ="' + resultArt.idArt + '" id="editArt" + resultArt.idArt + ''>i class="fa-regular fa-pen-to-square fa-lg"</i></button>';

        divArt.append(deleteArt);
        divArt.append(updateArt);
        arts.append(divArt);
    })
])

```

```

if (isset($_GET["all"])) {
    if ($_GET["all"] == 1) {
        $requete = $bdd->prepare('SELECT * FROM articles
INNER JOIN souscategorie ON souscategorie.id = articles.id_sousCategorie
INNER JOIN categorie ON categorie.idCat=souscategorie.id_parent');
        $requete->execute();
        $result = $requete->fetchAll(PDO::FETCH_ASSOC); // IMPORTANT fetchAll pour afficher
plusieurs trucs, si fetch ça n'affiche que le premier resultat
    }
}

```

Une fois les données récupérées, on crée plusieurs div grâce à la méthode createElement(), voilà comment ces div vont s'imbriquer :

```

▼<div id="articles">
  ▶<div id="formArtPromo">...</div> (flex)
  ▼<div id="arts"> (flex)
    ▼<div class="divArt"> (flex)
      ▶<div class="deleteArt">...</div>
      ▶<div class="updateArt">...</div>
    </div>

```

La méthode append() permet d'ajouter un élément, ici par exemple la div « divArt » à l'intérieur de la div « arts ». `arts.append(divArt);`

Pour supprimer ou modifier un article le processus est le même donc je n'expliquerai que pour la modification.

Nous avons crée une div deleteArt et une div updateArt, qui supprime ou modifie l'article grâce à un bouton.

Dans updateArt un bouton est nommé « editArt », on l'utilise pour la suite.

```

let update = document.getElementsByName("editArt");
console.log(update)
for (let i = 0; i < update.length; i++) {

  update[i].addEventListener('click', () => {
    let id2 = update[i].getAttribute('data-idArt');
    let imgArt = document.getElementById("imgArt" + id2);
    let titreArt = document.getElementById("titreArt" + id2);
    let descriptionArt = document.getElementById("description" + id2);
    let prixArt = document.getElementById("prixArt" + id2);
    let catArt = document.getElementById("categorieArtUp-select" + id2);
    let sousCatArt = document.getElementById("sousCatArtUp-select" + id2);
    let quantiteArt = document.getElementById("quantiteArt" + id2);
    let promoArt = document.getElementById("promoArt" + id2);
    let img = imgArt.value;
    let titre = titreArt.value;
    let description = descriptionArt.value;
    let prix = prixArt.value;
    let cat = catArt.value;
    let sousCat = sousCatArt.value;
    let promo = promoArt.value;
    let quantite = quantiteArt.value;
    let date = new Date();
    if ((sousCat != "Sous-catégorie") && (cat != "Catégorie")) {
      fetch("traitementPanel.php", {
        method: "POST",
        headers: {
          'Content-type': "multipart/form-data"
        },

```

```

        body: JSON.stringify({
            "titreUpdate": titre,
            "imgUpdate": img,
            "descriptionUpdate": description,
            "prixUpdate": prix,
            "catUpdate": cat,
            "sousCatUpdate": sousCat,
            "quantite": quantite,
            "promoArt": promo,
            "date": date,
            "idUpdate": id2,
            "action": "updateArt",
        })
    }).then(response => response.json()).then(data => window.location.reload() // window.location.reload(), permet de voir en direct la suppression
        .catch(error => console.log(error));

```

A chaque bouton, nous allons attribué l'id de l'article correspondant, pour être sur de cibler correctement notre article.

Comme expliquer précédemment, on envoi nos données dans une page de traitement :

```

<hr id='hr1'>
<div><span class="petitTitre">Proceder au paiement :</span></div>
<form id="payment-form" style="margin-top: 10px;">
    <div id="link-authentication-element">
        |   <!--Stripe.js injects the Link Authentication Element-->
    </div>
    <div id="payment-element">
        |   <!--Stripe.js injects the Payment Element-->
    </div>
    <button id="submit">
        |   <div class="spinner hidden" id="spinner"></div>
        |   <span id="button-text">Acheter</span>
    </button>
    <div id="payment-message" class="hidden"></div>
</form>

```

```

async function handleSubmit(e) {
    e.preventDefault();
    setLoading(true);

    const { error } = await stripe.confirmPayment({
        elements,
        confirmParams: {
            // Make sure to change this to your payment completion page
            return_url: `http://localhost/boutique-en-ligne/php/confirmationPanier.php`,
            receipt_email: cart.user.email,
        },
    });

    if (error.type === "card_error" || error.type === "validation_error") {
        showMessage(error.message);
    } else {
        showMessage("An unexpected error occurred.");
    }

    setLoading(false);
}

```

traitementPanel.php ou l'on effectuera la requête sql pour modifier l'article.

```
if ($_POST["action"] === "updateArt") {
    $requete = $bdd->prepare('UPDATE `articles` SET `titreArt`=?, `description`=?, `prix`=?, `date`=?,
    `id_categorie`=?, `id_souscategorie`=?, `quantite`=?, `imgArt`=?, `promotion`=? WHERE idArt=?');
    $requete->execute(array($_POST["titreUpdate"], $_POST["descriptionUpdate"], $_POST["prixUpdate"], $_POST
    ["date"], $_POST["catUpdate"], $_POST["sousCatUpdate"], $_POST["quantite"], $_POST["imgUpdate"], $_POST
    ["promoArt"], $_POST["idUpdate"]));
    $message["statut"] = "OK";
    echo json_encode($message);
}
```

C'est une requête « UPDATE », pour mettre à jour l'article dans notre base de données.

Et ensuite la modification est prise en compte est affiché dans le panel admin.

Voici les codes pour la suppression :

```
let artDeleteBtn = document.getElementsByName("deleteArt");
for (let i = 0; i < artDeleteBtn.length; i++) {
    artDeleteBtn[i].addEventListener('click', () => {
        let id3 = artDeleteBtn[i].getAttribute('data-idArt');
        if (window.confirm("Voulez vous vraiment supprimer l'article")) {
            fetch("traitementPanel.php", {
                method: "POST",
                body: JSON.stringify({
                    "idDelete": id3,
                    "action": "deleteArt"
                })
            }).then(response => response.json()).then(data => window.location.reload() // window.
            location.reload(), permet de voir en direct la suppression
            .catch(error => console.log(error));
        }
    })
}
```

```
if (($_POST["action"] == "deleteArt")) {
    \Stripe\Stripe::setApiKey($stripeSecretKey);
    $stripe = new \Stripe\StripeClient($stripeSecretKey);
    $paymentIntentId = $_GET['payment_intent'];
    if (!isset($paymentIntentId) && !empty($paymentIntentId)) {
        $request = $bdd->prepare('SELECT * FROM commande WHERE paymentIntentId = ?');
        $request->execute([$paymentIntentId]);
        if ($request->rowCount() > 0) {
            header('Location: panier.php');
            $paymentIntent = $stripe->paymentIntents->retrieve($paymentIntentId);
            if ($paymentIntent->status === 'succeeded') {
                $user = json_decode($paymentIntent->metadata->user, true);
                $request = $bdd->prepare('SELECT * FROM `panier` WHERE `id_user` = ?');
                $request->execute([$user['id']]);
                $cart = $request->fetchAll(PDO::FETCH_ASSOC);
                $products = [];
                if ($cart) {
                    foreach ($cart as $productInCart) {
                        $product_id = $productInCart['id_article'];
                        $product_quantity = $productInCart['quantite_art'];
                        $productRequest = $bdd->prepare('SELECT * FROM `articles` WHERE `idArt` = ?');
                        $productRequest->execute([$product_id]);
                        $product = $productRequest->fetch(PDO::FETCH_ASSOC);
                        $product['quantite_art'] = $product_quantity;
                        $products[] = $product;
                    }
                    $prixTotal = $paymentIntent->amount / 100;
                    $request2 = $bdd->prepare('INSERT INTO `commande`(`adresse`, `id_user`, `phone`, `date`,
                    `prixTotal`, `paymentIntentId`) VALUES (?, ?, ?, ?, ?, ?)');
                    $request2->execute([$user['adresse'], $user['id'], $user['phone'], date('Y-m-d'), $prixTotal,
                    $paymentIntentId]);
                    $idcommande = $bdd->lastInsertId();
                    foreach ($products as $product) {
                        $articleIDPanier = $product['idArt'];
                        $request3 = $bdd->prepare('INSERT INTO `commandpanier`(`id_commande`, `id_article`,
                        `quantite_art`) VALUES (?, ?, ?, ?)');
                        $request3->execute([$idcommande, $articleIDPanier, $product['quantite_art']]);
                    }
                    $request4 = $bdd->prepare('DELETE FROM `panier` WHERE `id_user` = (?)');
                    $request4->execute([$user['id']]);
                } else {
                    header('Location: panier.php');
                }
            }
        }
    }
}
```

2.4.5 Paiement : Intégration Stripe

Stripe est une solution de paiement qui va permettre au client de régler sa commande en carte bancaire Visa ou MasterCard.

Son intégration est organisée autour d'une API de type REST.

Afin de réaliser l'intégration de la solution de paiement j'ai utilisé la checkout session de Stripe.

Pour que le formulaire de paiement s'affiche, il faut que le panier contienne au moins un article, et que l'utilisateur ai rempli son adresse de livraison et son numéro de téléphone.

```
<div class="lepaiement"><?php  
if (count($products) > 0 && $user->isPhoneExist($bdd) && $adresse->itExist($bdd)) {  
?>
```

Tout le code qui va suivre est généré par Stripe :

Nous avons créé une fonction asynchrone appelée handleSubmit, prenant comme paramètre l'événement "e". À l'intérieur de cette fonction, nous appelons setLoading pour afficher un état de chargement en passant le paramètre "True".

Ensuite, nous faisons appel à la fonction confirmPayment de la classe Stripe. L'utilisation de "await" permet d'attendre la fin de la requête (promesse) car nous ne pouvons l'utiliser que dans une fonction asynchrone. Nous transmettons les éléments de Stripe, la page de retour et l'e-mail pour le reçu en tant que paramètres.

Nous vérifions le type de la variable "error" pour fournir un message d'erreur à l'utilisateur. Une fois que toutes les opérations sont terminées, nous rappelons setLoading en lui passant "false" en paramètre pour masquer l'état de chargement.

Dans la page confirmationPanier.php, nous appelons la méthode setApiKey de la classe Stripe qui se trouve dans le fichier init.php (dans le dossier Stripe). Nous créons ensuite un nouvel objet de la classe StripeClient, ce qui nous permettra d'effectuer des requêtes à l'API Stripe. Ensuite, nous récupérons l'ID du paiement depuis l'URL et vérifions qu'il existe et n'est pas vide. Nous vérifions également que le paiement n'a pas déjà été validé. Si le paiement a déjà été validé, nous redirigeons l'utilisateur vers la page panier.

Si tout est en ordre, nous récupérons le paiement depuis l'API Stripe et vérifions qu'il a bien été validé en vérifiant qu'il est marqué comme "succeeded". Ensuite, nous récupérons le panier de l'utilisateur avec une requête SQL "select * from panier". Nous créons un tableau qui contiendra les produits de la commande. En parcourant le panier de l'utilisateur, nous ajoutons chaque produit au tableau des produits de la commande. Nous calculons ensuite le prix total de la commande et l'enregistrons dans la base de données avec une requête SQL "insert into". Nous récupérons l'ID de la commande que nous venons d'insérer en utilisant la fonction lastInsertId(). Enfin, nous parcourons les produits de la commande et les enregistrons dans la table de liaison "commandpanier" avant de supprimer le panier de l'utilisateur.

Si le paiement n'a pas été validé ou si l'ID du paiement n'existe pas ou est vide, nous redirigeons l'utilisateur vers la page panier.

Quand la commande est confirmée, on affiche un petit message de confirmation avec le numéro de la commande qui est l'id de la commande.

```
<h1>Confirmation de commande</h1>
<p>Merci pour votre commande !</p>
<p>Numéro de commande : <?= $idcommande ?></p>
```

2.5 Veille sur les vulnérabilités de sécurité

2.5.1 Injection SQL

Les injections SQL sont une forme de cyberattaque qui exploite les vulnérabilités dans les applications ou les sites web qui utilisent des requêtes SQL pour interagir avec les bases de données. Le principe fondamental de l'injection SQL est d'insérer du code SQL malveillant dans une requête afin d'obtenir un accès non autorisé à la base de données ou de manipuler les données à des fins malveillantes.

Les injections SQL surviennent généralement lorsque les données fournies par les utilisateurs ne sont pas correctement filtrées ou validées avant d'être utilisées dans les requêtes SQL. Les attaquants exploitent cette faiblesse en insérant du code SQL dans les champs de saisie de l'application, tels que les formulaires de recherche, les champs de connexion, ou tout autre champ où l'utilisateur peut entrer des données.

Pour se protéger contre les injections SQL, il est essentiel de mettre en œuvre des techniques de sécurité appropriées, telles que l'utilisation de requêtes préparées (paramétrées) qui permettent de séparer les données des commandes SQL (comme nous l'avons fait), la validation et le filtrage minutieux des entrées utilisateur, ainsi que la limitation des priviléges d'accès aux bases de données pour les utilisateurs du système.

2.5. HTMLSpecialChars & passwordHash():

La fonction htmlspecialchars est utilisée en PHP pour convertir les caractères spéciaux en entités HTML, empêchant ainsi les attaques par injection de code. Elle remplace les caractères tels que les guillemets doubles, les guillemets simples, les ampersands, les chevrons, etc., par leur équivalent en HTML. Cela permet d'afficher correctement les données saisies par les utilisateurs sans compromettre la sécurité du site. L'utilisation de htmlspecialchars aide à prévenir les attaques XSS (cross-site scripting) en évitant que du code malveillant soit exécuté lors de l'affichage des informations dans le navigateur.

```
$titreCat = htmlspecialchars($_POST['titreCat']);
$imgCat = htmlspecialchars($_POST['imgCat']);
```

La fonction password_hash en PHP est utilisée pour hacher les mots de passe de manière sécurisée avant de les stocker dans une base de données. Elle prend le mot de passe en clair comme entrée et génère un hachage cryptographiquement fort à usage unique. Cela protège les mots de passe des attaques de force brute et des accès non autorisés. Le résultat du hachage comprend le sel utilisé pour renforcer la sécurité du hachage. Pour vérifier les mots de passe, on utilise la fonction

`password_verify` qui compare le mot de passe en clair avec le hachage stocké dans la base de données, renvoyant vrai si les mots de passe correspondent.

```
$password = password_hash(trim($this->password), PASSWORD_DEFAULT);
```

3. Media Queries

Enfin, pour rendre notre site responsive, on a utilisé les medias queries qui permettent de définir des styles différents en fonction de la taille de l'écran sur lequel une page web est affichée. Elles permettent de rendre un site web de s'adapter à différents types d'appareils et de tailles d'écran.

```
@media (max-width: 1024px) { ...  
}  
@media (max-width: 768px) { ...  
}  
@media (max-width: 425px) { ...  
}
```

4. Recherche effectué depuis un site anglophone

Quand nous avons voulu héberger notre site sur Plesk, nous avions une erreur par rapport au format json et on ne comprenait pas pourquoi, malgré nos tests avec var-dump. Beaucoup de nos fichiers javascript ne pouvaient plus s'afficher.

Nous avons trouvé cette page :

PHP PDO: charset, set names?

Asked 12 years, 7 months ago Modified 1 year, 6 months ago Viewed 310k times



I had this previously in my normal mysql_* connection:

216

```
mysql_set_charset("utf8",$link);  
mysql_query("SET NAMES 'UTF8'");
```



Do I need it for the PDO? And where should I have it?



```
host=$host;dbname=$db", $user, $pass, array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
```

php mysql pdo

La personne montre le code qu'elle avait précédemment dans sa connexion normal mysql, et demande si elle en a besoin en PDO et où elle devrait le placer.

You'll have it in your connection string like:

```
"mysql:host=$host;dbname=$db;charset=utf8mb4"
```

HOWEVER, prior to PHP 5.3.6, the charset option was ignored. If you're running an older version of PHP, you must do it like this:

```
$dbh = new PDO("mysql:host=$host;dbname=$db", $user, $password);
$dbh->exec("set names utf8mb4");
```

On lui répond qu'elle l'a déjà dans sa ligne de connexion. Mais que si elle à une version plus vieille que PHP 5.3.6 , elle doit écrire le code du dessous. Car avant PHP 5.3.6, l'option charset était ignorée.

En ajoutant la ligne exec à notre configuration de base de donnée on a réglé le problème, même si notre version est beaucoup plus récente que 5.3.6

```
<?php
session_start();

$bdd = new PDO("mysql:host=localhost;dbname=boutique", 'root', '');
$bdd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$bdd->exec('SET NAMES utf8');

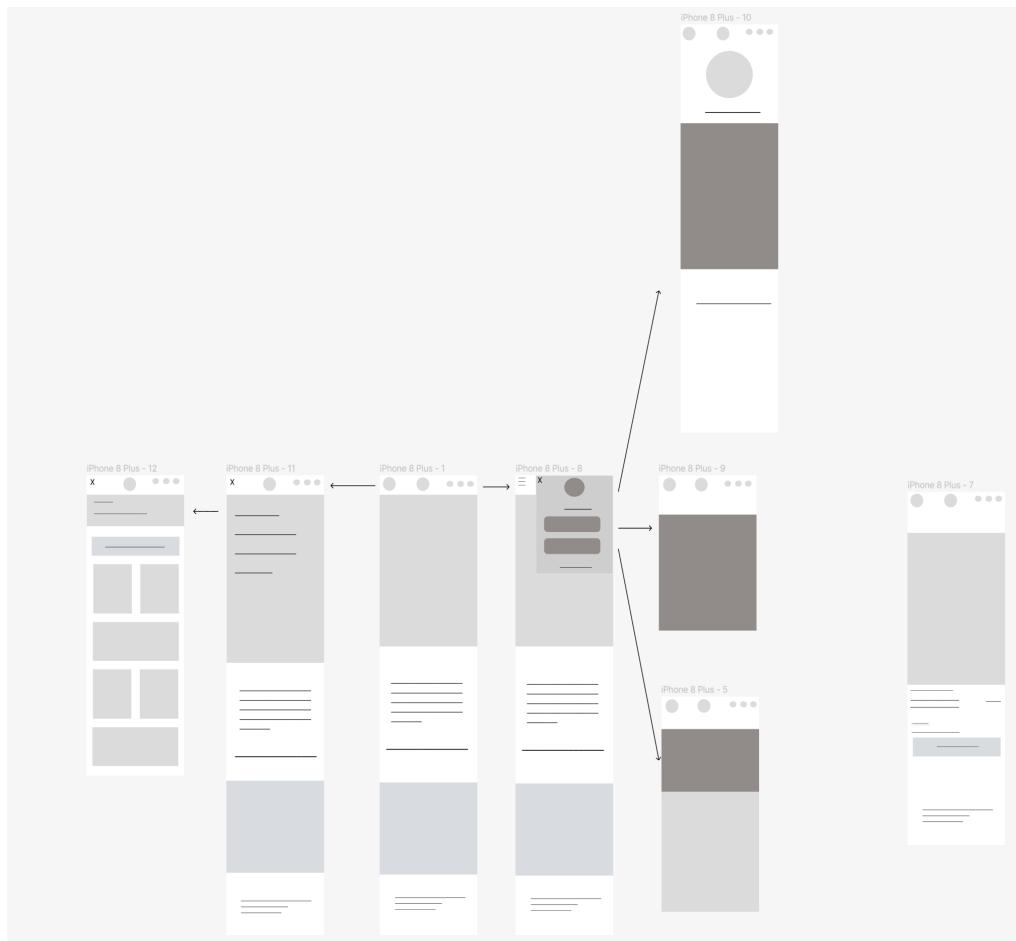
?>
```

ANNEXES

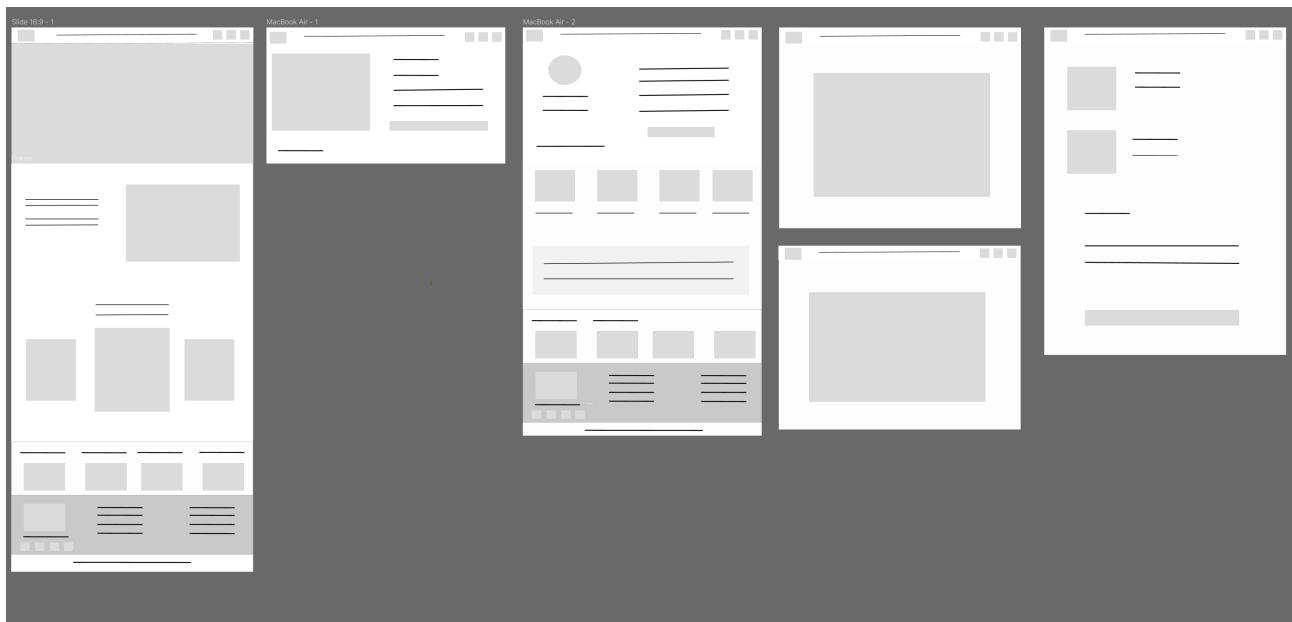
[Mon GitHub](#)
[Mon portfolio](#)

Wireframes

Wireframe téléphone :



Wireframe PC :

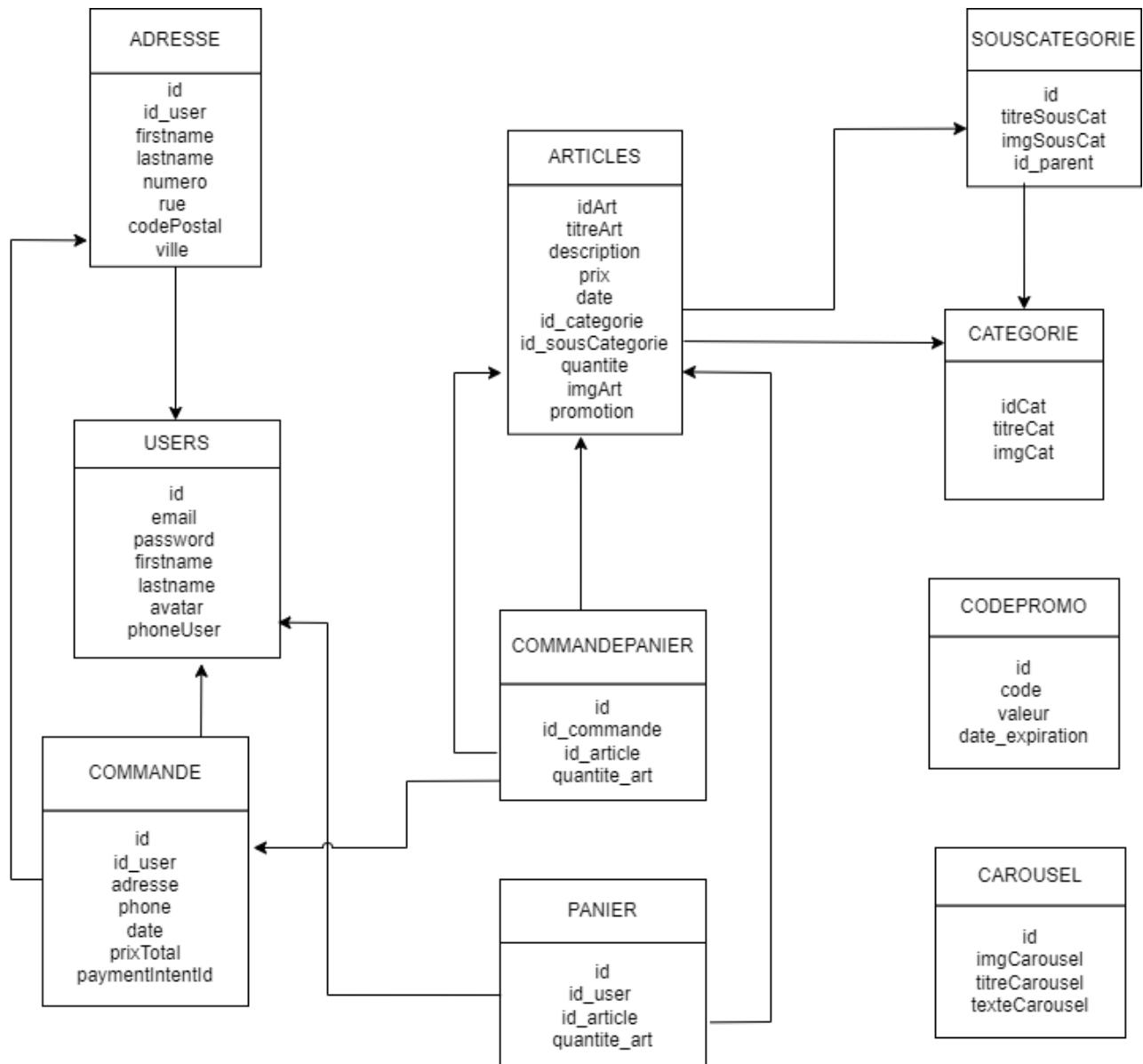


Maquette

Maquette téléphone :

Maquette PC :

Modèle logique des données



REMERCIEMENTS

Je souhaite exprimer ma profonde gratitude envers le centre de formation "La Plateforme" pour les projets pertinents auxquels j'ai participé. Ces projets ont été d'une importance capitale pour mon développement professionnel dans le domaine du développement informatique. Ils m'ont offert des opportunités précieuses pour mettre en pratique mes connaissances et relever des défis stimulants, contribuant ainsi à affiner mes compétences techniques et à développer une vision plus complète.

Je suis véritablement enthousiaste à l'idée de continuer à appliquer les enseignements acquis grâce à ces projets dans le futur. Ils ont joué un rôle décisif dans ma progression professionnelle et m'ont conféré la confiance nécessaire pour faire face à de nouvelles épreuves. Je tiens à remercier chaleureusement le centre de formation pour cette expérience enrichissante qui a grandement contribué à mon développement personnel et professionnel.