

**DOSSIER DE PROJET POUR
LE TITRE PROFESSIONNEL DE
DÉVELOPPEUR WEB & WEB MOBILE**

Créer une boutique en ligne et son identité.



Table des matières :

Compétences du référentiel couvertes par le projet	3
Résumé	3
Spécifications fonctionnelles	4
Description de l'existant	4
Périmètre du projet	4
Cible adressée	4
Arborescence du site	4
Description des fonctionnalités	5
1.Authentification classique	5
3.Catalogue produit et filtre	5
4.Fiche produit	5
5.Gestion de la quantité	5
6.Panier client	6
7.Fonctionnalité de recherche produit	6
8.Espace client	6
9.Administration	6
9.1 Ajout des catégories produit	6
9.2 Gestion des produits	7
9.3 Système de promotion	7
10.Solution de paiement	7
Spécifications techniques	8
Choix techniques et environnement de travail	8
Architecture du projet	9
Réalisations	9
1.Charte graphique	9-10
2.Maquette	10
3.Conception de la base de données	11
4.Extraits de code significatifs	12
4.1 Authentification	12
4.2 Panier client	13-14
4.3 Intégration Paypal	15-18
5. Veille sur les vulnérabilités de sécurité.	19
5.1 Exposition des données sensibles	19-20
5.2 Injection SQL	21-22
6. Recherche effectuées à partir d'un site anglophone	22-23
Annexe	24
Wireframe	24
Maquette	25-27
MCD (UML)	28

Compétences du référentiel couvertes par le projet :

Le projet couvre les compétences énoncées ci-dessous.

Pour l'activité 1, "Développer la partie front-end d'une application web et web mobile en intégrant les recommandations de sécurité":

- Maquetter une application
- Réaliser une interface utilisateur web ou mobile statique et adaptable
- Développer une interface utilisateur web dynamique
- Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

Pour l'activité 2, "Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité.":

- Créer une base de données
- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile
- Élaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce.

Résumé :

Nous mettons tout en œuvre pour développer une boutique en ligne attrayante pour La Sappe. Notre objectif est de permettre aux clients de découvrir facilement notre large gamme de vêtements éco-responsables, de naviguer aisément à travers les catégories de produits et de passer des commandes en quelques clics. En intégrant des fonctionnalités sécurisées de paiement en ligne et de gestion des commandes, nous assurons une expérience d'achat fluide et fiable.

Spécifications fonctionnelles :

1 .Description de l'existant

Pour La Sappe, l'enjeu principal réside dans l'augmentation de sa part de marché et de sa visibilité en ligne. En créant une boutique en ligne professionnelle et conviviale, nous permettons à l'entreprise d'optimiser son processus de vente, de réduire les erreurs et de faciliter les transactions pour ses clients fidèles ainsi que d'ajouter des nouveaux produits si besoin.

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 adressé

La Sappe se positionne comme un acteur engagé dans la mode en ligne éco-responsable.

Chaque achat effectué chez La Sappe est un pas vers un avenir plus durable, tout en restant à la pointe des tendances de la mode.

4 .Arborescence du site

L'arborescence du site se décline comme suit :

- Page d'accueil
- Page connexion
- Page inscription
- Page boutique
- Page détails
- Page mon compte
- Page editer profile
- Page editer mot de passe
- Page panier
- Page administration

Description des fonctionnalités :

1 .Authentification classique

Il a été convenu de pouvoir s'inscrire et se connecter de manière classique via un formulaire d'inscription/connexion celui-ci est enregistré dans notre base de données.

2 .Catalogue produits et filtre

La page boutique 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 et sa quantité disponible.

Nous avons implémenté un système de filtre pour trier par vêtements.

Un filtre doit être implémenté afin de trier les articles en fonction de leur prix croissant et décroissant le nom ou encore le couleur mais ceci arrivera par la suite.

3 .Fiche détails

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

- *Plusieurs photos du produit*
- *Le nom du produit*
- *Le prix*
- *La description du produit*
- *La quantité ou la rupture de stocks.*

4 .Gestion de la quantité

L'utilisateur devra être en mesure d'avoir la quantité en temps réel c'est pourquoi après chaque achat d'un clients les panier sont rafraîchies en base de données et au prochain rafraîchissement que le clients fera la quantité sera ajustée , non seulement dans les panier mais aussi en boutique.

5 .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
- La quantité demandée par le client
- Le total des articles sélectionnés
- Un bouton de validation du panier débouchant sur le processus de commande.

Le client devra être en mesure de supprimer un article du panier.

6 .Fonctionnalité de recherche de produit

Le client devra être en mesure d'effectuer une recherche de produit dans la barre de navigation. Afin de faciliter la recherche, un système de recherche avec autocomplétion a été mis en place.

7 .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.

Il aura également la capacité de consulter ses précédentes commandes, dans l'onglet commande passée.

9. Administration :

Le gérant du site devra avoir accès à un espace sécurisé lui permettant d'administrer le site.

9.1 Ajout des catégories produit

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

9.2 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
- Décrire le produit
- Uploader une ou plusieurs images du produit
- Établir les stocks du produit des articles.
- Définir une date de mise en ligne.

9.3 Système de promotion

L'administrateur devra être en mesure de mettre des articles en promotion.

Pour informer le client de la promotion en cours, un bandeau rouge est affiché en dessous du produit.

10. Solution de paiement :

La solution de paiement choisie est celle proposée par Paypal. Cette solution permettra au client de procéder au paiement de manière sécurisée par carte bancaire ou grâce à son compte paypal.

Spécifications techniques :

Choix techniques et environnement de travail :

Technologies utilisées pour la partie back-end :

- Le projet sera réalisé avec le langage PHP (Hypertext Preprocessor)
- Base de données SQL
- SDK Paypal (software development kit)

Technologies utilisées pour la partie front-end:

- Le projet sera réalisé avec du HTML et CSS.
- Bootstrap 5.1
- Javascript afin de dynamiser le site et d'améliorer l'expérience.

L'environnement de développement est le suivant:

- Editeur de code: Visual Studio Code
- Outil de versioning: GIT, Github.
- Maquettage: Figma(maquette) , EdrawMax(MCD)

ORGANISATIONS:

Du point de vue de l'organisation, j'ai utilisé Notion afin de dynamiser le projets avec mon collaborateur.

Cela simplifie le partage des assets (logo,image,screenshot) mais aussi les tâches à effectuer.

GITHUB:

Pour le partage du projet nous avons utilisé github avec le système de branch et de merging. Au cas où une erreur arriverait, une version de back-up serait disponible.

En ce qui concerne les branch ,nous utilisons une branch par fonctionnalité.

Architecture du projet :

Mon projet utilise l'architecture basée sur le **\$_SESSION**. Je l'ai conçu pour créer une application web où les utilisateurs peuvent se connecter et interagir avec une interface utilisateur.

Lorsqu'un utilisateur se connecte, j'utilise la variable **\$_SESSION** pour stocker des informations spécifiques à sa session, comme son ID ou son nom d'utilisateur. Ces données sont sécurisées et accessibles uniquement pendant la session en cours.

La variable **\$_SESSION** me permet également de suivre l'état de l'utilisateur tout au long de son expérience sur l'application. Je peux stocker des préférences, des paniers d'achat ou d'autres données pertinentes qui doivent être conservées d'une page à l'autre.

Grâce à cette architecture, les utilisateurs peuvent rester connectés et bénéficier d'une expérience personnalisée, tandis que je peux gérer l'authentification et suivre leur activité de manière sécurisée.

Réalisations :

1.Charte graphique :

La polices d'écriture principale: Lobster , cursive

Fonts

HEADINGS

Lobster, cursive

BODY

Lobster, cursive

Les couleurs principaux :



Le logo de l'entreprise :



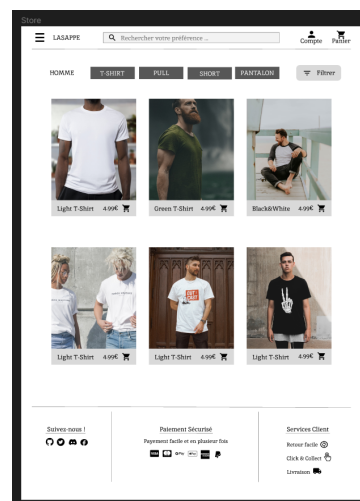
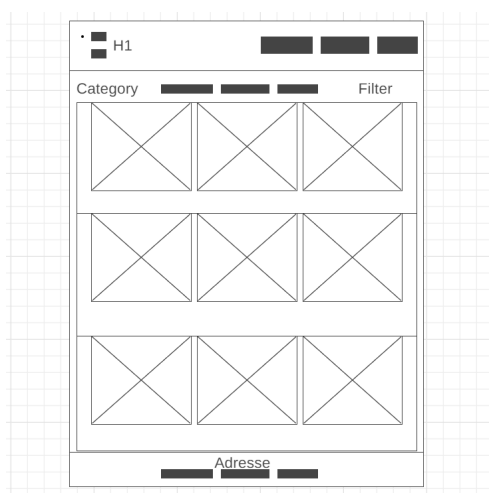
2.Maquette :

La maquette a été réalisée avec le logiciel gratuit Figma

Réalisé par nos soins elle s'inspire du wireframe préalablement fait

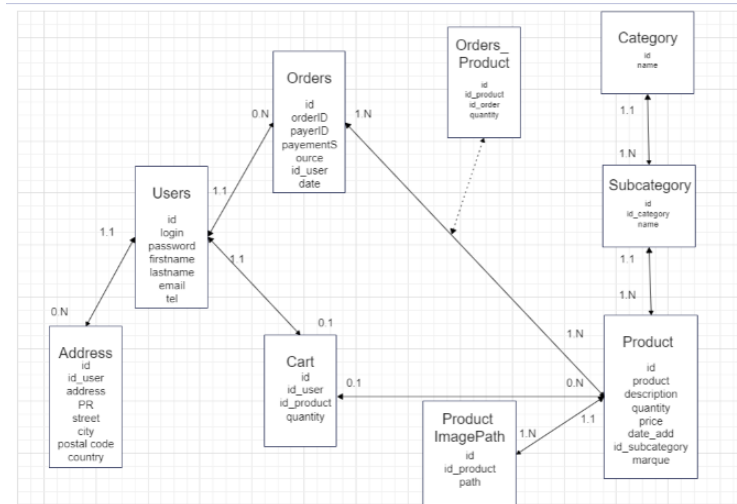
Nous nous sommes inspirés de ce site de pré apporté courant pour l'expérience utilisateurs et son confort.

Vous retrouverez les maquettes dans les annexes.



3. Conception de la base de données :

Au regard des fonctionnalités demandées par l'entreprise, j'ai développé le modèle conceptuelle de données comme ceci en UML :



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

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. Celle-ci est liées à la table address, la table panier et également la table commande.

Il y a ensuite la table **product** qui va être reliée à la table category , ImagePath. Toutes ces liaisons vont permettre de déterminer les images des produits ainsi que sa catégorie et sa sous-catégorie.

Ainsi liée à la table panier et commande cela me permettra de mettre le produits concerné dans le panier de l'user.

Enfin, il y a la table **order** qui permet de recueillir les informations relatives aux commandes passées. Cette dernière est reliée à la table users et produit. 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.

4.Extraits de codes significatifs :

4.1 Authentification :

Tout d'abord l'utilisateur doit s'inscrire s'il ne l'est pas déjà.

Cela lui permettra d'avoir un panier et sa page profil. Il est bien évidemment possible d'aller voir la boutique sans se connecter mais ne pourra pas ajouter à son panier et donc procéder au paiement.

Il n'est pas possible d'avoir deux fois le même nom d'utilisateur par choix de développement j'ai donc rajouté une petite fonctionnalité à l'inscription si le nom de l'utilisateur est déjà existant alors je lui propose des suggestions pour faciliter le confort de l'utilisateur pour cela je fait comme ceci :

Grâce à cette condition, je sais si l'utilisateur est déjà inscrit en base de données.

```
if ($user->verify_login($bdd) == false) { ?>
```

si cela est faux alors j'exécute un script javascript tout d'abord je récupère la valeur que l'utilisateur a voulu rentrer dans la variable **loginValue**

```
let loginValue = "<?= $_POST['login'] ?>";
```

Je stocke cela dans un tableau avec des suggestions de login pour faciliter l'inscription à l'utilisateur.

je crée ma fonction pour faire apparaître les suggestions dans les div

```
// créer une fonction pour générer les suggestions
function generateSuggestion(value) {
    var suggestion = document.createElement('p');
    suggestion.textContent = value;
    suggestion.addEventListener('click', function() {
        // Récupérer la valeur du paragraphe cliqué
        var valeur = suggestion.textContent;

        // Effacer la valeur actuelle du champ d'entrée
        loginInput.value = '';

        // Ajouter la valeur au champ d'entrée
        loginInput.value += valeur;
    });

    return suggestion;
}
```

```
// Générer les suggestions
var suggestions = [
    loginValue + '_user',
    loginValue + '_BigOne',
    loginValue + '_mister',
    loginValue + 'mystic',
    loginValue + '193',
    loginValue + '83',
    loginValue + '_Flex_',
];
```

Il me reste plus cas appeler cet function comme ceci dans un forEach car c'est un tableau.:

```
// Ajouter les suggestions à l'élément parent
suggestions.forEach(function(suggestion) {
    var suggestionElement = generateSuggestion(suggestion);
    suggestionContainer.appendChild(suggestionElement);
});
```

Cela a pour effet de générer les suggestions qu'il y a dans le tableau autant de fois que nécessaire et de l'append c'est à dire "ajouter" dans la div souhaiter.

4.2 Panier Client :

Pour la gestion du panier client j'ai choisi de stocker ce dernier en base de donné ce qui permet de changer de support et d'avoir ainsi toujours son panier à jour.

Il y a plusieurs méthodes comme ajouter un article, le supprimer tout en modifiant la quantité ajouté

Voici un exemple de la méthode pour ajouter un article au panier :

Tout d'abord je récupère l'article et l'user qui veut l'ajouter comme cela

```
$produit = htmlspecialchars($_POST['addcart']);
$user = $_SESSION['user']->id;

$recupUser = $bdd->prepare("SELECT * FROM cart WHERE id_user = ? AND id_product = ?");
$recupUser->execute([$user, $produit]);
```

le **\$_SESSION['user']->id** est l'id unique des users je m'en sers pour savoir à quelle utilisateurs j'ai affaires. Ensuite le **\$_POST['addcart']** est la variable GLOBAL pour savoir quelle articles a été appelé lui aussi me renvoie un id unique du produits.

Je questionne ma base de données grâce à cette requête "sélectionne moi tout dans la table panier avec l'id de cet utilisateur et l'id de ce produits la" je donne les variable \$produits, \$user et cela me renvoi l'utilisateur avec l'article associée.

Ensuite je questionne cet variables si le tableau renvoyé est vide c'est à dire que il y a pas d'article associé à cet users je peux donc en conclure qu'il n'a pas cet article dans son panier

```
if ($recupUser->rowCount() > 0) {
    $cartItem = $recupUser->fetch();
    // Selectionne les produits pour comparée les quantité entre la table cart & la table product
    $product = $bdd->prepare("SELECT * FROM product WHERE id = ?");
    $product->execute([$produit]);
    $rp = $product->fetchAll(PDO::FETCH_ASSOC);
```

Une fois la condition vérifiée, je fetch la valeurs du cart.

Je repose donc une question à ma base de données cette fois ci pour savoir le nombre de quantité de l'article toujours par rapport à l'id unique du produits.

Je fetchAll cela pour récupérer les informations sous forme de tableau.

Puis je donne comme conditions si la valeurs du panier est inférieur ou égal a la quantité du stock du produits alors ne fait rien sinon rajoute +1 à la quantité du panier

```
if ($cartItem['quantity'] >= $rp[0]['quantity']) {
    $quantity = $cartItem['quantity'];
} else {
    $quantity = $cartItem['quantity'] + 1;
}
```

J'ai plus cas faire un update sur ma base de donnée pour modifier la quantité

```
$addcart = $bdd->prepare("UPDATE cart SET quantity = ? WHERE id_user = ? AND id_product = ?");
$addcart->execute([$quantity, $user, $produit]);
```

Tout cela a pour effet de bien vérifier la quantité ajouté et de ne pas pouvoir ajouter des produits alors qu'ils sont en rupture de stocks.

Articles :

Vous avez (3) articles			Quantité	Prix(€)	
	T-shirt	nike	1	12€	
	2P/CK - Short	PULL&BELL	2	34.98€	

4.3 Intégration paypal :

Paypal est une solution de paiement qui va permettre à au client de régler sa commande en carte bancaire ou avec son compte paypal..

Son intégration est organisée autour d'un SDK (software development kits).

Afin de réaliser l'intégration de la solution de paiement j'ai utilisé le sandbox la version de "test".

Cela pourra me permettre de payer avec une fausse carte bancaire pour tester les achats.

Pour ce faire j'ai commencé par ajouter le sdk en script.

```
<script src="https://www.paypal.com/sdk/js?client-id=
```

Après client-id il faut renseigner ca clé d'api pour pouvoir exécuter le script :

```
-ATmGe5jbhPDfZtqeNcPZw_gcJU1YELNoRjhJFwkD_ixpd3yXgr-vYRmG6UrQFXonZ0BTvvcLdGd32Md_
```

Ensuite le bouton paypal apparaîtra on pourra déjà acheter mais il me reste a faire ma relation entre mon panier et paypal.

Pour cela je commence par récupérer mes produits sous forme de tableau

```
let rows = document.querySelectorAll(".liste");

let elements = []; // Déplacer la déclaration de la variable à l'extérieur de la boucle for

for (let i = 0; i < rows.length; i++) {
  let name = rows[i].querySelector("td:nth-child(2)").innerHTML;
  let marque = rows[i].querySelector("td:nth-of-type(3)").innerHTML;
  let quantity = rows[i].querySelector("td:nth-of-type(4)").innerHTML;
  let price = rows[i].querySelector("td:nth-of-type(5)").innerHTML.replace("€", "");
  console.log(price);

  let element = {
    name: name,
    description: marque,
    quantity: quantity,
    unit_amount: {
      value: parseInt(price),
      currency_code: "USD"
    }
  };

  elements.push(element);
}
```

Je débute par prendre les articles de mon panier grace a la class **liste**.

Je déclare un tableau vide **elements** ensuite je boucle afin de récupérer le prix la quantité, le nom, la marque de chaque produits de mon panier.

J'envoie ensuite ces valeurs dans **element** et je le push dans mon tableau vide ce qui aura pour effet de les envoyer dedans avec leurs prix leurs marque ext .

```
paypal.Buttons({  
  // Configurer la transaction  
  createOrder: function(data, actions) {  
    let produits = elements;  
  
    var total_amount = produits.reduce(function(total, product) {  
      return total + product.unit_amount.value * product.quantity;  
    }, 0);  
  
    return actions.order.create({  
      purchase_units: [{  
        items: produits,  
        amount: {  
          value: total_amount,  
          currency_code: "USD",  
          breakdown: {  
            item_total: {  
              value: total_amount,  
              currency_code: "USD"  
            }  
          }  
        }  
      }]  
    })  
  },  
});
```

Ensuite quand le bouton paypal est appuyé cela va créer une order je donne mon tableau **elements** a la variable **produits**. Le reste c'est paypal qui s'en occupe étant donné que c'est de leur côté il ne vaut mieux pas trop y toucher a moins de rajouter des valeurs par exemple l'utilisateur son adresse ...

Ce qui nous intéresse c'est d'effacer le panier et modifier les quantités quand une commande a été passée et valider.

Voici onApprove(data) cela me renvoie si une erreur est survenue ou si le paiement a été passé.

A partir de là je peux faire ce que je veux. Mais va falloir exécuter du code côté serveur car je suis toujours en javascript, je vais donc

```
onApprove(data) {  
  console.log(data);  
  
  fetch("traitement-order.php", {  
    method: "POST",  
    body: JSON.stringify(data),  
    headers: {  
      'Content-Type': 'multipart/form-data'  
    },  
  }).then(reponse => {  
    return reponse.json()  
  }).then(data => {  
    console.log(data);  
    console.log("valider");  
    window.location.reload();  
  }).catch(error => {  
    console.log(error);  
    window.location.reload();  
  })  
}
```


devoir utiliser le fetch en POST sur une page dédiée à cela. Ensuite j'attendrai sa réponse en asynchrone pour attendre une réponse.

```
$contentJson = file_get_contents("php://input");  
// Récupère les informations du paiement  
$_POST = json_decode($contentJson, true);  
$userId = $_SESSION['user']->id;
```

Passons en PHP sur ma page traitement-order.php. Tout d'abord nous devons récupérer les données envoyées par le POST.

Lorsqu'une information est envoyée au serveur via une requête POST, elle est enregistrée dans un fichier temporaire.

La Commande **file_get_contents('php://input')** lit les informations brutes envoyées à PHP - non traitées avant qu'elles ne soient placées dans variable GLOBALE **\$_POST**.

```
// Effectuer une requête SELECT pour récupérer les données de la table "cart"  
$req = $bdd->prepare("SELECT * FROM cart WHERE id_user = ?");  
$req->execute([$userId]);  
$cartData = $req->fetchAll(PDO::FETCH_ASSOC);  
  
$del = $bdd->prepare("DELETE FROM cart WHERE id_user = ?");  
$del->execute([$userId]);
```

Puis ayant les réponse de **\$_POST** je peux passer au requête SQL. D'abord récupérons les informations du panier pour plus tard, cela nous servira. Ensuite supprimons ce panier par rapport à l'id utilisateurs.

```
// Insérer les informations dans la table "order"  
$insertOrder = $bdd->prepare("INSERT INTO 'orders' ( 'orderId', 'payerID', 'paymentSource', 'id_user', 'date' ) VALUES ( ?, ?, ?, ?, ? )");  
$insertOrder->execute([$POST['orderId'], $POST['payerID'], $POST['paymentSource'], $userId, date("Y-m-d H:i:s")]);  
  
// Selectionne l'id order  
$selectOrder = $bdd->prepare("SELECT id FROM orders WHERE id_user = ? ORDER BY id DESC;");  
$selectOrder->execute([$userId]);  
$orderId = $selectOrder->fetch(PDO::FETCH_ASSOC);
```

Par la suite créons une commande nous pouvons mettre une date, une orderId, payerID et le paymentSource pour savoir avec quoi il a payé. Ce sont des variables qui m'ont été renvoyé par paypal quand le paiement est passé. Après mon insertion je re-sélectionne cette commande. Je fais cela pour associer les produits commandés à l'orderId. Je pourrais par la suite retrouver les produits par rapport à la commande.

```

foreach ($cartData as $row) {

    $productId = $row['id_product'];
    $productQuantity = $row['quantity'];

    // Crée une order associé au produits
    $insertOrder = $bdd->prepare("INSERT INTO `orders_product` ( `id_product`, `id_order`, `quantity`) VALUES (?, ?, ?)");
    $insertOrder->execute([$productId, $orderId['id'], $productQuantity]);

    // Calcule la quantité restante
    $subtractQuantity = $bdd->prepare("SELECT quantity - ? AS stock_disponible FROM product WHERE id = ?");
    $subtractQuantity->execute([$productQuantity, $productId]);
    $subtractQuantityRes = $subtractQuantity->fetch(PDO::FETCH_NUM);

    // Met a jour les quantité dans la table produits et la table paniers
    $updateProduct = $bdd->prepare("UPDATE `product` SET `quantity` = ? WHERE `id` = ?");
    $updateProduct->execute([$subtractQuantityRes[0], $productId]);

    $updateCart = $bdd->prepare("UPDATE `cart` SET `quantity` = ? WHERE id_product = ?");
    $updateCart->execute([$subtractQuantityRes[0], $productId]);

}

$deleteCart = $bdd->prepare("DELETE FROM cart WHERE quantity = 0");
$deleteCart->execute();

```

Dans la foulée je vais maintenant faire une boucle sur le nombre de produits dans le panier grâce à ma variable **\$cartData** préalablement fait . Déjà pour associer chaque produit à la commande passée mais surtout pour modifier les quantités.

D'abord je veux savoir le nombre de quantité restante dans la table produit, je vais faire une requête SQL pour qu'il me retourne le nombre de quantité dans la variable **\$subtractQuantityRes**.

Je n'ai plus cas modifier les quantités d'abord dans la table produit, puis les panier des utilisateurs. Ce qui aura pour effet de modifier les panier des utilisateurs si un autre compte a acheté le même produits.

J'ai ensuite rajouté une requête tout à la fin pour supprimer le panier quand la quantité arrive à zéro.

Repassons côté javascript tout d'abord j'attend la réponse en asynchrone afin de laisser le temps à la page traitement-order.php. Par la suite si la data est validé alors rafraîchi moi la page sinon si c'est une erreur alors catch la moi et affiche la moi cela me permet de débayer plus facilement.

```

}).then(reponse => {
    return reponse.json()
}).then(data => {
    console.log(data);
    console.log("valider");
    window.location.reload();
}).catch(error => {
    console.log(error);
    window.location.reload();
})

```

5. Vulnérabilités de sécurité

5.1 Exposition des données sensibles :

Lorsque vous surfez sur Internet, votre navigateur utilise le protocole HTTP (Hypertext Transfer Protocol) pour afficher les pages web, et le protocole Transmission Control Protocol/Internet Protocol (TCP/IP) pour les transmettre. Si le serveur web établit la connexion TCP avec le navigateur, une réponse avec le code statut et le fichier demandé (généralement le fichier index.html pour la page web) sera transmise.

Les données transitant en HTTP peuvent être interceptées, car elles transitent en clair.

Comment éviter d'exposer les données sensibles en transit ?

- Utilisez le HTTPS pour l'ensemble de votre site, même s'il ne contient pas de données sensibles.
- Utilisez les requêtes GET pour récupérer les informations et POST pour modifier les informations.
- Sécurisez vos sessions en ajoutant une date d'expiration, en sécurisant l'ID et en ne mettant pas cet ID dans l'URL.

Les données sensibles ne sont pas seulement en transit, elles sont aussi stockées en base de données.

Pour protéger certaines données stockées sur une application, il est possible d'utiliser des algorithmes de hachage.

L'intérêt des algorithmes de hachage est qu'ils permettent de calculer une empreinte

(ou hash) d'une chaîne de caractères, par exemple. Cette empreinte est utile pour éviter de stocker en clair le mot de passe dans la base de données.

Comment éviter d'exposer les données stockées ?

- Sécurisez votre base de données avec le chiffrement.

- Utilisez des algorithmes de hachage sécurisés tels que :

SHA-256 (Secure Hash Algorithm 256 bits) : SHA-256 fait partie de la famille des algorithmes SHA-2 et est actuellement considéré comme sûr. Il est largement utilisé, notamment pour les signatures numériques, la vérification de l'intégrité des données et les protocoles de sécurité.

SHA-3 (Secure Hash Algorithm 3) : SHA-3 est le dernier algorithme de hachage standardisé par le NIST (National Institute of Standards and Technology). Il est considéré comme sûr et offre une alternative aux algorithmes SHA-2. Il existe différentes variantes de SHA-3, telles que SHA-3-256 et SHA-3-512.

bcrypt : bcrypt est un algorithme de hachage spécifiquement conçu pour le stockage sécurisé des mots de passe. Il utilise une fonction de hachage adaptative qui ralentit considérablement le processus de hachage, ce qui le rend plus résistant aux attaques par force brute.

Argon2 : Argon2 est un algorithme de hachage de mot de passe gagnant du concours Password Hashing Competition en 2015. Il est considéré comme l'un des meilleurs choix pour le hachage sécurisé des mots de passe, offrant une résistance élevée aux attaques par force brute et par dictionnaire.

Il est **important** de choisir attentivement l'algorithme de hachage en fonction des besoins spécifiques de l'application et des recommandations en matière de sécurité. Il est également essentiel de tenir compte des pratiques actuelles en matière de sécurité et de suivre les mises à jour des recommandations de l'industrie pour garantir un niveau de sécurité adéquat.

5.2 Injection SQL :

Cette vulnérabilité permet à un attaquant d'injecter des données non maîtrisées qui seront exécutées par l'application et qui permettent d'effectuer des actions qui ne sont normalement pas autorisées.

Ce type d'attaque s'effectue généralement grâce aux champs présents dans les formulaires.

Dans le cas d'une attaque par injection SQL, au lieu de mettre un nom d'utilisateur et un mot de passe sur une page de connexion, un utilisateur malveillant entrera des données directement interprétées par le moteur SQL, ce qui lui permettra de modifier le comportement de votre application.

Comment s'en prémunir ?

1 - Utiliser des requêtes préparées ou des paramètres liés :

Utilisez des requêtes préparées ou des paramètres liés (paramètres nommés ou paramètres positionnels) fournis par votre langage de programmation ou votre framework. Cela permet de séparer clairement les instructions SQL de vos données d'entrée et d'éviter l'injection SQL.

Exemple :

```
$stmt = $bdd->prepare("SELECT * FROM users WHERE username = :username AND city = :city");  
$stmt->bindParam(':username', $name);  
$stmt->bindParam(':city', $city);  
$stmt->execute();
```

2 - Échapper les caractères spéciaux :

Si vous construisez des requêtes SQL en utilisant des chaînes de caractères, assurez-vous d'échapper correctement les caractères spéciaux (tels que les guillemets simples, les guillemets doubles et les caractères d'échappement) présents dans les données d'entrée. Cela empêche les attaquants de modifier la structure de la requête SQL.

Exemple :

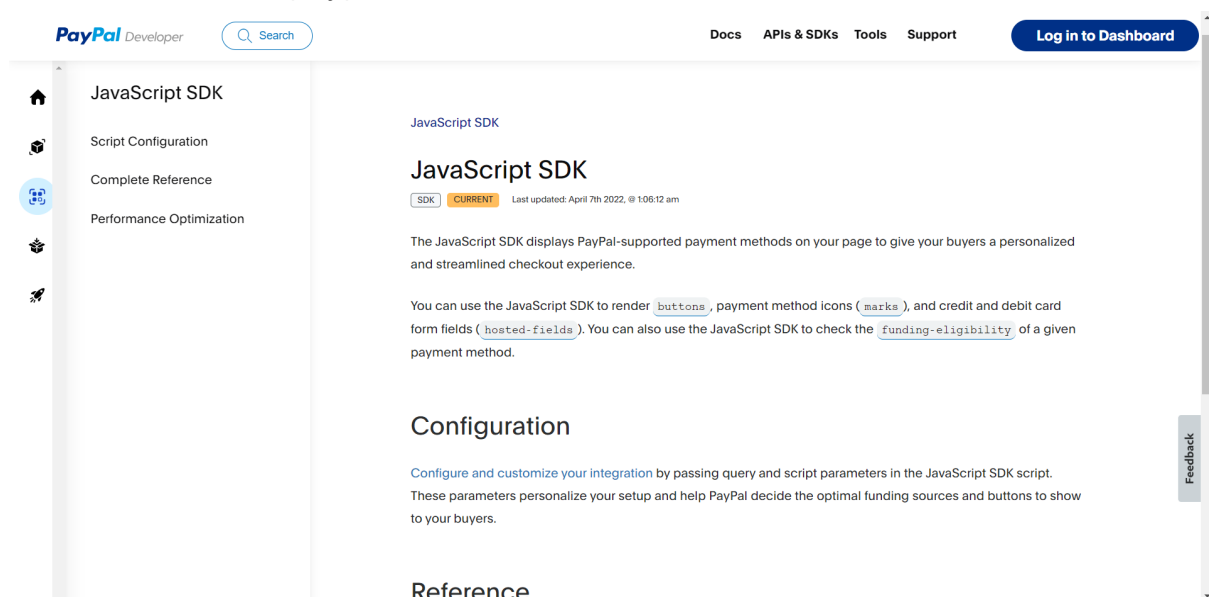
```
$name = $bdd->quote($city);
$street = $bdd->quote($zip);
$sql = "SELECT * FROM address WHERE city = $city AND zip = $zip";
$result = $bdd->query($sql);
```

3 - Valider et filtrer les entrées de l'utilisateur :






Effectuez une validation et un filtrage rigoureux sur les entrées de l'utilisateur avant de les utiliser dans des requêtes SQL. Vérifiez que les données correspondent au format attendu et rejetez toute entrée suspecte ou non valide.

6. Recherche effectuées à partir d'un site anglophone :

Ayant implémenter le système Paypal j'ai dû faire face a la longue documentation de paypal.





N'ayant pas de problème avec l'anglais j'ai pris mon temps et j'ai fait étape par étape, ce qui c'est plutôt bien passée. Très poussé et détaillé la documentation m'a beaucoup facilité les choses. C'est aussi grace a la documentation que j'ai appris que paypal m'étais à disposition des cartes de test pour pouvoir tester le paiement.

Value	Description	Button
<code>gold</code>	Recommended People around the world know us for the color gold and research confirms it. Extensive testing determined just the right shade and shape that help increase conversion. Use it on your website to leverage PayPal's recognition and preference.	 The safer, easier way to pay
<code>blue</code>	First alternative If gold doesn't work for your site, try the PayPal <code>blue</code> button. Research shows that people know it is our brand color, which provides a halo of trust and security to your experience.	 The safer, easier way to pay
<code>silver</code> <code>white</code> <code>black</code>	Second alternatives If gold or blue doesn't work for your site design or aesthetic, try the <code>silver</code> , <code>white</code> , or <code>black</code> buttons. Because these colors are less capable of drawing people's attention, we recommend these button colors as a second alternative.	 The safer, easier way to pay  The safer, easier way to pay  The safer, easier way to pay

C'est aussi grâce à cet documentation que l'on peut modifier le bouton paypal , sa couleur , son shape et plein d'autres paramètres.

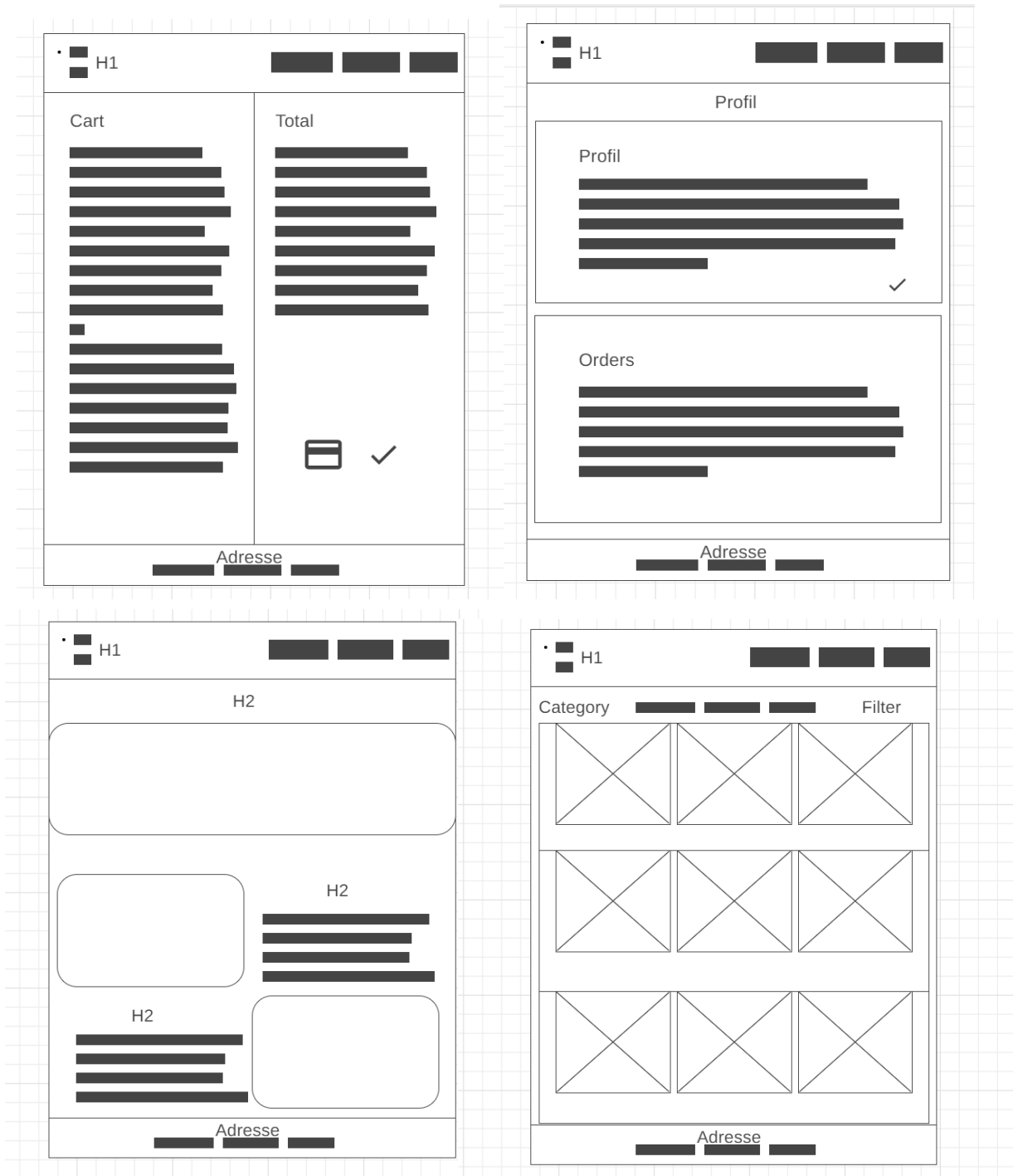
Shape

Set the `style.shape` option to one of these values:

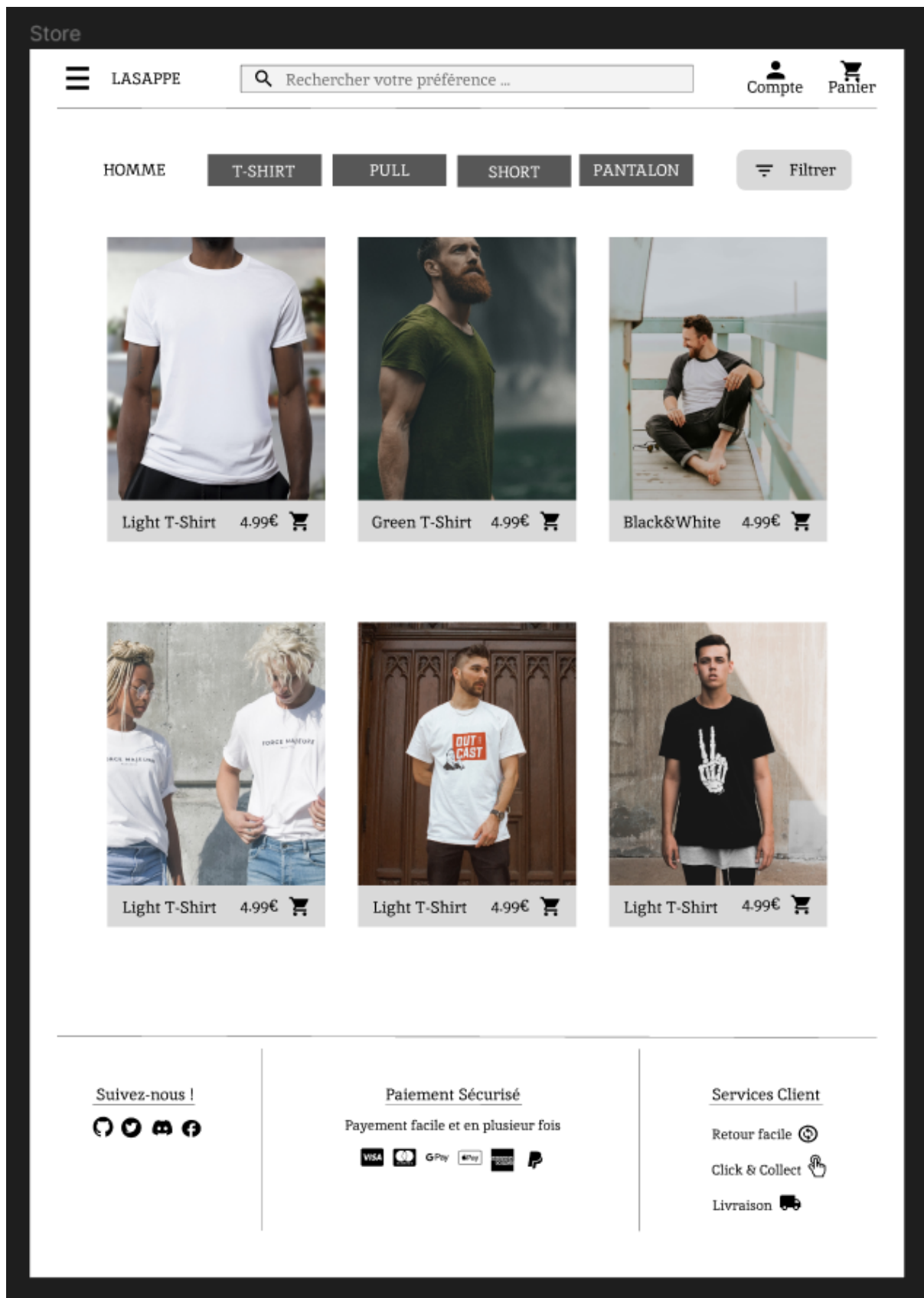
Value	Description	Button
<code>rect</code>	Recommended The default button shape.	 The safer, easier way to pay
<code>pill</code>	A secondary button shape option.	 The safer, easier way to pay

ANNEXE

Wireframe :



Maquette :



Vêtements Eco Friendly

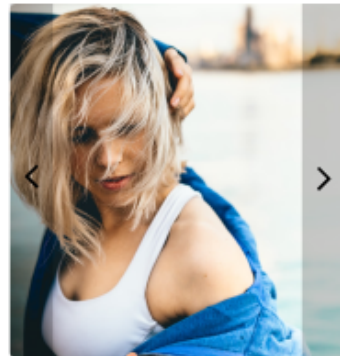


Nouvelles collections : "Eco-Friendly" et "Sportswear"

Votre marque Lasappe a créé deux nouvelles collections pour 2023. La première est la collection "Eco-Friendly", qui utilise des matériaux durables et respectueux de l'environnement, tels que le coton biologique et les fibres recyclées. La seconde est la collection "Sportswear", qui comprend des vêtements fonctionnels pour les activités sportives.

Nos meilleurs ventes :

Fabriqués à partir de coton de haute qualité, ils offrent un confort exceptionnel et un ajustement parfait. Disponibles dans une variété de couleurs classiques et de tailles, ils sont populaires auprès de tous les clients à la recherche d'un vêtement polyvalent et intemporel.



Suivez-nous !



Paiement Sécurisé

Payement facile et en plusieurs fois



Services Client

Retour facile

Click & Collect

Livraison



Panier

Articles :

Vous avez actuellement 0 articles

Récapitulatif :

Article(o)	00.00€
------------	--------

TOTAL	00.00€
--------------	--------

[Procéder au paiement](#)

Payer en 3x dès 100€ d'achat

Suivez-nous !Paiement Sécurisé

Payement facile et en plusieurs fois

Services Client

Retour facile

Click & Collect

Livraison

MCD (UML) :

