

PHP User Story Prioritaires

SAE 3.01



Nom : BOLOUUIHA Yassir, RUIZ Nicolas, IBRAHIM Marwane, LOVIN Alex
Informatique 2^{ème} Année

Sommaire :

Introduction.....	4
Objectif du livrable.....	4
User Story.....	5
1. Affichage du menu des catégories dynamique :.....	5
1.1 La page header.php.....	5
1.2 La page traitCategorie.php.....	11
2. La page Catégorie :.....	14
2.1 La page categoriPere.php :.....	14
2.2 La page categoriEnfants.php :.....	17
3. La page Description Produit :.....	21
4. La page Panier:.....	28
5. Le Processus de Commande :.....	29
6. La Fonctionnalité de Recherche :.....	30
7. La page Compte.....	34
7.1 Contexte.....	34
7.2 Modification des informations personnelles.....	35
7.2.1 Explication du code.....	35
7.2.2 Captures d'écran.....	39
7.3 Modification des informations personnelles.....	40
7.3.1 Explication du code.....	40
7.3.2 Captures d'écran.....	42
7.4 Gestion des adresses.....	43
7.4.1 Explication du code - Ajouter Adresse.....	43
7.4.2 Explication du code Modifier Adresse.....	45
7.4.3 Explication du code Supprimer Adresse.....	47
7.4.4 Captures d'écran.....	49
7.5 Gestion des cartes bancaires.....	50
7.5.1 Explication du code Ajout d'une carte.....	50
7.5.2 Explication du code Suppression d'une carte.....	53
7.5.3 Captures d'écran.....	55
7.6 Affichage des commandes.....	56

7.6.1 Explication du code.....	56
7.6.3 Captures d'écran.....	61
Conclusion.....	62

Introduction

Objectif du livrable

L'objectif de ce livrable est de présenter l'implémentation des fonctionnalités des user story prioritaires en php. Pour cela nous allons vous présenter une à une les différentes user story, regarder leur résultat sur le site, puis analyser le code correspondant aux fonctionnalités ajoutées.

User Story

1. Affichage du menu des catégories dynamique :

La fonctionnalité d'affichage des catégories dynamiques représente le fait d'afficher un menu déroulant des catégories principales c'est à dire celles qui n'ont pour catégorie parent que les 4 grosses catégories (enfant, ado, jeune-adulte, et Adulte). Le but de ce menu est qu'en passant la souris sur l'une de ces catégories principales un second menu déroulant s'affiche comportant toutes les sous-catégorie de la catégorie actuelle.

1.1 La page header.php

Avant d'aller plus loin, voici plusieurs captures d'écrans du résultat obtenu de ce menu déroulant.



Voici le menu déroulant avant que l'on clique dessus.



Et le voici après, il affiche ici les catégories principales.

X Ludorama

Rechercher...

Jouets en Bois Jeux de Société Figurines Puzzles Peluches Électronique Véhicules Extérieur Créativité Construction Jeux Vidéo
Puzzles classiques Jeux éducatifs Figurines en Bois

Et le voici enfin quand je passe la souris sur l'une des catégories principales (*Jouets en Bois*), cela affiche donc en dessous les sous catégories appartenant à la catégorie "Jouets en Bois".

Tout cela se gère avec du php, du css, et du javascript. Ainsi nous allons analyser à présent le code qui correspond à tout cela. Et nous allons commencer avec la partie php.

Ce code est le code de la page header.php puisque ce menu est présent dans le header qui lui est présent sur toutes les pages de notre site.

```
<li>
<label class="burger" for="burgerToggle">
<input type="checkbox" id="burgerToggle">
<ul class="categories">
<?php
    include ("connect.inc.php");
    $stmt = $pdo->prepare("SELECT DISTINCT C.IDCATEG, C.NOMCATEG
                           FROM CATEGORIE C
                           JOIN CATPERE CP ON C.IDCATEG = CP.IDCATEG
                           WHERE CP.IDCATEG_PERE BETWEEN 11 AND 14;");
    $stmt->execute();
    $categories = $stmt->fetchAll();
    foreach ($categories as $categorie) {
        echo "<li><a href='traitCategorie.php?idCategPere=".$categorie['IDCATEG']."'>".$categorie['NOMCATEG']."</a>";
        echo "<div class='containerSouscategories'> <ul>";
        $stmt = $pdo->prepare("SELECT DISTINCT C.IDCATEG, C.NOMCATEG
                               FROM CATEGORIE C JOIN CATPERE CP ON C.IDCATEG = CP.IDCATEG
                               WHERE CP.IDCATEG_PERE = :IDCATPERE;");
        $stmt->execute(['IDCATPERE' => $categorie['IDCATEG']]);
        $sous_categories = $stmt->fetchAll();
        foreach ($sous_categories as $sous_categorie) {
            echo "<a href='traitCategorie.php?idCateg=".$sous_categorie['IDCATEG']."'>".$idCategPere."<".$categorie['IDCATEG']."'>
                  <li>".$sous_categorie['NOMCATEG']."</li></a>";
        }
        echo "</ul> </div>";
        echo "</li>";
    }
    ?>
</ul>
<span></span>
<span></span>
<span></span>
</label>
</li>
```

Ici notre menu déroulant est à l'intérieur d'une balise "</i>" car tous les

éléments du header sont dans une liste. L'utilisation des éléments nommés "burger" ou des 3 balise "" à la fin sont utilisés uniquement pour afficher un effet esthétique à l'ouverture et à la fermeture du menu, ces éléments représentent les 3 bars sur lesquels on peut cliquer pour déployer le menu.

Ici nous allons nous intéresser à tout le reste.

La première requête SQL récupère toutes les catégories principales c'est à dire les catégories qui n'ont pour parent que les 4 grandes catégories comme dit précédemment. Les grandes catégories sont d'id 11, 12, 13, et 14 c'est pourquoi nous cherchons les catégories qui ont une catégorie père ayant un id entre ces 4 nombres.

Dans une boucle foreach, pour chacune de ces catégories je met en place un lien avec la balise "<a>" qui redirige vers la page traitCategorie.php en envoyant aussi son idCategorie.

Ensuite la deuxième requête SQL récupère les catégories qui ont pour parent la catégorie actuelle de la boucle c'est-à-dire toutes les sous-catégories de cette dernière. Puis j'affiche le résultat dans une liste qui apparaît en dessous de la catégorie parent.

Le principale problème n'était pas d'afficher tous ces résultats mais de les cacher tant que l'utilisateur ne désire pas les voir. Pour cela tout a été géré en css et javascript. Commençons d'abord par le plus court, c'est-à-dire le javascript.

```
<script>
    document.addEventListener('DOMContentLoaded', () => {
        const burgerToggle = document.getElementById('burgerToggle');
        const categoriesMenu = document.querySelector('.categories');
        // Fermer le menu déroulant si on clique ailleurs
        document.addEventListener('click', (event) => {
            if (!event.target.closest('.burger')) {
                burgerToggle.checked = false; // Décocher la checkbox pour fermer le menu
            }
        });
    });
</script>
```

```
// Empêcher le clic sur le menu burger de se propager
burgerToggle.addEventListener('click', (event) => {
    event.stopPropagation();
});
});
</script>
```

On commence tout d'abord par s'assurer que le code ne soit exécuté uniquement quand l'utilisateur le souhaite avec la première ligne.

On récupère ensuite les éléments qui nous intéressent, c'est-à-dire la checkbox "burgerToggle" et la liste "categories".

Une fois cela fait, on fait en sorte que si l'utilisateur clique ailleurs le menu se ferme automatiquement.

Et pour finir on empêche la propagation, un clic sur le "burger" (ou des zones ciblées) est isolé pour éviter une fermeture involontaire du menu.

Passons à présent à la partie css, cette dernière est immense et comporte des centaines de lignes de code c'est pourquoi je vais uniquement sélectionner les plus importantes. Nous ne nous attarderons pas sur le css du "burger" car ce n'est qu'une partie esthétique qui ne nous importe peu.

```
/* Par défaut, le menu est caché */
.categories {
    overflow-y: auto;
    position: relative;
    top: 5em;
    left: -0.5em;
    background-color: var(--clair-purple);
    list-style-type: none;
    margin: 0;
    padding: 0;
    width: 100vw;
    max-height: 20em;
    z-index: 1000;
    display: none; /* Cacher le menu par défaut */
    justify-content: flex-start; /* Alignement des éléments en ligne */
```

```
}
```



```
.burger input:checked ~ .categories {  
    display: flex; /* Afficher le menu lorsque le checkbox est coché */  
}
```

Ici on cache les catégories par défauts en utilisant “display : none” puis on les affiche uniquement quand la checkbox “burger” est cochée.

```
.categories li:hover {  
    background-color: var(--medium-purple);  
    transition: background 0.3s;  
    height: 5em;  
}
```

Quand l'une des catégories est survolée par la souris (:hover) la taille des éléments augmentent afin de laisser apparaître les sous catégories qui était caché derrière la div.

```
.containerSouscategories {  
    display: none; /* Cacher les sous-catégories par défaut */  
    position: absolute; /* Positionner les sous-catégories sous la catégorie parent */  
    background-color: var(--clair-purple);  
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
    z-index: 1000;  
    left: 0;  
}  
  
/* Afficher les sous-catégories lors du survol */  
.categories li:hover .containerSouscategories {  
    display: block; /* Afficher les sous-catégories au survol */  
}
```

Pour finir, les sous-catégories sont cachés par défauts et ne s'affiche que quand l'utilisateur survole l'une des catégories principales, cela permet d'éviter que toutes les sous-catégories s'affiche en même temps.

1.2 La page traitCategorie.php

Cette page permet de gérer la redirection vers les catégories en fonction de la où a cliqué l'utilisateur. Comme vous l'avez vu le principe est simple, si l'utilisateur clique sur une catégorie parent principale, il est redirigé avec uniquement la valeur de l'id de la catégorie, dans le cas où il clique sur une sous-catégorie il est redirigé avec les valeurs de l'id de la catégorie et de l'id de la catégorie parent.

```
<?php
session_start();
$idCateg = isset($_GET['idCateg']) ? $_GET['idCateg'] : null;
$idCategPere = isset($_GET['idCategPere']) ? $_GET['idCategPere'] : null;

if ($idCategPere !== null && $idCateg == null) {    // Si on a uniquement une catégorie parente
    header("Location: categoriePere.php?idCategorie=".$idCategPere);
    exit();
} else if ($idCateg !== null && $idCategPere !== null) {    // Si on a une catégorie enfant et une catégorie parente
    header("Location: categorieEnfants.php?idCategorie=".$idCateg. "&idCategoriePere=".$idCategPere);
    exit();
} else {
    echo "Aucune catégorie sélectionnée";
    header("Location: index.php");
    exit();
}
?>
```

La page traitCategorie.php vérifie donc tout simplement 3 cas :

- Si on a uniquement une catégorie Parente, c'est-à-dire si on a cliqué sur une catégorie principale alors on est redirigé vers la page categoriePere.php et on envoie aussi l'id de la catégorie en question.
- Si on a une catégorie enfant et une catégorie parent alors on redirige vers la page categorieEnfants.php et on y envoie les deux id correspondant.
- Le dernier cas ne peut arriver que en cas d'erreur si aucune catégorie n'est sélectionnée alors on redirige sur la page d'accueil.

2. La page Catégorie :

La page de catégorie est la page qui affiche tous les produits correspondant à une seule catégorie. Pour cela, j'ai créé deux pages différentes : la page des catégories pères, et la page des catégories enfants. Ce choix se traduit par la structuration de notre base de données, en effet, notre base de données comporte la table "CATEGORIE" qui est reliée à la table "CATPERE".

Ainsi, j'ai choisi de créer deux pages différentes pour gérer l'affichage des produits appartenant à une catégorie.

2.1 La page categoriPere.php :

Cette page permet d'afficher toutes les sous-catégories qui appartiennent à la catégorie demandé et d'afficher pour chacune tous ses produits.

Voici un exemple si je clique sur la catégorie Parent "Jouets en Bois" qui à 3 sous-catégories qui sont les suivantes : puzzle classique, jeux éducatifs, et figurines en bois.

Ludorama

Rechercher...

Puzzles classiques

Filtres

[Age](#)

[Prix](#)



Puzzle en bois
12.99€



Casse-tête en bois
14.99€



Labyrinthe en bois
16.99€



Puzzle en bois animaux
14.99€



Puzzle en bois chiffres
12.99€



Puzzle en bois lettres
13.99€



Puzzle en bois formes
11.99€

Jeux éducatifs

Filtres

[Age](#)

[Prix](#)



Train en bois
29.99€



Cheval à bascule
49.99€



Boîte à formes
14.99€



Hape
19.99€



Abacus en bois
19.99€



Alphabet en bois
14.99€



Jeu de formes en bois
24.99€

Figurines en Bois




La page affiche donc le titre de la sous-catégorie, puis tous les produits qui font partie à la fois de la sous-catégorie et de la catégorie parent.

Analysons à présent comment cela est géré en php.

```
<?php
session_start();

$idCategorie = htmlspecialchars($_GET['idCategorie']);

?>
```

Voici tout d'abord la première partie de la page qui récupère tout simplement l'id de la Catégorie sélectionnée.

```
<div class="containerProduit">
    <?php
        include("connect.inc.php");
        $stmt = $pdo->prepare("SELECT DISTINCT C.IDCATEG, C.NOMCATEG
                                FROM CATEGORIE C JOIN CATPERE CP ON C.IDCATEG = CP.IDCATEG
                                WHERE CP.IDCATEG_PERE = :IDCATPERE;");
        $stmt->execute(["IDCATPERE" => $idCategorie]);
        $sous_categories = $stmt->fetchAll();

        foreach ($sous_categories as $sous_categorie) {
            echo "<div class=\"sousCategorieContainer\">";
            echo "<h1 class=\"titreSousCateg\">".$sous_categorie['NOMCATEG']."</h1>";
            echo "<ul>";
            try {
                $stmt = $pdo->prepare(" SELECT DISTINCT P.IDPROD, P.NOMPROD,
                                         P.DESCPROD, P.PRIXHT, P.COULEUR, P.COMPOSITION, P.POIDSProduit, P.QTESTOCK
                                         FROM PRODUIT P
                                         JOIN APPARTENIRCATEG A1 ON P.IDPROD = A1.IDPROD
                                         JOIN CATEGORIE C1 ON A1.IDCATEG = C1.IDCATEG
                                         JOIN CATPERE CP ON C1.IDCATEG = CP.IDCATEG
                                         JOIN APPARTENIRCATEG A2 ON P.IDPROD = A2.IDPROD
                                         WHERE C1.IDCATEG = :IDSousCATEG
                                         AND CP.IDCATEG_PERE = :IDCATPERE
                                         AND A2.IDCATEG = CP.IDCATEG_PERE;");
                $stmt->execute(["IDSousCATEG" => $sous_categorie['IDCATEG'], "IDCATPERE" => $idCategorie]);
                $produits = $stmt->fetchAll();
            } catch (Exception $e) {
                echo "Error: " . $e->getMessage();
            }
            echo "</ul>";
            echo "</div>";
        }
    </?php>
</div>
```

Voici nos deux requêtes SQL très importantes pour la suite. La première requête permet simplement de récupérer toutes les sous-catégorie de notre catégorie parent et de les mettre dans notre variable “*sous_categories*”

On lance ensuite une boucle foreach pour chacune de nos sous-catégories. Dans cette boucle on affiche le titre puis une liste des produits.

Pour afficher cette liste des produits il nous faut une autre requête SQL, cette dernière va nous permettre de récupérer tous les produits qui appartiennent à la fois à la sous-catégories actuelle de la boucle et à la fois à la catégorie parent. On exécute donc ensuite notre requête avec pour paramètre les deux pré-requis cités plus tôt et on récolte tous ses produits dans la variable "*produits*".

```
foreach ($produits as $produit) {
    echo "<li>";
    echo "<a href='descProduit.php?idProd=".$produit['IDPROD']."'>";
    echo "<div class=\"produitCard\">";
    echo "<div class=\"imageContainer\">";
    echo "<img src='./images/prod". htmlspecialchars($produit['IDPROD']).".png' alt='Image du produit'>";
    echo "</div>";
    echo "<div class=\"infoContainer\">";
    echo "<h2>".$produit['NOMPROD']."</h2>";
    echo "<p>".$produit['PRIXHT']."€</p>";
    echo "</div>";
    echo "</a>";
    echo "</li>";
}
```

Nous lançons ensuite une nouvelle boucle foreach imbriqué dans la précédente afin d'afficher un à un les produits. L'image du produit est sélectionnée dans notre dossier "./images/" avec pour nom "prod(*IDPROD*)". Et puis tout ce contenu est à l'intérieur d'une balise "<a>" qui est un lien menant vers la page de description du produit.

2.2 La page categoriEnfants.php :

Cette page ressemble beaucoup à la précédente à part qu'elle affiche uniquement les produits d'une seule sous-catégorie. Voici un aperçu de la page

de la sous-catégorie “Figurine Classique” qui est une catégorie enfant de la catégorie “Figurines”.

Filtres

Figurine Classique

Age
Prix



Figurine dinosaure

14.99€



Figurine super-héros

19.99€



Figurine animal

12.99€



Figurine chevalier

14.99€



Barbie

59.99€



Schleich

9.99€



Ludorama figurine

49.99€

La page affiche donc le titre de la sous-catégorie, puis affiche une liste des produits qui la composent.

Analysons à présent le rendu php qui mène à un tel résultat.

```
<?php
session_start();
$idCategorie = htmlspecialchars($_GET['idCategorie']);
$idCategoriePere = htmlspecialchars($_GET['idCategoriePere']);
?>
```

Ici, comme pour la page categoriePere.php, on récupère l'id de la catégorie sélectionnée mais ici on récupère également l'id de sa catégorie parent qui est envoyé depuis le lien de redirection.

```
<div class="containerProduit">
    <?php
        include("connect.inc.php");
        $stmt = $pdo->prepare("SELECT DISTINCT P.IDPROD, P.NOMPROD, P.DESCPROD, P.PRIXHT, P.COULEUR,
                                P.COMPOSITION, P.POIDSPRODUCT, P.QTESTOCK
                            FROM PRODUIT P JOIN APPARTENIRCATEG A1 ON P.IDPROD = A1.IDPROD
                            JOIN CATEGORIE C1 ON A1.IDCATEG = C1.IDCATEG
                            JOIN CATPERE CP ON C1.IDCATEG = CP.IDCATEG
                            JOIN APPARTENIRCATEG A2 ON P.IDPROD = A2.IDPROD
                            WHERE C1.IDCATEG = :IDSOUSCATEG
                            AND CP.IDCATEG_PERE = :IDCATPERE
                            AND A2.IDCATEG = CP.IDCATEG_PERE;");
        $stmt->execute(["IDCATPERE" => $idCategoriePere, "IDSOUSCATEG" => $idCategorie]);
        $produits = $stmt->fetchAll();

        $stmt2 = $pdo->prepare("SELECT * FROM CATEGORIE WHERE IDCATEG = :IDCATEG;");
        $stmt2->execute(["IDCATEG" => $idCategorie]);
        $categorie = $stmt2->fetch();
```

On commence tout d'abord par effectuer une requête SQL afin de récupérer tous les produits qui appartiennent à la sous-catégorie sélectionnée mais aussi à sa catégorie parent. Cette requête est très similaire à la requête de la page précédente, une fois la requête effectuée on récupère son résultat dans la variable "*produits*".

```

echo "<div class=\"sousCategorieContainer\">";
    echo "<h1 class=\"titreSousCateg\">".$categorie["NOMCATEG"]."</h1>";
    echo "<ul>";
        foreach ($produits as $produit) {
            $imagePath = "./images/prod". htmlspecialchars($produit['IDPROD']) . ".png" ;
            echo "<li>";
                echo "<a href='descProduit.php?idProd=".$produit['IDPROD']."'>";
                echo "<div class=\"produitCard\">";
                    echo "<div class=\"imageContainer\">";
                        echo "<img src='".$imagePath' alt='image principale du produit'
                                id='main-product-image'>";
                    echo "</div>";
                    echo "<div class=\"infoContainer\">";
                        echo "<h2>".$produit['NOMPROD']. "</h2>";
                        echo "<p>".$produit['PRIXHT']."€</p>";
                    echo "</div>";
                echo "</a>";
            echo "</li>";
        }
    echo "</ul>";
    echo "</div>";

```

Après avoir récupéré les produits, on lance une boucle foreach comme dans la page précédente et on y affiche tous les produits avec leur image et un lien pour accéder à leur page de description de produit.

Ainsi, voilà comment est gérer l'affichage des produits en fonction d'une catégorie sélectionnée.

3. La page Description Produit :

La page de description de produit est la page sur laquelle nous pouvons voir les détails relatifs à un produit ainsi que les avis, grâce à l'url nous accédons aux produits via leurs id, par exemple avec l'url suivant : /descProduit.php?idProd=11. De plus nous pouvons ajouter le produit au panier. Voici un exemple de la page :



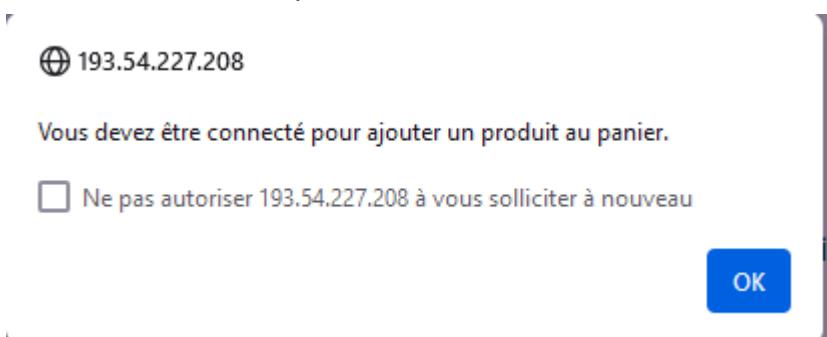
Voici l'onglet Avis où on peut déposer un avis et voir ceux des autres clients :



Lorsque l'on clique sur ajouter au panier nous avons d'abord une pop up de confirmation :



Si nous ne sommes pas connecté nous avons ceci :



Sinon un message nous confirmant l'ajout au panier :



Voyons maintenant comment tout ça se manifeste sous forme de code :

```
if (!isset($_GET['idProd']) || empty($_GET['idProd'])) {
```

```

echo "Produit non spécifié.";

exit();

}

$idProd = intval($_GET['idProd']); // Sécurisation de l'entrée

// Récupération des informations du produit depuis la base de données

$stmt = $pdo->prepare("

SELECT p.*,

m.NOMMARQUE,

GROUP_CONCAT(c.NOMCATEG SEPARATOR ', ') AS CATEGORIES

FROM PRODUIT p

LEFT JOIN MARQUE m ON p.IDMARQUE = m.IDMARQUE

LEFT JOIN APPARTENIRCATEG ac ON p.IDPROD = ac.IDPROD

LEFT JOIN CATEGORIE c ON ac.IDCATEG = c.IDCATEG

WHERE p.IDPROD = ?

GROUP BY p.IDPROD

");

$stmt->execute([$idProd]);

$produit = $stmt->fetch();

```

```
if (!$produit) {

    echo "Produit introuvable.";

    exit();

}

// Définition de l'image principale du produit (colonne IMAGE dans PRODUIT)

$imagePath = "./images/prod". htmlspecialchars($produit['IDPROD']) . ".png" ; // Image
par défaut si aucune image n'est trouvée

?>
```

Ici on vérifie que nous avons bien un produit existant, si c'est le cas on effectue une requête pour récupérer les informations de ce produits.

```
<?php if ($produit['DISPONIBLE'] != 1): ?>

    <p class="stop-sell">Nous ne vendons plus ce produit</p>

    <?php else: ?>

        <?php if ($produit['QTESTOCK'] > 0): ?>

            <button class="btn-cart" onclick="confirmAddToCart()">Ajouter au
panier</button>

        <?php else: ?>

            <p class="out-of-stock">Rupture de stock</p>

        <?php endif; ?>
```

```
<?php endif; ?>
```

Partie importante aussi, ici on vérifie que le produit est disponible, c'est à dire qu'on le vend. Si nous le vendons plus nous affichons à la place du bouton "Ajouter au panier" un message pour dire qu'on ne vend plus le produit. Si nous le vendons encore on vérifie le stock, si il est en rupture de stock on l'affiche sinon on affiche simplement le bouton qui a la fonction confirmAddToCart() associé.

```
function confirmAddToCart() {

    const confirmation = confirm("Voulez-vous ajouter ce produit au panier ?");

    if (confirmation) {

        addToCart();

    }

}

function addToCart() {

    const productId = <?php echo $idProd; ?>

        const clientId = <?php echo isset($_SESSION['user']['IDCLIENT']) ? $_SESSION['user']['IDCLIENT'] : 'null'; ?>;

    if (!clientId) {

        alert("Vous devez être connecté pour ajouter un produit au panier.");

        return;

    }

}
```

```
}

fetch('add_to_panier.php', {

    method: 'POST',

    headers: {

        'Content-Type': 'application/json'

    },

    body: JSON.stringify({ idProd: productId, idClient: clientId })

})

.then(response => response.json())

.then(data => {

    if (data.success) {

        alert("Produit ajouté au panier avec succès !");

    } else {

        alert("Erreur lors de l'ajout du produit au panier : " + data.message);

    }

})

.catch(error => {

    console.error('Erreur:', error);

    alert("Erreur lors de l'ajout du produit au panier.");
})
```

```
});
```

```
}
```

Ces deux fonctions sont complémentaires, confirmAddToCart affiche la pop-up de confirmation et AddToCart utilise add_to_panier.php, que voici :

```
$stmt = $pdo->prepare("SELECT * FROM PANIER WHERE IDCLIENT = ? AND IDPROD = ? AND IDCARTE = 0");

$stmt->execute([$idClient, $idProd]);

$existingEntry = $stmt->fetch();

if ($existingEntry) {

    // Mettre à jour la quantité si le produit existe déjà

    $stmt = $pdo->prepare("UPDATE PANIER SET QUANTITEPROD = QUANTITEPROD + 1 WHERE IDCLIENT = ? AND IDPROD = ? AND IDCARTE = 0");

    $stmt->execute([$idClient, $idProd]);

} else {

    // Ajouter une nouvelle entrée dans le panier

    $stmt = $pdo->prepare("INSERT INTO PANIER (IDCLIENT, IDPROD, QUANTITEPROD) VALUES (?, ?, 1)");

    $stmt->execute([$idClient, $idProd]);

}
```

Voici la partie importante, ici on vérifie que le produit ne soit pas déjà dans notre panier, si c'est le cas on va ajouter 1 à la quantité présente dans notre panier, sinon on va l'insérer

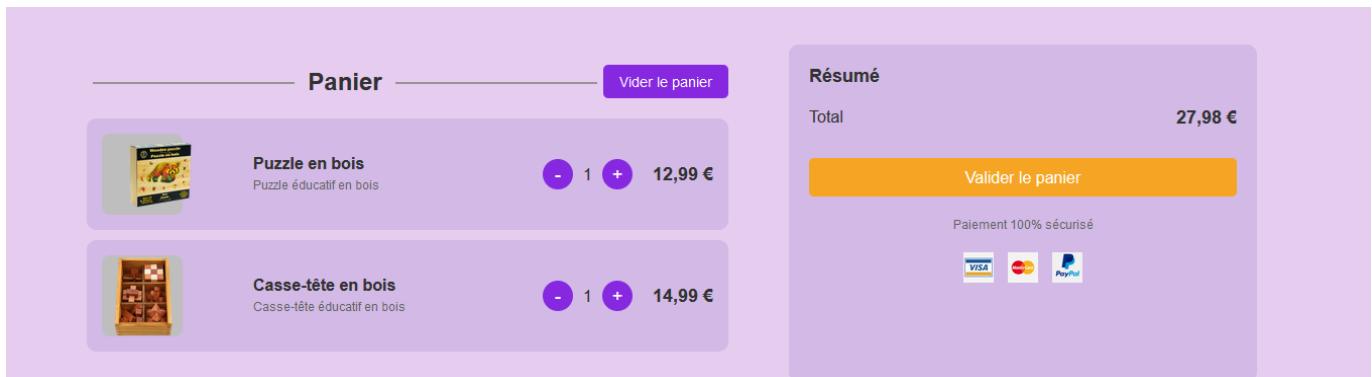
dans notre panier.

4. La page Panier:

Nous avons ensuite la page panier sur laquelle nous pouvons voir les produits que nous avons ajoutés au panier, tout d'abord si nous ne sommes pas connectés la page ressemble à ceci :



Si nous sommes bien connectés, la page affiche les produits dans notre panier, le prix du panier(HT), des boutons pour modifier la quantité à commander pour chaque produit, un bouton pour vider le panier et un bouton pour commander le panier :



Nous allons voir comment cela se manifeste dans le code :

```
$idClient = intval($_SESSION['user']['IDCLIENT']);
```

```
// Récupère les produits du panier pour l'utilisateur connecté

$stmt = $pdo->prepare("

    SELECT p.NOMPROD, p.IDPROD, p.PRIXHT, p.DESCPROD, pa.QUANTITEPROD

    FROM PANIER pa

    JOIN PRODUIT p ON pa.IDPROD = p.IDPROD

    WHERE pa.IDCLIENT = ? AND pa.IDCOMMANDE = 0

");

$stmt->execute([$idClient]);

$produitsPanier = $stmt->fetchAll();
```

Ici on récupère les produits de notre panier, la condition IDCOMMANDE=0 nous permet de vérifier que le panier est le panier actuel et n'a pas été commandé.

```
if (empty($produitsPanier)) {

    echo "<p>Votre panier est vide.</p>";

} else {

    // Affiche chaque produit du panier

    foreach ($produitsPanier as $produit) {

        $imagePath = "./images/prod" .
htmlspecialchars($produit['IDPROD']) . ".png"; // Chemin d'image

        $prix = number_format($produit['PRIXHT'], 2, ',', ' ') . " €";
}
```

```
echo "

<div class='product'>

    <div class='productImage'>

        <img src='{$imagePath}' alt="" .

htmlspecialchars($produit['NOMPROD']) . "'>


    </div>

    <div class='productDetails'>

        <h3>" . htmlspecialchars($produit['NOMPROD']) . "</h3>

        <p>" . htmlspecialchars($produit['DESCPROD']) . "</p>


    </div>

    <div class='productQuantity'>

        <button onclick='updateQuantity(" . $produit['IDPROD'] .
", -1)' " . ($produit['QUANTITERPROD'] <= 1 ? "disabled" : "") . ">-</button>

        " . htmlspecialchars($produit['QUANTITEPROD']) . "

        <button onclick='updateQuantity({$produit['IDPROD']}, 1)'>+</button>


    </div>

    <div class='productPrice'>

        {$prix}

    </div>


</div>";
```

```
}
```

```
}
```

Si il n'y a aucun produit dans le panier, on affiche "Votre panier est vide." sinon on affiche les produits avec les deux boutons qui ont updateQuantity() associé qui appelle la page update_panier.php que voici :

```
$stmt = $pdo->prepare("SELECT * FROM PANIER WHERE IDCLIENT = ? AND IDPROD = ?");

$stmt->execute([$idClient, $idProd]);

$existingEntry = $stmt->fetch();

if ($existingEntry) {

    $newQuantity = $existingEntry['QUANTITEPROD'] + $delta;

    if ($newQuantity > 0) {

        $stmt = $pdo->prepare("UPDATE PANIER SET QUANTITEPROD = ? WHERE IDCLIENT = ? AND IDPROD = ?");

        $stmt->execute([$newQuantity, $idClient, $idProd]);

    } else {

        // Supprimer le produit si la quantité devient 0 ou moins

        $stmt = $pdo->prepare("DELETE FROM PANIER WHERE IDCLIENT = ? AND IDPROD = ?");

        $stmt->execute([$idClient, $idProd]);
    }
}
```

```
    } else {

        echo json_encode(['success' => false, 'message' => 'Produit non trouvé dans
le panier.']);

        exit;

    }
```

On vérifie que le produit soit bien dans le panier puis on ajoute ou enlève 1 à la quantité en fonction du bouton cliqué.

```
<?php

if (!empty($produitsPanier)) {

    $total = 0;

    foreach ($produitsPanier as $produit) {

        $total += $produit['PRIXHT'] * $produit['QUANTITEPROD'];

    }

    $totalFormatted = number_format($total, 2, ',', ' ') . " €";

    echo "

<h4>Résumé</h4>

<div class='priceSummary'>

    <span>Total</span>

    <span class='price'>{$totalFormatted}</span>

</div>
```

```
<form action='commande.php' method='post'>

    <button type='submit' class='btn validate'>Valider le
panier</button>

</form>

<div class='paymentSecurity'>

    <p>Paiement 100% sécurisé</p>

    <div class='paymentIcons'>

        <img src='images/visa.png' alt='Visa'>

        <img src='images/mastercard.png' alt='MasterCard'>

        <img src='images/paypal.png' alt='PayPal'>

    </div>

</div>";

}

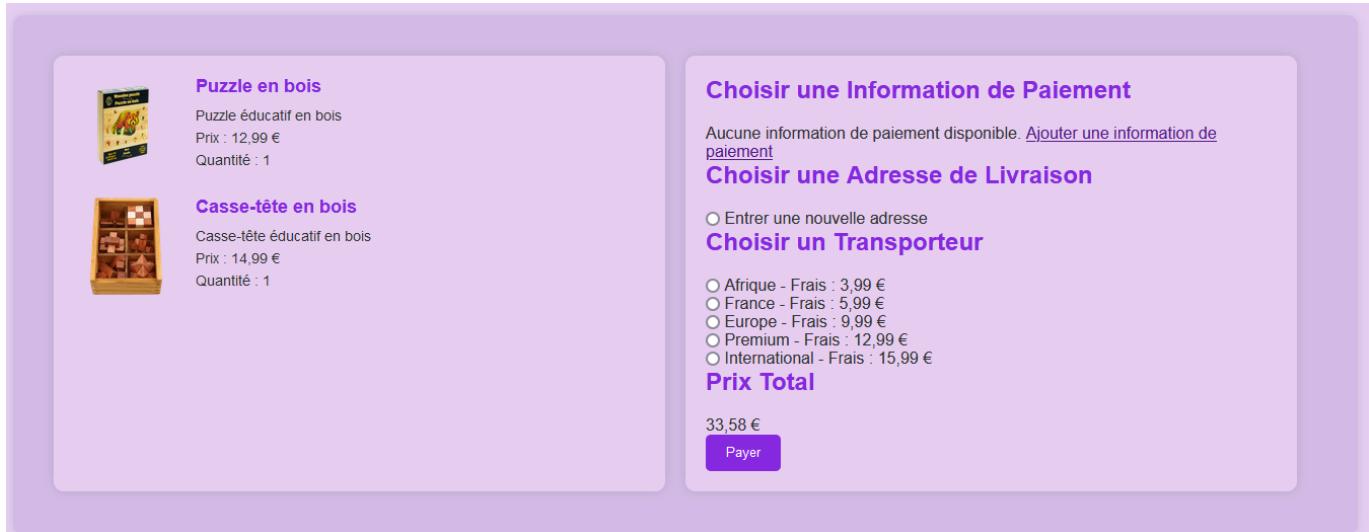
?>
```

Enfin on calcule et affiche le prix du panier et le bouton qui nous redirige vers commande.php

5. Le Processus de Commande :

Les commandes sont gérées via plusieurs pages, des informations de paiement jusqu'à la confirmation de la commande.

5.1 La page commande.php



Puzzle en bois  Puzzle éducatif en bois Prix : 12,99 € Quantité : 1	Casse-tête en bois  Casse-tête éducatif en bois Prix : 14,99 € Quantité : 1	Choisir une Information de Paiement Aucune information de paiement disponible. Ajouter une information de paiement Choisir une Adresse de Livraison <input type="radio"/> Entrer une nouvelle adresse Choisir un Transporteur <input type="radio"/> Afrique - Frais : 3,99 € <input type="radio"/> France - Frais : 5,99 € <input type="radio"/> Europe - Frais : 9,99 € <input type="radio"/> Premium - Frais : 12,99 € <input type="radio"/> International - Frais : 15,99 € Prix Total 33,58 € <input type="button" value="Payer"/>
---	---	--

Sur cette page on peut voir les produits que l'on va commander et les différentes informations nécessaires pour effectuer la commande.

```
<?php
include("header.php");

// Récupérer les produits du panier pour l'utilisateur connecté
$idClient = intval($_SESSION['user']['IDCLIENT']);
$stmt = $pdo->prepare(
    "SELECT p.NOMPROD, p.IDPROD, p.PRIXHT, p.DESCPROD, pa.QUANTITEPROD
     FROM PANIER pa
    JOIN PRODUIT p ON pa.IDPROD = p.IDPROD
   WHERE pa.IDCLIENT = ? AND pa.IDCOMMANDE = 0
");
$stmt->execute([$idClient]);
$produitsPanier = $stmt->fetchAll();

// Récupérer les informations de paiement pour l'utilisateur connecté
$stmt = $pdo->prepare(
    "SELECT ip.NUMCB, ip.NOMCOMPLETCB, ip.DATEEXP
     FROM INFORMATIONPAIEMENT ip
    JOIN POSSEDERIP pi ON ip.NUMCB = pi.NUMCB
   WHERE pi.IDCLIENT = ?
");

```

```

");
$stmt->execute([$idClient]);
$infosPaiement = $stmt->fetchAll();

// Récupérer les adresses pour l'utilisateur connecté
$stmt = $pdo->prepare("
    SELECT a.IDADRESSE, a.NUMRUE, a.NOMRUE, a.COMPLEMENTADR, a.NOMVILLE, a.CODEPOSTAL,
a.PAYS
    FROM ADRESSE a
    JOIN POSSEDERADR pa ON a.IDADRESSE = pa.IDADRESSE
    WHERE pa.IDCLIENT = ?
");
$stmt->execute([$idClient]);
$adresses = $stmt->fetchAll();

// Récupérer les transporteurs
$stmt = $pdo->prepare("
    SELECT IDTRANSPORTEUR, TYPEEXP, FRAISEXP
    FROM TRANSPORTEUR
    ORDER BY FRAISEXP ASC
");
$stmt->execute();
$transporteurs = $stmt->fetchAll();
?>

```

Tout d'abord nous effectuer toute les requêtes pour récupérer les informations à afficher.

```

<form action="process_payment.php" method="post" onsubmit="return confirmPayment()">
    <?php
    if (empty($infosPaiement)) {
        echo "<p>Aucune information de paiement disponible. <a
href='info_paiement.php'>Ajouter une information de paiement</a></p>";
    } else {
        foreach ($infosPaiement as $info) {
            echo "
                <p><a href='info_paiement.php'>Ajouter une information de paiement</a></p>
                <div class='payment-method'>

```

```

        <input type='radio' id='payment_{$info['NUMCB']}' name='numCB'
value='{$info['NUMCB']}' required>
            <label for='payment_{$info['NUMCB']}'>Carte se terminant par <span>" .
substr($info['NUMCB'], -4) . "</span> - Expire le {$info['DATEEXP']}

```

```

<div class="payment-method">
    <label for="complementAdr">Complément d'adresse</label>
    <input type="text" id="complementAdr" name="complementAdr">
</div>

<div class="payment-method">
    <label for="nomVille">Ville</label>
    <input type="text" id="nomVille" name="nomVille">
</div>

<div class="payment-method">
    <label for="codePostal">Code postal</label>
    <input type="text" id="codePostal" name="codePostal">
</div>

<div class="payment-method">
    <label for="pays">Pays</label>
    <input type="text" id="pays" name="pays">
</div>

</div>
<h2>Choisir un Transporteur</h2>
<?php
if (empty($transporteurs)) {
    echo "<p>Aucun transporteur disponible.</p>";
} else {
    foreach ($transporteurs as $transporteur) {
        echo "
<div class='transport-method'>
    <input type='radio' id='transport_{$transporteur['IDTRANSPORTEUR']}' name='idTransporteur' value='{$transporteur['IDTRANSPORTEUR']}' data-frais='{$transporteur['FRAISEXP']}' required>
        <label for='transport_{$transporteur['IDTRANSPORTEUR']}'>
            {$transporteur['TYPEEXP']} - Frais : " .
number_format($transporteur['FRAISEXP'], 2, ',', ' ') . " €
        </label>
    </div>";
    }
}
?>
```

```
<h2>Prix Total</h2>
<p id="totalPrice"><?php echo number_format($totalTTC, 2, ',', ' '); ?> €</p>
<input type="hidden" id="totalPriceInput" name="totalPrice" value="<?php echo
$totalTTC; ?>">
<button type="submit" class="btn validate">Payer</button>
</form>
```

Ici nous affichons les informations de paiement disponibles avec un bouton qui nous redirige vers info_paiement.php pour en entrer de nouvelles, puis on affiche les différentes adresses avec un formulaire pour entrer une nouvelle adresse si on le souhaite et enfin on affiche les différents transporteurs.

5.2 La page info_paiement.php



Numéro de carte

Nom sur la carte

Date d'expiration

 MM/AA

CVV

Ajouter

Comme on peut le voir on a un formulaire pour entrer les information de paiement.

```
// Nettoyer les entrées avec trim()
$cardNumber = trim($_POST['cardNumber']);
$cardName = trim($_POST['cardName']);
$expiryDate = trim($_POST['expiryDate']);
$cvv = trim($_POST['cvv']);
$idClient = intval($_SESSION['user']['IDCLIENT']);
```

```

// Regex patterns
$cardNumberPattern = '/^\d{16}$/';
$cardNamePattern = '/^([A-Za-z\s]+)$/';
$expiryDatePattern = '/^(0[1-9]|1[0-2])\/(0[1-9]|1[0-2])$/';
$cvvPattern = '/^\d{3}$/';

// Validate inputs
if (!preg_match($cardNumberPattern, $cardNumber)) {
    $error = 'Le numéro de carte doit contenir 16 chiffres.';
} elseif (!preg_match($cardNamePattern, $cardName)) {
    $error = 'Le nom doit contenir uniquement des lettres et des espaces.';
} elseif (!preg_match($expiryDatePattern, $expiryDate)) {
    $error = 'La date d\'expiration doit être au format MM/AA et respecter les
contraintes spécifiées.';
} elseif (!preg_match($cvvPattern, $cvv)) {
    $error = 'Le CVV doit contenir 3 chiffres.';
} else {
    // Convert expiry date to YYYY-MM-DD format
    $expiryDateFormatted = convertExpiryDateToFullDate($expiryDate);

    try {
        // Commencer une transaction
        $pdo->beginTransaction();

        // Insérer les informations de paiement dans la table INFORMATIONPAIEMENT
        $stmt = $pdo->prepare("INSERT INTO INFORMATIONPAIEMENT (NUMCB, NOMCOMPLETCB,
DATEEXP, CRYPTOGRAMME) VALUES (?, ?, ?, ?)");
        $stmt->execute([$cardNumber, $cardName, $expiryDateFormatted, $cvv]);

        // Insérer la relation entre le client et les informations de paiement dans
la table POSSEDERIP
        $stmt = $pdo->prepare("INSERT INTO POSSEDERIP (NUMCB, IDCLIENT) VALUES (?, ?)");
        $stmt->execute([$cardNumber, $idClient]);

        // Valider la transaction
    }
}

```

```

    $pdo->commit();

    // Rediriger vers la page de commande pour effacer les messages d'erreur
    header("Location: commande.php");
    exit();
} catch (Exception $e) {
    // Annuler la transaction en cas d'erreur
    $pdo->rollBack();
    $error = "Erreur lors de l'ajout des informations de paiement : " .
    $e->getMessage();
}
}

// Fonction pour convertir MM/YY en YYYY-MM-DD
function convertExpiryDateToFullDate($expiryDate) {
    // Séparer MM et YY
    list($month, $year) = explode('/', $expiryDate);

    // Ajouter "20" au début de l'année (puisque l'année commence à partir de 2025 selon
    // la regex)
    $year = '20' . $year;

    // Créer une date au premier jour du mois
    $date = new DateTime("$year-$month-01");

    // Obtenir le dernier jour du mois
    $lastDay = $date->format('t');

    // Retourner la date au format YYYY-MM-DD
    return "$year-$month-$lastDay";
}

```

Nous avons donc des vérifications pour les entrées dans le formulaire grâce à des regex, pour la date le traitement est un peu différent car on la rentre dans le format MM/YY et dans la BD on doit la rentrer sous la forme d'une date total donc la fonction

convertExpiryDateToFullDate convertit la date dans la bonne forme enfin on insère les données dans la BD.

5.3 Process_payment.php

Process_payment gère le paiement dans son ensemble :

```
$numCB = $_POST['numCB'];
$idClient = intval($_SESSION['user']['IDCLIENT']);
$numRue = $_POST['numRue'];
$nomRue = $_POST['nomRue'];
$complementAdr = $_POST['complementAdr'];
$nomVille = $_POST['nomVille'];
$codePostal = $_POST['codePostal'];
$pays = $_POST['pays'];
$idAdresse = $_POST['idAdresse'];
$idTransporteur = $_POST['idTransporteur'];
$statutLivraison = 'En cours'; // Par défaut, le statut de livraison est "En cours"

// Générer le numéro de suivi (3 lettres suivies de 9 chiffres)
$lettres = strtoupper(substr(md5(rand()), 0, 3)); // 3 lettres aléatoires
$chiffres = str_pad(rand(100000000, 999999999), 9, '0', STR_PAD_LEFT); // 9
chiffres
$codeSuivi = $lettres . $chiffres;

// Récuperer la date de commande
$dateCommande = date("Y-m-d");
```

Ici on récupère d'abord toute les informations fournis sur la page commande.php.

```
// Commencer une transaction
$pdo->beginTransaction();
```

```

if($idAdresse === 'new'){
    // Insérer l'adresse dans la table Adresse
    $stmt = $pdo->prepare("INSERT INTO ADRESSE ( NUMRUE, NOMRUE, COMPLEMENTADR,
NOMVILLE, CODEPOSTAL, PAYS) VALUES (?, ?, ?, ?, ?, ?)");
    $stmt->execute([ $numRue, $nomRue, $complementAdr, $nomVille, $codePostal,
$pays]);

    // Récupérer l'ID de l'adresse générée
    $idAdresse = $pdo->lastInsertId();

    $stmt = $pdo->prepare("INSERT INTO POSSEDERADR (IDADRESSE, IDCLIENT) VALUES
(?, ?)");
    $stmt->execute([$idAdresse, $idClient]);
}

```

Ici si l'adresse est une nouvelle, on l'insère dans la base de données.

```

// Insérer la commande dans la table Commande
$stmt = $pdo->prepare("INSERT INTO COMMANDE (IDCLIENT, IDTRANSPORTEUR, NUMCB,
IDADRESSE, TYPEREGLEMENT, DATECOMMANDE, STATUTLIVRAISON, CODESUIVI) VALUES (?, ?, ?, ?, ?, ?, ?, ?)");
$stmt->execute([$idClient, $idTransporteur, $numCB, $idAdresse, 'CB',
$dateCommande, $statutLivraison, $codeSuivi]);

// Récupérer le numéro de commande généré
$numCommande = $pdo->lastInsertId();

// Mettre à jour la date de commande dans le panier
$stmt = $pdo->prepare("UPDATE PANIER SET IDCOMMANDE = ? WHERE IDCLIENT = ? AND
IDCOMMANDE = 0");
$stmt->execute([$numCommande, $idClient]);

// Mettre à jour le stock des produits
$stmt = $pdo->prepare("
    UPDATE PRODUIT p
    JOIN PANIER pa ON p.IDPROD = pa.IDPROD
    SET p.QTESTOCK = p.QTESTOCK - pa.QUANTITEPROD
")

```

```

        WHERE pa.IDCLIENT = ? AND pa.IDCOMMANDE = ?
    ");
$stmt->execute([$idClient, $numCommande]);

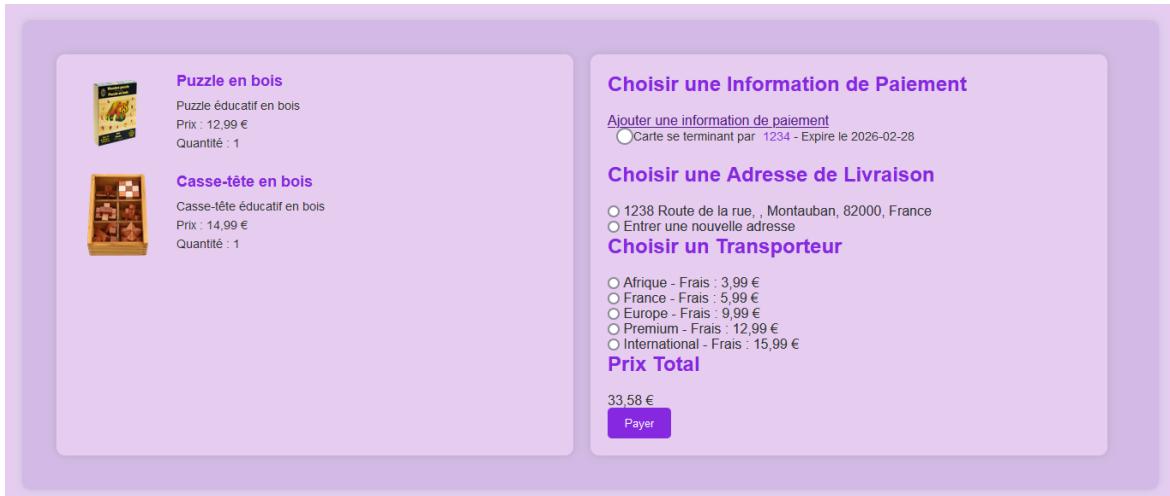
// Valider la transaction
$pdo->commit();

// Rediriger vers une page de confirmation
header("Location: confirmation.php?commande=" . $numCommande);
exit();

```

Ensuite on insère toute les informations dans la BD et on modifie panier en mettant le numéro de la commande dans panier. Une fois tout ça effectuer, si il n'y a pas eu d'erreur on est rediriger vers la page confirmation.

5.4 Confirmation.php



The screenshot shows a web page for a shopping cart. On the left, there are two items listed:

- Puzzle en bois**: A wooden educational puzzle. Price: 12.99 €, Quantity: 1.
- Casse-tête en bois**: A wooden educational puzzle. Price: 14.99 €, Quantity: 1.

On the right, there are three sections for payment and delivery:

- Choisir une Information de Paiement**: Options include adding payment info or using a card ending in 1234 (valid until 2026-02-28).
- Choisir une Adresse de Livraison**: Set to 1238 Route de la rue, , Montauban, 82000, France.
- Choisir un Transporteur**: Options include Afrique, France, Europe, Premium, and International, each with a price of 3.99 €, 5.99 €, 9.99 €, 12.99 €, and 15.99 € respectively.

Prix Total: 33.58 €. A purple "Payer" button is at the bottom.

```

if (!isset($_GET['commande'])) {
    echo "Numéro de commande manquant.";
    exit();
}

$numCommande = intval($_GET['commande']);

// Récupérer les informations de la commande
$stmt = $pdo->prepare(
    "SELECT c.NUMCOMMANDE, c.DATECOMMANDE, c.STATUTLIVRAISON, c.CODESUIVI,
        a.NUMRUE, a.NOMRUE, a.COMPLEMENTADR, a.NOMVILLE, a.CODEPOSTAL, a.PAYS,
        t.TYPEEXP, t.FRAISEXP
    FROM COMMANDE c
    JOIN ADRESSE a ON c.IDADRESSE = a.IDADRESSE
    JOIN TRANSPORTEUR t ON c.IDTRANSPORTEUR = t.IDTRANSPORTEUR
    WHERE c.NUMCOMMANDE = ?
");

$stmt->execute([$numCommande]);
$commande = $stmt->fetch();

if (!$commande) {
    echo "Commande non trouvée.";
    exit();
}

// Récupérer les produits de la commande
$stmt = $pdo->prepare(
    "SELECT p.NOMPROM, p.PRIXHT, pa.QUANTITEPROD
    FROM PANIER pa
    JOIN PRODUIT p ON pa.IDPROD = p.IDPROD
    WHERE pa.IDCLIENT = ? AND pa.IDCOMMANDE = ?
");

$stmt->execute([$_SESSION['user']['IDCLIENT'], $commande['NUMCOMMANDE']]);
$produitsPanier = $stmt->fetchAll();

// Calculer le prix total de la commande

```

```
$totalHT = 0;
foreach ($produitsPanier as $produit) {
    $totalHT += $produit['PRIXHT'] * $produit['QUANTITEPROD'];
}
$totalTTC = $totalHT * 1.2; // Ajout de la TVA (20%)
$total = $totalTTC + $commande['FRAISEXP'];

?>
```

D'abord on récupère comme d'habitude les informations relatives à la commande et on calcule le prix total de la commande en ajoutant le prix du transport et la TVA.

```
<h1>Merci pour votre commande !</h1>
<p>Votre paiement a été traité avec succès. Vous recevrez bientôt un email de confirmation.</p>

<h2>Détails de la commande</h2>
<p><strong>Prix de la commande :</strong> <?php echo number_format($total, 2, ',', ','); ?> €</p>
<p><strong>Numéro de commande :</strong> <?php echo htmlspecialchars($commande['NUMCOMMANDE']); ?></p>
<p><strong>Date de commande :</strong> <?php echo htmlspecialchars($commande['DATECOMMANDE']); ?></p>
<p><strong>Statut de livraison :</strong> <?php echo htmlspecialchars($commande['STATUTLIVRAISON']); ?></p>
<p><strong>Code de suivi :</strong> <?php echo htmlspecialchars($commande['CODESUIVI']); ?></p>

<h2>Adresse de livraison</h2>
<p><?php echo htmlspecialchars($commande['NUMRUE']) . ' ' .
htmlspecialchars($commande['NOMRUE']); ?></p>
<p><?php echo htmlspecialchars($commande['COMPLEMENTADR']); ?></p>
<p><?php echo htmlspecialchars($commande['NOMVILLE']) . ', ' .
htmlspecialchars($commande['CODEPOSTAL']) . ', ' . htmlspecialchars($commande['PAYS']); ?></p>
```

```
<h2>Type d'expédition</h2>
<p><?php echo htmlspecialchars($commande['TYPEEXP']); ?></p>
```

Enfin on affiche toute ces informations.

6. La Fonctionnalité de Recherche :

Rappelons la User Story :

En tant que client, je souhaite pouvoir chercher des produits par mots-clés et filtres (prix, marques, notes) afin de trouver rapidement les produits recherchés.

Importance

Must (Default)

Critères d'acceptation

Barre de recherche avec suggestions de mots-clés

Filtres avancés pour affiner les résultats (prix, notes, marques, etc.)

Affichage rapide des résultats en fonction des critères choisis

Détails

Implémenter une barre de recherche avec des filtres combinés pour permettre des recherches précises et rapides.

Afin de faire cela nous avons réfléchi à comment faire. Nous nous sommes inspiré de la page de recherche de LDLC :

LES RÉSULTATS POUR JEU

2507 PRODUITS	59 CATÉGORIES	100 MARQUES
2 507 produits correspondent		
Trier par Filtrer		
FILTRER LES PRODUITS : <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> CHERCHER PAR MOT-CLES <input type="text" value="Désignation, modèle ..."/> 🔍 </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> CATEGORIES Sélectionner ▼ </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> MARQUE Sélectionner ▼ </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> PRIX 4 3100 En € </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> ETAT DU PRODUIT Sélectionner ▼ </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> GENRE Sélectionner ▼ </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> MULTIJOUER Sélectionner ▼ </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> UNIVERS Sélectionner ▼ </div> <div style="border: 1px solid #ccc; padding: 5px; text-align: center; margin-bottom: 5px;"> + DE FILTRES ▾ </div> <div style="display: flex; justify-content: space-between; align-items: center;"> Voir uniquement les produits en stock </div> <div style="display: flex; justify-content: space-between; align-items: center; margin-top: 5px;"> Voir uniquement les produits vendus par LDLC </div>	<div style="display: flex; flex-wrap: wrap;"> <div style="width: 50%; margin-bottom: 10px;"> <div style="display: flex; justify-content: space-between;"> <div style="flex: 1;">  </div> <div style="flex: 1;"> <p>Ultimate Guard - Tapis de jeu 90 Mystic Space 90 x 90 cm Ultimate Guard - Tapis de jeu 90 Mystic Space 90 x 90 cm</p> </div> <div style="flex: 1; text-align: right;"> DISPO Exclu Web EN STOCK 35€⁹⁵ </div> </div> </div> <div style="display: flex; justify-content: space-between; align-items: center;">  <div style="flex: 1;"> <p>Harry Potter - Jeu d'Echecs Poudlard Jeu d'Echecs Harry Potter, modèle Poudlard.</p> </div> <div style="flex: 1; text-align: right;"> DISPO Exclu Web EN STOCK 233€⁹⁵ </div> </div> <div style="display: flex; justify-content: space-between; align-items: center;">  <div style="flex: 1;"> <p>Ultimate Guard - 50 pochettes Premium Soft Sleeves jeu de Tarot Pack de 50 pochettes Ultimate Guard Premium Soft Sleeves jeu de Tarot.</p> </div> <div style="flex: 1; text-align: right;"> DISPO Exclu Web EN STOCK 6€⁹⁵ </div> </div> <div style="display: flex; justify-content: space-between; align-items: center;">  <div style="flex: 1;"> <p>Ultimate Guard - 50 pochettes Premium Soft Sleeves jeux de plateau au format grand carré 50 pochettes Ultimate Guard Premium Soft Sleeves jeux de plateau au format grand carré.</p> </div> <div style="flex: 1; text-align: right;"> DISPO Exclu Web EN STOCK 6€⁹⁵ </div> </div> <div style="display: flex; justify-content: space-between; align-items: center;">  <div style="flex: 1;"> <p>Ultimate Guard - Tapis de jeu Monochrome Violet 61 x 35 cm Ultimate Guard - Tapis de jeu Monochrome Violet 61 x 35 cm</p> </div> <div style="flex: 1; text-align: right;"> DISPO Exclu Web EN STOCK 13€⁹⁵ </div> </div> </div>	

Afin de faire cela nous avons réfléchi à comment faire. Premièrement la barre de recherche à été mise dans le header afin d'y avoir accès dans toutes les pages.


```

<form method="GET" action="recherche.php">
  <input type="text" class="barreRecherche" name="mot_cle"
placeholder="Rechercher...">
</form>
</li>

```

Afin de créer la page recherche nous avons créé 2 procédures stockées sur phpmyadmin. Nous utilisons ces 2 procédures dans rechercheAvancee.php. Voici le code de rechercheAvancee.php

```
<?php
// rechercheAvancee.php
// Recherche avec les critères dynamiques
function rechercheAvancee($criteres, $pdo, $limit, $offset)
{
    try {
        // Convertir les critères des prix en NULL si vides
        $prix_min = isset($criteres['prix_min']) && $criteres['prix_min'] !== '' ? (float)$criteres['prix_min'] : NULL;
        $prix_max = isset($criteres['prix_max']) && $criteres['prix_max'] !== '' ? (float)$criteres['prix_max'] : NULL;

        // Préparer l'appel à la procédure stockée
        $stmt = $pdo->prepare("CALL SP_RECHERCHE_AVANCEE(:mot_cle, :categorie, :marque, :prix_min, :prix_max, :en_stock, :limit, :offset)");

        // Passer les paramètres correctement typés
        $stmt->bindValue(':mot_cle', $criteres['mot_cle'] ?? '', PDO::PARAM_STR);
        $stmt->bindValue(
            ':categorie',
            isset($criteres['categorie']) && $criteres['categorie'] !== '' ? (int)$criteres['categorie'] : NULL,
            isset($criteres['categorie']) && $criteres['categorie'] !== '' ? PDO::PARAM_INT : PDO::PARAM_NULL
        );

        $stmt->bindValue(':marque', $criteres['marque'] !== '' ? $criteres['marque'] : NULL, PDO::PARAM_STR);
        $stmt->bindValue(':prix_min', $prix_min, $prix_min === NULL ? PDO::PARAM_NULL : PDO::PARAM_STR);
    }
}
```

```

$stmt->bindValue(':prix_max', $prix_max, $prix_max === NULL ? PDO::PARAM_NULL :
PDO::PARAM_STR);

$stmt->bindValue(':en_stock', $criteres['en_stock'] !== NULL ?
(int)$criteres['en_stock'] : NULL, PDO::PARAM_INT);

$stmt->bindValue(':limit', (int)$limit, PDO::PARAM_INT);
$stmt->bindValue(':offset', (int)$offset, PDO::PARAM_INT);

// Débogage : Requête simulée
$requeteDebug = sprintf(
    "CALL SP_RECHERCHE_AVANCEE(:mot_cle = '%s', :categorie = %s, :marque = '%s',
:prix_min = %s, :prix_max = %s, :en_stock = %s, :limit = %d, :offset = %d)",
    $criteres['mot_cle'] ?? '',
    $criteres['categorie'] !== '' ? (int)$criteres['categorie'] : "NULL",
    $criteres['marque'] ?? "NULL",
    $prix_min !== NULL ? $prix_min : "NULL",
    $prix_max !== NULL ? $prix_max : "NULL",
    $criteres['en_stock'] !== NULL ? (int)$criteres['en_stock'] : "NULL",
    (int)$limit,
    (int)$offset
);

// Exécuter la procédure stockée
$stmt->execute();

// Récupérer les résultats
$resultats = $stmt->fetchAll(PDO::FETCH_ASSOC);

// Retourner les résultats
return $resultats;
} catch (PDOException $e) {
    // Gérer les erreurs SQL
    echo "<h4>Erreur SQL :</h4><pre>" . $e->getMessage() . "</pre>";
    return [];
}
}

```

```
// Fonction pour compter le total des produits
function countProduits($criteres, $pdo)
{
    try {
        // Conversion des critères de prix
        $prix_min = !empty($criteres['prix_min']) ? (float)$criteres['prix_min'] : NULL;
        $prix_max = !empty($criteres['prix_max']) ? (float)$criteres['prix_max'] : NULL;

        // Préparer l'appel à la procédure stockée
        $stmt = $pdo->prepare("CALL SP_COUNT_PRODUITS(:mot_cle, :categorie, :marque,
        :prix_min, :prix_max, :en_stock)");

        // Binder les paramètres dynamiquement
        $stmt->bindValue(':mot_cle', $criteres['mot_cle'] ?? '', PDO::PARAM_STR);
        $stmt->bindValue(':categorie', isset($criteres['categorie']) &&
$criteres['categorie'] !== '' ? (int)$criteres['categorie'] : NULL, PDO::PARAM_INT);
        $stmt->bindValue(':marque', $criteres['marque'] ?? NULL, PDO::PARAM_STR);
        $stmt->bindValue(':prix_min', $prix_min, is_null($prix_min) ? PDO::PARAM_NULL : PDO::PARAM_STR);
        $stmt->bindValue(':prix_max', $prix_max, is_null($prix_max) ? PDO::PARAM_NULL : PDO::PARAM_STR);
        $stmt->bindValue(':en_stock', isset($criteres['en_stock']) ?
(int)$criteres['en_stock'] : NULL, PDO::PARAM_INT);

        // Exécuter la procédure
        $stmt->execute();
        $result = $stmt->fetch(PDO::FETCH_ASSOC);

        // Retourner le total ou 0 si non défini
        return $result['total'] ?? 0;
    } catch (PDOException $e) {
        // Afficher les erreurs SQL pour débogage
        echo "<h4>Erreur SQL :</h4><pre>" . $e->getMessage() . "</pre>";
        return 0;
    }
}
```

```
}
```

rechercheAvancee.php sert à appeler les 2 procédures stockées. countProduit permet d'obtenir le nombre de produit (qui sera utilisé pour un système de pagination)
rechercheAvancee permet d'obtenir les produits existant avec tous les filtres possibles

Nous remarquons que la barre de recherche relance sur recherche.php.

```
<?php
//recherche.php
require_once 'rechercheAvancee.php'; // Inclure la fonction rechercheAvancee
require_once 'connect.inc.php'; // Connexion PDO

// Récupérer la page actuelle (par défaut 1 si non définie)
$page = isset($_GET['page']) && is_numeric($_GET['page']) ? (int)$_GET['page'] : 1;
$produitsParPage = 12; // Nombre de produits par page

// Calculer l'offset
$offset = ($page - 1) * $produitsParPage;

// Récupérer les termes de recherche depuis la barre de recherche
$criteres = [
    'mot_cle' => $_GET['mot_cle'] ?? '',
    'categorie' => $_GET['categorie'] ?? NULL,
    'marque' => $_GET['marque'] ?? NULL,
    'prix_min' => $_GET['prix_min'] ?? NULL,
    'prix_max' => $_GET['prix_max'] ?? NULL,
    'en_stock' => isset($_GET['en_stock']) ? 1 : NULL, // Si en_stock est défini, on le
    met à 1, sinon NULL
];
```

```
// Charger les catégories dynamiquement
$stmt = $pdo->query("SELECT IDCATEG, NOMCATEG FROM CATEGORIE ORDER BY NOMCATEG ASC");
$categories = $stmt->fetchAll(PDO::FETCH_ASSOC);

// Charger les marques dynamiquement
$stmt = $pdo->query("SELECT DISTINCT NOMMARQUE FROM MARQUE ORDER BY NOMMARQUE ASC");
$marques = $stmt->fetchAll(PDO::FETCH_COLUMN);

// Récupérer les produits via la fonction rechercheAvancee
$resultats = rechercheAvancee($criteres, $pdo, $produitsParPage, $offset);

// Compter le total des produits pour générer la pagination
$totalProduits = countProduits($criteres, $pdo);
$totalPages = ceil($totalProduits / $produitsParPage);
?>
```

Que fait ce code-ci ?

Afin de faciliter la lecture de la page nous avons décidé de séparer le code en plusieurs parties. 2 pour être plus précis. La première est le côté traitement de données dans recherche.php (code donné plus haut). Ce traitement de données permet de récupérer toutes les données nécessaires au bon fonctionnement de la page. Que se soit les différentes marques ou les différents. Nous remarquons l'appel de rechercheAvancee qui appelle SP_RECHERCHE_AVANCEE et de countProduits qui appelle SP_COUNT_PRODUITS.

```

<div class="recherche-main">
    <!-- Section des filtres -->
    <div class="recherche-filtres">
        <h3>Filtres</h3>
        <form action="recherche.php" method="get">
            <label for="mot_cle">Mot-clé :</label>
            <input type="text" name="mot_cle" id="mot_cle" value=<?= htmlspecialchars($criteres['mot_cle']) ?>> <br>

            <label for="categorie">Catégorie :</label>
            <select name="categorie" id="categorie">
                <option value="">-- Toutes les catégories --</option>
                <?php foreach ($categories as $categorie): ?>
                    <option value=<?= htmlspecialchars($categorie['IDCATEG']) ?>>
                        <?= $criteres['categorie'] == $categorie['IDCATEG'] ?>
                            'selected' : '' ?>>
                    <?= htmlspecialchars($categorie['NOMCATEG']) ?>
                </option>
            <?php endforeach; ?>
        </select><br>
        <label for="marque">Marque :</label>
        <select name="marque" id="marque">
            <option value="">-- Toutes les marques --</option>
            <?php foreach ($marques as $marque): ?>
                <option value=<?= htmlspecialchars($marque) ?>>
                    <?= $criteres['marque'] == $marque ? 'selected' : '' ?>>
                    <?= htmlspecialchars($marque) ?>
                </option>
            <?php endforeach; ?>
        </select><br>
        <label for="prix_min">Prix minimum :</label>
        <input type="number" step="0.01" name="prix_min" id="prix_min" value=<?= htmlspecialchars($criteres['prix_min']) ?>> <br>
        <label for="prix_max">Prix maximum :</label>

```

```
<input type="number" step="0.01" name="prix_max" id="prix_max"
value=<?= htmlspecialchars($criteres['prix_max']) ?>><br>
<label for="en_stock">
    En stock uniquement
    <input type="checkbox" name="en_stock" id="en_stock" <?=
$criteres['en_stock'] ? 'checked' : '' ?>>
</label><br>
<button type="submit">Rechercher</button>
</form>
</div>
```

Cette section la permet d'avoir cette partie là de la page.

Filtres

Mot-clé :

Catégorie :

Marque :

Prix minimum :

Prix maximum :

En stock uniquement

Avec toutes les catégories et toutes les marques.

Filtres

Mot-clé :

Catégorie :

Marque :

Prix minimum :

Prix maximum :

En stock :

Janod

LeapFrog

Lego

Little Tikes

Ludorama

Mattel

Mega Bloks

Melissa & Doug

Nerf

Playmobil

Ravensburger

Schleich

Sony

Spin Master

Step2

VTech

Filtres

Mot-clé :

Catégorie :

Marque :

Prix minimum :

Prix maximum :

En stock uniquement :

Rechercher :

-- Toutes les catégories --

- Jouets en Bois
- Jeux de Société
- Figurines
- Puzzles
- Peluches
- Électronique
- Véhicules
- Extérieur
- Créativité
- Construction
- Jeux Vidéo

```
<div class="recherche-resultats">
    <h3>Résultats</h3>
    <div class="produits-grid">
        <?php if (empty($resultats)): ?>
            <p>Aucun produit trouvé.</p>
        <?php else: ?>
            <?php foreach ($resultats as $produit): ?>
                <div class="produit-item">
                    <!-- Générer dynamiquement l'image en fonction de l'ID du
produit -->
                    ">
                </div>
            <?php endforeach; ?>
    </div>
</div>
```

```

        class="produit-image"
        width="100">

        <!-- Afficher les informations du produit -->
        <h4><?= htmlspecialchars($produit['NOMPROD']) ?></h4>
        <p><?= htmlspecialchars($produit['DESCPROD']) ?></p>
        <p>Prix : <?= number_format($produit['PRIXHT'], 2) ?> €</p>
        <p class="<?= $produit['QTESTOCK'] ?> 0 ? 'en-stock' :
        'rupture-stock' ?>">
            <?= $produit['QTESTOCK'] ?> 0 ? 'En stock' : 'Rupture de
            stock' ?>
        </p>

        <!-- Ajouter un bouton pour accéder à la description
détailée -->
        <a href="descProduit.php?idProd=<?=
        htmlspecialchars($produit['IDPROD']) ?>" class="btn-details">Voir les détails</a>
        </div>

        <?php endforeach; ?>
        <?php endif; ?>
    </div>
</div>

```

Cette partie la permet d'afficher les différents produits

Résultats		
 <p>Train en bois</p> <p>Train avec rails en bois</p> <p>Prix : 29.99 €</p> <p>En stock</p> <p>Voir les détails</p>	 <p>Train électrique</p> <p>Train électrique miniature</p> <p>Prix : 49.99 €</p> <p>En stock</p> <p>Voir les détails</p>	 <p>Brio</p> <p>Trains miniatures Brio</p> <p>Prix : 12.99 €</p> <p>En stock</p> <p>Voir les détails</p>

et enfin la partie pagination qui permet de choisir la page que l'on veut

```
<div class="pagination">
    <?php if ($totalPages > 1): ?>

        <?php for ($i = 1; $i <= $totalPages; $i++): ?>
            <a href=<?= htmlspecialchars("recherche.php?page=$i&" .
http_build_query([
                'mot_cle' => $criteres['mot_cle'] ?? NULL,
                'categorie' => isset($criteres['categorie']) &&
$criteres['categorie'] !== '' ? $criteres['categorie'] : NULL,
                'marque' => $criteres['marque'] ?? NULL,
                'prix_min' => $criteres['prix_min'] ?? NULL,
                'prix_max' => $criteres['prix_max'] ?? NULL,
                'en_stock' => $criteres['en_stock'] !== NULL ? 1 : ''
            ])) ?>
                class=<?= $i == $page ? 'active' : '' ?>">
                    <?= $i ?>
                </a>
            <?php endfor; ?>
        <?php else: ?>
            <p>Aucune autre page.</p>
        <?php endif; ?>
    </div>

    <?php include("footer.php") ?>
</body>

</html>
```

Aucune autre page.

Autre exemple avec jeu.

Filtres

Mot-clé : jeu
 Catégorie : Toutes les catégories
 Marque : Toutes les marques
 Prix minimum :
 Prix maximum :
 En stock uniquement

Résultats

 <p>Jeu de mémoire Jeu de société pour mémoire Prix : 14.99 € En stock Voir les détails</p>	 <p>Jeu de société Jeu éducatif sur les animaux Prix : 24.99 € En stock Voir les détails</p>	 <p>Jeu de cartes Jeu de cartes pour enfants Prix : 9.99 € En stock Voir les détails</p>	 <p>Jeu de construction Blocs de construction en bois Prix : 29.99 € En stock Voir les détails</p>	 <p>Jeu de société classique Jeu de société pour toute la famille Prix : 24.99 € En stock Voir les détails</p>	 <p>Jeu de dominos Jeu de dominos en bois Prix : 19.99 € En stock Voir les détails</p>
 <p>Jeu de mémoire Jeu de mémoire pour enfants Prix : 14.99 € En stock Voir les détails</p>	 <p>Jeu de quilles Jeu de quilles en bois Prix : 29.99 € En stock Voir les détails</p>	 <p>Jeu de l'oie Jeu de société classique Prix : 19.99 € En stock Voir les détails</p>	 <p>Console de jeux Console de jeux pour enfants Prix : 59.99 € En stock Voir les détails</p>	 <p>Lego starWars Jeu de construction Lego StarWars Prix : 29.99 € En stock Voir les détails</p>	 <p>Hasbro Game of thrones Jeux de société Hasbro Game of thrones Prix : 69.99 € En stock Voir les détails</p>

1 2 3 4

1

2

3

4

7. La page Compte

7.1 Contexte

La page "compte" est essentielle dans un système de commerce électronique, car elle centralise la gestion des informations personnelles et des préférences de l'utilisateur, qui sont cruciales pour le processus de commande. Elle permet aux utilisateurs de mettre à jour leurs adresses, de visualiser et de gérer leurs commandes, et de modifier leurs moyens de paiement, assurant ainsi que les transactions se déroulent avec les informations les plus actuelles et précises. Ces fonctionnalités garantissent une expérience de checkout fluide et personnalisée, directement liée à la user story prioritaire du processus de commande.

En outre, la page "compte" offre la possibilité de se déconnecter qui contribue à la sécurité en permettant aux utilisateurs de terminer activement leur session, ce qui réduit le risque d'accès non autorisé après que l'utilisateur ai fini d'utiliser le service. La page "compte" joue donc un rôle complémentaire en offrant cette option de déconnexion, tout en centralisant la gestion des informations essentielles qui influencent directement des processus critiques comme les commandes, améliorant ainsi l'efficacité et l'expérience utilisateur globale.

7.2 Modification des informations personnelles

7.2.1 Explication du code

Le formulaire pour modifier les informations personnelles est intégré dans une fenêtre modale. Le formulaire est pré-rempli avec les données actuelles de l'utilisateur, récupérées de la base de données. Lorsque l'utilisateur soumet le formulaire, les données sont envoyées à `updateInfoCli.php` avec un paramètre spécifiant le type de modification (`typeModif=1`).

```
<!-- Fenêtre contextuelle pour modifier les informations client -->
<div id="modal-infoClient" class="modal">
    <div class="modal-content">
        <span class="close">&times;</span>
        <h2>Modifier mes informations</h2>
        <form method="post" action="updateInfoCli.php?typeModif=1">
            <label for="nomCli">Nom :</label>
            <input type="text" id="nomCli" name="nomCli" value=<?php echo "'". htmlspecialchars($user['NOMCLIENT']). "'?>required>
            <label for="prenomCli">Prénom :</label>
            <input type="text" id="prenomCli" name="prenomCli" value=<?php echo "'". htmlspecialchars($user['PRENOMCLIENT']). "'?>required>
            <label for="emailCli">Email :</label>
            <input type="text" id="emailCli" name="emailCli" value=<?php echo "'". htmlspecialchars($user['EMAIL']). "'?>required>
            <label for="dateN">Date de naissance :</label>
            <input type="date" id="dateN" name="dateN" value=<?php echo "'". $user['DATEN']. "'?>required>
            <?php if (!empty($user['NUMTEL'])): ?>
                <label for="telephone-container">Numéro de Téléphone :</label>
                <div class="telephone-container">
                    <select id="indicatif" name="indicatif">
                        <?php
                            $indicatifs = json_decode(file_get_contents('indicatifs.json'), true);
                            // On extrait l'indicatif du numéro existant (les deux premiers chiffres après le 00)
                            $currentIndicatif = !empty($user['NUMTEL']) ? '+' . substr($user['NUMTEL'], 2, 2) : '+33';
                        <?php
                        foreach ($indicatifs as $indicatif) {
                            $selected = ($currentIndicatif === $indicatif['code']) ? 'selected' : '';
                            echo "<option value=\"{$indicatif['code']}\" {$selected}>{$indicatif['emoji']} {$indicatif['pays']}";
                        }
                    </select>
                    <input type="text" id="numTel" name="numTel" value=<?= !empty($user['NUMTEL']) ? substr($user['NUMTEL'], 4 : '' ?>">
                </div>
            <?php else: ?>
                <label for="telephone-container">Numéro de Téléphone :</label>
                <div class="telephone-container">
                    <select id="indicatif" name="indicatif">
```

```

<?php
$indicatifs = json_decode(file_get_contents('indicatifs.json'), true);
foreach ($indicatifs as $indicatif) {
    $selected = ($indicatif['code'] === '+33') ? 'selected' : '';
    // Par défaut, on sélectionne la France
    echo "<option value=\"{$indicatif['code']}\" {$selected}>{$indicatif['emoji']} {$indicatif['pays']}
";
}
?>
</select>
<input type="text" id="numTel" name="numTel">
</div>
<?php endif; ?>
<button type="submit" id="submitBtn-modif-Info-Cli">Modifier</button>
</form>
</div>
</div>

```

dans updateInfoCli.php :

```

// Traitement principal
$success = false;
$error = null;

switch ($_GET["typeModif"]) {
    case 1: // Modification infos personnelles
        $success = updatePersonalInfo($pdo, $id_client, $_POST);
        break;
}

```

Ensuite la fonction updatePersonalInfo() est appelée dans le switch case.

```

function updatePersonalInfo($pdo, $id_client, $data) {
    $numTel = null;
    if (!empty($data["numTel"])) {
        $indicatif = str_replace("+", "00", $data["indicatif"]);
        $numero = $indicatif . $data["numTel"];
        $numTel = strlen($numero) > 11 ? $numero : null;
    }
    $query = $pdo->prepare("
        UPDATE CLIENT
        SET NOMCLIENT = :nomCli,
            PRENOMCLIENT = :prenomCli,
            NUMTEL = :numTel,
            EMAIL = :emailCli,
            DATEN = :dateN
        WHERE IDCLIENT = :idClient
    ");

    $success = $query->execute([
        ":idClient" => $id_client,
        ":nomCli" => $data["nomCli"],
        ":prenomCli" => $data["prenomCli"],
    ]);
}

```

```
":numTel" => $numTel,
":emailCli" => $data["emailCli"],
":dateN" => $data["dateN"]
]);

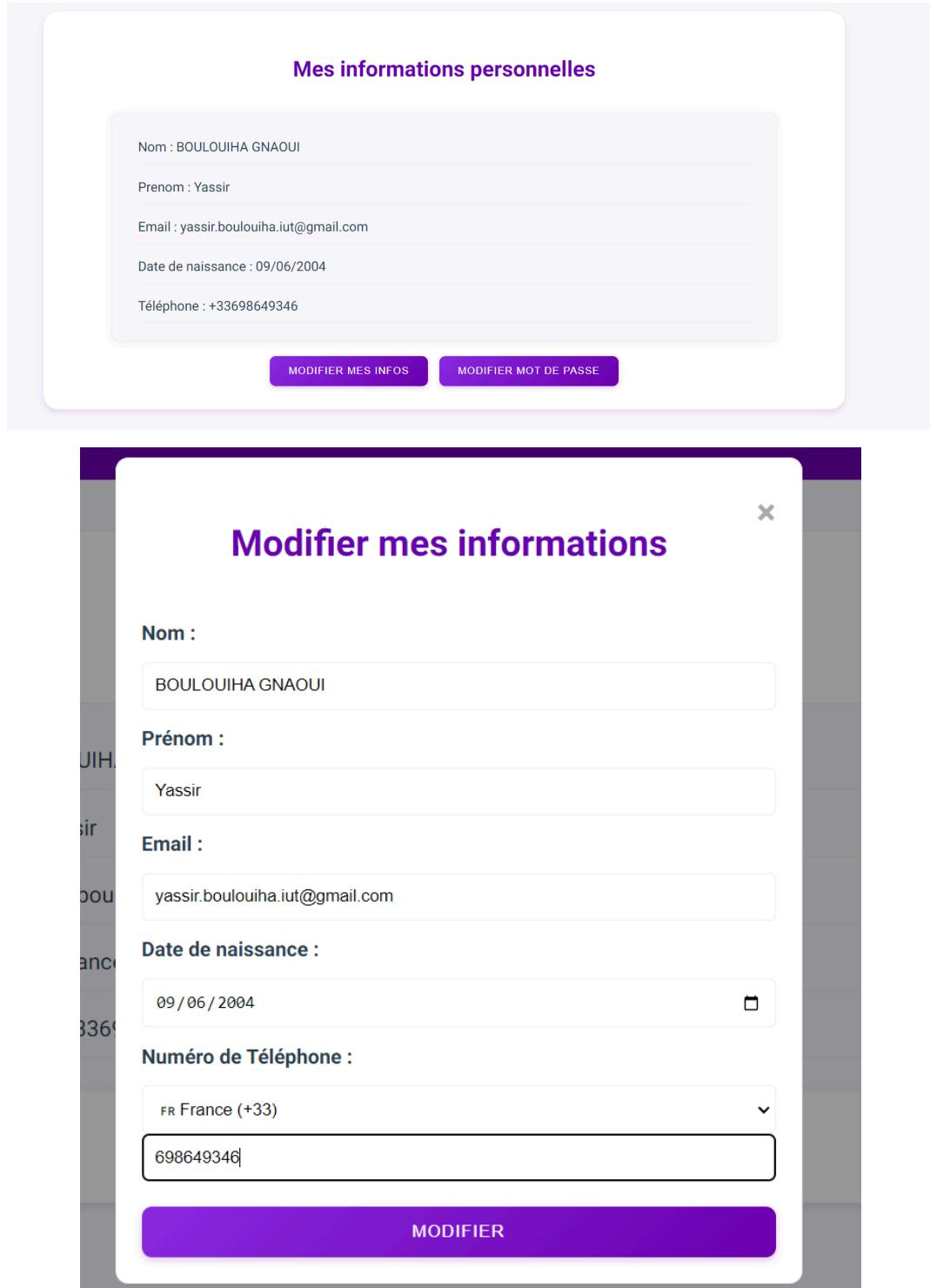
if ($success) {
    $_SESSION["user"]["NOMCLIENT"] = $data["nomCli"];
    $_SESSION["user"]["PRENOMCLIENT"] = $data["prenomCli"];
    $_SESSION["user"]["NUMTEL"] = $numTel;
    $_SESSION["user"]["EMAIL"] = $data["emailCli"];
    $_SESSION["user"]["DATEN"] = $data["dateN"];
}

return $success;
}
```

Le script PHP updatePersonalInfo est conçu pour mettre à jour les informations personnelles d'un client dans une base de données. Voici les étapes clés de son fonctionnement :

1. **Vérification du numéro de téléphone :** Si un numéro de téléphone est fourni, le script remplace le signe "+" par "00" pour l'indicatif international, puis concatène cet indicatif avec le numéro de téléphone. Si la longueur du numéro résultant dépasse 11 caractères, il est conservé ; sinon, il est ignoré (mis à null).
2. **Préparation de la requête SQL :** Une requête SQL est préparée pour mettre à jour les champs spécifiques du client dans la table CLIENT : nom, prénom, numéro de téléphone, email et date de naissance.
3. **Exécution de la requête :** La requête est exécutée avec les données fournies. Si l'exécution est réussie, les informations du client sont également mises à jour dans la session PHP pour refléter les changements.
4. **Retour du résultat :** Le script retourne un booléen indiquant si la mise à jour a été réussie ou non.

7.2.2 Captures d'écran



The image consists of two screenshots of a web application interface.

Screenshot 1: Mes informations personnelles

This screenshot shows a summary of personal information:

- Nom : BOULOUIHA GNAOUI
- Prenom : Yassir
- Email : yassir.boulouih.iut@gmail.com
- Date de naissance : 09/06/2004
- Téléphone : +33698649346

Buttons at the bottom: MODIFIER MES INFOS and MODIFIER MOT DE PASSE.

Screenshot 2: Modifier mes informations

This screenshot shows a modal dialog for editing personal information:

- Nom :** BOULOUIHA GNAOUI
- Prénom :** Yassir
- Email :** yassir.boulouih.iut@gmail.com
- Date de naissance :** 09 / 06 / 2004
- Numéro de Téléphone :** FR France (+33) 698649346

A large purple "MODIFIER" button is at the bottom.

7.3 Modification des informations personnelles

7.3.1 Explication du code

Le formulaire pour modifier le mot de passe est intégré dans une fenêtre modale. L'utilisateur doit entrer son ancien mot de passe et le nouveau. Les données sont postées à updateInfoCli.php avec typeModif=2.

```
<div id="modal-modifier-mdp" class="modal">
    <div class="modal-content">
        <span class="close">&times;</span>
        <h2>Modifier mon Mot de Passe</h2>
        <form method="post" action="updateInfoCli.php?typeModif=2">
            <label for="ancien-mdp">Ancien mot de passe:</label>
            <input type="password" id="ancien-mdp" name="ancien-mdp" required>
            <label for="nouveau-mdp">Nouveau mot de passe:</label>
            <input type="password" id="nouveau-mdp" name="nouveau-mdp" required>
            <label for="ancien-mdp">Confirmer le nouveau mot de passe:</label>
            <input type="password" id="nouveau-mdp2" name="nouveau-mdp2" required>
            <div id="password-message" style="color: red; display: none;"></div>
            <button type="submit" id="changer-mdp">Modifier</button>
        </form>
    </div>
</div>
```

Dans updateInfoCli.php, la modification est traitée par un switch case :

```
case 2: // Modification mot de passe
    $result = updatePassword($pdo, $id_client, $_POST);
    if ($result === 'wrong_password') {
        header("Location:
compte.php?modif=erreur&typeModif=2&error=wrong_password");
        exit();
    }
    $success = $result;
    break;
```

```
function updatePassword($pdo, $id_client, $data) {
    $query = $pdo->prepare("SELECT PASSWORD FROM CLIENT WHERE IDCLIENT = :idClient");
    $query->execute([":idClient" => $id_client]);
    $currentPassword = $query->fetchColumn();

    if (!password_verify($data["ancien-mdp"], $currentPassword)) {
        return 'wrong_password';
    }

    $newPasswordHash = password_hash($data["nouveau-mdp"], PASSWORD_DEFAULT);
    $query = $pdo->prepare(
        "UPDATE CLIENT
         SET PASSWORD = :newPassword
         WHERE IDCLIENT = :idClient
    ");
    return $query->execute([
        ":idClient" => $id_client,
        ":newPassword" => $newPasswordHash
    ]);
}
```

La fonction `updatePassword()` vérifie l'ancien mot de passe avec `password_verify` avant de hasher le nouveau mot de passe avec `password_hash` et de l'insérer dans la base de données.

7.3.2 Captures d'écran

The screenshot shows a modal window titled "Modifier mon Mot de Passe". It contains three input fields: "Ancien mot de passe:", "Nouveau mot de passe:", and "Confirmer le nouveau mot de passe:". Below the fields is a purple "MODIFIER" button. The entire form is set against a dark background.

Ancien mot de passe:

Nouveau mot de passe:

Confirmer le nouveau mot de passe:

MODIFIER

7.4 Gestion des adresses

7.4.1 Explication du code - Ajouter Adresse

Pour ajouter une adresse, l'utilisateur remplit un formulaire dans une fenêtre modale. Les données sont envoyées via POST à updateInfoCli.php avec le paramètre typeModif=3.

```
<!-- Fenêtre contextuelle pour modifier les adresses -->
<div id="modal-ajouter-adresse" class="modal">
    <div class="modal-content">
        <span class="close">&times;</span>
        <h2>Ajouter une adresse</h2>
        <form method="post" action="updateInfoCli.php?typeModif=3">
            <label for="numRue">Numéro de rue :</label>
            <input type="number" id="numRue" name="numRue" required min="1">

            <label for="nomRue">Nom de la rue :</label>
            <input type="text" id="nomRue" name="nomRue" required>

            <label for="complement">Complément d'adresse :</label>
            <input type="text" id="complement" name="complement">

            <label for="ville">Ville :</label>
            <input type="text" id="ville" name="ville" required>

            <label for="codePostal">Code postal :</label>
            <input type="number" id="codePostal" name="codePostal" required min="1" max="99999999" oninput="javascript:if (this.value.length > 8) this.value = this.value.slice(0, 8)">

            <label for="pays">Pays :</label>
            <input type="text" id="pays" name="pays" required>

            <button type="submit" id="submitBtn-ajout-adresse">Ajouter</button>
        </form>
    </div>
</div>
```

Dans updateInfoCli.php, l'ajout est traité par un switch case :

```
case 3: // Ajout adresse
    $success = addAddress($pdo, $id_client, $_POST);
    break;
```

```

function addAddress($pdo, $id_client, $data) {
    try {
        if (!validateAddressData($data)) {
            return false;
        }

        $pdo->beginTransaction();

        $query = $pdo->prepare("
            INSERT INTO ADRESSE (NUMRUE, NOMRUE, COMPLEMENTADR, NOMVILLE, CODEPOSTAL, PAYS)
            VALUES (:numRue, :nomRue, :complement, :ville, :codePostal, :pays)
        ");

        if (!$query->execute([
            ':numRue' => $data['numRue'],
            ':nomRue' => $data['nomRue'],
            ':complement' => !empty($data['complement']) ? $data['complement'] : null,
            ':ville' => $data['ville'],
            ':codePostal' => $data['codePostal'],
            ':pays' => $data['pays']
        ])) {
            throw new Exception();
        }

        $idAdresse = $pdo->lastInsertId();

        $query = $pdo->prepare("
            INSERT INTO POSSEDERADR (IDCLIENT, IDADRESSE)
            VALUES (:idClient, :idAdresse)
        ");

        if (!$query->execute([
            ':idClient' => $id_client,
            ':idAdresse' => $idAdresse
        ])) {
            throw new Exception();
        }

        $pdo->commit();
        return true;
    } catch (Exception $e) {
        if ($pdo->inTransaction()) {
            $pdo->rollBack();
        }
        return false;
    }
}

```

La fonction addAddress() commence par valider les données d'adresse avec validateAddressData(), puis utilise une transaction SQL pour insérer les données dans la table ADRESSE. Si l'insertion réussit, une seconde insertion lie l'adresse au client dans la table POSSEDERADR. En cas d'erreur, la transaction est annulée avec un rollback.

7.4.2 Explication du code Modifier Adresse

```
case 4: // Modification adresse
    if (!isset($_POST['idAddress'])) {
        header("Location: compte.php?modif=erreur&typeModif=4");
        exit();
    }
    $success = updateAddress($pdo, $id_client, $_POST['idAddress'], $_POST);
    break;
```

La modification utilise un formulaire similaire, envoyé avec typeModif=4. Le formulaire inclut l'ID de l'adresse à modifier. Celui-ci est pré-rempli avec les informations de l'adresse à modifier grâce avec ce script javascript qui se base sur les informations html de l'adresse qui devraient a priori être bonnes parce que c'est le script qui les génère, dans le cas où l'utilisateur voudrait modifier ces informations depuis le html client de façon mal-intentionnée une double vérification des champs est faite, en javascript et en php après coup.

```
function updateAddress($pdo, $id_client, $id_address, $data) {
    try {
        if (!validateAddressData($data)) {
            throw new Exception("Données d'adresse invalides");
        }

        // Vérifier que l'adresse appartient bien au client
        $query = $pdo->prepare(
            "SELECT COUNT(*) FROM POSSEDERADR
             WHERE IDCLIENT = :idClient AND IDADRESSE = :idAddress
        ");
    }
```

```

$query->execute([
    ':idClient' => $id_client,
    ':idAddress' => $id_address
]);

if ($query->fetchColumn() == 0) {
    throw new Exception("Cette adresse n'appartient pas à ce client");
}

$query = $pdo->prepare("
    UPDATE ADRESSE
    SET NUMRUE = :numRue,
        NOMRUE = :nomRue,
        COMPLEMENTADR = :complement,
        NOMVILLE = :ville,
        CODEPOSTAL = :codePostal,
        PAYS = :pays
    WHERE IDADRESSE = :idAddress
");

return $query->execute([
    ':idAddress' => $id_address,
    ':numRue' => $data['numRue'],
    ':nomRue' => $data['nomRue'],
    ':complement' => !empty($data['complement']) ? $data['complement'] : null,
    ':ville' => $data['ville'],
    ':codePostal' => $data['codePostal'],
    ':pays' => $data['pays']
]);
} catch (Exception $e) {
    return false;
}
}

```

La fonction `updateAddress()` vérifie d'abord que l'adresse appartient au client avec une requête préparée. Si l'adresse est validée, elle est mise à jour dans la base de données. Les erreurs potentielles déclenchent également un rollback.

7.4.3 Explication du code Supprimer Adresse

```
case 5: // Suppression adresse
    if (!isset($_POST['idAddress'])) {
        header("Location: compte.php?modif=erreur&typeModif=5");
        exit();
    }
    $success = deleteAddress($pdo, $id_client, $_POST['idAddress']);
    break;
```

La suppression est déclenchée par un formulaire envoyé avec typeModif=5, incluant l'ID de l'adresse à supprimer.

```
function deleteAddress($pdo, $id_client, $id_address) {
    try {
        $pdo->beginTransaction();

        // Vérifier que l'adresse appartient bien au client
        $query = $pdo->prepare(
            "SELECT COUNT(*) FROM POSSEDERADR
             WHERE IDCLIENT = :idClient AND IDADRESSE = :idAddress
        ");
        $query->execute([
            ':idClient' => $id_client,
            ':idAddress' => $id_address
        ]);

        if ($query->fetchColumn() == 0) {
            throw new Exception("Cette adresse n'appartient pas à ce client");
        }
        // Supprimer d'abord la liaison
        $query = $pdo->prepare(
            "DELETE FROM POSSEDERADR
             WHERE IDCLIENT = :idClient AND IDADRESSE = :idAddress
        ");
    }
```

```

if (!$query->execute([
    ':idClient' => $id_client,
    ':idAddress' => $id_address
])) {
    throw new Exception("Erreur lors de la suppression de la liaison");
}

// Puis supprimer l'adresse
$query = $pdo->prepare(
    "DELETE FROM ADRESSE
     WHERE IDADRESSE = :idAddress
");

if (!$query->execute([':idAddress' => $id_address])) {
    throw new Exception("Erreur lors de la suppression de l'adresse");
}

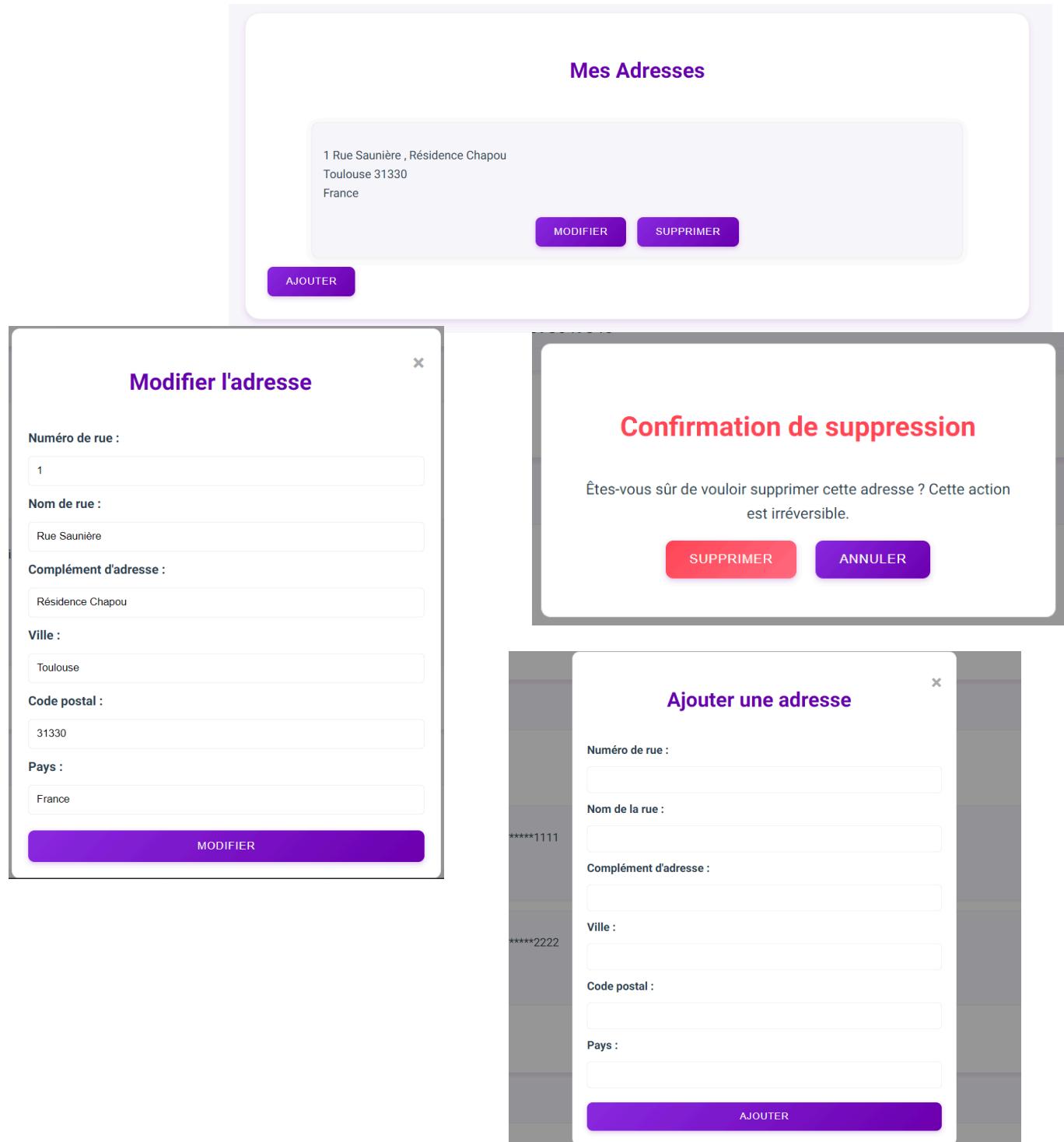
$pdo->commit();
return true;

} catch (Exception $e) {
    if ($pdo->inTransaction()) {
        $pdo->rollBack();
    }
    return false;
}
}

```

La fonction `deleteAddress()` utilise une transaction pour d'abord supprimer la liaison entre l'adresse et le client dans `POSSEDERADR`, puis supprime l'adresse elle-même dans `ADRESSE`. Comme pour les autres opérations, un rollback est utilisé en cas d'erreur pour maintenir l'intégrité des données.

7.4.4 Captures d'écran



The image displays four screenshots of a web application interface for managing addresses:

- Mes Adresses**: A list view showing one address: "1 Rue Saunière , Résidence Chapou Toulouse 31330 France". It includes "MODIFIER" and "SUPPRIMER" buttons.
- Modifier l'adresse**: A form for editing an address. Fields include: Numéro de rue (1), Nom de rue (Rue Saunière), Complément d'adresse (Résidence Chapou), Ville (Toulouse), Code postal (31330), and Pays (France). A "MODIFIER" button is at the bottom.
- Confirmation de suppression**: A confirmation dialog asking "Êtes-vous sûr de vouloir supprimer cette adresse ? Cette action est irréversible." with "SUPPRIMER" and "ANNULER" buttons.
- Ajouter une adresse**: A form for adding a new address. Fields include: Numéro de rue, Nom de la rue (*****1111), Complément d'adresse, Ville (*****2222), Code postal, and Pays. An "AJOUTER" button is at the bottom.

7.5 Gestion des cartes bancaires

7.5.1 Explication du code Ajout d'une carte

```
<!-- Fenêtre contextuelle pour ajouter une carte -->
<div id="modal-ajouter-carte" class="modal">
    <div class="modal-content">
        <span class="close">&times;</span>
        <h2>Ajouter une carte bancaire</h2>
        <form method="post" action="updateInfoCli.php?typeModif=6">
            <label for="numCB">Numéro de carte :</label>
            <input type="text" id="numCB" name="numCB" required maxlength="19"
placeholder="XXXX XXXX XXXX XXXX">

            <label for="nomCompletCB">Nom sur la carte :</label>
            <input type="text" id="nomCompletCB" name="nomCompletCB" required maxlength="50">

            <label for="dateExp">Date d'expiration :</label>
            <input type="month" id="dateExp" name="dateExp" required>

            <label for="cryptogramme">Cryptogramme :</label>
            <input type="password" id="cryptogramme" name="cryptogramme" required
maxlength="3" pattern="[0-9]{3}">

            <button type="submit" id="submitBtn-ajout-carte">Ajouter</button>
        </form>
    </div>
</div>
```

Pour ajouter une carte bancaire, un formulaire dans une fenêtre modale est utilisé, envoyé avec typeModif=6.

```
case 6: // Ajout carte bancaire
    $success = addPaymentCard($pdo, $id_client, $_POST);
    if (!$success && isset($_SESSION['card_error']) && $_SESSION['card_error'] ===
'wrong_holder') {
        unset($_SESSION['card_error']);
        header("Location: compte.php?modif=erreur&typeModif=6&error=wrong_holder");
        exit();
    }
    break
```

La fonction `addPaymentCard()` effectue plusieurs vérifications et opérations :

1. **Validation des données de la carte :** Les données telles que le numéro de la carte, le nom du titulaire, le cryptogramme, et la date d'expiration sont validées pour s'assurer qu'elles respectent les formats et critères attendus.
2. **Vérification de l'existence de la carte :** Avant d'ajouter une nouvelle carte, le système vérifie si le numéro de la carte existe déjà dans la base de données.
3. **Comparaison des noms des titulaires :** Si la carte existe déjà, le système compare le nom du titulaire enregistré avec celui fourni dans le formulaire. Cette étape est cruciale pour s'assurer que la carte n'est pas utilisée frauduleusement par une autre personne. Si les noms ne correspondent pas, la transaction est annulée, et une erreur est retournée à l'utilisateur.
4. **Insertion de la carte :** Si la carte n'existe pas déjà ou si les noms correspondent, les informations de la carte sont insérées dans la table `INFORMATIONPAIEMENT`.
5. **Liaison de la carte au client :** Après l'insertion réussie de la carte, une liaison est créée entre le numéro de la carte et l'ID du client dans la table `POSSEDERIP` pour associer la carte au compte utilisateur.

```
function addPaymentCard($pdo, $id_client, $data) {
    try {
        if (!validateCardData($data)) {
            return false;
        }
        $pdo->beginTransaction();
        // Nettoyage du numéro de carte et du nom
        $numCB = str_replace(' ', '', $data['numCB']);
        $nomCompletCB = strtoupper(trim($data['nomCompletCB']));
        // Vérifier si la carte existe déjà
        $query = $pdo->prepare("
            SELECT NOMCOMPLETCB
            FROM INFORMATIONPAIEMENT
            WHERE NUMCB = :numCB
        ");
        $query->execute([':numCB' => $numCB]);
        $result = $query->fetch();
        if ($result) {
            return false;
        }
        $query = $pdo->prepare("INSERT INTO INFORMATIONPAIEMENT (NUMCB, NOMB, CRYPT, DATEEXPIRATION, IDCLIENT) VALUES (:numCB, :nomCB, :crypt, :dateExpiration, :idClient)");
        $query->execute([
            ':numCB' => $numCB,
            ':nomCB' => $nomCompletCB,
            ':crypt' => $data['crypt'],
            ':dateExpiration' => $data['dateExpiration'],
            ':idClient' => $id_client
        ]);
        $pdo->commit();
        return true;
    } catch (Exception $e) {
        $pdo->rollBack();
        return false;
    }
}
```

```

$query->execute([':numCB' => $numCB]);
$existingCard = $query->fetch();

if ($existingCard) {
    // La carte existe déjà, comparons les noms
    $existingName = strtoupper(trim($existingCard['NOMCOMPLETCB']));

    if ($existingName === $nomCompletCB) {
        // Les noms correspondent, on ajoute juste la liaison
        $query = $pdo->prepare("
            INSERT INTO POSSEDERIP (NUMCB, IDCLIENT)
            VALUES (:numCB, :idClient)
        ");

        if (!$query->execute([
            ':numCB' => $numCB,
            ':idClient' => $id_client
        ])) {
            throw new Exception("Erreur lors de la liaison de la carte au client");
        }

        $pdo->commit();
        return true;
    } else {
        // Les noms ne correspondent pas
        $pdo->rollBack();
        $_SESSION['card_error'] = 'wrong_holder';
        return false;
    }
}

// La carte n'existe pas, on l'ajoute normalement
$dateExp = date('Y-m-t', strtotime($data['dateExp'] . '-01'));

$query = $pdo->prepare("
    INSERT INTO INFORMATIONPAIEMENT (NUMCB, NOMCOMPLETCB, DATEEXP, CRYPTOGRAMME)
    VALUES (:numCB, :nomCompletCB, :dateExp, :cryptogramme)
");

if (!$query->execute([
    ':numCB' => $numCB,
    ':nomCompletCB' => $nomCompletCB,
    ':dateExp' => $dateExp,
    ':cryptogramme' => $data['cryptogramme']
])) {
    throw new Exception("Erreur lors de l'ajout de la carte");
}

```

```

// Liaison avec le client
$query = $pdo->prepare("
    INSERT INTO POSSEDERIP (NUMCB, IDCLIENT)
    VALUES (:numCB, :idClient)
");

if (!$query->execute([
    ':numCB' => $numCB,
    ':idClient' => $id_client
])) {
    throw new Exception("Erreur lors de la liaison de la carte au client");
}

$pdo->commit();
return true;

} catch (Exception $e) {
    if ($pdo->inTransaction()) {
        $pdo->rollBack();
    }
    return false;
}
}

```

7.5.2 Explication du code Suppression d'une carte

La suppression d'une carte utilise typeModif=7, similaire à la suppression d'une adresse, mais pour les cartes bancaires.

```

case 7: // Suppression carte
    if (!isset($_POST['numCB'])) {
        header("Location: compte.php?modif=erreur&typeModif=7");
        exit();
    }
    $success = deletePaymentCard($pdo, $id_client, $_POST['numCB']);
    break;
}

```

La fonction `deletePaymentCard()` :

1. Vérifie que la carte appartient au client.
2. Supprime la carte de `INFORMATIONPAIEMENT` et la liaison dans `POSSEDERIP`.

```
function deletePaymentCard($pdo, $id_client, $numCB) {
    try {
        $pdo->beginTransaction();

        // Vérifier que la carte appartient bien au client
        $query = $pdo->prepare("SELECT COUNT(*) FROM POSSEDERIP
                               WHERE IDCLIENT = :idClient AND NUMCB = :numCB");
        $query->execute([
            ':idClient' => $id_client,
            ':numCB' => $numCB
        ]);

        if ($query->fetchColumn() == 0) {
            throw new Exception("Cette carte n'appartient pas à ce client");
        }

        // Supprimer d'abord la liaison
        $query = $pdo->prepare("DELETE FROM POSSEDERIP
                               WHERE IDCLIENT = :idClient AND NUMCB = :numCB");
        $query->execute([
            ':idClient' => $id_client,
            ':numCB' => $numCB
        ]);

        if (!$query->execute([
            ':idClient' => $id_client,
            ':numCB' => $numCB
        ])) {
            throw new Exception("Erreur lors de la suppression de la liaison");
        }

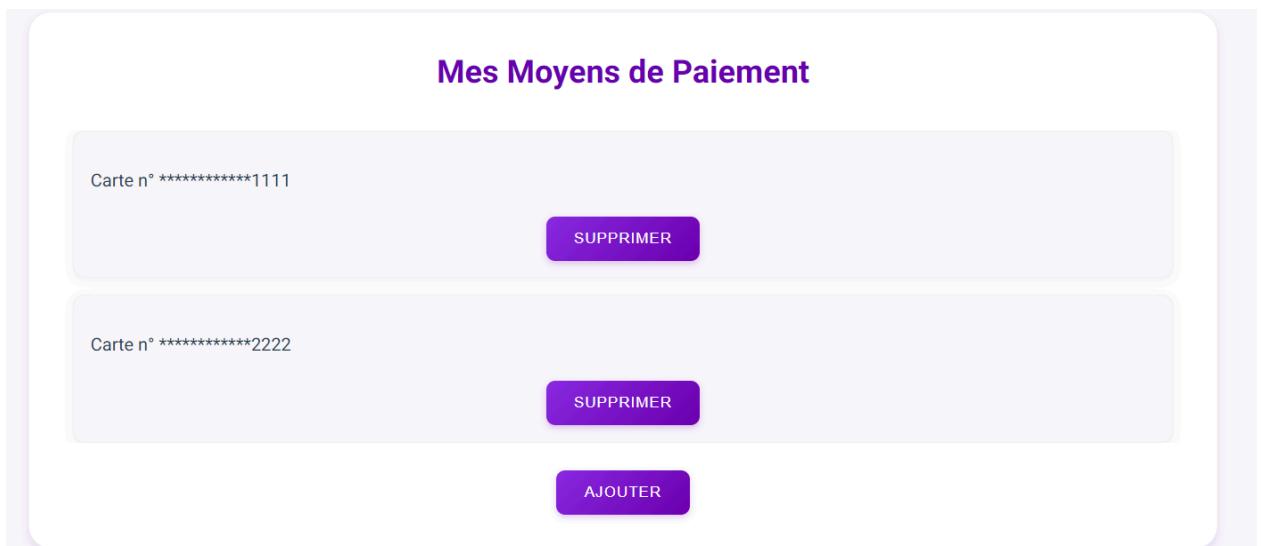
        // Puis supprimer la carte
        $query = $pdo->prepare("DELETE FROM INFORMATIONPAIEMENT
                               WHERE NUMCB = :numCB");
        $query->execute([
            ':numCB' => $numCB
        ]);

        if (!$query->execute([
            ':numCB' => $numCB
        ])) {
            throw new Exception("Erreur lors de la suppression de la carte");
        }
    }

    $pdo->commit();
    return true;
}
```

```
        } catch (Exception $e) {
            if ($pdo->inTransaction()) {
                $pdo->rollBack();
            }
            return false;
        }
    }
```

7.5.3 Captures d'écran



Ajouter une carte bancaire

Numéro de carte :

Nom sur la carte :

Date d'expiration :

 /

Cryptogramme :

AJOUTER



7.6 Affichage des commandes

7.6.1 Explication du code

L'affichage des commandes utilise une requête SQL complexe qui joint plusieurs tables (COMMANDE, TRANSPORTEUR, PANIER, PRODUIT) pour récupérer toutes les informations nécessaires :

```
$query = $pdo->prepare("
    SELECT
        C.*,
        T.TYPEEXP, T.FRAISEXP, T.FRAISKG, T.DELAILIVRAISON,
        P.QUANTITEPROD,
        PR.IDPROD, PR.NOMPROD, PR.PRIXHT, PR.COMPOSITION, PR.COULEUR
    FROM COMMANDE C, TRANSPORTEUR T, PANIER P, PRODUIT PR
    WHERE C.IDTRANSPORTEUR = T.IDTRANSPORTEUR
    AND P.IDCOMMANDE = C.NUMCOMMANDE
    AND P.IDCLIENT = C.IDCLIENT
    AND P.IDPROD = PR.IDPROD
    AND P.IDCOMMANDE != 0
    AND C.IDCLIENT = :idClient
    ORDER BY C.DATECOMMANDE DESC
");
```

Les données sont ensuite organisées dans un tableau associatif \$commandes où chaque commande contient :

Informations générales (date, statut, code de suivi, type de règlement)

Informations de transport (type, frais, délai)

Liste des produits (nom, quantité, prix)

```
$query->execute(['idClient' => $id_client]);
$commandes = [];

// Organiser les données
while($row = $query->fetch()) {
    if (!isset($commandes[$row['NUMCOMMANDE']])){
        $commandes[$row['NUMCOMMANDE']] = [
            'date' => new DateTime($row['DATECOMMANDE']),
            'statut' => $row['STATUTLIVRAISON'],
            'suivi' => $row['CODESUIVI'],
            'reglement' => $row['TYPEREGLEMENT'],
            'transport' => [
                'type' => $row['TYPEEXP'],
                'frais' => $row['FRAISEXP'],
                'delai' => $row['DELAILIVRAISON']
            ],
            'produits' => []
        ];
    }
}
```

Chaque produit de la commande est ajouté à la liste des produits avec son ID, permettant un lien direct vers la page du produit :

```
$commandes[$row['NUMCOMMANDE']]['produits'][] = [
    'idprod' => $row['IDPROD'],
    'nom' => $row['NOMPROD'],
    'quantite' => $row['QUANTITEPROD'],
    'prix' => $row['PRIXHT']
];
}
```

L'affichage est ensuite réalisé en parcourant ce tableau structuré, avec des liens cliquables vers les produits et un affichage conditionnel du code de suivi.

```

if (count($commandes) > 0):
    foreach($commandes as $numCommande => $commande): ?>
        <div class="commande-item">
            <div class="commande-header">
                <div class="commande-info">
                    <h3>Commande n°<?= $numCommande ?></h3>
                    <p class="date">Du <?= $commande['date']->format('d/m/Y') ?></p>
                </div>
                <div class="commande-status <?= strtolower($commande['statut']) ?>">
                    <?= $commande['statut'] ?>
                </div>
            </div>

            <div class="commande-details">
                <div class="produits-list">
                    <?php foreach($commande['produits'] as $produit): ?>
                        <div class="produit-item">
                            <a href="descProduit.php?idProd=<?= $produit['idprod'] ?>">
                                <?= $produit['nom'] ?>
                            </a>
                            <span class="produit-quantite"><?= $produit['quantite'] ?></span>
                            <span class="produit-prix"><?= number_format($produit['prix'] * $produit['quantite'], 2) ?> €</span>
                        </div>
                    <?php endforeach; ?>
                </div>
            </div>

            <div class="livraison-info">
                <p>Mode de livraison : <?= $commande['transport'][['type']] ?></p>
                <p>Délai estimé : <?= $commande['transport'][['delai']] ?> jours</p>
                <?php if ($commande['suivi']): ?>
                    <p>Code de suivi : <?= $commande['suivi'] ?></p>
                <?php endif; ?>
            </div>
        </div>
    <?php endforeach; ?>
    <div class="pagination">
        <div class="pagination-numbers"></div>
    </div>
    <?php else: ?>
        <p class="no-commandes">Vous n'avez pas encore passé de commande.</p>
    <?php endif; ?>

```

Il y a un système de pagination qui est géré côté client en JavaScript. Les commandes sont d'abord toutes chargées dans le HTML avec la classe commande-item, puis le JavaScript gère leur affichage :

```
<div class="commande-list">
    <!-- Toutes les commandes sont chargées ici -->
</div>
<div class="pagination">
    <div class="pagination-numbers"></div>
</div>
```

Le JavaScript initialise la pagination en :

1. Récupérant toutes les commandes
2. Calculant le nombre de pages nécessaires
3. Crément les boutons de pagination
4. Affichant uniquement les commandes de la première page

```
document.addEventListener('DOMContentLoaded', () => {
    const commandeItems = document.querySelectorAll('.commande-item');
    const commandesParPage = 2;
    let pageCourante = 1;

    // Calcul du nombre total de pages
    const nombrePages = Math.ceil(commandeItems.length / commandesParPage);

    // Création des boutons de pagination
    const paginationContainer = document.querySelector('.pagination-numbers');
    for (let i = 1; i <= nombrePages; i++) {
        const button = document.createElement('button');
        button.className = 'page-number';
        button.textContent = i;
        button.addEventListener('click', () => {
            pageCourante = i;
            afficherCommandes(i);
        });
        paginationContainer.appendChild(button);
    }
}
```

```
// Fonction pour afficher les commandes de la page sélectionnée
function afficherCommandes(page) {
    const debut = (page - 1) * commandesParPage;
    const fin = debut + commandesParPage;

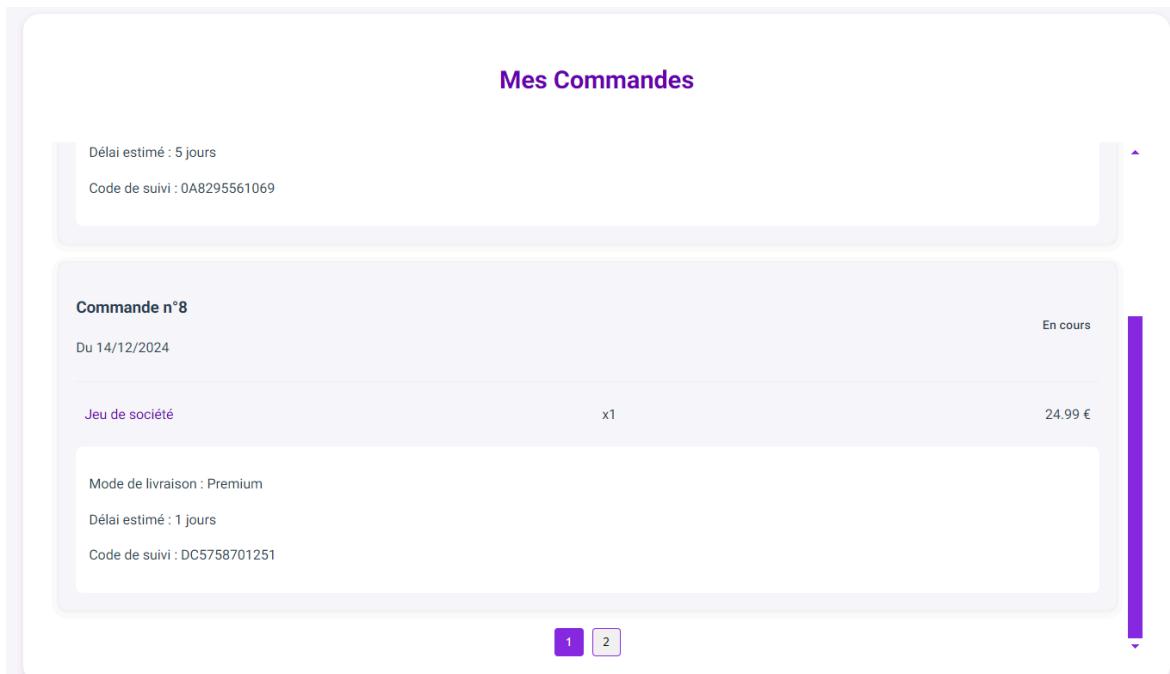
    commandeItems.forEach((item, index) => {
        item.style.display = (index >= debut && index < fin) ? 'block' : 'none';
    });

    // Mise à jour du style du bouton de page active
    document.querySelectorAll('.page-number').forEach(btn => {
        btn.classList.toggle('active', parseInt(btn.textContent) === page);
    });
}

// Affichage initial des commandes
if (commandeItems.length > 0) {
    afficherCommandes(1);
}
});
```

Cette approche permet une navigation fluide entre les pages sans recharge du site, tout en limitant le nombre de commandes affichées simultanément pour une meilleure lisibilité.

7.6.3 Captures d'écran



Mes Commandes

Délai estimé : 5 jours
Code de suivi : OA8295561069

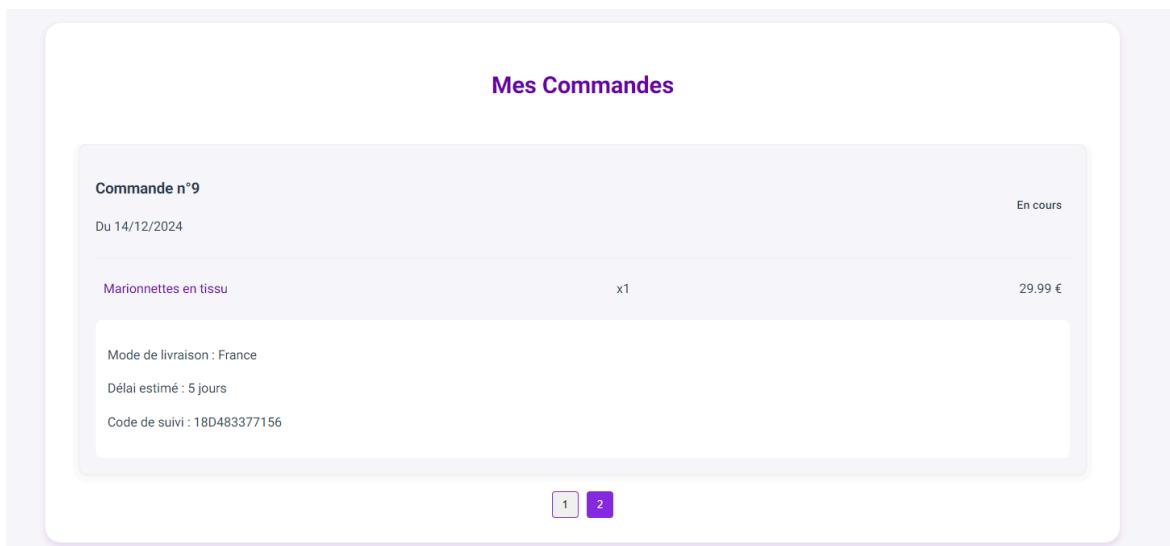
Commande n°8

Du 14/12/2024 En cours

Jeu de société	x1	24.99 €
----------------	----	---------

Mode de livraison : Premium
Délai estimé : 1 jours
Code de suivi : DC5758701251

1 2



Mes Commandes

Commande n°9

Du 14/12/2024 En cours

Marionnettes en tissu	x1	29.99 €
-----------------------	----	---------

Mode de livraison : France
Délai estimé : 5 jours
Code de suivi : 18D483377156

1 2

Conclusion

Nous avons donc pu voir notre système d'authentification sécurisé pour le site e-commerce **Ludorama**. On a respecté les bonnes pratiques pour atteindre les objectifs, notamment :

1. **Sécurisation des données des utilisateurs** : Les mots de passe sont cryptés avant d'être enregistrés dans la base de données, ce qui les protège même si quelqu'un accède aux données.
2. **Gestion des utilisateurs** : Le système permet d'avoir plusieurs utilisateurs avec des droits différents (par exemple, un client ou un administrateur).
3. **Sessions et cookies sécurisés** : Les sessions permettent de garder les utilisateurs connectés, et les cookies peuvent se souvenir des connexions tout en étant protégés contre les attaques.

L'interface est simple et facile à utiliser, avec des fonctionnalités comme l'inscription, la connexion, et la gestion de son compte. Les tests réalisés montrent que tout fonctionne comme prévu dans plusieurs situations.