Să se creeze o aplicație ce conține pagina Home ce conține ultimele produse adăugate in coș. Pagina Produse ce contine toate produsele. La click pe un produs din pagina produse se lanseaza o pagina numita product ce contine detalii despre produsul ce urmeaza să se adauge in coș. Pagina cos cumparaturi ce contine cod sursa destinat gestiunii cosului de cumparaturi.

Stuctură aplicație

functions.php - Acest fișier va conține toate funcțiile de care avem nevoie pentru sistemul nostru de coș de cumpărături (antet șablon, subsol șablon și funcții de conectare la baza de date).

index.php - Acest fișier va conține șablonul principal (antet, subsol etc.) și rutare de bază, astfel încât să putem include paginile de mai jos.

home.php - Acest fișier va fi pagina de pornire care va conține o imagine prezentată și 4 produse adăugate recent.

products.php - Acest fișier va fi pentru afișarea tuturor produselor cu paginare de bază.

product.php - Acest fișier va afișa un produs (depinde de solicitarea GET) și va conține un formular care va permite clientului să schimbe cantitatea și să adauge în coș produsul.

cart.php - Pagina coșului de cumpărături care va lista toate produsele care au fost adăugate în coș, împreună cu cantitățile, prețurile totale și prețul subtotal.

placeorder.php - Un mesaj de bază care va fi afișat utilizatorului atunci când plasează o comandă.

style.css - Foaia de stil (CSS3) pe care o vom folosi pentru site-ul nostru al coșului de cumpărături.

imgs - Director care va conține toate imaginile pentru sistemul nostru de coș de cumpărături (imagini prezentate, imagini de produs etc.). Puteți descărca exemplele de imagini făcând clic pe numele fișierului din containerul structurii de fișiere.

Baza de date o numim: magazin

Tabea produse

id - ID unic pentru produs, acesta va fi incrementat automat.

nume - numele produsului.

desc - Descrierea produsului.

preț - Prețul produsului.

rrp - Prețul cu amănuntul, dacă doriți un produs la vânzare, puneți valoarea mai mare decât prețul.

data_added - Data la care produsul a fost adăugat, îl vom folosi pentru sortarea produselor.

Instrucțiunea SQL va introduce, de asemenea, 4 exemple de produse care vor fi utilizate în scopuri de testare. Le puteți schimba / elimina mai târziu.

```
CREATE TABLE IF NOT EXISTS `products` (
	`id` int(11) NOT NULL AUTO_INCREMENT,
	`name` varchar(200) NOT NULL,
	`desc` text NOT NULL,
	`price` decimal(7,2) NOT NULL,
	`rrp` decimal(7,2) NOT NULL DEFAULT '0.00',
	`quantity` int(11) NOT NULL,
	`img` text NOT NULL,
	`date_added` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
	PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8;

INSERT INTO `products` (`id`, `name`, `desc`, `price`, `rrp`, `quantity`, `img`, `date_added`) VALUES
(1, 'Smart Watch', '<p>Unique watch made with stainless steel, ideal for those that prefer interative watches.</p>\r\n<h3>Features</h3>\r\n<ul>\r\n<li>Powered by Android with built-in apps.</li>\r\n<li>Adjustable to fit most.</li>\r\n<li>Long battery life, continuous wear for up to 2 days.</li>\r\n<li>Lightweight design, comfort on your wrist.</li>\r\n</ul>', '29.99', '0.00', 10, 'watch.jpg', '2019-03-13 17:55:22'),
(2, 'Wallet', '', '14.99', '19.99', 34, 'wallet.jpg', '2019-03-13 18:52:49'),
(3, 'Headphones', '', '19.99', '0.00', 23, 'headphones.jpg', '2019-03-13 18:47:56'),
(4, 'Digital Camera', '', '69.99', '0.00', 7, 'camera.jpg', '2019-03-13 17:42:04');
```

Pas3. Design cos cu css

Pag style.css

```
* {
	box-sizing: border-box;
	font-family: -apple-system, BlinkMacSystemFont, "segoe ui", roboto, oxygen, ubuntu, cantarell, "fira sans", "droid sans", "helvetica neue", Arial, sans-serif;
	font-size: 16px;
	-webkit-font-smoothing: antialiased;
	-moz-osx-font-smoothing: grayscale;
}
html {
	height: 100%;
}
body {
	position: relative;
	min-height: 100%;
	color: #555555;
	background-color: #FFFFFF;
	margin: 0;
	padding-bottom: 100px; /* Same height as footer */
}
h1, h2, h3, h4, h5 {
	color: #394352;
}
```

```
.content-wrapper {
	width: 1050px;
	margin: 0 auto;
}
header {
	border-bottom: 1px solid #EEEEEE;
}
header .content-wrapper {
	display: flex;
}
header h1 {
	display: flex;
	flex-grow: 1;
	flex-basis: 0;
	font-size: 20px;
	margin: 0;
	padding: 24px 0;
}
header nav {
	display: flex;
	flex-grow: 1;
	flex-basis: 0;
	justify-content: center;
	align-items: center;
}
header nav a {
	text-decoration: none;
	color: #555555;
	padding: 10px 10px;
	margin: 0 10px;
}
header nav a:hover {
	border-bottom: 1px solid #aaa;
}
header .link-icons {
	display: flex;
	flex-grow: 1;
	flex-basis: 0;
	justify-content: flex-end;
	align-items: center;
	position: relative;
}
header .link-icons a {
	text-decoration: none;
	color: #394352;
	padding: 0 10px;
}
header .link-icons a:hover {
```

```
        color: #4e5c70;
}
header .link-icons a i {
        font-size: 18px;
}
header .link-icons a span {
        display: inline-block;
        text-align: center;
        background-color: #63748e;
        border-radius: 50%;
        color: #FFFFFF;
        font-size: 12px;
        line-height: 16px;
        width: 16px;
        height: 16px;
        font-weight: bold;
        position: absolute;
        top: 22px;
        right: 0;
}
main .featured {
        display: flex;
        flex-direction: column;
        background-image: url(imgs/featured-image.jpg);
        background-repeat: no-repeat;
        background-size: cover;
        height: 500px;
        align-items: center;
        justify-content: center;
        text-align: center;
}
main .featured h2 {
        display: inline-block;
        margin: 0;
        width: 1050px;
        font-family: Rockwell, Courier Bold, Courier, Georgia, Times, Times New Roman, serif;
        font-size: 68px;
        color: #FFFFFF;
        padding-bottom: 10px;
}
main .featured p {
        display: inline-block;
        margin: 0;
        width: 1050px;
        font-size: 24px;
        color: #FFFFFF;
}
main .recentlyadded h2 {
```

```
        display: block;
        font-weight: normal;
        margin: 0;
        padding: 40px 0;
        font-size: 24px;
        text-align: center;
        width: 100%;
        border-bottom: 1px solid #EEEEEE;
}
main .recentlyadded .products, main .products .products-wrapper {
        display: flex;
        flex-wrap: wrap;
        align-items: center;
        justify-content: space-between;
        padding: 40px 0 0 0;
}
main .recentlyadded .products .product, main .products .products-wrapper .product {
        display: block;
        overflow: hidden;
        text-decoration: none;
        width: 25%;
        padding-bottom: 60px;
}
main .recentlyadded .products .product img, main .products .products-wrapper .product img {
        transform: scale(1);
        transition: transform 1s;
}
main .recentlyadded .products .product .name, main .products .products-wrapper .product .name {
        display: block;
        color: #555555;
        padding: 20px 0 2px 0;
}
main .recentlyadded .products .product .price, main .products .products-wrapper .product .price {
        display: block;
        color: #999999;
}
main .recentlyadded .products .product .rrp, main .products .products-wrapper .product .rrp {
        color: #BBBBBB;
        text-decoration: line-through;
}
main .recentlyadded .products .product:hover img, main .products .products-wrapper .product:hover
img {
        transform: scale(1.05);
        transition: transform 1s;
}
main .recentlyadded .products .product:hover .name, main .products .products-wrapper
.product:hover .name {
        text-decoration: underline;
```

```
}
main > .product {
	display: flex;
	padding: 40px 0;
}
main > .product > div {
	padding-left: 15px;
}
main > .product h1 {
	font-size: 34px;
	font-weight: normal;
	margin: 0;
	padding: 20px 0 10px 0;
}
main > .product .price {
	display: block;
	font-size: 22px;
	color: #999999;
}
main > .product .rrp {
	color: #BBBBBB;
	text-decoration: line-through;
	font-size: 22px;
	padding-left: 5px;
}
main > .product form {
	display: flex;
	flex-flow: column;
	margin: 40px 0;
}
main > .product form input[type="number"] {
	width: 400px;
	padding: 10px;
	margin-bottom: 15px;
	border: 1px solid #ccc;
	color: #555555;
	border-radius: 5px;
}
main > .product form input[type="submit"] {
	background: #4e5c70;
	border: 0;
	color: #FFFFFF;
	width: 400px;
	padding: 12px 0;
	text-transform: uppercase;
	font-size: 14px;
	font-weight: bold;
	border-radius: 5px;
```

```
        cursor: pointer;
}
main > .product form input[type="submit"]:hover {
        background: #434f61;
}
main > .products h1 {
        display: block;
        font-weight: normal;
        margin: 0;
        padding: 40px 0;
        font-size: 24px;
        text-align: center;
        width: 100%;
}
main > .products .buttons {
        text-align: right;
        padding-bottom: 40px;
}
main > .products .buttons a {
        display: inline-block;
        text-decoration: none;
        margin-left: 5px;
        padding: 12px 20px;
        border: 0;
        background: #4e5c70;
        color: #FFFFFF;
        font-size: 14px;
        font-weight: bold;
        border-radius: 5px;
}
main > .products .buttons a:hover {
        background: #434f61;
}
main .cart h1 {
        display: block;
        font-weight: normal;
        margin: 0;
        padding: 40px 0;
        font-size: 24px;
        text-align: center;
        width: 100%;
}
main .cart table {
        width: 100%;
}
main .cart table thead td {
        padding: 30px 0;
        border-bottom: 1px solid #EEEEEE;
```

```
}
main .cart table thead td:last-child {
	text-align: right;
}
main .cart table tbody td {
	padding: 20px 0;
	border-bottom: 1px solid #EEEEEE;
}
main .cart table tbody td:last-child {
	text-align: right;
}
main .cart table .img {
	width: 80px;
}
main .cart table .remove {
	color: #777777;
	font-size: 12px;
	padding-top: 3px;
}
main .cart table .remove:hover {
	text-decoration: underline;
}
main .cart table .price {
	color: #999999;
}
main .cart table a {
	text-decoration: none;
	color: #555555;
}
main .cart table input[type="number"] {
	width: 68px;
	padding: 10px;
	border: 1px solid #ccc;
	color: #555555;
	border-radius: 5px;
}
main .cart .subtotal {
	text-align: right;
	padding: 40px 0;
}
main .cart .subtotal .text {
	padding-right: 40px;
	font-size: 18px;
}
main .cart .subtotal .price {
	font-size: 18px;
	color: #999999;
}
```

```
main .cart .buttons {
	text-align: right;
	padding-bottom: 40px;
}
main .cart .buttons input[type="submit"] {
	margin-left: 5px;
	padding: 12px 20px;
	border: 0;
	background: #4e5c70;
	color: #FFFFFF;
	font-size: 14px;
	font-weight: bold;
	cursor: pointer;
	border-radius: 5px;
}
main .cart .buttons input[type="submit"]:hover {
	background: #434f61;
}
main .placeorder h1 {
	display: block;
	font-weight: normal;
	margin: 0;
	padding: 40px 0;
	font-size: 24px;
	text-align: center;
	width: 100%;
}
main .placeorder p {
	text-align: center;
}
footer {
	position: absolute;
	bottom: 0;
	border-top: 1px solid #EEEEEE;
	padding: 20px 0;
	width: 100%;
}
```

Pas4. Fișierul conectare.php conține toate funcțiile pe care le vom folosi în construirea coșului de cumpărături, acesta include antetul șablonului, subsolul șablonului și funcțiile de conectare la baza de date.

conectare.php :

```
<?php
function pdo_connect_mysql() {
   // Actualizați detaliile de mai jos cu detaliile dvs. MySQL
   $DATABASE_HOST = 'localhost';
   $DATABASE_USER = 'root';
   $DATABASE_PASS = '';
   $DATABASE_NAME = 'magazin';
   try {
        return new PDO('mysql:host=' . $DATABASE_HOST . ';dbname=' . $DATABASE_NAME . ';charset=utf8', $DATABASE_USER, $DATABASE_PASS);
   } catch (PDOException $exception) {
        //  Dacă există o eroare la conexiune, opriți scriptul și afișați eroarea.
        exit('Failed to connect to database!');
   }
}
// Template
function template_header($title) {
// Obțineți cantitatea de articole din coșul de cumpărături, aceasta va fi afișată în antet.
$num_items_in_cart = isset($_SESSION['cart']) ? count($_SESSION['cart']) : 0;
echo <<<EOT
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
                <title>$title</title>
                <link href="style.css" rel="stylesheet" type="text/css">
                <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
        </head>
        <body>
     <header>
       <div class="content-wrapper">
         <h1>Cos cumparaturi exemplu</h1>
         <nav>
           <a href="index.php">Home</a>
           <a href="index.php?page=products">Produse</a>
         </nav>
         <div class="link-icons">
           <a href="index.php?page=cart">
                <i class="fas fa-shopping-cart"></i>
<span>$num_items_in_cart</span>
```

```
                                        </a>
            </div>
          </div>
      </header>
      <main>
EOT;
}
// Template footer
function template_footer() {
$year = date('Y');
echo <<<EOT
      </main>
      <footer>
        <div class="content-wrapper">
          <p>&copy; $year, cos cumparaturi exemplu didactic</p>
        </div>
      </footer>

   </body>
</html>
EOT;
}

?>
```

Aceste funcții ne vor face mult mai ușor conectarea la baza de date și formatarea paginilor noastre. Vom include acest fișier în majoritatea fișierelor noastre PHP. În loc să scriem același cod în mod repetat, putem executa cu ușurință numele funcției. De asemenea, putem modifica aspectul sistemului coșului de cumpărături fără a edita fiecare fișier.

Folosim Font Awesome (bibliotecă gratuită de pictograme) pentru pictogramele noastre de fonturi. Foaia de stil CDN este inclusă în secțiunea cap HTML (funcția antet șablon).

5. Crearea fișierului index

Fișierul index.php va fi practic fișierul nostru principal pentru accesarea paginilor. Vom configura rutarea de bază și vom folosi cererile GET pentru a determina care pagină este care.

```
<?php
session_start();
// Includeți funcții și conectați-vă la baza de date folosind PDO MySQL
```

```
include 'conectare.php';
$pdo = pdo_connect_mysql();
// hom este pgina start
$page = isset($_GET['page']) && file_exists($_GET['page'] . '.php') ? $_GET['page'] : 'home';
// Includeți și afișați pagina solicitată
include $page . '.php';
?>
```

Pas1. creăm sesiunea cu funcția session_start. Cu aceasta putem stoca produsele care sunt adăugate în coș, initial, scriptul se va conecta la MySQL folosind funcția de conectare la baza de date pe care am creat-o mai devreme, în fișierul conectare.php.

Metoda de rutare de bază utilizată mai sus verifică dacă variabila de solicitare GET ($ _GET ['pagina']) există. Dacă nu, pagina implicită va fi setată la pagina principală, dacă există, va fi pagina solicitată.

De exemplu, dacă dorim să accesăm pagina produselor, putem naviga la http: //localhost/cos/index.php? Page = products.

Dacă navigați la pagina index.php din browser, pagina va apărea necompletată, deoarece nu am editat încă fișierul home.php.

Pas4. home.php Pagina principală va fi prima pagină pe care o vor vedea clienții noștri. Pentru această pagină, putem adăuga o imagine și un text prezentat, împreună cu o listă cu 4 produse adăugate recent. Codul din linii 1,2,3,4,5, executa o interogare SQL care va primi cele mai recente 4 produse adăugate. Tot ce avem de-a face cu această interogare este să comandăm după coloana date_added și să limităm cu 4, apoi stocați rezultatul în variabila $recently_added_products (ca o matrice asociată).

Creați un șablon de pornire: <?=template_header('Home')?>
Adăugați după eticheta de închidere PHP:Adăugați după eticheta de închidere PHP:

```
<?php
// Afisarea celor mai recente 4 produse
$stmt = $pdo->prepare('SELECT * FROM products ORDER BY date_added DESC LIMIT 4');
$stmt->execute();
$recently_added_products = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>
<?=template_header('Home')?>

<div class="content-wrapper">

    <h2>Gadgets</h2>
    <p>gadget-uri esențiale pentru utilizarea de zi cu zi</p>
</div>

    <h2>Produse adăugate recent</h2>
   <div class="product content-wrapper">
        <?php foreach ($recently_added_products as $product): ?>
        <a href="index.php?page=product&id=<?php echo $product['id']?>" class="product">
```

```
        <img src="imgs/<?php echo $product['img']?>" width="200" height="200" alt="<?php echo $product['name']?>">
        <span class="name"><?php echo $product['name']?></span>
        <span class="price">
           &dollar;<?php echo $product['price']?>
           <?php if ($product['rrp'] > 0): ?>
           <span class="rrp">&dollar;<?php echo $product['rrp']?></span>
           <?php endif; ?>
        </span>
    </a>
    <?php endforeach; ?>

</div>

<?=template_footer()?>
```

Obs: toate imaginile produsului se află în directorul img.

Pagina products.phpPagina produselor va fi locul unde vor merge clienții noștri pentru a naviga prin toate produsele noastre. Vom limita numărul de produse de afișat pe fiecare pagină și vom adăuga paginare care să le permită clienților să navigheze între pagini.

```
<?php
// Cantitățile de produse de afișat pe fiecare pagină
$num_products_on_each_page = 4;
// Pagina curentă, în adresa URL, va apărea ca index.php? Page = products & p = 1, index.php? Page = products & p = 2, etc ..
$current_page = isset($_GET['p']) && is_numeric($_GET['p']) ? (int)$_GET['p'] : 1;
// Selectproduse ordonate dupa data adaugarii
$stmt = $pdo->prepare('SELECT * FROM products ORDER BY date_added DESC LIMIT ?,?');
// bindValue ne va permite să folosim întreg în instrucțiunea SQL, trebuie să folosim pentru LIMIT
$stmt->bindValue(1, ($current_page - 1) * $num_products_on_each_page, PDO::PARAM_INT);
$stmt->bindValue(2, $num_products_on_each_page, PDO::PARAM_INT);
$stmt->execute();
// Preluează produsele din baza de date și returnează rezultatul ca matrice
$products = $stmt->fetchAll(PDO::FETCH_ASSOC);
// Preluam numar total al produselor
$total_products = $pdo->query('SELECT * FROM products')->rowCount();
?>
<?=template_header('Produse')?>

<div class="products content-wrapper">
   <h1>Produse</h1>
   <p><?php echo $total_products?> Produse</p>
```

```
  <div class="products-wrapper">
    <?php foreach ($products as $product): ?>
    <a href="index.php?page=product&id=<?php echo $product['id']?>" class="product">
      <img src="imgs/<?php echo $product['img']?>" width="200" height="200" alt="<?php echo $product['name']?>">
      <span class="name"><?php echo $product['name']?></span>
      <span class="price">
        &dollar;<?php echo $product['price']?>
        <?php if ($product['rrp'] > 0): ?>
        <span class="rrp">&dollar;<?php echo $product['rrp']?></span>
        <?php endif; ?>
      </span>
    </a>
    <?php endforeach; ?>
  </div>
  <div class="buttons">
    <?php if ($current_page > 1): ?>
    <a href="index.php?page=products&p=<?php echo $current_page-1?>">Prev</a>
    <?php endif; ?>
    <?php if ($total_products > ($current_page * $num_products_on_each_page) - $num_products_on_each_page + count($products)): ?>
    <a href="index.php?page=products&p=<?php echo $current_page+1?>">Next</a>
    <?php endif; ?>
  </div>
</div>

<?=template_footer()?>
```

Actualizarea variabilei $ num_products_on_each_page va limita numărul de produse de afișat pe fiecare pagină. Pentru a determina pe ce pagină se află clientul, putem folosi o solicitare GET, în adresa URL aceasta va apărea ca index.php? Page = products & p = 1 etc, iar în scriptul nostru PHP parametrul p pe care îl putem prelua cu $ _GET [„p”] variabilă. Presupunând că solicitarea este validă, codul va executa o interogare care va prelua produsele limitate din baza noastră de date.

Numărul total de produse

Adăugați după: $total_products = $pdo->query('SELECT * FROM products')->rowCount();

product.php -Pagina produsului va afișa toate detaliile pentru un produs specificat, determinate de variabila ID solicitare GET. Clienții pot vizualiza prețul, imaginea și descrierea. Clientul va putea modifica cantitatea și adăuga în coș printr-un clic de buton. Codul (1-16)va verifica dacă variabila de identificare solicitată (cerere GET) există. Dacă este specificat, codul va continua să extragă produsul din tabelul de produse din baza noastră de date. Dacă produsul nu există în baza de date, codul va afișa o eroare simplă, funcția exit () va împiedica executarea ulterioară a scriptului și va afișa eroarea.

```
<?php
```

```
// Verificați pentru a vă asigura că parametrul id este specificat în adresa URL
if (isset($_GET['id'])) {
   // Pregătiți instrucțiunea și executați, previne injecția SQL
   $stmt = $pdo->prepare('SELECT * FROM products WHERE id = ?');
   $stmt->execute([$_GET['id']]);
   // Selectare produs din bd si returnare array
   $product = $stmt->fetch(PDO::FETCH_ASSOC);
   // Verific daca produsul exista (array nu este gol)
   if (!$product) {
      // ID-ul produsului nu există (matricea este goală)
      exit('Produs nu exista!');
   }
} else {
   // ID-ul produsului nu este specificat
   exit('Product nu exista!');
}
?>
<?php echo template_header('Product')?>

<div class="product content-wrapper">
   <img src="imgs/<?php echo $product['img']?>" width="500" height="500" alt="<?php echo $product['name']?>">
   <div>
      <h1 class="name"><?php echo $product['name']?></h1>
      <span class="price">
         &dollar;<?php echo $product['price']?>
         <?php if ($product['rrp'] > 0): ?>
         <span class="rrp">&dollar;<?=$product['rrp']?></span>
         <?php endif; ?>
      </span>
      <form action="index.php?page=cart" method="post">
         <input type="number" name="quantity" value="1" min="1" max="<?php echo $product['quantity']?>" placeholder="Quantity" este obligatoriu>
         <input type="hidden" name="product_id" value="<?php echo $product['id']?>">
         <input type="submit" value="Adaug la cos">
      </form>
      <div class="description">
         <?php echo $product['desc']?>
      </div>
   </div>
</div>

<?=template_footer()?>
```

Pagina coșului de cumpărături -cart.php- este locul în care clientul va putea vedea o listă a produselor sale adăugate în coșul de cumpărăturipot elimina produsele și de a actualiza cantitățile.

```php
<?php
// Dacă utilizatorul a dat clic pe butonul Adăugare la coș de pe pagina produsului, putem verifica datele formularului
if (isset($_POST['product_id'], $_POST['quantity']) && is_numeric($_POST['product_id']) && is_numeric($_POST['quantity'])) {
  // Setați variabilele de postare astfel încât să le identificăm cu ușurință, de asemenea, asigurați-vă că sunt întregi
  $product_id = (int)$_POST['product_id'];
  $quantity = (int)$_POST['quantity'];
  // Pregătiți instrucțiunea SQL, practic verificăm dacă produsul există în baza noastră de date
  $stmt = $pdo->prepare('SELECT * FROM products WHERE id = ?');
  $stmt->execute([$_POST['product_id']]);
  // Aduceți produsul din baza de date și returnează rezultatul ca matrice
  $product = $stmt->fetch(PDO::FETCH_ASSOC);
  // Se verifica daca produsul exista (array nu este gol)
  if ($product && $quantity > 0) {
    // Produsul există în baza de date, acum putem crea / actualiza variabila de sesiune pentru coș
    if (isset($_SESSION['cart']) && is_array($_SESSION['cart'])) {
      if (array_key_exists($product_id, $_SESSION['cart'])) {
        // Produsul există în coș, așa că trebuie doar să actualizați cantitatea
        $_SESSION['cart'][$product_id] += $quantity;
      } else {
        // Produsul nu este în coș, așa că adăugați-l
        $_SESSION['cart'][$product_id] = $quantity;
      }
    } else {
      // se adauga primul produs in cosul gol
      $_SESSION['cart'] = array($product_id => $quantity);
    }
  }//stop retrimiterea
  header('location: index.php?page=cart');
  exit;
}
// Eliminați produsul din coș, verificați dacă URL-ul param „elimină”, acesta este codul produsului, asigurați-vă că este un număr și verificați dacă este în coș
if (isset($_GET['remove']) && is_numeric($_GET['remove']) && isset($_SESSION['cart']) && isset($_SESSION['cart'][$_GET['remove']])) {
  // Stergere produs din cos
  unset($_SESSION['cart'][$_GET['remove']]);
}
// Actualizați cantitățile de produse în coș dacă utilizatorul face clic pe butonul „Actualizare” de pe pagina coșului de cumpărături
if (isset($_POST['update']) && isset($_SESSION['cart'])) {
  // Buclă prin datele de postare, astfel încât să putem actualiza cantitățile pentru fiecare produs din coș
  foreach ($_POST as $k => $v) {
    if (strpos($k, 'quantity') !== false && is_numeric($v)) {
      $id = str_replace('quantity-', '', $k);
```

```
            $quantity = (int)$v;
            // Verifica si validam
            if (is_numeric($id) && isset($_SESSION['cart'][$id]) && $quantity > 0) {
                // Udate cantitate nou
                $_SESSION['cart'][$id] = $quantity;
            }
        }
    }
    // stop retrimitere...
    header('location: index.php?page=cart');
    exit;
}
//Trimiteți utilizatorul la pagina comenzii de plasare dacă face clic pe butonul Plasați comanda, de asemenea coșul nu trebuie să fie gol
if (isset($_POST['placeorder']) && isset($_SESSION['cart']) && !empty($_SESSION['cart'])) {
    header('Location: index.php?page=placeorder');
    exit;
}
// Verificați variabila sesiunii pentru produsele din coș
$products_in_cart = isset($_SESSION['cart']) ? $_SESSION['cart'] : array();
$products = array();
$subtotal = 0.00;
// Daca exista produse in cos
if ($products_in_cart) {
    //  Există produse în coș, așa că trebuie să selectăm acele produse din baza de date
    // Products in cos sunt de tip array deci SQL statement IN (?,?,?,...etc)
    $array_to_question_marks = implode(',', array_fill(0, count($products_in_cart), '?'));
    $stmt = $pdo->prepare('SELECT * FROM products WHERE id IN (' . $array_to_question_marks . ')');
    //Avem nevoie doar de cheile matrice, nu de valori, cheile sunt id-urile produselor
    $stmt->execute(array_keys($products_in_cart));
    // Preluează produsele din baza de date și returnează rezultatul ca matrice
    $products = $stmt->fetchAll(PDO::FETCH_ASSOC);
    // Calculeaza un subtotal
    foreach ($products as $product) {
        $subtotal += (float)$product['price'] * (int)$products_in_cart[$product['id']];
    }
}
?>
<?=template_header('Cart')?>

<div class="cart content-wrapper">
    <h1>Cos cuparaturi</h1>
    <form action="index.php?page=cart" method="post">
        <table>
            <thead>
                <tr>
                    <td colspan="2">Produs</td>
                    <td>Pret</td>
```

```
            <td>Cantitate</td>
            <td>Total</td>
<td>Eliminare</td>
          </tr>
        </thead>
        <tbody>
          <?php if (empty($products)): ?>
          <tr>
            <td colspan="5" style="text-align:center;">Aveti produse in Cos</td>
          </tr>
          <?php else: ?>
          <?php foreach ($products as $product): ?>
          <tr>
            <td class="img">
              <a href="index.php?page=product&id=<?=$product['id']?>">
                <img src="imgs/<?=$product['img']?>" width="50" height="50"
alt="<?=$product['name']?>">
              </a>
            </td>
            <td>
              <a href="index.php?page=product&id=<?=$product['id']?>"><?=$product['name']?></a>
              <br>

            </td>

            <td class="price">&dollar;<?=$product['price']?></td>
            <td class="quantity">
              <input type="number" name="quantity-<?=$product['id']?>"
value="<?=$products_in_cart[$product['id']]?>" min="1" max="<?=$product['quantity']?>"
placeholder="Quantity" required>
            </td>
            <td class="price">&dollar;<?=$product['price'] * $products_in_cart[$product['id']]?></td>
<td> <a href="index.php?page=cart&remove=<?=$product['id']?>"
class="remove">Stergere</a></td>
          </tr>
          <?php endforeach; ?>
          <?php endif; ?>
        </tbody>
      </table>
      <div class="subtotal">
        <span class="text">Subtotal</span>
        <span class="price">&dollar;<?=$subtotal?></span>
      </div>
      <div class="buttons">
        <input type="submit" value="Update" name="update">
        <input type="submit" value="Place Order" name="placeorder">
      </div>
    </form>
```

```
</div>

<?=template_footer()?>
```

În codul de mai sus folosim variabilele de sesiune PHP. Putem folosi sesiunile PHP pentru a ne aminti produsele din coșul de cumpărături, de exemplu, atunci când un client navighează către o altă pagină etc., coșul de cumpărături va conține în continuare produsele adăugate anterior până la expirarea sesiunii. Codul de mai sus va verifica dacă un produs a fost adăugat în coș. Dacă reveniți la fișierul product.php, puteți vedea că am creat un formular HTML. Verificăm aceste valori de formular, dacă produsul există, continuați să verificați produsul selectându-l din tabelul nostru de produse din baza noastră de date. Nu am vrea ca clienții să manipuleze sistemul și să adauge produse inexistente. Coșul= variabila de sesiune va fi o matrice asociată de produse și, cu această matrice, putem adăuga mai multe produse în coșul de cumpărături. Cheia matricei va fi ID-ul produsului, iar valoarea va fi cantitatea. Dacă un produs există deja în coșul de cumpărături, tot ce trebuie să facem este să actualizăm cantitatea.

Pagina placeorder.php va anunța clientul că a făcut o comandă. Clientul trebuie să aibă produse în coșul de cumpărături și a făcut clic pe butonul Plasați comanda de pe pagina coșului de cumpărături.