# Heet Clothing Webshop Dokumentáció



BKSZC Pogány Frigyes Technikum

# Webshop

szoftverfejlesztő és -tesztelő vizsgaremek

2025

Készítette: Konzulens:

Ruzsa Levente XY

Zeljenka MIlán

## A feladat leírása

## .A weboldalunk egy webshopot tartalmaz, ahol ruhákat lehet vásárolni.

Weboldalunk célja az, hogy megfizethető áron az ügyfelek megvásárolják a nekik tetsző árut.

## Kínálataink jó minőségű ruhákat és eredeti márkákat tartalmaz.

## Adatbázison csomó különböző fazontól független terméket lehet találni. Vásárláshoz csinálnunk kell egy fiókot, vagy ha már van egy létrehozott fiókunk akkor már csak bejelentkezni kell.Vásárolni kívánt terméket kosárba kell helyezni és utána megtudjuk venni a terméket, amelyről nyugtát is kapunk.

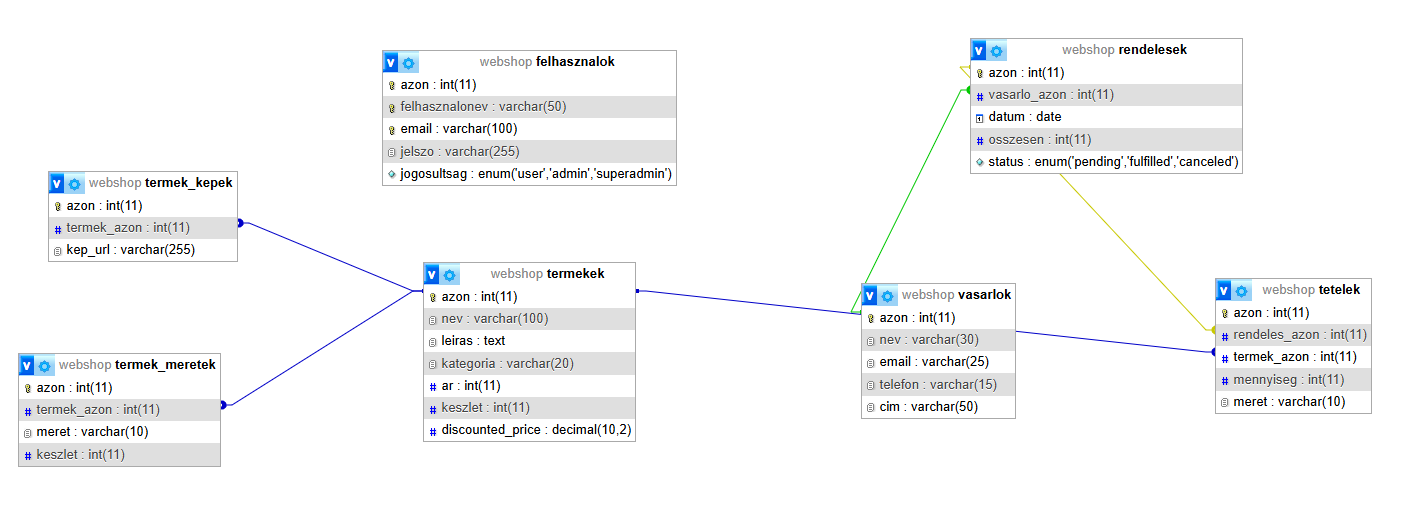
## GitHub repository

Az alkalmazások kódjai ebben a repository-ban érhetők el:

[GitHub - Ruzsa-Levente/vizsgaremek: webshop vizsgaremek](https://github.com/Ruzsa-Levente/vizsgaremek)

## Az adatbázis elkészítése

Az adatbázist a MySQL adatbázis-kezelővel és a PHPMyAdmin programmal készítettük el. Az adatokat három táblára bontottuk:



## A vásárlók adatai

A vásárló azonosítója (automatikusan növekedik), ez a kulcs

A vásárló neve, nem lehet üres

A vásárló e-mail cím, nem lehet üres

A vásárló telefonszáma

A vásárló címe

Ennek megfelelően vásárlók tábla szerkezete:



Az alkalmazás teszteléséhez az alábbi tesztadatokat vittük be:



## A Termékek tábla adatai

A termék azonosítója (automatikusan növekedik), ez a kulcs

A termék neve, nem lehet üres

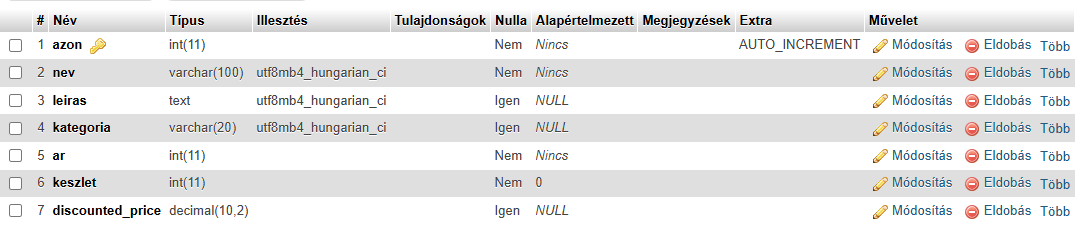
A termék leírása

A termék milyen ruházati kategóriába sorolható

A termék ára, nem lehet üres

Készleten lévő mennyiség, nem lehet üres

Ennek megfelelően termékek tábla szerkezete:



Az alkalmazás teszteléséhez az alábbi tesztadatokat vittük be:



## Rendelések tábla adatai

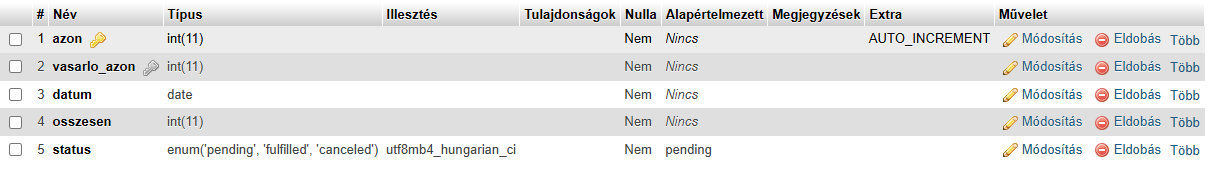
A rendelés azonosítója (automatikusan növekedik), ez a kulcs

A vásárló rendelésének azonosítója, nem lehet üres, külső kulcs

A rendelés dátuma, nem lehet üres

A rendelés összértéke

Ennek megfelelően a rendelések tábla szerkezete:



Az alkalmazás teszteléséhez az alábbi tesztadatokat vittük be:



## Tételek

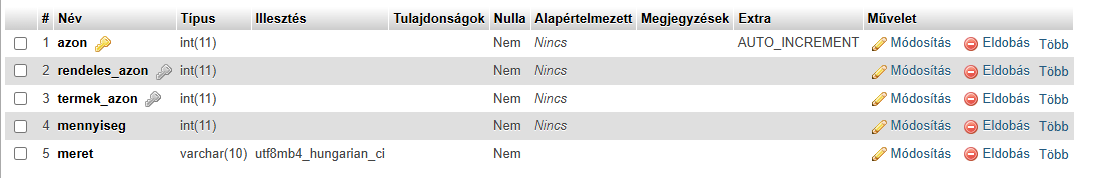
A tétel azonosítója (automatikusan növekedik), ez a kulcs

A tétel azonosítója (automatikusan növekedik), ez a kulcs

A termék tétel azonosítója, nem lehet üres, külső kulcs

A tétel mennyiségi darabszáma, nem lehet üres

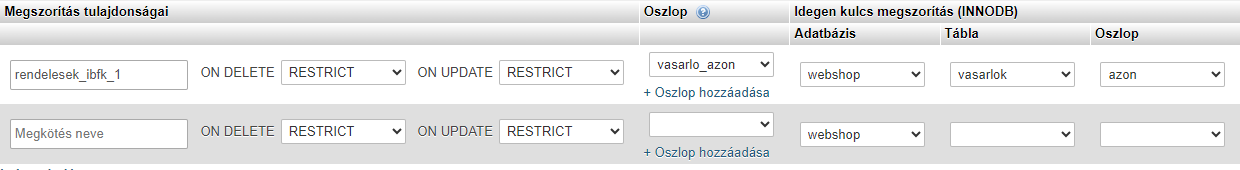
Ennek megfelelően a tételek tábla szerkezete:



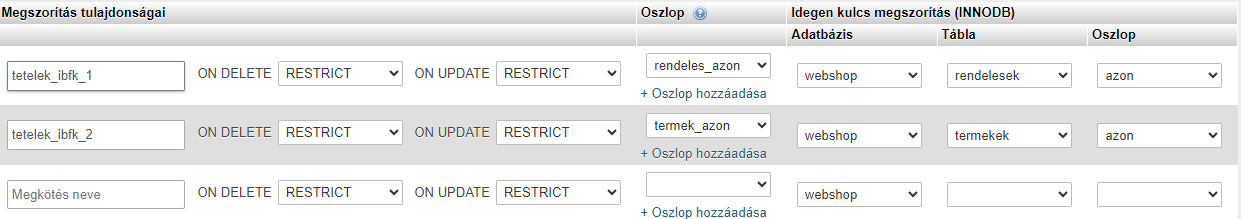
Az alkalmazás teszteléséhez az alábbi tesztadatokat vittük be:



Ahhoz, hogy a több-több kapcsolatban álló főtáblákat lebontsuk két egy-több kapcsolatúra, az altáblát hozzájuk kell fűznünk külső kulcsok megadásával megszorítást létrehozva.



Másik táblához is létrehozzuk a külső kulcsokat:



## 1. Általános Áttekintés

A Heet Clothing webshop egy divatruházati termékeket értékesítő online platform, amely a Bootstrap keretrendszert használja a reszponzív dizájn és komponensek (pl. navigációs sáv, űrlapok) megvalósításához. A dinamikus funkcionalitást PHP és JavaScript kódok biztosítják, például a session kezelés, adatbázis műveletek és kosárkezelés terén. Minden oldal egységes navigációs sávval és lábléccel rendelkezik, a fő tartalom pedig az egyedi <main> szekcióban jelenik meg.

## 2. Navigációs Sáv

A navigációs sáv minden oldalon megtalálható, a webshop navigációjának központi eleme. Bootstrap navbar komponensként valósul meg, reszponzív és a képernyő tetején rögzített (fixed-top).

### Funkciók:

* **Toggler gomb**: Kis képernyőkön a menü kinyitását és bezárását vezérli (data-bs-toggle="collapse").
* **Keresőmező**: Beviteli mező (<input>) és kereső ikon (<i>), amely a keresést indítja. A keresési kifejezést PHP állítja be (<?= htmlspecialchars($searchTerm) ?>).
* **Logó**: A webshop logója, amely a kezdőlapra (index.php) linkel.
* **Menüpontok**: Lista (<ul class="navbar-nav">) linkekkel: Home, Clothes, Sale, Sign Up, About Us.
* **Ikonok**: Kosár (fa-shopping-cart) és felhasználói ikon (fa-user), amelyek a számlázási (billing.php) és bejelentkezési (login.php) oldalakra vezetnek. A kosár mellett a tételek száma (<span id="cart-count">) látható, amit JavaScript frissít.

### Kód Részlet:

<nav class="navbar navbar-expand-lg fixed-top">

<div class="container-fluid nav-container d-flex flex-column align-items-center">

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">

<span class="navbar-toggler-icon"></span>

</button>

<div class="search-container d-flex align-items-center">

<i class="fas fa-search" id="search-toggle"></i>

<input type="text" id="search-input" placeholder="Search..." value="<?= htmlspecialchars($searchTerm) ?>">

</div>

<a class="navbar-brand logo mb-2" href="index.php">

<img src="kepek/heet-logo-white.png" alt="Webshop Logo">

</a>

<div class="collapse navbar-collapse justify-content-center" id="navbarNav">

<ul class="navbar-nav">

<li class="nav-item"><a class="nav-link" href="index.php">Home</a></li>

<li class="nav-item"><a class="nav-link" href="clothes.php">Clothes</a></li>

<li class="nav-item"><a class="nav-link" href="sale.php">Sale</a></li>

<li class="nav-item"><a class="nav-link" href="signup.php">Sign Up</a></li>

<li class="nav-item"><a class="nav-link" href="about.php">About Us</a></li>

</ul>

</div>

<div class="icon-container position-absolute top-50 translate-middle-y d-flex align-items-center">

<div class="cart-icon"><a href="billing.php"><i class="fas fa-shopping-cart"></i><span id="cart-count">0</span></a></div>

<div class="user-icon"><a href="login.php"><i class="fas fa-user"></i></a></div>

</div>

</div>

</nav>

## 3. Lábléc

Minden oldalon egységes lábléc található, amely alapvető információkat és linkeket biztosít.

### Funkciók:

* **Információ**: Copyright szöveg és ÁSZF link.
* **Közösségi média**: Ikonok linkekkel a Facebook, Instagram és TikTok oldalakra.

### Kód Részlet:

<footer class="footer text-white text-center py-4 mt-auto">

<div class="container">

<p>© 2025 Heet Clothing | All rights reserved.</p>

<a href="aszf.php" class="text-white text-decoration-none">ÁSZF</a>

<div class="social-icons mt-3">

<a href="https://www.facebook.com" class="text-white mx-3"><i class="fab fa-facebook-f"></i></a>

<a href="https://www.instagram.com" class="text-white mx-3"><i class="fab fa-instagram"></i></a>

<a href="https://www.tiktok.com" class="text-white mx-3"><i class="fab fa-tiktok"></i></a>

</div>

</div>

</footer>

## 4. Kezdőlap (index.php)

A webshop főoldala, amely a felhasználók bevonzására szolgál.

### Funkciók:

* **Hero szekció**: Kiemelt kép és cselekvésre ösztönző gomb.
* **Slider**: Swiper használatával trending és akciós termékek bemutatása.
* **PHP**: Termékek lekérdezése az adatbázisból a kiemelt szekciókhoz.

## 5. Ruhák Listázása (clothes.php)

Ez az oldal a termékeket listázza, keresési és rendezési lehetőségekkel.

### Funkciók:

* **Termékkártyák**: Bootstrap rácsban (row, col) jelennek meg.
* **Rendezési menü**: Legördülő lista (select) a rendezési opciókhoz.
* **Dinamikus lekérdezés**: PHP és SQL alapján keresés és rendezés.

### Kód Részlet:

$sql = "SELECT azon, nev, ar, discounted\_price FROM termekek WHERE keszlet > 0";

if ($searchTerm) {

$sql .= " AND (nev LIKE :searchTerm OR leiras LIKE :searchTerm)";

}

$stmt = $pdo->prepare($sql);

$stmt->bindValue(':searchTerm', '%' . $searchTerm . '%');

$stmt->execute();

$products = $stmt->fetchAll(PDO::FETCH\_ASSOC);

## 6. Termék Részletei (product.php)

Egy adott termék részletes adatait mutatja be.

### Funkciók:

* **Képgaléria**: Több termékkép megjelenítése.
* **Méretválasztó**: Elérhető méretek kiválasztása.
* **Kosárba helyezés**: Gomb a termék kosárba tételéhez.
* **PHP**: Termékadatok lekérdezése azonosító alapján.

### Kód Részlet:

$sql = "SELECT \* FROM termekek WHERE azon = :id";

$stmt = $pdo->prepare($sql);

$stmt->bindParam(':id', $product\_id);

$stmt->execute();

## 7. Akciós Termékek Listázása (sale.php)

A leárazott termékeket mutatja be, keresési és rendezési funkciókkal.

### Funkciók:

* **Akciós termékek**: Csak a leárazott termékek jelennek meg.
* **Keresés és rendezés**: Név, ár és népszerűség szerinti opciók.
* **Árak**: Eredeti és akciós árak kijelzése.

### Kód Részlet:

$sql = "SELECT azon, nev, ar, discounted\_price

FROM termekek

WHERE keszlet > 0

AND discounted\_price IS NOT NULL

AND discounted\_price < ar";

if ($searchTerm) {

$sql .= " AND (nev LIKE :searchTerm OR leiras LIKE :searchTerm)";

}

$sort\_order = "ORDER BY nev ASC";

if (isset($\_GET['sort'])) {

switch ($\_GET['sort']) {

case 'price\_asc': $sort\_order = "ORDER BY discounted\_price ASC"; break;

case 'price\_desc': $sort\_order = "ORDER BY discounted\_price DESC"; break;

case 'popularity': $sort\_order = "ORDER BY RAND()"; break;

case 'name\_asc': $sort\_order = "ORDER BY nev ASC"; break;

case 'name\_desc': $sort\_order = "ORDER BY nev DESC"; break;

}

}

$sql .= " $sort\_order";

$stmt = $pdo->prepare($sql);

if ($searchTerm) {

$stmt->bindValue(':searchTerm', '%' . $searchTerm . '%', PDO::PARAM\_STR);

}

$stmt->execute();

$products = $stmt->fetchAll(PDO::FETCH\_ASSOC);

## 8. Regisztrációs Oldal (signup.php)

Új felhasználók regisztrációját teszi lehetővé.

### Funkciók:

* **Űrlap**: Felhasználónév, e-mail és jelszó bekérése.
* **Bejelentkezési link**: Már regisztrált felhasználóknak.
* **PHP**: Adatok tárolása az adatbázisban.

## 9. Bejelentkezési Oldal (login.php)

Felhasználók autentikációját kezeli.

### Funkciók:

* **Űrlap**: Felhasználónév és jelszó bevitele.
* **Session ellenőrzés**: Bejelentkezett felhasználók átirányítása.

### Kód Részlet:

if (isset($\_SESSION['loggedin']) && $\_SESSION['loggedin'] === true) {

header("Location: profile.php");

}

## 10. Számlázási Oldal (billing.php)

A kosár tartalmát és a szállítási adatokat kezeli.

### Funkciók:

* **Kosár összefoglaló**: Tételek és árak listázása.
* **Szállítási űrlap**: Adatok bekérése.
* **JavaScript**: Dinamikus összesítés frissítése.

### Kód Részlet:

<main class="main-content">

<div class="container">

<section id="billing" class="billing-container mx-auto mt-6">

<h2 class="text-center mb-4">Your Receipt</h2>

<div class="cart-wrapper d-flex flex-wrap">

<div class="cart-left">

<h4>Kosár tartalma</h4>

<ul id="cart-items" class="list-unstyled"></ul>

<p id="total">Total: 0.00Ft</p>

</div>

<div class="cart-right">

<h4 class="text-center">Summary</h4>

<form id="checkout-form">

<label for="name">Teljes név:</label>

<input type="text" id="name" class="form-control" required>

<label for="email">E-mail:</label>

<input type="email" id="email" class="form-control" required>

<label for="phone">Telefonszám:</label>

<input type="tel" id="phone" class="form-control" required>

<label for="delivery">Szállítási mód:</label>

<select id="delivery" class="form-select">

<option value="home">Házhozszállítás</option>

<option value="pickup">Áruházi átvétel</option>

</select>

<button type="submit" id="place-order-btn" class="btn btn-success w-100 mt-3">Megrendelés</button>

</form>

</div>

</div>

</section>

</div>

</main>

## 11. Fizetési Oldal (payment.php)

A rendelés véglegesítését szolgálja.

### Funkciók:

* **Űrlap**: Szállítási és kapcsolati adatok bekérése.
* **Szállítási módok**: JavaScript vezérli az űrlap láthatóságát.

## 12. Sikeres Vásárlás Oldal (success.php)

A vásárlás befejezése után jelenik meg.

### Funkciók:

* **Összegzés**: Vásárló adatai és tételek.
* **Akciós árak**: Figyelembe veszi a kedvezményeket.

### Kód Részlet:

$rendeles\_id = isset($\_GET['rendeles\_id']) ? (int)$\_GET['rendeles\_id'] : 0;

$stmt = $conn->prepare("

SELECT v.nev, v.email, v.telefon, v.cim

FROM vasarlok v

JOIN rendelesek r ON v.azon = r.vasarlo\_azon

WHERE r.azon = ?

");

$stmt->bind\_param("i", $rendeles\_id);

$stmt->execute();

$vasarlo = $stmt->get\_result()->fetch\_assoc();

## 13. "About Us" Oldal (about.php)

A Heet Clothing céget mutatja be.

### Funkciók:

* **Tartalom**: Céginformációk szekciókban.
* **Animációk**: animate.css effektek.

### Kód Részlet:

<main class="main-content">

<section id="about" class="about-section container">

<h2 class="about-title animate\_\_fadeIn">About Heet</h2>

<p class="about-subtitle animate\_\_fadeInUp animate\_\_delay-1s">Welcome to Heet – where fashion meets passion...</p>

<div class="about-content">

<div class="about-item animate\_\_fadeInLeft animate\_\_delay-2s">

<h3>Who Are We?</h3>

<p>Heet is more than just a clothing brand...</p>

</div>

<div class="about-item animate\_\_fadeInRight animate\_\_delay-2s">

<h3>Who Is Our Target Audience?</h3>

<p>Our brand is built for those who live life on their own terms...</p>

</div>

<div class="about-item animate\_\_fadeInUp animate\_\_delay-3s">

<h3>Where Can You Find Us?</h3>

<p>Shop Heet online or visit us in person...</p>

</div>

</div>

</section>

</main>

## 14. ÁSZF Oldal (aszf.php)

A webshop jogi feltételeit tartalmazza.

### Funkciók:

* **Tartalom**: Statikus szöveg, interakció nélkül.

### Kód Részlet:

<main class="main-content">

<div class="container">

<section id="about" class="mx-auto">

ÁLTALÁNOS SZERZŐDÉSI FELTÉTELEK (ÁSZF)<br>

Heet Clothing Webáruház<br>

Hatálybalépés dátuma: 2025. március 18.<br>

1. Bevezetés<br>

Jelen Általános Szerződési Feltételek...

</section>

</div>

</main>

## 15. Adminisztrációs Oldal (admin.php)

Adminisztrátorok vezérlőpultja.

### Funkciók:

* **Felhasználói adatok**: Bejelentkezett admin információi.
* **Linkek**: Felhasználók, rendelések és termékek kezelése.
* **Kijelentkezés**: Űrlap a kilépéshez.

### Kód Részlet:

<main>

<section class="profile-section container-distance2">

<div class="profile-container">

<h2>Admin Panel</h2>

<p>Üdv, <?= htmlspecialchars($\_SESSION['username']); ?>!</p>

<p>Email: <?= htmlspecialchars($\_SESSION['email']); ?></p>

<p>Jogosultság: <?= htmlspecialchars($\_SESSION['jogosultsag']); ?></p>

<hr>

<h3>Admin Functions</h3>

<ul>

<li><a href="manage\_users.php">Felhasználók kezelése</a></li>

<li><a href="manage\_orders.php">Rendelések kezelése</a></li>

<li><a href="manage\_products.php">Termékek kezelése</a></li>

</ul>

<form action="php/logout.php" method="POST">

<button type="submit">Kijelentkezés</button>

</form>

</div>

</section>

</main>

## 16. Termék Hozzáadása Oldal (add\_product.php)

Új termékek felvitelét teszi lehetővé.

### Funkciók:

* **Űrlap**: Termékadatok, méretek és képek bekérése.
* **JavaScript**: Dinamikus méret hozzáadása/törlése.

### Kód Részlet:

<main class="container-distance">

<h2 id="manage2-text">Add New Product</h2>

<form action="add\_product.php" method="POST" enctype="multipart/form-data">

<label for="name">Product Name</label>

<input type="text" id="name" name="name" required><br>

<label for="description">Description</label>

<textarea id="description" name="description" required></textarea><br>

<label for="category">Category</label>

<input type="text" id="category" name="category"><br>

<label for="price">Price</label>

<input type="number" id="price" name="price" required><br>

<h3>Sizes</h3>

<div id="sizes-container">

<div class="size-row">

<input type="text" name="sizes[0][meret]" placeholder="Size (e.g., S, M, L)" required>

<input type="number" name="sizes[0][keszlet]" placeholder="Stock" min="0" required>

<button type="button" class="remove-size-btn" onclick="removeSizeRow(this)">Remove</button>

</div>

</div>

<button type="button" id="add-size-btn">Add Size</button><br>

<label for="images">Product Images</label>

<input type="file" name="images[]" accept="image/\*" multiple><br>

<button type="submit">Add Product</button>

</form>

</main>

<script>

document.getElementById('add-size-btn').addEventListener('click', function() {

const container = document.getElementById('sizes-container');

const index = container.children.length;

const newRow = document.createElement('div');

newRow.className = 'size-row';

newRow.innerHTML = `

<input type="text" name="sizes[${index}][meret]" placeholder="Size (e.g., S, M, L)" required>

<input type="number" name="sizes[${index}][keszlet]" placeholder="Stock" min="0" required>

<button type="button" class="remove-size-btn" onclick="removeSizeRow(this)">Remove</button>

`;

container.appendChild(newRow);

});

function removeSizeRow(button) {

button.parentElement.remove();

}

</script>

## 17. Termék Szerkesztése (edit\_product.php)

Meglévő termékek módosítására szolgál.

### Funkciók:

* **Űrlap**: Előre kitöltött adatok szerkesztése.
* **Képkezelés**: Új képek feltöltése, régiek törlése.

### Kód Részlet:

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$stmt = $pdo->prepare("UPDATE termekek SET nev = ?, leiras = ?, ar = ? WHERE azon = ?");

$stmt->execute([$\_POST['name'], $\_POST['description'], $\_POST['price'], $productId]);

}

## 18. Felhasználók Kezelése (manage\_users.php)

Felhasználói adatok adminisztrációja.

### Funkciók:

* **Lista**: Táblázatban megjelenő felhasználók.
* **Szerepkörök**: Superadmin módosíthatja őket.

### Kód Részlet:

$sql = "SELECT azon, felhasznalonev, email, jogosultsag FROM felhasznalok";

$result = $conn->query($sql);

## 19. Rendelések Kezelése (manage\_orders.php)

Rendelések áttekintése adminisztrátorok számára.

### Funkciók:

* **Táblázat**: Rendelések listázása.
* **Státusz**: Módosítási lehetőség.

### Kód Részlet:

$sql = "SELECT rendelesek.azon, vasarlok.nev, rendelesek.datum FROM rendelesek JOIN vasarlok ON rendelesek.vasarlo\_azon = vasarlok.azon";

$result = $conn->query($sql);

## 20. Rendelés Részletei (view\_order.php)

Egy adott rendelés részletes adatai.

### Funkciók:

* **Részletek**: Vásárló, tételek és összeg.
* **Státusz módosítás**: Legördülő menüvel.

### Kód Részlet:

session\_start();

if (!isset($\_SESSION['loggedin']) || $\_SESSION['loggedin'] !== true ||

!isset($\_SESSION['jogosultsag']) || ($\_SESSION['jogosultsag'] !== 'admin' && $\_SESSION['jogosultsag'] !== 'superadmin')) {

header("Location: index.php");

exit();

}

$order\_id = $\_GET['order\_id'];

$sql\_order = "SELECT rendelesek.azon, vasarlok.nev, vasarlok.email, rendelesek.datum, rendelesek.osszesen, rendelesek.status

FROM rendelesek

JOIN vasarlok ON rendelesek.vasarlo\_azon = vasarlok.azon

WHERE rendelesek.azon = ?";

$stmt = $conn->prepare($sql\_order);

$stmt->bind\_param("i", $order\_id);

$stmt->execute();

$order = $stmt->get\_result()->fetch\_assoc();

## Dinamikus Funkcionalitás

### Session Kezelés

* **PHP**: Bejelentkezés ellenőrzése és jogosultságok validálása.

session\_start();

if (!isset($\_SESSION['loggedin'])) {

header("Location: index.php");

}

### Adatbázis Műveletek

* PDO használata biztonságos lekérdezésekhez, SQL injekció ellen védelemmel.

### Akciós Termékek Kezelése

* Akciós árak (discounted\_price) megjelenítése és számítása a listákban és rendelésekben.

### Rendelés Állapotának Módosítása

* Adminisztrátorok számára státuszfrissítés (update\_order\_status.php).

### Termékek Rendezése

* SQL ORDER BY alapján rendezés ár, népszerűség vagy név szerint.

## 21. Termék Törlése (delete\_product.php)

Ez a fájl egy termék törlését kezeli az adatbázisból és a szerverről.

**Funkciók:**

* **Képfájlok törlése**: A termékhez tartozó képeket eltávolítja a kepek mappából.
* **Adatbázis műveletek**: Törli a képeket a termek\_kepek táblából, majd magát a terméket a termekek táblából.
* **Tranzakció**: Biztosítja az adatok konzisztenciáját.

**Kód Részlet:**

if (isset($\_GET['id']) && is\_numeric($\_GET['id'])) {

$productId = $\_GET['id'];

try {

$pdo->beginTransaction();

$sql = "SELECT kep\_url FROM termek\_kepek WHERE termek\_azon = ?";

$stmt = $pdo->prepare($sql);

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

$images = $stmt->fetchAll(PDO::FETCH\_ASSOC);

foreach ($images as $image) {

$imagePath = "../kepek/" . $image['kep\_url'];

if (file\_exists($imagePath)) {

unlink($imagePath);

}

}

$sql = "DELETE FROM termek\_kepek WHERE termek\_azon = ?";

$stmt = $pdo->prepare($sql);

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

$sql = "DELETE FROM termekek WHERE azon = ?";

$stmt = $pdo->prepare($sql);

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

$pdo->commit();

header("Location: ../manage\_products.php");

exit();

} catch (Exception $e) {

$pdo->rollBack();

echo "Error: " . $e->getMessage();

}

}

**22. Bejelentkezési Feldolgozás (login\_process.php)**

A bejelentkezési űrlap adatait ellenőrzi és kezeli.

**Funkciók:**

* **Hitelesítés**: Ellenőrzi a felhasználónév és jelszó párost.
* **Session beállítás**: Sikeres bejelentkezés esetén session változókat állít be.
* **Hibaüzenetek**: Átirányítással jelzi a hibát.

**Kód Részlet:**

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$username = trim($\_POST['username']);

$password = trim($\_POST['password']);

try {

$sql = "SELECT \* FROM felhasznalok WHERE felhasznalonev = :username";

$stmt = $pdo->prepare($sql);

$stmt->execute(['username' => $username]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($user && password\_verify($password, $user['jelszo'])) {

$\_SESSION['loggedin'] = true;

$\_SESSION['user\_id'] = $user['azon'];

$\_SESSION['username'] = $user['felhasznalonev'];

$\_SESSION['email'] = $user['email'];

$\_SESSION['jogosultsag'] = $user['jogosultsag'];

header("Location: ../profile.php");

exit();

} else {

header("Location: ../login.php?error=1");

exit();

}

} catch (PDOException $e) {

header("Location: ../login.php?error=db");

exit();

}

}

**23. Felhasználó Törlése (delete\_user.php)**

Egy felhasználó törlését végzi el jogosultsági ellenőrzésekkel.

**Funkciók:**

* **Önmaga törlésének tiltása**: A bejelentkezett felhasználó nem törölheti saját fiókját.
* **Jogosultság ellenőrzés**: Admin nem törölhet másik admin vagy superadmin felhasználót.
* **Tranzakció**: Biztonságos törlés.

**Kód Részlet:**

if (isset($\_POST['user\_id']) && is\_numeric($\_POST['user\_id'])) {

$userId = $\_POST['user\_id'];

$loggedInUserId = $\_SESSION['user\_id'];

$loggedInRole = $\_SESSION['jogosultsag'];

if ($userId == $loggedInUserId) {

echo "You cannot delete your own account.";

exit();

}

if ($loggedInRole === 'admin') {

$sql = "SELECT jogosultsag FROM felhasznalok WHERE azon = ?";

$stmt = $pdo->prepare($sql);

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

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($user && ($user['jogosultsag'] === 'admin' || $user['jogosultsag'] === 'superadmin')) {

echo "You cannot delete other admins or superadmins.";

exit();

}

}

try {

$pdo->beginTransaction();

$sql = "DELETE FROM felhasznalok WHERE azon = ?";

$stmt = $pdo->prepare($sql);

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

$pdo->commit();

header("Location: ../manage\_users.php");

exit();

} catch (Exception $e) {

$pdo->rollBack();

echo "Error: " . $e->getMessage();

}

}

**24. Rendelés Törlése (delete\_order.php)**

Egy rendelés törlését kezeli, a készlet visszaállításával.

**Funkciók:**

* **Készlet visszaállítás**: A törölt rendelés tételei alapján növeli a termékek készletét.
* **Tételek törlése**: Előbb a tetelek táblából, majd a rendelesek táblából töröl.

**Kód Részlet:**

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['order\_id'])) {

$order\_id = $\_POST['order\_id'];

$sql\_get\_items = "SELECT termek\_azon, mennyiseg FROM tetelek WHERE rendeles\_azon = ?";

$stmt\_get\_items = $conn->prepare($sql\_get\_items);

$stmt\_get\_items->bind\_param("i", $order\_id);

$stmt\_get\_items->execute();

$result = $stmt\_get\_items->get\_result();

while ($row = $result->fetch\_assoc()) {

$product\_id = $row['termek\_azon'];

$quantity = $row['mennyiseg'];

$sql\_update\_stock = "UPDATE termekek SET keszlet = keszlet + ? WHERE azon = ?";

$stmt\_update\_stock = $conn->prepare($sql\_update\_stock);

$stmt\_update\_stock->bind\_param("ii", $quantity, $product\_id);

$stmt\_update\_stock->execute();

}

$sql\_delete\_items = "DELETE FROM tetelek WHERE rendeles\_azon = ?";

$stmt\_items = $conn->prepare($sql\_delete\_items);

$stmt\_items->bind\_param("i", $order\_id);

$stmt\_items->execute();

$sql\_delete\_order = "DELETE FROM rendelesek WHERE azon = ?";

$stmt\_order = $conn->prepare($sql\_delete\_order);

$stmt\_order->bind\_param("i", $order\_id);

$stmt\_order->execute();

header("Location: ../manage\_orders.php");

exit();

}

**25. Adatbázis Kapcsolat (connect.php)**

Az adatbázis kapcsolatot inicializálja PDO és MySQLi használatával.

**Funkciók:**

* **PDO**: Strukturált lekérdezésekhez.
* **MySQLi**: Alternatív kapcsolati lehetőség.

**Kód Részlet:**

$dsn = "mysql:host=localhost;dbname=webshop;charset=utf8mb4";

$username = "root";

$password = "";

try {

$pdo = new PDO($dsn, $username, $password);

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch (PDOException $e) {

echo "Connection failed: " . $e->getMessage();

exit();

}

$conn = new mysqli("localhost", $username, $password, "webshop");

if ($conn->connect\_error) {

die("Kapcsolati hiba: " . $conn->connect\_error);

}

**26. Rendelés Feldolgozása (process\_order.php)**

A kosár tartalmát menti rendelésként az adatbázisba.

**Funkciók:**

* **Vásárló rögzítése**: Alapértelmezett "Vendég" adatokkal.
* **Rendelés létrehozása**: Tételek és összeg mentése.
* **Készlet csökkentése**: Méret-specifikus készlet frissítése.
* **JSON válasz**: Siker vagy hibaüzenet visszaadása.

**Kód Részlet:**

$data = json\_decode(file\_get\_contents("php://input"), true);

if (!$data || empty($data['cartItems'])) {

echo json\_encode(["success" => false, "message" => "Nincs rendelési adat!"]);

exit;

}

$conn->begin\_transaction();

try {

$vasarlo\_nev = "Vendég";

$vasarlo\_email = "guest@example.com";

$stmt = $conn->prepare("INSERT INTO vasarlok (nev, email) VALUES (?, ?)");

$stmt->bind\_param("ss", $vasarlo\_nev, $vasarlo\_email);

$stmt->execute();

$vasarlo\_id = $stmt->insert\_id;

$datum = date("Y-m-d");

$osszesen = $data["total"];

$stmt = $conn->prepare("INSERT INTO rendelesek (vasarlo\_azon, datum, osszesen) VALUES (?, ?, ?)");

$stmt->bind\_param("isi", $vasarlo\_id, $datum, $osszesen);

$stmt->execute();

$rendeles\_id = $stmt->insert\_id;

foreach ($data["cartItems"] as $item) {

$termek\_nev = $item["name"];

$mennyiseg = $item["quantity"];

$meret = $item["size"];

$stmt = $conn->prepare("SELECT azon FROM termekek WHERE nev = ?");

$stmt->bind\_param("s", $termek\_nev);

$stmt->execute();

$termek = $stmt->get\_result()->fetch\_assoc();

$termek\_azon = $termek["azon"];

$stmt = $conn->prepare("UPDATE termek\_meretek SET keszlet = keszlet - ? WHERE termek\_azon = ? AND meret = ?");

$stmt->bind\_param("iis", $mennyiseg, $termek\_azon, $meret);

$stmt->execute();

$stmt = $conn->prepare("INSERT INTO tetelek (rendeles\_azon, termek\_azon, mennyiseg, meret) VALUES (?, ?, ?, ?)");

$stmt->bind\_param("iiis", $rendeles\_id, $termek\_azon, $mennyiseg, $meret);

$stmt->execute();

}

$conn->commit();

echo json\_encode(["success" => true, "message" => "A rendelés sikeresen feldolgozva!"]);

} catch (Exception $e) {

$conn->rollback();

echo json\_encode(["success" => false, "message" => $e->getMessage()]);

}

**27. Kijelentkezés (logout.php)**

A felhasználói munkamenetet megszünteti.

**Funkciók:**

* **Session törlése**: Minden session adat eltávolítása.
* **Átirányítás**: Vissza a bejelentkezési oldalra.

**Kód Részlet:**

session\_start();

session\_unset();

session\_destroy();

header("Location: ../login.php");

exit();

**28. Rendelés Állapot Frissítése (update\_order\_status.php)**

A rendelés állapotát módosítja adminisztrátorok számára.

**Funkciók:**

* **Státusz frissítés**: Az új állapot mentése az adatbázisba.
* **Üzenetek**: Sikeres vagy hibás művelet jelzése session változóval.

**Kód Részlet:**

if ($\_SERVER["REQUEST\_METHOD"] == "POST" && isset($\_POST['order\_id']) && isset($\_POST['new\_status'])) {

$order\_id = $\_POST['order\_id'];

$new\_status = $\_POST['new\_status'];

$sql = "UPDATE rendelesek SET status = ? WHERE azon = ?";

$stmt = $conn->prepare($sql);

$stmt->bind\_param("si", $new\_status, $order\_id);

if ($stmt->execute()) {

$\_SESSION['success\_message'] = "Rendelés állapota frissítve!";

} else {

$\_SESSION['error\_message'] = "Hiba történt a frissítés során.";

}

header("Location: ../view\_order.php?order\_id=$order\_id");

exit();

}

**29. Felhasználói Szerepkör Frissítése (update\_user\_role.php)**

Superadmin módosíthatja a felhasználók jogosultságát.

**Funkciók:**

* **Szerepkör ellenőrzés**: Csak érvényes szerepkörök engedélyezettek.
* **Frissítés**: Jogosultság mentése az adatbázisba.

**Kód Részlet:**

if ($\_SERVER["REQUEST\_METHOD"] === "POST" && isset($\_POST['user\_id'], $\_POST['new\_role'])) {

$user\_id = intval($\_POST['user\_id']);

$new\_role = $\_POST['new\_role'];

if (!in\_array($new\_role, ['user', 'admin', 'superadmin'])) {

exit("Érvénytelen jogosultság.");

}

$stmt = $conn->prepare("UPDATE felhasznalok SET jogosultsag = ? WHERE azon = ?");

$stmt->bind\_param("si", $new\_role, $user\_id);

if ($stmt->execute()) {

header("Location: ../manage\_users.php");

exit();

} else {

echo "Hiba a frissítés során.";

}

}

**30. Regisztráció Feldolgozása (signup\_process.php)**

Új felhasználó regisztrációját kezeli.

**Funkciók:**

* **Adatok ellenőrzése**: Felhasználónév és e-mail egyediségének biztosítása.
* **Jelszó titkosítás**: password\_hash használata.
* **Automatikus bejelentkezés**: Sikeres regisztráció után.

**Kód Részlet:**

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

try {

$felhasznalonev = htmlspecialchars($\_POST['felhasznalonev'], ENT\_QUOTES, 'UTF-8');

$email = htmlspecialchars($\_POST['email'], ENT\_QUOTES, 'UTF-8');

$jelszo = password\_hash($\_POST['jelszo'], PASSWORD\_DEFAULT);

$check\_sql = "SELECT \* FROM felhasznalok WHERE felhasznalonev = :felhasznalonev OR email = :email";

$check\_stmt = $pdo->prepare($check\_sql);

$check\_stmt->execute(['felhasznalonev' => $felhasznalonev, 'email' => $email]);

if ($check\_stmt->rowCount() > 0) {

echo "Error: Username or email already exists!";

exit();

}

$sql = "INSERT INTO felhasznalok (felhasznalonev, email, jelszo) VALUES (:felhasznalonev, :email, :jelszo)";

$stmt = $pdo->prepare($sql);

$stmt->execute(['felhasznalonev' => $felhasznalonev, 'email' => $email, 'jelszo' => $jelszo]);

$sql = "SELECT \* FROM felhasznalok WHERE felhasznalonev = :felhasznalonev";

$stmt = $pdo->prepare($sql);

$stmt->execute(['felhasznalonev' => $felhasznalonev]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

$\_SESSION['loggedin'] = true;

$\_SESSION['user\_id'] = $user['azon'];

$\_SESSION['username'] = $user['felhasznalonev'];

$\_SESSION['email'] = $user['email'];

$\_SESSION['jogosultsag'] = $user['jogosultsag'];

header("Location: ../index.php?signup=success");

exit();

} catch (PDOException $e) {

echo "Error: " . $e->getMessage();

}

}

**Dinamikus Funkcionalitás Kiegészítése**

**Tranzakciók Használata**

* Termék és rendelés törlése során tranzakciók biztosítják az adatok konzisztenciáját.

**Készletkezelés**

* Rendelés feldolgozásakor a készlet csökken, törléskor pedig visszapótlódik.

**JSON Kommunikáció**

* A process\_order.php JSON formátumban kommunikál az ügyféllel.

**Biztonság**

* Jelszavak titkosítása (password\_hash), bemenetek tisztítása (htmlspecialchars).

## 31. Kliensoldali Logika (script.js)

A script.js fájl a webshop kliensoldali funkcionalitását biztosítja, JavaScript használatával. Ez a fájl felelős a kosárkezelésért, a navigációs elemek interaktivitásáért, a keresési funkciókért, a termékoldali képgalériáért, valamint az animációk és dinamikus tartalom megjelenítéséért. A kód modularisan épül fel, különböző függvényekkel, amelyek az oldalak közötti konzisztens működést biztosítják. A localStorage használatával a kosár tartalma tartósan tárolódik a böngészőben, így az oldalak újratöltése vagy navigáció során sem vesznek el az adatok.

### Főbb Funkciók és Szerepek:

1. **Kosárkezelés**:
   * **Globális kosár**: A cart tömb tárolja a kosár tartalmát (termék neve, ára, képe, mérete, mennyisége).
   * **Hozzáadás**: Az addToCart függvény méretellenőrzéssel ad termékeket a kosárhoz, meglévő termék esetén növeli a mennyiséget.
   * **Módosítás**: Az increaseQuantity és removeFromCart függvények a mennyiséget növelik vagy csökkentik, az utóbbi törli a terméket, ha a mennyiség 1-re csökken.
   * **Törlés**: A deleteFromCart azonnal eltávolít egy terméket.
   * **Frissítés**: Az updateReceipt a kosár tartalmát dinamikusan megjeleníti a billing.php oldalon, az updateCartCount pedig a navigációs sávban lévő kosár darabszámot frissíti.
2. **Tárolás és Betöltés**:
   * A saveCartToLocalStorage a kosarat JSON formátumban menti a localStorage-ba.
   * A loadCartFromLocalStorage betölti a mentett kosarat az oldal betöltésekor, így a felhasználói élmény folytonos marad.
3. **Navigáció és Interakció**:
   * **Keresősáv**: A toggleSearch és hideSearch függvények kezelik a keresősáv láthatóságát, az Enter billentyű lenyomására pedig a keresési kifejezést a clothes.php-re irányítják.
   * **Kosár oldalsáv**: A toggleCart váltja a kosár oldalsáv megjelenését.
   * **Termékoldalra navigálás**: A redirectToProduct átirányít a product.php-re a termék azonosítóval.
4. **Termékoldal Slider**:
   * A showImage, prevImage és nextImage függvények a termék képgalériájának navigációját kezelik, ciklikusan váltva a képek között.
5. **Animációk és Görgetés**:
   * A handleScrollAnimations görgetéskor animálja a marketing szekciókat (visible osztály hozzáadása).
   * A scrollSuggestedLeft és scrollSuggestedRight a javasolt termékek görgetését biztosítja sima animációval.
6. **Rendelés Feldolgozása**:
   * A billing.php oldalon a placeOrderBtn eseménykezelője összegyűjti a szállítási adatokat és a kosár tartalmát, majd a localStorage-ba menti, mielőtt a pay.php-ra irányít.
   * A pay.php oldalon a rendelési összefoglaló dinamikusan töltődik be a localStorage-ból, és a payNowBtn a fizetést indítja egy fetch kéréssel a process\_payment.php felé.

### Részletes Kód Részletek:

#### Kosárkezelés

let cart = [];

function addToCart(productName, price, imageUrl, size) {

if (!size || size === '') {

alert('Kérlek, válassz méretet!');

return;

}

let existingProduct = cart.find(item => item.name === productName && item.size === size);

if (existingProduct) {

existingProduct.quantity += 1;

} else {

cart.push({ name: productName, price: price, imageUrl: imageUrl, size: size, quantity: 1 });

}

saveCartToLocalStorage();

updateReceipt();

updateCartCount();

}

function updateReceipt() {

const cartItems = document.getElementById('cart-items');

cartItems.innerHTML = '';

let total = 0;

cart.forEach(item => {

const li = document.createElement('li');

li.innerHTML = `

<div class="cart-item-content">

<img src="${item.imageUrl}" alt="${item.name}">

<span>${item.name} (${item.size})</span>

<span>${(item.price \* item.quantity).toFixed(2)} Ft</span>

</div>

<div class="quantity-controls">

<button onclick="removeFromCart('${item.name}', '${item.size}')">−</button>

<span>${item.quantity}</span>

<button onclick="increaseQuantity('${item.name}', '${item.size}')">+</button>

</div>

<button onclick="deleteFromCart('${item.name}', '${item.size}')"><i class="fas fa-times"></i></button>

`;

cartItems.appendChild(li);

total += item.price \* item.quantity;

});

document.getElementById('total').textContent = `Összesen: ${total.toFixed(2)} Ft`;

}

#### Keresősáv Kezelése

searchInput.addEventListener('keydown', function(e) {

if (e.key === 'Enter') {

e.preventDefault();

const searchTerm = this.value.trim();

if (searchTerm) {

window.location.href = `clothes.php?search=${encodeURIComponent(searchTerm)}`;

} else {

window.location.href = 'clothes.php';

}

}

});

searchToggle.addEventListener('click', function(event) {

event.stopPropagation();

searchContainer.classList.toggle('active');

if (searchContainer.classList.contains('active')) {

searchInput.focus();

}

});

#### Rendelés Feldolgozása és Fizetés

const placeOrderBtn = document.getElementById("place-order-btn");

if (placeOrderBtn) {

placeOrderBtn.addEventListener("click", function (e) {

e.preventDefault();

const name = document.getElementById("name").value.trim();

const email = document.getElementById("email").value.trim();

const phone = document.getElementById("phone").value.trim();

if (!name || !email || !phone) {

alert("Töltsd ki az összes mezőt!");

return;

}

const cartItems = cart.map(item => ({

name: item.name,

price: item.price \* item.quantity,

size: item.size,

quantity: item.quantity

}));

const total = cartItems.reduce((sum, item) => sum + item.price, 0).toFixed(2);

const orderDetails = {

cartItems: cartItems,

total: total,

delivery: document.getElementById("delivery").value,

customer: { name, email, phone }

};

localStorage.setItem("orderDetails", JSON.stringify(orderDetails));

window.location.href = "pay.php";

});

}

const payNowBtn = document.getElementById("pay-now-btn");

if (payNowBtn) {

payNowBtn.addEventListener("click", function () {

const orderDetails = JSON.parse(localStorage.getItem("orderDetails"));

fetch("php/process\_payment.php", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(orderDetails)

})

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

.then(data => {

if (data.success) {

localStorage.removeItem("cart");

localStorage.removeItem("orderDetails");

window.location.href = `succes.php?rendeles\_id=${data.rendeles\_id}`;

} else {

alert("Hiba: " + data.message);

}

});

});

}

### Bővebb Leírás:

A script.js a webshop interaktív magja, amely összeköti a frontend elemeket a backend logikával. A kosárkezelés különösen kiemelkedő, mivel a localStorage használatával a kosár tartalma megmarad az oldalak közötti navigáció során, így a felhasználó nem veszíti el a kiválasztott termékeket. A méretellenőrzés biztosítja, hogy csak érvényes adatok kerüljenek a kosárba, míg a dinamikus frissítések (pl. updateReceipt) valós időben mutatják a kosár állapotát.

A keresősáv funkcionalitása egyszerű, de hatékony: az Enter billentyű lenyomására azonnal átirányít a clothes.php-re a keresési kifejezéssel, vagy minden termékre, ha a mező üres. A navigációs elemek, mint a kosár oldalsáv és a slider, javítják a felhasználói élményt, különösen mobil nézetben.

A rendelés feldolgozása két lépésben történik: először a billing.php oldalon gyűjti össze az adatokat, majd a pay.php oldalon küldi el a process\_payment.php-nek egy AJAX kéréssel. Ez a megközelítés minimalizálja az oldal újratöltéseket, és modern, aszinkron élményt nyújt. A Swiper könyvtár integrálása a főoldalon (pl. index.php) vizuálisan vonzóvá teszi a kiemelt termékek bemutatását, míg a görgetési animációk (pl. handleScrollAnimations) dinamikussá teszik az oldalt.

### Használt Technológia:

* **JavaScript**: Alapvető logika és DOM manipuláció.
* **localStorage**: Kosár tartós tárolása.
* **Swiper**: Slider funkcionalitás.
* **Fetch API**: Aszinkron kommunikáció a backenddel.

## Dinamikus Funkcionalitás Kiegészítése

### Kosár Perszisztencia

* A localStorage biztosítja, hogy a kosár tartalma megmaradjon az oldalak között.

### Valós Idejű Frissítések

* A kosár és az összeg dinamikusan frissül minden művelet után.

### Aszinkron Kommunikáció

* A fizetési folyamat AJAX használatával történik, minimalizálva az oldalbetöltést.

## 32. Stílusok és Reszponzív Dizájn (styles.css)

A styles.css fájl a Heet Clothing webshop vizuális megjelenését és reszponzív dizájnját szabályozza. Ez a fájl definiálja az oldalak alapvető stílusait, a navigációs sávot, a termékkártyákat, a kosár és fizetési felületet, valamint az animációkat és interaktív elemeket. A CSS modern technikákat használ, mint a CSS változók (:root), a frosted glass hatás (backdrop-filter), és a Flexbox/Grid elrendezések, hogy egy esztétikus és felhasználóbarát élményt nyújtson minden eszközön.

### Főbb Stílusok és Szerepek:

1. **Globális Stílusok**:
   * **Változók**: A :root blokkban definiált CSS változók (pl. --primary-color: #c89666) egységes színpalettát biztosítanak az oldal számára.
   * **Alapbeállítások**: A \* szelektor nullázza a margókat és paddingokat, az html sima görgetést (scroll-behavior: smooth), a body pedig flexibilis elrendezést (flex-direction: column) biztosít.
2. **Navigációs Sáv (Navbar)**:
   * **Stílus**: Fekete háttér (#050505), árnyék (box-shadow), és fehér szöveg a kontraszt érdekében.
   * **Reszponzivitás**: Mobil nézetben hamburger menü (navbar-toggler) aktiválódik, a menü oldalsávként csúszik be (navbar-collapse).
   * **Keresősáv**: Alapértelmezett rejtett állapotból animáltan nyílik ki (search-container.active).
3. **Hős Szakasz (Hero)**:
   * **Kialakítás**: Magas kontrasztú háttérkép (hero-image), középre igazított szöveg (hero-content), és árnyékolt címek a jobb olvashatóságért.
   * **Interakció**: A kép hoverkor enyhén nagyít (transform: scale(1.05)).
4. **Termékkártyák és Rácsok**:
   * **Termékkártyák**: Árnyékos, lekerekített dizájn (product-card), hover effekttel (translateY(-5px)), és zoomolható képekkel.
   * **Rács**: A product-grid Grid elrendezést használ (grid-template-columns: repeat(auto-fit, minmax(250px, 1fr))), hogy alkalmazkodjon a képernyőmérethez.
5. **Kosár és Fizetési Felület**:
   * **Kosár**: A cart-item Flexbox elrendezéssel jeleníti meg a termékeket, mennyiségi gombokkal (quantity-btn) és törlés gombbal (remove-btn).
   * **Fizetés**: A #payment szekció frosted glass hatást használ (backdrop-filter: blur(15px)), reszponzív űrlapokkal és animált gombokkal (#pay-now-btn).
6. **Animációk**:
   * **Fade és Slide**: Az @keyframes fadeIn, fadeInUp, stb. animációk sima megjelenést biztosítanak (pl. .animate\_\_fadeInUp).
   * **Hover Effektek**: Gombok és kártyák transform és box-shadow átmenetekkel reagálnak az egérre.
7. **Reszponzív Dizájn**:
   * **Media Query-k**: Több töréspont (768px, 576px, stb.) biztosítja a mobil, tablet és asztali nézetek optimalizálását.
   * **Példa**: Mobil nézetben a navigáció oldalsávvá alakul, a termékkártyák kisebbek lesznek, és a szövegek mérete csökken.

### Részletes Kód Részletek:

#### Navigációs Sáv

.navbar {

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

padding: 20px 0;

background: #050505;

}

.nav-link {

color: white !important;

margin: 0 15px;

text-transform: uppercase;

}

.nav-link:hover {

color: #e1b382 !important;

}

@media (max-width: 991px) {

.navbar-collapse {

position: fixed;

top: 0;

left: -100%;

width: 250px;

height: 100%;

transition: left 0.3s ease;

}

.navbar-collapse.show {

left: 0;

}

}

#### Termékkártyák

.product-card {

background-color: var(--card-background);

border-radius: 20px;

box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);

transition: transform 0.3s ease, box-shadow 0.3s ease;

}

.product-card:hover {

transform: translateY(-5px);

box-shadow: 0 15px 30px rgba(0, 0, 0, 0.15);

}

.product-card\_\_image img {

width: 100%;

height: 100%;

object-fit: cover;

transition: transform 0.3s ease;

}

.product-card:hover .product-card\_\_image img {

transform: scale(1.05);

}

#### Fizetési Szakasz

#payment {

width: 80%;

max-width: 600px;

margin: 40px auto;

background: rgba(255, 255, 255, 0.1);

backdrop-filter: blur(15px);

padding: 20px;

border-radius: 8px;

transition: transform 0.3s ease;

}

#payment:hover {

transform: translateY(-5px);

}

#pay-now-btn {

background: var(--primary-color);

padding: 12px;

transition: background 0.3s ease, transform 0.3s ease;

}

#pay-now-btn:hover {

background: #e1b382;

transform: scale(1.05);

}

@media (max-width: 576px) {

#payment {

width: 95%;

padding: 10px;

}

}

### Bővebb Leírás:

A styles.css a webshop vizuális identitásának alapja, amelyet a --primary-color (barna árnyalat: #c89666) és a modern dizájnelemek, mint a frosted glass hatás és lekerekített sarkok határoznak meg. A navigációs sáv reszponzív kialakítása kiemelkedő: asztali nézetben vízszintes, mobil nézetben oldalsávként működik, ami javítja a használhatóságot kisebb képernyőkön.

A termékkártyák és rácsok dinamikus elrendezése (Grid és Flexbox) biztosítja, hogy a tartalom alkalmazkodjon a képernyőmérethez, miközben az animációk (pl. fadeInUp, slideUp) vizuális vonzerőt adnak. A fizetési és kosár felületek modern megjelenése a backdrop-filter használatával kiemeli a prémium érzetet, míg a hover effektek (pl. transform: scale(1.05)) interaktivitást sugallnak.

A reszponzív dizájn különösen erős: a @media lekérdezések finomhangolják a betűméreteket, paddingokat és elrendezéseket különböző eszközökre. Például mobil nézetben a recommended-item kártyák szélessége 160px-re csökken, és az árak egymás alá rendeződnek (flex-direction: column), hogy elférjenek a kisebb képernyőkön.

### Használt Technológia:

* **CSS Változók**: Egységes szín- és stíluskezelés.
* **Flexbox és Grid**: Rugalmas elrendezések.
* **Backdrop-filter**: Modern frosted glass hatás.
* **Transitions/Animations**: Sima vizuális átmenetek.
* **Media Queries**: Reszponzív dizájn minden eszközre.

## Vizuális és Funkcionális Kiemelések

### Frosted Glass Hatás

* A #payment, .profile-container, stb. szekciókban alkalmazva, modern és elegáns megjelenést biztosít.

### Animált Interakciók

* Gombok és kártyák hoverkor kiemelkednek, javítva a felhasználói visszajelzést.

### Reszponzív Optimalizáció

* Mobil nézetben kompakt, asztali nézetben tágas elrendezés a maximális olvashatóság és használhatóság érdekében.