

**VÁCI SZAKKÉPZÉSI CENTRUM
BORONKAY GYÖRGY
MŰSZAKI TECHNIKUM ÉS
GIMNÁZIUM**



VIZSGAREMEK

Wood Design Store



Konzulens: Wiesel Csaba

Készítette: Piszár Tamás
Forgács Tamás

Konzultációs lap

Vizsgázók neve: Piszár Tamás

Forgács Tamás

Vizsgaremek címe: Wood Design Store

Sorszám	A konzultáció időpontja	A konzulens aláírása
1.	2023.10.17.	
2.	2023.11.21.	
3.	2023.12.13.	
4.	2024.01.16	
5.	2024.02.20.	
6.	2024.03.12	

A vizsgaremek beadható:

Vác, 2024.....

A vizsgaremeket átvettem:

Vác, 2024.....

Konzulens

A szakképzést folytató
intézmény felelőse

Tartalomjegyzék

Konzultációs lap	2
Témaválasztás Indoklása és Dokumentációja: Fa Termékek Webáruháza	7
Témaválasztás	7
1. Fejlesztői dokumentáció	8
Áttekintés.....	8
Fő Komponensek	8
Adatbázis Kapcsolat (MySQL)	8
Felhasználókezelés.....	8
Termékkezelés	8
Kosárkezelés.....	8
Felhasználói Felület.....	8
Fontos Fájlok és Mappák	8
Kapcsolódás az adatbázishoz	9
Felhasználói bejelentkezés ellenőrzése	9
Felhasználói adatok lekérése	10
Kosárba helyezés.....	10
Kosár megjelenítése és kezelése	11
Fizetési folyamat	11
1.1 Felhasználói bejelentkezési rendszer	12
1.2 Felhasználói munkamenetek	13
1.3 Adminisztrációs felületet	14
1.4 Új adminisztrátor	15
1.5 Felhasználói adatok frissítése	16
1.6 Felhasználó végleges kitiltásának feloldása.....	17
1.7 Felhasználó korlátozása	18
1.8 Felhasználó szerkesztése	19
1.9 Admin adatok frissítése	21
1.10 Felhasználó Törlése.....	22
1.11 Felhasználók Listázása	23
1.12 Fizetés és Szállítás Űrlap	24
1.13 Készlet Módosítása Űrlap	25
1.14 Kosár Tartalom	26
Cél	26

Technikai Implementáció.....	26
Főbb Komponensek	26
Felhasznált Technológiák	26
Fejlesztői Megjegyzés	26
1.15 Bejelentkező Oldal	27
Cél	27
Technikai Implementáció.....	27
Főbb Komponensek	27
Fejlesztői Megjegyzés	27
1.16 Felhasználó Bejelentkezés	28
1.17 Felhasználó Regisztráció	30
Ez a PHP szkript a felhasználók regisztrációját végzi egy webalkalmazásban.	30
1.18 Rólunk Oldal	31
Oldalstruktúra	31
Technikai Részletek	31
1.19 Termék hozzáadása űrlap PHP kódja:.....	33
Funkciók	33
Adatmezők	33
1.20 Termékek sorrendjének módosítása	34
1.21 Termékek törlése	35
1.22 Termékek listázása	36
2. Adatszerkezet.....	37
Bevezetés	37
Rendszerkövetelmények.....	37
Folyamat	37
Változók és Objektumok	38
Hibakezelés	38
Biztonsági megfontolások.....	38
Az "Admin_users" tábla	39
Tábla struktúra	39
Használat.....	40
Példa SQL lekérdezések	40
A "termékek" tábla	40
Tábla struktúra	41

Használat.....	41
Biztonsági megfontolások.....	42
A “users” tábla	43
Használat.....	43
Biztonsági megfontolások.....	43
Példa phpMyAdmin műveletek	44
3. Felhasználói Dokumentáció	45
Áttekintés.....	45
Hogyan működik?	45
Használati utasítások	45
3.1 Főbb funkciók és tartalom	45
Használati utasítások	46
Fontos tudnivalók	46
Technikai részletek.....	46
Kosárhasználati útmutató.....	46
Egyéb Fontos Tudnivalók	46
3.2 Technikai Részletek és Frissítések.....	47
Kosárhasználati útmutató folytatása	47
Egyéb Fontos Tudnivalók (folytatás).....	47
Technikai Részletek.....	47
3.3 Jogi Tudnivalók és Adatvédelem.....	48
Kapcsolat és További Információk	48
Visszajelzés és Értékelés	48
4. Tesztelés.....	49
4.1 Tesztelési lépések	50
4.2 Tesztelési funkciók és dokumentáció	51
Felhasználó bejelentkezésének ellenőrzése:	53
5. Irodalomjegyzék.....	61
6. Mellékletek	62

Témaválasztás Indoklása és Dokumentációja: Fa Termékek Webáruháza

A Fa Termékek Webáruháza projekt célja egy online platform létrehozása, amely kifejezetten fa alapú termékekre specializálódik. Ennek az indoklásnak és dokumentációnak a célja, hogy részletesen bemutassa a projekt hátterét, céljait, funkcionalitását, valamint a technológiai megoldásokat és rendszerarchitektúrát.

Témaválasztás

A fa termékek népszerűsége és keresletük növekedése napjainkban egyre inkább észrevehetővé válik. A természetes anyagok iránti igény és a fenntarthatóság fontossága miatt a fák és fa alapú termékek újra előtérbe kerülnek a lakberendezési és kerti kiegészítők piacán. Ennek megfelelően a Fa Termékek Webáruháza projekt egy olyan válasz a piacon tapasztalható igényekre, amely lehetővé teszi a vásárlók számára, hogy könnyen megtalálják és megvásárolják a kívánt fa termékeket egy kényelmes és inspiráló online környezetben.

A témaválasztás indokai között szerepel:

1. **Növekvő kereslet:** A fa alapú termékek iránti kereslet folyamatosan növekszik, ami a projekt sikeres piaci bevezetésének előfeltétele.
2. **Fenntarthatóság:** A fa termékek egyre inkább a fenntarthatóság és környezetvédelem szimbólumává válnak, amelyek elősegítik a természetes anyagokkal való kapcsolatot.
3. **Inspiráció és kényelem:** Az online webáruházak előnyeit kihasználva, a Fa Termékek Webáruháza lehetőséget nyújt az inspiráló vásárlási élményre és a kényelmes otthoni berendezésre.

1. Fejlesztői dokumentáció

Ez a dokumentáció a webshop PHP kódját részletesen bemutatja és dokumentálja. A kód egy egyszerű webáruházat valósít meg, amely lehetővé teszi a felhasználók számára a termékek böngészését, kosárba helyezését és a fizetést.

Áttekintés

A PHP kód a következő főbb funkciókat valósítja meg:

- Felhasználók bejelentkezése és regisztrációja.
- Termékek listázása kategóriánként.
- Kosár kezelése: termékek hozzáadása, eltávolítása, kosár tartalmának megjelenítése.
- Átírányítás a fizetési oldalra.

Fő Komponensek

Adatbázis Kapcsolat (MySQL)

A `mysql` osztályt használva kapcsolódik az adatbázishoz, ahol tárolódnak a felhasználók és a termékek adatai.

Felhasználókezelés

- Bejelentkezés: Az email cím alapján ellenőrzí a felhasználó jelszavát és bejelentkezteti, ha a jelszó helyes.
- Regisztráció: Új felhasználók regisztrálása az adatbázisban.

Termékkezelés

- Termékek listázása: A készleten lévő termékek megjelenítése az adatbázisból.
- Kosárba helyezés: A felhasználók által kiválasztott termékek hozzáadása a kosárhoz.

Kosárkezelés

- Kosár megjelenítése: A felhasználó kosarában lévő termékek listázása.
- Termék eltávolítása a kosárból.

Felhasználói Felület

- HTML és CSS használata a felhasználói felület kialakításához.
- Reszponzív dizájn: A felület alkalmazkodik a különböző eszközök képernyőméretéhez.

Fontos Fájlok és Mappák

- **index.php**: A főoldal, amelyen keresztül a felhasználók böngészhetik a termékeket és kezelhetik a kosarukat.
- **login.php, register.php**: Bejelentkezési és regisztrációs oldalak.
- **logout.php**: Kijelentkezési folyamatot kezeli.
- **fizetes.php**: Az átirányítás a fizetési folyamathoz.

Kapcsolódás az adatbázishoz

A kód kezdetén inicializáljuk a PHP session-t és csatlakozunk az adatbázishoz a MySQL adatbázis szerveren. A kapcsolatot felépítjük a megadott felhasználónévvel, jelszóval és adatbázisnévvel. Ha a kapcsolat nem sikerül, akkor hibaüzenetet jelenítünk meg és leállítjuk a programot.

```
// Adatbázis kapcsolódás
$conn = new mysqli($servername, $username, $password, $dbname);
```

Felhasználói bejelentkezés ellenőrzése

A felhasználói bejelentkezés ellenőrzését egy előre készített "users" táblában tárolt adatok alapján végezzük. A felhasználói e-mail alapján keresünk a táblában, és ha megtaláljuk a felhasználót, ellenőrizzük a megadott jelszót a táblában tárolttal. Ha a jelszó helyes, beállítjuk a felhasználó nevét a session változóba. Ha nem, akkor hibaüzenetet generálunk.

```
// Felhasználó bejelentkezésének ellenőrzése
if ($user = $result->fetch_assoc()) {
    if (password_verify($password, $user['jelszo'])) {
        $_SESSION['username'] = $user['vezeteknev'] . ' ' . $user['keresztnev'];
    } else {
        $error = "Hibás jelszó.";
    }
}
```

Bejelentkezés

Email cím:

Jelszó:

Felhasználói adatok lekérése

Ha a felhasználó bejelentkezett, további adatokat kérhetünk le a felhasználóról, például vezetéknév, keresztnév, e-mail cím, utolsó bejelentkezés ideje stb. Ezeket az adatokat a "users" táblából kérjük le az e-mail cím alapján.

```
1 // Felhasználói adatok lekérése
2 if (isset($_SESSION['email'])) {
3     $email = $_SESSION['email'];
4     $sql = "SELECT vezeteknev, keresztnév, email, utolso_belepés, regisztracio_datuma FROM users WHERE email = ?";
5     $stmt = $conn->prepare($sql);
6     $stmt->bind_param("s", $email);
7     $stmt->execute();
8     $result = $stmt->get_result();
9     if ($user = $result->fetch_assoc()) {
10         $userdata = $user;
11     }
12 }
```

Termékek megjelenítése

A weboldal főoldalán a termékeket a "termékek" táblából kérjük le, amelyeket a "megjelenitesi_hely" mező értéke alapján választunk ki. A termékek listáját a felhasználó számára megjelenítjük, beleértve a termék nevét, leírását, árát és készleten lévő darabszámát.

```
// Termékek lekérése és megjelenítése
while($row = $result_keszleten->fetch_assoc()):
?>
<div class="product">
    " style="width:100%;"/>
    <h3><?php echo $row['termek_nev']; ?></h3>
    <p><?php echo $row['termek_leiras']; ?></p>
    <p>Ár: <?php echo $row['termek_ar']; ?> Ft</p>
    <p>Készleten: <?php echo $row['keszleten']; ?> db</p>
    <button onclick="addToCart('<?php echo $row['termek_nev']; ?>', <?php echo $row['termek_ar']; ?>, <?php echo $row['keszleten']; ?>, '0', '0', '0')">Kosárba</button>
</div>
<?php endwhile; ?>
```

Kosárba helyezés

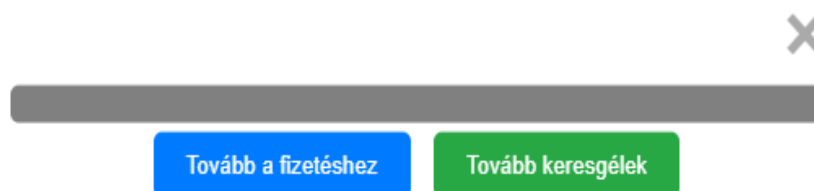
Minden termék mellett egy gomb található, amely lehetővé teszi a felhasználók számára, hogy a terméket a kosárba helyezték. A gombra kattintva a termék adatait, például a nevét, árát és készletszámát összegyűjtjük, majd ezeket az adatokat a kosárba helyezéshez tároljuk. Amennyiben a termékből még van készleten, a kosárba helyezés sikeres lesz, egyébként hibaüzenet jelenik meg.

```
// Termék hozzáadása a kosárhoz
function addToCart(productName, productPrice, productKeszleten, width, height, length)
{
    // Kosárba helyezés logikája
}
```

Kosár megjelenítése és kezelése

A felhasználói felületen egy ikonra kattintva megjelenik a kosár tartalma. Itt a felhasználók láthatják, hogy milyen termékeket helyeztek a kosárba, a mennyiségüket és az egyes termékek árát. Lehetőség van a termékek törlésére a kosárból is. A kosár tartalmát egy model ablakban jelenítjük meg.

```
// Kosár tartalmának megjelenítése és kezelése
function updateCartUI() {
    // Kosár tartalmának frissítése a felhasználói felületen
}
```



Fizetési folyamat

A felhasználók tovább léphetnek a fizetési folyamathoz, amelyre egy gomb megnyomásával kerül sor. Ekkor átirányítjuk őket a "fizetes.php" oldalra, ahol a vásárlást folytathatják.

```
function proceedToCheckout() {
    // Átirányítás a fizetes.php oldalra
    window.location.href = "fizetes.php";
}
```

Eseménykezelők:

```
document.addEventListener('DOMContentLoaded', function() {
    // Eseménykezelők regisztrálása az oldal betöltésekor
});
```

1.1 Felhasználói bejelentkezési rendszer

Ez a PHP kód egy egyszerű felhasználói bejelentkezési rendszert valósít meg: Admin

```
2 session_start();
3 $error = "";
4
5 $db = new mysqli('localhost', 'root', '', 'users');
6
7 if ($_SERVER["REQUEST_METHOD"] == "POST") {
8     $username = $_POST['username'];
9     $password = $_POST['password'];
10
11     $query = $db->prepare("SELECT * FROM admin_users WHERE username = ?");
12     $query->bind_param('s', $username);
13     $query->execute();
14     $result = $query->get_result();
15
16     if ($result->num_rows == 1) {
17         $user = $result->fetch_assoc();
18         if (password_verify($password, $user['password'])) {
19             $_SESSION['loggedin'] = true;
20             $_SESSION['username'] = $username;
21             header('Location: admin.php');
22             exit;
23         } else {
24             // Hibás jelszó
25             $error = "Hibás felhasználónév vagy jelszó!";
26         }
27     } else {
28         // Hibás felhasználónév
29         $error = "Hibás felhasználónév vagy jelszó!";
30     }
31 }
```

1. **session_start():** Ez a függvény indítja el a munkamenetet, ami lehetővé teszi az adatok tárolását az egyes látogatók számára.
2. **\$error változó:** Ez a változó a bejelentkezési hibák kezelésére szolgál. Ha valamilyen hiba történik (pl. helytelen felhasználónév vagy jelszó), akkor ebbe a változóba kerül a hibaüzenet.
3. **Adatbázis csatlakozás létrehozása:** Az adatbázis kapcsolatot hoz létre az adatbázis szerverrel a mysqli osztály segítségével. A 'localhost', 'root', '', 'users' paraméterek az adatbázis szerver elérési útját, felhasználónevét, jelszavát és az adatbázis nevét adják meg. Ezt a kapcsolatot a \$db változó tárolja.
4. **POST kérés kezelése:** Ellenőrzi, hogy a kérés POST metódussal érkezett-e a szerverre. Ha igen, akkor a beérkező felhasználónév és jelszó értékeket elmenti a \$username és \$password változókba.
5. **Felhasználó azonosítása az adatbázisban:** A felhasználó által megadott felhasználónév alapján lekérdezi az adatbázisból az adott felhasználó adatait. Ha a lekérdezés eredménye egyetlen sor, akkor a felhasználó létezik az adatbázisban, és a jelszava helyes.
6. **Jelszó ellenőrzése:** A password_verify() függvény segítségével összehasonlítja a felhasználó által megadott jelszót a tárolt jelszó hash-szel az adatbázisban. Ha a jelszavak megegyeznek, akkor a bejelentkezés sikeres.
7. **Munkamenet változók beállítása:** Ha a bejelentkezés sikeres, akkor az \$_SESSION['loggedin'] és \$_SESSION['username'] változókat beállítja, majd átirányítja a felhasználót az adminisztrációs felületre (admin.php).
8. **Hibakezelés:** Ha a felhasználónév vagy a jelszó helytelen, akkor a \$error változóban tárolja a hibaüzenetet, amit később megjeleníthet a felhasználónak.

1.2 Felhasználói munkamenetek

Ez a PHP kód a felhasználói munkamenetek lezárását és a felhasználó kijelentkeztetését végzi el:

```
2 session_start();
3
4 $_SESSION = array();
5
6 if (ini_get("session.use_cookies")) {
7     $params = session_get_cookie_params();
8     setcookie(session_name(), '', time() - 42000,
9         $params["path"], $params["domain"],
10        $params["secure"], $params["httponly"]
11    );
12 }
13
14 session_destroy();
15
16 header('Location: admin_bejelentkezes.php');
17 exit;
18
```

1. **session_start()**: Elindítja a munkamenetet, hogy a PHP tudja kezelni a munkamenet változókat.
2. **\$_SESSION törlése**: A **\$_SESSION** globális tömb ürítése révén minden munkamenet változót töröl a felhasználói munkamenetből. Ez azért fontos, hogy teljesen megszüntesse a felhasználó bejelentkezett állapotát és más munkamenet-specifikus adatokat.
3. **session_get_cookie_params()** és **setcookie()**: Ezek a függvények beállítják a munkamenet cookie-jának élettartamát, így a böngészőben tárolt munkamenet azonosító hamarosan lejár és érvénytelen lesz. Ez egyfajta biztonsági intézkedés, hogy megakadályozza a lejárt munkamenet cookie használatát.
4. **session_destroy()**: Ez a függvény teljesen megszünteti a munkamenetet, beleértve a munkamenet kezelő szerveroldali változókat és a munkamenet azonosító törlését is.
- 5.

Bejelentkezés

<p>Email cím:</p> <input type="text"/> <p>Jelszó:</p> <input type="password"/> <div style="text-align: center;">Bejelentkezés Admin Bejelentkezés</div>	<p>Webadmin Bejelentkezés</p> <p>Felhasználónév</p> <input type="text"/> <p>Jelszó</p> <input type="password"/> Bejelentkezés
--	---

- 6.
7. **header()** és **Location átirányítás**: Ez a sor átirányítja a felhasználót a bejelentkező oldalra (**admin_bejelentkezes.php**). Ez gyakorlatilag visszairányítja a felhasználót a bejelentkező oldalra, miután sikeresen kijelentkezett.
8. **exit()**: Megállítja a script futását, miután az átirányítás megtörtént, így további felesleges kód futtatása nem következik be.

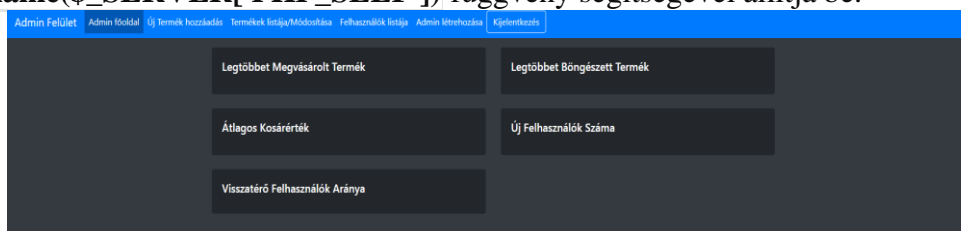
9.

1.3 Adminisztrációs felület

Ez a PHP kód egy adminisztrációs felületet hoz létre, amely biztonságos bejelentkezést igényel:

```
2 include('auth.php');
3
4 if ($_SERVER["REQUEST_METHOD"] == "POST") {
5     $username = $_POST["username"];
6     $password = $_POST["password"];
7
8     include 'db_config.php';
9     $sql = "SELECT * FROM admin_users WHERE username = ?";
10    $stmt = $conn->prepare($sql);
11    $stmt->bind_param("s", $username);
12    $stmt->execute();
13    $result = $stmt->get_result();
14
15    if ($result->num_rows == 1) {
16        $row = $result->fetch_assoc();
17        if (password_verify($password, $row["password"])) {
18            $_SESSION["loggedin"] = true;
19            $_SESSION["username"] = $username;
20            header("Location: admin_felulet.php");
21            exit;
22        } else {
23            $login_err = "Hibás felhasználónév vagy jelszó!";
24        }
25    } else {
26        $login_err = "Hibás felhasználónév vagy jelszó!";
27    }
28 }
```

1. **Authentikációs ellenőrzés (include('auth.php')):** Ez a sor hivatkozik egy olyan fájlra, amely tartalmazza az autentikációs ellenőrzést végző kódot. Ez valószínűleg azért van ott, hogy biztosítsa, hogy csak bejelentkezett felhasználók férjenek hozzá az adminisztrációs felülethez.
2. **POST kérés kezelése:** Ellenőrzi, hogy a kérés POST metódussal érkezett-e a szerverre. Ha igen, akkor a beérkező felhasználónév és jelszó értékeket elmenti a `$username` és `$password` változókba.
3. **Adatbázis csatlakozás létrehozása:** Az adatbázis kapcsolatot hoz létre az adatbázis szerverrel a `db_config.php` fájlban tárolt beállítások alapján. Ezután lekérdezi az adatbázisból az adminisztrátor felhasználó adatait a megadott felhasználónév alapján.
4. **Jelszó ellenőrzése:** Ha van egyetlen találat a felhasználónevet tartalmazó rekordok között, akkor ellenőrzi a megadott jelszót a tárolt, titkosított jelszóval. Ha a megadott jelszó helyes, beállítja a munkamenet változókat a bejelentkezett állapot jelezésére, majd átirányítja a felhasználót az adminisztrációs felületre.
5. **Átirányítás a bejelentkezési oldalra:** Ha a bejelentkezés sikertelen, akkor egy hibaüzenetet állít be a `$login_err` változóban, amit később megjeleníthet a felhasználónak.
6. **HTML kód az adminisztrációs felülethez:** Ez a rész a HTML kódot tartalmazza az adminisztrációs felület megjelenítéséhez. A Bootstrap keretrendszer segítségével történik a stílusozás, ami modern és responszív megjelenést biztosít az oldalnak.
7. **Navbar:** Az adminisztrációs felület felső sávjában található menüpontok, amelyek között lehet navigálni. Az aktív oldalt jelző osztályt a `basename($_SERVER['PHP_SELF'])` függvény segítségével állítja be.



1.4 Új adminisztrátor

Ez a PHP kód egy új adminisztrátor felhasználót hoz létre az adatbázisban, és egy egyszerű űrlapot jelenít meg az adminisztrációs felületen a felhasználó létrehozásához:

```
2 include('auth.php');
3 $conn = new mysqli('localhost', 'root', '', 'users');
4
5 $success = $error = '';
6 if ($SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['username']) && isset($_POST['password'])) {
7     $username = string|null mysqli::$connect_error
8     $password = https://www.php.net/manual/class.mysqli.php, PASSWORD_DEFAULT);
9
10    if ($conn->connect_error) {
11        die("Connection failed: " . $conn->connect_error);
12    }
13
14    $stmt = $conn->prepare("SELECT id FROM admin_users WHERE username = ?");
15    $stmt->bind_param("s", $username);
16    $stmt->execute();
17    $result = $stmt->get_result();
18    if ($result->num_rows > 0) {
19        $error = 'Ez a felhasználónév már foglalt.';
20    } else {
21
22        $stmt = $conn->prepare("INSERT INTO admin_users (username, password) VALUES (?, ?)");
23        $stmt->bind_param("ss", $username, $password);
24        if ($stmt->execute()) {
25            $success = 'Az admin felhasználó sikeresen létrehozva.';
26        } else {
27            $error = 'Hiba történt a felhasználó létrehozása közben.';
28        }
29    }
30    $conn->close();
31 }
```

1. **Authentikációs ellenőrzés (include('auth.php')):** Ez a sor hivatkozik egy olyan fájlra, amely tartalmazza az autentikációs ellenőrzést végző kódot. Ez biztosítja, hogy csak bejelentkezett felhasználók férjenek hozzá az adminisztrációs felülethez.
2. **Adatbázis csatlakozás létrehozása:** Az adatbázis kapcsolatot hoz létre az adatbázis szerverrel a **mysqli** osztály segítségével. Az adatbázis szerver elérési útját, felhasználónevét, jelszavát és az adatbázis nevét a megfelelő helyeken kell megadni.
3. **POST kérés kezelése és felhasználó létrehozása:** Ellenőrzi, hogy a kérés POST metódussal érkezett-e a szerverre és hogy a **username** és **password** mezők ki vannak-e töltve. Ezután ellenőrzi, hogy a megadott felhasználónév már létezik-e az adatbázisban. Ha nem létezik, akkor a megadott felhasználónevet és jelszót hozzáadja az adatbázishoz.
4. **HTML kód az új adminisztrátor létrehozásához:** Ez a rész a HTML űrlapot tartalmazza az adminisztrátor felhasználó létrehozásához. A Bootstrap keretrendszer segítségével történik a stílusozás, ami modern és responszív megjelenést biztosít az oldalnak.
5. **Siker és hibaiüzenetek megjelenítése:** A sikeres vagy sikertelen művelet után megjelenik egy üzenet a felhasználónak a **\$success** vagy **\$error** változóban tárolt üzenet alapján.

1.5 Felhasználói adatok frissítése

Ez a PHP kód felelős az adatbázisban egy felhasználó adatainak frissítéséért:

```
1 <?php
2 include('auth.php');
3 $conn = new mysqli('localhost', 'root', '', 'users');
4
5 if ($conn->connect_error) {
6     die("Connection failed: " . $conn->connect_error);
7 }
8
9 $id = $_POST['id'] ?? null;
10 $vezeteknev = $_POST['vezeteknev'] ?? '';
11 $keresztnev = $_POST['keresztnev'] ?? '';
12 $lakcim = $_POST['lakcim'] ?? '';
13 $telefonszam = $_POST['telefonszam'] ?? '';
14 $email = $_POST['email'] ?? '';
15
16 if ($id) {
17     $stmt = $conn->prepare("UPDATE users SET vezeteknev = ?, keresztnev = ?, lakcim = ?, telefonszam = ?, email = ? WHERE id = ?");
18     $stmt->bind_param("sssssi", $vezeteknev, $keresztnev, $lakcim, $telefonszam, $email, $id);
19
20     if ($stmt->execute()) {
21         echo "A felhasználó adatai sikeresen frissítve.";
22         header("location: felhasznalok_listazasa.php");
23         exit();
24     } else {
25         echo "Hiba történt: " . $conn->error;
26     }
27 } else {
28     echo "Nincs megadva felhasználói azonosító.";
29 }
30
31 $conn->close();
32 >>
```

1. **Authentikációs ellenőrzés (include('auth.php')):** Ez a sor hivatkozik egy olyan fájlra, amely tartalmazza az autentikációs ellenőrzést végző kódot. Ez biztosítja, hogy csak bejelentkezett felhasználók férjenek hozzá az adminisztrációs funkciókhoz.
2. **Adatbázis csatlakozás létrehozása:** Az adatbázis kapcsolatot hoz létre az adatbázis szerverrel a **mysqli** osztály segítségével. Ezután ellenőrzi, hogy sikeres volt-e a kapcsolódás. Ha nem sikerült, akkor kiírja a kapcsolódási hibát és megszakítja a script futását.
3. **POST adatok kezelése:** A beérkező POST adatokat elmenti változókba, vagy alapértelmezett értékeket állít be, ha a POST adatok nem érkeztek meg.
4. **Adatok frissítése az adatbázisban:** Ellenőrzi, hogy van-e megadva felhasználói azonosító. Ha van, akkor az adott felhasználó adatait frissíti az adatbázisban a kapott adatok alapján.
5. **Siker és hibaüzenetek megjelenítése:** Ha a frissítés sikeres, kiírja, hogy "A felhasználó adatai sikeresen frissítve." és átirányítja a felhasználót a felhasználók listázása oldalra. Ha hiba történt, kiírja az adatbázis hibáját.
6. **Adatbázis kapcsolat lezárása:** Bezárja az adatbázis kapcsolatot a **close()** metódussal.

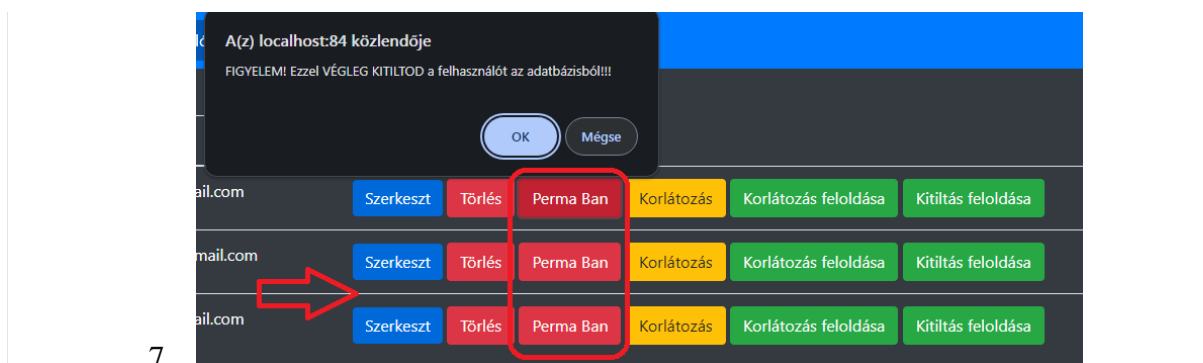
Admin Felület	Admin főoldal	Új Termék hozzáadás	Termékek listája/Módosítása	Felhasználók listája	Admin létrehozása	Kijelentkezés
Felhasználók Listázása						
Vezetéknév	Keresztnev	Lakcím	Telefonszám	Email		
pi	to	dp	06 123456789	asd@gmail.com	Szerkeszt	Törölés
vez	ker	dp	06 123456789	email@gmail.com	Perma Ban	Korlátozás
asd1	asd2	asd utca 5	123456789	asd@gmail.com	Korlátozás feloldása	Kitiltás feloldása

1.6 Felhasználó végleges kitiltásának feloldása

Ez a PHP kód felelős egy felhasználó végleges kitiltásának feloldásáért az adatbázisban. Nézzük meg részleteiben:

```
2 ini_set('display_errors', 1);
3 error_reporting(E_ALL);
4 include('auth.php');
5 require_once 'db_config.php';
6
7 if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET['id']) && is_numeric($_GET['id'])) {
8     $id = $_GET['id'];
9
10    $sql_update = "UPDATE users SET vegleges_kitiltva = 0 WHERE id = ?";
11    if($stmt_update = $conn->prepare($sql_update)) {
12        $stmt_update->bind_param("i", $id);
13        if($stmt_update->execute()) {
14            echo "A felhasználó végleges kitiltása sikeresen feloldva.";
15        } else {
16            echo "Hiba történt a végleges kitiltás feloldása során: " . $stmt_update->error;
17        }
18        $stmt_update->close();
19    } else {
20        echo "Nem sikerült előkészíteni az SQL frissítést: " . $conn->error;
21    }
22 } else {
23     echo "Hibás kérés. A kitiltás feloldásához szükséges egy érvényes felhasználói ID.";
24 }
25 $conn->close();
```

1. **Hibaüzenetek bekapcsolása** (`ini_set('display_errors', 1);` `error_reporting(E_ALL);`): Ezek a sorok bekapcsolják és beállítják az összes hibajelzést, hogy az összes hibát és figyelmeztetést megjelenítsék, így könnyebben lehet hibákat keresni és javítani.
2. **Authentikációs ellenőrzés** (`include('auth.php')`): Ez a sor hivatkozik egy olyan fájlra, amely tartalmazza az autentikációs ellenőrzést végző kódot. Ez biztosítja, hogy csak bejelentkezett felhasználók férjenek hozzá az adminisztrációs funkciókhoz.
3. **Adatbázis kapcsolat létrehozása** (`require_once 'db_config.php';`): Ez a sor hivatkozik a `db_config.php` fájlra, amely tartalmazza az adatbázishoz való kapcsolódáshoz szükséges beállításokat és kódokat.
4. **GET kérés kezelése és végleges kitiltás feloldása**: Ellenőrzi, hogy a kérés GET metódussal érkezett-e a szerverre, és hogy az érkező kérés tartalmazza-e a `id` paramétert, amely egy érvényes felhasználói azonosítót jelöl. Ha ezek a feltételek teljesülnek, akkor előkészíti az SQL lekérdezést a végleges kitiltás feloldására és végrehajtja azt az adatbázison keresztül. A sikeres végrehajtás után kiírja a sikerüzenetet, ha hiba történik, akkor pedig a hibaüzenetet.
5. **Hibás kérés kezelése**: Ha a GET kérés nem tartalmaz érvényes felhasználói azonosítót, akkor kiírja a hibaüzenetet.
6. **Adatbázis kapcsolat lezárása**: Bezárja az adatbázis kapcsolatot a `close()` metódussal.



1.7 Felhasználó korlátozása

Ez a kód egy felhasználó korlátozását teszi lehetővé az adatbázisban, és egy űrlapot jelenít meg az adminisztrációs felületen a korlátozás beállításához:

```

2  include('auth.php');
3  require_once 'db_config.php';
4  date_default_timezone_set('Europe/Budapest');
5
6  if ($_SERVER["REQUEST_METHOD"] == "POST") {
7      $id = $_POST['id'] ?? null;
8      $ban_vege = $_POST['ban_vege'] ?? null;
9
10     if(isset($id) && is_numeric($id) && isset($ban_vege)) {
11         $sql_update = "UPDATE users SET ban_vege = ? WHERE id = ?";
12         if($stmt_update = $conn->prepare($sql_update)) {
13             $stmt_update->bind_param("si", $ban_vege, $id);
14             if($stmt_update->execute()) {
15                 echo "A felhasználó sikeresen korlátozva " . $ban_vege . "-ig.";
16             } else {
17                 echo "Hiba történt a korlátozás során: " . $stmt_update->error;
18             }
19             $stmt_update->close();
20         } else {
21             echo "Nem sikerült előkészíteni az SQL frissítést: " . $conn->error;
22         }
23         $conn->close();
24     } else {
25         echo "Hibás kérés.";
26     }
27 }

```

1. **Authentikációs ellenőrzés** (`include('auth.php')`): Ez a sor hivatkozik egy olyan fájlra, amely tartalmazza az autentikációs ellenőrzést végző kódot. Ez biztosítja, hogy csak bejelentkezett felhasználók férjenek hozzá az adminisztrációs funkciókhoz.
2. **Adatbázis kapcsolat létrehozása** (`require_once 'db_config.php';`): Ez a sor hivatkozik a `db_config.php` fájlra, amely tartalmazza az adatbázishoz való kapcsolódáshoz szükséges beállításokat és kódokat.
3. **POST adatok kezelése és korlátozás beállítása**: Ellenőrzi, hogy a kérés POST metódussal érkezett-e a szerverre, és hogy a POST adatok tartalmazzák-e a szükséges `id` és `ban vege` paramétereket. Ha minden adat rendelkezésre áll, előkészíti az SQL lekérdezést a felhasználó korlátozásának beállítására az adatbázisban, majd végrehajtja azt. Sikeres végrehajtás esetén kiírja a sikerüzenetet, ellenkező esetben a hibaüzenetet.
4. **Hibás kérés kezelése**: Ha a POST kérés nem tartalmaz minden szükséges adatot, kiírja a hibaüzenetet.
5. **Adatbázis kapcsolat lezárása**: Bezárja az adatbázis kapcsolatot a `close()` metódussal.

6. **HTML űrlap a korlátozás beállításához:** Ez a rész a HTML űrlapot tartalmazza a felhasználó korlátozásának beállításához. A felhasználó az űrlapon kiválaszthatja a korlátozás végét, majd elküldheti az űrlapot a beállítás végrehajtásához.

Felhasználók Listázása									
Vezetéknév	Keresztnév	Lakcím	Telefonszám	Email	Szerkeszt	Töröl	Perma Ban	Korlátozás	Korlátozás feloldása
pi	to	dp	06 123456789	asd@gmail.com	Szerkeszt	Töröl	Perma Ban	Korlátozás	Korlátozás feloldása
vez	ker	dp	06 123456789	email@gmail.com	Szerkeszt	Töröl	Perma Ban	Korlátozás	Korlátozás feloldása
asd1	asd2	asd utca 5	123456789	asd@gmail.com	Szerkeszt	Töröl	Perma Ban	Korlátozás	Korlátozás feloldása

Ez a kód lehetővé teszi egy felhasználó korlátozását az adatbázisban, és egy felhasználóbarát űrlapot biztosít a korlátozás beállításához az adminisztrációs felületen.

Korlátozás vége:

Felhasználó korlátozása

1.8 Felhasználó szerkesztése

Ez a kód egy felhasználó szerkesztésére szolgáló űrlapot jelenít meg az adminisztrációs felületen:

```
2 include('auth.php');
3 $conn = new mysqli('localhost', 'root', '', 'users');
4
5 if ($conn->connect_error) {
6     die("Connection failed: " . $conn->connect_error);
7 }
8
9 $id = isset($_GET['id']) ? (int) $_GET['id'] : null;
10
11 if (!$id) {
12     echo "Nincs megadva id.";
13     exit;
14 }
15
16 $stmt = $conn->prepare("SELECT * FROM users WHERE id = ?");
17 $stmt->bind_param("i", $id);
18 $stmt->execute();
19 $result = $stmt->get_result();
20
21 if (!$user = $result->fetch_assoc()) {
22     echo "Nincs ilyen id-jű felhasználó.";
23     exit;
24 }
```

1. **Authentikációs ellenőrzés (include('auth.php')):** Ez a sor hivatkozik egy olyan fájlra, amely tartalmazza az autentikációs ellenőrzést végző kódot. Ez biztosítja, hogy csak bejelentkezett felhasználók férjenek hozzá az adminisztrációs funkciókhoz.
2. **Adatbázis kapcsolat létrehozása:** A `$conn` változó egy adatbázis kapcsolatot hoz létre az adatbázis szerverrel. Ha a kapcsolat létrejön, akkor folytatódik a kód futása, ha nem, akkor kiírja a kapcsolódási hibát.
3. **GET adatok kezelése:** Az `$id` változóba eltároljuk a `$_GET['id']` értékét, ha az létezik és szám típusú. Ha nincs megadva `id`, akkor kiírja, hogy nincs megadva `id`, és kilép a kódból.
4. **Felhasználó lekérdezése:** Létrehoz egy SQL lekérdezést a `users` táblából az adott `id` alapján. Ezután végrehajtja a lekérdezést, és az eredményt eltárolja a `$user` változóban.

5. **Adatok megjelenítése az űrlapon:** Az űrlapon megjelenítjük a felhasználó adatait az input mezőkben. Az alapértelmezett értékek a `$user` tömbből származnak, amelyet az adatbázisból kaptunk.
6. **HTML űrlap a felhasználó adatainak szerkesztéséhez:** Ez a rész tartalmazza az űrlapot, ahol a felhasználó adatait szerkeszthetjük. Az input mezőkben megjelenítjük a felhasználó jelenlegi adatait, és azokat lehet szerkeszteni.

7.



Felhasználó Szerkesztése

Vezetéknév
peida

Keresztnév
nev

Lakcím
Pest Buda

Telefonszám
+3670789365

Email
peida@gmail.com

Frissítés

1.9 Admin adatok frissítése

Ez a PHP kód az adatbázisban egy felhasználó adatainak frissítését végzi el az adminisztrációs felületen keresztül:

```
2 include('auth.php');
3 $conn = new mysqli('localhost', 'root', '', 'users');
4
5 if ($conn->connect_error) {
6     die("Connection failed: " . $conn->connect_error);
7 }
8
9 $id = $_POST['id'] ?? null;
10 $vezeteknev = $_POST['vezeteknev'] ?? '';
11 $keresztnev = $_POST['keresztnev'] ?? '';
12 $lakcim = $_POST['lakcim'] ?? '';
13 $telefonszam = $_POST['telefonszam'] ?? '';
14 $email = $_POST['email'] ?? '';
15
16 if ($id) {
17     $stmt = $conn->prepare("UPDATE users SET vezeteknev = ?, keresztnev = ?, lakcim = ?, telefonszam = ?, email = ? WHERE id = ?");
18     $stmt->bind_param("sssssi", $vezeteknev, $keresztnev, $lakcim, $telefonszam, $email, $id);
19
20     if ($stmt->execute()) {
21         echo "A felhasználó adatai sikeresen frissítve.";
22         header("Location: felhasznalok_listazasa.php");
23         exit();
24     } else {
25         echo "Hiba történt: " . $conn->error;
26     }
27 } else {
28     echo "Nincs megadva felhasználói azonosító.";
29 }
```

1. **Authentikációs Ellenőrzés:** Az első sor az **auth.php** fájl beillesztését végzi, amelyben az autentikációs ellenőrzés kódja található. Ez biztosítja, hogy csak bejelentkezett felhasználók férhessenek hozzá az adminisztrációs funkciókhoz.
2. **Adatbázis Kapcsolat Létrehozása:** Ezután a kód létrehoz egy kapcsolatot az adatbázissal a **mysqli** osztály segítségével. Ha a kapcsolat létrejött, a kód folytatódik, ha nem, akkor kiírja a kapcsolódási hibát és megszakítja a futást.
3. **POST Adatok Feldolgozása:** Az **\$_POST** tömbből kinyeri az adatokat, például az azonosítót, vezetéknévet, keresztnévet, lakcímet, telefonszámot és e-mail címet. Ezeket az adatokat az űrlapról kapja meg, amikor azokat elküldik.
4. **Adatok Frissítése az Adatbázisban:** Ezután egy **UPDATE** SQL lekérdezést hajt végre az adatbázison, amely frissíti a megadott azonosítójú felhasználó adatait. A **bind_param** metódus segítségével paraméterezett lekérdezést készít, hogy elkerülje a SQL injection támadásokat.
5. **Siker/Fiasco Üzenetek Megjelenítése:** Ha a lekérdezés sikeresen végrehajtódott, akkor kiírja a "A felhasználó adatai sikeresen frissítve." üzenetet, és átirányítja a felhasználót a **felhasznalok_listazasa.php** oldalra. Ha hiba történt, akkor kiírja a hibaüzenetet.
6. **Adatbázis Kapcsolat Bezárása:** Végül bezárja az adatbázis kapcsolatot a **close** metódussal, hogy ne terhelje tovább a szerver erőforrásait.

Pelda	Nev	Pest Buda	+3670542789	pelda@gmail.com	Szerkeszt
-------	-----	-----------	-------------	-----------------	-----------

Ez a kód hatékonyan frissíti a felhasználó adatait az adatbázisban az adminisztrációs felületen keresztül, és kezeli az esetleges hibákat is.

1.10 Felhasználó Törlése

Ez a PHP szkript lehetővé teszi felhasználók törlését az adatbázisból az azonosítójuk alapján:

```
2  include('auth.php');
3  require_once 'db_config.php';
4
5  if(isset($_GET['id']) && is_numeric($_GET['id'])) {
6      $id = $_GET['id'];
7
8      $sql = "DELETE FROM users WHERE id = $id";
9
10     if ($conn->query($sql) === TRUE) {
11         echo "A felhasználó sikeresen törölve.";
12     } else {
13         echo "Hiba történt a törlés során: " . $conn->error;
14     }
15
16     $conn->close();
17 } else {
18     echo "Hibás kérés.";
19 }
20 header('Location: felhasznalok_listazasa.php');
21 exit();
```

Használati Utasítások

1. **Bejelentkezés:** Győződjön meg róla, hogy bejelentkezett a rendszerbe és rendelkezik megfelelő jogosultságokkal a felhasználók törléséhez.
2. **Kérés Küldése:** Nyissa meg a böngészőt, és adja meg a szkript URL-jét a következő formátumban: **[http://weboldal.hu/torles.php?id=\[felhasználó azonosítója\]](http://weboldal.hu/torles.php?id=[felhasználó azonosítója])**. A **[felhasználó azonosítója]** helyére írja be a törlendő felhasználó egyedi azonosítóját az adatbázisban.
3. **Megerősítés:** Nyomja meg az "Enter" billentyűt a kérés elküldéséhez.

Figyelmeztetések

- **Azonosító Helyesbítése:** Ellenőrizze, hogy a megadott azonosítóval rendelkező felhasználó valóban törölhető-e, és nincs-e kapcsolatban más adatokkal, amelyeket meg kell őrizni.

1.11 Felhasználók Listázása

Ez a PHP szkript lehetővé teszi a rendszerben tárolt felhasználók listázását, és lehetőséget biztosít a felhasználókkal kapcsolatos műveletek elvégzésére, például szerkesztésre, törlésre, korlátozásra és kitiltásra.

```
7
8 $conn = new mysqli($servername, $username, $password, $dbname);
9
10 if ($conn->connect_error) {
11     die("Connection failed: " . $conn->connect_error);
12 }
13
14 $sql = "SELECT id, vezeteknev, keresztnév, lakcim, telefonszam, email FROM users";
15
16 $result = $conn->query($sql);
17
18 if ($result === false) {
19     die("Hiba történt: " . $conn->error);
20 }
21
22 $users = [];
23 while ($row = $result->fetch_assoc()) {
24     $users[] = $row;
25 }
```

Használati Utasítások

1. **Bejelentkezés:** Győződjön meg róla, hogy bejelentkezett a rendszerbe, és rendelkezik a megfelelő jogosultságokkal a felhasználók kezeléséhez.
2. **Felhasználók Listázása:** Nyissa meg a böngészőt, és adja meg a szkript URL-jét a következő formátumban: **http://weboldal.hu/felhasznalok_listazasa.php**.
3. **Felhasználók Megtekintése:** A felhasználók listája megjelenik a weboldalon táblázatos formában. A táblázat minden sorában egy felhasználó adatai találhatók, beleértve a vezetéknévét, keresztnévét, lakcímét, telefonszámot és e-mail címet.
4. **Felhasználókkal Kapcsolatos Műveletek:** Minden felhasználó sorának végén különböző gombok találhatók, amelyek lehetővé teszik a felhasználókkal kapcsolatos műveletek elvégzését:
 - **Szerkesztés:** A "Szerkeszt" gombra kattintva a felhasználó adatait szerkesztheti.
 - **Törlés:** A "Törlés" gombra kattintva a felhasználót törölheti az adatbázisból. A törlés előtt megjelenik egy megerősítő párbeszédablak.
 - **Perma Ban:** A "Perma Ban" gombra kattintva a felhasználót véglegesen kitilthatja az adatbázisból. A kitiltás előtt megjelenik egy megerősítő párbeszédablak.
 - **Korlátozás:** A "Korlátozás" gombra kattintva korlátozhatja a felhasználó hozzáférését bizonyos funkciókhoz.
 - **Korlátozás Feloldása:** A "Korlátozás feloldása" gombra kattintva feloldhatja a felhasználó korlátozását. A feloldás előtt megjelenik egy megerősítő párbeszédablak.
 - **Kitiltás Feloldása:** A "Kitiltás feloldása" gombra kattintva feloldhatja a felhasználó kitiltását. A feloldás előtt megjelenik egy megerősítő párbeszédablak.
5. **Kijelentkezés:** Ha végezte a műveleteket, kattintson a "Kijelentkezés" gombra a biztonságos kijelentkezéshez.

Figyelmeztetések

- **Adatvesztés Veszélye:** A "Törlés" és "Perma Ban" műveletek véglegesen eltávolítják a felhasználót az adatbázisból, és nem lehet visszavonni. Győződjön meg róla, hogy ezeket a műveleteket csak akkor hajtja végre, ha biztos benne, hogy szükséges.

1.12 Fizetés és Szállítás Űrlap

Ez a PHP szkript felelős a felhasználóknak a szállítási és fizetési információk megadásáért egy webshopban.

```
2  $db = new mysqli('localhost', 'root', '', 'users');
3
4  if (isset($_SESSION['userid'])) {
5      $userid = $_SESSION['userid'];
6      $query = $db->prepare("SELECT * FROM users WHERE id = ?");
7      $query->bind_param('i', $userid);
8      $query->execute();
9      $result = $query->get_result();
10     $user = $result->fetch_assoc();
11
12     echo '
13     <input type="text" id="vezeteknev" name="vezeteknev" value="' . $user['vezeteknev'] . '" required>
14     <input type="text" id="keresztnev" name="keresztnev" value="' . $user['keresztnev'] . '" required>
15     <input type="text" id="lakcim" name="lakcim" value="' . $user['lakcim'] . '" required>
16     <input type="tel" id="telefonszam" name="telefonszam" value="' . $user['telefonszam'] . '" required>
17     <input type="email" id="email" name="email" value="' . $user['email'] . '" required>
18     ';
19 }
```

Használati Utasítások

1. **Bejelentkezés:** Amennyiben be van jelentkezve a rendszerbe, a szükséges adatok részben előre kitöltődnek a felhasználói adatok alapján.
2. **Szállítási és Fizetési Adatok Megadása:** Töltse ki a szállítási és fizetési információkhoz szükséges mezőket az űrlapon. Szükséges mezők a következők: vezetéknév, keresztnév, e-mail cím, telefonszám, adószám, város, utca, irányítószám, szállítási és fizetési módszer választása.
3. **Kosár Tartalmának Megtekintése:** A jobb oldalon megjelenítésre kerül a kosár tartalma, amely tartalmazza a kiválasztott termékeket, azok darabszámát és árát.
4. **Összesítés és Továblépés:** Az összesítésben látható a kosárban található termékek végösszege. Ha minden szükséges adatot megadott, kattintson a "Tovább a fizetési oldalra" gombra a további lépésekhez.

1.13 Készlet Módosítása Űrlap

Ez a PHP szkript lehetővé teszi a felhasználók számára egy termék készletének módosítását egy webáruházban.

```
2  require_once 'db_config.php';
3
4  if (isset($_GET['termek_id'])) {
5      $termek_id = $_GET['termek_id'];
6
7      $sql = "SELECT termék_nev, termék_keszlet FROM termekek WHERE termék_id = ?";
8      $stmt = $conn->prepare($sql);
9      $stmt->bind_param("i", $termek_id);
10     $stmt->execute();
11     $result = $stmt->get_result();
12     if ($row = $result->fetch_assoc()) {
13         $termek_nev = $row['termek_nev'];
14         $aktualis_keszlet = $row['termek_keszlet'];
15     }
16 }
17
18 if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['modosit'])) {
19     $uj_keszlet = $_POST['uj_keszlet'];
20     $updateSql = "UPDATE termekek SET termék_keszlet = ? WHERE termék_id = ?";
21     $stmt = $conn->prepare($updateSql);
22     $stmt->bind_param("ii", $uj_keszlet, $termek_id);
23
24     if ($stmt->execute()) {
25         echo "<script>alert('Készlet sikeresen frissítve.');
```

Használati Utasítások

1. **Termék Kiválasztása:** Amennyiben a termék készletét szeretné módosítani, először ki kell választania a megfelelő terméket. Ehhez használja a "Termék ID" paramétert a URL-ben, például: `http://weboldal.hu/keszlet_modositasa.php?termek_id=[termék_id]`, ahol `[termék_id]` a kívánt termék azonosítója az adatbázisban.
2. **Új Készlet Megadása:** Töltse ki az "Új Készlet" mezőt a kívánt mennyiségű készlet megadásához.
3. **Módosítás Végrehajtása:** Kattintson a "Módosít" gombra a készlet módosításának végrehajtásához.

Figyelmeztetések

- **Adatellenőrzés:** Győződjön meg róla, hogy a megadott készlet mennyiség érvényes és pozitív egész szám.
- **Adatbiztonság:** Mindig ellenőrizze, hogy a termék azonosítója és a készlet mennyisége helyes és megfelelően módosul-e az adatbázisban.
- **Megerősítés:** Győződjön meg arról, hogy a módosítás végrehajtását megelőzően megfelelően beállította az új készletmennyiséget.

1.14 Kosár Tartalom

Cél

Az oldal célja, hogy megjelenítse a felhasználó kosarában lévő termékeket, valamint lehetőséget biztosítson a felhasználónak a fizetési folyamat elindítására.

```
18 <script>
19     var cart = [];
20
21     function displayCartContent() {
22         var cartContentElement = document.getElementById('cartContent');
23         if (cart.length === 0) {
24             cartContentElement.innerHTML = "<p>A kosár üres.</p>";
25         } else {
26             var content = "<ul>";
27             for (var i = 0; i < cart.length; i++) {
28                 content += "<li>" + cart[i].name + " - " + cart[i].quantity + " db</li>";
29             }
30             content += "</ul>";
31             cartContentElement.innerHTML = content;
32         }
33     }
34
35     function proceedToCheckout() {
36         alert("Fizetési folyamat megkezdése...");
37     }
38
39     displayCartContent();
40 </script>
```

Technikai Implementáció

- Az oldal HTML, CSS és JavaScript nyelveken készült.
- Az HTML struktúra megfelelően tagolt, és a `<meta>` címkék beállítása a megfelelő karakterkódolást és nézet méretet biztosítja.
- A CSS stílusokat a `<style>` címkékben az oldal tetején találhatók.
- A JavaScript funkciók az oldal alján találhatók, az elemek megjelenítése és a fizetési folyamat kezelése céljából.

Főbb Komponensek

1. Kosár Tartalom Megjelenítése:

- Az oldal betöltésekor azonnal megjeleníti a felhasználó kosarában lévő termékeket vagy egy üzenetet, ha a kosár üres.
- Ha a kosár nem üres, akkor minden termék nevét és mennyiségét listázza ki.

2. Fizetési Folyamat:

- A "Tovább a fizetéshez" gombra kattintva egy egyszerű üzenetablak jelenik meg, ami jelzi a fizetési folyamat megkezdését. Ebben az implementációban a fizetési folyamat nem történik meg az oldalon, csak egy üzenet jelenik meg.

Felhasznált Technológiák

- **HTML:** Az oldal struktúrájának és tartalmának leírására.
- **CSS:** Az oldal megjelenésének formázására.
- **JavaScript:** Az interaktivitás biztosítása és a felhasználói műveletek kezelése céljából.
- **Viewport Meta Tag:** A mobilbarát elrendezés és a megfelelő nézet méret beállítása.

Fejlesztői Megjegyzés

- Az oldal jelenleg csak egy egyszerű funkcióval rendelkezik, de könnyen bővíthető más funkciókkal, például a termékek törlésével vagy módosításával a kosárból.

- A fizetési folyamatot megvalósító JavaScript funkciók és a fizetési API integrációja további fejlesztési lehetőségeket kínál a jövőben.

1.15 Bejelentkező Oldal

Cél

Az oldal célja, hogy lehetővé tegye a felhasználók számára a bejelentkezést. Ellenőrzi a felhasználó által megadott adatok helyességét, és hibaüzeneteket jelenít meg, ha szükséges.

```

2 include 'db_config.php';
3 date_default_timezone_set('Europe/Budapest');
4
5 if ($_SERVER["REQUEST_METHOD"] == "POST") {
6     $email = $_POST["email"];
7     $jelszo = $_POST["jelszo"];
8
9     $sql = $conn->prepare("SELECT * FROM users WHERE email = ?");
10    $sql->bind_param("s", $email);
11    $sql->execute();
12    $result = $sql->get_result();
13
14    if ($result->num_rows > 0) {
15        $row = $result->fetch_assoc();
16        if (password_verify($jelszo, $row["jelszo"])) {
17            if (empty($row['ban_vege']) && new DateTime() < new DateTime($row['ban_vege'])) {
18                echo "<div class='error-message'>A felhasználói fiók jelenleg korlátozva van. Korlátozva: " . $row['ban_vege'] . "</div>";
19            } elseif ($row['vegleges_kiltva'] == 1) {
20                echo "<div class='error-message'>A felhasználói fiók véglegesen ki van tiltva.</div>";
21            } else {
22                session_start();
23                $_SESSION["username"] = $row["vezeteknev"] . ' ' . $row["keresztnev"];
24                header("Location: fooldal.php");
25                exit();
26            }
27        } else {
28            echo "<div class='error-message'>Hibás jelszó!</div>";
29        }
30    } else {
31        echo "<div class='error-message'>Nincs ilyen felhasználó!</div>";
32    }
33 }
34
35 $conn->close();

```

Technikai Implementáció

- Az oldal PHP és HTML nyelveken készült.
- Az adatbáziskapcsolatot a **db_config.php** fájl tartalmazza.
- Az adatok ellenőrzése és a bejelentkezési folyamat a PHP kód segítségével történik.
- Az oldal megjelenése és stílusa a CSS stíluslapban van definiálva.

Főbb Komponensek

1. Bejelentkezési Űrlap:

- Az űrlap tartalmazza az email cím és jelszó mezőket, valamint egy "Bejelentkezés" gombot.
- Ha a felhasználó hibás adatokat ad meg, hibaüzenet jelenik meg az űrlap alatt.

2. Bejelentkezési Folyamat:

- A bejelentkezési adatokat ellenőrzi az adatbázisban tárolt adatokkal.
- Ha a bejelentkezés sikeres, az oldal átirányít a főoldalra (**fooldal.php**).

3. Admin Bejelentkezés Gomb:

- Az "Admin Bejelentkezés" gomb egy másik oldalra irányít, ahol az adminisztrátorok be tudnak jelentkezni.

Fejlesztői Megjegyzés

- Az oldal egyszerű és felhasználóbarát felületet kínál a bejelentkezéshez.

- A kódot könnyen bővíthetővé lehet tenni további funkciókkal, például elfelejtett jelszó visszaállításával vagy többfaktoros azonosítással.

1.16 Felhasználó Bejelentkezés

Ez a PHP szkript az alkalmazás felhasználóinak bejelentkezését végzi:

```

2  session_start();
3  include 'db_config.php';
4
5  if ($_SERVER["REQUEST_METHOD"] == "POST") {
6      $email = $_POST["email"];
7      $jelszo = $_POST["jelszo"];
8
9      $sql = "SELECT * FROM users WHERE email='$email'";
10     $result = $conn->query($sql);
11
12     if ($result->num_rows > 0) {
13         $row = $result->fetch_assoc();
14
15         if (password_verify($jelszo, $row["jelszo"])) {
16             $_SESSION['user_id'] = $row['id'];
17             $_SESSION['user_email'] = $row['email'];
18
19             header("Location: dashboard.php");
20             exit();
21         } else {
22             echo "Hibás jelszó!";
23         }
24     } else {
25         echo "Nincs ilyen felhasználó!";
26     }
27 }
28 $conn->close();

```

Működés

1. **Munkamenet inicializálása:** A szkript először inicializálja vagy folytatja a munkamenetet, hogy tárolni tudja a bejelentkezett felhasználó adatait.
2. **Adatbázis kapcsolat:** A szkript csatlakozik az adatbázishoz a **db_config.php** konfigurációs fájl segítségével.
3. **Űrlapkezelés:** Ellenőrzi, hogy a beérkező kérés POST metódussal érkezett-e. Ha igen, beolvassa és sanitizálja az űrlapról érkező e-mail cím és jelszó adatokat.
4. **Felhasználó ellenőrzése:** A szkript lekérdezi az adatbázisból a megadott e-mail címmel rendelkező felhasználót. Ha talál eredményt, ellenőrzi a megadott jelszót a tárolt jelszóval.
5. **Bejelentkezés kezelése:** Ha a jelszó helyes, beállítja a felhasználó munkameneti változóit, majd átirányítja a felhasználót a vezérlőpulthoz (**dashboard.php**). Ha a bejelentkezés sikertelen, hibaüzenetet jelenít meg.
6. **Adatbázis kapcsolat lezárása:** A szkript lezárja az adatbáziskapcsolatot.

Fájlok

- **db_config.php:** Az adatbáziskapcsolat konfigurációs beállításait tartalmazó fájl.
- **dashboard.php:** A felhasználó vezérlőpultját kezelő fájl.

1.17 Felhasználó Regisztráció

Ez a PHP szkript a felhasználók regisztrációját végzi egy webalkalmazásban.

```
2  error_reporting(E_ALL);
3  ini_set('display_errors', 1);
4
5  include 'db_config.php';
6
7  if ($_SERVER["REQUEST_METHOD"] == "POST") {
8      $vezeteknev = mysqli_real_escape_string($conn, $_POST["vezeteknev"]);
9      $keresztnev = mysqli_real_escape_string($conn, $_POST["keresztnev"]);
10     $lakcim = mysqli_real_escape_string($conn, $_POST["lakcim"]);
11     $telefonszam = mysqli_real_escape_string($conn, $_POST["telefonszam"]);
12     $email = mysqli_real_escape_string($conn, $_POST["email"]);
13     $jelszo = password_hash(mysqli_real_escape_string($conn, $_POST["jelszo"]), PASSWORD_DEFAULT);
14
15     $sql = "INSERT INTO users (vezeteknev, keresztnev, lakcim, telefonszam, email, jelszo)
16           VALUES ('$vezeteknev', '$keresztnev', '$lakcim', '$telefonszam', '$email', '$jelszo')";
17
18     if ($conn->query($sql) === TRUE) {
19         header("Location: registration_success.php");
20         exit();
21     } else {
22         echo "Error: " . $sql . "<br>" . $conn->error;
23     }
24 }
25
26 $conn->close();
```

Működés

1. **Hibaüzenetek bekapcsolása:** Az `error_reporting` és `ini_set` függvények beállításával az összes hibaüzenet megjelenítése engedélyezett, ami segít a hibakeresésben.
2. **Adatbázis kapcsolat:** A szkript csatlakozik az adatbázishoz a `db_config.php` konfigurációs fájl segítségével.
3. **Űrlapkezelés:** Ellenőrzi, hogy a beérkező kérés POST módszerrel érkezett-e. Ha igen, beolvassa és sanitizálja az űrlapról érkező adatokat.
4. **Felhasználói adatok beszúrása:** A szkript beszúrja a felhasználó által megadott adatokat az adatbázisba. A jelszó titkosításra kerül a `password_hash` függvény segítségével.
5. **Beszúrás eredményének kezelése:** Ha a beszúrás sikeres, az űrlapról átirányítja a felhasználót a sikeres regisztrációt jelző oldalra (`registration_success.php`). Ha a beszúrás sikertelen, hibaüzenetet jelenít meg.
6. **Adatbázis kapcsolat lezárása:** A szkript lezárja az adatbáziskapcsolatot.

Fájlok

- `db_config.php`: Az adatbáziskapcsolat konfigurációs beállításait tartalmazó fájl.
- `registration_success.php`: A sikeres regisztrációt jelző oldal.

Megjegyzés

A kódban fontos megjegyezni a felhasználó által megadott adatok sanitizálását, hogy megelőzzük a SQL injection támadásokat. A jelszavak titkosítása pedig az adatbiztonság fontos eleme.

1.18 Rólunk Oldal



A Rólunk oldal a Wood Design Store webáruház része, amelynek célja bemutatni a vállalatot, annak tevékenységét és értékeit a látogatók számára.

Oldalstruktúra

Az oldal struktúrája a következő elemekből áll:

- **HTML Fájl:** A **rolunk.php** nevű HTML fájl tartalmazza az oldal kódját és szerkezetét.
- **Fő Címe:** Az oldal fő címe "Rólunk".
- **Navigációs Menü:** A navigációs menü a weboldal többi részére mutató linkeket tartalmaz.
- **Tartalomtéma:** A tartalom részletesen bemutatja a Wood Design Store vállalatot, annak céljait, termékeinek jellemzőit és az ügyfélszolgálatot.

Technikai Részletek

- **Dokumentum Típusa:** Az oldal HTML5 dokumentumtípust használ.
- **Nyelvi Támogatás:** Az oldal a magyar nyelvet használja.
- **Karakterkódolás:** A karakterkódolás UTF-8.
- **Megjelenítési Beállítások:** A viewport beállítása lehetővé teszi az oldal megfelelő megjelenítését különböző eszközökön.
- **Stíluslapok:** A megjelenésért felelős stíluslapok külső forrásból történő betöltése a <https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css> címről.
- **Stílusok:** Az oldal stílusai a `<style>` elemen belül vannak definiálva.

1.19 Termék hozzáadása űrlap PHP kódja:

```
2 include('auth.php');
3 require_once 'db_config.php';
4
5 if ($_SERVER["REQUEST_METHOD"] == "POST") {
6     $nev = $conn->real_escape_string($_POST['termek_nev']);
7     $leiras = $conn->real_escape_string($_POST['termek_leiras']);
8     $ar = $conn->real_escape_string($_POST['termek_ar']);
9     $kep = $conn->real_escape_string($_POST['termek_kep']);
10    $keszleten = $conn->real_escape_string($_POST['keszleten']);
11    $megjelenitesi_hely = $conn->real_escape_string($_POST['megjelenitesi_hely']);
12
13    $sql = "INSERT INTO termek (termek_nev, termék_leiras, termék_ar, termék_kep, készleten, megjelenitesi_hely)
14          VALUES ('$nev', '$leiras', '$ar', '$kep', '$keszleten', '$megjelenitesi_hely')";
15
16    if ($conn->query($sql) === TRUE) {
17        echo "Új termék sikeresen hozzáadva.";
18    } else {
19        echo "Hiba: " . $sql . "<br>" . $conn->error;
20    }
21
22    $conn->close();
23 }
```

Funkciók

- **Adatok Felvitele:** Az űrlap lehetővé teszi a termék nevének, leírásának, árának, képének, készletmennyiségének és megjelenítési helyének megadását.
- **Adatok Ellenőrzése:** A beérkező adatokat megfelelően ellenőrzi és validálja az űrlap, hogy biztosítsa az adatok integritását és hibamentes feldolgozását.
- **Adatok Tisztítása:** A beérkező adatokat megfelelően tisztítja és formázza a PHP kód, hogy elkerülje a biztonsági problémákat, például az SQL injection támadásokat.
- **Adatok Beszúrása:** Az űrlap elküldésekor a PHP kód beszúrja az új termék adatait az adatbázisba, biztosítva ezzel a frissített készletet és az új termék megjelenését a webáruházban.
- **Visszajelzés:** A felhasználók visszajelzést kapnak az űrlap elküldését követően arról, hogy a művelet sikeres volt-e vagy sikertelen.

Adatmezők

Az űrlap az alábbi adatmezőket tartalmazza a termék részletes adatainak megadásához:

- **Termék Név:** A termék nevének megadása.
- **Leírás:** Részletes leírás a termékről.
- **Ár:** A termék ára.
- **Kép URL:** A termék képének URL-je.
- **Készleten:** A rendelkezésre álló készlet mennyisége.
- **Megjelenítési Hely:** A termék megjelenítési helyének kiválasztása a webáruházban.

1.20 Termékek sorrendjének módosítása

Ez a PHP kód egy olyan funkciót valósít meg, amely lehetővé teszi a termékek sorrendjének módosítását az adatbázisban. A kód az alábbiak szerint működik:

```
2  include('db_config.php');
3
4  $termek_id = $_GET['termek_id'];
5
6  $sql = "SELECT sorrend FROM termekek WHERE termek_id = $termek_id";
7  $result = $conn->query($sql);
8  if ($result->num_rows > 0) {
9      $row = $result->fetch_assoc();
10     $currentOrder = $row['sorrend'];
11     $newOrder = $currentOrder - 1;
12
13     $conn->begin_transaction();
14     try {
15         $sql = "UPDATE termekek SET sorrend = $currentOrder WHERE sorrend = $newOrder";
16         $conn->query($sql);
17
18         $sql = "UPDATE termekek SET sorrend = $newOrder WHERE termek_id = $termek_id";
19         $conn->query($sql);
20
21         $conn->commit();
22     } catch (Exception $e) {
23         $conn->rollback();
24     }
25 }
26
27 header('Location: termekek_listazasa.php');
28 ?>
```

4. Adatbázis csatlakozás: Először is, a kód importálja az adatbázis konfigurációs fájlt (**db_config.php**), hogy kapcsolódjon az adatbázishoz.
5. URL paraméter beolvasása: A kód beolvassa a **termek_id** paramétert az URL-ből, amely azon termék azonosítóját tartalmazza, amelynek a sorrendjét módosítani szeretnénk.
6. Sorrend frissítése: A kód lekérdezi az adatbázisból az adott termékhez tartozó sorrendet. Ezután megpróbálja csökkenteni a termék sorrendjét egyel az előző sorrend alapján. A tranzakció biztosítja, hogy mindkét frissítés csak akkor kerüljön végrehajtásra, ha mindkettő sikeres.
7. Hiba kezelése: Ha valamilyen hiba történik a tranzakció végrehajtása közben, akkor a tranzakciót visszavonja (rollback), hogy ne legyenek inkonzisztens adatok az adatbázisban.
8. Átirányítás: Végül a kód átirányítja a felhasználót a termékek listázása oldalra (**termekek_listazasa.php**), hogy a felhasználó láthassa a frissített sorrenddel rendelkező termékeket.

Ez a funkció lehetővé teszi a termékek sorrendjének egyszerű módosítását az adminisztrátorok számára, ami segíthet a webáruházban való tartalom szervezésében és megjelenítésében.

1.21 Termékek törlése

Ez a PHP kód egy olyan funkciót valósít meg, amely lehetővé teszi egy adott termék törlését az adatbázisból. A kód a következő lépéseket hajtja végre:

```
2 require_once 'db_config.php';
3
4 if (isset($_GET['termek_id'])) {
5     $termek_id = (int)$_GET['termek_id'];
6
7     if ($termek_id > 0) {
8         $sql = "DELETE FROM termek WHERE termek_id = ?";
9         $stmt = $conn->prepare($sql);
10        $stmt->bind_param("i", $termek_id);
11
12        if ($stmt->execute()) {
13            echo "<script>alert('A termék sikeresen törölve.');
```

1. Adatbázis csatlakozás: Először is, a kód importálja az adatbázis konfigurációs fájlt (**db_config.php**), hogy kapcsolódjon az adatbázishoz.
2. Paraméterek beolvasása: A kód ellenőrzi, hogy a **termek_id** paraméter meg van-e adva az URL-ben. Ha igen, akkor az értékét átalakítja egész számmá.
3. Termék törlése: Ha a **termek_id** érvényes és nagyobb nullánál, akkor a kód előkészíti és végrehajtja a törlési műveletet az adatbázisban. Ehhez először SQL lekérdezést készít a **DELETE** utasítással, majd az értéket paraméterként köti a lekérdezéshez.
4. Visszajelzés: Ha a törlés sikeres, akkor üzenet jelenik meg, amely értesíti a felhasználót a sikerességről, és átirányítja a felhasználót a termékek listázása oldalra. Ellenkező esetben hibaüzenet jelenik meg.
5. Adatbázis kapcsolat lezárása: Végül a kód bezárja az adatbázis kapcsolatot.

Ez a funkció lehetővé teszi az adminisztrátorok számára, hogy könnyen és hatékonyan töröljék a termékeket az adatbázisból, ami segíthet a webáruház tartalmának frissítésében és karbantartásában.

1.22 Termékek listázása

Ez a PHP kód egy olyan funkciót valósít meg, amely lehetővé teszi az aktív megjelenő termékek listázását és módosítását az adminisztrátorok számára. A kód a következő lépéseket hajtja végre:

```
2 include('auth.php');
3 require_once 'db_config.php';
4
5 $sql = "SELECT * FROM termekek ORDER BY sorrend ASC";
6 $result = $conn->query($sql);
```

1. Adatbázis csatlakozás: Először is, a kód importálja az adatbázis konfigurációs fájlt (**db_config.php**), hogy kapcsolódjon az adatbázishoz.
2. Termékek lekérdezése: A kód lekéri az összes terméket az adatbázisból a **termekek** táblából, rendezve a **sorrend** mező alapján növekvő sorrendben.
3. HTML kimenet: A kód HTML kódot generál a termékek listázásához és megjelenítéséhez a weboldalon. Minden termék adatait táblázat formájában jeleníti meg, beleértve a termék nevét, leírását, árát, képét és egy sor műveletet, amelyek lehetővé teszik a termék sorrendjének módosítását, módosítását vagy törlését.
4. Műveletek hivatkozásai: A műveleteket végző gombok hivatkozásokat tartalmaznak a szükséges PHP fájlokra (**termek_sorrend_le.php**, **termek_sorrend_fel.php**, **termek_modositasa.php**, **termek_torlese.php**), amelyek felelősek a termékek sorrendjének módosításáért, módosításáért vagy törléséért. Ezen kívül a törléshez egy megerősítési párbeszédablakot is megjelenít a felhasználónak.
5. Stílusok és formázás: A kód CSS stílusokat tartalmaz a megjelenítés formázásához, például a háttérszín, a betűszín, a táblázat formázása és a gombok stílusának meghatározásához.

Ez a funkció lehetővé teszi az adminisztrátorok számára, hogy könnyen kezeljék és módosítsák a termékek listáját a webáruházban, biztosítva ezzel a felhasználóknak a friss és aktuális termékkínálatot.

2. Adatszerkezet

Bevezetés

Ez a részletes dokumentáció bemutatja a PHP nyelven írt MySQL adatbázisból történő felhasználói adatok lekérésének folyamatát. A kód a felhasználói munkamenet során tárolt email cím alapján keres egy felhasználót az adatbázisban, majd visszaadja az ehhez a felhasználóhoz tartozó adatokat.

Rendszerkövetelmények

- PHP 5.6 vagy újabb
- MySQL szerver

Tábla	Művelet	Sorok	Típus	Illesztés	Méret	Felülírás
<input type="checkbox"/> admin_users	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	2	InnoDB	utf8_hungarian_ci	16.0 KB	-
<input type="checkbox"/> kosar	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	0	InnoDB	utf8_hungarian_ci	16.0 KB	-
<input type="checkbox"/> termek	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	21	InnoDB	utf8_hungarian_ci	16.0 KB	-
<input type="checkbox"/> users	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	4	InnoDB	utf8_hungarian_ci	16.0 KB	-

Folyamat

- **Felhasználói munkamenet ellenőrzése:** Először is meg kell győződni arról, hogy a felhasználó be van-e jelentkezve, és ha igen, akkor van-e tárolt email címe a munkamenetben. Ez biztosítja, hogy csak bejelentkezett felhasználók adatait kérjük le.
- **Adatbáziskapcsolat felépítése:** A kódban először fel kell építeni egy kapcsolatot az adatbázissal. Ez általában egy előre definiált kapcsolatobjektumon (**\$conn**) keresztül történik.
- **SQL lekérdezés előkészítése:** A felhasználóhoz tartozó adatok lekéréséhez SQL lekérdezést kell készíteni. A lekérdezésben paraméterként a felhasználó email címét használjuk.
- **SQL lekérdezés végrehajtása:** A lekérdezést először előkészítjük (**prepare()** metódus), majd a paramétereket hozzárendeljük (**bind_param()** metódus). Ezután végrehajtjuk a lekérdezést (**execute()** metódus).

Az SQL-lekérdezés végrehajtása sikerült									
EXPLAIN SELECT * FROM `admin_users`;									
[Szerkesztés helyben] [Módosítás] [SQL magyarázat átugrása] [PHP-kód létrehozása]									
Extra options									
id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	admin_users	ALL	NULL	NULL	NULL	NULL	2	

- **Eredményfeldolgozás:** A lekérdezés végrehajtása után megkapjuk az eredményt. Ha van találat (a felhasználó email címe alapján), akkor ezt feldolgozzuk és eltároljuk a **\$userdata** változóban.

Változók és Objektumok

- **\$_SESSION['email']**: A felhasználó munkamenetben tárolt email címe.
- **\$conn**: Az adatbáziskapcsolatot reprezentáló objektum.
- **\$email**: A felhasználó email címe, amely alapján keresünk az adatbázisban.
- **\$sql**: Az SQL lekérdezés string formában, amely a felhasználó adatainak lekérésére szolgál.
- **\$stmt**: Az előkészített SQL állítás (statement) objektuma.
- **\$result**: Az SQL lekérdezés végrehajtása után kapott eredmény objektuma.
- **\$userdata**: A felhasználó adatait tartalmazó asszociatív tömb, ha van találat.

Hibakezelés

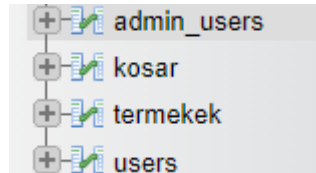
A kód jelenleg nem tartalmaz hibakezelést. Fontos lenne ellenőrizni a lekérdezés eredményét (**\$result**), hogy megbizonyosodjunk arról, hogy sikeres volt-e a lekérdezés végrehajtása. Ezenkívül érdemes lenne kezelni az esetleges adatbázis hibákat is.

Biztonsági megfontolások

A jelenlegi kód sebezhető lehet SQL befecskendezésre. Fontos, hogy mindig használjunk paraméterezett lekérdezéseket (mint itt a **bind_param()**), hogy megakadályozzuk az ilyen típusú támadásokat.






2.1 Adattáblák: Szakdoga

- admin_users
- kosar
- termek
- users



Az "Admin_users" tábla

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	admin_users	ALL	NULL	NULL	NULL	NULL	2	

			id	username	password
<input type="checkbox"/>	 Módosítás	 Másolás	 Törölés	1 admin	\$2y\$10\$apsmcLrRVDMYsOv3dvV/0eE3fCc3QqJuiLwM8Kp/Aqt...
<input type="checkbox"/>	 Módosítás	 Másolás	 Törölés	2 admin2	\$2y\$10\$qNmdyKvWYZa9ptMgBDf9x.qHAjkBz30Er1ICddz9tNj...

Tábla struktúra

Az "admin_users" tábla a következő mezőket tartalmazza:

id: Az adminisztrátor egyedi azonosítója (Primary Key). Ez egy automatikusan növekvő egész szám típusú mező.

username: Az adminisztrátor felhasználóneve. Egyedi kell hogy legyen és maximum 255 karakter hosszú string típusú mező.

password: Az adminisztrátor jelszava. Biztonsági okokból érdemes hashelve tárolni (pl. bcrypt) és maximum 255 karakter hosszú string típusú mező.

email: Az adminisztrátor e-mail címe. Egyedi kell hogy legyen és maximum 255 karakter hosszú string típusú mező.

created_at: Az adminisztrátor létrehozásának időbélyegzője. Ez egy dátum/idő típusú mező, amely automatikusan kitöltődik az adminisztrátor létrehozásakor.

Használat

Az "admin_users" tábla lehetővé teszi az adminisztrátorok hozzáadását, módosítását, törlését és lekérdezését az adminisztrációs rendszeren keresztül. Az alábbi műveleteket végezhetjük el ezzel a táblával:

- **Adminisztrátorok létrehozása:** Új adminisztrátorokat lehet létrehozni a "username", "password" és "email" mezők kitöltésével. A "created_at" mező automatikusan kitöltődik a létrehozáskor.
- **Adminisztrátorok frissítése:** Az adminisztrátorok adatait lehet frissíteni, például felhasználónév vagy jelszó módosításával.
- **Adminisztrátorok törlése:** Az adminisztrátorokat lehet törölni az adatbázisból.
- **Adminisztrátorok lekérdezése:** Lekérdezhetjük az összes adminisztrátor adatait, vagy keresést végezhetünk az adminisztrátorok között.

Példa SQL lekérdezések

Adminisztrátorok lekérdezése:

```
SELECT * FROM admin_users;
```

Új adminisztrátor hozzáadása:

“INSERT INTO admin_users (username, password, email) VALUES ('admin', 'hashed_password', 'admin@example.com');”

Adminisztrátor adatainak frissítése:

```
UPDATE admin_users SET password = 'new_hashed_password' WHERE username = 'admin';
```

Adminisztrátor törlése:

```
DELETE FROM admin_users WHERE id = 1;
```

A “termékek” tábla

Tábla struktúra

A "termékek" tábla a következő mezőket tartalmazza:

	termek_id	termek_nev	termek_leiras	termek_ar	termek_kep	készleten
--	-----------	------------	---------------	-----------	------------	-----------

id: A termék egyedi azonosítója (Primary Key). Ez egy automatikusan növekvő egész szám típusú mező.

név: A termék neve. Maximum 255 karakter hosszú string típusú mező.

leírás: A termék leírása. Szöveg típusú mező.

ár: A termék ára. Decimális típusú mező, ami a termék árát tárolja.

készleten: A termék készletmennyisége. Egész szám típusú mező, ami a termék darabszámát tárolja.

image_url: A termék képének elérési útja. Szöveg típusú mező, ami a kép elérési útvonalát tárolja.

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	termékek	ALL	NULL	NULL	NULL	NULL	21	

Használat

A "products" tábla lehetővé teszi a termékek hozzáadását, módosítását, törlését és lekérdezését a webáruházban. Az alábbi műveleteket végezhetjük el ezzel a táblával:

- **Termékek létrehozása:** Új termékeket lehet létrehozni a "név", "leírás", "ár", "készleten" és "image_url" mezők kitöltésével.

Nyissa meg a phpMyAdmin felhasználói felületét.

Válassza ki a "termékek" táblát a bal oldali navigációs panelből.

Kattintson az "Insert" fülre az adatok beillesztéséhez.

Adja meg az új termék adatait az egyes mezőkben.

Kattintson az "OK" gombra a művelet végrehajtásához.

- **Termékek frissítése:** A termékek adatait lehet frissíteni, például név, leírás, ár, készletmennyiség vagy kép módosításával.

Nyissa meg a phpMyAdmin felhasználói felületét.

Válassza ki a "termékek" táblát a bal oldali navigációs panelből.

Kattintson az "Edit" fülre az adatok szerkesztéséhez.

Módosítsa a termék adatait az egyes mezőkben.

Kattintson az "OK" gombra a művelet végrehajtásához.

- **Termékek törlése:** A termékeket lehet törölni az adatbázisból.

Nyissa meg a phpMyAdmin felhasználói felületét.
































































Válassza ki a "termékek" táblát a bal oldali navigációs panelből.

Jelölje ki azokat a termékeket, amelyeket törölni szeretne a táblából.

A táblázat alatt válassza ki a "Törölés" lehetőséget a kiválasztott termékek törléséhez.

- **Termékek lekérdezése:** Lekérdezhetjük az összes termék adatait, vagy keresést végezhetünk a termékek között.

A "termékek" tábla lehetővé teszi a termékek kezelését egy webáruházban. Ez a dokumentáció részletesen bemutatta a tábla struktúráját, használatát és biztonsági megfontolásait, valamint bemutatott példa műveleteket a phpMyAdmin segítségével a termékek kezelésére.

			termek_id	termek_nev	termek_leiras	termek_ar	termek_kep	keszleten	
<input type="checkbox"/>				10	Karosszék Comfivo	A Karosszék Comfivo 204 kényelmes ülőhelyet és mod...	154000.00	img/karosszek1.jpg	2
<input type="checkbox"/>				11	Rojaplast Reggio fenyőfából készült kerti bútor sz...	Rojaplast Reggio fenyőfából készült kerti bútor sze...	239999.00	img/kesztermek1.jpg	1
<input type="checkbox"/>				12	Fa játszóház Wickey Smart GreenHouse	A fa játszóházunk hintával a kertbe csábítja a gye...	829990.00	img/kesztermek2.jpg	1
<input type="checkbox"/>				13	Garthen kormány rusztikus dekoráció	A barna tülevélű fából készült, 80 cm átmérőjű, 8 ...	20490.00	img/disztermek1.jpg	117
<input type="checkbox"/>				14	Égetett fa virágtartó kút	Az égetett fa virágtartó kút természetes szépséget...	9990.00	img/disztermek2.jpg	46
<input type="checkbox"/>				16	Natúr fa kisasztal - Kicsi	A natúr fa kisasztal kicsi méretben praktikus és...	62990.00	img/disztermek4.jpg	21
<input type="checkbox"/>				17	Natúr fa kisasztal - Nagy	A natúr fa nagy kisasztal elegáns és praktikus meg...	71490.00	img/disztermek5.jpg	17
<input type="checkbox"/>				18	LEGO Wood gyerek tárolódoboz	A tömör tölgyfa és a legendás építőjáték designján...	50389.00	img/disztermek6.jpg	58
<input type="checkbox"/>				19	Tactic Sport Klasszikus Bordásfal Tornatermi Kivit...	A gyermekek mozgáskészségét izomzatát fejlesztik a...	33490.00	img/kesztermek3.jpg	14
<input type="checkbox"/>				20	vidaXL impregnált fenyőfa kerti hintapad	Ez a fából készült hintapad kertje vagy más kültér...	112270.00	img/kesztermek4.jpg	3
<input type="checkbox"/>				21	Fa homokozó kalózhajó Mestia	A kalózhajót utánzó, fából készült homokozóban gye...	76140.00	img/kesztermek5.jpg	2
<input type="checkbox"/>				22	Buffalo Napóleon Tölgy 8ft pool biliárd asztal	A Buffalo Napóleon Tölgy 8ft pool biliárd asztal e...	1179990.00	img/kesztermek6.jpg	1
<input type="checkbox"/>				23	Dawson modern fa dohányzóasztal	Praktikus, sokoldalú és multifunkcionális, kerek a...	45790.00	img/disztermek3.jpg	4
<input type="checkbox"/>				24	Masszív Fa Hintaló	A Masszív Fa Hintaló időtlen játék, amely a gyerek...	174990.00	img/disztermek7.jpg	2
<input type="checkbox"/>				25	Ostheimer harkályfigura fából.	Mivel mi, harkályok, szeretünk apró rovarokat és k...	6990.00	img/disztermek8.jpg	8
<input type="checkbox"/>				26	Fa kerti ház	Fa kerti ház - Élénk Kaland Fa kerti ház gyerekekn...	315580.00	img/kesztermek7.jpg	1
<input type="checkbox"/>				27	Bár asztal Comfivo	A Comfivo 259 bár asztal elegáns megjelenést kölcs...	424990.00	img/kesztermek8.jpg	5
<input type="checkbox"/>				28	Asztal Houston	A Houston ebédlőasztal modern és praktikus megoldá...	240700.00	img/kesztermek9.jpg	7
<input type="checkbox"/>				29	Szék Houston	A Houston szék kényelmes és stílusos választás a n...	59200.00	img/kesztermek10.jpg	8
<input type="checkbox"/>				30	Fa könyvtár	Ez a stabil, fa alakú polc remekül mutat majd otth...	21030.00	img/disztermek9.jpg	9
<input type="checkbox"/>				31	Rusztikus Teakfa Fali Dekoráció	Újítsd meg környezetet egyszerűen ezzel a dekorá...	12190.00	img/disztermek10.jpg	10

Biztonsági megfontolások

Fontos biztonsági megfontolások az alábbiak:

- Paraméterezett lekérdezések használata: Mindig használjunk paraméterezett lekérdezéseket az adatbázishoz való hozzáférés során a SQL injection támadások elkerülése érdekében.
- Jogosultságkezelés: Biztosítsuk, hogy csak az adminisztrátorok számára elérhető funkciókat csak az adminisztrátorok tudják elérni.

A "users" tábla

A "users" tábla használatához az alábbiak szükségesek:

- MySQL szerver
- PHP vagy más programnyelv a kapcsolat kialakításához és az adatok kezeléséhez

Használat

A "users" tábla lehetővé teszi a felhasználók hozzáadását, módosítását, törlését és lekérdezését a webalkalmazásban. Az alábbi műveleteket végezhetjük el ezzel a táblával:

	id	vezeteknev	keresztnev	lakcim	telefonszam	email	jelszo	ban_vege	vegleges_kitiltva
<input type="checkbox"/>	1	pi	to	dp	06 123456789	asd@gmail.com	\$2y\$10\$FkcTy7fMdBt9XJU5TCdKH.VOFIPxR5A5az.ooV4hZ9l...	NULL	0
<input type="checkbox"/>	2	vez	ker	dp	06 123456789	email@gmail.com	\$2y\$10\$DUKsNlUJ.FmPPs2976XW0.4R.Eap6KF2kyRsmmnNja8...	NULL	0
<input type="checkbox"/>	3	asd1	asd2	asd utca 5	123456789	asd@gmail.com	\$2y\$10\$g1FAFZpUTaITLC1WqaJg.KY8ARu2TzafWwFxi2bs...	NULL	0
<input type="checkbox"/>	4	ig	en	buda	+361234567	igen@gmail.com	\$2y\$10\$K.7QyXrNN2GXNNdVDK6NtuPehaimu1bAWvndxA2Ftu2...	NULL	0

- **Felhasználók létrehozása:** Új felhasználókat lehet létrehozni a "first_name", "last_name", "phone", "address", "email", "password" és "status" mezők kitöltésével.
- **Felhasználók frissítése:** A felhasználók adatait lehet frissíteni, például keresztnév vagy vezetéknév, telefonszám, lakcím, e-mail cím, jelszó vagy státusz módosításával.
- **Felhasználók törlése:** A felhasználókat lehet törölni az adatbázisból.
- **Felhasználók lekérdezése:** Lekérdezhetjük az összes felhasználó adatait, vagy keresést végezhetünk a felhasználók között.
- **Felhasználó státuszának változtatása:** A felhasználó státuszát lehet változtatni (pl. bannolás, korlátozás, feloldás stb.).

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	users	ALL	NULL	NULL	NULL	NULL	3	

Biztonsági megfontolások

Fontos biztonsági megfontolások az alábbiak:

- A jelszavak biztonságos tárolása: Használjunk erős hashelést a jelszavak tárolására, például bcrypt algoritmust.
- Paraméterezett lekérdezések használata: Mindig használjunk paraméterezett lekérdezéseket az adatbázishoz való hozzáférés során a SQL injection támadások elkerülése érdekében.
- Jogosultságkezelés: Az adminisztrátoroknak megfelelő jogosultsággal kell rendelkezniük a felhasználók státuszának változtatásához.

Példa phpMyAdmin műveletek

Felhasználók létrehozása:

9. Nyissa meg a phpMyAdmin felhasználói felületét.
10. Válassza ki a "users" táblát a bal oldali navigációs panelből.
11. Kattintson az "Insert" fülre az adatok beillesztéséhez.
12. Adja meg az új felhasználó adatait az egyes mezőkben.
13. Kattintson az "OK" gombra a művelet végrehajtásához.

Felhasználók frissítése:

14. Nyissa meg a phpMyAdmin felhasználói felületét.
15. Válassza ki a "users" táblát a bal oldali navigációs panelből.
16. Kattintson az "Edit" fülre az adatok szerkesztéséhez.
17. Módosítsa a felhasználó adatait az egyes mezőkben.
18. Kattintson az "OK" gombra a művelet végrehajtásához.

Felhasználók törlése:

19. Nyissa meg a phpMyAdmin felhasználói felületét.
20. Válassza ki a "users" táblát a bal oldali navigációs panelből.
21. Jelölje ki azokat a felhasználókat, akiket törölni szeretne a táblából.
22. A táblázat alatt válassza ki a "Delete" lehetőséget a kiválasztott felhasználók törléséhez.

Összegzés

A "users" tábla lehetővé teszi a felhasználók kezelését egy webalkalmazásban. Ez a dokumentáció részletesen bemutatta a tábla struktúráját, használatát és biztonsági megfontolásait, valamint bemutatott példa műveleteket a phpMyAdmin segítségével a felhasználók kezelésére.

3. Felhasználói Dokumentáció

Áttekintés

A "Webadmin Bejelentkezés" oldal lehetővé teszi a webadminisztrátorok számára a webadmin felület elérését és a bejelentkezést. Ez az oldal biztosítja a biztonságos hozzáférést a webes adminisztrációs felülethez, ahol a weboldal tartalmát és beállításait lehet módosítani.

Hogyan működik?

Bejelentkezési adatok megadása: A felhasználónak meg kell adnia a felhasználónevét és a jelszavát a megfelelő mezőkbe.

Bejelentkezés gomb megnyomása: Miután a felhasználó megadta a bejelentkezési adatokat, meg kell nyomni a "Bejelentkezés" gombot.

Hitelesítés: Az oldal ellenőrzi a megadott felhasználónevet és jelszót az adatbázisban tárolt adatokkal.

Hitelesítés eredménye: Ha a megadott adatok helyesek, a felhasználó sikeresen bejelentkezik az adminisztrációs felületre, és átirányításra kerül a "admin.php" oldalra.

Hibás bejelentkezés: Ha a megadott felhasználónév vagy jelszó helytelen, a felhasználó hibaüzenetet kap, és nem sikerül bejelentkezni.

Használati utasítások

Felhasználónév és jelszó megadása: Adja meg a felhasználónevét és a jelszavát a megfelelő mezőkbe.

Bejelentkezés: Kattintson a "Bejelentkezés" gombra a bejelentkezéshez.

Hiba esetén: Ha a bejelentkezés nem sikerül, ellenőrizze a megadott felhasználónevet és jelszót, majd próbálja meg újra.

A "fooldal.php" fájl a webshop főoldalát tartalmazza. Ez az oldal az első, amit a felhasználó lát, amikor a webshopra navigál. A főoldal tartalmazza a webshop logóját, a navigációs menüt, promóciós szekciókat, kiemelt termékeket és a lábléceket.

3.1 Főbb funkciók és tartalom

Fejléc (Header): A fejléc tartalmazza a webshop logóját, ami visszavezet a főoldalra, és az oldal tetején jelenik meg.

Navigációs Menü: A navigációs menü lehetővé teszi a felhasználók számára, hogy a különböző oldalak között navigáljanak a webshopban. A menüpontok a következők:

- Főoldal
- Kész Termékek
- Dísztermékek
- Rólunk

- Kapcsolat
- Bejelentkezés
- Regisztráció

Promóciós Szekció: A promóciós szekcióban aktuális akciókat vagy promóciókat mutat be a webshop.

Kiemelt Termékek: A főoldalon kiemelt termékek láthatóak, amelyek gyors hozzáférést biztosítanak a legnépszerűbb vagy legújabb termékekhez.

Lábléc (Footer): A lábléc az oldal alján található, tartalmazza a webshop nevét, a jogi nyilatkozatot és egyéb linkeket.

Használati utasítások

- **Navigáció:** A navigációs menü segítségével böngészhet a webshop különböző részei között.
- **Termékek megtekintése:** Kattintson a kiemelt termékek címére vagy az "Add to Cart" gombra a részletes termékinformációk megtekintéséhez.
- **Bejelentkezés/Regisztráció:** Ha már van fiókja, jelentkezzen be, vagy regisztráljon egy új fiók létrehozásához.

Fontos tudnivalók

- **Termékinformációk:** A kiemelt termékeket csak a termék nevére kattintva vagy az "Add to Cart" gombra kattintva lehet megtekinteni.
- **Fiók szükségessége:** A vásárláshoz be kell jelentkeznie vagy regisztrálnia kell egy fiókot.

Technikai részletek

- **Nyelv és Technológia:** Az oldal HTML, CSS és JavaScript nyelven van írva, és egy PHP alapú webshop backenddel kommunikál.
- **Responszív dizájn:** Az oldal reszponzív dizájnt használ, ami lehetővé teszi az optimális megjelenítést különböző eszközökön és képernyőméreteken.

Kosárhasználati útmutató

- **Kosár Tartalma:** A "Kosár Tartalma" részben láthatja a kosárba helyezett termékeket, azok árait és mennyiségét.
- **Törlés a Kosárból:** Ha szeretné eltávolítani a terméket a kosárból, kattintson a "Töröl" gombra a termék mellett.
- **Összesen:** Az összesített végösszeget láthatja az "Összesen" sorban.

Egyéb Fontos Tudnivalók

- **Kapcsolat és Támogatás:** Ha további kérdései vagy problémái vannak az oldal használata során, vagy szeretné megosztani visszajelzéseit, kérjük, lépjen kapcsolatba a webshop ügyfélszolgálatával.

- **Adatvédelem és Biztonság:** Az oldal adatvédelmi és biztonsági irányelveiről és gyakorlatáról további információkat találhat a webshop adatvédelmi tájékoztatójában.

3.2 Technikai Részletek és Frissítések

- **Frissítések és Karbantartás:** Az oldal fejlesztése és karbantartása folyamatos, hogy biztosítsa a lehető legjobb felhasználói élményt és biztonságot.
- **Hibabejelentés:** Ha technikai hibákat vagy problémákat észlel az oldal használata során, kérjük, jelezze azokat a webshop ügyfélszolgálatának.

Kosárhasználati útmutató folytatása

- **Kijelentkezés:** Ha befejezte a vásárlást és további teendői nincsenek az oldalon, kérjük, használja a "Kijelentkezés" lehetőséget a biztonságos kilépéshez az oldalról.

Egyéb Fontos Tudnivalók (folytatás)

- **Felhasználási Feltételek:** Az oldal használata során elfogadja és betartja a webshop által előírt felhasználási feltételeket és szabályokat. Kérjük, olvassa el a Felhasználási Feltételeket az oldalon található linkeken keresztül.
- **Frissítések és Hibajavítások:** Az oldal folyamatos fejlesztés alatt áll, és időről időre frissítések és hibajavítások lesznek végrehajtva a jobb felhasználói élmény és a biztonság érdekében.

Technikai Részletek

- **Adatvédelem és Biztonság:** A webshop nagy hangsúlyt fektet az adatvédelemre és a biztonságra. Az Ön adatait bizalmasan kezeljük, és csak az oldal működéséhez és a vásárlási folyamat teljesítéséhez szükséges mértékben használjuk fel.
- **Technikai Támogatás:** Ha technikai kérdései vagy problémái merülnek fel az oldal használata során, kérjük, vegye fel a kapcsolatot az oldal üzemeltetőjének technikai támogatási csapatával.

3.3 Jogi Tudnivalók és Adatvédelem

- **Cookie-k használata:** Az oldal cookie-kat használ az Ön webes élményének személyre szabása és javítása érdekében. A cookie-k információkat tárolnak az Ön böngészőjében, például az oldal beállításait vagy bejelentkezési adatait, hogy ne kelljen újra beírnia azokat az oldal újbóli meglátogatásakor.
- **Adatvédelem:** Az Ön személyes adatait bizalmasan kezeljük és nem adjuk át harmadik félnek. Az adatok védelme érdekében az oldal biztonsági intézkedéseket alkalmaz, hogy megelőzze az illetéktelen hozzáférést, a változtatást vagy a jogosulatlan adatokhoz való hozzáférést.

Kapcsolat és További Információk

- **Ügyfélszolgálat:** Ha további kérdései vagy észrevételei vannak az oldallal vagy a vásárlási folyamattal kapcsolatban, kérjük, lépjen kapcsolatba az oldal ügyfélszolgálatával. A kapcsolatfelvétel módjait megtalálhatja az "Elérhetőségek" vagy "Kapcsolat" menüpontokban.
- **Hírlevél:** Iratkozzon fel hírlevelünkre, hogy értesüljön az új termékekről, akciókról és egyéb fontos információkról. A feliratkozási lehetőséget az oldal alján található űrlapon keresztül érheti el.

Visszajelzés és Értékelés

- **Vásárlói Visszajelzések:** Nagyra értékeljük visszajelzéseit és észrevételeit az oldallal kapcsolatban. Ha elégedett vagy elégedetlen volt az élményével, kérjük, ossza meg velünk véleményét az oldal visszajelzési űrlapján keresztül.
- **Termékértékelések:** Kérem, ossza meg velünk véleményét a vásárolt termékekről. Az Ön visszajelzése segít más vásárlóknak a döntésben, és hozzájárul az oldal fejlesztéséhez és a szolgáltatásaink minőségének javításához.

4. Tesztelés

A kódban van egy bejelentkezési funkció, amely lehetővé teszi a felhasználók számára a webadmin felületre való bejelentkezést. Ez a funkció tesztelésre szorul annak érdekében, hogy megbizonyosodjunk róla, hogy helyesen működik-e. A következő lépésekkel lehetne tesztelni:

```
// Tesztelés: Ellenőrizzük a felhasználónév és jelszó adatokat
$query = $db->prepare("SELECT * FROM admin_users WHERE username = ?");
$query->bind_param('s', $username);
$query->execute();
$result = $query->get_result();

if ($result->num_rows === 1) {
    $user = $result->fetch_assoc();
    if (password_verify($password, $user['password'])) {
        // Tesztelés: Ellenőrizzük a sikeres bejelentkezést és a munkamenet
        $_SESSION['loggedin'] = true;
        $_SESSION['username'] = $username;
        header('Location: admin.php');
        exit;
    } else {
        // Hibás jelszó
        $error = "Hibás felhasználónév vagy jelszó!";
    }
} else {
    // Hibás felhasználónév
    $error = "Hibás felhasználónév vagy jelszó!";
}
```

Ezek a kiemelt részek azok, amelyek fontosak a bejelentkezési funkció helyes működésének ellenőrzéséhez és teszteléséhez. Az itt lévő tesztek segítenek biztosítani, hogy a bejelentkezési folyamat megfelelően működik, és kezeljék az esetleges hibákat vagy kivételeket a felhasználói interakció során.

4.1 Tesztelési lépések

1. **Adatbázis ellenőrzése:** Ellenőrizze, hogy az adatbázis tartalmaz-e adminisztrátor felhasználókat, és hogy a jelszavak helyesen vannak-e titkosítva.
2. **Felhasználói adatok bevitele:** Töltse ki a bejelentkezési űrlapot a meglévő adminisztrátor felhasználó adataival (felhasználónév és jelszó).
3. **Bejelentkezési funkció tesztelése:** Nyomja meg a "Bejelentkezés" gombot, hogy elküldje az adatokat a szervernek és végre legyen hajtva a bejelentkezési funkció.
4. **Helyes bejelentkezés ellenőrzése:** Ha a bejelentkezés sikeres volt, ellenőrizze, hogy a felhasználó át lett-e irányítva az adminisztrációs felületre.
5. **Hibás bejelentkezés ellenőrzése:** Ha a bejelentkezés sikertelen volt (hibás felhasználónév vagy jelszó), ellenőrizze, hogy megjelenik-e a megfelelő hibaüzenet a felhasználónak.
6. **Hibás adatok bevitele:** Tesztelje, hogy mi történik, ha hibás adatokat ad meg a bejelentkezési űrlapon, például üres mezők vagy helytelen formátumú adatok.
7. **Adatbázis lekérdezés ellenőrzése:** Győződjön meg róla, hogy a felhasználói adatok helyesen vannak-e lekérdezve az adatbázisból a megadott felhasználónév alapján.
8. **Jelszó ellenőrzése:** Ellenőrizze, hogy a megadott jelszó helyes-e a tárolt jelszóval összehasonlítva, és hogy a felhasználó bejelentkezett-e a helyes jelszóval.
9. **Felhasználói munkamenet kezelése:** Győződjön meg róla, hogy a felhasználói munkamenet megfelelően kezelésre került-e, és hogy a felhasználó belépett-e a rendszerbe, miután helyesen megadta az adatokat.

Ezek a lépések segítenek ellenőrizni és biztosítani a bejelentkezési funkció helyes működését az adminisztrációs felületen keresztül.

4.2 Tesztelési funkciók és dokumentáció

Munkamenet-kezelés ellenőrzése:

Munkamenet törlése: A `session_destroy()` függvény segítségével a munkamenetet teljesen töröljük. Ezt ellenőrizhetjük úgy, hogy megnézzük, hogy a felhasználó kilépése után a munkamenet változók üresek-e.

Cookie-k törlése: Ha a PHP beállítások engedélyezik a süтик használatát a munkamenetek azonosítására (`session.use_cookies` beállítás), akkor a `setcookie()` függvénnyel a munkamenet cookie-jait is töröljük. Ezt ellenőrizhetjük a böngésző süti-jeinek beállításaiban vagy azáltal, hogy megnézzük a HTTP kéréseket és válaszokat.

Dokumentáció:

- **Munkamenet törlése:** A munkamenetek törlése a felhasználó kijelentkezésekor fontos biztonsági intézkedés, hogy ne lehessen az előző munkamenet adataira hivatkozni a kijelentkezés után. A `session_destroy()` függvény felelős a munkamenet teljes törléséért.
- **Cookie-k törlése:** Ha a munkamenet-kezelés a PHP beállításaiban sütiket használ (`session.use_cookies`), akkor a `setcookie()` függvény segítségével töröljük ezeket a sütiket. Ez biztosítja, hogy a böngésző ne tárolja tovább a munkamenet azonosítóját.

Tesztelési eljárások:

Manuális tesztelés: Futtassuk a kódot, majd ellenőrizzük, hogy a munkamenet változók üresek-e a kilépés után, és a böngészőben nincsenek-e a munkamenethez kapcsolódó süтик.

Automatizált tesztelés: Használjunk automatizált tesztelési keretrendszert (pl. PHPUnit), hogy ellenőrizzük a kijelentkezési folyamatot, beleértve a munkamenetek törlését és a süтик törlését is.

A megadott kódban egy adatbázisból lekérdezett adatokat JSON formátumban visszaadó funkció található, amely akkor fut le, ha a `$_GET['fooldal']` változó értéke megtalálható. A kód az adatbázis kapcsolódásával kezdődik, majd lekérdezi a "termek" tábla minden rekordját, és a lekérdezés eredményét JSON formátumban visszaadja.

```
2  if (isset($_GET['fooldal'])) {  
3      $servername = "localhost";  
4      $username = "root";  
5      $password = "";  
6      $dbname = "users";  
7  
8      $conn = new mysqli($servername, $username, $password, $dbname);  
9  
10     if ($conn->connect_error) {  
11         die("Connection failed: " . $conn->connect_error);  
12     }  
13     $sql_keszleten = "SELECT * FROM termek";  
14     $result_keszleten = $conn->query($sql_keszleten);  
15     echo json_encode($result_keszleten);  
16 }
```

Adatok lekérése és JSON formátumban visszaadása:

Adatbázis kapcsolódás ellenőrzése: Ellenőrizzük, hogy sikeresen létrejött-e a kapcsolat az adatbázissal. Ha nem sikerült a kapcsolódás, akkor megfelelő hibaüzenetet kell kiírni.

Adatok lekérése: A kód lekérdezi a "termek" tábla összes rekordját.

JSON formátumban visszaadás: Az adatokat JSON formátumban visszaadjuk a kliensnek. A JSON formátum lehetővé teszi az adatok egyszerű és strukturált átvitelét.

Tesztelési eljárások:

Manuális tesztelés: Kérjük le a "fooldal" paraméterrel az oldalt, majd ellenőrizzük, hogy a kapott válasz JSON formátumban tartalmazza-e a "termek" tábla adatait.

Automatizált tesztelés: Használjunk automatizált tesztelési keretrendszert (pl. PHPUnit vagy Selenium), hogy ellenőrizzük a funkció helyes működését különböző körülmények között.

Felhasználó bejelentkezésének ellenőrzése:

Munkamenet ellenőrzése: Ellenőrizzük, hogy van-e érvényes munkamenet (`$_SESSION['loggedin']`) a felhasználó számára. Ha nincs, az azt jelenti, hogy a felhasználó nincs bejelentkezve.

Átirányítás: Ha a felhasználó nincs bejelentkezve, akkor átirányítjuk őt a bejelentkező oldalra (`admin_bejelentkezés.php`).

Kilépés a szkriptből: Az `exit` utasítás megszakítja a kód további végrehajtását, így csak a bejelentkezett felhasználók férhetnek hozzá az adminisztrációs felülethez.

```
2 session_start();
3 if (!isset($_SESSION['loggedin'])) {
4     header('Location: admin_bejelentkezés.php');
5     exit;
6 }
7 ?>
```

Felhasználó azonosítása és ellenőrzése: Először az oldal ellenőrzi, hogy a **GET** kérésben szereplő **id** paraméter megfelelő-e, és szám-e. Ezután lekéri az adott felhasználóhoz tartozó **vegleges_kitiltva** mező értékét az adatbázisból. Ha a felhasználó már véglegesen ki van tiltva (**vegleges_kitiltva == 1**), akkor a végrehajtás leáll, és hibaüzenetet ad ki. Ha nem található a felhasználó az adatbázisban, szintén hibaüzenetet ad ki.

Felhasználó végleges kitiltása: Ha minden ellenőrzés sikeres volt, akkor végrehajtódik a felhasználó végleges kitiltása az adatbázisban. Az **UPDATE SQL** utasítással beállítja a **ban_vege** értékét **NULL**-ra és a **vegleges_kitiltva** értékét **1**-re a megadott felhasználóhoz tartozó rekordon. Ezután kiírja, hogy a művelet sikeres volt.

Átirányítás: Végül az oldal átirányítja a felhasználót a **felhasznalok_listazasa.php** oldalra, ahol a felhasználók listája található.

```

2 include('auth.php');
3 require_once 'db_config.php';
4
5 if(isset($_GET['id']) && is_numeric($_GET['id'])) {
6     $id = $_GET['id'];
7
8     $sql_check = "SELECT vegleges_kitiltva FROM users WHERE id = ?";
9     if($stmt_check = $conn->prepare($sql_check)) {
10         $stmt_check->bind_param("i", $id);
11         if($stmt_check->execute()) {
12             $stmt_check->store_result();
13             if($stmt_check->num_rows > 0) {
14                 $stmt_check->bind_result($vegleges_kitiltva);
15                 $stmt_check->fetch();
16                 if($vegleges_kitiltva == 1) {
17                     die("Ez a felhasználó már véglegesen ki van tiltva.");
18                 }
19             } else {
20                 die("Nem található felhasználó ezzel az azonosítóval.");
21             }
22         } else {
23             die("Hiba történt a lekérdezés végrehajtása során: " . $stmt_check->error);
24         }
25         $stmt_check->close();
26     } else {
27         die("Nem sikerült előkészíteni az SQL lekérdezést: " . $conn->error);
28     }
29
30     $sql_update = "UPDATE users SET ban_vege = NULL, vegleges_kitiltva = 1 WHERE id = ?";
31     if($stmt_update = $conn->prepare($sql_update)) {
32         $stmt_update->bind_param("i", $id);
33         if($stmt_update->execute()) {
34             echo "A felhasználó véglegesen ki lett tiltva.";
35         } else {
36             die("Hiba történt a frissítés során: " . $stmt_update->error);
37         }
38         $stmt_update->close();
39     } else {
40         die("Nem sikerült előkészíteni az SQL frissítést: " . $conn->error);
41     }
42
43     header('location: felhasznalok_listazasa.php');
44     exit();
45 } else {
46     die("Hibás kérés.");
47 }

```

Az első rész az **ini_set** és **error_reporting** függvények hívása a hibák kijelzésének beállítására és az összes hiba megjelenítésére szolgál. Ez segíthet a hibakeresésben fejlesztés közben, de éles környezetben érdemes lenne kikapcsolni.

A következő rész az **if (\$_SERVER["REQUEST_METHOD"] == "GET" && isset(\$_GET['id']) && is_numeric(\$_GET['id']))** feltétel ellenőrzi, hogy a kérés metódusa GET-e, van-e 'id' nevű paramétere a kérésnek és ez az 'id' numerikus-e. Ha mindezek igazak, akkor a kód folytatódik, egyébként hibát dob.

Ezután a kód végrehajtja a feloldást a felhasználó végleges kitiltásának (amennyiben az 'id' érvényes és numerikus). Ha a kitiltás sikeresen feloldódik, akkor a kód kiírja a sikerüzenetet, egyébként a hibát.

```

2  ini_set('display_errors', 1);
3  error_reporting(E_ALL);
4  include('auth.php');
5  require_once 'db_config.php';
6
7  if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET['id']) && is_numeric($_GET['id'])) {
8      $id = $_GET['id'];
9
10     $sql_update = "UPDATE users SET vegleges_kitiltva = 0 WHERE id = ?";
11     if($stmt_update = $conn->prepare($sql_update)) {
12         $stmt_update->bind_param("i", $id);
13         if($stmt_update->execute()) {
14             echo "A felhasználó végleges kitiltása sikeresen feloldva.";
15         } else {
16             echo "Hiba történt a végleges kitiltás feloldása során: " . $stmt_update->error;
17         }
18         $stmt_update->close();
19     } else {
20         echo "Nem sikerült előkészíteni az SQL frissítést: " . $conn->error;
21     }
22 } else {
23     echo "Hibás kérés. A kitiltás feloldásához szükséges egy érvényes felhasználói ID.";
24 }
25 $conn->close();
26 ?>

```

if (\$_SERVER["REQUEST_METHOD"] == "GET" && isset(\$_GET['id']) && is_numeric(\$_GET['id'])) { \$id = \$_GET['id'];

Az első rész az **if (\$_SERVER["REQUEST_METHOD"] == "GET" && isset(\$_GET['id']) && is_numeric(\$_GET['id']))** feltétel ellenőrzi, hogy a kérés metódusa GET-e, van-e 'id' nevű paramétere a kérésnek, és ez az 'id' numerikus-e. Ha mindezek igazak, akkor a kód folytatódik, egyébként hibát dob.

Ezután a kód végrehajtja a korlátozás feloldását a felhasználókban (amennyiben az 'id' érvényes és numerikus). Ha a feloldás sikeresen megtörténik, akkor a kód kiírja a sikerüzenetet, egyébként a hibát.

Végül a kód a felhasználók listájának oldalára irányítja a felhasználót, majd kilép. Ez azt jelenti, hogy ha a kód eléri ezt a pontot, akkor már nincs szükség további kódvégrehajtásra, és azonnal továbbblendül a felhasználó a listázási oldalra. Ez az utolsó

sor tesztelhető funkciót nyújt, mivel irányítja a felhasználót egy másik oldalra, ami azt jelenti, hogy a kód működése kipróbálható.

```
5  if ($_SERVER["REQUEST_METHOD"] == "GET" && isset($_GET['id']) && is_numeric($_GET['id'])) {
6      $id = $_GET['id'];
7
8      $sql = "UPDATE users SET ban_vege = NULL WHERE id = ?";
9
10     $stmt = $conn->prepare($sql);
11
12     if ($stmt) {
13         $stmt->bind_param("i", $id);
14
15         if ($stmt->execute()) {
16             echo "A korlátozás sikeresen feloldva.";
17         } else {
18             echo "Hiba történt a korlátozás feloldása során: " . $stmt->error;
19         }
20
21         $stmt->close();
22     } else {
23         echo "Nem sikerült előkészíteni az SQL utasítást.";
24     }
25
26     $conn->close();
27 } else {
28     echo "Hibás kérés.";
29 }
30
31 header('Location: felhasznalok_listazasa.php');
32 exit();
```

Az első rész az `if ($_SERVER["REQUEST_METHOD"] == "POST")` feltétel ellenőrzi, hogy a kérés módszere POST-e. Ha igen, akkor a kód folytatódik, különben nem történik semmi.

Ezt követően az adatokat ellenőrzik és előkészítik a frissítéshez. Ha az adatok érvényesek és a frissítés sikeres, akkor kiírják a sikerüzenetet, egyébként hibát jelentenek.

Az oldal alján egy űrlap található, amely lehetővé teszi a felhasználó korlátozását. A form elküldésekor az űrlap azonosítója és a korlátozás vége POST kérést küld a saját oldalra.

Ez tesztelhető működést biztosít a felhasználó korlátozásához, valamint az űrlap adatok POST kéréssel történő továbbításához.

```
6  if ($_SERVER["REQUEST_METHOD"] == "POST") {
7      $id = $_POST['id'] ?? null;
8      $ban_vege = $_POST['ban_vege'] ?? null;
9
10     if(isset($id) && is_numeric($id) && isset($ban_vege)) {
11         $sql_update = "UPDATE users SET ban_vege = ? WHERE id = ?";
12         if($stmt_update = $conn->prepare($sql_update)) {
13             $stmt_update->bind_param("si", $ban_vege, $id);
14             if($stmt_update->execute()) {
15                 echo "A felhasználó sikeresen korlátozva " . $ban_vege . "-ig.";
16             } else {
17                 echo "Hiba történt a korlátozás során: " . $stmt_update->error;
18             }
19             $stmt_update->close();
20         } else {
21             echo "Nem sikerült előkészíteni az SQL frissítést: " . $conn->error;
22         }
23         $conn->close();
24     } else {
25         echo "Hibás kérés.";
26     }
27 }
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") { $id = $_POST['id'] ?? null;
$ban_vege = $_POST['ban_vege'] ?? null;
```

Ez a részlet biztosítja, hogy a szükséges adatok rendelkezésre állnak, mielőtt megjelenítenék vagy módosítanák a felhasználó adatait.

```

$id = isset($_GET['id']) ? (int) $_GET['id'] : null;

if (!$id) {
    echo "Nincs megadva id.";
    exit;
}

$stmt = $conn->prepare("SELECT * FROM users WHERE id = ?");
$stmt->bind_param("i", $id);
$stmt->execute();
$result = $stmt->get_result();

if (!$user = $result->fetch_assoc()) {
    echo "Nincs ilyen id-jű felhasználó.";
    exit;
}

```

Az első részben ellenőrzik a kapcsolódást az adatbázishoz. Ha nem sikerül a kapcsolódás, akkor hibát dob és kilép a program.

A következő részben ellenőrzik, hogy van-e megadva **id**. Ha nem, akkor hibát dob és kilép.

Ezután lekérdezik a felhasználót az adatbázisból az adott **id** alapján. Ha nincs ilyen felhasználó az adatbázisban, akkor hibát dob és kilép.

A HTML rész a szerkesztőfelületet tartalmazza, ahol megjelenítik a felhasználó adatait egy űrlapon, amelyet a felhasználó módosíthat. Ez tesztelési célokat szolgál, mivel lehetővé teszi a felhasználó adatainak módosítását, majd az adatok elküldésével a **felhasznalo_frissitese.php** oldalnak lehetőséget ad az adatok frissítésére az adatbázisban.

Ez a részlet a következőt teszi:

Ellenőrzi, hogy van-e **id** paraméter a GET kérésben.

Ha van, akkor megpróbálja az **id** értékét integerré konvertálni.

Ellenőrzi, hogy az **id** értéke nem üres és nem nulla.

Lekéri a felhasználó adatait az adatbázisból az **id** alapján.

Ellenőrzi, hogy a lekérdezés sikeres volt-e és hogy van-e találat. Ha nincs, akkor hibaüzenetet dob.

```

5  if(isset($_GET['id']) && is_numeric($_GET['id'])) {
6      $id = $_GET['id'];
7
8      $sql = "DELETE FROM users WHERE id = $id";
9
10     if ($conn->query($sql) === TRUE) {
11         echo "A felhasználó sikeresen törölve.";
12     } else {
13         echo "Hiba történt a törlés során: " . $conn->error;
14     }
15
16     $conn->close();
17 } else {
18     echo "Hibás kérés.";
19 }
20 header('Location: felhasznalok_listazasa.php');
21 exit();

```

Az első részben ellenőrzik, hogy van-e **id** paraméter a GET kérésben és hogy az **id** numerikus-e. Ha nincs vagy nem numerikus, akkor hibát jelez és kilép a program.

Ezután előkészítik a törlési műveletet az adatbázisban az adott **id** alapján.

A kód végrehajtja a törlési műveletet az adatbázisban, és ellenőrzi, hogy a művelet sikeres volt-e. Ha igen, akkor kiírja a sikerüzenetet, egyébként hibát jelez.

Végül a kód átirányítja a felhasználót a **felhasznalok_listazasa.php** oldalra, ami azt jelenti, hogy ha a kód ezt a pontot eléri, akkor már nincs szükség további kódvégrehajtásra, és azonnal továbbblendül a felhasználó a listázási oldalra. Ez tesztelhető működést nyújt, mivel irányítja a felhasználót egy másik oldalra.

```

4  if (isset($_SESSION['userid'])) {
5      $userid = $_SESSION['userid'];
6      $query = $db->prepare("SELECT * FROM users WHERE id = ?");
7      $query->bind_param('i', $userid);
8      $query->execute();
9      $result = $query->get_result();
10     $user = $result->fetch_assoc();
11
12     echo '
13 <input type="text" id="vezeteknev" name="vezeteknev" value="" . $user['vezeteknev'] . "" required>
14 <input type="text" id="keresztnev" name="keresztnev" value="" . $user['keresztnev'] . "" required>
15 <input type="text" id="lakcim" name="lakcim" value="" . $user['lakcim'] . "" required>
16 <input type="tel" id="telefonszam" name="telefonszam" value="" . $user['telefonszam'] . "" required>
17 <input type="email" id="email" name="email" value="" . $user['email'] . "" required>
18 ';
19 }

```

1. Először kapcsolódik az adatbázishoz.
2. Ellenőrzi, hogy van-e **\$_SESSION['userid']** érték, ami azt jelzi, hogy a felhasználó be van jelentkezve.
3. Ha a felhasználó be van jelentkezve, akkor lekérdezi a felhasználó adatait az adatbázisból az **id** alapján.
4. Az adatokat megjeleníti az űrlapon előre kitöltött mezőkben.

```

10 if ($conn->connect_error) {
11     die("Connection failed: " . $conn->connect_error);
12 }
13
14 $sql = "SELECT * FROM users WHERE email = ?";
15 $stmt = $conn->prepare($sql);
16 $stmt->bind_param("s", $email);
17 $stmt->execute();
18 $result = $stmt->get_result();
19 if ($user = $result->fetch_assoc()) {
20     if (password_verify($password, $user['jelszo'])) {
21         $_SESSION['username'] = $user['vezeteknev'] . ' ' . $user['keresztnev'];
22     } else {
23         $error = "Hibás jelszó.";
24     }
25 }
26
27 if (!empty($error)) {
28     echo "<p>$error</p>";
29 }
30 $userdata = null;
31 if (isset($_SESSION['email'])) {
32     $email = $_SESSION['email'];
33     $sql = "SELECT vezeteknev, keresztnev, email, utolso_beletes, regisztracio_datuma FROM users WHERE email = ?";
34     $stmt = $conn->prepare($sql);
35     $stmt->bind_param("s", $email);
36     $stmt->execute();
37     $result = $stmt->get_result();
38     if ($user = $result->fetch_assoc()) {
39         $userdata = $user;
40     }
41 }
42 $sql_keszleten = "SELECT * FROM termek";
43 $result_keszleten = $conn->query($sql_keszleten);

```

Ebben a kódrészletben többféle tesztelési funkcionalitás is található:

Felhasználó bejelentkezés ellenőrzése: A kód megvizsgálja, hogy a felhasználó be van-e jelentkezve azáltal, hogy ellenőrzi, hogy létezik-e `$_SESSION['username']`. Ha igen, akkor megjeleníti a felhasználó nevét a fejlécben, és lehetővé teszi a kijelentkezést. Ellenkező esetben megjeleníti a bejelentkezési és regisztrációs linkeket.

Kosárba helyezés funkció: A kód lehetővé teszi a termékek kosárba helyezését azáltal, hogy hozzáad egy eseményfigyelőt minden "Kosárba" gombhoz, amely a termékhez tartozik. Amikor a felhasználó rákattint a gombra, a termék adatait hozzáadja a kosárhoz, amelyet a helyi tárolásban tárol, és frissíti a kosár tartalmát az oldalon.

Kosár megjelenítése: A kód megjeleníti a kosár tartalmát egy modális ablakban, amelyet a felhasználó fel tud hívni az oldalon található "Kosaram" gombbal. A kosár tartalma dinamikusan frissül a felhasználó által hozzáadott és eltávolított tételek alapján.

Ezek a tesztelési funkciók segítenek abban, hogy a weboldal működése és felhasználói élménye megfelelő legyen. Az ellenőrzések és funkciók segítenek a hibák felderítésében és azok kezelésében, valamint a weboldal funkcionalitásának tesztelésében és javításában.

Egységtesztek: Ezek az apró tesztek ellenőrzik az egyes függvények működését. Például, ha lenne egy függvény a készletmódosítás végrehajtására, akkor annak tesztje ellenőrizné, hogy a megfelelő adatokat módosítja-e az adatbázisban.

Integrációs tesztek: Ezek a tesztek a rendszer részeinek együttes működését ellenőrzik. Például, egy integrációs teszt ellenőrizhetné, hogy az űrlap megfelelően kommunikál-e az adatbázissal, és helyesen frissíti-e a készletet.

Elfogadási tesztek: Ezek a tesztek az alkalmazás teljes funkcionalitását ellenőrzik a felhasználói szempontból. Például, az egyik ilyen teszt ellenőrizhetné, hogy a felhasználó

az űrlapon keresztül meg tudja-e változtatni a termék készletét, és megkapja-e a megfelelő visszajelzést a módosításról.

5. Irodalomjegyzék

1. Frontend technológiák:

- Flavio Copes: "Front-End Developer Handbook 2019"
- Jon Duckett: "HTML and CSS: Design and Build Websites"
- Kyle Simpson: "You Don't Know JS" (könyvsorozat)
- Eloquent JavaScript: <https://eloquentjavascript.net/>

2. Backend technológiák:

- Robin Nixon: "Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5"
- Node.js Documentation: <https://nodejs.org/en/docs/>
- Express.js Documentation: <https://expressjs.com/>

3. JavaScript és jQuery:

- David Flanagan: "JavaScript: The Definitive Guide"
- Douglas Crockford: "JavaScript: The Good Parts"
- jQuery Documentation: <https://api.jquery.com/>
- MDN Web Docs: JavaScript Guide: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>

4. PHP és MySQL:

- Matt Zandstra: "PHP Objects, Patterns, and Practice"
- David Sklar: "Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5"
- PHP Documentation: <https://www.php.net/docs.php>
- MySQL Documentation: <https://dev.mysql.com/doc/>

5. CSS és stílusrendszerek:

- Eric A. Meyer, Estelle Weyl: "CSS: The Definitive Guide"
- Jonathan Snook, Steve Smith, Cameron Adams, et al.: "The Art and Science of CSS"
- Sass Documentation: <https://sass-lang.com/documentation>
- Bootstrap Documentation: <https://getbootstrap.com/docs/>

6. Adatbázisok és adatmodellálás:

- Abraham Silberschatz, Henry F. Korth, S. Sudarshan: "Database System Concepts"
- Rick F. van der Lans: "Introduction to SQL"
- MySQL Documentation: <https://dev.mysql.com/doc/>
- PHPMyAdmin Documentation: <https://docs.phpmyadmin.net/>

Ezek a források segítettek a frontend, backend, JavaScript, PHP, CSS, PHPMyAdmin és MySQL témájú ismereteink elmélyítésében és a Wood Design Store webshop elkészítésében.

6. Mellékletek

<https://www.w3schools.com/sql/>

https://www.w3schools.com/sql/sql_intro.asp

https://www.w3schools.com/sql/sql_create_db.asp

<https://www.reddit.com>

[https://codefinity.com/get-](https://codefinity.com/get-started/spa/v16/spv16/quarter?utm_source=google&utm_medium=cpc&utm_campaign=20955067105&utm_content=161127022780&utm_term=coding&gad_source=1&gclid=CjwKCAjwoa2xBhACEiwA1sb1BMhk3dyIIBPpw52gJNpoUQdSowHt_zfmYKUByVUPR-P9lw5FBeDsRxOCJrkQAvD_BwE)

[started/spa/v16/spv16/quarter?utm_source=google&utm_medium=cpc&utm_campaign=20955067105&utm_content=161127022780&utm_term=coding&gad_source=1&gclid=CjwKCAjwoa2xBhACEiwA1sb1BMhk3dyIIBPpw52gJNpoUQdSowHt_zfmYKUByVUPR-P9lw5FBeDsRxOCJrkQAvD_BwE](https://codefinity.com/get-started/spa/v16/spv16/quarter?utm_source=google&utm_medium=cpc&utm_campaign=20955067105&utm_content=161127022780&utm_term=coding&gad_source=1&gclid=CjwKCAjwoa2xBhACEiwA1sb1BMhk3dyIIBPpw52gJNpoUQdSowHt_zfmYKUByVUPR-P9lw5FBeDsRxOCJrkQAvD_BwE)

<https://www.phptutorial.net>

<https://discord.com>

<https://hu.wikipedia.org/>

<https://github.com/bbLDxd/Vizsgaremek.git>

git@github.com:bbLDxd/Vizsgaremek.git

