**Baranya Vármegyei Szakképzési Centrum**

**Simonyi Károly Technikum és Szakképző Iskola**

**Vizsgaremek**

**Pécs**

**2025**

Készítették: Pál Rajmund, Futó Csaba

**Baranya Vármegyei Szakképzési Centrum**

**Simonyi Károly Technikum és Szakképző Iskola**

**Pécs**

**2025**

Szakma megnevezése: Szoftverfejlesztő és –tesztelő

A szakma azonosító száma: 5 0613 12 03

Készítették: Pál Rajmund, Futó Csaba

**Vizsgaremek**

Wimu Webshop

**Tartalomjegyzék**

[**1. Bevezető** 7](#_Toc195128886)

[**1.1 Projekt célja és motiváció** 7](#_Toc195128887)

[**1.2 Tanultak és új ismeretek** 7](#_Toc195128888)

[**1.3 Későbbi tervek** 7](#_Toc195128889)

[**1.4 Csapatmunka és szerepkörök** 8](#_Toc195128890)

[**1.5** **Összegzés** 8](#_Toc195128891)

[**2. Témaválasztás:** 8](#_Toc195128892)

[**2.1 Közös cél: egy életképes prototípus létrehozása** 8](#_Toc195128893)

[**2.2 Mit hoz a jővő?** 9](#_Toc195128894)

[**3. Az alkalmazott fejlesztői eszközök:** 9](#_Toc195128895)

[**3.1. Programozási és leíró nyelvek & keretrendszerek** 9](#_Toc195128896)

[**3.2. Fejlesztői környezetek & eszközök** 9](#_Toc195128897)

[**3.3. Design & Médiaeszközök** 10](#_Toc195128898)

[**3.4. Könyvtárak & Függőségek** 10](#_Toc195128899)

[**3.5. Egyéb eszközök** 10](#_Toc195128900)

[**3.6. Biztonsági eszközök** 10](#_Toc195128901)

[**3.7. Kompatibilitás & tesztelés** 10](#_Toc195128902)

[**4. Tervezési módszer** 10](#_Toc195128903)

[**4.1 Alkalmazás tervezése:** 10](#_Toc195128904)

[**4.2 Tervezési módszertan:** 11](#_Toc195128905)

[**4.3 Vizuális modellezés:** 11](#_Toc195128906)

[**4.4 OOP megvalósulása:** 11](#_Toc195128907)

[**5. Adatmodell leírása:** 12](#_Toc195128908)

[13](#_Toc195128909)

[**5.1 Főbb táblák:** 13](#_Toc195128910)

[14](#_Toc195128911)

[**5.2 Fő funkciók:** 14](#_Toc195128912)

[**5.3 Táblák kapcsolatai** 14](#_Toc195128913)

[**5.4 Indexek és optimalizálás** 15](#_Toc195128914)

[**5.5 Biztonsági intézkedések** 15](#_Toc195128915)

[**5.6 Jelentős funkciók és adatbázis integráció** 15](#_Toc195128916)

[**5.7 Későbbi fejlesztések** 16](#_Toc195128917)

[**6. Részletes feladatspecifikáció, algoritmusok** 17](#_Toc195128918)

[**6.1. belepes.php – Belépési folyamat** 17](#_Toc195128919)

[**6.2 Főbb függvények és metódusok specifikációja** 18](#_Toc195128920)

[**6.3 Vásárlói adatok validálása (2\_vasarloi\_adatok.php)** 19](#_Toc195128921)

[**6.4. Fizetési mód kiválasztása (3\_fizetesi\_modok.php)** 19](#_Toc195128922)

[**6.5. Rendelés összegzése (4\_rendeles\_osszegzes.php)** 20](#_Toc195128923)

[**6.6 Rendelés feldolgozása (5\_rendeles\_elkuld.php)** 21](#_Toc195128924)

[**6.7. Bejelentkezési rendszer (belepes.php)** 22](#_Toc195128925)

[**6.8 Algoritmusok leírása** 22](#_Toc195128926)

[**6.9 Rendelésfeldolgozás folyamata** 23](#_Toc195128927)

[**6.10. Termékkereső algoritmus (index.php)** 23](#_Toc195128928)

[**6.11. Kosár megtekintése (kosar\_megtekintes.php)** 24](#_Toc195128929)

[**6.12. Termék kosárba tétele (kosarba\_tesz.php)** 25](#_Toc195128930)

[**6.13. Profil módosítás (profil\_modosit.php és** 26](#_Toc195128931)

[**6.14. Regisztráció (reg.php és reg\_ellenoriz.php)** 27](#_Toc195128932)

[**6.15. Termék listázás és keresés (index.php)** 28](#_Toc195128933)

[**6.16. Algoritmusok struktogramokkal** 29](#_Toc195128934)

[**7. Unit teszt** 31](#_Toc195128935)

[**7.1. Bevezetés** 31](#_Toc195128936)

[**7.2. Tesztelési módszerek** 34](#_Toc195128937)

[**7.3 Fehér doboz tesztelés** 34](#_Toc195128938)

[**7.4. Tesztelési szintek** 35](#_Toc195128939)

[**7.5 Integrációs tesztelés** 35](#_Toc195128940)

[**7.6 Rendszertesztelés** 35](#_Toc195128941)

[**7.7. Tesztelési technikák** 36](#_Toc195128942)

[**7.8 Ekvivalencia particionálás** 36](#_Toc195128943)

[**7.9 Állapotgép-alapú tesztelés** 36](#_Toc195128944)

[**7.10. Tesztkörnyezet és eredmények** 36](#_Toc195128945)

[**7.11. Összegzés és javaslatok** 37](#_Toc195128946)

[**8. Fejlesztés** 37](#_Toc195128947)

[**8.1 Tényleges rendelés** 37](#_Toc195128948)

[**8.2 Számla kiállítás** 37](#_Toc195128949)

[**8.3 Email cím megerősítés** 37](#_Toc195128950)

[**8.4 Több termék feltöltés a webshopba** 37](#_Toc195128951)

[**8.5 Szállítási cím változtatás** 38](#_Toc195128952)

[**8.6 Admin felület fejlesztése** 38](#_Toc195128953)

[**9. Összegzés** 38](#_Toc195128954)

[**9.1 Szakmai fejlődés** 38](#_Toc195128955)

[**9.2 Jövőbeli célok** 39](#_Toc195128956)

# **1. Bevezető**

## **1.1 Projekt célja és motiváció**

A Wimu Webshop létrehozásával célunk egy egyszerű, ugyanakkor hatékony online vásárlási platform fejlesztése volt, amelyen keresztül elsajátíthattuk a webalkalmazások teljes életciklusát. A projektet azért választottuk, mert lehetőséget biztosított számunkra a gyakorlati ismeretek mélyítésére, az adatbázis-tervezéstől kezdve a felhasználói felület optimalizálásán át egészen a modern webfejlesztési technológiák alkalmazásáig. A kezdeti technikai kihívások – mint például a kosárkezelés és a session-kezelés – leküzdése után sikeresen implementáltunk összetettebb funkciókat is, többek között különböző fizetési módok integrálását, valamint a raktárkészlet pontos nyilvántartását. A fejlesztési folyamat során kiemelt figyelmet fordítottunk a skálázhatóságra, a biztonságos adatkezelésre, valamint a felhasználói élmény folyamatos javítására, hogy a végeredmény egy professzionális és megbízható webshop legyen.

## **1.2 Tanultak és új ismeretek**

*A fejlesztés során elsajátítottuk:*

**Adatbázis-kezelés**: MySQL táblák tervezése (pl. termékek, felhasználók, rendelések).

**Biztonsági technikák**: Jelszavak titkosítása (Hash), SQL injection elleni védelem.

**Folyamatkezelés**: Felhasználói regisztráció, kosár tartalmának mentése, rendelés véglegesítése.

**Fizetési integráció**: PayPal SDK használata bankkártyás tranzakciókhoz.

**Reszponzív design**: Bootstrap segítségével mobilbarát felületek kialakítása.

## **1.3 Későbbi tervek**

**Jövőbeli fejlesztési céljaink**:

**Email-es regisztráció**: Jelszó-visszaállítás és regisztráció megerősítése.

**Rendeléskövetés**: Felhasználók számára áttekinthető szállítási állapot.

**Admin felület**: Termék- és rendeléskezelés intuitív vezérlőpultja.

**Értékelési rendszer**: Vásárlói visszajelzések megjelenítése.

## **1.4 Csapatmunka és szerepkörök**

*A csapatban a feladatok így oszlottak meg:*

**Csaba:** Backend fejlesztés (PHP, adatbázis logika, rendelésfeldolgozás).

**Rajmund:** Frontend tervezés (HTML/CSS, Bootstrap).

A kommunikációhoz Discordot használtunk. A kódot GitHubon tároltuk.

*Projektszervezési Eszközök*

**Discord (Kommunikáció):** Feladatok szétosztása és állapotuk nyomon követése (pl. "Kosár implementálása", "Fizetési módok hozzáadása").

**GitHub:** Kódmegosztás és együttműködés (branch-ek használata).

**Messenger:** Megbeszéltük, hogymikor csináljuk a vizsgaremeket.

## **Összegzés**

A Wimu Webshop projekt lehetővé tette, hogy egy valós alkalmazás fejlesztésének minden szakaszában részt vegyünk. A nehézségek (pl. adatbázis-frissítések szinkronizálása) megoldása közben mélyebb betekintést kaptunk a webes rendszerek működésébe. A projekt továbbfejlesztésével célunk, hogy egy teljes körű, felhasználóbarát webshoppá nőjön ki.

# **2. Témaválasztás:**

*Miért ezt választottuk?*

Én Csaba azért ezt szerettem volna választani, mert nagyon sok webshop van a világban, de én is meg akartam nézni a folyamatát, hogy hogyan épül fel egy webshop.

Én Rajmund azért ezt választottam, mert mindig is érdekelt az üzleti oldala a dolgoknak, illetve mindig akartam egy saját boltot/webshopot nyitni.

## **2.1 Közös cél: egy életképes prototípus létrehozása**

Bár különböző motivációk vezettek minket, a cél egyértelmű volt: egy működő webshop prototípus, ami:

**Technikailag megalapozott**: Biztonságos, skálázható, könnyen bővíthető.

**Felhasználóbarát**: Intuitív navigáció, reszponzív design, gyors betöltés.

**Üzletileg releváns**: Valós igényeket szolgál (pl. címkezelés külföldre szállításhoz).

A projekt során megtanultuk, hogy a kommunikáció és a kompromisszumok nélkülözhetetlenek egy csapatban. Például amikor Rajmund egy összetett animációt akart a kosárhoz, de Csaba rámutatott, hogy az lelassítaná az oldalt, kompromisszumként egyszerűbb, de hatékony megoldást választottunk.

## **2.2 Mit hoz a jővő?**

A webshop jelenleg egy **MVP** (Minimum Viable Product), de alapozóként szolgál további fejlesztésekhez:

**Email-es regisztráció**: Jelszó-visszaállítás és automatikus értesítések implementálása.

**Analitika**: Google Analytics integráció a vásárlói viselkedés nyomon követésére.

**SEO-optimalizálás**: A termékoldalak tartalmának finomhangolása keresőmotorok számára.

Csaba számára ez a projekt egy technikai referenciapont, Rajmundnak pedig egy üzleti alapkő. Mindketten abban reménykedünk, hogy a jövőben ezt a prototípust egy teljes értékű online vállalkozássá lehet fejleszteni.

# **3. Az alkalmazott fejlesztői eszközök:**

## **3.1. Programozási és leíró nyelvek & keretrendszerek**

**PHP**: A backend logika (pl. kosárkezelés, felhasználókezelés) megvalósítása.

**MySQL:** Relációs adatbázis a termékek, felhasználók és rendelések tárolására.

**HTML/CSS/JavaScript:** Frontend felület reszponzív design-nal és interaktív elemekkel.

**Bootstrap 5**: Gyors és egységes felhasználói felület kialakítása (pl. kártyák, navigációs sáv).

## **3.2. Fejlesztői környezetek & eszközök**

**Visual Studio Code**: Fő IDE kódírásra, debugolásra (PHP, HTML, CSS, JS).

**XAMPP**: Lokális szerver környezet (Apache, MySQL, PHP) teszteléshez.

**phpMyAdmin**: Adatbázisok kezelése, SQL lekérdezések futtatása.

**Git & GitHub**: Verziókövetés és csapatmunka koordinálása.

## **3.3. Design & Médiaeszközök**

**PiktoChart**: Banner szerkesztése (pl. img/banner2.png).

**Leonardo.ai, MS Copilot**: Termék képek

**Bootstrap Icons**: Ikongyűjtemény a felhasználói felülethez.

## **3.4. Könyvtárak & Függőségek**

**jQuery:** Dinamikus űrlapkezelés és AJAX kérések (pl. kosár frissítése).

**Popper.js & Bootstrap JS**: Interaktív elemek (dropdown menük, tooltip-ek).

## **3.5. Egyéb eszközök**

**Messenger/Discord**: Feladatok szétosztása és állapotkövetés.

**Google Chrome DevTools**: Frontend hibakeresés.

## **3.6. Biztonsági eszközök**

**BCrypt:** Jelszavak titkosítása (password\_hash() és password\_verify()).

**Prepared Statements**: SQL injection elleni védelem (pl. mysqli->prepare()).

**Session Hardening**: Cookie-k biztonságos beállításai (httponly, secure flag-ek).

## **3.7. Kompatibilitás & tesztelés**

**Cross-Browser Tesztelés**: Chrome, Firefox, Safari.

**Reszponzív Design Tesztelés**: Mobil-, tablet- és asztali nézet.

# **4. Tervezési módszer**

## **4.1 Alkalmazás tervezése:**

A Wimu Webshop tervezése során a felhasználói igények és üzleti célok összehangolására fókuszáltunk. A tervezés három fő szakaszra oszlott:

**Követelményelemzés**: A vásárlók és adminisztrátorok igényeinek feltárása (pl. egyszerű regisztráció, kosárkezelés, rendeléskövetés).

**Rendszerfelépítés**: A háromrétegű modell (frontend, backend, adatbázis) kialakítása.

*Funkcionális specifikációk:*

Felhasználói szerepkörök (vendég, regisztrált felhasználó, admin).

Modulok (termékkatalógus, kosár, fizetési rendszer).

A tervezés során kiemelt szerepet kapott a reszponzív design, hogy a webshop minden eszközön (mobil, tablet, asztali) optimálisan működjön.

## **4.2 Tervezési módszertan:**

A projekt agilis módszertant követett, amely lehetővé tette a folyamatos visszajelzés alapján történő iteratív fejlesztést.

**Csapatmunka**: Megbeszélés alapján összedolgoztunk, hogy mikor ér rá a másik kolléga.

**Felhasználó tesztelés**: Felhasználóként bejelentkeztünk leteszteltünk főbb funkciókat.

## **4.3 Vizuális modellezés:**

A rendszer vizuális reprezentációja kulcsfontosságú volt a tervezésben:

*Bootstrap-ek:*

Főoldal elrendezése (termékkártyák, navigációs sáv).

Kosár oldal felépítése (táblázatos nézet, gombok).

*Adatbázis séma tervezése:*

ER-diagram készült a táblák és kapcsolatok szemléltetésére (pl. arucikk ↔ kategoriak, ugyfel ↔ rendelesek).

Normalizálás az ismétlődések elkerülése végett.

*Folyamatábrák:*

Belépési és rendelési folyamatok vizualizációja.

## **4.4 OOP megvalósulása:**

Bár a projekt főként PHP-t használ, objektumorientált elemeket is építettünk bele:

**Adatbázis kezelés OOP stílusban**:

A mysqli osztály használata prepared statement-ekkel:

$parancs = $kapcsolat->prepare("SELECT \* FROM ugyfel WHERE email=?");

$parancs->bind\_param("s", $email);

$parancs->execute();

Transaction kezelés a rendeléseknél

**Osztálytervek jövőbeli bővítéshez**:

**User osztály**: Felhasználói adatok és jogosultságok kezelése.

**Cart osztály**: Kosár műveletek (hozzáadás, törlés, frissítés).

**Product osztály**: Termékek lekérdezése és szűrése.

*Öröklés és egységbezárás:*

A Payment főosztályból származó alosztályok (pl. PayPalPayment, BankTransfer) a fizetési módok kezelésére. (fejlesztés alatt)

Privát metódusok biztonságos adatkezeléshez (pl. jelszó titkosítás).

**MySql**: Adatbáziskapcsolat egyszeres példányosítása.

# **5. Adatbázis ábrája:**

## 

## **5.1 Főbb táblák:**

* **Termékek** (arucikk)
  + Adatok: név, ár, készlet, kategóriák (3 szint), leírások.
  + Példa: Laptopok, workstationok.
* **Kategóriák** (kategoriak)
  + Hierarchikus struktúra (pl. Elektronika → Laptopok → Gaming).
  + Szülő-mezőkkel (szulo1, szulo2).
* **Felhasználók** (ugyfel)
  + Regisztráció, címek, titkosított jelszó (BCrypt), admin jogok (role).
* **Kosár** (kosar)
  + Session vagy felhasználóhoz kötött, ideiglenes adatok a rendelés előtt.
* **Rendelések** (rendelesek + rendeles\_tetelek)
  + Fizetési módok, címek, egyedi kód (67e7c1d056397).
  + Tételek: termék ID, darabszám, ár.

# 

## **5.2 Fő funkciók:**

* **Többszintű kategóriák**: Szűrés és navigáció egyszerűsítése.
* **Kosárkezelés**: Session-alapú vagy felhasználói fiókhoz kötött.
* **Rendelésfeldolgozás**: Szállítási/számlázási címek, külföldi támogatás.
* **Biztonság:** Titkosított jelszavak, bejelentkezési napló (naplo).

## **5.3 Táblák kapcsolatai**

Az adatbázis relációit idegen kulcsok (Foreign Keys) biztosítják:

*Termék → Kategóriák:*

ALTER TABLE arucikk ADD FOREIGN KEY (kat1) REFERENCES kategoriak(id);

Egy termék legfeljebb 3 kategóriába sorolható (kat1, kat2, kat3).

*Kosár → Termékek:*

ALTER TABLE kosar ADD FOREIGN KEY (arucikk\_id) REFERENCES arucikk(id);

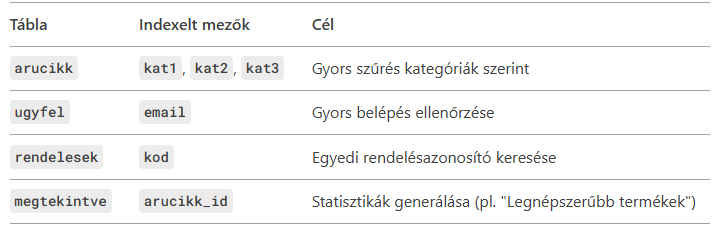
Egy kosár tétel mindig egy létező termékre hivatkozik.

*Rendelések → Felhasználók:*

ALTER TABLE rendelesek ADD FOREIGN KEY (ugyfel\_id) REFERENCES ugyfel(id);

Minden rendelés egy regisztrált felhasználóhoz tartozik.

## **5.4 Indexek és optimalizálás**

A teljesítmény növelése érdekében stratégiai indexek lettek létrehozva:

## **5.5 Biztonsági intézkedések**

**Jelszavak titkosítása**:

A ugyfel.jelszo mezőben BCrypt hashelés kerül használatra.

Példa: $2y$10$... – a 10 a titkosítási költség faktort jelöli.

**SQL Injection védelem**:

*Prepared statement-ek használata PHP-ben:*

$parancs = $kapcsolat->prepare("SELECT \* FROM ugyfel WHERE email=?");

$parancs->bind\_param("s", $email);

*Munkamenet kezelés:*

A ugyfel.session\_id és ervenyes mezők biztosítják a jogosulatlan hozzáférés kiszűrését.

## **5.6 Jelentős funkciók és adatbázis integráció**

*Többszintű kategóriák*

**Megvalósítás**: A kategoriak tábla hierarchikus struktúrája lehetővé teszi:

Akár 3 szintű kategória fa (pl. Elektronika → Laptopok → Gaming).

Rugalmas szűrés a terméklistákban (kat1, kat2, kat3 mezők).

*Kosárkezelés*

Vendég vásárlók: A session\_id menti a kosarat 7 napig (DELETE FROM kosar WHERE mikor<'$egyhete').

Készlet ellenőrzés: A kosar és arucikk táblák tranzakcióban frissülnek, hogy elkerüljük a túlértékesítést.

*Rendeléskövetés*

Státuszok: A rendelesek tábla fizetesi\_mod és teljesitve mezői nyomon követik a fizetést és szállítást.

## **5.7 Későbbi fejlesztések**

**Elavult kosarak törlése**: Heti automatikus törlés cron job segítségével.

**Full-text keresés**: A arucikk táblában a termékleírások gyors keresése.

**Többnyelvű támogatás**: Új mezők (pl. leiras\_en) a lokalizációhoz.

Ez az adatbázis szerkezet biztosítja a Wimu Webshop skálázhatóságát és a jövőbeli bővítések lehetőségét.

# **6. Részletes feladatspecifikáció, algoritmusok**

## **6.1. belepes.php – Belépési folyamat**

Funkció: Felhasználó hitelesítése e-mail és jelszó alapján.

*Paraméterek:*

$\_POST['email']: Felhasználó e-mail címe.

$\_POST['jelszo']: Felhasználó jelszava.



**Működés**:

JavaScript validálja az e-mail formátumot és a jelszó meglétét.

Szerver oldalon ellenőrzi az adatbázisban a felhasználó létezését (SELECT \* FROM ugyfel WHERE email='$email').

Hash-elt jelszó ellenőrzése password\_verify() függvénnyel.

Naplózza a sikertelen próbálkozásokat (napló tábla).

Ha 3 sikertelen próbálkozás van 30 percen belül, tiltja a belépést.

Sikeres belépés esetén beállítja a session-t és cookie-kat, átirányít a főoldalra.

Visszatérési érték: HTTP átirányítás hibákra (hiba=1, tilos=1) vagy sikeres belépés esetén index.php.

## **6.2 Főbb függvények és metódusok specifikációja**

*1. Kosár tartalmának betöltése (1\_kosar\_tartalma.php)*

**Funkció**: A felhasználó kosarában lévő termékek megjelenítése, összegzés.



*Paraméterek:*

session\_id: A felhasználó session azonosítója (ha nincs bejelentkezve).

ugyfel\_id: A bejelentkezett felhasználó azonosítója.

*Adatbázis lekérdezés:*

SELECT \* FROM kosar WHERE session\_id = ? OR ugyfel\_id = ?;

**Kimenet**: HTML tábla a termékekkel, mennyiséggel és összesített árakkal.

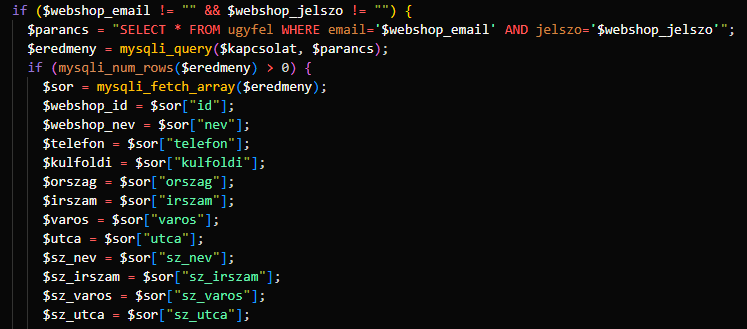
## **6.3 Vásárlói adatok validálása (2\_vasarloi\_adatok.php)**

*JavaScript függvény* ellenoriz()*:*

**Funkció**: Ellenőrzi, hogy minden kötelező mező ki van-e töltve.

**Ellenőrzött mezők**: Név, telefonszám, cím, számlázási adatok.

**Hibakezelés**: Alert üzenetek üres mezők esetén.



**Adatbázis frissítés**:

UPDATE ugyfel SET nev=?, telefon=?, ... WHERE id=?;

Paraméterek: Felhasználó által megadott adatok (pl. $\_POST['nev'], $\_POST['telefon']).

## **6.4. Fizetési mód kiválasztása (3\_fizetesi\_modok.php)**

JavaScript függvény ellenoriz():

**Funkció**: Ellenőrzi, hogy legalább egy fizetési mód ki van-e választva.

**Integráció**: PayPal SDK használata bankkártyás fizetéshez.

**Konverzió**: HUF → USD fix árfolyammal (pl. osszeg / 350).

*Adatátvitel:*

<input type="hidden" name="fizet" value="1|2|3">

Értékek: 1 (banki átutalás), 2 (utánvét), 3 (PayPal).



## **6.5. Rendelés összegzése (4\_rendeles\_osszegzes.php)**

**Funkció**: Megjeleníti a felhasználó kosarát és a kiválasztott fizetési módot.

*Adatbázis lekérdezés:*

SELECT arucikk.nev, kosar.db, arucikk.ar\_huf

FROM kosar

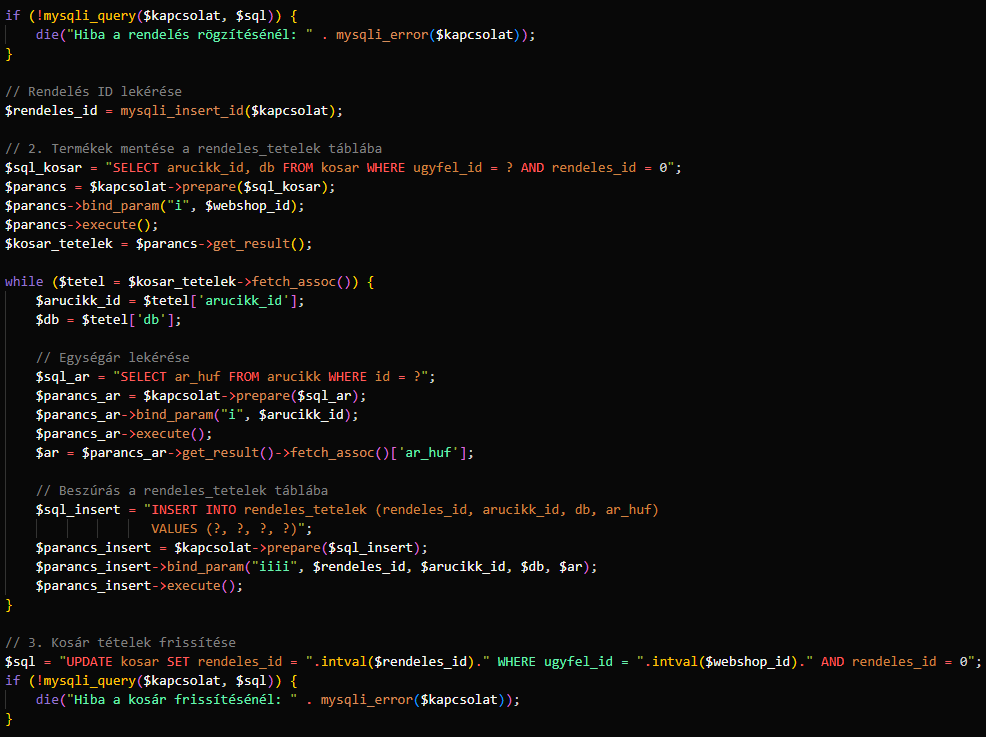
JOIN arucikk ON kosar.arucikk\_id = arucikk.id

WHERE ugyfel\_id = ?;



**Kimenet**: Tábla a termékek részleteivel és egy véglegesítő gomb.

## **6.6 Rendelés feldolgozása (5\_rendeles\_elkuld.php)**

****

*Lépések:*

*Rendelés rögzítése:*

INSERT INTO rendelesek (ugyfel\_id, kod, fizetendo, ...);

*Termékek mentése:*

INSERT INTO rendeles\_tetelek (rendeles\_id, arucikk\_id, db, ar\_huf);

*Raktárkészlet frissítése:*

UPDATE arucikk SET raktaron = raktaron - ? WHERE id = ?;

*Kosár ürítése:*

DELETE FROM kosar WHERE ugyfel\_id = ?;

**Kimenet**: Sikeres rendelés üzenet és visszaigazoló e-mail szimuláció.

## **6.7. Bejelentkezési rendszer (belepes.php)**

**Funkció**: Felhasználó hitelesítése és session kezelése.

*Ellenőrzések:*

**Jelszó ellenőrzés**: password\_verify($jelszo, $sor['jelszo']).

*Brute force védelem:*

SELECT COUNT(\*) FROM naplo WHERE email=? AND mikor > ? AND sikertelen=1;

Blokkolás, ha 3+ sikertelen próbálkozás van 30 percen belül.

**Cookie-k:** webshop\_email és webshop\_jelszo 7 napig érvényesek.

## **6.8 Algoritmusok leírása**

**1. Kosár frissítési algoritmus**

Cél: A kosárban lévő termékek mennyiségének módosítása.

*Lépések:*

A felhasználó módosítja a mennyiséget a kosár oldalon.

Űrlap elküldése a mit=modosit paraméterrel.

*Adatbázis frissítés:*

foreach ($\_POST as $kosar\_id => $db) {

UPDATE kosar SET db = ? WHERE id = ?;

}

Összeg újraszámolása és átirányítás.

## **6.9 Rendelésfeldolgozás folyamata**

**Struktogram**:

[START]

↓

[Adatbázis kapcsolat]

↓

[Rendelés létrehozása → rendelesek tábla]

↓

[Termékek másolása → rendeles\_tetelek tábla]

↓

[Raktárkészlet csökkentése]

↓

[Kosár ürítése]

↓

[Sikeres oldal megjelenítése]

[END]

## **6.10. Termékkereső algoritmus (index.php)**

*Megjegyzések:*

**Biztonsági hiányosságok:** SQL injection kockázat néhány lekérdezésben (pl. 5\_rendeles\_elkuld.php).

Jelszavak tárolása nyers formában helyett password\_hash() használata ajánlott.

**Optimalizálási lehetőségek:**

Tranzakciók használata rendelésfeldolgozásnál (ACID tulajdonságok biztosítása).

Cache-elés a gyakran lekérdezett termékadatokhoz.

## **6.11. Kosár megtekintése (kosar\_megtekintes.php)**

**Funkció**: A felhasználó kosarában lévő termékek megjelenítése, törlési műveletek kezelése, összegzés.

*Paraméterek*:

$\_POST['torles']: "egy" vagy "osszes" érték (egy elem vagy teljes kosár törlése).

$\_POST['arucikk\_id']: Törlendő termék azonosítója.

$\_POST['db']: Törlendő mennyiség (csak egy elem törlése esetén).

*Működés:*

Session/Cookie alapján felhasználó azonosítása:

Ha bejelentkezett (ugyfel\_id), különben session\_id használata.

*Kosár tartalmának lekérdezése:*

SELECT \* FROM kosar

WHERE (ugyfel\_id = ? OR session\_id = ?) AND rendeles\_id = 0;

*Raktár visszaállítás minden tételre:*

UPDATE arucikk SET raktaron = raktaron + ? WHERE id = ?;

*Kosár ürítése:*

DELETE FROM kosar WHERE (ugyfel\_id = ? OR session\_id = ?);

*Raktár visszaállítás:*

UPDATE arucikk SET raktaron = raktaron + ? WHERE id = ?;

*Tétel eltávolítása:*

DELETE FROM kosar WHERE (ugyfel\_id = ? OR session\_id = ?) AND arucikk\_id = ? LIMIT 1;

**Kimenet**: HTML tábla a kosár tartalmával és összesített árral.

## **6.12. Termék kosárba tétele (kosarba\_tesz.php)**

**Funkció**: Termék hozzáadása a kosárhoz vagy mennyiség frissítése.

*Paraméterek:*

$\_POST['arucikk\_id']: Termék azonosítója.

$\_POST['db']: Hozzáadandó mennyiség.

*Készlet ellenőrzése:*

SELECT raktaron FROM arucikk WHERE id = ?;

Ha kert\_mennyiseg > raktaron: hibaüzenet.

Felhasználó azonosítása: Bejelentkezett (ugyfel\_id) vagy vendég (session\_id).

*Ha a termék már szerepel a kosárban:*

UPDATE kosar SET db = db + ? WHERE (ugyfel\_id = ? OR session\_id = ?) AND arucikk\_id = ?;

*Új termék esetén:*

INSERT INTO kosar (arucikk\_id, ugyfel\_id/session\_id, db) VALUES (?, ?, ?);

*Raktár frissítése:*

UPDATE arucikk SET raktaron = raktaron - ? WHERE id = ?;

**Visszatérés**: JSON válasz (success: true vagy hibaüzenet).

**6.13. Profil módosítás (profil\_modosit.php és profil\_modosit\_2.php)**

**Funkció**: Felhasználói adatok és jelszó módosítása.

*Paraméterek:*

$\_POST['nev']: Új név.

$\_POST['email']: Új email.

$\_POST['uj\_jelszo'] és $\_POST['uj\_jelszo\_2']: Új jelszó és megerősítés.

*JavaScript:*

*Üres mezők ellenőrzése.*

Email formátum ellenőrzése (helyescim()).

Jelszó egyezés ellenőrzése.

Szerver oldali ellenőrzés:

*Email egyediség:*

SELECT id FROM ugyfel WHERE email = ? AND id != ?;

Jelszó hash-elés: password\_hash().

*Adatbázis művelet:*

UPDATE ugyfel SET nev = ?, email = ?, jelszo = ? WHERE id = ?;

*Kimenet*:

Sikeres frissítés: $\_SESSION['siker'] üzenet.

Hiba: $\_SESSION['hiba'] üzenet.

## **6.14. Regisztráció (reg.php és reg\_ellenoriz.php)**

**Funkció**: Új felhasználó regisztrálása.

*Paraméterek:*

$\_POST['emailcim']: Felhasználó email címe.

$\_POST['nev']: Teljes név.

$\_POST['jelszo'] és $\_POST['jelszo2']: Jelszó és megerősítés.

Email formátum, jelszó egyezés, üres mezők.

*Jelszó komplexitás (8 karakter, nagybetű, szám):*

preg\_match('/^(?=.\*[A-Z])(?=.\*\d).{8,}$/', $jelszo);

*Email egyediség:*

SELECT \* FROM ugyfel WHERE email = ?;

*Adatbázis művelet:*

INSERT INTO ugyfel (email, nev, jelszo, reg\_idopont) VALUES (?, ?, ?, ?);

*Kimenet:*

**Sikeres regisztráció**: Átirányítás bejelentkezésre.

**Hiba**: Üzenet a hibáról (pl. "Email már foglalt").

## **6.15. Termék listázás és keresés (index.php)**

**Funkció**: Termékek szűrése kategóriák és keresési feltételek alapján.

*Paraméterek:*

$\_POST['kat1'], $\_POST['kat2'], $\_POST['kat3']: Kategóriák.

$\_POST['mitkeres']: Keresett kulcsszó.

$\_POST['sorrend']: Rendezési szempont (ár, név, népszerűség).

*Kategóriák hierarchikus szűrése:*

Ha kat3 > 0: kat1, kat2, és kat3 alapján.

Ha kat2 > 0: kat1 és kat2 alapján.

Alapértelmezett: kat1 vagy összes termék.

*Keresés a termék nevében, leírásában:*

SELECT \* FROM arucikk

WHERE (nev LIKE ? OR leiras LIKE ?)

AND katszuro\_feltetel

ORDER BY sorrend;

*Pagináció:*

oldal és laponkent paraméterek kezelése.

**Kimenet**: Termékek megjelenítése kártyákon, keresési eredmények frissítése.

## **6.16. Algoritmusok struktogramokkal**

*1. Regisztrációs Folyamat:*

[START]

↓

[Űrlap betöltése (reg.php)]

↓

[Adatok elküldése → reg\_ellenoriz.php]

↓

[Validáció: Email, jelszó]

↓

IF (Hiba) → Hibaüzenet

ELSE

↓

[Adatbázisba beszúrás]

↓

[Sikeres regisztráció üzenet]

[END]

*2. Profil Módosítás:*

[START]

↓

[Adatok bekérése (profil\_modosit.php)]

↓

[Validáció: Email, jelszó]

↓

IF (Hiba) → Hibaüzenet

ELSE

↓

[Adatbázis frissítése]

↓

[Sikeres üzenet]

[END]

*3. Kosárba Tétel:*

[START]

↓

[Termék kiválasztása]

↓

[Készlet ellenőrzése]

↓

IF (Nincs elég készlet) → Hiba

ELSE

↓

[Kosár frissítése (INSERT/UPDATE)]

↓

[Raktárkészlet csökkentése]

↓

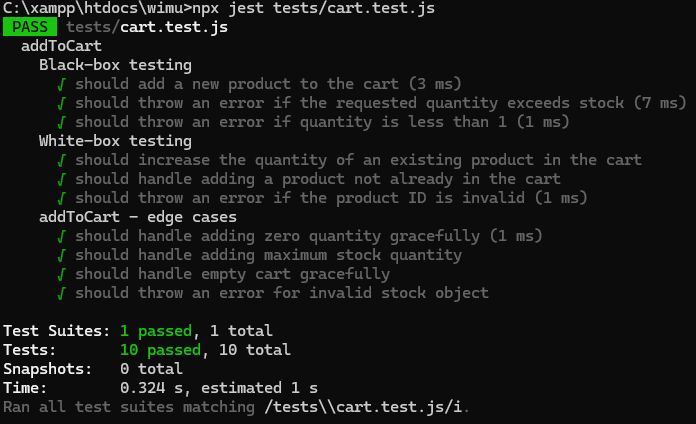
[Sikeres válasz] [END]

# **7. Unit teszt**

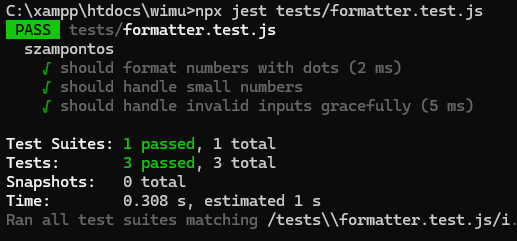
## **7.1. Bevezetés**

A dokumentáció célja, hogy bemutassa a különböző tesztelési módszereket, szinteket és technikákat egy webalkalmazás tesztelési folyamatában. A dokumentáció a következő modulokra fókuszál:

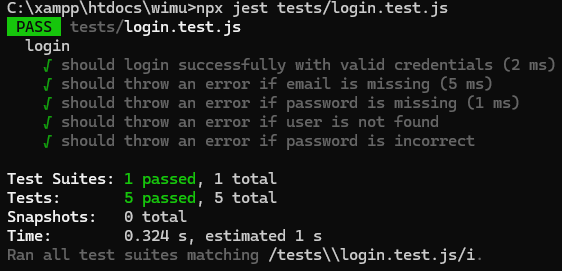
*Kosárkezelés* (cart.js)



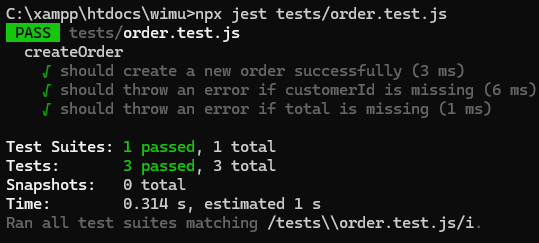
*Számformázás* (formatter.js)

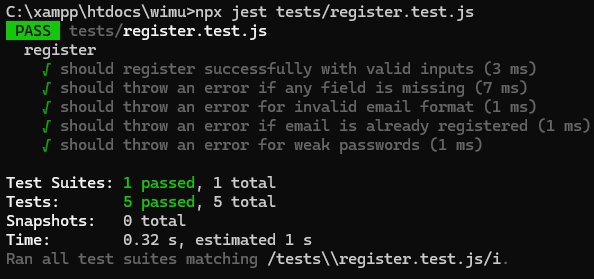


*Bejelentkezés* (login.js)

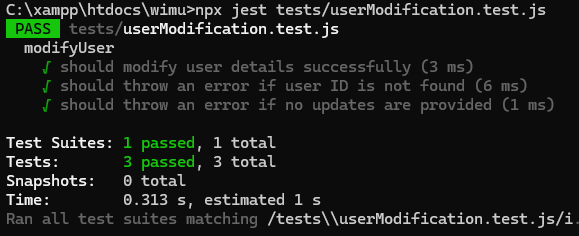


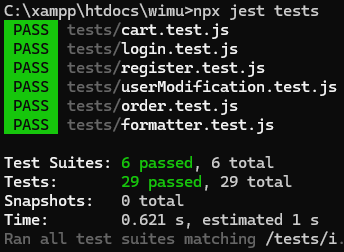
*Rendeléskezelés* (order.js)



*Regisztráció* (register.js)

*Felhasználómódosítás* (userModification.js)



*Teljes teszt futtatása:*

## **7.2. Tesztelési módszerek**

*Fekete Doboz Tesztelés*

A bemenet-kimenet viselkedést vizsgálja, anélkül, hogy a belső implementációt ismerné.

*Példák:*

*Kosárkezelés* (cart.test.js):

should add a new product to the cart: Ellenőrzi, hogy új termék hozzáadása esetén a kosár tartalmazza-e az új elemet.

should throw an error if quantity is less than 1: Ellenőrzi, hogy a rendszer hibát dob-e, ha a mennyiség érvénytelen.

*Bejelentkezés* (login.test.js):

should login successfully with valid credentials: Ellenőrzi, hogy érvényes hitelesítő adatok esetén sikeres bejelentkezés történik-e.

## **7.3 Fehér doboz tesztelés**

A belső logikára és struktúrára fókuszál, például feltételes ágak, ciklusok vagy hibakezelés tesztelésére.

*Példák:*

*Kosárkezelés* (cart.test.js):

should increase the quantity of an existing product in the cart: Ellenőrzi, hogy a rendszer frissíti-e a mennyiséget, ha a termék már szerepel a kosárban.

should throw an error if the product ID is invalid: Ellenőrzi a hibakezelést érvénytelen termékazonosító esetén.

*Számformázás (formatter.test.js):*

should handle negative numbers: Ellenőrzi, hogy a függvény helyesen formáz-e negatív számokat.

## **7.4. Tesztelési szintek**

*Unit Tesztelés*

Egyedi függvények vagy modulok tesztelése elszigetelten.

*Példák:*

Számformázás (formatter.test.js):

should format numbers with dots: Ellenőrzi a szampontos függvényt különböző bemenetekkel (pl. 1000 → 1.000).

*Regisztráció* (register.test.js):

should throw an error for weak passwords: Ellenőrzi a jelszóerősség követelményeit.

## **7.5 Integrációs tesztelés**

Modulok közötti interakciók tesztelése (jelenlegi példák főleg unit tesztek, de példa integrációs tesztre):

Rendelés létrehozása (order.test.js):

should create a new order successfully: Ellenőrzi, hogy a rendelés létrehozása frissíti-e a rendelések listáját.

## **7.6 Rendszertesztelés**

A teljes alkalmazás végpontokon keresztül történő tesztelése (nincs benne a példákban, de példa lehetne):

Bejelentkezés + Kosárkezelés:

Egy felhasználó bejelentkezik, hozzáad egy terméket a kosarához, majd létrehoz egy rendelést.

## **7.7. Tesztelési technikák**

*Határérték-elemzés*

**Kosárkezelés** (cart.test.js):

should handle adding maximum stock quantity: Ellenőrzi, hogy a rendszer engedélyezi-e a maximális raktárkészlet hozzáadását (pl. 10 darab).

## **7.8 Ekvivalencia particionálás**

*Bejelentkezés* (login.test.js):

**Érvényes partíció**: test@example.com + password123 → Sikeres bejelentkezés.

**Érvénytelen partíció**: unknown@example.com + bármilyen jelszó → "User not found" hiba.

## **7.9 Állapotgép-alapú tesztelés**

*Felhasználómódosítás* (userModification.test.js):

should modify user details successfully: Ellenőrzi, hogy a felhasználói adatok frissülnek-e a megfelelő állapotváltozással.

## **7.10. Tesztkörnyezet és eredmények**

**Keretrendszer**: Jest (JavaScript tesztelési keretrendszer).

**Tesztlefedettség**: A unit tesztek lefedik a kritikus funkciókat (pl. hibakezelés, formázás).

*Talált Hibák:*

A szampontos függvény nem kezeli a tizedes számokat konzisztensen (pl. 1234.56 → 1.234.56, ami nem szabványos).

A modifyUser függvény nem ellenőrzi az érvénytelen mezőket (pl. role: 'invalid\_role').

## **7.11. Összegzés és javaslatok**

**Erősségek**: A tesztek átfogóan lefedik az alapvető funkciókat és hibakezeléseket.

*Fejlesztési lehetőségek:*

Bővíteni kell az integrációs teszteket (pl. kosár + rendelés létrehozása).

Implementálni kell tizedes számok kezelését a szampontos függvényben.

**Következő lépések:**Tesztelési automatizáció bevezetése CI/CD folyamatokba.

# **8. Fejlesztés**

## **8.1 Tényleges rendelés**

A rendelés még fejlesztés alatt van, még nincsen beépítve, hogy ténylegesen lehessen rendelni.

## **8.2 Számla kiállítás**

A számlát gyorsan be lehet integrálni a webshopba. Csak még fejlesztés alatt van.

## **8.3 Email cím megerősítés**

Egy levelet küld a felhasználónak, aki megadta az email címét, majd ott meg kell erősítenie az ügyfélnek. Így működne a jelszó/email cím változtatás az oldalon, hogy meg kell erősítenie.

## **8.4 Több termék feltöltés a webshopba**

Több kategóriát és termékeket szeretnénk feltölteni az oldalra, hogy még több rendelés legyen.

## **8.5 Szállítási cím változtatás**

Most úgy van, hogy ha rendel az ügyfél, akkor megadja a szállítási és számlázási adatokat. Majd szeretnénk ezen változtatni úgy, hogy többféle szállítási címeket tudjon megadni a felhasználó és módosítani azt.

## **8.6 Admin felület fejlesztése**

Az admin felületnek a feljesztése. Mint például, hogy a rendeléseket a felületen lehessen elfogadni vagy elutasítani. A felhasználók adatainak módosítása, törlése.

# **9. Összegzés**

## **9.1 Szakmai fejlődés**

*Webfejlesztési technológiák elsajátítása:*

A projekt során mélyreható tapasztalatot szereztünk a PHP és MySQL alapú backend-fejlesztésben. Kiemelten foglalkoztunk a felhasználókezelés, a kosárrendszer működésének kialakításával, valamint a hatékony SQL lekérdezések tervezésével és optimalizálásával. Megismerkedtünk a biztonságos programozási gyakorlatokkal, különös tekintettel az SQL injection elleni védekezésre (prepared statements alkalmazása), a jelszavak biztonságos tárolására és kezelésére (pl. password\_hash, password\_verify), valamint a session-ök megfelelő kezelésére.

*Komplex rendszertervezés:*

A webshop moduláris felépítése során fontos szempont volt az egyes funkciók – regisztráció, kosárkezelés, rendelés – összehangolt működésének biztosítása. Ennek megvalósítása hozzájárult rendszerszintű gondolkodásunk fejlődéséhez, és betekintést adott a skálázható alkalmazások tervezési elveibe.

*Frontend fejlesztés és UX:*

A felhasználói felület reszponzív kialakítását Bootstrap segítségével végeztük, míg a dinamikus funkciók – mint például a kosár valós idejű frissítése – JavaScript és AJAX alkalmazásával valósultak meg. A felhasználói élmény javítását célzó fejlesztések között szerepeltek a JavaScript-alapú űrlapvalidációk, hibakezelési mechanizmusok, valamint vizuális visszajelzések tervezése, amelyek segítik az átlátható és gördülékeny vásárlást.

*Tesztelési gyakorlat:*

A projekt során különböző tesztelési módszereket alkalmaztunk a szoftver megbízhatóságának növelése érdekében. Használtunk fekete doboz tesztelést a főbb funkciók – például kosárba helyezés, jelszóváltoztatás – működésének ellenőrzésére, valamint unit teszteket az egyes komponensek hibáinak proaktív azonosítására.

## **9.2 Jövőbeli célok**

*A rendszer funkcionalitásának bővítése:*

A közeljövőben tervezzük további fizetési lehetőségek – például Stripe vagy közvetlen bankkártyás fizetés – integrálását, illetve egy ajánlórendszer bevezetését, amely felhasználói viselkedési minták alapján személyre szabott termékjavaslatokat kínálna.

*Optimalizálás és technológiai fejlesztés:*

A teljesítmény növelése érdekében célunk az adatbázis indexelése, valamint gyorsítótárazási megoldások bevezetése a gyakran lekérdezett adatok esetében. Emellett hosszú távú célként szerepel a felhasználói élmény jelentős javítása egy modern Single Page Application (SPA) architektúra megvalósításával React vagy Vue.js keretrendszer segítségével.

*Tesztelési folyamatok fejlesztése:*

A jelenlegi manuális tesztelési eljárásokat szeretnénk automatizálni, bevezetve integrációs teszteket, amelyek a különálló modulok együttműködését vizsgálják. Továbbá célunk CI/CD eszközök – például GitHub Actions – alkalmazása a fejlesztési folyamatok hatékonyabbá tételére.

*Biztonság fokozása:*

A rendszer biztonságának növelése érdekében tervezzük kétlépcsős azonosítás (2FA) bevezetését, valamint rendszeres biztonsági auditok és automatikus sebezhetőségi vizsgálatok alkalmazását a felhasználói adatok védelmének biztosítására.