**Baranya Megyei SZC Simonyi Károly Technikum**

**és Szakképző Iskola**

Vizsgaremek

|  |  |
| --- | --- |
| Készítették: | ***Hermann Máté***  ***Gál Martin***  ***Pancza Milán*** |

**PÉCS 2025**

*Baranya Megyei SZC Simonyi Károly Technikum és Szakképző Iskola*

*Szakma megnevezése: Szoftverfejlesztő és –tesztelő*

*A szakma azonosító száma: 5 0613 12 03*

***Záródolgozat***

***Cím: Webshop Hálózata***

*Készítette: Név:* Hermann Máté

*Név:* Gál Martin

*Név:* Pancza Milán

***2025***

***Nyilatkozat***

Aluírott, büntetőjogi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy a benyújtott záródolgozatom saját, önálló munkám. Az abban hivatkozott nyomtatott és elektronikus szakirodalom felhasználása a szerzői jogok szabályainak megfelelően készült. Tudomásul veszem, hogy záródolgozat esetén plágiumnak számít:

* szószerinti, vagy attól kismértékben eltérő idézet közlése idézőjel és hivatkozás megjelölése nélkül.
* tartalmi idézet hivatkozás megjelölése nélkül
* más publikált gondolatainak (cikk, dolgozat) sajátomként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén záródolgozatom visszautasításra kerül.

Kelt.: Pécs, 2025. …………………….

................................................

Tanuló

................................................

Tanuló

................................................

Tanuló

Tartalomjegyzék

[1. PROJEKT BEMUTATÁSA 5](#_Toc195019671)

[2. Telepítési záradék 8](#_Toc195019672)

[1. Szükséges alkalmazások és eszközök 8](#_Toc195019673)

[2. Telepítés lépései 8](#_Toc195019674)

[3. Figma Desing Látványterv 10](#_Toc195019675)

[4. Adatbázis 11](#_Toc195019676)

[5. Backend 16](#_Toc195019677)

[6. Postman 19](#_Toc195019678)

[1. API dokumentáció: 21](#_Toc195019679)

[7. Frontend 25](#_Toc195019680)

[8. Angular 33](#_Toc195019681)

[1. Angular projektstruktúra bemutatása 34](#_Toc195019682)

[2. Főbb mappák: 34](#_Toc195019683)

[3. Rövid bemutatása a főbb komponenseknek: 35](#_Toc195019684)

[4. Űrlapkezelés és validáció Angularban 37](#_Toc195019685)

[5. Állapotkezelés Angularban 38](#_Toc195019686)

[6. Reszponzivitás és felhasználói élmény (UX) 38](#_Toc195019687)

[7. Admin felület – működés és funkciók 40](#_Toc195019688)

[9. ChatGPT/Mesterséges Intelligencia 41](#_Toc195019689)

[10. Záró gondolatok 45](#_Toc195019690)

[11. Források 46](#_Toc195019691)

# PROJEKT BEMUTATÁSA

A projekt a következő publikus GitHub linken tekinthető meg:

<https://github.com/hermannmate420/ProjectFeladat>

A vizsgaremekünk témája egy weboldalt hoz létre, ami a Retro Vintage fantázianévre hallgat. Ez a weboldal arra szolgál a kliens megtekintők számára, hogy igazi ereklyéket, „kincseket” tudjanak vásárolni interneten keresztül. Oldalunk lehetőséget ad különböző termékek megszerzésére, vásárlására, emellett más gyűjtőkkel való beszélgetésre, esetleges adok-kapok cserék lebonyolítására.

Kifejezetten hasonló projektet nem találtunk az adatok gyűjtése közben, így úgymond minimálisan eredeti ötlettel állhattunk elő.

A weboldal ugyebár 4 részből áll össze.

1. Első körben egy Figma design-t hoztunk létre az általános színek kiválasztására, a megadott terv elrendezésére, és a hivatkozások megtervezésére, annak érdekében, hogy tudjuk, melyik gomb melyik oldalhoz fog kötődni.
2. Tartalmaz egy adatbázist, amelyet a MAMP nevű programban és phpMyAdmin-ban hoztunk létre.
3. Készült egy Backend felület, ami a Java programban készült el.
4. Tartalmaz egy Frontend-et, ami első körben Visual Studio Code-ban készült el, mint alap vázlat. Majd ezt követően az Angular keretrendszert használtuk fel a teljes projekt külső megjelenítésére és tesztelésére.

A projekt fejlesztése során többféle technológiát alkalmaztunk a különböző rétegekhez. A **backend oldalt Java nyelven** készítettük, ahol REST API-n keresztül biztosítjuk a kliensoldali kommunikációt. Az **adatbázis-kezeléshez MySQL rendszert** használtunk, a kapcsolódó adminisztrációt pedig a phpMyAdmin felületen végeztük. A **frontend felületet kezdetben HTML, CSS és JavaScript segítségével** alakítottuk ki, majd a végleges verzióban **Angular keretrendszerrel** valósítottuk meg az alkalmazás dinamikus működését. A rendszer kialakítása során szem előtt tartottuk a moduláris fejlesztést, a felhasználói élményt, valamint a biztonságot is, különös tekintettel az autentikációra és jelszókezelésre.

A csapatmunka bemutatása:

* + - 1. Hermann Máté – Gál Martin:
         1. Elkészítették az adatbázist, töltötték fel tárolt eljárásokkal, emellett töltötték fel adatokkal.
      2. Hermann Máté:
         1. Készítette el a Backend kódot a Java programban. Például a Login-hoz tartozó kódot is.
      3. Gál Martin – Pancza Milán:
         1. Elkészítették a Frontend kódot első körben Visual Studio Code-ban majd pedig annak mintájára Angular keretrendszert használva fejlesztették tovább, alakították át a design-t.
      4. Pancza Milán:
         1. Készítette el a dokumentációt, emellett pedig a Figma programot felhasználva készítette el a látványtervet.

# Telepítési záradék

## **Szükséges alkalmazások és eszközök**

| **Alkalmazás** | **Verzió** | **Funkció** | **Minimális gépigény** |
| --- | --- | --- | --- |
| **Java JDK** | 1.8 (Default) | Backend futtatása, Java kód fordítása | 1 GHz CPU, 512 MB RAM |
| **Apache Maven** | 3.6+ | Java projekt buildelése és war fájl generálása | 500 MB szabad tárhely |
| **WildFly szerver** | 26.1.1 Final | Java EE alkalmazás futtatása (war fájl) | 1 GHz CPU, 1 GB RAM |
| **MySQL** | 5.7.24 | Adatbázis-kezelés | 1 GHz CPU, 512 MB RAM |
| **phpMyAdmin** | 5.1.2 | Adatbázis vi1zuális kezelése | Webes böngésző |
| **Angular CLI** | 19 | Frontend buildelése és fejlesztői szerver | 2 GB RAM, Node.js szükséges |
| **Node.js + npm** | 23.4.0 + 10.9.2 | Angular futtatásához szükséges | 1 GHz CPU, 2 GB RAM |
| **Postman** | bármely | REST API-k tesztelése | 1. RAM |

## 2. Telepítés lépései

**1. Backend (Java + Maven + WildFly)**

1. Telepítsd a Java JDK-t, Maven-t, és a WildFly szervert.
2. Nyisd meg a vintage\_project mappát NetBeans-ben vagy IntelliJ IDEA-ban.
3. Buildeld a projektet Maven-nél:

* mvn clean install

1. A build eredményeként létrejön a vintage\_project.war fájl a target/ mappában.
2. Másold a .war fájlt a WildFly standalone/deployments/ könyvtárába.
3. Indítsd el a WildFly szervert:

* standalone.bat (Windows) vagy standalone.sh (Linux/Mac)

1. A backend elérhető lesz a http://localhost:8080/vintage\_project címen.

**2. Adatbázis (MySQL + phpMyAdmin)**

1. Hozz létre egy adatbázist (pl. vintage\_db).
2. Importáld az SQL fájlt (database.sql vagy hasonló).
3. A Java backend persistence.xml vagy ApplicationConfig.java fájljában állítsd be az adatbázis csatlakozási adatokat (felhasználónév, jelszó, JDBC URL).

**3. Frontend (Angular)**

1. Telepítsd a Node.js-t és Angular CLI-t:

* npm install -g @angular/cli

1. Navigálj az Angular projekt mappájába (cd frontend)
2. Telepítsd a csomagokat:

* npm install

1. Indítsd el fejlesztői módban:

* ng serve

1. A frontend alkalmazás elérhető lesz a http://localhost:4200 címen.
2. **Összefoglalás**

A rendszer fejlesztéséhez és futtatásához egy átlagos fejlesztői számítógép is elegendő. A minimális javasolt hardverkonfiguráció:

* 4 GB RAM
* 2 magos processzor
* 10 GB szabad tárhely
* Internetkapcsolat (npm, Maven telepítéshez)

A telepítési lépések végrehajtása után a rendszer használatra kész.

# Figma Desing Látványterv

A **Figma** egy népszerű online design és prototípus-készítő eszköz, amelyet főként UI/UX tervezéshez használnak.

* **Mire jó?**
  + - Web- és mobilalkalmazások felhasználói felületének tervezésére
    - Prototípusok és interaktív modellek készítésére
    - Együttműködésre a csapatok között valós időben
* **Mire használható?**
  + - Grafikai tervezésre és vektorgrafikák létrehozására
    - Drótvázak (wireframe-ek) és vizuális koncepciók készítésére
    - Design rendszerek és UI komponensek kezelésére

Fő előnye, hogy böngészőből fut, így nincs szükség telepítésre, és könnyen megoszthatók a projektek másokkal.

A képen képernyőkép, szöveg, Multimédiás szoftver, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

1. ábra: Figma Desing látványterv

A tervezés során törekedtünk arra, hogy az oldal megjelenése a célközönség ízléséhez igazodjon, ezért a Figma eszközt nem csupán látványterv készítésére használtuk, hanem a teljes **felhasználói élmény (UX)** átgondolására is. A komponens-alapú tervezés lehetővé tette az ismétlődő elemek (pl. gombok, kártyák, navigáció) újra felhasználását, ami az Angular komponenslogikájával is összhangban van. A színpaletta kialakításánál figyelembe vettük a **kontrasztarányokat és olvashatóságot**, különösen a termékoldalak és a vásárlási folyamat során. A prototípus tesztelésével még fejlesztés előtt kiszűrhettük az esetleges navigációs problémákat, így jelentősen csökkentettük az újra tervezési igényt a fejlesztés közben.

A képen képernyőkép, pálmafa, szöveg, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

2. ábra: Figma Desing szín paletta

# Adatbázis

Az adatbáziskezelésre mi a MAMP-ot használtuk. Ennek a programnak az előnyeit, és hogy mire használjuk itt olvashatják:

A **MAMP** egy lokális szerverkörnyezet, amely segítségével fejlesztők könnyen futtathatnak és tesztelhetnek PHP-alapú weboldalakat a saját gépükön.

* **Mire jó?**
  + Apache, MySQL és PHP egyetlen csomagban történő telepítésére
  + Webfejlesztési projektek tesztelésére internetkapcsolat nélkül
* **Mire használható?**
  + WordPress és más PHP-alapú rendszerek futtatására
  + Adatbázis-kezelésre a **phpMyAdmin** segítségével
  + Webalkalmazások fejlesztésére és hibakeresésére

A **phpMyAdmin** egy böngészőalapú eszköz, amely megkönnyíti a MySQL-adatbázisok kezelését grafikus felületen keresztül.

A következő ábrán látható lesz, hogy milyen táblákat hoztunk létre a weboldalhoz.

A képen szöveg, képernyőkép, Betűtípus, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

3. ábra: Az adatbázisban elkészült táblák

Az adatbázis tervezésénél az **adatnormalizálás alapelveit** követtük annak érdekében, hogy elkerüljük az adatredundanciát és biztosítsuk az adatok konzisztenciáját. A táblák közötti kapcsolatokat **idegen kulcsok (foreign key)** segítségével definiáltuk, például a rendelt\_termékek tábla rendelés\_id mezője a rendelés tábla azonosítójára hivatkozik, míg a termék\_id a termékek táblához kapcsolódik.

A felhasználók tábla kapcsolódik a rendelés és a szállítási\_cím táblákhoz is, lehetővé téve a több szállítási cím és több rendelés kezelését egyazon felhasználóhoz kapcsolódva. A teszt\_visszajelzés tábla segítségével visszajelzések rögzítésére is lehetőséget biztosítunk, amelyeket később az adminisztrációs felületen lehet értékelni.

Emellett több **tárolt eljárást (stored procedure)** hoztunk létre, amelyekkel bizonyos lekérdezéseket vagy műveleteket automatizálni tudunk – például a termékkategóriák listázása, új rendelések rögzítése vagy felhasználói adatok frissítése során. Ez egyrészt gyorsítja a működést, másrészt biztonságosabbá is teszi az adatbázissal való kommunikációt.

A tárolt eljárások között külön funkciót kapott a **felhasználói jelszavak kezelése** is. A rendszer a jelszavak titkosítását és ellenőrzését **az adatbázis szintjén végzi**, nem a backend oldalon. Ehhez **SHA-1 algoritmust** használunk.

Amikor egy felhasználó bejelentkezik vagy jelszót módosít, a backend továbbítja az adatokat a megfelelő SQL eljárásnak, amely a jelszót SHA-1 algoritmussal titkosítja, majd az adatbázisban tárolt titkosított változattal összehasonlítja. Amennyiben egyezést talál, sikeres bejelentkezés történik, vagy az új jelszót menti el az adatbázis.

Ez a megközelítés garantálja, hogy a jelszavak biztonságosan legyenek kezelve anélkül, hogy a backendnek közvetlen hozzáférése lenne a titkosításhoz vagy az eredeti jelszavakhoz.

Ezeken a táblákon felül a Procedures alatt létrehoztunk tárolt eljárásokat is.

A képen szöveg, képernyőkép, szám, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

4. ábra: Tárolt eljárások

Ezeken felül pedig feltöltöttük a tábláinkat adatokkal, amiket később a weboldalon meg fogunk jeleníteni. A projekt haladásával a képen látható termékek bővültek, ennek megtörténéséhez igénybe vettük a mesterséges intelligenciát, ami mindegyik kategóriánkhoz generált számunkra 10-10 terméket. Ennek következményeképpen a termékeink száma több mint 200-ra nőtt.

A képen szöveg, képernyőkép, szám, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

5. ábra: Kategóriák tábla feltöltött formában

# Backend

A rendszer backend oldala Java nyelven készült, Maven projektstruktúrát használva. A fejlesztés során célunk egy modulárisan felépített, könnyen karbantartható és biztonságos szerveroldali alkalmazás létrehozása volt, amely biztosítja a frontend számára az összes szükséges adatkezelési funkciót.

A backend alkalmazást **WildFly alkalmazásszerveren** futtatjuk, amely megbízható és jól integrálható Java EE környezetet biztosít. A WildFly támogatja a RESTful szolgáltatásokat, így ideális választás volt az API-alapú kommunikációhoz. A Maven projektmenedzsment segítségével hatékonyan kezeltük a függőségeket és a build folyamatokat.

A backend a következő főbb funkciókat valósítja meg:

* Felhasználói regisztráció, bejelentkezés és jogosultságkezelés
* Jelszóváltoztatás és validáció
* Termékek és kategóriák lekérdezése, kezelése
* Kosárműveletek, rendelés leadása
* Admin funkciók: statisztikák, felhasználók kezelése

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

6. ábra: Backend projekt felépítése (Maven struktúrában, NetBeans környezetben)

**Magyarázó szöveg:**

A fenti ábrán látható a vintage\_project Java alapú backend szerkezete, amelyet **Maven projektként** hoztunk létre. A fájlszerkezet jól elkülöníti a különböző logikai rétegeket:

**Config csomag:**

* A projekt konfigurációs beállításait tartalmazza, például a JWT kezelését (JWT.java) és a kulcsgenerálást (GenerateKey.java).

**Controller csomag:**

* Itt találhatók a REST végpontokat kezelő osztályok: például a UserController.java és ProductController.java, amelyek felelősek a felhasználói és termékekkel kapcsolatos HTTP kérések kezeléséért.

**Exceptions és filters csomag:**

* A ExceptionLogger.java fájl segít a hibák naplózásában.
* A corsFilter.java felelős a CORS beállításokért, hogy a frontend hozzáférhessen a backendhez.

**Model csomag:**

* Az adatmodelleket tartalmazza: például User.java, Products.java és Category.java. Ezek határozzák meg, hogy az egyes entitások milyen adatokat tartalmaznak.

**Service csomag:**

* Az üzleti logikáért felelős osztályokat tartalmazza.  
  A UserService.java és ProductService.java valósítják meg azokat a műveleteket, amelyekkel a controller osztályok dolgoznak.

**Test Packages:**

* A model és service rétegekhez tartozó egységtesztek külön csomagban helyezkednek el (UserModelTest.java, UserServiceTests.java), ezzel támogatva a kód tesztelhetőségét és fenntarthatóságát.

# Postman

**Mi az a Postman?**

A **Postman** egy fejlesztői eszköz, amit elsősorban API-k (alkalmazásprogramozási interfészek) tesztelésére, dokumentálására és fejlesztésére használnak. Különösen népszerű backend-fejlesztők, tesztelők, DevOps szakemberek és szoftverfejlesztők körében.

**Előnyei**

1. **Felhasználóbarát felület**: Könnyen átlátható GUI, még azoknak is, akik nem szuper jártasak az API-k világában.
2. **Könnyű API-tesztelés**: Egyszerűen lehet GET, POST, PUT, DELETE stb. kéréseket küldeni, megnézni a válaszokat és hibakeresni.
3. **Környezeti változók kezelése**: Többféle környezetet (pl. teszt, staging, éles) lehet definiálni változókkal.
4. **Automatizálható tesztek**: JavaScript-ben írt teszt szkriptekkel ellenőrizhető, hogy az API válasza megfelelő-e.
5. **Csapatmunka támogatása**: Megoszthatók a kollekciók, dokumentációk és workspace-ek egy csapaton belül.
6. **Mock szerverek létrehozása**: Tesztelhető egy API akkor is, ha a backend még nem készült el.
7. **Dokumentáció generálás**: Automatikusan generálható és frissíthető API dokumentáció.

**Hátrányai**

1. **Erőforrásigényes lehet**: Főleg nagy kollekciók vagy gyenge gépeken lassulhat.
2. **Ingyenes verzió korlátai**: Néhány fejlettebb funkció (pl. együttműködés, analitika) csak a fizetős verzióban érhető el.
3. **Nagy tanulási görbe haladó szinten**: A basic funkciók egyszerűek, de a pre-request szkriptek, chaining és komplex környezetek időigényesebbek lehetnek.
4. **Offline használat korlátozott**: Egyes funkciók internetkapcsolatot igényelnek, főleg csapatmunkánál.

**Mire használják az emberek?**

* **API-k tesztelése**: Kérések küldése REST, SOAP vagy GraphQL API-khoz.
* **Fejlesztés közbeni ellenőrzés**: Backend endpointok tesztelése frontend implementáció előtt.
* **Automatizált tesztek futtatása**: Validálás, hogy egy API mindig a kívánt formátumban válaszol.
* **API dokumentáció készítése és megosztása**: Könnyen olvasható és megosztható doksik a csapat számára.
* **Mock API szerverek futtatása**: Hasznos, ha a kliens fejlesztése párhuzamosan történik a backenddel.
* **Monitoring**: Bizonyos időközönként automatikus lekérdezések küldése API-khoz.

A képen szöveg, képernyőkép, tervezés látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

7. ábra: A Postman-ben készült POST, GET, PUT kérések

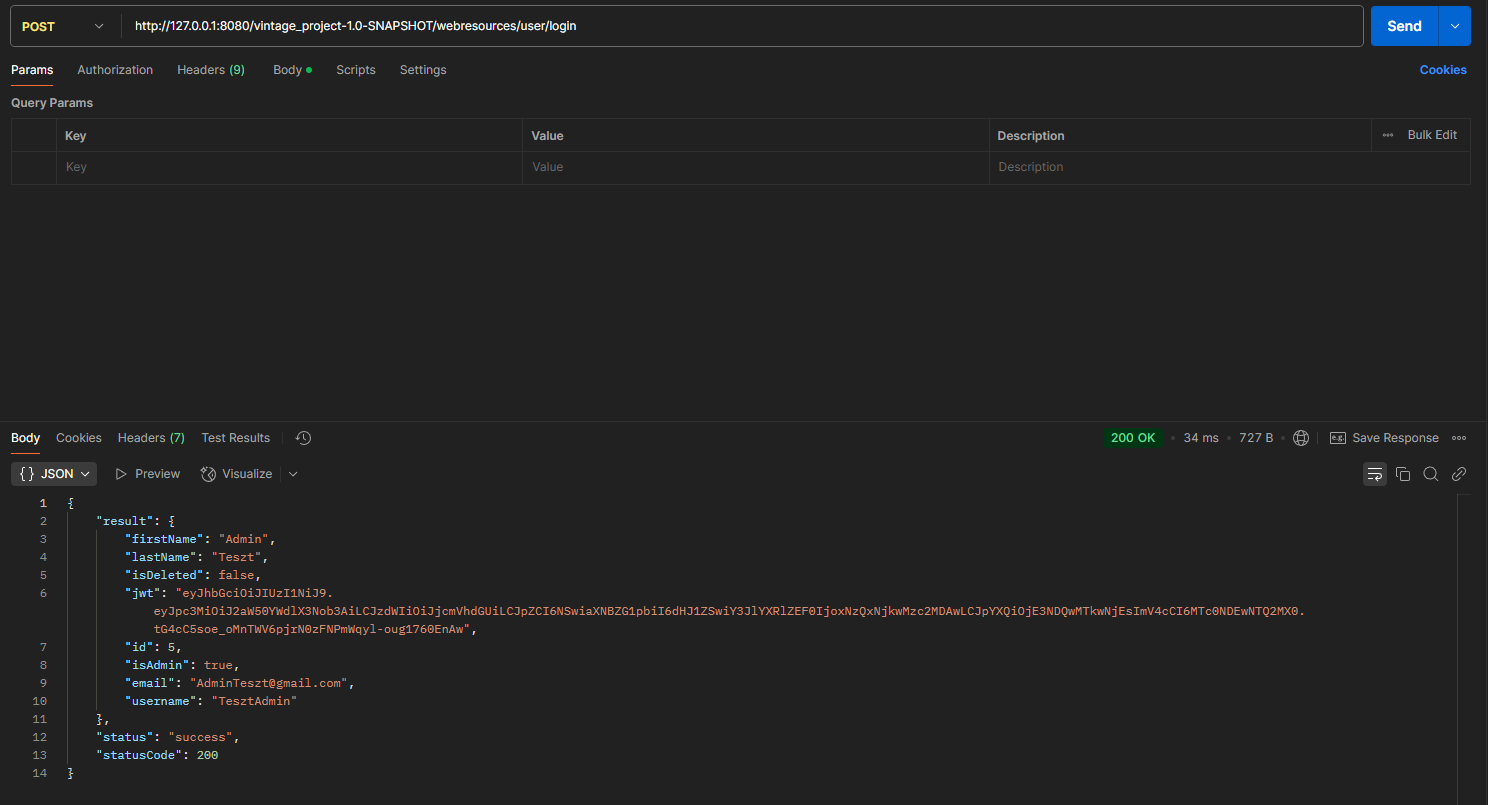
A képen látható kérések tartalmazzák a weboldalunk regisztrációját, bejelentkezését, jelszó változtatás lehetőségét, a profil adatok megváltoztatásának lehetőségét, a regisztráció utáni e-mail küldést. Ezen felül pedig innen lehet lekérni a termékeket is, amik felkerültek az oldalon lévő Shop-ba.

## API dokumentáció:

Az alábbi táblázat és leírás a **Retro Vintage** alkalmazásban használt REST API végpontokat mutatja be, amelyek a frontend és backend közti kommunikációt biztosítják.

|  |  |  |  |
| --- | --- | --- | --- |
| Felhasználói műveletek | | | |
| /user/login | POST | Bejelentkezés email + jelszó alapján | Nyilvános |
| /user/registerUser | GET (!) | Felhasználó regisztráció (paraméterek testben) | Token |
| /user/registerAdmin | POST | Admin felhasználó létrehozása | Admin, Token |
| /user/changePassword | PUT | Jelszó megváltoztatása | Token |
| /user/getUserById | GET | Felhasználó lekérdezése ID alapján | Token |
| /user/update/{modifierId}/update/{targetId} | PUT | Felhasználó adatainak módosítása | Token |
| /user/getAllUser | GET | Összes felhasználó listázása | Token |
| /user/update/{modifierId}/update/{targetId} | PUT | Felhasználó (látszólag logikai) törlése. | Token |
| /user/reactivate-request | POST | E-mail küldése újra aktiváláshoz. | Nyilvános |
| /user/reactivate-from-token?token= | PUT | Felhasználó újra aktiválása token alapján. | Nyilvános |
| /user/reactivate-user/{id} | PUT | Admin felhasználó által történő reaktiválás. | Admin Token |
| /user/reactivatable?email= | GET | Ellenőrzés, hogy újra aktiválható-e az adott email. | Nyilvános |
| /user/forgot-password | POST | Elfelejtett jelszó újra generálása (e-mail alapján). | Nyilvános |
| Termékek kezelése | | | |
| /product/getAllProducts | GET | Összes termék lekérdezése. | Nyilvános |
| /product/getProductById?id={id} | GET | Egy termék részletes lekérdezése ID alapján. | Nyilvános |
| **Kommunikáció és képfeltöltés** | | | |
| /user/send | POST | Sablon alapú e-mail küldése (pl. regisztráció után). | Token |
| /user/{id}/upload-profile-picture | POST | Profilkép feltöltése az adott felhasználóhoz. | Token |
| /user/uploads/{filename} | GET | Profilkép lekérdezése URL alapján. | Nyilvános |

A Postman segítségével a weboldalunk API-végpontjait teszteltük, például a regisztráció, bejelentkezés, jelszómódosítás vagy profiladatok módosításának folyamatait. Az alábbi képen egy sikeres **POST típusú bejelentkezési kérés** látható:



8. ábra: Sikeres bejelentkezés válasza Postman-ben (admin felhasználó)

A válaszból látható, hogy a rendszer egy JSON objektumban adja vissza a bejelentkezett felhasználó adatait, valamint egy JWT tokent, amely a további hitelesített kérésekhez szükséges. Emellett a státusz is success, a státuszkód pedig 200, ami sikeres válaszra utal.

A Postman lehetővé tette, hogy a backend endpointokat még a frontend integráció előtt részletesen teszteljük, így meggyőződhettünk arról, hogy az API megfelelően működik, és időben tudtunk hibát keresni vagy módosítani, ha szükséges volt.

**Fájlkezelés és profilkép feltöltés**

A rendszer támogatja a felhasználói profilképek feltöltését. Ehhez egy dedikált backend végpontot biztosítunk, amely lehetővé teszi képfájlok POST típusú elküldését a szerverre.

**Feltöltés folyamata**

A fájl feltöltéséhez a frontend vagy Postman segítségével egy **multipart/form-data** típusú POST kérést küldünk az alábbi URL-re:

POST /user/{id}/upload-profile-picture

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

9. ábra: Fájl feltöltése Postman-ben multipart/form-data használatával

A kérésben a file kulcs alatt kell megadni a képfájlt. Sikeres feltöltés esetén a szerver a képet elmenti az uploads/ könyvtárba, majd az adott fájlnév alapján a kép URL-ről elérhetővé válik.

Kép elérési út: GET /user/uploads/{filename}

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

10. ábra: Feltöltött profilkép URL-en keresztüli elérése böngészőből

Ezt az URL-t a frontend a profiloldalon jeleníti meg a felhasználóhoz kapcsolódóan.

**Biztonsági megjegyzés:**

A feltöltés csak bejelentkezett felhasználók számára engedélyezett. A végpont védve van token-alapú hitelesítéssel, így csak a megfelelő jogosultsággal rendelkező felhasználók tölthetnek fel képeket.

Tokenkezelés és jogosultság Angularban:

* 1. Jelszó megváltoztatása

PUT /user/changePassword

Lehetővé teszi a felhasználónak a jelszava módosítását. A kérésben meg kell adni az új jelszót, és a backend elvégzi a titkosítást, majd eltárolja az adatbázisban. A funkció token védett.

* 1. Felhasználói adatok módosítása

PUT /user/update/{modifierId}/update/{targetId}

Ez a végpont lehetővé teszi a felhasználói adatok frissítését (pl. név, email). A modifierId az, aki módosítja, a targetId pedig akinek az adatait módosítjuk – ez biztosítja az auditálhatóságot.

* 1. Felhasználó logikai törlése

PUT /user/deleteUser/{id}

A rendszer nem fizikailag törli a felhasználót, hanem egy isDeleted mezőt true-ra állít. Ez lehetővé teszi a visszaállítást, és biztosítja az adatmegőrzést.

* 1. Felhasználók listázása

GET /user/getAllUser

Admin jogosultság szükséges. Az összes regisztrált felhasználó visszaadását végzi (beleértve a törölteket is, ha nincsenek kiszűrve).

**Emailküldés sablon alapján (Backend funkció)**

A rendszer képes automatikus sablonalapú e-mailek küldésére, például a regisztráció vagy újra aktiválás után. Az e-mailküldés funkció REST végponton keresztül valósul meg.

**Végpont:** POST /user/send

A kérés törzsében egy JSON objektumot küldünk, amely tartalmazza:

* a címzett e-mail címét (to),
* a tárgyat (subject),
* a használt sablon nevét (template),
* valamint opcionálisan egy placeholders objektumot, amely dinamikusan behelyettesíthető értékeket tartalmaz (pl. név, link).

**Jogosultság:**

Ez a végpont **token védett**, csak hitelesített felhasználók hívhatják meg. A fejlécben token kulcs alatt kell átadni a JWT-t.

**Backend működés:**

A UserService osztály tartalmazza a logikát, amely sablon alapján generálja a HTML vagy szöveges e-mailt, majd elküldi azt a megadott SMTP szerveren keresztül.

A képen szöveg, képernyőkép, étel látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

11. ábra: Egy template verzió, jelen helyzetben az újra aktiváláshoz

**Tokenkezelés és jogosultság Angularban**

A Retro Vintage alkalmazás a bejelentkezést követően **JWT tokent** kap a szervertől, amelyet a frontend az **Angular oldalon localStorage-ben tárol**. Ez a token szükséges az összes védett végponthoz történő hozzáféréshez, és ezen keresztül történik a felhasználó azonosítása és szerepkörének (pl. admin) kezelése.

**Hol történik a tárolás?**

A token mentése az AuthService vagy a LoginComponent megfelelő metódusában történik:

localStorage.setItem('token', response.token);

A rendszer betöltésekor az Angular ellenőrzi, hogy van-e token, és ha igen, lekérdezi a felhasználói adatokat az API-n keresztül. Ez alapján történik meg:

* a navigációs lehetőségek megjelenítése (pl. admin link csak adminnak),
* a jogosulatlan hozzáférés elutasítása.

**Token ellenőrzés**

A token használatával a frontend az API-kérések fejlécébe automatikusan beilleszti a token mezőt: headers: new HttpHeaders().set('token', localStorage.getItem('token') || '')

A képen szöveg, Betűtípus, szám, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

12. ábra: Tokenkezelés és jogosultság Angularban

# Frontend

El is érkeztünk a Frontendhez, ezt első körben normál Visual Studio Code-ban fejlesztettük le, bármiféle fajta keretrendszer használata nélkül. Majd ezt követően vittük át az Angular keretrendszerbe, és kezdtük el a korábbi kód mintájára alakítani a weboldalt.

Egy pár szó a VS Code-ról:

A **Visual Studio Code (VS Code)** egy ingyenes, könnyű, mégis erőteljes forráskódszerkesztő, amelyet a Microsoft fejlesztett.

* **Mire jó?**
  + Különböző programozási nyelvekben való fejlesztésre (pl. JavaScript, Python, PHP, C++ stb.).
  + Kódszerkesztésre, hibakeresésre és verziókezelésre.
  + Bővítményekkel testre szabható fejlesztői környezet létrehozására.
* **Előnyei:**
  + Ingyenes és platformfüggetlen (Windows, macOS, Linux).
  + Rengeteg bővítmény érhető el hozzá.
  + Beépített terminál és Git-integráció.
  + Intelligens kódkiegészítés (IntelliSense).
* **Hátrányai:**
  + Nagy projektek esetén több memóriát fogyaszthat.
  + Néha lassabb lehet a sok bővítmény miatt.

Főként web- és szoftverfejlesztők használják, mert gyors, rugalmas és könnyen testre szabható.

Ezt követően, essen egy pár szó az Angular keretrendszerről is.

Elsődleges feladatnak tartottuk, hogy ne azonnal Angularban fejlesszünk mert amikor a projektet elkezdtük, még éppen csak láttunk egy keveset az Angular-ból, így jött az alternatíva, hogy kódoljunk normál VS Code-ban. Ennek az alap szerkezetét láthatjuk a következő ábrán:

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

13. ábra: VS Code alapszerkezete

Ebben az alap szerkezetben/vázban létrehoztunk egy kezdetleges felületet, ahol mind az elrendezést, mind a színek párosítását meg tudtuk oldani. Első körben elkészültek a megadott oldalak mappái, majd azon belül mindegyik kapott egy index.html-t és egy css fájlt. Ezek ugye arra szükségesek, hogy meg tudjuk írni a kódot majd pedig képesek legyünk formázni, design-olni őket. Egy két mappa kapott 1-1 javascript fájlt is. Ennek az oka az volt, hogy tartalmaz valamit az oldal, amihez elengedhetetlen volt egy minimális javascript kód megírása.

Ezt követően megkezdődött a tényleges kód írása.

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

14. ábra: A Login page kódolt formája index.html-ben

Itt pedig a kezdetlegesen, de elkészült oldalt láthatjuk.

A képen képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

15. ábra: Az elkészült Login page VS Code-ban

Mivel ez egy teljesen kezdetleges alapnak készült, ezért nem is fektettünk bele hatalmas nagy energiát. Ellenben mondjuk a Home page-el.

Első részben létrehoztunk egy navigációs sávot (nav-bar-t) ami mindegyik oldalon megegyezik.



16. ábra: Navigációs sáv

Ehhez a Bootsrap 5.0-ás verzióját használtuk fel először, majd pedig cseréltük ki a saját magunk által megadott nevek gombjaira.

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

17. ábra: A navigációs sáv kódja

Annak érdekében, hogy a nav-bar úgy nézzen ki és működjön egy ilyen kódot kellett létrehozni.

Ezt követte a body, ami a weboldal központját alkotja, ebben elhelyeztünk carouselt is, kiemeltünk egy pár termékcsaládot, amik a legkellendőbbek, ellenben mindent beégetett képekkel és szöveggel oldottunk meg.

A képen szöveg, Grafikus tervezés, képernyőkép, poszter látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

18. ábra: A Home Body kezdetleges kinézete

Az itt lévő Carouselekhez kellett a js fájl ugyanis a termékek a megadott időkorláton túl automatikusan váltják magukat. Ez mindhárom kártyán 5 másodperc, emellett pedig 5-5 termék váltja egymást.

A képen szöveg, képernyőkép, képernyő, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

19. ábra: A Carouselek automatikus váltása

Ezeken felül, minden oldal kapott egy Footert, ahol egy pár szóban a készítőkről és az információkról, kapcsolattartásról stb.

A képen szöveg, képernyőkép, szoftver, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

20. ábra: A Footer kialakítása, elrendezése

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

21. ábra: A Footer kódolása

# Angular

A **Angular** egy népszerű, nyílt forráskódú JavaScript-alapú keretrendszer, amelyet a Google fejlesztett.

* **Mire jó?**
  + Nagy és összetett webalkalmazások fejlesztésére.
  + Egyoldalas alkalmazások (**SPA - Single Page Application**) készítésére.
  + Dinamikus, reszponzív és moduláris weboldalak létrehozására.
* **Előnyei:**
  + Strukturált és jól szervezett fejlesztési környezet.
  + Kétirányú adatkapcsolat (**two-way data binding**), amely megkönnyíti az adatok

kezelését.

* + Beépített eszközök és funkcionalitások (pl. Dependency Injection, Routing).
  + Nagy közösségi támogatás és hosszú távú Google-támogatás.
* **Hátrányai:**
  + Meredek tanulási görbe a komplexitása miatt.
  + Nagyobb teljesítményigényű, mint más könnyebb keretrendszerek (pl. React, Vue).

Az Angular főként nagyobb vállalati szintű projektekhez ideális, ahol a skálázhatóság és a robusztus szerkezet kiemelten fontos.

## Angular projektstruktúra bemutatása

Az Angular projektünk logikus és áttekinthető mappastruktúrával épül fel, amely segíti a fejlesztést, tesztelést és karbantartást. A mappaelrendezés követi az Angular ajánlott struktúráját, néhány testreszabott elnevezéssel és gyakorlatias csoportosítással.

## Főbb mappák:

**src/app/**

Ez a projekt fő munkaterülete, itt találhatók az alkalmazás komponensei, szolgáltatásai és útvonalai.

**src/app/\_component/**

A komponensek strukturáltan, egy mappába rendezve helyezkednek el. Minden komponens saját alkönyvtárat kapott, amely tartalmazza a .ts, .html és .css fájlokat. Ez a felépítés elősegíti a könnyebb olvashatóságot és karbantartást.

A képen szöveg, képernyőkép, szoftver, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

22. ábra: A mappaszerkezet és projektstruktúra

**src/app/services/**

A különféle Angular szolgáltatások (service-ek) itt találhatók. Ezek biztosítják az adatok továbbítását a komponensek és a backend API között.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

23. ábra: A services mappa tartalma

**src/app/models/**

Ebben a mappában tároljuk az adatmodelleket, például a User, Product, vagy Order interfészeket. Ezek definiálják, milyen típusú adatokkal dolgozik az alkalmazás.

A képen szöveg, Betűtípus, képernyőkép, Grafika látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

24. ábra: Az adatmodellek

**src/app/app-routing.module.ts**

Az útvonalkezelésért felelős fájl, ahol beállítottuk az oldalhoz tartozó komponenseket és azok URL-címeit. Itt történik az Angular Router konfigurálása.

**src/app/app.module.ts**

Az Angular alkalmazás főmodulja, amelyben minden szükséges komponenst, szolgáltatást és modult regisztrálunk.

Ez a struktúra lehetővé teszi, hogy bármely fejlesztő gyorsan átlássa az alkalmazás logikáját, és könnyen bekapcsolódhasson a munkába.

## 3. Rövid bemutatása a főbb komponenseknek:

1. LoginComponent – Bejelentkezés kezelése

Ez a komponens felelős a felhasználók bejelentkeztetéséért. A felhasználó megadja az e-mail címét és jelszavát, majd a rendszer validálja az adatokat, és POST kérést küld a backend felé.

Főbb fájlok:

login.component.ts: kezeli az űrlap működését, validációt, és API-hívást.

login.component.html: a bejelentkezési űrlap HTML szerkezete.

login.component.css: az oldal megjelenésének formázása.

Működés:

Form validáció: kötelező mezők, jelszóhossz, e-mail formátum.

Hibaüzenet megjelenítés: pl. "Hibás felhasználónév vagy jelszó".

Sikeres bejelentkezés esetén token mentése (pl. localStorage), majd átirányítás pl. a profile oldalra.

2. ProfileComponent – Felhasználói adatok kezelése

A ProfileComponent lehetővé teszi a felhasználó számára, hogy megtekintse és módosítsa a regisztrációkor megadott adatait, valamint megváltoztassa a jelszavát.

Főbb fájlok:

profile.component.ts: adatok lekérése, mentése, validáció.

profile.component.html: form a név, email, cím stb. módosításához.

profile.component.css: stílus és reszponzív elrendezés.

Funkciók:

Adatok betöltése az aktuális felhasználóhoz (pl. GET /user/:id).

Jelszómódosító szekció, amely az előre meghatározott jelszókövetelményeket ellenőrzi.

Adatok mentése PUT kérésen keresztül.

Sikeres mentés után értesítő üzenet (pl. "A módosítások elmentve").

3. AdminDashboardComponent – Admin funkciók és statisztikák

Az admin felület egy összetettebb komponens, amely külön jogosultsággal érhető el. Ez a komponens ad hozzáférést a weboldal üzemeltetéséhez szükséges funkciókhoz.

Főbb fájlok:

admin-dashboard.component.ts: API-hívások kezelése, logika.

admin-dashboard.component.html: statisztikák, táblázatok, gombok.

admin-dashboard.component.css: reszponzív, admin-panel stílus.

Fő funkciók:

Regisztrációk száma, statisztikák megjelenítése (pl. napi/heti/havi bontásban).

Felhasználók listázása, törlése vagy szerkesztése.

Termékek kezelése: új termék hozzáadása, meglévők törlése vagy módosítása.

Jogosultság ellenőrzés: csak admin felhasználó férhet hozzá (általában auth service + guard által védve).

## 4. Űrlapkezelés és validáció Angularban

**LoginComponent – űrlap validáció**

* Kötelező mezők: e-mail és jelszó
* E-mail mezőnél type="email" és mintaellenőrzés is használható
* Jelszónál minimális hosszúság (pl. 8 karakter)

**Hibakezelés:**

* Ha bármelyik mező hibás, a "Bejelentkezés" gomb le van tiltva
* Hiba esetén: „Hibás e-mail vagy jelszó”
* Sikeres bejelentkezéskor: átirányítás a profile oldalra

**RegisterComponent – jelszókövetelmények validálása**

* Jelszónak meg kell felelnie a dokumentumban is leírt szabályoknak:
  + Legalább 1 kisbetű
  + Legalább 1 nagybetű
  + Legalább 1 szám
  + Legalább 1 speciális karakter
  + Legalább 8 karakter hosszúság

**Hibaüzenet:**

* „A jelszónak legalább 8 karakter hosszúnak kell lennie, tartalmaznia kell kis- és nagybetűt, számot és speciális karaktert.”

**ProfileComponent – adatfrissítés + jelszómódosítás**

* A profiladatok módosítása űrlapon történik: név, cím, e-mail stb.
* Jelszómódosítás egy külön mezőpárral:
  + régi jelszó
  + új jelszó
  + új jelszó megerősítése

**Validáció:**

* Az új jelszó és a megerősítés egyezését ellenőrzi a rendszer
* A rendszer nem engedi a mentést, ha nem megfelelő a formátum vagy nem egyeznek a jelszavak

**Sikeres beküldés után:**

* A submit() függvény POST vagy PUT kérést küld a backend felé
* A válasz alapján értesítjük a felhasználót:
  + „Sikeres módosítás”
  + vagy „Hiba történt a mentés során”

## 5. Állapotkezelés Angularban

Az alkalmazás nem használ különálló állapotkezelő könyvtárat (pl. NgRx, Akita), de **alapszintű állapotkezelést a komponensek és szolgáltatások szintjén** megvalósítottunk.

**Bejelentkezett felhasználó kezelése**

* A felhasználói adatok (pl. token, email, szerepkör) a bejelentkezés után elmentésre kerülnek a böngésző localStorage-ébe.
* Ez lehetővé teszi, hogy a felhasználó kilépés nélkül oldalváltások során is bejelentkezve maradjon.

**Navigáció jogosultság alapján**

* Egyes oldalak (pl. admin) csak akkor érhetők el, ha a token alapján azonosítjuk a felhasználót.

## **Reszponzivitás és felhasználói élmény (UX)**

A webalkalmazás kialakításánál kiemelten fontos szempont volt, hogy az oldal minden eszközön – legyen az **asztali számítógép, tablet vagy mobiltelefon** – átlátható és könnyen kezelhető maradjon.

**Alkalmazott technikák:**

* Az oldal elrendezéséhez **Flexbox** és **CSS Grid** technikákat használtunk.
* A kezdeti fázisban **Bootstrap 5** keretrendszerre építettünk, majd azt fokozatosan leváltottuk saját stílusokra.
* A komponenseket úgy alakítottuk ki, hogy **automatikusan alkalmazkodjanak a képernyő méretéhez** (pl. oszlopok egymás alá kerülnek mobilon).

**Tesztelési módszerek:**

* A reszponzív működést **Chrome Developer Tools** segítségével teszteltük, különféle képernyőméretek és eszközprofilok használatával (pl. iPhone X, iPad).
* Ezen kívül használtuk a **Mobil Szimulátor – Reszponzív tesztelés** nevű Chrome-bővítményt ([link](https://chromewebstore.google.com/detail/mobil-szimul%C3%A1tor-reszponz/ckejmhbmlajgoklhgbapkiccekfoccmk)), amely lehetővé tette, hogy az oldalt több különböző mobilnézetben is valósághűen ellenőrizzük.

**UX szempontok:**

* A navigációs sáv minden nézeten elérhető, mobilon lenyíló menüvé alakul.
* A legfontosabb funkciók (pl. Bejelentkezés, Kosár, Profil) kiemelt helyen jelennek meg.
* A betűméret, kontraszt és vizuális hierarchia úgy lett kialakítva, hogy az oldal olvasható és értelmezhető legyen akár kis kijelzőn is.

A fentiek biztosítják, hogy az oldal **ergonomikus, átlátható és könnyen használható legyen minden eszközön**, ami ma már alapkövetelmény minden modern webalkalmazásnál.

## Admin felület – működés és funkciók

Az adminisztrációs felületet kizárólag admin jogosultságú felhasználók érhetik el. Célja, hogy az oldal üzemeltetője teljes kontrollt gyakorolhasson a felhasználók, termékek és statisztikák felett.

**Elérhetőség és jogosultság**

* Az admin oldalhoz való hozzáférést AuthGuard vagy komponens szintű jogosultság-ellenőrzés védi.
* A bejelentkezés után a felhasználó szerepkörét (pl. admin) az Angular a token vagy az API-válasz alapján ellenőrzi.
* Jogosulatlan elérés esetén a felhasználó visszairányításra kerül az alap oldalra.

**Fő funkciók**

**Statisztikák megjelenítése**

* Az oldal betöltése után a rendszer lekéri az aktuális felhasználói adatokat (pl. regisztrációk száma, aktív felhasználók, rendelések).
* Ezek grafikon vagy egyszerű számláló formájában jelennek meg.
* A megjelenített statisztikák segítik az admin döntéshozatalát és az oldal forgalmának követését.

**Felhasználók kezelése**

* Az admin megtekintheti az összes regisztrált felhasználót.
* Lehetőség van:
  + Felhasználó törlésére
  + Felhasználói adatok módosítására
  + Szerepkörök (admin/felhasználó) kezelésére

**Termékek kezelése**

* A meglévő termékeket szerkesztheti vagy törölheti.
* Új termékeket tölthet fel:
  + Név, leírás, ár, kategória, kép
* Feltöltés után a termékek azonnal megjelennek a webshop felületén.

**Technikai háttér**

* Minden admin funkció egy-egy dedikált komponensben valósul meg: pl. AdminDashboardComponent, UserListComponent, ProductEditorComponent.
* Az API-kommunikációt service-ek végzik, pl. admin.service.ts.
* A formokat Angular validációval védtük (kötelező mezők, karakterkorlátok).

# ChatGPT/Mesterséges Intelligencia

A mesterséges intelligencia használata, előnyei, hátrányai:

* Használata:
  + **Írás**: Esszék, e-mailek, novellák, versek, hivatalos dokumentumok írására vagy szerkesztésére.
  + **Programozás**: Kódírás, hibakeresés, algoritmusok megértése.
  + **Tanulás**: Fogalmak elmagyarázása, vizsgafelkészülés, jegyzetelés.
  + **Kreatív ötletelés**: Karakternevek, sztorik, kampányötletek, játékötletek.
  + **Nyelvi segítség**: Fordítás, nyelvtani magyarázat, szókincsfejlesztés.
  + **Mindennapi kérdések**: Tippek életmódhoz, utazáshoz, receptek, ajándékötletek.
  + **Szórakozás**: Viccek, találós kérdések, játékötletek, akár szerepjáték is.
* Előnyei:
  + **Gyors és rendelkezésre állok**: Napi 24 órában elérhető vagyok, nincs alvás vagy ebédszünet.
  + **Sokféle témában tudok segíteni**: Legyen szó írástechnikáról, kódolásról, receptötletekről vagy akár érvelésről egy esszéhez – sok mindenhez értek.
  + **Nyelvtudás**: Több nyelven is kommunikálok, köztük magyarul is elég jól.
  + **Nem ítélkezem**: Bármit kérdezhetsz, nem foglak elítélni vagy kinevetni.
  + **Kreatív vagyok**: Tudok verset írni, mesét kitalálni, vicceket mondani, vagy akár neveket adni karaktereidnek.
  + **Rugalmas stílusban írok**: Lehetek formális, laza, humoros, érzelmes – ahogy szeretnéd.
* Hátrányai:
  + **Nem vagyok ember**: Hiányzik az emberi tapasztalat, empátia mélysége, vagy az igazi "megérzés".
  + **Nem tévedhetetlen**: Néha pontatlan vagy elavult információt adhatok főleg, ha friss hírekről van szó.
  + **Nem vagyok tudatában semminek**: Nem emlékszem rád automatikusan, hacsak te nem engedélyezed a memóriát a beállításokban.
  + **Nem tudok hozzáférni mindenhez**: Nem látok rá privát adatbázisokra, zárt weboldalakra, vagy titkos információkra.
  + **Néha túl „okoskodó” vagy „általános”**: Egyes válaszaim túl általánosak lehetnek, ha nem kapok elég konkrét kontextust.

A mesterséges intelligenciát, avagy a konkrétan a ChatGPT-t főként a problémáink megoldásához, avagy az újabb ismeretek gyűjtéséhez használtuk. Ezen felül segítségünkre volt az adatbázisunk feltöltésében is ahol termékeket generált számunkra, majd a későbbiekben a termékekhez képet is generáltunk vele.

Mint ahogy a képen is látható, a mi általunk megadott prompt-ok alapján generált számunkra megfelelő termékeket:

A képen szöveg, elektronika, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

25. ábra: Kérésünkre kért termékek generálása

Majd miután minden terméket legenerált, kértük, hogy készítsen hozzájuk képeket is annak érdekében, hogy a termékeknek legyen látszata is, ne pedig csak egy rövid leírást kapjanak a „vevők”.

A képen óra, karóra, Zsebóra látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

26. ábra: A megadott termékekhez generált képek

Ezeken felül pedig az használt programok előnyeit, hátrányait és azt, hogy mire használják őket az emberek, azt is megírattuk vele. Annak érdekében, hogy minél pontosabb információt kapjunk.

# Záró gondolatok

A Retro Vintage webalkalmazás elkészítése során nemcsak egy működőképes, vizuálisan esztétikus és technikailag megalapozott webshopot hoztunk létre, hanem valós projektmunkán keresztül mélyítettük el tudásunkat a szoftverfejlesztés különböző területein.

A fejlesztés során komplexebb technológiákat is megismertünk és alkalmaztunk, mint például:

* **MySQL** adatbázis tervezés és SQL-alapú logika (tárolt eljárások)
* **Java + Maven** alapú backend fejlesztés WildFly szerveren
* **Angular keretrendszer** használata a modern frontend fejlesztéshez
* **REST API-k kommunikációjának tesztelése Postman segítségével**
* **Mesterséges intelligencia** gyakorlati alkalmazása adatgenerálásra és képkészítésre

A munka során megtapasztaltuk az önálló és csapatban történő fejlesztés kihívásait is: a tervezéstől a hibakeresésen át az éles működésig. A problémamegoldó képességünk és technikai tudásunk egyaránt sokat fejlődött, különösen az adatok kezelése, a jogosultságkezelés és a komponensalapú gondolkodás területén.

Ez a projekt nemcsak a szakmai vizsga teljesítéséhez járult hozzá, hanem erős alapot adott jövőbeli fejlesztési munkákhoz is.

# Források

A projekt fejlesztése során több forrásból származó technológiai, biztonsági és jogi előírást, dokumentációt és ajánlást vettünk figyelembe. Az alábbi lista tartalmazza azokat a legfontosabb forrásokat, amelyek a rendszer működését, biztonságát és jogszerűségét alapozzák meg vagy segítik elő.

**Adatbiztonság és jelszókezelés**

* OWASP Password Storage Cheat Sheet:  
  <https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html>
* OWASP Authentication Cheat Sheet:  
  <https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html>
* Jelszókövetelmények (kisbetű, nagybetű, szám, speciális karakter) kialakítása OWASP ajánlás alapján történt.

**Adatvédelem (GDPR)**

* Az Európai Unió általános adatvédelmi rendelete (GDPR):  
  <https://eur-lex.europa.eu/legal-content/HU/TXT/?uri=CELEX%3A32016R0679>
* Adatkezelési logika során figyelembe vettük a minimális adatgyűjtés és a célhoz kötöttség elvét.

**REST API fejlesztés és tesztelés**

* REST API konvenciók és ajánlások:  
  <https://restfulapi.net>
* Postman dokumentáció és API tesztelés:  
  <https://learning.postman.com/docs/getting-started/introduction/>

**Technológiák hivatalos dokumentációja**

* Java 17 (LTS):  
  <https://docs.oracle.com/en/java/javase/17>
* Maven:  
  <https://maven.apache.org/guides/index.html>
* WildFly:  
  <https://docs.wildfly.org>
* Angular:  
  <https://angular.io/docs>
* Node.js:  
  <https://nodejs.org/en/docs>
* MySQL:  
  <https://dev.mysql.com/doc/>
* phpMyAdmin:  
  <https://docs.phpmyadmin.net>
* Bootstrap (v5):  
  <https://getbootstrap.com/docs/5.0/getting-started/introduction/>