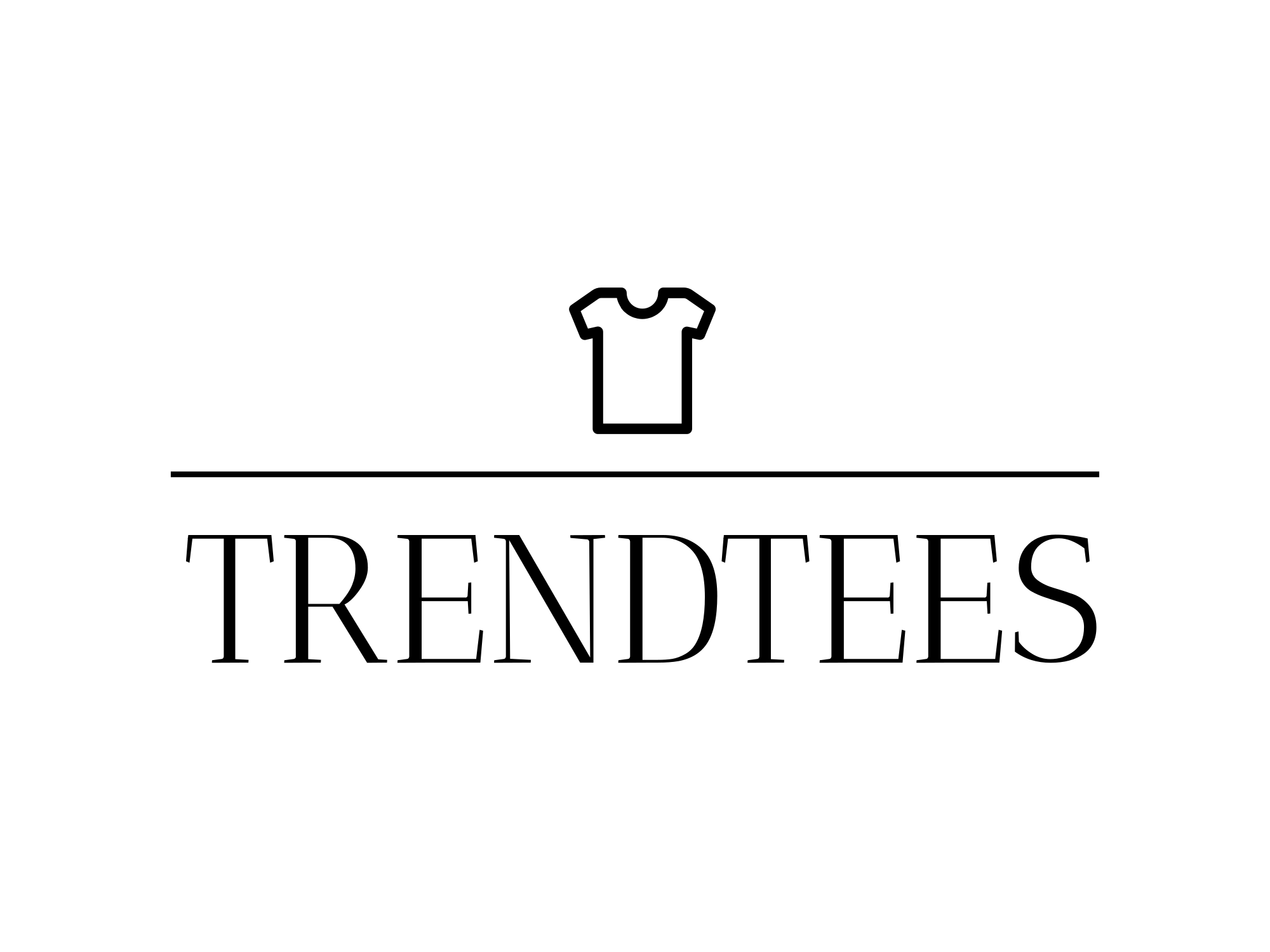
******

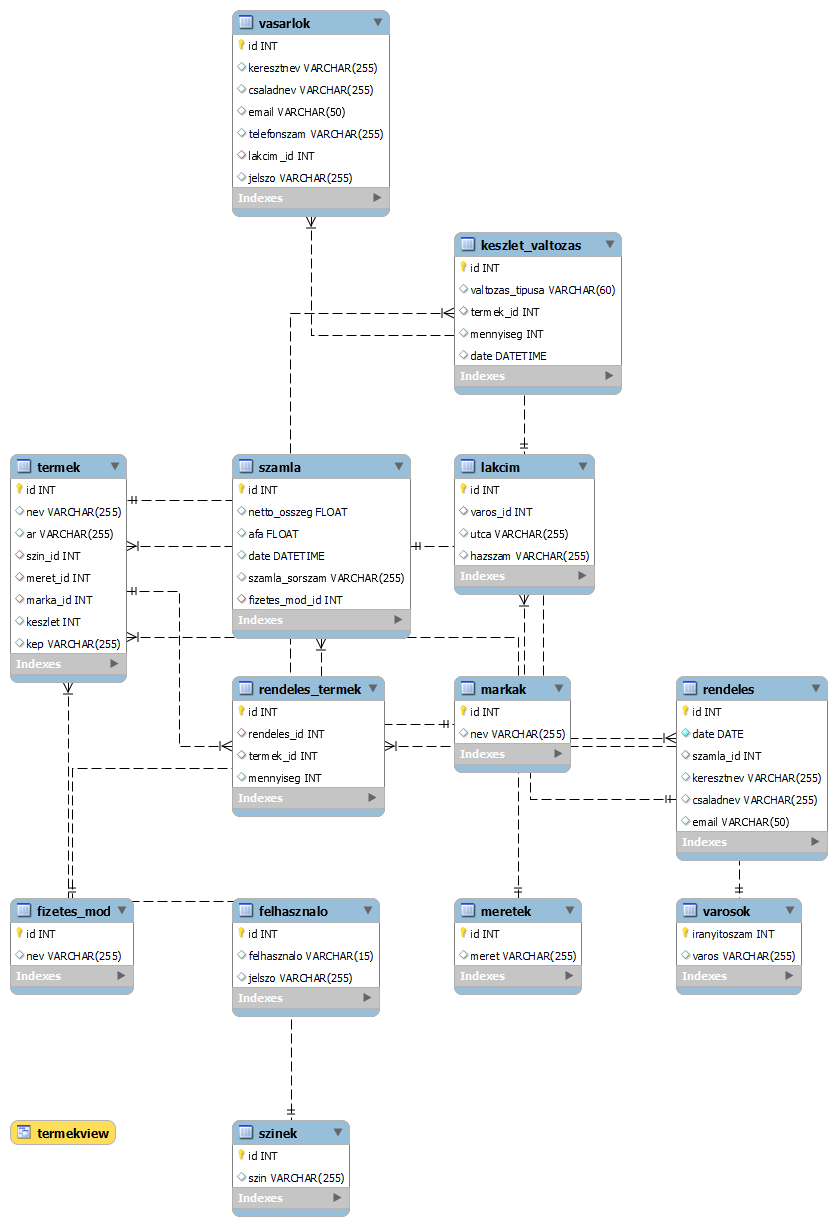
# **TrendTees Webshop záródolgozat**

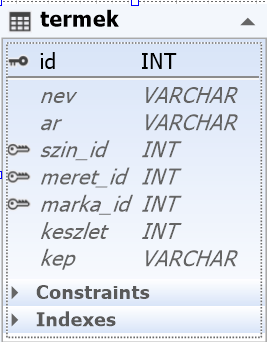
A záródolgozatot készítette:  
  
Kőműves Zoltán és Molnár Zsombor

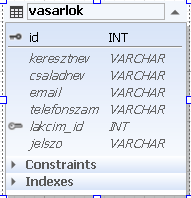
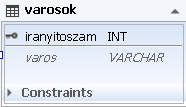
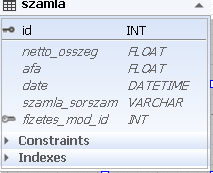
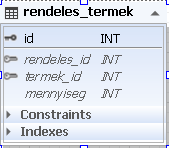
***1.Témaválasztás:***

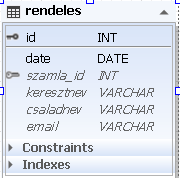
A **TrendTees** egy saját fejlesztésű póló webshop, ahol különböző márkák pólóit kínáljuk megvásárlásra. Számunkra egyértelmű volt, hogy egy ilyen webshopot hozunk létre, mivel mindketten rajongunk a pólókért.

***2.Rendszerleírás:***  
Ennek a webshopnak a célja, az adott készletünk kiárusítása. Ami letisztult felhasználói felülettel rendelkezik, amely lehetővé teszi a könnyű böngészést és a kívánt pólók gyors megtalálását. A felhasználók egyszerűen navigálhatnak márkák , méretek vagy színek alapján, hogy gyorsan és hatékonyan megtalálják a keresett terméket.

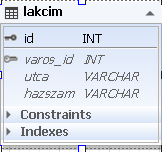
1. Architektúra és tervezés:  
   ***Adatbázis és a kapcsolati ábra:*** 

***A fő tábla a termek:***  
id: a póló azonosítója  
nev: a póló neve  
ar: a póló ára  
szin\_id: a szín azonosítója  
meret\_id: a méret azonosítója  
marka\_id: a márka azonosítója  
keszlet: az adott pólóból mennyi van készleten  
kep: a pólók képei  
  
 ***A szinek tábla:***id***:*** a színek azonosítója  
  
szin: a szín neve  
  
  
  
***A meretek tábla:***id: a méretek azonosítója  
  
meret: a méret neve

***A vásárlók tábla:***id : a vásárló azonosítója  
  
keresztnev: a vásárló keresztneve  
  
csaladnev: a vásárló családneve  
  
email: a vásárló email címe  
  
telefonszam: a vásárló telefonszáma  
  
lakcim\_id: a lakcím azonosítója  
  
jelszo: a vásárló jelszava  
  
  
******  
***A varosok tábla:***iranyitoszam:az iranyitoszam azonosítója  
varos: a városok nevei  
  
  
***A szamla tábla:***id: a számla azonosítója  
nettó\_osszeg: mennyibe került a rendeles  
afa: a számlához tartozó áfa  
date: dátum  
szamla\_sorszam: a számla sorszáma  
fizetes\_mod\_id: a fizetési mód azonosítója  
  
  
***A rendeles\_termek tábla:***id: a rendelés termék azonosítója  
rendeles\_id: a rendelés azonosítója  
termek\_id: a termék azonosítója   
mennyiseg: a darabszám az adott termékről

***A rendeles tábla:***id: a rendelés azonosítója   
  
date: a rendelés dátuma  
  
szamla\_id: a számla azonosítója  
  
keresztnev: a vásárló keresztneve  
  
csaladnev: a vásárló családneve  
  
email: a vásárló email címe

******  
  
***A markak tábla:***id: a márka azonosítója  
  
nev: a márka neve

******

***A lakcim tábla:***id: a lakcím azonosítója  
  
varos\_id: a város azonosítója  
  
utca: az adott utca  
  
hazszam: a adott házszám   
  
  
  
  
**A keszlet\_valtozas tábla:**id: a készlet változás azonosítója  
  
valtozas\_tipusa: kiadás vagy bevétel  
  
termek\_id: a termék azonosítója  
  
mennyiseg: hogy hány darab  
  
date: dátum

***A fizetesi\_mod tábla:***id: a fizetési mód azonosítója  
  
nev: fizetésimód neve (bankártya, készpénz stb)  
  
  
  
  
***A felhasznalo tabla:***id: a felhasználó azonosítója  
felhasznalo: jogosultság neve pl: (admin)  
jelszo: az adott felhasználóhoz tartozó jelszó

# Backend(Szerveroldal)

***1.Node.js***

Jellemzők**:** Egy modern, minimalista és rendkívül rugalmas webalkalmazás-keretrendszer Node.js-hez, amely gyors és hatékony webszerverek létrehozását teszi lehetővé. Különböző hasznos funkciókat és middleware-eket kínál, például útvonalkezelést, HTTP kérés- és válaszobjektumok feldolgozását, sablonmotorok integrálását és számos egyéb praktikus eszközt.

**Felhasznált külső modulok:**

**Függőségek és "dependencies":**

**1.axios**: "^1.7.9",  
Egy népszerű HTTP kliens könyvtár, amely segíti az aszinkron HTTP kérések küldését és fogadását (GET, POST, stb.) Node.js-ben és böngészőkben.

**2.cookie-parser**: "~1.4.4",  
Middleware a Node.js és Express számára, amely segít a bejövő http-kérések sütijeinek (cookie-k) elemzésében és kezelésében.

**3.cors**: "^2.8.5",  
Egy middleware, amely segít az engedélyezett források közötti erőforrás-megosztást (CORS) konfigurálni Express alapú alkalmazásokban.

**4.debug**: "~2.6.9",  
Egy kis méretű hibakereső eszköz, amely lehetővé teszi a fejlesztők számára, hogy könnyebben nyomon kövessék és naplózzák a futó alkalmazások eseményeit.

**5.dotenv**: "^16.4.7",  
Egy csomag, amely segít a környezeti változók (.env fájl) kezelésében és betöltésében, így az érzékeny adatok (pl. API kulcsok) nincsenek közvetlenül a kódban.

**6.express**: "~4.16.1",  
Az egyik legnépszerűbb Node.js webkeretrendszer, amely segíti a REST API-k és webalkalmazások gyors fejlesztését.

**7.jsonwebtoken**: "^9.0.2",  
Egy csomag, amely a JSON Web Token-ek (JWT) kezelésére szolgál, lehetővé téve a biztonságos hitelesítést és jogosultságkezelést.

**8.morgan**: "~1.9.1",  
Egy HTTP kérések naplózására szolgáló middleware, amely segít a szerverforgalom elemzésében és hibakeresésben.

**9.mysql2**: "^3.12.0",  
Egy MySQL adatbázis-kezelő kliens Node.js számára, amely gyorsabb és modernebb alternatívája az eredeti mysql csomagnak.

**10.nodemon**: "^3.1.9"  
Egy fejlesztői eszköz, amely figyeli a fájlváltozásokat és automatikusan újraindítja a Node.js alkalmazást, így megkönnyíti a fejlesztési folyamatot.

Könyvtárstruktúra:

**App.js:**   
Ez egy **Node.js + Express backend alkalmazás**, amely beállítja az alapvető middleware-eket (CORS, logger, JSON kezelés, cookie-parsing, statikus fájlok), és három útvonalat (/, /users, /termekek) kezel külön routerekkel. Betölti a .env fájlból a **JWT titkos kulcsot** és biztosítja, hogy a frontend (http://localhost:5173) elérhesse az API-t.

*Routes/:* Az útvonal kezeléséért felelős fájlok  
*index.js:* Ez egy Express útvonalkezelő modul, amely egy alapértelmezett GET végpontot (/) hoz létre, és egy üres választ küld vissza

termek.js: Ez az **Express útvonalkezelő** a termékekkel és kapcsolódó műveletekkel foglalkozik. Lehetővé teszi:

* **Terméklista lekérdezését** különböző paraméterekkel (szűrők, lapozás),
* **Egy adott termék lekérdezését** azonosító alapján,
* **Termék törlését** azonosító alapján,
* **Termék adatainak módosítását** (pl. márka, ár),
* Valamint tartalmaz **fizetéssel és kosárba helyezéssel** kapcsolatos végpontokat is (pl. fizetési mód lekérdezése, rendelés hozzáadása a kosárhoz).

A kódban a Db modul hívásával végez adatbázis-műveleteket, és hiba esetén megfelelő HTTP-válaszokat küld a kliensnek.

users.js: Ez az Express útvonalkezelő modul a felhasználók regisztrációját, bejelentkezését, fiók törlését és jelszó módosítását valósítja meg:

* **Regisztráció (/register)**  
  Az új felhasználó adatait fogadja (email, jelszó, keresztnév, vezetéknév, telefonszám), ellenőrzi, hogy minden mező meg van-e adva, majd beszúrja az adatbázisba. Sikeres regisztráció esetén JWT tokennel tér vissza, amely tartalmazza a felhasználó azonosítóját és email címét.
* **Bejelentkezés (/login)**  
  Az email és jelszó alapján keres a felhasználók között, ellenőrzi a jelszó helyességét, majd sikeres hitelesítés esetén szintén JWT tokennel és felhasználói adatokkal válaszol.
* **Token alapú hitelesítés**  
  Az authenticateToken middleware a HTTP kérés authorization fejlécében lévő token alapján ellenőrzi a felhasználó hitelességét, és érvényes token esetén engedélyezi a további műveleteket.
* Fiók törlése (/account DELETE)  
  Az előzőleg hitelesített felhasználó azonosítója alapján történik a törlés, melyet az adatbázisban hajt végre.
* **Jelszó módosítása (/account/password PUT)**  
  A felhasználó megadja a jelenlegi és az új jelszavát. A rendszer dekódolja a tokenből az email címet, ellenőrzi a jelenlegi jelszó helyességét, majd ha minden rendben van, frissíti az új jelszót az adatbázisban.  
  *(Megjegyzés: a kódban két azonos útvonaldefiníció szerepel, ami redundáns, így csak egy példány szükséges.)*

Ez a modul tehát a felhasználói hitelesítés alapjait, a regisztrációt, bejelentkezést, fiók törlését és jelszóváltoztatást valósítja meg, biztonságos token alapú hitelesítéssel.

*vasarlok.js:* Ez az Express útvonalkezelő modul a vásárlók (vasárlók) kezelését végzi. A fő funkciók a következők:

* **Összes vásárló lekérdezése:**  
  Egy GET kérésre lekéri az összes vásárlót az adatbázisból, és JSON formátumban adja vissza.
* **Szűrt lista készítése:**  
  A /filter útvonalon keresztül a lekérdezésben megadott "marka" alapján keres a vásárlók között, részleges egyezés esetén visszaadva a találatokat.
* **Egy vásárló lekérdezése ID alapján:**  
  Az útvonal paraméterként kapott azonosító (azonosito) alapján megkeresi az adott vásárlót. Ha nincs ilyen vásárló, 404-es hibát küld.
* **Új vásárló hozzáadása:**  
  A POST kérés során a kliens által küldött vásárló adatok alapján beszúr egy új rekordot az adatbázisba. Itt ellenőrzi, hogy a szükséges mezők (pl. marka, email) meg legyenek adva.
* **Vásárló adatainak módosítása:**  
  A PUT kérés segítségével, az URL-ben megadott ID alapján frissíti a vásárló adatait, ha a kötelező mezők (marka, email) megvannak.

contact.js:

Ez az Express útvonalkezelő modul egy POST végpontot definiál, amelynek célja, hogy a felhasználó által küldött űrlapadatokból (név, vezetéknév, email, telefon, üzenet) összeállítson egy e-mailt, majd az emailService modul segítségével elküldje azt a megadott címre ([trendteesshop12@gmail.com](mailto:trendteesshop12@gmail.com)).

A kód működése a következő:

* A kliens POST kérést küld az űrlap adataival.
* A szerver a request body-ból kinyeri a szükséges mezőket, majd egy formázott szövegben összeállítja az e-mail tartalmát.
* Az emailService.sendEmail metódust hívja meg az e-mail elküldéséhez, ahol meghatározza a címzettet, tárgyat és a szöveges tartalmat.
* Sikeres küldés esetén a szerver 200-as státusszal, hiba esetén pedig 500-as hibakóddal válaszol.

Middleware:

Middleware-ek az Express.js keretrendszerben segítenek az HTTP kérések feldolgozásában és a különböző feladatok elvégzésében a kérés és a válasz között. A middleware-ek feldolgozzák a kéréseket a beérkezési sorrendben, és módosíthatják azokat vagy adhatnak hozzájuk információt.

* verifyToken: Ez a middleware ellenőrzi a bejelentkezett felhasználó token-jének érvényességét. Ha a token érvényes, akkor továbbengedi az útvonalra, különben hibát dob.

API végpontok:

Nyilvános végpontok:

 **GET /** - Termékek lekérdezése szűrési feltételekkel

 **GET /payment** - Fizetési mód lekérdezése név alapján

 **DELETE /account** – Fiók törlése ID alapján

 **PUT /account/password** – Jelszó megváltoztatása

 **GET /payment** - Fizetési mód lekérdezése név alapján

 **POST /register** – Felhasználó adatainak beszúrása regisztráció után

 **POST /login** – Bejelentkezés e-mail alapján

 **POST /addtocart** – Számla létrehozása

## **📌 API Végpontok Áttekintése**

### 🔹 **Regisztráció (**POST /register**)**

* **Célja:** Új felhasználó létrehozása az adatbázisban.
* **Adatok:** email, password, firstName, lastName, phone
* **Működés:**
  1. Ellenőrzi, hogy minden adat meg van-e adva.
  2. Beilleszti az új felhasználót az adatbázisba.
  3. Generál egy JWT tokent.
  4. Visszaküldi a felhasználói adatokat és a tokent.

### ***🔹 Bejelentkezés (POST /login)***

* **Célja:** Meglévő felhasználó bejelentkeztetése.
* **Adatok:** email, password
* **Működés:**
  1. Kikeresi az adatbázisból az adott email címhez tartozó felhasználót.
  2. Ellenőrzi a megadott jelszó helyességét.
  3. Generál egy JWT tokent és visszaküldi a felhasználónak.

### ***🔹 Felhasználói fiók törlése (DELETE /account)***

* **Célja:** Bejelentkezett felhasználó törlése az adatbázisból.
* **Hitelesítés:** **JWT token szükséges**
* **Működés:**
  1. **A authenticateToken middleware** ellenőrzi, hogy a felhasználó rendelkezik-e érvényes tokennel.
  2. Lekérdezi a tokenből az azonosítót (req.user.id).
  3. Kitörli az adatbázisból a felhasználót.

### ***🔹 Jelszó módosítása (PUT /account/password)***

* **Célja:** Felhasználó jelszavának frissítése.
* **Adatok:** currentPassword, newPassword
* **Működés:**
  1. Kinyeri a JWT tokenből a felhasználó email címét.
  2. Ellenőrzi, hogy a megadott **jelenlegi jelszó helyes-e**.
  3. Ha igen, frissíti a jelszót az adatbázisban.

### ***🔹 Autentikációs middleware (authenticateToken)***

* **Célja:** Ellenőrzi, hogy a kérésben kapott token érvényes-e.
* **Működés:**
  1. A req.headers.authorization fejlécből kinyeri a tokent.
  2. Megpróbálja ellenőrizni (jwt.verify(token, SECRET\_KEY)).
  3. Ha érvényes, a felhasználói adatokat (id, email) beállítja req.user változóként.
  4. Ha hibás, **401 (Nincs token)** vagy **403 (Érvénytelen token)** hibát ad vissza.

**Frontend**

**Vite:**

**Leírás:**

A Vite egy fejlesztőkörnyezet másnéven egy build tool, amit frontend alkalmazások gyors fejlesztésére és futtatására használnak.

* **Fő jellemzők:**
  + Gyors és hatékony
  + Automatikusan kezeli a statikus fájlokat

**React.js:**

* **Leírás:** A React.js egy nyílt forráskódú JavaScript könyvtár, amelyet a Meta fejlesztett ki. A React célja, hogy segítsen a fejlesztőknek hatékony és dinamikus felhasználói felületeket kialakítani.
* **Fő jellemzők:**
* Komponens-alapú fejlesztési módszer: A felhasználói felület komponensekre van felosztva, ami lehetővé teszi az egyszerű és újrafelhasználható kód írását.
* Egyirányú adatáramlás: Adatai egyszerűen és hatékonyan áramolhatnak a komponensek között.

**Tailwind:**

* **Leírás :**A Tailwind CSS egy nyílt forráskódú, előre definiált CSS keretrendszer, amelyet Adam Wathan és csapata fejlesztett. Célja, hogy gyors és egyszerű módon segítse a reszponzív weboldalak kialakítását anélkül, hogy egyedi CSS fájlokat kellene írni.
* **Fő jellemzők:**
  + A Tailwind CSS egy reszponzív, utility-first CSS keretrendszer, amely beépített osztályokat biztosít az elrendezésekhez és stílusokhoz. Nem tartalmaz előre definiált komponenseket, de könnyen létrehozhatók és testreszabhatók az egyedi dizájnok.

### ***🔹 regisztráció( POST / register)***

**Leírás:** Új felhasználók létrehozása és regisztrációja az alkalmazásban. A regisztrációs űrlapon a felhasználók megadják az alapvető adatokat, mint név, telefonszám, email cím és jelszó. A rendszer ellenőrzi az email cím duplikációját. Sikeres regisztráció után a felhasználók beléphetnek a rendszerbe.

### ***🔹 Termékek lekérdezése (GET / )***

**Leírás:**  
A felhasználók különböző szűrők (méret, márka, szín, kulcsszó) alapján kereshetnek termékeket az adatbázisban. Az eredmények oldalszámozva jelennek meg, és a szűrők változásával dinamikusan frissülnek. Ha nincs találat, a rendszer erről értesíti a felhasználót.

### ***🔹 Fizetési mód lekérdezése (GET / payment )***

**Leírás:**  
Egy adott fizetési mód lekérdezése a megadott név alapján. Ha a fizetési mód nem található az adatbázisban, a rendszer 404-es hibával válaszol.

### ***🔹 Számla létrehozása (POST /addtocart )***

**Leírás:**  
A felhasználók a fizetési oldalon beírt adatokat tudják felvinni az adatbázisba, és kapnak egy számlát a végén. Illetve ezután kapnak e-mailt a vásárlásról.

# ***Implementáció és fejlesztési terv***

## Asztali admin felület

Az admin felületet a C# programozási nyelven implementáltam. A C# egy modern és rugalmas nyelv a Microsoft által fejlesztett .NET keretrendszerben, amely lehetővé teszi a Windows alkalmazások fejlesztését. Az admin felület létrehozásához számos beépített eszközt és funkciót használtam a .NET keretrendszerből, például a Windows Forms (WinForms) könyvtárat, amely lehetővé teszi az egyszerű és hatékony grafikus felhasználói felületek (GUI) létrehozását.

A C# nyelvben különféle osztályokat és vezérlőket használtam az admin felület különböző részeinek megtervezésére és megvalósítására. A felhasználói interakciók kezelésére eseménykezelőket implementáltam, hogy a felhasználó egyszerűen és hatékonyan kommunikálhasson az admin felülettel. Emellett a MySQL adatbázis-kezelő rendszerrel való kommunikációhoz a MySql.Data.MySqlClient névtérben található funkciókat és osztályokat használtam, amelyek segítségével például adatokat lehet lekérdezni, módosítani vagy törlését végezni a háttérben.

Összességében a C# és a .NET keretrendszer lehetővé tette számomra, hogy hatékonyan és rugalmasan megtervezzem és implementáljam az admin felületet, amely egyszerűen kezelhető és intuitív módon nyújt hozzáférést a rendszer adminisztrációs funkcióihoz.

## ***Webes alkalmazás***

Backend:

A szerveroldali logikát Node.js és Express.js segítségével valósítottam meg. Ezek a technológiák lehetővé teszik a dinamikus adatfeldolgozást és az API-k készítését, ami lehetővé teszi a kliensoldali alkalmazásnak az adatok dinamikus frissítését.

Frontend:

A weboldal frontend részét React keretrendszerrel és Tailwind stíluskeretrendszerrel készítettem el. A React lehetővé teszi a felhasználói felület dinamikus építését és kezelését, míg a Tailwind reszponzív dizájnt biztosít, így az oldal megjelenése tökéletesen alkalmazkodik különböző eszközökön és kijelzőméreteken.

# ***Funkcionális követelmények***

## Asztali admin felület

#### Adatok megjelenítése

Az admin felületen az összes fontos adat megjeleníthető, mint például a felhasználók adatai, a rendelések részletei, stb. Az adatok áttekinthető formában jelennek meg, ami lehetővé teszi az adminisztrátorok számára a gyors és hatékony navigációt.

#### Felhasználó létrehozása (admin)

Az admin felhasználóknak lehetőségük van új felhasználók létrehozására az admin felületen keresztül. Ez magában foglalhatja az új adminisztrátorok létrehozását.

#### Rendelések kezelése

Az admin felületen keresztül az adminisztrátorok hozzáférhetnek és kezelhetik a rendeléseket. Ez magában foglalja az új rendelések fogadását, meglévő rendelések módosítását vagy törlését, valamint a rendelésekkel kapcsolatos egyéb tevékenységeket.

#### Szűrés az adatoknál

A felhasználóknak lehetőségük van szűrni az adatokat az admin felületen különböző szempontok alapján. Ez segít az adminisztrátoroknak az adatok gyors és hatékony keresésében és szűrésében, ami lehetővé teszi számukra, hogy könnyebben megtalálják azokat az információkat, amelyekre szükségük van.

#### Törlés funkció (admin törlése, rendelés törlése)

Az adminisztrátoroknak lehetőségük van a felhasználók vagy rendelések törlésére az admin felületen keresztül. Ez fontos lehetőség, amely lehetővé teszi az adminisztrátoroknak a rendszer tisztán tartását és az esetleges hibák vagy nem kívánt adatok kezelését.

#### Módosítás (rendelés státusza)

Az adminisztrátorok módosíthatják a rendelések státuszát az admin felületen keresztül. Ez lehetővé teszi számukra, hogy frissítsék a rendelések állapotát, és például jelölhessék a rendelést feldolgozottnak vagy futárnál.

## ***Webes alkalmazás***

#### Kezdőoldal

A kezdőoldal az első pont, ahol a felhasználók a webshopba érkeznek. Ez az oldal tartalmaz egy termékeink gombot illetve navigációt az oldal többi pontjához.

#### Partnereink

A partnereink oldalon található az összes olyan márka, akinek a pólója megtalálható a termékeink oldalon. Rákattintással a márkára át dob az adott oldalra.

#### Termékeink

Ezen az oldalon található az összes póló. A keresés lehetséges keresőmezővel, vagy méret, márka, és szín választásával. Az összetett keresés is működik. Van árkategória szerinti szűrés is, ahol a csuszka változtatásával egyből reagál rá az oldal. A szűrés gomb csak ha választunk valami feltételt kell megnyomni. A pólók kiskártyákban jelennek meg. A kiskártya tartalmazza a leírását a pólónak: a méretet, a márkáját, a színt, és az árát. A kosárba gomb megnyomásával kosárba tudjuk tenni az általunk választott terméket. Rákattintással a kártyára át dob a póló részletező oldalára.

#### Termék részletező oldal

A termék részletező oldalon megtalálható három kép a pólóról. A képek mellett lehet választani méretet, illetve mennyiséget hozzáadni, a kosárba rakás gombbal pedig innen is bekerül a kosárba. A három kép alatt találhatók az elérhető színek, ezekre ha rá kattintunk akkor át dob az általunk választott termékhez. Illetve ha rá kattintunk a nagyképekre, akkor kidobja meg nagyobba és tudjuk lapozni őket.

#### Rólunk

Megtalálható két kép rólunk, illetve egy rövid leírás a weboldalról és, hogy mi a célunk.

#### Kapcsolat

A felhasználó itt tudja felvenni velünk a kapcsolatot, ha van bármi problémája. Az adatai megadásával el tudja küldeni, és küldés után egy visszajelzés fogadja, hogy sikeres volt az elküldés, viszont ha valamelyik mezőt nem tölti ki , vagy üresen marad, akkor nem lehet elküldeni.

#### Fiók

A fiók fülre ráhúzva az egeret előjön három lehetőség.

* Bejelentkezés
* Regisztráció
* Kapcsolat

#### Regisztráció

A regisztráció lehetővé teszi a felhasználók számára, hogy létrehozzanak egy fiókot a webshopban. Ez magában foglalja az alapvető információk megadását, mint például a név, telefonszám, e-mail cím és jelszó.

#### Bejelentkezés

A bejelentkezési lehetőség lehetővé teszi a már regisztrált felhasználók számára, hogy belépjenek a saját fiókjukba a webshopban. A bejelentkezés az e-mail cím és jelszó megadásával történik.

#### Felhasználói Adatok

A felhasználói adatok részben a felhasználók adatai találhatóak, például a név, e-mail. A felhasználóknak lehetőségük van adataik módosítására vagy frissítésére ezen a részen.

Illetve ki tudja törölni a fiókját is.

#### Kosár

A kosár az a hely, ahol a felhasználók átmenetileg tárolhatják a kívánt termékeket vásárlás előtt. Ez lehetővé teszi számukra, hogy könnyen áttekintsék a kiválasztott termékeket, és módosítsák a mennyiséget vagy töröljék a termékeket szükség esetén.

#### Rendelés Leadás

A rendelés leadás rész lehetővé teszi a felhasználók számára, hogy befejezzék a vásárlást a kiválasztott termékekkel. Ez magában foglalja a szállítási és fizetési információk megadását, valamint a rendelés véglegesítését.

#### Rendelési előzmények

Ez az oldal csak regisztráció, illetve ha már regisztrált a felhasználó bejelentkezés után jelenik meg. Megjelenik egy kép, az adatok a rendelésről, a rendelés sorszáma stb…

#### Kijelentkezés

Ez csak bejelentkezve vagy beregisztrálva jelenik meg a fiók fülben, ha rá nyomunk kijelentkezik és visszadob Kezdőoldalra.

# ***Nem funkcionális követelmények***

## Asztali admin felület

* Modern stílus
* Átláthatóság
* Egyszerű kezelés

## Webes alkalmazás

* Felhasználó barát kialakítás
* Átláthatóság
* Keresőbarát
* Gyors navigáció
* E-mailes visszajelzés a vásárlásról

# ***Felhasználói dokumentáció***

## Redszerterv (A program célja, lényegesebb funkciói)

### Asztali admin felület:

Az adminisztrációs felület lényeges szerepet tölt be az alkalmazás teljes körű kezelésében és ellenőrzésében. Számos alapvető funkcióval rendelkezik, amelyek segítik az adminisztrátorokat a hatékony működésben és az adatok hatékony kezelésében.

Az adatok megtekintése és keresése egyike az adminisztrációs felület fő funkcióinak. Ez a lehetőség lehetővé teszi az adminisztrátorok számára, hogy könnyen hozzáférjenek és megtekintsék az adatbázisban tárolt információkat, mint például a felhasználók adatai, termékek, rendelések stb. Ezáltal az adminisztrátorok pontos képet kaphatnak az adatok és termékek jelenlegi állapotáról, ami kulcsfontosságú az alkalmazás hatékony kezelésében és fejlesztésében. Részletesebben a program bemutatásában láthatók a funkciók.

### Webes alkalmazás:

A webshop célja, hogy hatékonyan értékesítse a készletét. Az oldalon lehetőség van szabadon böngészni és válogatni a pólók között, részletes leírások segítségével információt szerezni a különböző pólókról, valamint kereséseket végezni a kívánt típusú pólók között. Emellett a felhasználóknak lehetőségük van regisztrálni az oldalon, így képesek lesznek vásárlást végrehajtani, viszont ha nem szeretnének regisztrálni a vásárlás anélkül is megvalósítható. Az oldal minden eszközre reszponzív dizájnt kapott.

## HW követelmények

Az oldal használatához elengedhetetlen egy megfelelően működő és megjelenítésre alkalmas elektronikai eszköz, például számítógép, laptop vagy okostelefon, amely lehetővé teszi a weboldal megtekintését és böngészését. Emellett szükséges az internet elérhetőség is annak érdekében, hogy a felhasználók hozzáférhessenek az oldalhoz és használhassák annak szolgáltatásait.

## SW követelmények

Az alap operációs rendszerbe beépített böngészők általában megfelelőek a weboldal megtekintéséhez.

Ilyen böngészők például:

* Google Chrome
* Opera
* Edge
* Firefox
* Ezek a böngészők rendelkeznek a szükséges funkciókkal és kompatibilitással ahhoz, hogy az átlagos felhasználók számára kielégítő böngészési élményt nyújtsanak.

## ***A program bemutatása***

### ***Asztali admin felület***

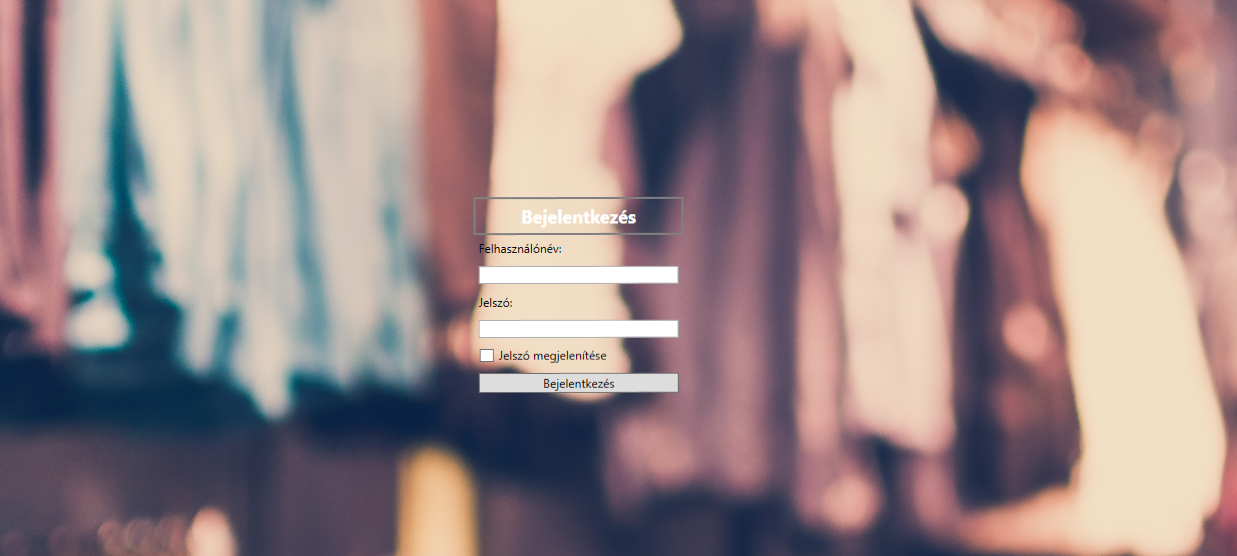
Az admin felületet a C# programozási nyelven implementáltam. A C# egy modern és rugalmas nyelv a Microsoft által fejlesztett .NET keretrendszerben, amely lehetővé teszi a Windows alkalmazások fejlesztését. Az admin felület létrehozásához számos beépített eszközt és funkciót használtam a .NET keretrendszerből, például a Windows Forms (WinForms) könyvtárat, amely lehetővé teszi az egyszerű és hatékony grafikus felhasználói felületek (GUI) létrehozását.

A C# nyelvben különféle osztályokat és vezérlőket használtunk az admin felület különböző részeinek megtervezésére és megvalósítására. A felhasználói interakciók kezelésére eseménykezelőket implementáltam, hogy a felhasználó egyszerűen és hatékonyan kommunikálhasson az admin felülettel. Emellett a MySQL adatbázis-kezelő rendszerrel való kommunikációhoz a MySql.Data.MySqlClient névtérben található funkciókat és osztályokat használtam, amelyek segítségével például adatokat lehet lekérdezni, módosítani vagy törlését végezni a háttérben.

Összességében a C# és a .NET keretrendszer lehetővé tette számunkra, hogy hatékonyan és rugalmasan megtervezzük és implementáljuk az admin felületet, amely egyszerűen kezelhető és intuitív módon nyújt hozzáférést a rendszer adminisztrációs funkcióihoz.

#### Bejelentkezés

Itt tud bejelentkezni az adott admin először a felhasználó nevet utána pedig a jelszót adja meg, majd a belejentkezés gombra kattintva engedi be a felhasználót.



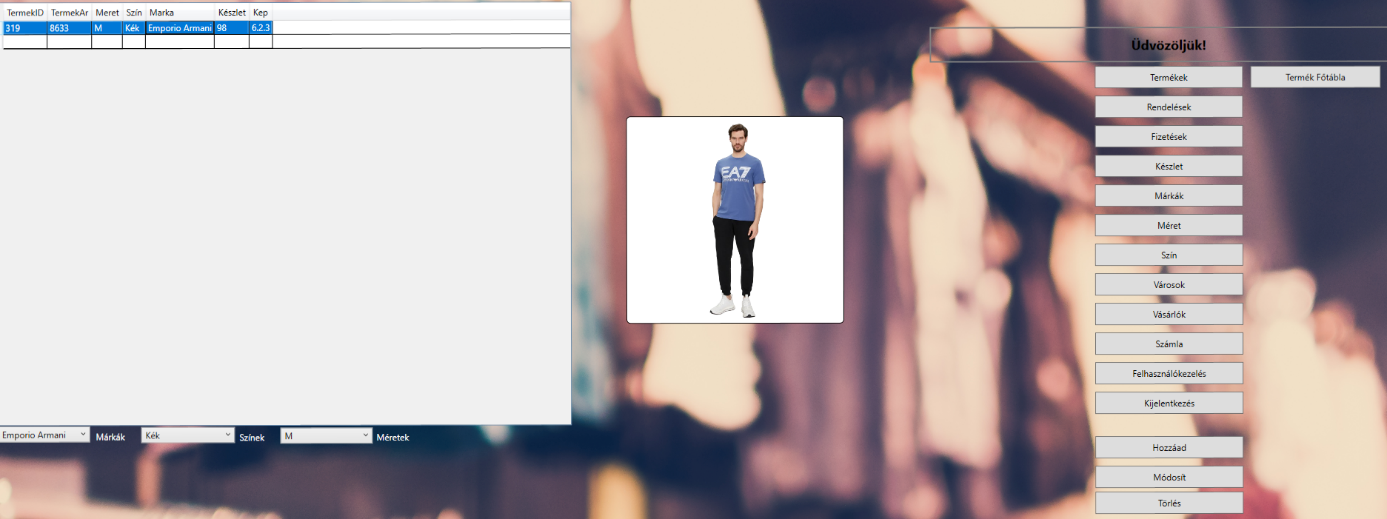
A következő kell lehet tesztelni:

|  |  |  |
| --- | --- | --- |
| Felhasználó név | Jelszó | Jogosultság |
| admin | MolnarZsombor54321 | admin |
| titkárnő | KisLindaTitkarno123\_ | titkarno |

Jogosultságok**:**

Admin: Az admin tud létrehozni felhasználót, törölni, módosítani.

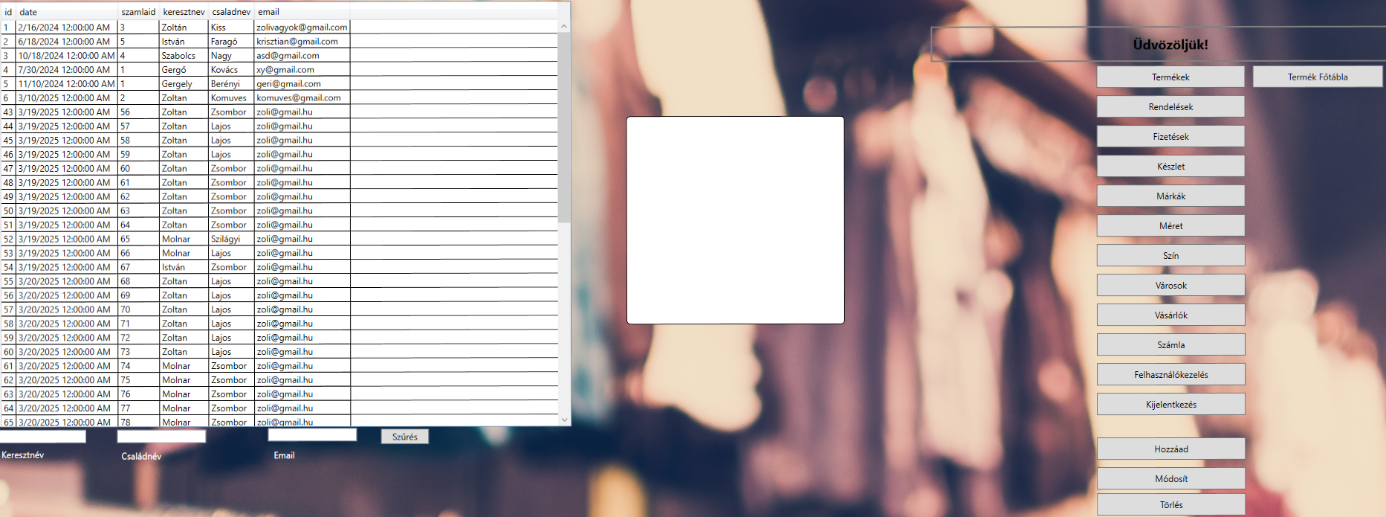
Titkárnő: A titkárnő tud létrehozni, módosítani, de viszont törölni nem.



A jobb oldalon az Admin tudja kezelni a táblákat, és a felhasználókat.

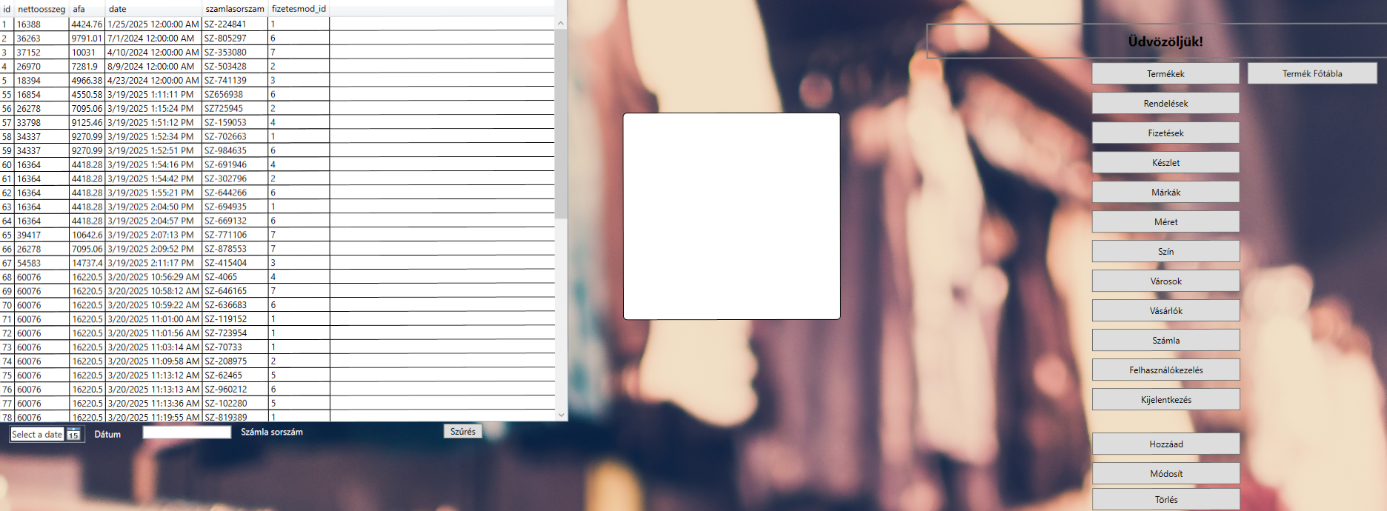
Előszőr megjelenik a teljes Termék view a Datagridbe. Itt a felhasználó tud keresni az adott pólóra, márka, szín, és méret alapján. A Datagrid pedig automatikusan betölti a pólót képpel együtt.

**Rendelések menüpont:**

****

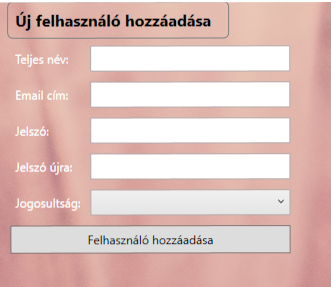
Betölti az összes adatot a Datagridbe, a titkárnő és admin tud szűrni keresztnév, családnév, email alapján. A szűrés gombra kattintva betölti a szűrt adatokat.

**Számla menüpont:**

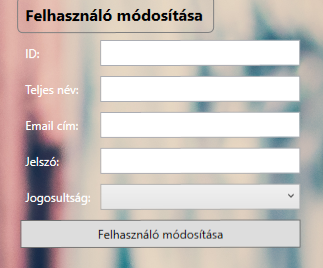
****Az összes számlát betölti a Datagridbe, ahol lehet szűrni dátum, és számla sorszám ala

**Felhasználó** **kezelés:**

**Felhasználó hozzáadása**

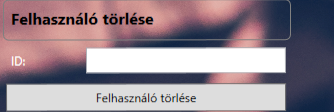
  
  
Itt lehet új felhasználót hozzáadni, az admin és a titkárnő egyaránt tud.

**Felhasználó módosítása:**

****

A felhasználónak itt lehet módosítani az adatait.

**Felhasználó törlése:**

****

A felhasználót itt lehet kitörölni ID alapján.

A felhasználónak két jogosultságot lehet megadni admin, titkárnő, regisztrációs felület azért nincs, mert csak ezzel a két felhasználóval lehet belépni.

Az admin teljes hozzáféréssel tud felhasználót hozzáadni, módosítani, és törölni, míg a titkárnő csak felhasználót tud hozzáadni, és módosítani, törölni nem tud.

A felhasználónak a jelszavát automatikusan sha256-os kódolással tárolja el.  
Vannak táblák például márka, méret, szín tábla amiből nem lehet törölni mert adatvesztéssel jár.

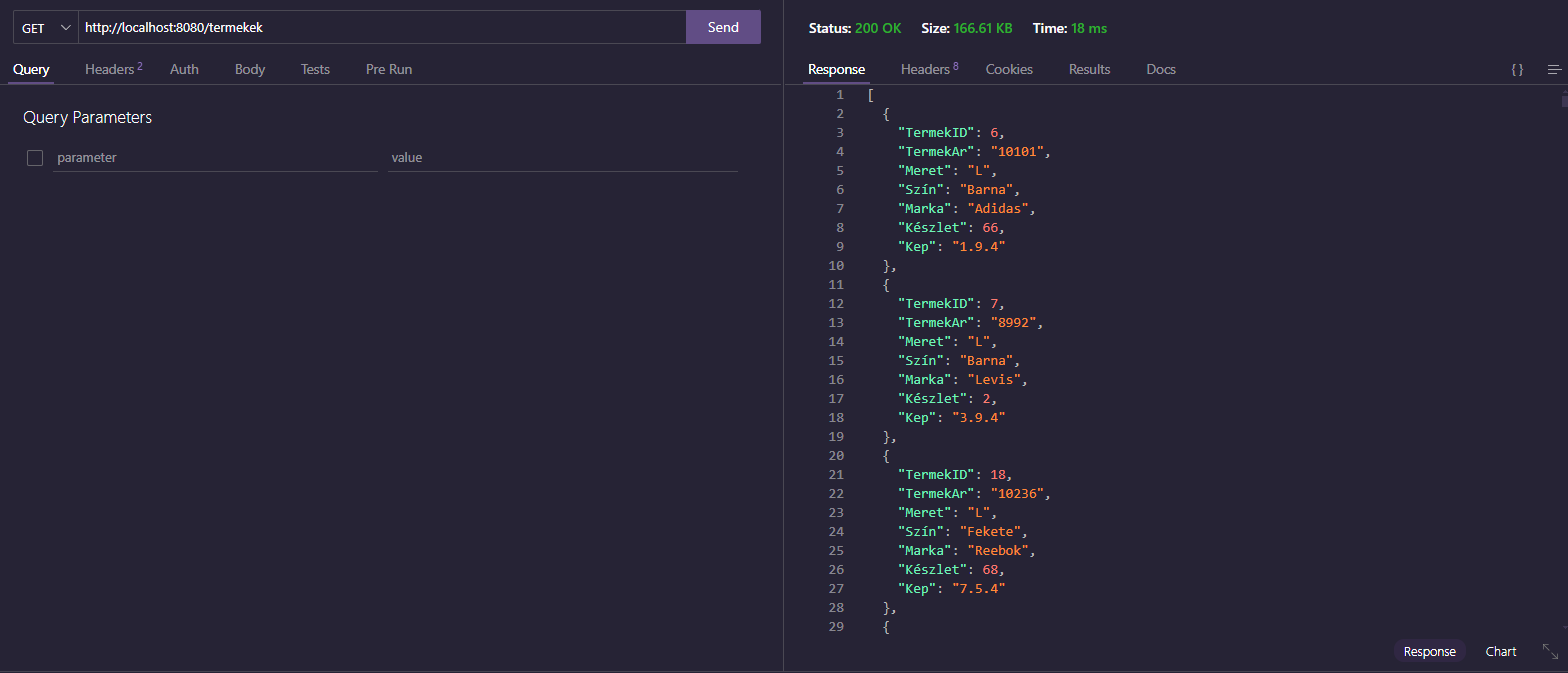
**A-hoz hogy a képek megjelenjenek a következő elérési utat kell megadni:**string id = sor["kep"].ToString();

string kepEleresiUt = $"C:\\Users\\13d\\Desktop\\webshop\_backend2\\pic\\{id}.png";

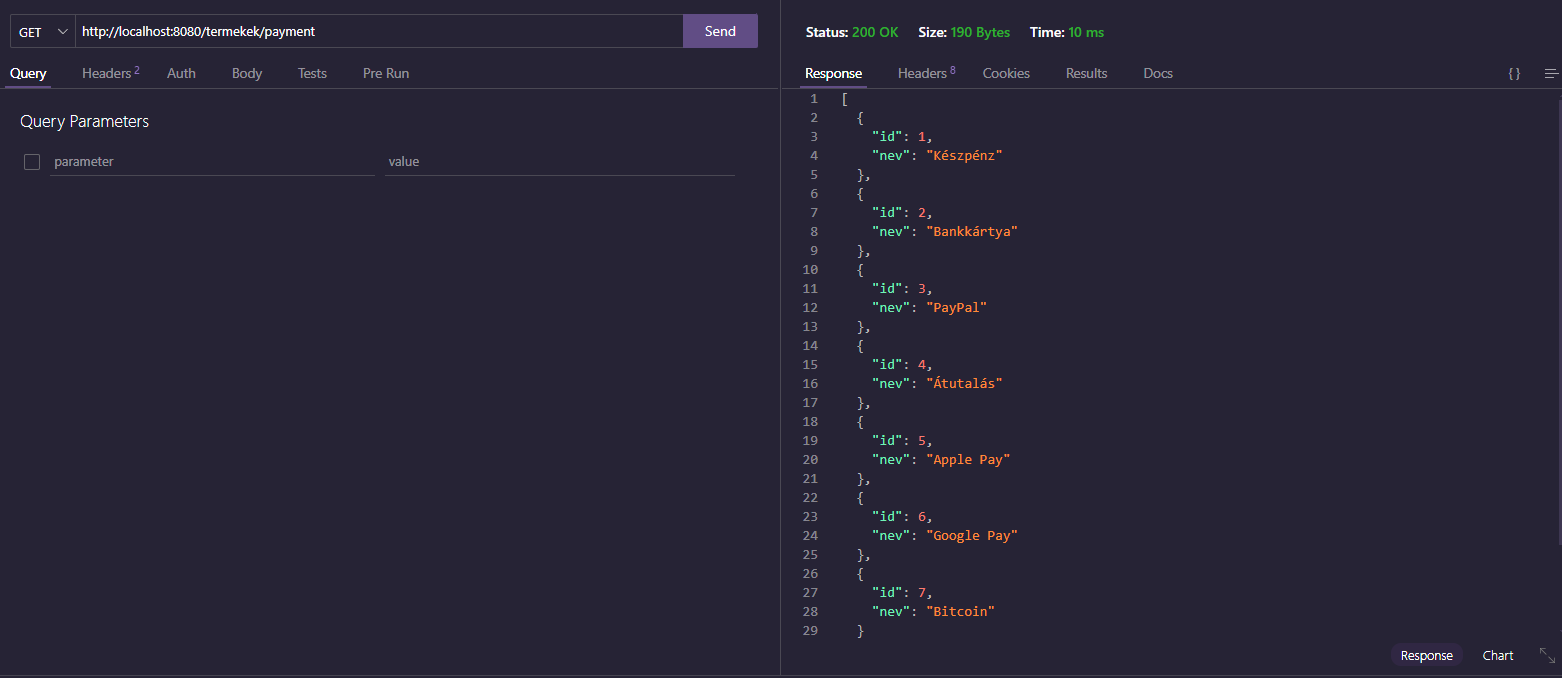
kepek.Source = new BitmapImage(new Uri(kepEleresiUt, UriKind.Absolute));

Backend végpont tesztek

# Termékek lekérdezése:



# Fizetési módok lekérdezése:



Telepítési útmutató (backend és frontend)

1. Lépjünk be a frontend mappába, nyissuk meg a terminált és először írjuk be, hogy npm install, utána pedig npm run dev
2. Lépjünk be a backend mappába, nyissuk meg a terminált és írjuk be, hogy npm install, utána pedig npm start
3. Az adatbázist webaruhaz néven hozzuk létre, mivel igy fog működni az oldal, és az adminpanel is.

Tartalom

[**TrendTees Webshop záródolgozat** 1](#_Toc194051694)

[1. Architektúra és tervezés: 3](#_Toc194051695)

[Backend(Szerveroldal) 8](#_Toc194051696)

[Könyvtárstruktúra 10](#_Toc194051697)

[ Fiók törlése 11](#_Toc194051698)

[📌 API Végpontok Áttekintése 12](#_Toc194051699)

[🔹 Regisztráció (POST /register) 12](#_Toc194051700)

[***🔹*** *Bejelentkezés (****POST /login****)* 13](#_Toc194051701)

[***🔹*** *Felhasználói fiók törlése (****DELETE /account****)* 13](#_Toc194051702)

[***🔹*** *Jelszó módosítása (****PUT /account/password****)* 13](#_Toc194051703)

[***🔹*** *Autentikációs middleware (****authenticateToken****)* 13](#_Toc194051704)

[***🔹*** *regisztráció( POST / register)* 14](#_Toc194051705)

[***🔹*** *Termékek lekérdezése (****GET /*** *)* 15](#_Toc194051706)

[***🔹*** *Fizetési mód lekérdezése (****GET / payment*** *)* 15](#_Toc194051707)

[***🔹*** *Számla létrehozása (****POST /addtocart*** *)* 15](#_Toc194051708)

[***Implementáció és fejlesztési terv*** 15](#_Toc194051709)

[Asztali admin felület 15](#_Toc194051710)

[***Webes alkalmazás*** 16](#_Toc194051711)

[***Funkcionális követelmények*** 16](#_Toc194051712)

[Asztali admin felület 16](#_Toc194051713)

[***Webes alkalmazás*** 17](#_Toc194051714)

[***Nem funkcionális követelmények*** 19](#_Toc194051715)

[Asztali admin felület 19](#_Toc194051716)

[Webes alkalmazás 19](#_Toc194051717)

[***Felhasználói dokumentáció*** 19](#_Toc194051718)

[Redszerterv (A program célja, lényegesebb funkciói) 19](#_Toc194051719)

[Asztali admin felület: 19](#_Toc194051720)

[Webes alkalmazás: 19](#_Toc194051721)

[HW követelmények 19](#_Toc194051722)

[SW követelmények 20](#_Toc194051723)

[***A program bemutatása*** 20](#_Toc194051724)

[***Asztali admin felület*** 20](#_Toc194051725)

[Termékek lekérdezése: 25](#_Toc194051726)

[Fizetési módok lekérdezése: 26](#_Toc194051727)