

**ZÁRÓDOLGOZAT**

Készítették:

Papp Patrícia – Nagy Dávid – Molnár Péter

Konzulens:

Farkas Zoltán

Miskolc

2025.

Miskolci SZC Kandó Kálmán Informatikai Technikum

Miskolci Szakképzési Centrum

SZOFTVERFEJLESZTŐ- ÉS TESZTELŐ SZAK

Dokumentáció

Papp Patrícia – Nagy Dávid – Molnár Péter

2023-2025

Tartalomjegyzék

Bevezetés [1](#_Toc161991981)

A rendszer célja2

A rendszer általános felépítése2

Adatbázis felépítése3

Backend működése8

Frontend működése13

A rendszer működésének menete18

Telepítés és futtatás19

Felhasználói útmutató20

Összegzés21

Munkamegosztás a projektben22

Forrásjegyzék22

**1. Bevezetés**

A záródolgozat célja, hogy bemutassa a vállalkozás által kínált termékeket és szolgáltatásokat, valamint biztosítson egy olyan online platformot, amely lehetővé teszi a potenciális vásárlók számára, hogy könnyedén megismerkedjenek az elérhető lehetőségekkel. Az online jelenlét és az automatizált rendszerek manapság alapvető fontosságúak egy vállalkozás számára, mivel a versenyképesség és a felhasználói élmény javítása érdekében elengedhetetlen, hogy a vásárlók és szolgáltatók közötti kapcsolatok digitálisan is gördülékenyen működjenek. Ebben az esetben ez egy weboldalon keresztül történik, ahol a termékek prezentálása mellett lehetőség van kapcsolódó szolgálatatásra időpontot foglalni.

A projekt keretében három fő komponens került kidolgozásra:

az adatbázis,

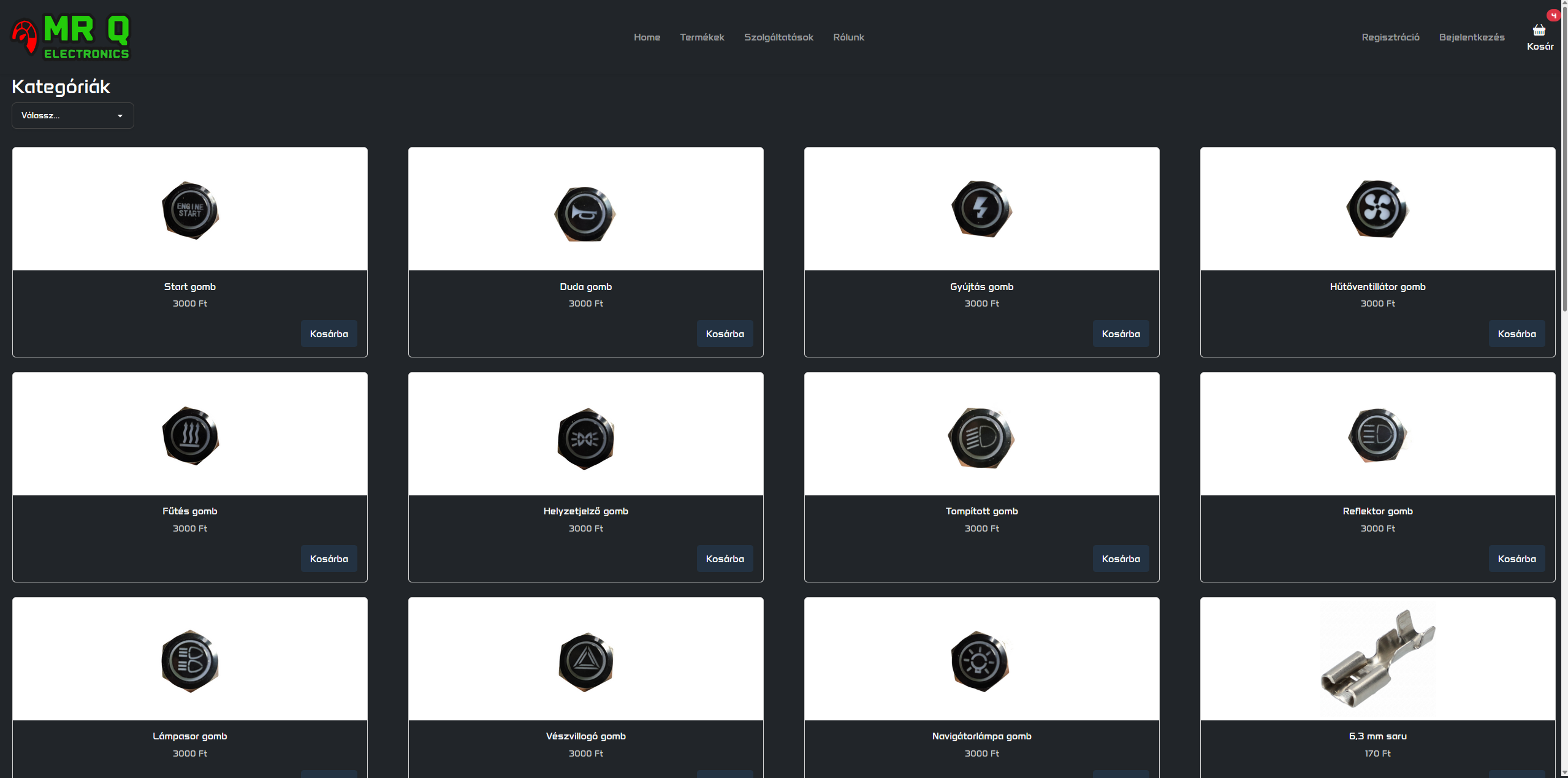
a backend (szerveroldali logika),

valamint a frontend (felhasználói felület).

A rendszer lehetővé teszi a termékek megjelenítését, kategorizálását és időpont foglalását és visszaigazolását a szolgálatatásra.  
 Az alkalmazás kulcsszereplője a könnyen használható admin felület, amely lehetővé teszi a termékek és szolgáltatások gyors hozzáadását, módosítását és eltávolítását, valamint az admin jogosultsággal rendelkező felhasználók nyomon követhetik a lefoglalt időpontokat.

Ez a záródolgozat részletesen bemutatja az alkalmazás kialakításának folyamatát, a használt technológiai eszközöket és megoldásokat, valamint a rendszer működését, amely megfelel a modern webalkalmazások elvárásainak.

Összességében a projekt célja nemcsak a vállalkozás sikerének hozzájárulása, hanem egy olyan platform létrehozása is, amely képes megfelelni a látogatók egyre magasabb felhasználói igényeinek.



**2. A rendszer célja**

A „Webstore” alkalmazás célja egy modern, felhasználóbarát webáruház létrehozása, amely lehetővé teszi a termékek kategóriákba rendezését, adminisztrációját, és a szolgáltatások bemutatását.  
A fejlesztés célcsoportja elsősorban kis- és középvállalkozások, akik gyorsan és egyszerűen szeretnének online jelenlétet biztosítani maguknak.

A rendszer céljai:

* Könnyű kezelhetőség a felhasználók számára
* Egyszerű adminisztrációs felület
* Mobilbarát kialakítás (reszponzív design)
* Adatbiztonság és megbízható működés

**3. A rendszer általános felépítése**

A rendszer három fő komponensből áll:

* **Frontend (React):** A felhasználói felület, ahol a látogatók böngészhetnek a termékek között, rendelhetnek és foglalhatnak időpontot, valamint az adminisztrátorok kezelhetik a kínálatot.
* **Backend (ASP.NET Core):** A szerveroldali logika, amely API-n keresztül kezeli az adatokat (termékek, kategóriák, felhasználók, időpontok stb.).
* **Adatbázis (SQL Server):** Az adatok tárolásáért felelős komponens, táblákkal a termékekről, kategóriákról, felhasználókról.

A komponensek REST API-n keresztül kommunikálnak, így bővíthető, moduláris rendszer jön létre.

**4. Adatbázis felépítése**

****

A MySQL egy nagyon gyors, stabil, relációs adatbázis-kezelő rendszer (angol rövidítéssel RDBMS). Az adatbázis lehetővé teszi az adatok hatékony tárolását, keresését, rendezését és kinyerését. A MySQL kiszolgáló az adatokhoz való hozzáférést szabályozva biztosítja, hogy egyidejűleg többen is használhassák az adatokat, gyorsabb hozzáférést kínál hozzájuk, és garantálja, hogy csak a jogosult Felhasználók szerezhetnek hozzáférést. Ezért a MySQL többfelhasználós, többszálú kiszolgáló.

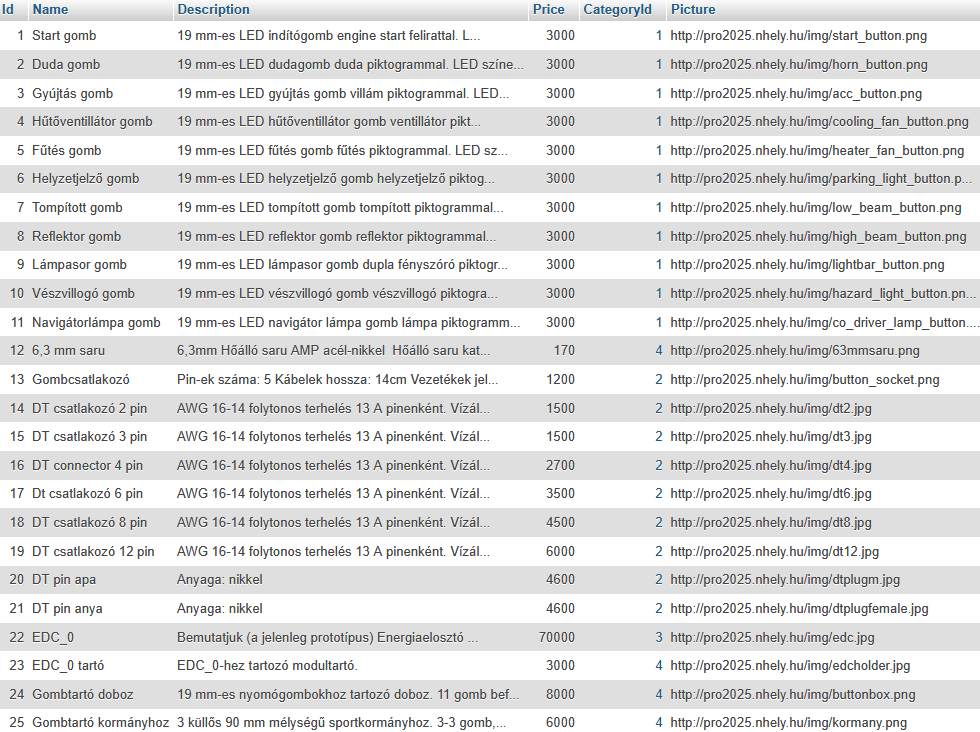
Strukturált lekérdező nyelvet (Structured Query Language- SQL), a szabványos adatbázis-lekérdező nyelvet használja.   
Webáruházunk szerveroldali kiszolgálását egy ASP.NET Core alapú Web API végzi, amely a háttérben MySQL adatbázissal kommunikál. Az adatbázis felel a termékek, felhasználók, időpontok adatainak tárolásáért és kezeléséért.

Az adatbázishoz való kapcsolódást az Entity Framework Core és a MySql.EntityFramework Core biztosítja.  
Az alábbi táblák alkotják az adatbázis gerincét, amelyek együtt biztosítják a rendszer működéséhez szükséges információkat:

* products tábla: Ez a tábla tartalmazza a webáruházban elérhető termékek adatait, például a termék nevét, leírását, árát, képének URL-jét és kategóráját. Minden termékhez egyedi azonosító (id) tartozik.
* users tábla: A felhasználói fiókok kezeléséhez szükséges adatokat tárolja, ideértve a felhasználónevet, email címet és a jelszó hash-t. Ezen tábla segítségével történik a bejelentkezés és jogosultságkezelés nyomon követése.
* categories tábla: A termékek kategóriákba sorolásához szükséges. Minden kategória rendelkezik egy névvel és id-vel, ennek segítségével könnyebben megtalálhatóak a termékek.

Az adatbázis SQL Server alapú, és az alábbi táblákat tartalmazza:

* **Appointments:** Időpontfoglalással kapcsolatos adatok tárolása
* **Aspnetusers:** ez a tábla tartalmazza az összes felhasználót, beleértve az adminisztrátorokat és a vásárlókat. A felhasználó adatai, mint például név, e-mail cím, stb., itt tárolódnak. A kapcsolódó szerepkörök és jogosultságok az Aspnetroles táblán keresztül kezelhetők.
* Aspnetuserroles táblában az egyes felhasználókhoz rendelt szerepköröket tárolják. (pld. admin, user)
* Aspnetroleclaims és Aspnetuserclaims táblák a szerepkörökhöz és felhasználókhoz rendelt specifikus jogosultságokat, jogokat és adatokat tárolják.
* **Aspnetuserlogins, Aspnetusertokens:**  
  Ezek a táblák a felhasználók hitelesítéséhez és jogosultságkezeléséhez szükséges információkat tartalmazzák.
* **Categories:** a termékek kategorizálására: gombok, csatlakozók, modulok, egyéb alkatrészek
* **Products:** Termékek adatainak tárolása: név, leírás, ár, kép, kategória



**Példa – Products tábla szerkezete:**

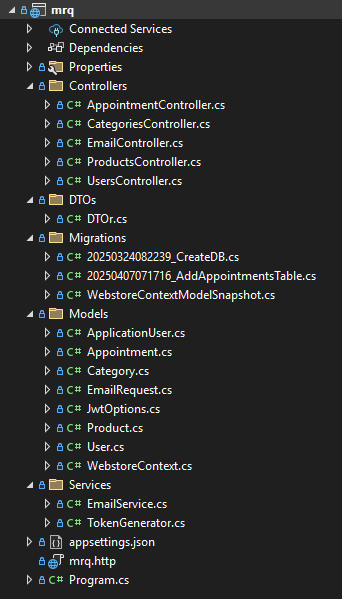
| **Oszlop** | **Típus** | **Megjegyzés** |
| --- | --- | --- |
| Id | int | Egyedi azonosító |
| Name | varchar | Termék neve |
| Price | int | Ár |
| Description | text | Leírás |
| Picture | text | Kép URL |
| CategoryId | int | Kapcsolódó kategória |

A táblák közötti idegen kulcsos kapcsolatok biztosítják az adatbázis integritását, és biztosítják, hogy az adatok konzisztens módon és pontosan legyenek kezelve. Például a CategoryId mező egy idegen kulcs a Products táblában, amely a Categories tábla Id mezőjére hivatkozik. Ez a kapcsolat lehetővé teszi a termékek kategóriákba sorolását, és biztosítja, hogy minden termékhez egy érvényes kategória tartozzon.  
  
Ez a kapcsolat alapvető fontosságú a webáruház működésében, mivel lehetővé teszi a felhasználók számára, hogy könnyen böngésszenek a termékek között kategóriák alapján. A kategóriák szerinti szűrés és keresés lehetősége jelentősen javítja a felhasználói élményt, mivel a vásárlók gyorsan és egyszerűen megtalálhatják a kívánt termékeket anélkül, hogy a teljes termékkínálatot átnéznék.  
  
A CategoryId idegen kulcs biztosítja, hogy egy termék csak akkor lehet elérhető a webáruházban, ha az hozzárendelésre került egy kategóriához. Ha egy kategória törlésre kerül a Categories táblából, az adatbázis automatikusan gondoskodik arról, hogy a kapcsolódó termékek ne maradjanak érvénytelen hivatkozások, vagy más módon biztosítja, hogy azok ne okozzanak hibát az alkalmazás működésében. Az idegen kulcsok így segítenek a rendszer adatbiztonságának és megbízhatóságának fenntartásában.  
  
A CategoryId mező szerepe nemcsak az adatkezelés szempontjából fontos, hanem az adatbázis logikájának tisztaságát és az üzleti logika érvényesítését is segíti, mivel az adatbázis szintjén biztosítja, hogy a termékek mindig egy érvényes kategóriába legyenek sorolva. Ezen túlmenően, az idegen kulcsok alkalmazása segít optimalizálni az adatbázis lekérdezéseit és biztosítja a rendszer adatbázis-szerkezetének bővíthetőségét, lehetővé téve új funkciók, például új kategóriák vagy további kapcsolódó táblák könnyű hozzáadását.

**FTP kapcsolat**

Webáruházunkban a képek feltöltése az FTP protokoll használatával történik, amely biztosítja a gyors és biztonságos adatátvitelt. Az FTP (File Transfer Protocol) lehetővé teszi a fájlok egyszerű kezelését és átvitelt a szerverek között.  
Az FTP kapcsolat létesítéséhez szükséges egy FTP kliens program. A kapcsolat létrehozásához meg kell adni a következőket:  
Host (szerver kiszolgáló URL): pro2025.nhely.hu  
Felhasználónév: felhasznalonev  
Jelszó: jelszo  
Csatlakozás után az egyik oldalt a számítógép mappái, a másik oldalt a távoli szerver fájljai jelennek meg.  
Itt külön ablakban, a megfelelő mappába navigálva adhatjuk hozzá a képeket és fájlokat.  
Miután a képek feltöltésre kerültek, a ezzel a módszerrel a webáruház automatikusan elérhetővé teszi őket a megfelelő oldalon az URL cím segítségével. Pl.:<http://pro2025.nhely.hu/img/edc.jpg>  
A feltöltött fájlok bármikor hozzáférhetők, és az URL-ek segítségével gyorsan megjeleníthetők az online áruházban. Az FTP protokoll lehetőséget ad arra is, hogy könnyen karbantartsuk és rendszerezzük a fájlokat, elkerülve a helyi tárolókapacitás túlterhelését.

**5. Backend működése**



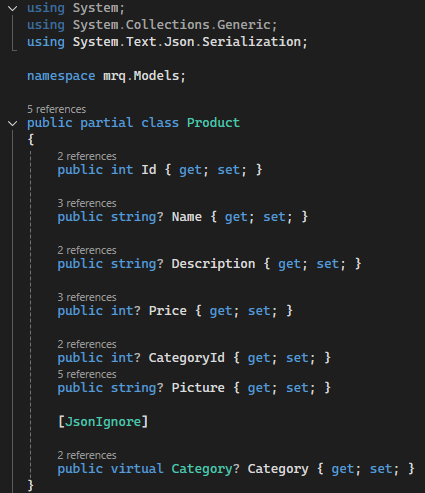
A backend ASP.NET Core technológiával készült, MVC architektúrában. A főbb elemek:

* **Controllers:** API végpontokat kezelnek (pl. ProductsController, CategoriesController)
* **Models:** Az adatmodellek definíciója (pl. Product, Category, Order)
* **Entity Framework Core:** ORM eszköz, ami az adatbázis-műveleteket leegyszerűsíti
* **Identity:** Felhasználókezelés és autentikáció

A Controllers (vezérlők) az MVC architektúra központi komponensei, amelyek felelősek az API végpontok kezeléséért. Ezek az osztályok biztosítják az alkalmazás üzleti logikáját, és kezelik a HTTP kéréseket, válaszokat, valamint az adatok feldolgozását. A vezérlők fogadják a bejövő HTTP kéréseket, meghatározzák, hogy mely adatokat kell visszaadni a felhasználónak, és végrehajtják az adatkezelési műveleteket a Models osztályokkal és az Entity Framework Core ORM eszközzel.  
  
A conrollerek az alkalmazás különböző részeihez tartozó végpontokat működtetik. Például:  
ProductsController: A termékekkel kapcsolatos műveleteket kezeli, mint a termékek listázása, hozzáadása, módosítása vagy törlése.  
CategoriesController: A kategóriák kezelésével kapcsolatos műveleteket végzi, például új kategória létrehozása vagy a meglévő kategóriák lekérdezése.  
Másik fontos controllerünk az EmailController, amely az e-mailek küldésére szolgáló végpontot biztosít. A vezérlő az EmailService szolgáltatás segítségével kezeli a regisztrációhoz és időpontfoglaláshoz szükséges e-mailek küldését és válaszol a kliens kéréseire.A SendEmail végpont felelős az e-mailek küldéséért. A POST kérésre várja az EmailRequest objektumot, amely tartalmazza az e-mail címzettjét, tárgyát és tartalmát. Ha bármelyik adat hiányzik, akkor egy BadRequest választ küldünk vissza, amely tartalmazza az "Email küldési hiba" üzenetet.

A Models mappában található osztályok a MySQL adatbázis tábláit tükrözik. Ezek az osztályok szolgálnak a relációs adatok objektum-alapú reprezentációjára. Az Entity Framework Core ezen modellek alapján végzi az adatbázisműveleteket.  
Az alkalmazás adatbázis-műveleteinek kezelése az Entity Framework Core (EF Core) ORM (Object-Relational Mapping) eszköz segítségével történik. Az EF Core lehetővé teszi, hogy az adatbázisműveletek, mint például lekérdezések, frissítések és törlés, közvetlenül C# objektumokon keresztül történjenek, anélkül hogy SQL kódot kellene írni. Ez leegyszerűsíti az adatbázissal való interakciót és tisztább kódot eredményez.

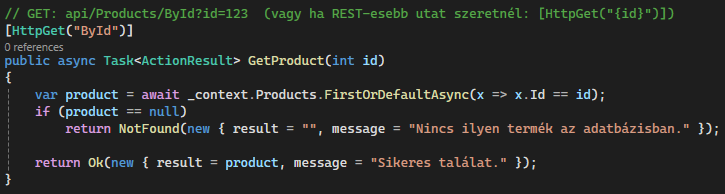
Adatmodellek létrehozása  
Az EF Core használatának első lépése, hogy létrehozzuk az adatmodell osztályokat, amelyek az alkalmazásban használt adatokat reprezentálják. Például a Product (termék) és Category (kategória) osztályok az adatbázis tábláit tükrözik. Ezek az osztályok tartalmazzák azokat az adatokat, amiket az alkalmazás kezelni fog, és lehetővé teszik az adatok manipulálását a kód szintjén.  
Az adatmodell osztályokban lévő tulajdonságok az adatbázis táblák oszlopait képviselik, és az EF Core automatikusan leképezi őket a megfelelő MySQL adatbázis táblákra. Az alábbi példa bemutatja, hogyan nézhet ki egy egyszerű adatmodell osztály:



• Identity – Felhasználókezelés és autentikáció  
Az ASP.NET Core Identity keretrendszere biztosítja a felhasználók regisztrációját, bejelentkezését, jelszókezelését és szerepköralapú jogosultságait. Támogatja a jelszó-hashelést, email-megerősítést és egyéni jogosultságokat valamint integrálható JWT tokenes autentikációval a React-hoz.  
Felhasználói adatokat ezután egy MySQL adatbázisban tárolja (pl. AspNetUsers, AspNetRoles táblákban).

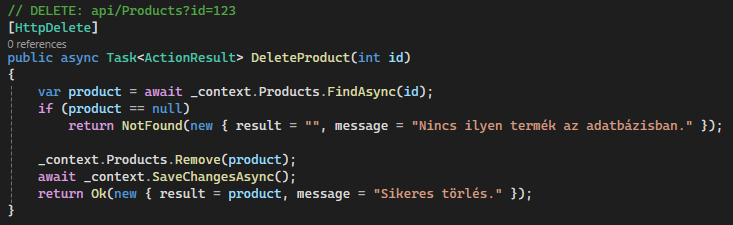
A backend API a CRUD műveleteket (Create – Létrehozás, Read – Lekérdezés, Update – Módosítás, Delete – Törlés) a REST architektúra szerint valósítja meg. Ezek a műveletek biztosítják az adatok kezelését (pl. termékek, kategóriák) a MySQL adatbázison keresztül, az Entity Framework Core ORM segítségével.  
Új termék létrehozása egy http POST kérésen kereszül történik.  
 A felhasználók vagy adminisztrátorok új termékeket, kategóriákat vagy egyéb entitásokat adhatnak hozzá a rendszerhez. Ehhez a backend az adatot a MySQL adatbázisba menti, és az új entitás létrehozásáról egy 201 Created válasz kíséretében értesíti a frontend alkalmazást.





A lekérdezési műveletek HTTP GET kérésekkel valósulnak meg, amelyek az adatokat kérdezik le a rendszerből. Az API különböző végpontokon biztosítja a teljes adatbázis tartalmának vagy egyes entitásoknak a lekérdezését. A válasz tartalmazza az adott entitások adatait. Az adatbázisból való lekérdezés során az adatok a MySQL adatbázisból kerülnek visszaadása az alkalmazás számára.

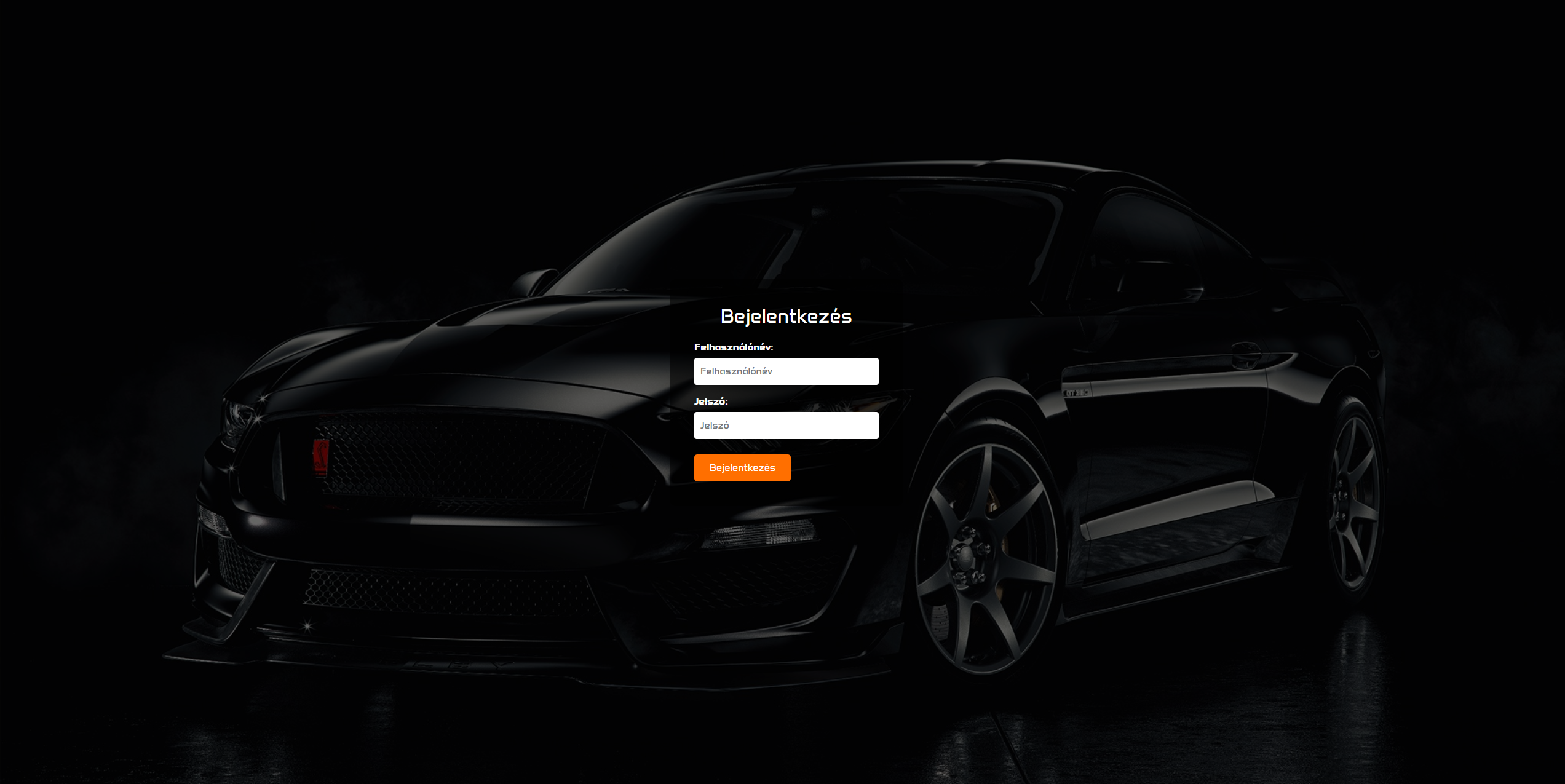
  
A módosítási műveletek HTTP PUT kérésekkel történnek, amelyek lehetővé teszik az entitások frissítését a rendszerben. A felhasználók vagy adminisztrátorok meglévő termékeket vagy más adatokat módosíthatnak a backend rendszerben. Az alkalmazás az új adatokat az adatbázisba menti, és ha a módosítás sikeres, 204 No Content válasz érkezik vissza, amely azt jelzi, hogy a művelet sikeresen végrehajtásra került.

  
A törlés HTTP DELETE kérésekkel történik. Ezen keresztül lehetőség van egy adott entitás eltávolítására az adatbázisból. Ha a törlés sikeresen megtörténik, a rendszer 204 No Content válaszban értesíti a frontend alkalmazást. A törlés nem csak az adatokat távolítja el, hanem a kapcsolódó referenciákat is eltávolítja, amennyiben szükséges.  
A CRUD műveletek segítségével a webáruház backend rendszerében biztosított az adatok létrehozása, lekérdezése, módosítása és törlése. Így könnyen karbantarthatóvá tudjuk tenni a termékeket és az adatokat a megadott adminisztrátorok által.

Az adatmodellek és a MySQL adatbázis közötti kapcsolatot az DbContext osztály biztosítja. A DbContext az az osztály, amely felelős az adatbázisműveletek végrehajtásáért, például az adatok lekérdezéséért, beszúrásáért, frissítéséért és törléséért. Az DbContext osztály tartalmazza az adatbázis táblákat reprezentáló DbSet típusú tulajdonságokat, amelyek lehetővé teszik az adatok egyszerű kezelését.

A backend REST API-t szolgáltat, amelyet a frontend használ.

**6. Frontend működése**

  
A frontend alkalmazás a React keretrendszer segítségével készült, amely lehetővé teszi a dinamikus és interaktív felhasználói felületek gyors fejlesztését. A React az egyik legnépszerűbb JavaScript könyvtár a modern webalkalmazások építésére, és számos fejlesztési előnnyel rendelkezik, mint például a komponens-alapú felépítés, a virtuális DOM gyors renderelése és a könnyű állapotkezelés.  
  
A projekt során törkedtünk a letisztult, modern dizájnra, valamint a felhasználói élményre (UX), hogy a felület könnyen navigálható és intuitív legyen. A felhasználói felület modern megjelenése és reszponzív viselkedése biztosítja, hogy az alkalmazás minden eszközön jól működjön, legyen szó asztali számítógépről, táblagépről vagy mobiltelefonról.

Főbb technológiák:

• **React.js** – komponens alapú felépítés

• **React Router** – navigáció kezelése

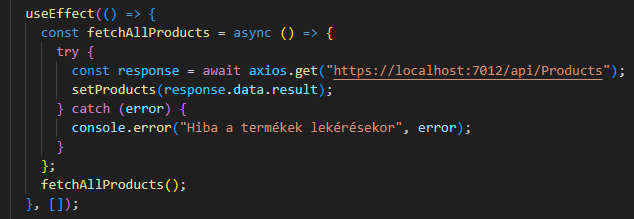
• **Axios** – API hívások

• **Bootstrap 4** – reszponzív dizájn

A React.js egy JavaScript könyvtár, amely lehetővé teszi a dinamikus, komponens-alapú felhasználói felületek építését. A React komponens-alapú megközelítése azt jelenti, hogy az alkalmazás felhasználói felületét kisebb, újrahasznosítható komponensekre bontjuk, amelyek önállóan kezelik a saját állapotukat és viselkedésüket, amely segít a kód modularizálásában és a karbantartás egyszerűsítésében.  
Például a ProductCard komponens, amely egy-egy terméket jelenít meg, és a termék adatait a props-k segítségével kapja meg. Ez a komponens könnyen újrahasználható, és más részein is elhelyezhető a weboldalnak, ahol a kívánt terméket szeretnénk megjeleníteni.

A React Router DOM egy olyan csomag, amely lehetővé teszi a React alkalmazásokban a navigációt. S segítségével a különböző oldalakat külön-külön komponensekbe tudjuk szervezni, és ezeket a komponenseket a Route komponens segítségével tudjuk megjeleníteni.

Az adatok változásait követően fontos, hogy az alkalmazás felülete szinkronban legyen az adatbázissal, hogy a felhasználó a legfrissebb információkat lássa anélkül, hogy manuálisan kellene frissíteni az oldalt. A React a komponensek újrarenderelésével biztosítja, hogy a felület az új állapotnak megfelelően frissüljön.  
.A React Router a navigáció kezelésére szolgál, lehetővé téve az egyszerű oldalváltásokat és a dinamikus tartalom megjelenítését.   
Az Axios egy népszerű JavaScript könyvtár, amely egyszerűsíti az HTTP kérések (GET, POST, PUT, DELETE) kezelését az API-val való kommunikáció során. Az Axios segítségével könnyen integrálhatjuk a REST API-t a frontend alkalmazásba, lehetővé téve a dinamikus adatlekérést és frissítést az adatbázisból.

  
Az Axios promisszal dolgozik, így lehetőséget biztosít aszinkron kérések küldésére és a válaszok kezelésére.  
A state management segítségével (pl. React useState, useEffect vagy globális állapotkezelők, mint Redux) a komponens állapotát módosíthatjuk, és az új adatokat dinamikusan megjeleníthetjük a felhasználói felületen anélkül, hogy a felhasználónak bármit is tennie kellene.  
A frontend megjelenését és elrendezését a Bootstrap 4 CSS keretrendszer segítségével alakítottuk ki. A Bootstrap egy könnyen használható, reszponzív dizájnra optimalizált keretrendszer, amely a felhasználói felület gyors fejlesztésére és testreszabására szolgál. A Bootstrap 4 tartalmazza az alapvető UI komponenseket (például gombok, navigációs sávok, kártyák, űrlapok), amelyek gyorsan és hatékonyan beépíthetők az alkalmazásba.  
A reszponzív dizájn biztosítja, hogy az alkalmazás a különböző képernyőméreteken (asztali, tablet, mobil) is jól jelenjen meg. A Bootstrap a grid rendszerét használja a rugalmas elrendezéshez, amely lehetővé teszi az elemek dinamikus elrendezését a képernyő szélességének függvényében. A kosár gombhoz használtunk a Bootstrap Icons-t, ezek egyszerűen integrálhatók weboldalakba és alkalmazásokba, és rugalmasan testreszabhatóak CSS stílusok segítségével. Valamint badge elemeket, ami egy kis, stilizált szövegbuborék, amelyet gyakran használnak az adatok kiemelésére, például értesítések, számlálók vagy státuszok megjelenítésére.

A React alapja a komponens, amely a felhasználói felület legkisebb újrahasználható egysége.

**Fő komponensek:**

* ProductList.jsx – termékek megjelenítése
* AddNewProduct.jsx – új termék hozzáadása
* CategoryList.jsx – kategóriák listázása
* AdminDashboard.jsx – admin felület
* Login.jsx és Register.jsx – hitelesítés
* AppointmentScheduler.jsx - időpontfoglalás

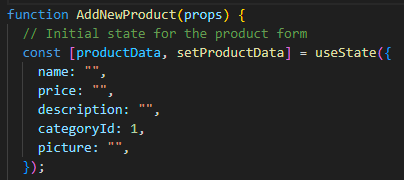
A React-ben a felhasználói felületet JSX-el írjuk, ami egy szintaktikai kiterjesztése a JavaScript-nek, amely lehetővé teszi HTML-szerű kód használatát a JavaScript-ben. A JSX kódot végül a React fordítja le JavaScript-re.

A hooks a React 16.8-ban bevezetett új funkciók, amelyek lehetővé teszik a funkcionális komponensek számára, hogy állapotot és egyéb React funkciókat használjanak, mint például életciklus metódusok és context használata.  
Leggyakoribb hook-ok:

• useState: Az állapot kezelésére szolgál.

• useEffect: Az életciklus metódusokat helyettesíti, és segít az oldal betöltése után végrehajtani műveleteket.

• useContext: Context API használata a funkcionális komponensekben.  
A state a komponens belső állapotát jelenti. A state lehetővé teszi, hogy egy komponens dinamikusan változtassa meg a megjelenített adatokat, és reagáljon a felhasználói interakciókra.

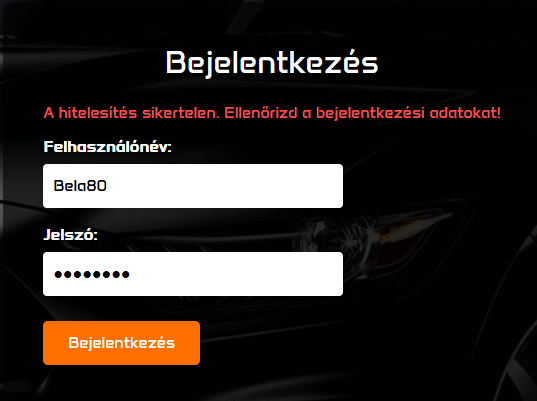


A **React Context** egy beépített API, amely lehetővé teszi, hogy adatokat globálisan elérhetővé tegyünk több komponens számára anélkül, hogy a props-okat minden szinten kézzel tovább kellene adni. Ez különösen hasznos olyan adatok kezelésénél, mint a téma, nyelv, felhasználói adatok, vagy bárminemű olyan állapotok, amelyek több komponens számára is elérhetőek kell legyenek. Segítségével hatékonyan és egyszerűen oszthatunk meg adatokat az alkalmazás különböző részei között, anélkül, hogy minden egyes komponensnek explicit módon továbbítanunk kellene a props-okat.

mondatokban

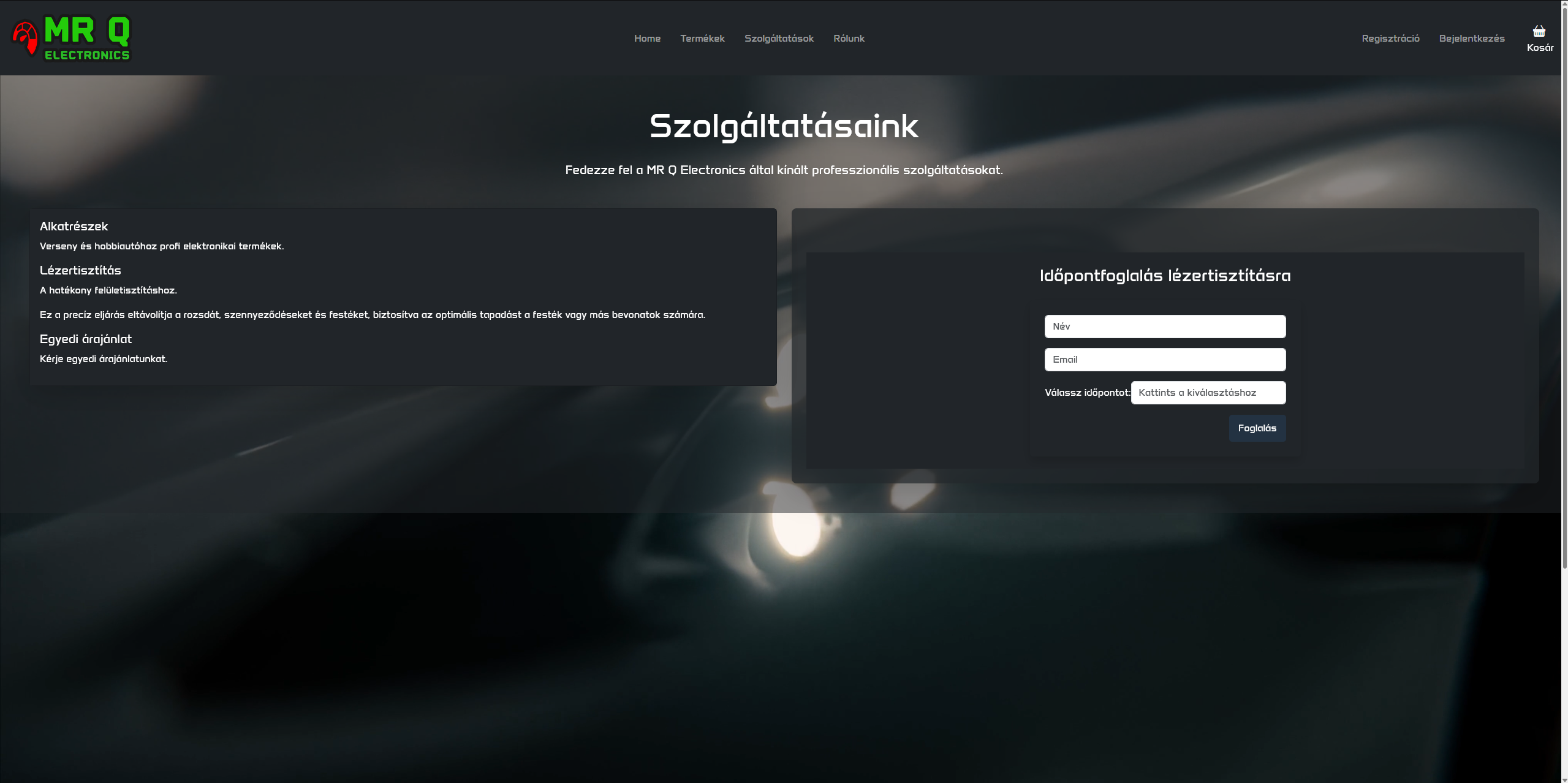


A **Login** komponens felelős a felhasználók bejelentkezésének kezeléséért az alkalmazásban. A felhasználó megadja a felhasználónevét és jelszavát, amit a rendszer egy POST kérés segítségével elküld a backendnek. A backend válaszként egy JSON Web Tokent (JWT) küld vissza, amely tartalmazza a felhasználó azonosító adatait és jogosultságait. Ezt a tokent a frontend elmenti a localStorage-ban, majd a jwt-decode könyvtár segítségével dekódolja, hogy a felhasználó adatait (pl. felhasználónév, szerepkör) kinyerje.



A hitelesítés után a felhasználó hozzáférést kap a rendszer védett részeihez, amennyiben admin jogosultsággal rendelkezik. Ha a bejelentkezés nem sikerül (például helytelen adatok miatt), akkor a felhasználó hibajelzést kap a képernyőn. A komponens a useState hookot használja a felhasználónevet, jelszót, hibát és a felhasználói adatokat tárolására, míg a useNavigate hookkal biztosítja a navigációt a sikeres bejelentkezést követően.

Az **AppointmentScheduler** komponens a felhasználók számára lehetővé teszi időpontok foglalását lézertisztításra. A komponens a felhasználói interakciók alapján összegyűjti a szükséges adatokat (név, email, időpont) és azokat egy API segítségével továbbítja a backend rendszerhez. Ezen kívül kezeli a már foglalt időpontokat, hibák kezelését és visszajelzéseket biztosít a felhasználók számára.



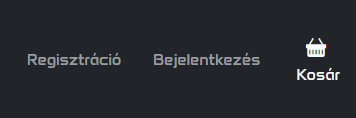
**Felhasználói élmény:**

* Letisztult design
* Könnyen átlátható menürendszer
* Egyszerű és gyors működés
* Mobilbarát megjelenés

A frontend közvetlenül a backend API-jához csatlakozik, így a működés valós időben reagál az adatok változására.

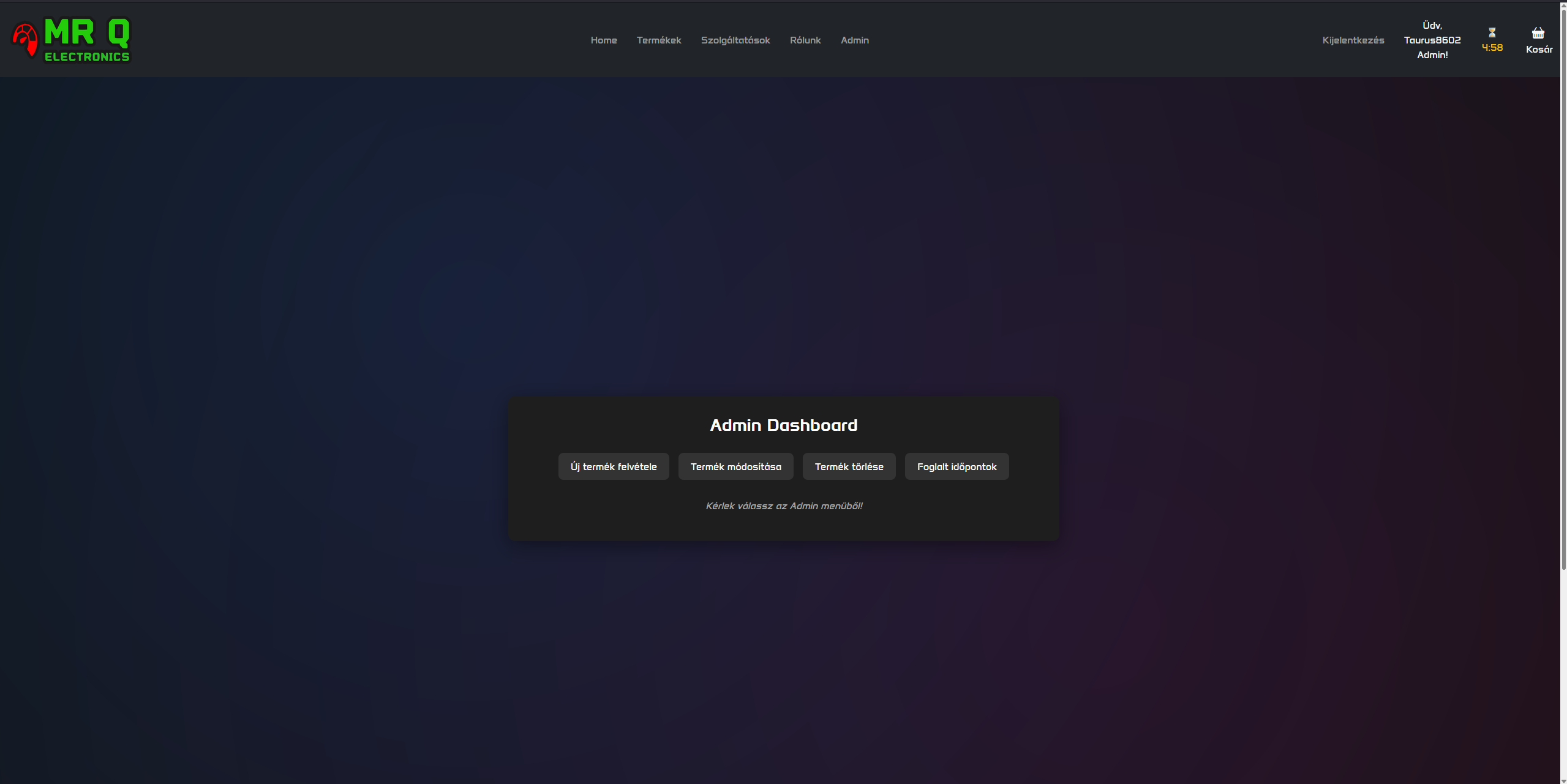
**7. A rendszer működésének menete**

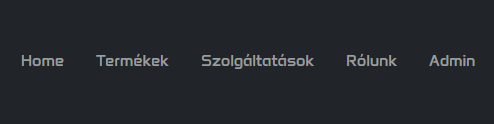
**Felhasználói oldal:**

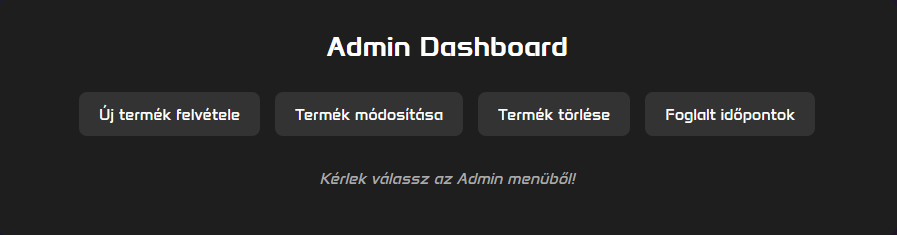


1. A látogató megnyitja a kezdőoldalt, ahol böngészhet a termékek között.
2. Regisztrálhat vagy bejelentkezhet.
3. Kosárba helyezheti a kiválasztott termékeket.
4. Megrendelés leadása után visszaigazolást kap.

**Adminisztrátor oldal:**







1. Belépés után elérhetővé válik az admin dashboard.
2. Itt új termékeket lehet létrehozni, módosítani és törölni.
3. A foglalt időpontok nyomon követésére is zolgál.

**8. Telepítés és futtatás**

**Backend (ASP.NET Core):**

1. Nyisd meg a projektet Visual Studio-ban.
2. Állítsd be az adatbázis kapcsolatot appsettings.json-ban.
3. Futtasd a migrációt:

dotnet ef database update

dotnet run

**Frontend (React):**

1. Navigálj a frontend mappába:

cd frontend

npm install

npm start

A rendszer ezután a http://localhost:3000 címen elérhető.

**9. Felhasználói útmutató**

**Vásárlói oldal:**

1. **Főoldal megnyitása:** A látogató először a főoldallal találkozik.
2. **Regisztráció:** Új felhasználó esetén a regisztrációs űrlap kitöltésével lehet fiókot létrehozni.
3. **Bejelentkezés:** A rendszerbe történő belépéshez e-mail és jelszó szükséges.
4. **Kosár használata:** A termékek mellett található gombbal kosárba helyezhetők, módosíthatók, törölhetők a kosárból
5. **Megrendelés:** A kosárban szereplő termékek megrendelhetők.
6. **Időpontfoglalás**: a szolgáltatások menüponton belül lehetőség van időpontot foglalni, amelyről visszaigazoló email-t kap a felhasználó.

**Admin oldal:**

1. **Bejelentkezés admin fiókkal**
2. **Termék hozzáadása:** Új termék űrlap kitöltése (név, leírás, ár, kategória stb.)
3. **Termék módosítása, törlése**
4. **Foglalások áttekintése:** a korábban érkezett foglalások áttekintése, törlése

**10. Összegzés**

A „Webstore” projekt egy jól strukturált, modern webalkalmazás, amely három fő technológiai komponensre épül: React frontend, ASP.NET Core backend és SQL Server alapú adatbázis.

A fejlesztés során figyelembe vettük:

* a felhasználói élményt,
* az adminisztratív egyszerűséget,
* valamint a skálázhatóságot és biztonságot.

A projekt célkitűzéseit sikerült megvalósítani. A rendszer stabilan működik, könnyen telepíthető és továbbfejleszthető.

**11. Munkamegosztás a projektben**

* **Papp Patrícia** – az adatbázis megtervezése és kialakítása (táblák, relációk, export)
* **Nagy Dávid** – backend fejlesztés (ASP.NET Core, API-k, felhasználókezelés)
* **Molnár Péter** – frontend fejlesztés (React, komponensek, dizájn)

A csapattagok rendszeresen egyeztettek a feladatok során, a projekt előrehaladását Trello-ban követték nyomon, a forráskódot pedig a GitHub-on tárolták.

**🔗 GitHub repository:**

<https://github.com/Peter86111/Projekt1.git>

**12. Forrásjegyzék**

1. Microsoft Docs – ASP.NET Core dokumentáció  
   Webcím: <https://learn.microsoft.com/en-us/aspnet/core>, letöltés dátuma: 2025.03.15.
2. JWT-tulajdonos hitelesítésének konfigurálása a ASP.NET Core-ban  
   Webcím: <https://learn.microsoft.com/hu-hu/aspnet/core/security/authentication/configure-jwt-bearer-authentication?view=aspnetcore-9.0>

Cikk - 2025. 01. 19

1. React dokumentáció – reactjs.org  
   Webcím: [https://reactjs.org](https://reactjs.org/), letöltés dátuma: 2025.03.15.
2. Bootstrap dokumentáció  
   Webcím: https://getbootstrap.com/docs/4.4, letöltés dátuma: 2025.03.16.
3. W3Schools SQL Tutorial  
   Webcím: https://www.w3schools.com/sql, letöltés dátuma: 2025.03.16.
4. Entity Framework Core dokumentáció  
   Webcím: <https://learn.microsoft.com/en-us/ef/core/>, letöltés dátuma: 2025.03.15
5. React Router

Webcím: https://reactrouter.com/, letöltés dátuma: 2025.04.07.

1. Bootstrap Icons és Badges

<https://getbootstrap.com/docs>, letöltés dátuma: 2025.04.07.