



ZÁRÓDOLGOZAT

Készítették:

Mecsei Péter – Sándor Kristóf Illés

Konzulens:

Kerényi Róbert Nándor

Miskolc

2024.

Miskolci SZC Kandó Kálmán Informatikai Technikum

Miskolci Szakképzési Centrum

SZOFTVERFEJLESZTŐ- ÉS TESZTELŐ SZAK

Webshop

Bemutatósa

Mecsei Péter – Sándor Kristóf Illés

2023-2024

Tartalomjegyzék

1 - Bevezetés ---	1.oldal
2 - Adatbázis és a phpMyAdmin ---	1.oldal
.....2.1 Fejlesztői környezet, maga a phpMyAdmin ---	1.oldal
.....2.2 Miért hasznos a phpMyAdmin ---	1--2.oldal
.....2.3 WampServer ---	2.oldal
.....2.4 Adatbázis elképzelés ---	2-3.oldal
.....2.5 Megvalósítás ---	3-4.oldal
.....2.5.1 Normál forma ---	4.oldal
3 – Frontend ---	4.oldal
.....3.1 React ---	4-5.oldal
.....3.2 Node.js ---	5.oldal
.....3.3 Visual Studio Code ---	6.oldal
.....3.4 JavaScript ---	6-7.oldal
.....3.5 HTML és CSS ---	7-8.oldal
.....3.6 A kezdetek ---	8-9.oldal
.....3.7 A webshop részletes bemutatása ---	9-10.oldal
.....3.7.1 Regisztráció ---	10.oldal
.....3.7.2 Bejelentkezés ---	11.oldal
.....3.7.3 Navigáció az oldalon ---	12.oldal
.....3.7.4 Kosár ---	13.oldal
.....3.7.5 Keresés ---	13-14.oldal
4 - Backend ---	14.oldal
.....4.1 WebAPI ---	14.oldal
.....4.1.1 Fontos tulajdonságai ---	14-15.oldal
.....4.1.2 WebAPI-t használó szoftverek ---	15.oldal
.....4.2 ASP.NET Core ---	15.oldal
.....4.2.1 Főbb jellemzői ---	15-16.oldal

.....	4.2.2 Felhasználási területei ---	16.oldal
.....	4.3 Visual Studio 2022 ---	16.oldal
.....	4.3.1 Főbb jellemzői ---	16-17.oldal
.....	4.4 C# (C Sharp) ---	17.oldal
.....	4.4.1 Alapvető jellemzői ---	17-18.oldal
.....	4.5 Entity Framework ---	18.oldal
.....	4.5.1 Fő jellemzői ---	18-19.oldal
.....	4.6 A megvalósítás, kezdetek ---	19-20.oldal
.....	4.6.1 Kontrollerek írása ---	20-21.oldal
5 -	WPF admin felület és karbantartó ---	21-22.oldal
.....	5.1 Fő jellemzői, komponensek ---	22.oldal
.....	5.2 Megvalósítás ---	22-23-24-25.oldal
6 -	Kommunikációs platformok ---	25.oldal
.....	6.1 Discord ---	25.oldal
.....	6.2 Trello ---	25-26.oldal
.....	6.3 Github ---	26-27.oldal
	6.4 Google Drive ---	27.oldal
7 -	Összesítés ---	28.oldal
.....	7.1 Sándor Kristóf ---	28.oldal
.....	7.2 Mecsei Péter ---	28.oldal
8 -	Forráslista ---	29.oldal

1. Bevezetés: Miért webshopot választottunk ?

Azért a webshopot választottunk mert szerintünk egy hétköznapi szoftver a webshop, de mégis megmérettetés számunkra, mivel az elkészítése során széles körben szerezhettünk tapasztalatot mind frontenden, backenden, adatbázisból és egy kicsit belenyúltunk a felhőalapú szolgáltatásokba, Egy webshop készítése során lehetőség nyílik különböző technológiák, eszközök és keretrendszerek alkalmazására, például React, ASP.NET Core, Node.js, MySQL.

A szoftver célja:

Az elsődleges cél egy olyan webshop készítése, amely egyszerű és könnyen kezelhető felhasználói felülettel rendelkezik. Ez a felhasználók számára megkönnyíti a böngészést és a vásárlást.

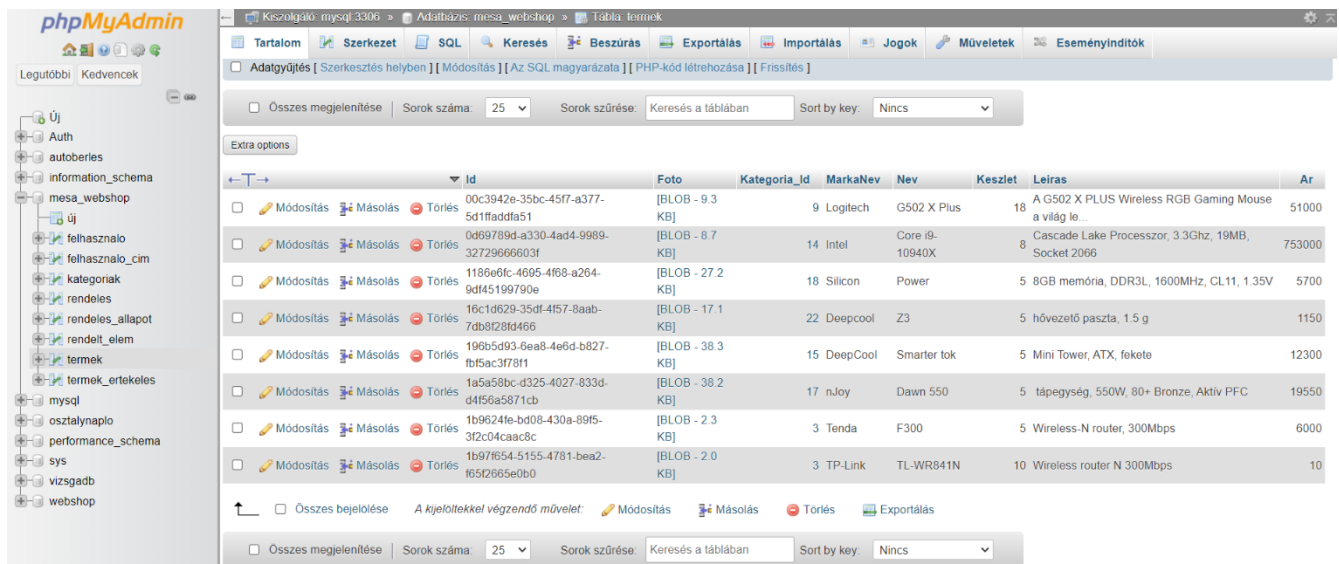
2. Adatbázis és a phpMyAdmin

2.1 Fejlesztői környezet:

Az adatbázisunk létrehozására a **phpMyAdmin** felületét választottuk.

A phpMyAdmin egy olyan webes alkalmazás, amelyet adatbázis kezelésére és adminisztrációjára használnak, különösen **MySQL** és MariaDB adatbázisok esetén.

Ez az alkalmazás lehetővé teszi az adatbázisok létrehozását, módosítását, törlését és karbantartását egy felhasználói felületen keresztül, ami egy böngésző segítségével elérhető. Népszerűségét elsősorban az egyszerű használat és a széles körű funkcionalitás jellemzi.



1. ábra A phpMyAdmin grafikus felülete

2.2 Néhány fő ok amiért hasznos, illetve amik miatt mi is ezt választottuk:

Grafikus felhasználói felület: Egy olyan grafikus felhasználói felületet kínál, ami lehetővé teszi az adatbázisok és azok tábláinak kezelését virtuálisan. Ez főleg azért hasznos mert meggyorsítja a munkánkat azzal, hogy nem kell parancssoros interfészt használnunk.

Adatbázisok karbantartása: Az adatbázisok karbantartása, például az indexek újragenerálása, a táblák optimalizálása vagy az adatok exportálása és importálása is könnyen elvégezhető pár kattintással.

Felhasználók és jogosultságok kezelése: A phpMyAdmin lehetőséget biztosít a felhasználók és a hozzájuk kapcsolódó jogosultságok kezelésére az adatbázisokon belül. Ennek köszönhetően alap szintű ellenőrzést biztosíthatunk az adatbázisokhoz való hozzáférések felett.

Adatbázisok és táblák kezelése: Ennek a felületnek köszönhetően könnyedén hozhatunk létre és módosíthatunk adatbázisokat, azok tábláit, illetve adatokat is hozzáadhatunk és módosíthatunk vagy törölhetünk.

Importálás és exportálás: Az Importálás és Exportálás funkciókkal könnyedén lehet adatokat importálni az adatbázisba, vagy exportálni adatokat más formátumokba, például SQL vagy CSV.

Összességében: A phpMyAdmin egy olyan alkalmazás, amely lehetővé teszi az adatbázisok kezelését egy intuitív webes felületen keresztül. Könnyen használható funkcióival és grafikus felületével segít a felhasználóknak hatékonyan kezelni és karbantartani az adatbázisokat anélkül, hogy közvetlenül parancssoros műveleteket kellene végezniük. Ezek miatt különösen hasznos minden olyan személy vagy szervezet számára, akik MySQL vagy MariaDB adatbázisokat használnak webes alkalmazások vagy projektek adatainak tárolására.

2.3 WampServer:

A WampServer egy ingyenes és nyílt forráskódú szoftvercsomag amelyet a webfejlesztők használnak lokális fejlesztőkörnyezet létrehozására Windows rendszeren. Ez a csomag tartalmazza az Apache HTTP szervert, a MySQL adatbáziskezelőt és a PHP programozási nyelvet. A WampServer lehetővé teszi hogy egyetlen kattintással telepítsd és futtasd ezeket a szerveralkalmazásokat, és könnyen fejlessz dinamikus webalkalmazásokat saját gépeden.



A WampServer tartalmazza a phpMyAdmin-t is, ezért használtuk mi a Wampot. Összességében könnyen üzemeltethető, jól konfigurálható és internetkapcsolat nélkül is használható.

2.4 Az elképzelés:

A tervezés alatt egy olyan adatbázist szerettünk volna létrehozni melynek felépítése segíti az adatbázis tábláiban lévő adatok **jó átláthatóságát** és **kezelhetőségét**. Emellett a

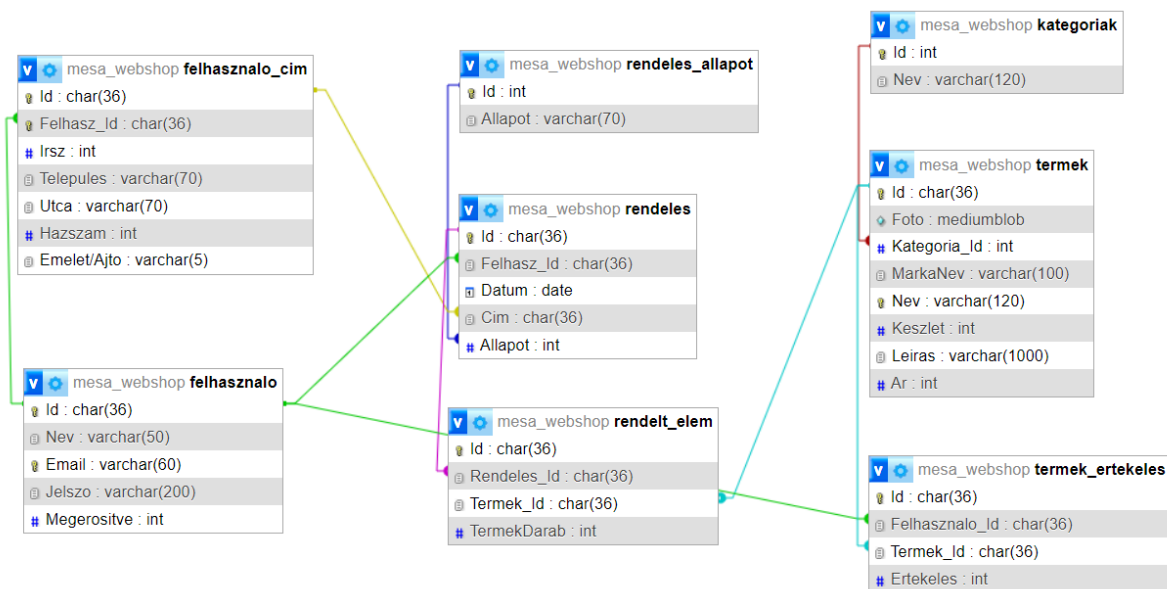
megvalósítás majd a tesztelés során létrejövő hibák, esetleges hiányosságok könnyen orvosolhatóak legyenek.

2.5 A megvalósítás:

A megvalósítás menetében próbáltunk az előző pont alapján létrehozott tervhez ragaszkodni. A terv elég jól sikerült, ugyanis csak apró módosításokat kellett eszközölnünk.

Természetesen a fejlesztés során még adódtak **apróbb gondok**, például hozzákellett adnunk egy plusz mezőt egy táblához, vagy éppen kikellett törölni, de alapjába véve egy nagyon jó adatbázist sikerült létrehozunk.

A táblák **összekötése** is remek lett mivel a backend fejlesztésének menetében egyértelmű volt hogy milyen lekérdezéseket kell alkalmaznunk, azokat hogy kell megvalósítanunk, valamint miként kell feltölteni adatokat bizonyos táblákba.



2. ábra Az adatbázis felépítése, kapcsolatok a táblák közt

A mellékelt ábrán jól látható pontosan az adatbázisunk felépítése, hogy milyen táblákat hoztunk létre és az adott táblák milyen mezőket tartalmaznak.

Igyekeztünk csak a **szükséges** és **fontos** dolgokat eltárolni az adatbázisban, ilyen például hogy minden rekord **egyedi azonosítóval** rendelkezik, ami megakadályozza a rekordok duplikálását, illetve hogy lekérdezésekkor és adatok feltöltésekor ne akadjon össze az adatbázis.

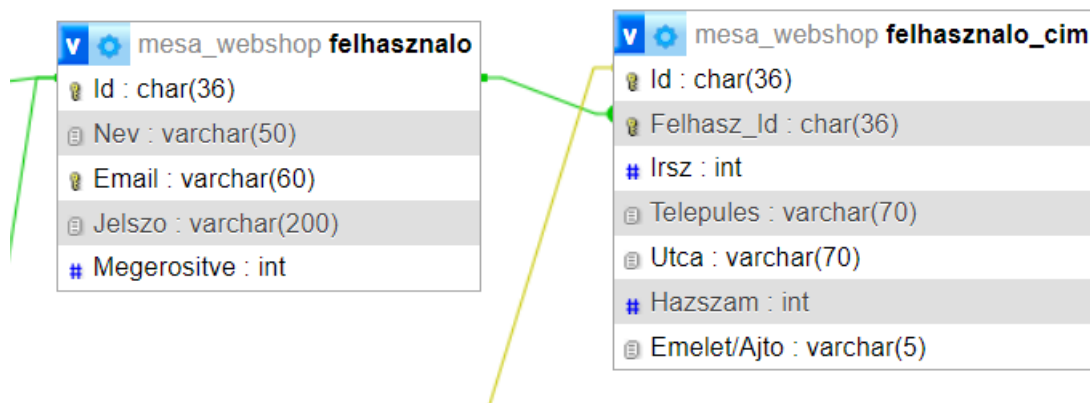
Fontos megjegyezni hogy a **termékeknek** a **képeit** is adatbázisban tároljuk, gondolkodtunk rajta hogy a képeket ftp szerveren tároljuk el, de egyszerűbbnek találtuk így hogy minden egy helyen. Elég volt csak írni pár függvényt arra hogy a képeket megtudjuk jeleníteni frontenden és hogy az admin felületünkről tudjunk hozzáadni vagy módosítani képet az adatbázisban.

Jól látható hogy például a 'felhasznalo' táblában az 'email' mező **egyedi kulccsal** van ellátva. Ez azért jó nekünk mert adatbázis szinten is védelmet nyújt a redundanciával szemben.

2.5.1 Normál forma: Az adatbázisban lévő táblák és a köztük lévő kapcsolatok struktúrája alapján megállapítható hogy az adatbázisunk harmadik normál formában van(3NF). A harmadik normál forma azt foglalja magába hogy **minden kulcs attribútum tranzitív függőségektől mentes** a kulcsokon belül. Ez azt jelenti, hogy nincs olyan nem kulcs attribútum amely más nem kulcs attribútumtól függ.

Emellett ha egy adatbázis a harmadik normál formában van, az azt jelenti, hogy automatikusan megfelel az első és a második normál formáknak is. Az adatbázis normál formái egymásra épülnek, tehát az alacsonyabb rendű normál formák követelményeit is teljesíteni kell ahhoz hogy magasabb normál formában legyen egy adatbázis.

A normál formákra való felosztás a tervetés során segít optimalizálni az adatszerkezetet, minimalizálni a redundanciát és biztosítani az adatintegritást.



3. ábra Példa a normál formára

A "felhasznalo" tábla attribútumai (Nev,Email,Jelszo,Megerositve) között nincs tranzitív függőség. A tranzitív függőség azt jelenti, hogy két nem-kulcs mező függ egymástól és közvetetten függ a tábla elsődleges kulcsától.

A "felhasznalo_cim" táblában a címek az egyedi felhasználókhöz vannak rendelve a "Felhasz_Id" segítségével.

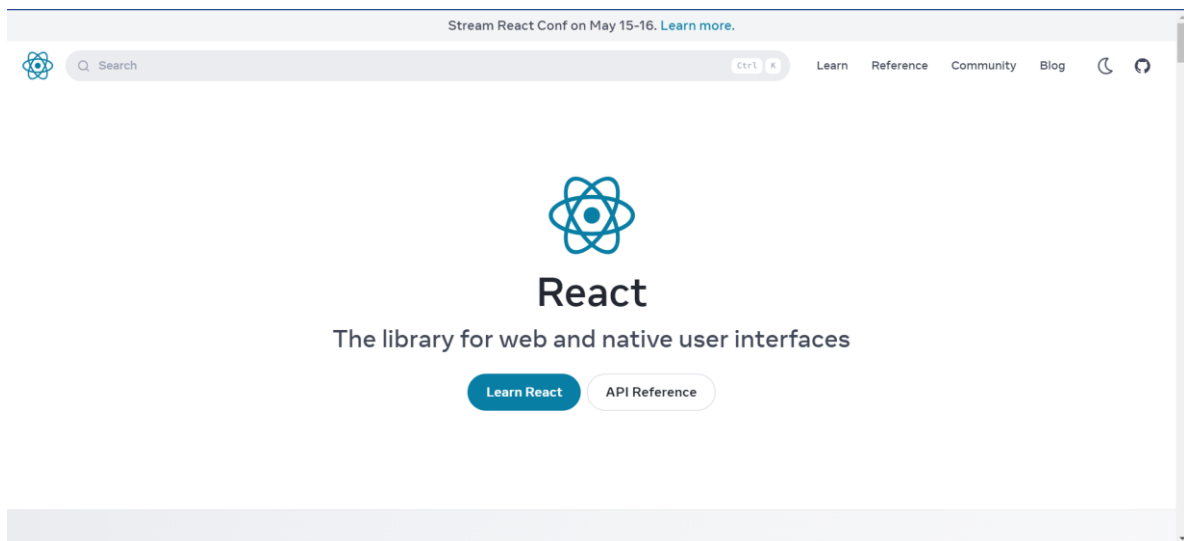
3.Frontend

3.1 React:

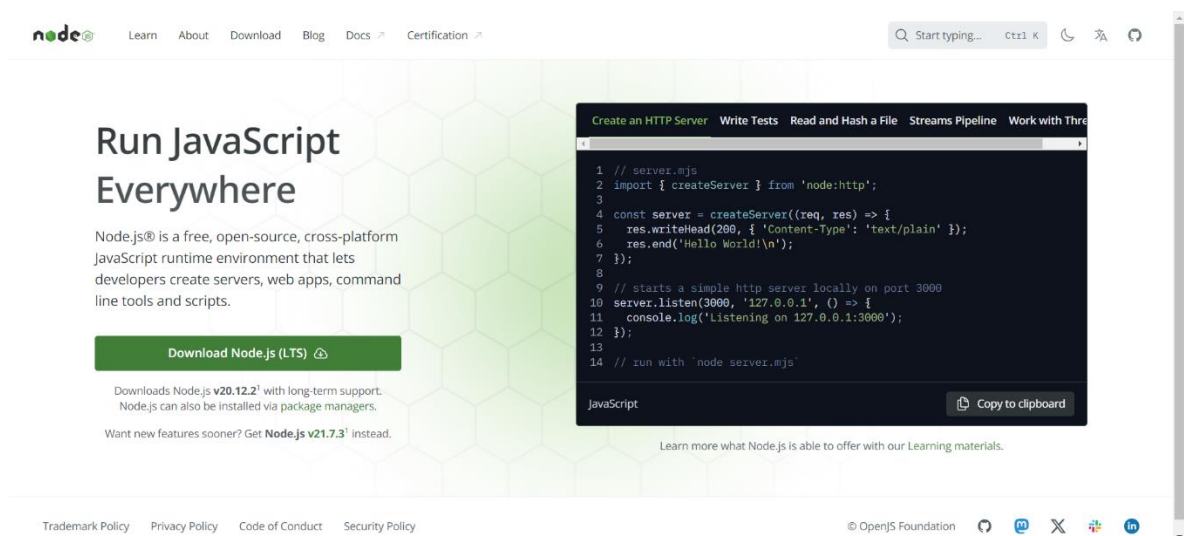
A React egy JavaScript könyvtár a felhasználói felületek készítéséhez. Nagyon népszerű, mert lehetővé teszi az alkalmazások moduláris felépítését.

Fontos a React mert lehetővé teszi számunkra, hogy webalkalmazásokat építsünk, amelyek gyorsak, hatékonyak és könnyen kezelhetők.

A React egy nagyon erős és dinamikus eszköz, amely lehetővé teszi, hogy interaktív webalkalmazásokat hozzunk létre különböző funkciókkal és komponensekkel.



3.2 Node.js:

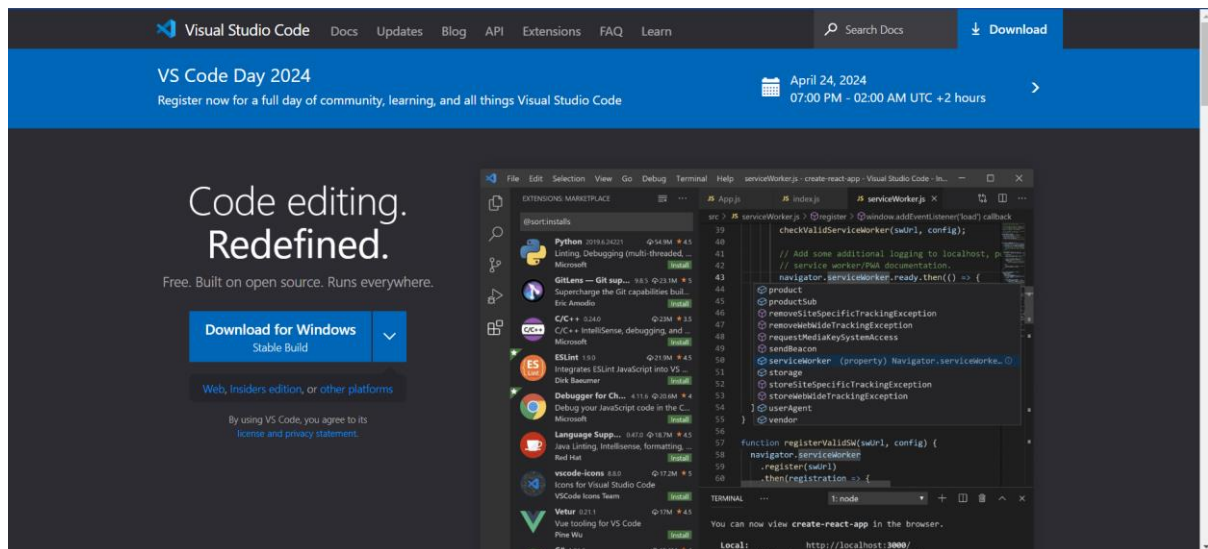


A **Node.js** egy JavaScript futtató környezet, amely lehetővé teszi a JavaScript kód futtatását a szerveroldalon. Ez azt jelenti, hogy lehetővé teszi számunkra, hogy szerveroldali alkalmazásokat írjunk JavaScriptben.

A Node.js mert lehetővé teszi számunkra, hogy teljes körű webalkalmazásokat hozzunk létre, amelyek mind a kliensoldalon, mind a szerveroldalon JavaScriptben vannak írva. Ez egyszerűsíti az alkalmazásfejlesztést és javítja a hatékonyságot.

A Node.js lehetővé teszi számunkra, hogy egyetlen nyelvet használjunk mind a kliens-, mind a szerveroldalon, ami sok fejlesztőnek kényelmes és hatékony megoldás.

3.3 Visual Studio Code:



Visual Studio Code egy ingyenes, nyílt forráskódú kódszerkesztő és fejlesztői környezet, amelyet a Microsoft fejlesztett ki. Ez egy nagyon népszerű eszköz a fejlesztők körében a web- és szoftverfejlesztéshez.

A Visual Studio Code egy könnyen használható és testreszabható kódszerkesztő, amely számos funkciót és bővítményt kínál a hatékonyabb és kényelmesebb fejlesztés érdekében.

A VS Code lehetővé teszi számunkra, hogy könnyedén írjunk, szerkesszünk és teszteljünk kódot különböző programozási nyelvekhez, beleértve a JavaScriptet és a Node.js-t. Emellett számos hasznos funkciót kínál, például kódszerkesztési segítséget, kiterjesztéseket és integrációt más fejlesztői eszközökkel.

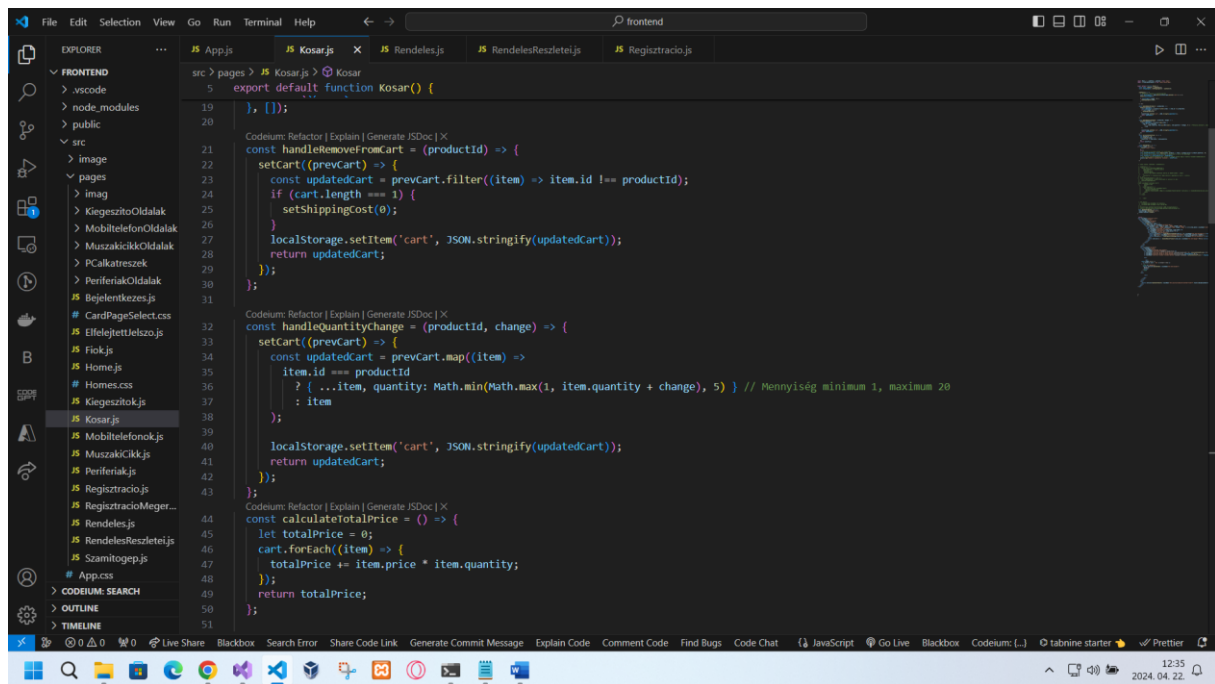
3.4 JavaScript:

A JavaScript egy programozási nyelv, amelyet általában a webfejlesztéshez használnak. A weboldalak interaktivitásának és dinamizmusának létrehozására szolgál.

JavaScript az egyik legelterjedtebb és legfontosabb nyelv a webfejlesztésben. Ez lehetővé teszi a weboldalak interaktív elemekkel való bővítését, mint például űrlapok, animációk, adatmanipuláció stb.

Mivel JavaScript nagyon sokoldalú, és használható kliensoldali és szerveroldali fejlesztésre is. Ezen felül számos keretrendszer és könyvtár létezik, mint például React, Angular vagy Vue.js, amelyek segítségével hatékonyabban lehet fejleszteni.

JavaScriptet a böngészőkben, webes alkalmazásokban, szerveroldali alkalmazásokban, mobilalkalmazásokban és még sok más helyen használják. Sokféle területen alkalmazzák, például webfejlesztés, játékfejlesztés, asztali alkalmazások stb.



```
export default function Kosar() {  
  // ...  
  const handleRemoveFromCart = (productId) => {  
    setCart((prevCart) => {  
      const updatedCart = prevCart.filter(item => item.id !== productId);  
      if (cart.length === 1) {  
        setShippingCost(0);  
      }  
      localStorage.setItem('cart', JSON.stringify(updatedCart));  
      return updatedCart;  
    });  
  };  
  
  const handleQuantityChange = (productId, change) => {  
    setCart((prevCart) => {  
      const updatedCart = prevCart.map(item => {  
        item.id === productId  
        ? { ...item, quantity: Math.min(Math.max(1, item.quantity + change), 5) } // Mennyiség minimum 1, maximum 20  
        : item  
      });  
      localStorage.setItem('cart', JSON.stringify(updatedCart));  
      return updatedCart;  
    });  
  };  
  
  const calculateTotalPrice = () => {  
    let totalPrice = 0;  
    cart.forEach(item => {  
      totalPrice += item.price * item.quantity;  
    });  
    return totalPrice;  
  };  
};
```

4. ábra Példa részlet a JavaScript-re

3.5 HTML,CSS:



A HTML (Hypertext Markup Language) egy olyan nyelv, amelyet a weboldalak strukturális felépítéséhez és tartalmának formázásához használnak. Ez jelenti azt, hogy az HTML kód meghatározza az oldal címeit, szövegét, képeit, linkeket és egyéb tartalmi elemeket.

Ez az alapja minden weboldalnak. Az HTML kód strukturálja az oldalt és meghatározza, hogy a tartalom hogyan jelenik meg a böngészőben.

Mivel az HTML könnyen tanulható és használható. Az alapvető címkék és attribútumok gyorsan elsajátíthatók, és azonnal láthatók az eredmények a böngészőben.

A HTML-t mindenhol használják, ahol weboldalakat készítenek. Ez magában foglalja a személyes weboldalakat, vállalati weboldalakat, blogokat, online áruházakat és sok más típusú weboldalt.



A **CSS** egy stílusleíró nyelv, amelyet a weboldalak megjelenésének és elrendezésének meghatározására használnak. Azaz, az CSS kód meghatározza, hogy az HTML elemek hogyan jelennek meg a böngészőben, például a színek, betűtípusok, elrendezések, stb.

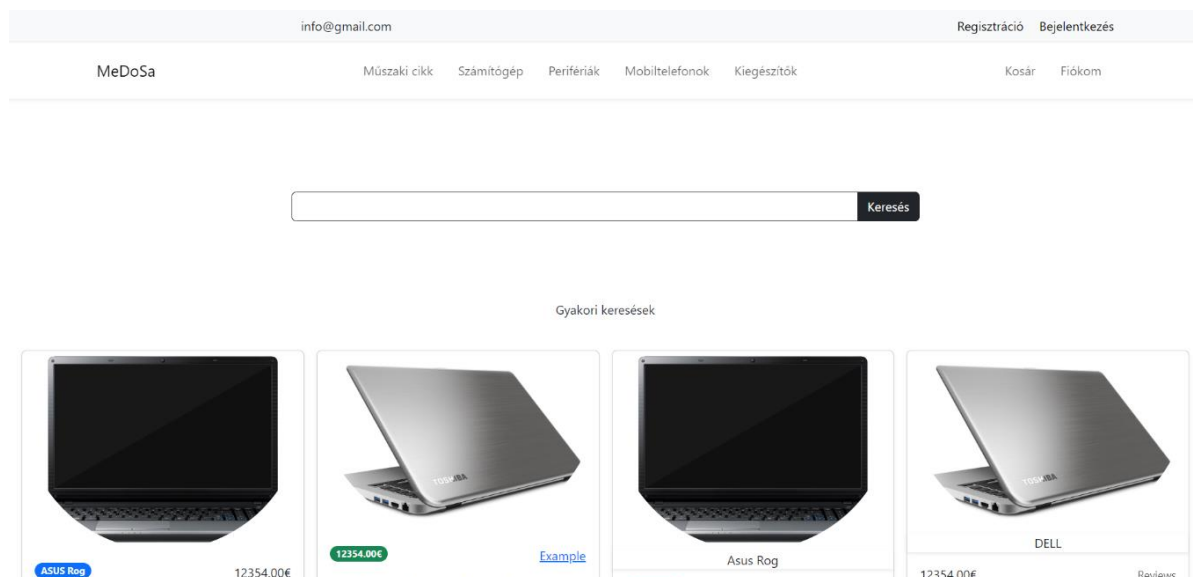
Lehetővé teszi, hogy szépen megjelenő és felhasználóbarát weboldalakat készítsünk. A CSS segítségével testreszabhatjuk az oldal megjelenését és elrendezését az egyedi igények szerint.

Mivel a CSS-t könnyen tanulható és alkalmazható. Az alapvető stílus tulajdonságok, mint például a háttérszín vagy a betűtípus, könnyen módosíthatók, és azonnal láthatók az eredmények a böngészőben.

A CSS-t mi nem egy külön mappában írtuk meg hanem mivel bootstrapp-et használtunk és csak a speciális esetekben volt szükség a CSS-re így a html-be implementáltuk.

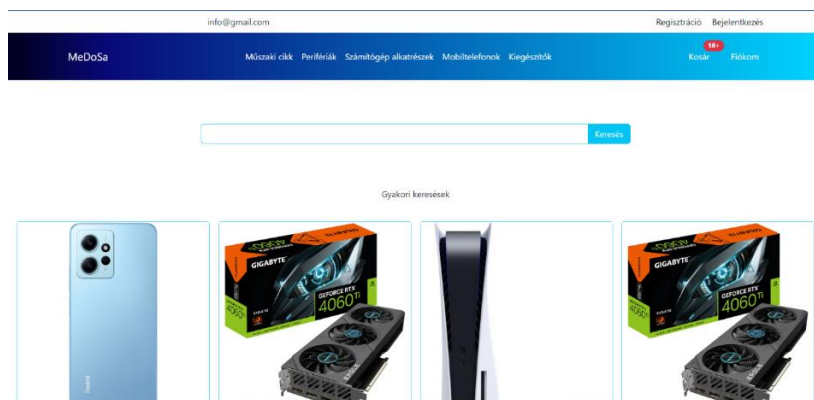
3.6 Kezdeti tervek

Első gondolatunk az volt hogy a webshop-on lehessen új illetve használt termékeket vásárolni de ezt az ötletet elvetettük. Webshopunk törekszik arra hogy a termékek minél szélesebb réteget öleljen át. Miután alaposan átgondoltuk, úgy döntöttünk, hogy webshopunk más irányba kanyarodik. Célunk az, hogy a termékválasztékunk minél szélesebb réteget öleljen át, és olyan kategóriákban kínáljunk termékeket, amelyek széleskörű érdeklődést kelthetnek. Így aztán a vásárlók számára sokféle lehetőséget kínálunk, és mindenki megtalálhatja a számára legmegfelelőbb termékeket a webáruházunkban. A célunk az, hogy a felhasználók könnyedén és gyorsan megtalálják, amit keresnek, és hogy minőségi, változatos termékek közül válogathassanak. Ezáltal célunk, hogy webshopunk egy igazi bevásárlóparadicsommá váljon, amelyet szívesen látogatnak az emberek, és ahol mindig megtalálják a legfrissebb és legkülönlegesebb termékeket.



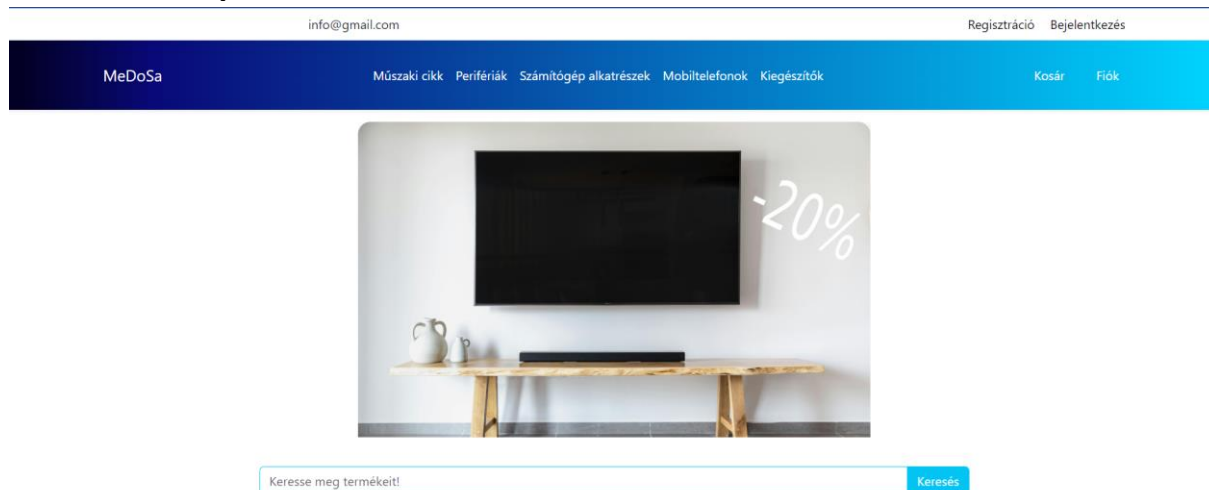
5. ábra Kezdeti design

Sok kritikákat kaptunk azzal kapcsolatban hogy nagyon dominál a fehér, illetve azon kívül csak fekete látható az oldalon . Szerettünk volna színeesebb illetve több dolgot elhelyezni a főoldalra de mivel szerintünk a kevesebb néha több ezért leginkább a színezést szerettük volna megoldani



6. ábra Véglegesített design

3.7 A webshop részletes bemutatása:



7. ábra A főoldal

A főoldalon található egy képekből álló ablak ami egy bizonyos időközönként cseréli magát azokra a képekre amiket mi megadtunk neki. A kezdőlapen regisztráció illetve ha van már fiókja akkor bejelentkezés is elérhető.

Regisztráció

Név

Név

Email cím

Email cím

Jelszó

Jelszó megerősítése

Regisztráció

Fiók megerősítése

8. ábra Regisztráció oldal

A regisztráció oldalon név, email, jelszót kell megadni illetve ha már volt regisztráció de véletlenül elkattintott akkor a regisztráció gomb mellett látható egy „Fiók megerősítés” gomb amire ha rákattint szintén megerősítheti a regisztrációt.

3.7.1 Regisztráció kódrészlet:

Itt látható hogy ha a felhasználó regisztrálni szeretne akkor minden mezőt köteles kitölteni illetve írni fogja ha a jelszó nem egyezik. Több módon is megvalósítható lenne a jelszó ellenőrzése, lehetne hogy folyamatosan jelezné a nem egyezést amíg azt helyesen nem írja be a mezőbe. De mi arra jutottunk hogy ez sokkal megterhelőbb egy felhasználó számára így az egyszerűbb megoldást alkalmaztuk akkor fogja leellenőrizni a jelszót amikor a regisztráció gombra kattintunk.

```

4  const Regisztracio = () => {
14  const handleConfirmPasswordChange = (e) => {
16    setConfirmPassword(value);
17  };
18
19  Codeium: Refactor | Explain | Generate | JSDoc | X
20  const handleSubmit = async (e) => {
21    e.preventDefault();
22
23    // Ellenőrzés, hogy minden mező ki legyen töltve és a jelszavak egyezzenek
24    let formIsValid = true;
25    if (!name) {
26      setNameError('A név megadása kötelező');
27      formIsValid = false;
28    } else {
29      setNameError('');
30    }
31
32    if (!email) {
33      setEmailError('Az email cím megadása kötelező');
34      formIsValid = false;
35    } else {
36      setEmailError('');
37    }
38
39    if (!password) {
40      setPasswordError('A jelszó megadása kötelező');
41      formIsValid = false;
42    } else {
43      setPasswordError('');
44    }
45
46    if (!confirmPassword) {
47      setConfirmPasswordError('A jelszó megerősítése kötelező');
48      formIsValid = false;
49    } else {
50      setConfirmPasswordError('');
51    }
52
53    if (password !== confirmPassword) {
54      setConfirmPasswordError('A jelszavak nem egyeznek');
55      return; // Ne folytassuk a további feldolgozást
56    } else {
57      setConfirmPasswordError(''); // Ha egyezik, töröljük az előző hibahüvelyt
58    }
59  }
60 }

```

9. ábra A regisztráció kódja

3.7.2 Bejelentkezés kód részlet:

```
useEffect(() => {  
  // Ellenőrizze, hogy van-e felhasználói adat a localStorage-ban  
  const loggedInUser = localStorage.getItem('user');  
  if (loggedInUser) {  
    // Ha van felhasználói adat, akkor beállítjuk a userLoggedIn állapotot true-ra  
    setUserLoggedIn(true);  
  }  
}, []);
```

```
Codeium: Refactor | Explain | Generate JSDoc | ×  
const handleLogin = async () => {  
  // Ellenőrzés, hogy minden mező ki legyen töltve  
  let formIsValid = true;  
  
  if (!email) {  
    setEmailError('Az email cím megadása kötelező');  
    formIsValid = false;  
  } else {  
    setEmailError('');  
  }  
  
  if (!password) {  
    setPasswordError('A jelszó megadása kötelező');  
    formIsValid = false;  
  } else {  
    setPasswordError('');  
  }  
  
  if (formIsValid) {  
    try {  
      console.log('Bejelentkezés...');  
  
      const response = await fetch('http://localhost:5259/User/login', {  
        method: 'POST',  
        headers: {  
          'Content-Type': 'application/json',  
        },  
        body: JSON.stringify({ email, password }),  
      });  
  
      if (response.ok) {  
        const userData = await response.json();  
        localStorage.setItem('user', userData.name);  
        alert('Sikeres bejelentkezés');  
        console.log('Sikeres bejelentkezés');  
        console.log();  
        setUserLoggedIn(true);  
        window.location.href = '/Fiok';  
      } else {  
        console.log('Sikertelen bejelentkezés');  
        setPasswordError('Hibás jelszó vagy email');  
      }  
    } catch (error) {  
      console.error('Hiba történt a bejelentkezés során:', error);  
    }  
  }  
};
```

10. ábra A bejelentkezés kódja

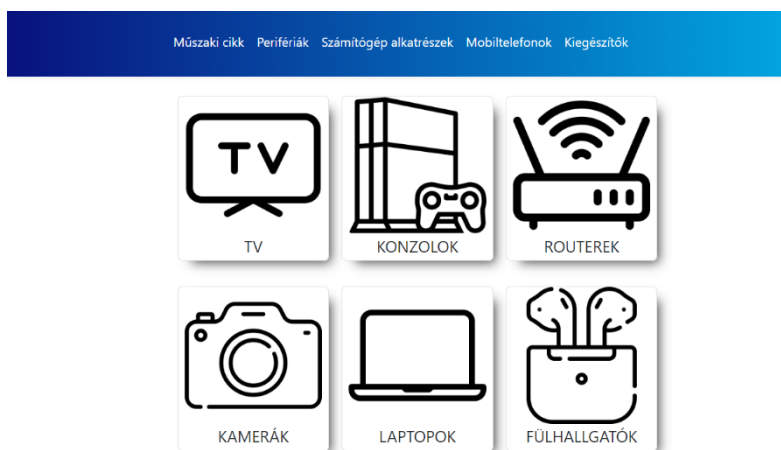
A kódban jól látható hogy a bejelentkezést sokkal egyszerűbb volt megvalósítani. Ez elküldi a backendnek a bejelentkezési adatokat(email cím és jelszó) és leellenőrzi létezik-e felhasználó ezzel email címmel és hogy egyezik e a megadott jelszó az adatbázisban eltároltal. Ha sikeres a bejelentkezés akkor a „Fiók” oldalra navigál minket.

Sok bejelentkezés formát próbáltunk ki de mi ezt a koncepciót választottuk mivel sok weboldal ezeket a módszereket használja illetve szerintünk ez a leg-praktikusabb is egyben.

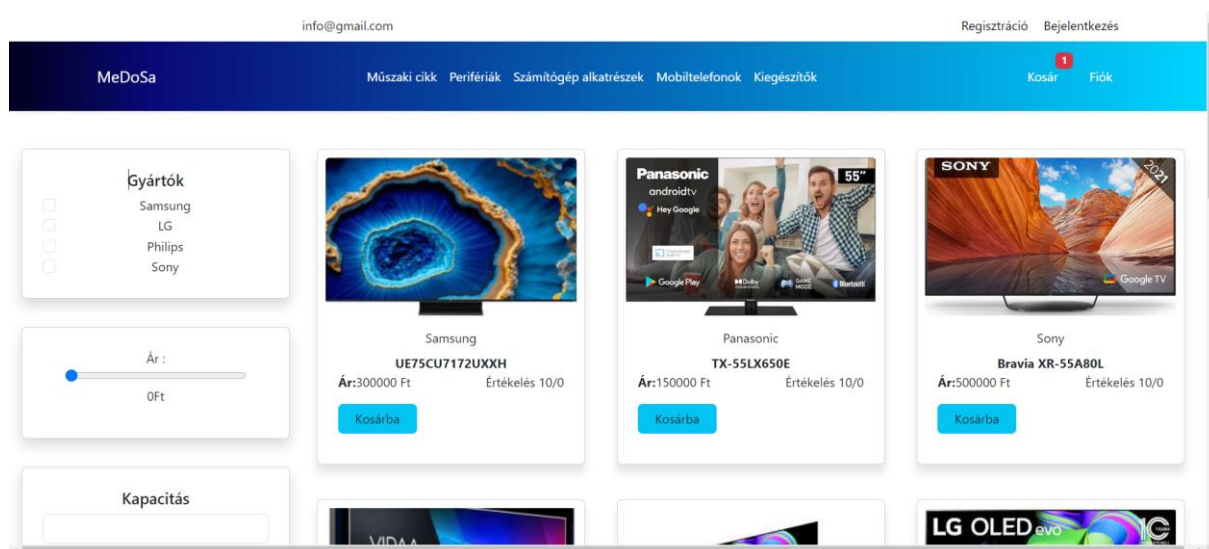
3.7.3 Navigálás a termékek között:

A pl.(„Műszaki cikk”)-re kattintva találunk 6 kis kártyát amik közül választhatunk hogy melyik termékek között szeretnénk keresni.

Szerettünk volna egy egyszerű oldalt ahol nagyobb kategóriákba szedve sokkal átláthatóbb legyen a termékek megtalálása.



11. ábra Kategória választó

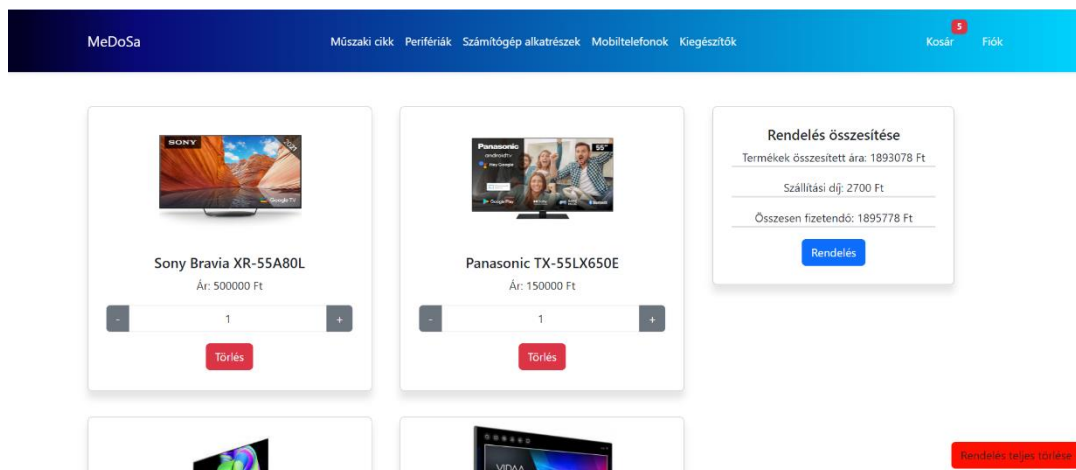


12. ábra Televíziók oldal

Itt látható hogy a Tv kártyára kattintva megtekinthető annak termékei.

Az oldal baloldalán látható a szűrések. Gyártó, Ár és más szűrő is megtalálható.

3.7.4 Kosár oldal:



13. ábra A kosár oldal

A „Kosár” gombon jól látható hogy mutatja mennyi terméket raktunk bele a kosárba.

Amikor a gombra kattintunk akkor megtekinthető annak tartalma, termék ára, illetve szállítási árral mennyibe kerül.

Az oldal jobb alsó sarkában található egy gomb ami az összes terméket kitörli.

A termékeknek a mennyiségét lehet módosítani.

Ha a „Rendelés” gombra kattintva átdob egy másik oldalra

14. ábra Rendelés oldal

Itt megtudjuk adni az adatainkat. Miután megadtuk az adatokat a „Tovább” gombbal megtekinthetjük hogy helyesek az adatok amiket megadtunk. Ezután már egy összegzés lehet látni ami után ha leellenőriztük az adatokat utána már „Vásárlás” gombot látunk amit megnyomva az adatok felmennek az adatbázisba. Ez egy hosszú folyamat de szerintünk ez egy biztonságos módja a helyes adatok megadására.

3.7.5 Keresés:

A keresés funkció segítségével a felhasználók gyorsan eljuthatnak az általuk keresett termékekhez, anélkül, hogy több oldalt kellene átböngészniük. Ez időt takarít meg és javítja a

felhasználói élményt. A keresés mezőbe kattintva a felhasználó kereshet gyártó szerint illetve termék név szerint.



15. ábra Leakciózott termékek egy slider-ben

4. Backend

4.1 WebAPI

A **WebAPI (Web Application Programming Interface)** egy olyan programozási interfész, amely lehetővé teszi a különböző szoftveralkalmazások közötti kommunikációt a világhálón keresztül.

Gyakran használják alkalmazások közötti adatcserére és kommunikációra. A WebAPI-k általában **HTTP (Hypertext Transfer Protocol) protokollt** használnak üzenetek küldésére és fogadására.

4.1.1 Fontos tulajdonságai, amiért választottuk:

Kommunikáció kliens és szerver között: A WebAPI-k lehetővé teszik a kliensalkalmazások számára, hogy kéréseket küldjenek a szervernek és válaszokat kapjanak. Ez lehetővé teszi az alkalmazások közti kommunikációt.

RESTful szolgáltatások: Sok WebAPI implementálja a **REST (Representational State Transfer)** architektúrát, ami egy olyan tervezési stílus amely egyszerű könnyen és jól érthető interfészt biztosít az alkalmazások közti kommunikációhoz. A RESTful API-k erőforrásokat kezelnek, amelyeket **egyedi URI**-kon keresztül érhetünk el, különböző HTTP módszerekkel (**GET, POST, PUT, DELETE**).

Adatcsere formátumok: A WebAPI-k támogatják a különböző adatcsere formátumokat, például **JSON (JavaScript Object Notation)** vagy **XML (Extensible Markup Language)**. Ezek a formátumok lehetővé teszik az adatok strukturált és egyszerű átvitelét alkalmazások közt.

Platformfüggetlenség: Mivel a HTTP-t használja alapvető kommunikációs protokollként, a WebAPI-k platformfüggetlenek, mivel szinte minden modern alkalmazás támogatja a

HTTP-t. Ez azt jelenti, hogy az alkalmazások különböző platformokon ,például webböngészőkön, mobilalkalmazásokban, asztali alkalmazásokban, képesek kommunikálni egymással.

4.1.2 Amik szintén WebAPI-t használnak:

Szociális hálózatok, például Facebook vagy Twitter, API-ja lehetővé teszi az alkalmazások számára, hogy posztokat, felhasználói adatokat stb. küldjenek, módosítsanak vagy lekérjenek.

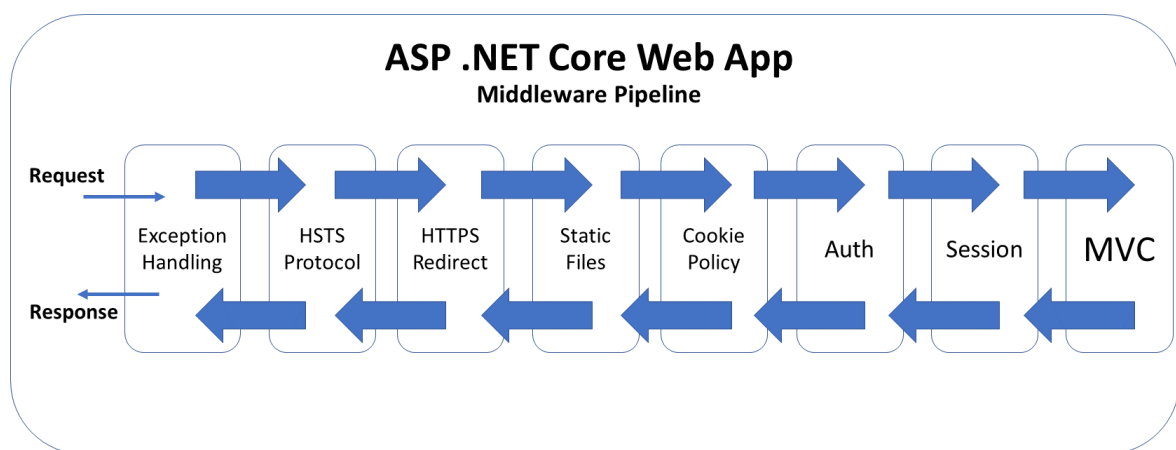
Pénzügyi szolgáltatók, például bankszámla kezelési rendszerek, API-ja a tranzakciók végrehajtását, egyenlegek lekérdezését teszi lehetővé.

Időjárás alkalmazások, például BBC Weather, az időjárással kapcsolatos adatokat (időkép, hőmérséklet, szél erőssége stb.) képesek lekérdezni különböző helyszínekre.

Ezek mellett van még egy fontos szempont ami miatt WebAPI-t használtunk, ez pedig a mi fejlesztőkörnyezetünkben elérhető **ASP.NET Core**-t használó WebAPI.

4.2 ASP.NET Core

Az ASP.NET Core egy **nyílt forráskódú, cross-platform** keretrendszer amelyet **webalkalmazások** és szolgáltatások fejlesztésére használnak. Az ASP.NET Core-t a Microsoft fejleszti és a .NET Core keretrendszer része. Ez a **keretrendszer** lehetővé teszi a modern és hatékony alkalmazások létrehozását különböző környezetekben, Windows, macOS és Linux rendszereken. **Modern**, könnyűsúlyú és **teljesítményorientált** keretrendszer, amely széles körű felhasználási lehetőséget kínál.



16. ábra Az ASP.NET Core működési ábrája

4.2.1 Főbb jellemzői:

Cross-platform támogatás: Teljes mértékben cross-platform, ami azt jelenti hogy képes futni Windows, macOS és Linux rendszeren is. Ez lehetővé teszi a fejlesztők számára, hogy webalkalmazásokat üzemeltethessenek és hozzanak részre bármilyen operációs rendszeren.

Modularitás: Az ASP.NET Core egy moduláris keretrendszer, ez azt jelenti hogy a fejlesztők csak azokat a komponenseket használhatják amelyekre szükségük van. Ez javítja a teljesítményt és csökkenti az alkalmazás méretét.

Nyílt forráskód: A teljesen nyílt forráskód lehetővé teszi a közösség számára, hogy hozzájáruljon a fejlesztéshez és a javításokhoz. Emellett ez növeli az átláthatóságot és az alkalmazások függetlenségét.

Korszerű architektúra: Támogatja a modern fejlesztési elveket, például az **MVC (Model-View-Controller)** tervezési mintát, valamint könnyen integrálható a fejlett fejlesztői eszközökkel, mint a Visual Studio IDE.

Teljesítményorientált: Optimalizált és hatékony teljesítményt nyújt a nagy terhelésű webalkalmazások számára.

Kompatibilitás: Lehetővé teszi a korábbi ASP.NET alkalmazások migrálását és frissítését a legújabb verzióra, miközben a régebbi technológiákat is támogatja.

4.2.2 Főbb felhasználási területei:

Webalkalmazások és weboldalak fejlesztése, például kereskedelmi vagy adminisztrációs felületek.

API-k készítése és üzemeltetése, amelyeket más alkalmazások használnak, mint a mi projektünk esetében is.

Valós idejű alkalmazások (real-time apps) készítése, például chat alkalmazások és élő közvetítések.

4.3 Microsoft Visual Studio 2022

A Visual Studio egy teljeskörű fejlesztői környezet (IDE), amelyet a Microsoft fejlesztett ki és tart karban. Ez a verzió a Visual Studio család legfrissebb kiadása, amely kifejezetten a legújabb technológiák és platformok támogatására, illetve a fejlesztői termelékenység növelésére összpontosít. Egyéni fejlesztésre, kisebb csapatoknak vagy akár nagy vállalatoknak is teljesen alkalmas felhasználásra mert széles körben fed le alkalmazási területeket, beleértve asztali, mobil, - és webalkalmazásokat.



4.3.1 Néhány főbb jellemző:

Teljes körű .NET támogatás: Lehetővé teszi .NET 6-tól felfelé minden .NET verzióban alkalmazások fejlesztését és kezelését. Mi a 7es .NET verziót hasznátuk.

Cross platform fejlesztés: Támogatja a fejlesztést Windows macOS és Linux rendszereken is. A .NET és az ASP.NET Core projektek teljes körűen fejleszthetőek és futtathatók ezen platformokon.

Teljesítmény optimalizálás: Az IDE javított teljesítményt kínál a nagyobb projektek és szolgáltatások számára. Az optimalizált építési folyamatok és a fejlesztői élmény javítása segíti a gyorsabb és hatékonyabb munkavégzést.

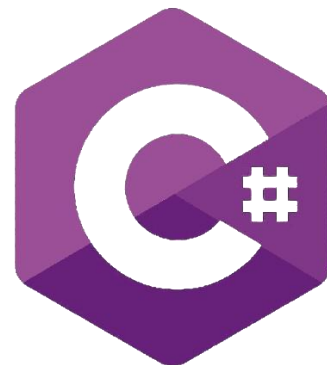
Modern fejlesztői eszközök: Számos modern eszközt és funkciót kínál, például IntelliCode, Live Share együttműködés, korszerű kód szerkesztési lehetőségek, valamint Azure integráció a felhőalapú fejlesztés támogatására.

Adatközpontú és AI alapú fejlesztői eszközök: Integrálja az adatközpontú elemzéseket és telemetriát, valamint számos AI alapú eszközt, amelyek segítik a kódszerkesztést, hibakeresést és optimalizálást.

Könnyű telepítés és frissítések: Könnyen telepíthető, frissíthető és lehetőséget biztosít a fejlesztők számára a testreszabott telepítési beállítások kiválasztásával és az új verziók gyors bevezetésére.

4.4 C# (C Sharp)

A C# egy modern, **objektumorientált** programozási nyelv, amelyet a Microsoft fejlesztett ki a .NET keretrendszerhez. Ezt a nyelvet úgy tervezték hogy hatékonyan és biztonságosan lehessen vele alkalmazásokat fejleszteni különböző platformokra. Jól használható asztali, mobil és webalkalmazások fejlesztésére, emellett videójátékok és számos egyéb alkalmazás fejlesztésére. A C# egy **erőteljes** és **sokoldalú** nyelv, amelyet széles körben használnak a professzionális szoftverfejlesztésben.



4.4.1 Alapvető jellemzői:

Objektumorientált programozás: A C# teljes mértékben objektumorientált nyelv, ami azt jelenti, hogy minden program egy vagy több osztályból áll, és az osztályok példányai között történik az adatok és műveletek cseréje. Az öröklődés, az absztrakció és a polimorfizmus alapvető fogalmak a C#-ban.

Erősen típusos nyelv: Az erősen típusosság azt jelenti, hogy a változók típusát előre kell deklarálni, majd a fordító ellenőrizni a típusmegfelelést a fordítási időben. Ez segít a hibák korai felfedezésében és a program stabilitásának növelésében és biztosításában.

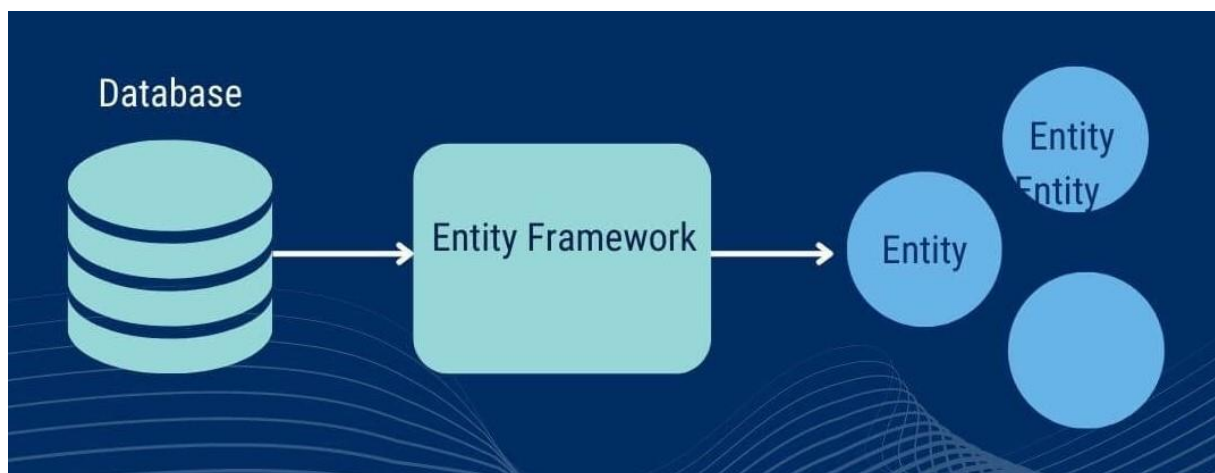
Modern nyelvi elemek: A C# folyamatosan fejlődik, és új nyelvi elemekkel gazdagodik minden új verzióban. Ezekre példák például: **lambda** kifejezések, **LINQ (Language Integrated Query)**, aszinkron programozás (**async/await**), dekonstruktorok stb.

.NET keretrendszer: Mivel a C# alapvetően a .NET keretrendszeren alapszik, amely számos előre elkészített osztályt, függvényt, és komponenst kínál az általános feladatokhoz. Ezért jól használható például felhasználói felületek készítéséhez, hálózati kommunikációhoz, adatbázis-kezeléshez és sok más feladathoz.

Nagy közösség és dokumentáció: Ez a nyelv rendelkezik egy aktív fejlesztői közösséggel, és gazdag dokumentációval rendelkezik a Microsoft által. Ez segít a fejlesztőknek a gyors problémamegoldásban és a fejlesztési folyamatok támogatásában.

4.5 Entity Framework (EF):

Az Entity Framework egy kényelmes és erőteljes adatelérési technológia és **ORM (Object Relational Mapping)** keretrendszer a .NET platformhoz. Ez lehetővé teszi az alkalmazások számára hogy könnyedén kommunikáljanak adatbázisokkal, **objektumorientált** módon. Az EF segítségével a fejlesztőknek nem kell SQL parancsokat használniuk az adatbázis eléréséhez, helyette az alkalmazások objektumokkal dolgozhatnak, amelyek közvetlenül tükrözik az adatbázis tábláit és azok kapcsolatait. Használata javítja a kód olvashatóságát és a termelékenységet, ezért ideális választás a .NET alkalmazásokhoz, függetlenül attól, hogy milyen platformra fejlesztünk alkalmazást (asztali, mobil, web).



17. ábra Az Entity Framework működési ábrája

4.5.1 Fő jellemzői:

Object Relational Mapping: Az **objektum relációs leképezés**, az adatbázis tábláit és relációit objektumokká alakítja a .NET alkalmazásokban. Ez lehetővé teszi az adatokhoz való hozzáférést és kezelést objektumorientált módon.

LINQ támogatás: Az EF lehetővé teszi a LINQ lekérdezések használatát az adatbázishoz való hozzáféréshez. Ennek segítségével az adatokat objektumokként kezelhetjük és

manipulálhatjuk, ami egyszerűbb és kényelmesebb a hagyományos SQL lekérdezésekhez képest.

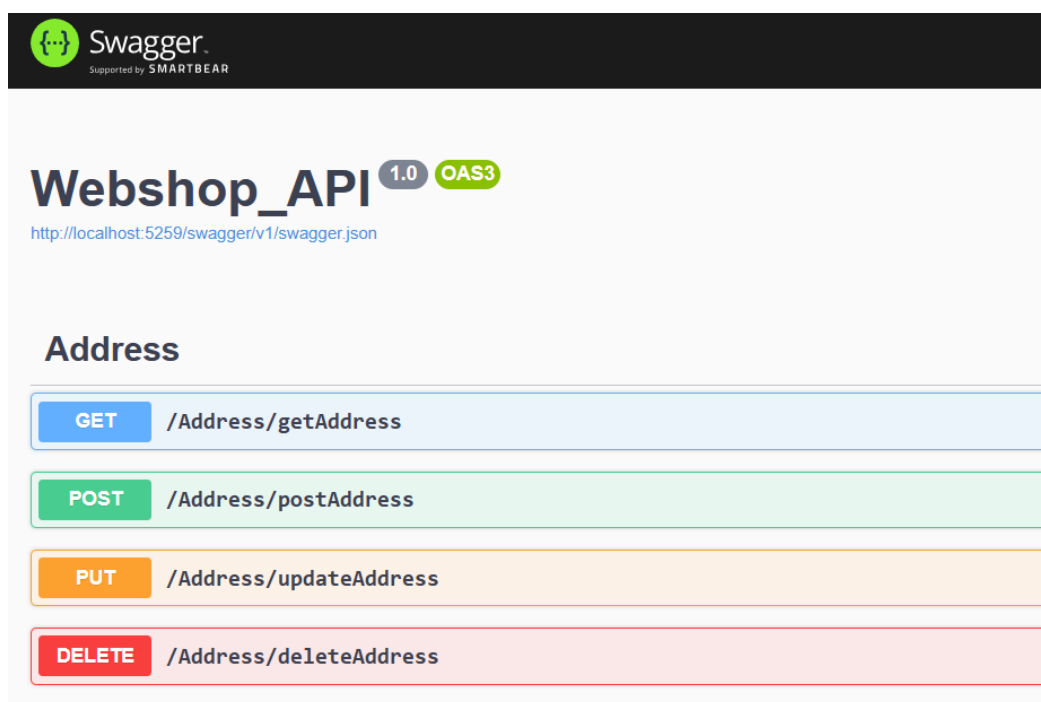
Adatbázisfüggetlenség: Absztrakt réteget biztosít az alkalmazás és az adatbázis között, így teszi lehetővé az alkalmazások számára hogy függetlenek legyenek az adott adatbázis motorral, például MySQL vagy PostgreSQL, is az adatbázis sémájával szemben.

Automatikus adatbázis migráció: Az Entity Framework lehetővé teszi az adatbázis sémaváltozások követését és az adatbázis automatikus migrálását az alkalmazás verzióváltásaihoz igazítva. Ez egyszerűsíti az adatbázis verziókezelését és frissítését.

Könnyű tesztelhetőség és karbantarthatóság: Az EF segítségével könnyen tesztelhetők az adatbázisfüggő alkalmazások, mivel a valós adatbázis helyett emulált entitásokkal dolgozhatunk a tesztekben. Emellett az Entity Framework segít az alkalmazások karbantartásában és a kód újrafelhasználásában is.

4.6 A megvalósítás, kezdetek

Először is létrehoztunk egy WebAPI projektet a Visual Studio 2022-ben. Azért döntöttünk WebAPI mellett mert nem tartalmaz olyan grafikus felületet amelyet nekünk kellett volna megírni, ezt a funkciót majd a frontend szolgáltatja, helyette egy **Swagger** nevű automatikusan generált dokumentáló és leíró formátum van beépítve. Erre a végpontok és a különböző **DTO-k (Data Transfer Object)** tesztelése miatt volt szükség, mert így nem kellett bármi külső programot, például Postman, használnunk.

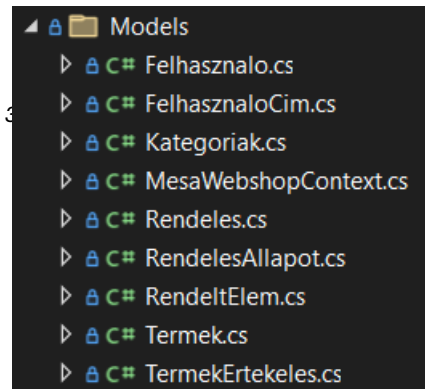


18. ábra A Swagger kinézete

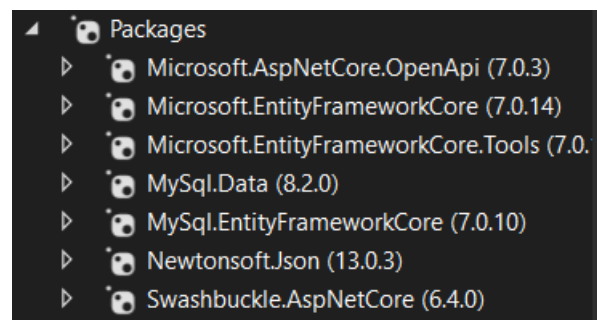
A következő lépésben úgynevezett **NuGet Package**eket kellett feltelepíteni a projekthez, ezek ahhoz kellenek hogy el tudjuk érni az adatbázist, és a **'scaffold-dbcontext'** paranccsal modelleket tudunk generálni az adatbázis sémájára. Ez kulcs fontosságú hiszen ezek nélkül nem tudnánk dolgozni az adatbázissal.

A **'scaffold-dbcontext'**-et az Entity Framework tartalmazza. Ahhoz hogy működjön, nem csak a telepített **NuGet Package**ekre van szükség, hanem definiálnunk kell egy **ConnectionString**et is, aminek a segítségével fog az Entity Framework hozzáférni az adatbázishoz.

A parancs, a **Package Manager Console**ban való sikeres futtatás után, létrehozza nekünk a **'Models'** nevű könyvtárat. Ez a mappa tartalmazni fogja az adatbázis tábláinak sémájára létrehozott objektumokat. Emellett tartalmazni fog még egy dbcontext állományt, ez a mi esetünkben a **'MesaWebshopContext'**, ez fogja lehetővé tenni hogy az adatbázissal elkezdhessünk műveletek végezni, például lekérdezések vagy új adatok feltöltése táblába. Ezen folyamatok után már meg is kezdődhetett a végpontok írása, amikkel majd a frontend eléri az adatbázist a megfelelő módokon.



19. ábra Az EF által létrehozott Models könyvtár



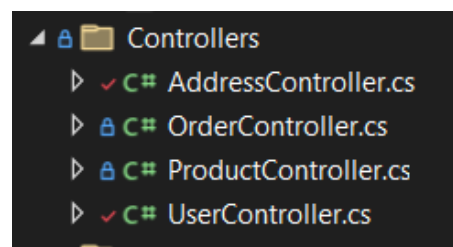
20. ábra Az általunk használt NuGet Package

4.6.1 Kontrollerek és végpontok írása

Kontrollerek: A WebAPI controller egy olyan osztály, ami kezeli a HTTP kéréseket és válaszokat. A kontrollerek függvényeket tartalmaznak, ezek definiálják a végpontokat, amelyek különböző műveleteket hajtanak végre. Ezeket a végpontokat hívhatják meg a kliensalkalmazások a bennük **definiált URL**-el, például **'http://localhost:5259/Product/getAll'**.

A kontrollereket a kezelendő táblák alapján kategorizáltuk, így 4 darab kontrollert sikerült létrehozni. Természetesen minden controllerben megvalósulnak a **CRUD (Create, Read, Update, Delete)** műveletek.

Az **'AddressController'** a felhasználó által megadott szállítási címeket kezeli, itt lehet címeket lekérdezni, létrehozni, módosítani és törölni.



21. ábra A létrehozott kontrollerek

Az **'OrderController'** a felhasználók által létrehozható és létrehozott rendelésekért felelős.

A 'ProductController' az adatbázisban lévő és a weboldalon megvásárolható termékeket kezeli.

A 'UserController' a felhasználók adataival foglalkozik, például regisztráció vagy bejelentkezés.

Nézzünk meg példának a 'Register' nevű végpontot. A végpontot a UserController tartalmazza.

Mindenek előtt definiálnunk kell a végpontunk HTTP metódusát, ez ebben az esetben **HttpPost**, mivel adatot szeretnénk feltölteni az adatbázisba. A metódus után megadjuk zárójelben az URL-t amin keresztül ellehet majd érni a végpontot.

A visszatérési érték egy 'ActionResult' lesz, ez teszi lehetővé hogy egy végpont különböző típusú válaszokat adjon vissza, például adatobjektumokat (JSON) vagy **HTTP státuskódokat**.

```
[HttpPost("register")]
0 references
public ActionResult Register(RegisterDto registerDto)
{
    using(var dbContext = new MeseWebshopContext())
    {
        try
        {
            var existingUser = dbContext.Felhasznalo.FirstOrDefault(x => x.Email == registerDto.Email);
            if (existingUser == null)
            {
                Extensions.SendVerificationEmail(registerDto.Email, registerDto.Name, "rg");
                var newUser = new Felhasznalo()
                {
                    Id = Guid.NewGuid(),
                    Nev = registerDto.Name,
                    Email = registerDto.Email,
                    Jelszo = GenerateSHA256(registerDto.Password),
                    Megerositve = 0,
                };

                dbContext.Felhasznalo.Add(newUser);
                dbContext.SaveChanges();

                return Ok("Sikeres regisztráció!");
            }
            else
            {
                return BadRequest("Már létezik felhasználó ezzel az email címmel!");
            }
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }
}
```

22. ábra Példa, a regisztrációért felelős végpontunk

Bemeneti paraméterként egy általunk létrehozott 'RegisterDto'-t kér. Ez tartalmazza a regisztrációhoz szükséges adatokat (név, email cím, jelszó).

Fontos hogy az egész művelet egy **try catch**-be legyen beletéve, ez azért fontos hogy ha bármi probléma miatt nem tud lefutni a kód, akkor ne állítsa meg a program futását, helyette csak egy hibát dob vissza.

A végpont először is megnézi hogy létezik-e már olyan felhasználó, aki a RegisterDto-ban megadott email címmel van regisztrálva. Ha van akkor értelemszerűen egy 400-as státuskóddal tér vissza (ez egy hibakód), ha nincs akkor küld egy **megerősítő emailt** a regisztrálni kívánó felhasználónak. Ez az email fogja tartalmazni azt a megerősítő kódot amivel a felhasználó megerősítheti fiókját, majd bejelentkezhet az oldalon.

Ezt követően létrehozunk egy új '**Felhasznalo**' objektumot, ennek beállítjuk a mezőit, majd a '**dbContext.Felhasznalo.Add**' paranccsal feltöltjük az adatbázisban található 'Felhasznalo' táblába. Utánna már csak elmentjük a módosításokat az adatbázisban, majd visszatérünk egy 200-as státuskóddal, itt ér véget a művelet.

5. WPF admin felület és karbantartó

A **WPF (Windows Presentation Foundation)** egy modern, **XAML alapú** keretrendszer a **.NET keretrendszeren** belül. Windows alkalmazások **grafikus felületének** készítésére és

kezelésére használják. Lehetővé teszi a gazdag és interaktív felhasználói élmények létrehozását, ezeket könnyen hozhatjuk létre és stílusosan formázhatjuk. A WPF általában a .NET alapú platformokon futtatható, például Windows.

5.1 Fő jellemzői, komponensek:

XAML: Az XAML támogatja a hierarchikus elemek elrendezését és a stílusok formázását, nagyon hasonló a felépítése a HTML-hez. A WPF tervezési felületét az XAML segítségével építjük fel, ez egy XML alapú nyelv a felületen lévő elemek elrendezésére és leírására.

Elemek és vezérlők: A WPF-ben számos beépített elem és vezérlő található, például gombok, szövegdobozok, listák, panelelemek, adattáblázatok stb. Ezeket a komponenseket XAML-ben vagy dinamikusan **C#** kódban testreszabhatjuk.

Adatkapcsolat és megjelenítés: A WPF támogatja az **adatkötést** (data binding), amely lehetővé teszi az adatok dinamikus megjelenítését egy felületen, például DataGrid-ben. Ennek segítségével könnyedén összeköthetjük az adatokat a GUI elemekkel, és automatikusan frissíthetők változás esetén.

5.2 Elképzelés és megvalósítás:

Mivel muszáj volt létrehozni egy asztali alkalmazást, ezért úgy döntöttünk hogy az admin felületünket itt valósítjuk meg. Ez azért jó nekünk mert így nem kell plusz energiát befektetni a weboldalunkba és külön felületet csinálni neki, illetve megvalósíthatunk egy adatkarbantartót. Az adatkarbantartó lehetővé teszi hogy az adatbázis tábláinak adatait módosítsuk, töröljük, vagy létrehozzunk újat.

Ha elindítjuk az alkalmazást először is bekell jelentkezni, az email és a jelszó is egyaránt az hogy 'admin'. Mivel az alkalmazás nem elérhető mindenki számára, csak a weboldal üzemeltetőjének, ezért nem volt szükség arra hogy az adatbázisban létrehozzunk egy külön admin felhasználót, és pluszban kezeljünk jogokat a felhasználóknak.



23. ábra A bejelentkező felület

Itt látható a bejelentkező felület, ami egy szolid design-t kapott. Az email és a jelszó mezők kitöltése után a bejelentkezés gombra kattintva, ha a megadott email és jelszó is 'admin', juthatunk be az alkalmazásba.



24. ábra A főoldal

A főoldalon 5darab menüpont közül választhatunk. Itt tudjuk kiválasztani hogy melyik adatbázis táblát szeretnénk kezelni. Illetve betudjuk zárni az alkalmazást, vagy kitudunk jelentkezni ami csak visszadob a bejelentkező oldalra.

A 'termékek' menüpontot kiválasztva az adatbázis termékeit láthatjuk kilistázva. Egy termére kattintva a DataGridben, az alkalmazás automatikusan kitölti a DataGrid mellett található mezőket. Itt átírhatjuk a kiválasztott termék adatait, majd a 'módosítás' gombra

kattintva elmenthetjük a módosításokat az adatbázisban. Emellett még letudjuk cserélni a kiválasztott termék fényképét, vagy ki is tudjuk törölni a terméket.

Id	Fotó	Kategória	Márka	Név	Leírás	Készlet	Ár (Ft)
00c3		9	Logitech	G502 X Plus	A G502 X PLUS W	18	51000
0d65		14	Intel	Core i9-10940X	Cascade Lake Pro	8	753000
1186		18	Silicon	Power	8GB memória, DD	5	5700
16c1		22	Deepcool	Z3	hővezető paszta,	5	1150

25. ábra Termékek karbantartó ablaka

Az 'új termék felvétele' gombra kattintva pedig egy új terméket vehetünk fel az adatbázisba. Itt természetesen meg kell adnunk minden adatát egy terméknek. Természetesen amint feltöltünk egy képet, akkor a placeholder kép lecserélődik a feltöltött képre. A termék feltöltését a 'termék hozzáadása' gombbal tudjuk véglegesíteni. A 'mégse' gombra kattintva megszakíthatjuk a folyamatot.

Minden menüpont ugyanerre a

26. ábra Új termék hozzáadása

sémára épül fel, csak a mezők száma és a mezők neve különbözik attól függően hogy melyik táblát szeretnénk kezelni.

A termékeket ezzel a függvénnyel tudjuk lekérni az adatbázisból. A 'JsonConvert' osztály a Newtonsoft.Json nevű NuGet Package tartalmazza.

```
1 reference
private List<ProductDto> GetProducts()
{
    List<ProductDto> datas = new List<ProductDto>();
    string path = App.url + "Product/getallforwpf";

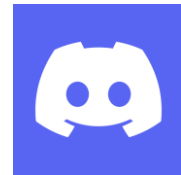
    WebClient client = new WebClient();
    client.Headers[HttpRequestHeader.ContentType] = "application/json";
    client.Encoding = Encoding.UTF8;
    try
    {
        string result = client.DownloadString(path);
        datas = JsonConvert.DeserializeObject<List<ProductDto>>(result);
        return datas;
    }
    catch (Exception ex)
    {
        return datas;
    }
}
```

Természetesen itt is le kellett scaffoldolnunk az adatbázist hogy hozzáférjünk a modelljeihez, tehát ugyanazok a package is felvannak még telepítve mint a backendben.

6. Kommunikációs felületek:

6.1 Discord:

Discord egy olyan platform, amely lehetővé teszi a felhasználók számára, hogy csoportos vagy egyéni kommunikációt folytassanak hangban, videóban és szöveges üzenetekben. Különböző funkciókat kínál a felhasználók számára, beleértve a hang- és videóchatet, a szöveges chatet, a képernyőmegosztást és sok más. A Discord-t eredetileg játékosoknak fejlesztették ki, hogy lehetőséget biztosítsanak számukra a játékok közötti kommunikációra, de mára már sok más közösség is használja, például tanulócsoporthok, munkahelyi csapatok és baráti társaságok.



A Discordon a felhasználók szerverekre és ezeken belül különböző csatornákra oszthatják a kommunikációt. Ez lehetővé teszi a különböző témák szerinti szervezést és a felhasználók számára a kényelmes navigációt.

Discordon belül a felhasználók szöveges üzeneteket küldhetnek egymásnak, akár közvetlenül, akár a szerveren belüli különböző csatornákon keresztül. A Discordon mi leginkább akkor kommunikáltunk mikor egy adott feladattal készen voltunk és az a másinak fontos volt hogy folytatni tudja a munkáját.

Azért a Discord-t választottuk mert ha valami probléma lenne amit nehéz lenne szövege formátumban leírni akkor van olyan lehetőségünk hogy felhívjuk egymást illetve a képernyő megosztás is egy opció.

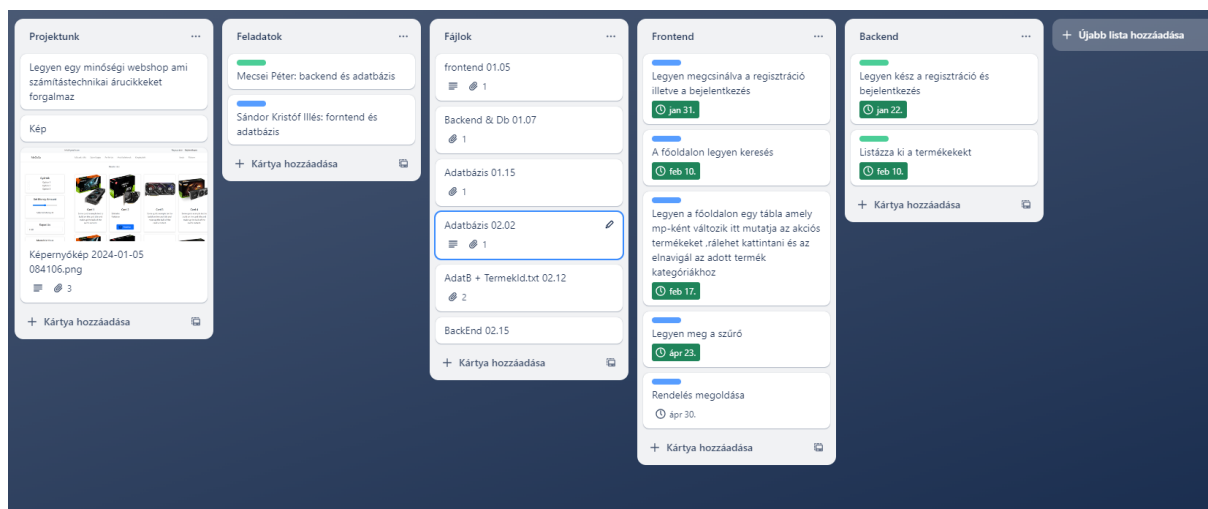
6.2 Trello:

A Trello egy online projektmenedzsment eszköz, amely lehetővé teszi a felhasználók számára a feladatok és projektek szervezését, nyomon követését és együttműködését. A Trello rendkívül intuitív és könnyen használható, és az egyszerű, lapokra és kártyákra épülő felülete révén kiválóan alkalmas egyéni felhasználóktól kezdve a vállalati csapatokig minden típusú projekt kezelésére.

A Trello fő egységei a táblák, listák és kártyák. A táblák reprezentálják a projekt teljes körét, a listák pedig a táblákban található folyamatokat vagy szakaszokat. A kártyák pedig az egyes feladatokat vagy teendőket jelölik.

A Trello egyszerű, Drag-and-Drop felülettel rendelkezik, amely lehetővé teszi a felhasználók számára a feladatok és kártyák egyszerű mozgatását és szervezését a különböző listák és

tablák között.



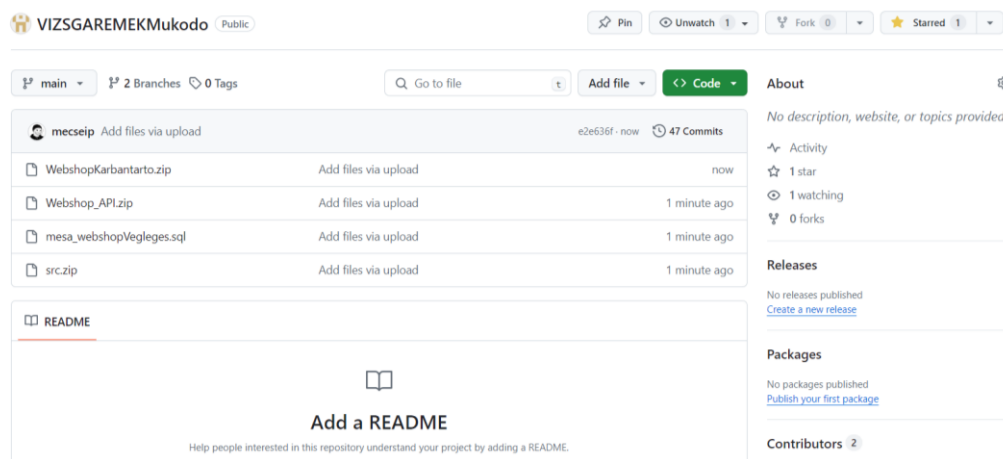
A Trello-t azért használtuk mivel ez az egyik oldal ahol feladatokat tudunk kiosztani a hét elején és így nyomon követve hogy ki mit csinál/csinált, ha valaki lemaradásban lenne akkor tudunk neki segíteni ha a személy kommunikációja nem kiváló.

6.3 Github:

A GitHub egy webes platform és szolgáltatás, amely lehetővé teszi a fejlesztőknek a kódmegosztást, verziókezelést és együttműködést szoftverfejlesztési projektek során. A GitHubot gyakran használják nyílt forráskódú projektekhez, de egyre inkább elterjedt a vállalati szoftverfejlesztési projektek során is.

A GitHub segítségével a fejlesztők verziókezelhetik a kódjukat, ami lehetővé teszi a változtatások nyomon követését, visszavonását és összefésülését a projekt többi részével.







A GitHub lehetővé teszi a fejlesztők számára, hogy megosszák kódjukat másokkal, legyen az nyilvános vagy privát repositorykban. Ez lehetővé teszi a fejlesztők számára a projekt munkájának megosztását és közös munkáját más fejlesztőkkel.



A GitHub-t mi azért használjuk mivel ide egyszerű és gyorsan fellehet tölteni fájlokat illetve ha a fájl régi változata már fent van a repository-ban akkor azt lecseréli arra az új fájlra amit fel akarunk tölteni.

6.4 Google Drive:

Egyszerű és biztonságos hozzáférés a tartalmaihoz. Bármilyen mobileszközről, táblagépről és számítógépről tárolhatja és megoszthatja a fájlokat és a mappákat, és együttműködhet rajtuk másokkal. Beépített védelem a rosszindulatú szoftverek, a spam és a zsarolóprogramok ellen. A Drive révén titkosított, biztonságos kapcsolaton keresztül férhet hozzá a fájljaihoz. Az Önnel megosztott fájlokat megelőző jelleggel megvizsgálja a rendszer, és ha rosszindulatú szoftvert, spamet, zsarolóprogramot vagy adathalászatot észlel, eltávolítja. Ráadásul a Drive natívan a felhőben működik, vagyis nincs szükség helyi fájlokra, és az eszközöket érintő kockázatok is minimalizálhatók.

Név ↑	Tulajdonos	Utolsó módosítás ▼	Fájl méret	
 01,05	 én	2024. jan. 5. én	—	⋮
 rekt.zip	 én	2024. jan. 1. én	86,8 MB	⋮
 Webshop_API.zip	 én	2024. febr. 23. én	28,9 MB	⋮

A Google Drive-ot az elején használtunk mivel információ hiányában itt osztottuk meg a nagy fájlokat.

7. Összefoglalás:

7.1 Sándor Kristóf:

Szerintem egységes volt a munka megosztás, tudtuk hogy mindketten másik ágba vagyunk érdekeltek és ezt kihasználva osztottuk fel a feladatokat. Folyamatos volt a kommunikáció egymást közt, Mecsei Péter hetente kérdezett az frontend állapotáról.

Terveinket sikerült megvalósítani, nem volt részletes elképzelés hogy miket szeretnénk megvalósítani, csak egy sablon weboldallal kezdtük és szerettük volna minél felhasználóbarátra elkészíteni. A frontend részt szerintem sikerült teljesíteni, illetve az adatbázisban a termék adatait én vittem fel. A frontend-en az elején nem voltak nagy elképzelések, úgy gondoltam hogy miközben mennek a fejlesztések aközben jönnek majd az ötletek és úgy tudom fejleszteni a weboldalt. A rendelés fizetést még nem szerettük volna elkészíteni mivel még nincs erre lehetőség hogy kártyával lehessen fizetni így ez még csak fejlesztés alatt van.

Az adatbázisban a termékek feltöltése nem okozott gondot. Ez lehetővé teszi, hogy hatékonyan kezeljük és rendezzük az összes termék adatait, ami kulcsfontosságú a vállalkozás szempontjából. Most, hogy a termékek az adatbázisban vannak, további lépések következhetnek.

7.2 Mecsei Péter:

Én azt gondolom hogy ez egy nagyon jól összerakott projekt volt. Az elején voltak nehézségek elképzelés terén mivel először nem webshopot kezdtünk el csinálni, aztán rájöttünk hogy ez egyszerűbb és jobb mint az eredeti elképzelés. Szerintem nagyon jól sikerült együtt dolgoznunk és a munkafelosztást is jól megcsináltuk.

Én csináltam a backendet, a wpf-es adatkarbantartót és az adatbázist. Sok sok tapasztalatot és tudást sikerült szerezni ezeknek a megvalósításának a folyamán. Bár néha fájdalmas volt és megterhelő agyilag hogy orvosoljak egy hibát, de szerintem teljesen megérte megcsinálni. Előnyös volt számunkra még hogy viszonylag hamar elkezdtünk dolgozni a projekt-el másokhoz képest így a végére kevésbé kellett sietni és közben tarthattunk kisebb nagyon pihenőket.

8. Források

1. Téma: phpMyAdmin, webcím: <https://www.phpmyadmin.net/> ,megtekintve: 2024.04.26.
2. Téma: MySQL, webcím: <https://www.mysql.com/> ,megtekintve: 2024.04.26.
3. Téma: Normál formák, webcím: <https://tudasbazis.sulinet.hu/> ,megtekintve: 2024.04.26.
4. Téma: WampServer, webcím: <https://www.wampserver.com/en/> ,megtekintve: 2024.04.26.
5. Téma: React, webcím: <https://react.dev/> ,megtekintve: 2024.04.26.
6. Téma: Node.js, webcím: <https://nodejs.org/en> ,megtekintve: 2024.04.26.
7. Téma: Visual Studio Code, webcím: <https://code.visualstudio.com/> ,megtekintve: 2024.04.26.
8. Téma: HTML és CSS, webcím: <https://www.codecademy.com/catalog/language/html-css> ,megtekintve: 2024.04.26.
9. Téma: ASP.NET Core és WebAPI, webcímek: <https://dotnet.microsoft.com/en-us/apps/aspnet/apis> , <https://www.tutorialsteacher.com/webapi/what-is-web-api> ,megtekintve: 2024.04.26.
10. Téma: Visual Studio 2022, webcím: <https://visualstudio.microsoft.com/vs/> , megtekintve: 2024.04.26.
11. Téma: C#, webcím: <https://learn.microsoft.com/en-us/dotnet/csharp/> , megtekintve: 2024.04.26.
12. Téma: Entity Framework, webcím: <https://learn.microsoft.com/en-us/ef/> , megtekintve: 2024.04.26.
13. Téma: WPF, webcím: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/> , megtekintve: 2024.04.26.
14. Képek:
 - a. WAMP logo: <https://hu.pinterest.com/pin/explore-the-modern-wampserver-logo--846817536174103292/> ,letöltve: 2024.04.26.
 - b. React : <https://react.dev/> ,képernyőkép készült: 2024.04.26.
 - c. Node.js: <https://nodejs.org/en> ,képernyőkép készült: 2024.04.26.
 - d. Visual Studio Code: <https://code.visualstudio.com/> ,képernyőkép készült: 2024.04.26.
 - e. HTML és CSS: <https://medium.com/@mfaisalh140202/mengenal-html-css-dan-tag-dalam-pengembangan-web-351d75483815> ,letöltve: 2024.04.26. és <https://www.linkedin.com/pulse/relationship-html-css-rezwana-islam-kak0c> ,letöltve: 2024.04.26.
 - f. ASP.NET Core: <https://wakeupandcode.com/middleware-in-asp-net-core/> ,letöltve 2024.04.26.
 - g. Entity Framework: <https://medium.com/@djouflegran/intro-to-entity-framework-core-in-net-6-c674e10edab3> ,letöltve: 2024.04.26.
 - h. Visual Studio logo: <https://softwarelicense4u.com/en/visual-studio-enterprise-2017/?wmc-currency=GBP> ,letöltve: 2024.04.26
 - i. C# logo: <https://www.kojac.nl/en/back-end/csharp> ,letöltve: 2024.04.26.
 - j. Discord logo: <https://apps.microsoft.com/detail/xpdc2rh70k22mn?hl=ar-SA&gl=NG> ,letöltve: 2024.04.26