

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  
*Bemutatása*

Mecsei Péter – Sándor Kristóf Illés

2023-2024

Tartalomjegyzék

Miért webshopot választottunk?:

Azért a webshopot választottunk mert szerintünk egy hétköznapi szoftver a webshop de mégis megmérettetés számunkra, illetve hasznos tapasztalatot tudunk szerezni frontend-en és backend-en.

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.

**Adatbázis és a phpMyAdmin**

**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.

A screenshot of a computer

Description automatically generated 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

**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.

**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 **A pink and black logo

Description automatically generated**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.

**Az elképzelés:**

A tervezés alatt egy olyan adatbázist szerettünk volna létrehozni melynek felépítése segíti az adatbázis tábláiban lévő adatok **jó átláthatóság**át és **kezelhetőség**ét. Emellett a megvalósítás majd a tesztelés során létrejövő hibák, esetleges hiányosságok könnyen orvosolhatóak legyenek.

**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 létrehoznunk.

A screenshot of a computer

Description automatically generatedA táblák **összekötése** is remek lett mivel a backend fejlesztésének menetében egyértelmű volt hogy milyen lekérdezéseket kell alkalmaznunk, azokat hogy kell megvalósítanunk, valamint miként kell feltöltenünk 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 azonosító**val rendelkezik, ami megakadályozza a rekordok duplikálását, illetve hogy lekérdezésekkor és adatok feltöltésekor ne akadjon össze az adatbázis.

Fontos megjegyezni hogy a **termékek**nek 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 kulcs**csal van ellátva. Ez azért jó nekünk mert adatbázis szinten is védelmet nyújt a redundanciával szemben.

**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 screenshot of a computer

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

2. 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 felhasználókhoz vannak rendelve a ’’Felhasz\_Id’’ segítségével.

**WebAPI backend**

**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 kommunikácóra. A WebAPI-k általában **HTTP (Hypertext Transfer Protocol) protokol**lt használnak üzenetek küldésére és fogadására.

**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.

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

**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.

A diagram of a web application

Description automatically generated

3. ábra Az ASP.NET Core működési ábrája

**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 rendeszen.

**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.

**Tejlesítményorientált:** 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.

**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.

A purple logo on a black background

Description automatically generated

**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.

**Néhány főbb jellemző:**

**Teljes körű .NET támogatás:** Lehetővé teszi .NET 6-tól felfelé minden .NET verizóban 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 optimalizásá:** Az IDE javított teljesítményt kínál a nagyob 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, valamit 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.

**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 A logo of a company

Description automatically generatedlehessen 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.

**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 ellenőrni 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)**, aszinkronprogramozás **(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 folamatok támogatásában.

**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).

**A blue background with arrows

Description automatically generated**

4. ábra Az Entity Framewrok működési ábrája

**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, ls 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 tesztelhetőek 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.

**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 kellet bármi külső programot, például Postman, használnunk.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

5. ábra A Swagger kinézete

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

6. ábra Az általunk felhasznált package-k

A screenshot of a computer

Description automatically generated A **’scaffold-dbcontext’**-et az Entity Framework tartalmazza. Ahhoz hogy működjön, nem csak a telepített **NuGet Package**kre 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.

7. ábra Az EF által létrehozott Models könyvtár

**Kontrollerek és végpontok írása**

**Kontrollerek:** A WebAPI kontroller 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 screenshot of a computer screen

Description automatically generated A kontrollereket a kezelendő táblák alapján kategorizáltuk, így 4darab kontrollert sikerült létrehozni. Természetesn minden kontrollerben megvalósulnak a **CRUD (Create, Read, Update, Delete)** műveletek.

Az ’AddressController’ a felhasználó által megaddott szállítási címeket kezeli, itt lehet címeket lekérdezni, létrehozni, módosítani és törölni.

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.

A computer screen shot of a program code

Description automatically generatedNézzünk meg példának a ’Register’ nevű végpontot. A végpontot a UserController tartalmazza.

8. ábra Példa, a regisztrációért felelős végpontunk

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 státuszkód**okat.

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 státuszkóddal tér vissza (ez egy hibakód) , ha nincs akkor küld egy **megerősítő email**t 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’ objektum**ot, 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 státuszkóddal, itt ér véget a művelet.

**WPF admin felület és karbantartó**

A **WPF (Windows Presentation Foundation)** egy modern, **XAML alapú** keretrendszer a **.NET keretrendszer**en belül. Windows alkalmazások **grafikus felület**ének 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.

**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 **adatkötés**t (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 frissíthetőek változás esetén.

**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 belefektetni a weboldalunkba és külön felületet csinálni neki, illetve megvalósíthatunk egy adatkarbantartót. Az adatkarbantartó lehetővé teszi hogy az adatbázis tábláinak adatatit módosítsuk, töröljük, vagy létrehozzunk újat.

Ha elindítjuk az alkalmazást először is bekell jelentkezni, az emial 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.

 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 emial és jelszó is ’admin’, juthatunk be az alkalmazásba.

9. ábra A bejelentkező felület

A computer screen with text and blue objects

Description automatically generated with medium confidence

10. ábra A főoldal

A főoldalon 5darab menupont 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 mellet 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.

A screenshot of a computer

Description automatically generated

11. ábra Termékek karbantartó ablaka

A screenshot of a computer

Description automatically generated Az ’új termék felvétele’ gombra kattintva pedig egy új terméket vehetünk fel az adatbázisba. Itt természetesen megkell 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.

12. ábra Új termék hozzáadása

A screen shot of a computer program

Description automatically generated Minden menüpont ugyanerre a 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. Természetesen itt is lekellett scaffoldolnunk az adatbázist hogy hozzáférjünk a modelljeihez, tehát ugyanazok a packagek is felvannak még telepítve mint a backendben.