

SZEGEDI SZAKKÉPZÉSI CENTRUM
VASVÁRI PÁL GAZDASÁGI ÉS INFORMATIKAI
TECHNIKUM

5 0613 12 03 Szoftverfejlesztő és -tesztelő

Webáruház és Készletkezelő

Készítette:
Imró-Bárkányi Erika
Szabácsi Noémi

Szeged

2022

Tartalom

BEVEZETÉS (Imrő-Bárkányi Erika, Szabácsi Noémi).....	4
1. DOKUMENTÁCIÓ KÉSZÍTÉSE.....	6
2. RENDSZERSPECIFIKÁCIÓ (Imrő-Bárkányi Erika).....	6
2.1 Funkcionális követelmények	7
2.2 Nem funkcionális követelmények	7
2.3 Hardver és Szoftverkövetelmény	8
3. ADATBÁZIS (Imrő-Bárkányi Erika)	9
3.1 Adatbázis tervezése.....	10
3.2 Adatbázis bemutatása.....	12
3.3 Az adatbázis tábláinak felépítése	12
4. FEJLESZTŐI DOKUMENTÁCIÓ	14
4.1 Asztali alkalmazás (Imrő-Bárkányi Erika).....	14
4.1.1 Alkalmazott fejlesztői eszközök, nyelvek	14
4.1.2 Asztali alkalmazás megvalósítása	15
4.1.3 Tesztek.....	17
4.2 WebApi (Imrő-Bárkányi Erika).....	19
4.2.1 Alkalmazott fejlesztői eszközök, nyelvek	19
4.2.2 WebApi megvalósítása	20
4.2.3 Tesztek.....	21
4.3 Mobil alkalmazás (Szabácsi Noémi)	25
4.3.1 Alkalmazott fejlesztői eszközök, nyelvek	25
4.3.2 Mobil alkalmazás megvalósítása	26
4.3.3 Tesztek.....	27
4.4 Webes alkalmazás (Szabácsi Noémi).....	28
4.4.1 Alkalmazott fejlesztői eszközök, nyelvek	28
4.4.2 Webes alkalmazás megvalósítása	29
4.4.3 Tesztek.....	31
5. FELHASZNÁLÓI DOKUMENTÁCIÓ	32
5.1 Asztali alkalmazás bemutatása (Imrő-Bárkányi Erika)	32
5.2 WebApi bemutatása (Imrő-Bárkányi Erika)	35
5.3 Mobil alkalmazás bemutatása (Szabácsi Noémi)	35
5.4 Webes alkalmazás bemutatása (Szabácsi Noémi)	37
6. FEJLESZTÉSI LEHETŐSÉGEK.....	39
6.1 Asztali alkalmazás (Imrő-Bárkányi Erika).....	39
6.2 WebApi (Imrő-Bárkányi Erika).....	39

6.3	Mobil alkalmazás (Szabácsi Noémi)	39
6.4	Webes alkalmazás (Szabácsi Noémi).....	39
7.	ÖSSZEGZÉS (Imrő-Bárkányi Erika, Szabácsi Noémi)	40

Köszönetnyilvánítás

Ábrajegyzék

Forrásjegyzék

Melléklet

HALLGATÓI NYILATKOZAT

Alulírottak **Imrő-Bárkányi Erika** és **Szabácsi Noémi** a Szegedi Szakképzési Centrum Vasvári Pál Gazdasági és Informatikai Technikum hallgatói kijelentjük, hogy a **Webáruház és Készletkezelő** című záródolgozat a saját munkánk.

Kelt: Sándorfalva, 2022. április 23.

aláírás

aláírás

BEVEZETÉS (Imrő-Bárkányi Erika, Szabácsi Noémi)

Az elmúlt években a világban történő események alakulásának (világméretű járvány) köszönhetően, nagyobb teret nyertek az internetes felületen történő értékesítések. Bár korábban nem volt jellemző az emberekre a netes vásárlási kedv, de az új helyzet, új helyzetet teremtett. Számos előnye van az online vásárlásnak, melyet egyre több vásárló vesz igénybe. Bármikor nézegetheti a termék kínálatot az interneten, hiszen ez nem függ a nyitvatartási időtől. Legnagyobb előnye, hogy nem kell időt tölteni a sorban állással, és természetesen ezek mellett az online fizetési lehetőség is adott. A vállalkozás részéről erőforrást spórolhat az internetes áruház, és akár nagyobb forgalmat is tud bonyolítani. A mai világban tehát elengedhetetlen, hogy a gazdaságos működés érdekében, a vállalkozás biztosítson internetes felületen keresztül is vásárlási lehetőséget a vásárlói részére.

Szakdolgozatunk témája egy élelmiszer értékesítésével foglalkozó kiskereskedelmi egység fejlesztése, amely magában foglal egy mobil alkalmazást, egy asztali alkalmazást és egy weboldalt. Választásunk azért erre a témára esett, mert van személyes kötődésünk a bolt tulajdonosához és szerettünk volna hozzájárulni a bolt fejlődéséhez. Az szóban forgó kiskereskedelmi egység (Továbbiakban: Ica bolt), eddig csak egy weblappal rendelkezett, ahol információt gyűjthettek a vásárlók a termékekkel és a bolttal kapcsolatosan. Sajnos a vállalkozás nem volt felkészülve egy világjárványra (mint ahogyan sokan mások sem), ami többek között a bolt látogatását is szabályozta. A vásárlók kiszolgálásának gördülékenysége érdekében kerestünk megoldást, annak a ténynek a figyelembe vételével, hogy az online vásárlással járó kiszállítást, gépjárműkapacitás hiányában nem tudta biztosítani az Ica bolt.

Az általunk megalkotott applikációk nem csak a vásárlók komfortérzetét, hanem a dolgozók kényelmét is szolgálják. A webes alkalmazáson keresztül a vásárlók a megvásárlandó termékeik összeválogatása után egy rendelést tudnak leadni, amely rendelésért be kell menni a boltba. Így gyakorlatilag a megérkezésük után megkapják a kért rendelésüket és a boltnak sem kell a kiszállítással foglalkozni. Ez egy kis falusi bolt, nem releváns a távolság. Az asztali alkalmazásnak köszönhetően, a dolgozók akár az otthonukból is tudnak a készlettel foglalkozni, mert az applikáción keresztül tudnak terméket módosítani, újat felvinni, megtalálják a beszerzési időt és az árakat. A mobil applikáció szintén a vásárlóknak szól. A bolt termékei közül ki tudják választani azokat, amelyeket szívesen vásárolnak, majd ezek kiválasztása után, abban az esetben ha az adott termék akciós lesz, kapni fognak egy értesítést. A projektünkbe még nagyon sok fejlesztési tartalék van, melyek

kisebb nagyobb prioritással rendelkeznek. Természetesen ezek a jövőben beépítésre kerülnek, hiszen csak így tudnak az applikációk együtt hatékonyabban működni, ezzel segítve a vásárlókat és a bolt dolgozóit. Nem utolsó sorban számunkra is nagyon fontos, hogy az Ica bolt gazdaságosan és korszerűen működjön, hiszen a családjunk tagja.

1. DOKUMENTÁCIÓ KÉSZÍTÉSE

Szakdolgozatunkat közösen készítettük a Microsoft Office csomag szövegszerkesztőjének segítségével. Mindketten az általunk készített alkalmazásokhoz szerkesztettünk leírást, ezt a cím mellett, nevünk feltüntetésével jelöltük. A bevezető és összefoglaló írását szintén megosztottuk egymás között.

A dokumentáció célja

- Az online megrendelés lépéseinek, valamint a weboldal egyéb részeinek ismertetése
- A mobil alkalmazás bemutatása
- Asztali alkalmazás funkcióinak bemutatása
- Tesztek ismertetése

2. RENDSZERSPECIFIKÁCIÓ (Imró-Bárkányi Erika)

A rendszer kialakításánál a megrendelő által megfogalmazott igényeket vettük figyelembe. A kialakult helyzetre való tekintettel, inkább a gyors igénybevehetőség és a főbb funkciók működése volt a cél. A vevőkör megtartása érdekében az új vásárlási csatorna kialakítása, a gördülékenyebb munkavégzés megoldása volt a fontos. A megvalósítandó feladatok egyike, hogy a weboldal megtekintését követően, lehetőség legyen kiválasztani terméket, majd ezt elhelyezni a kosárba és megrendelni. Mobilalkalmazás tekintetében pedig, egy olyan szolgáltatás volt a megrendelő kérése, hogy az alkalmazáson keresztül a vásárló a saját kedvenc termékét tudja megjelölni, amely ha akcióssá válik a boltban, akkor erről egy értesítést kapjon. Ezek olyan termékek amelyeket a vevő gyakrabban vásárol, különböző okok, szempontok miatt, mint például az íze, ára, összetétele stb miatt.

A vevőkör különböző összetételű korosztályt jelent, ezért fontos szempont volt, hogy felhasználóbarát, egyszerű, könnyen érthető és kezelhető funkciók legyenek az alkalmazások. A dolgozók esetében a termékek nyilvántartása már nem excel formában, hanem egy asztali alkalmazás keretében történne. A fenti igényekből jól látszik, hogy szerepet fog kapni egy központi adatbázis is, ahol termékek, vásárlók, adminok, rendelés adatok kerülnek eltárolásra. Összeségében egy olyan rendszerre lenne szükség, ahol egyszerű felületen, könnyen navigálva eléri a célját mind a vásárló és mind a dolgozó.

2.1 Funkcionális követelmények

Asztali alkalmazás (dolgozóknak):

- Terméklista szerkesztése
- Új termék felvétele
- Meglévő termék törlése
- Meglévő termék módosítása
- Akciós szórólap képeinek eltárolása

Weboldal (vevőknek):

- Előzetes regisztráció nélküli rendelési lehetőség a weboldalon keresztül
- Információk a termékekről, árakról, nyitvatartásról, elérhetőségről
- Kiválasztott termékek kosárba helyezése, mennyiségének növelése, csökkentése
- Kosárban lévő termékek törlése
- Aktuális újság megtekintése
- Rendelés elküldése a bolt részére

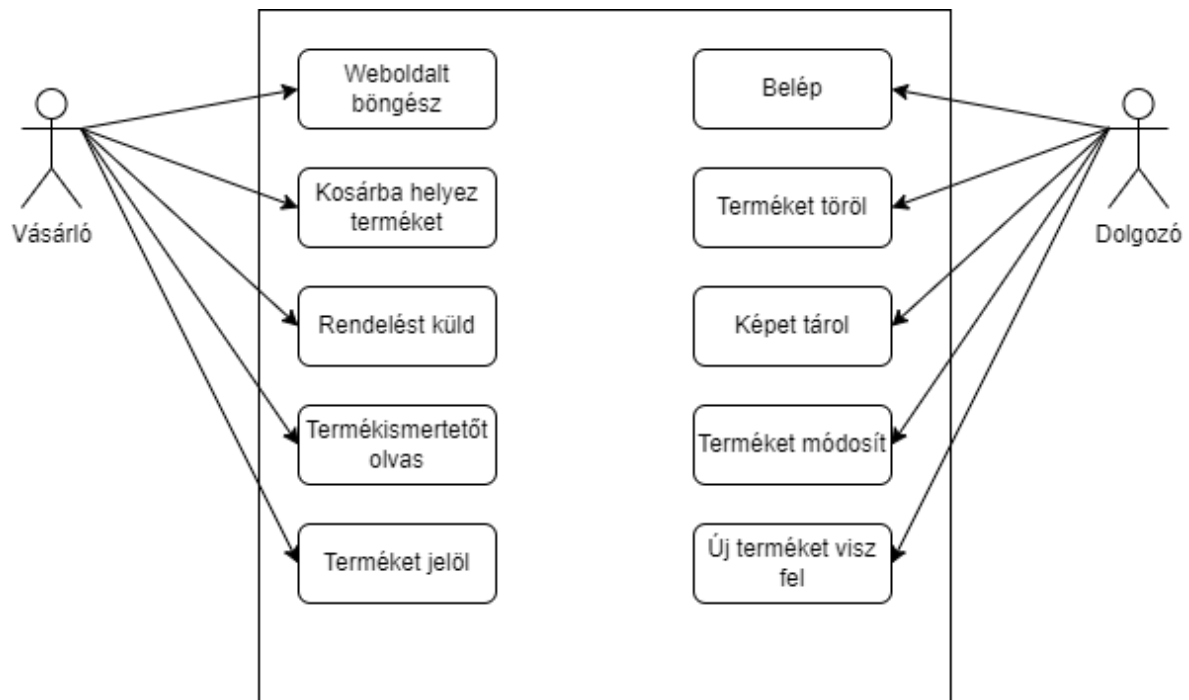
Mobilalkalmazás (vevőknek):

- Termékek megjelenítése
- Termékek kiválasztása, tárolása
- Visszajelzés küldése

2.2 Nem funkcionális követelmények

A rendszernek biztonságosnak kell lennie. Az adatokhoz csak az engedélyezett felhasználók férhetnek hozzá és könnyen karbantarthatónak kell lennie. A szoftvernek működőnek és használhatónak kell lennie. A megjelenésnek számítógépen és mobil eszközön is lehetőleg olyannak kell lennie, hogy megfelelő legyen a felhasználónak. Igyekeztünk minden alkalmazást úgy kialakítani, hogy azok minden eszközön könnyen kezelhetők, használhatóak, letisztultak legyenek. A mobil esetben a belépés után csak a termékek neveit tartalmazza az oldal, hogy feleslegesen ne terheljük a megjelenítést. A kiválasztott termék esetében természetesen a teljes információ megjelenik. Ugyanakkor több böngészőben (Mozilla Firefox, Opera GTX) és Android Studióban is ellenőrzésre került a megjelenítés működése. A weboldal és a mobilalkalmazás reszponzivitása szintén

kialakításra került. Ügyeltünk az asztali alkalmazásba való biztonságos belépés funkciójára is. A dolgozók jelszava titkosításra kerültek, hiszen itt a termékek törlése, módosítása történik.



1. ábra - Use-Case diagram

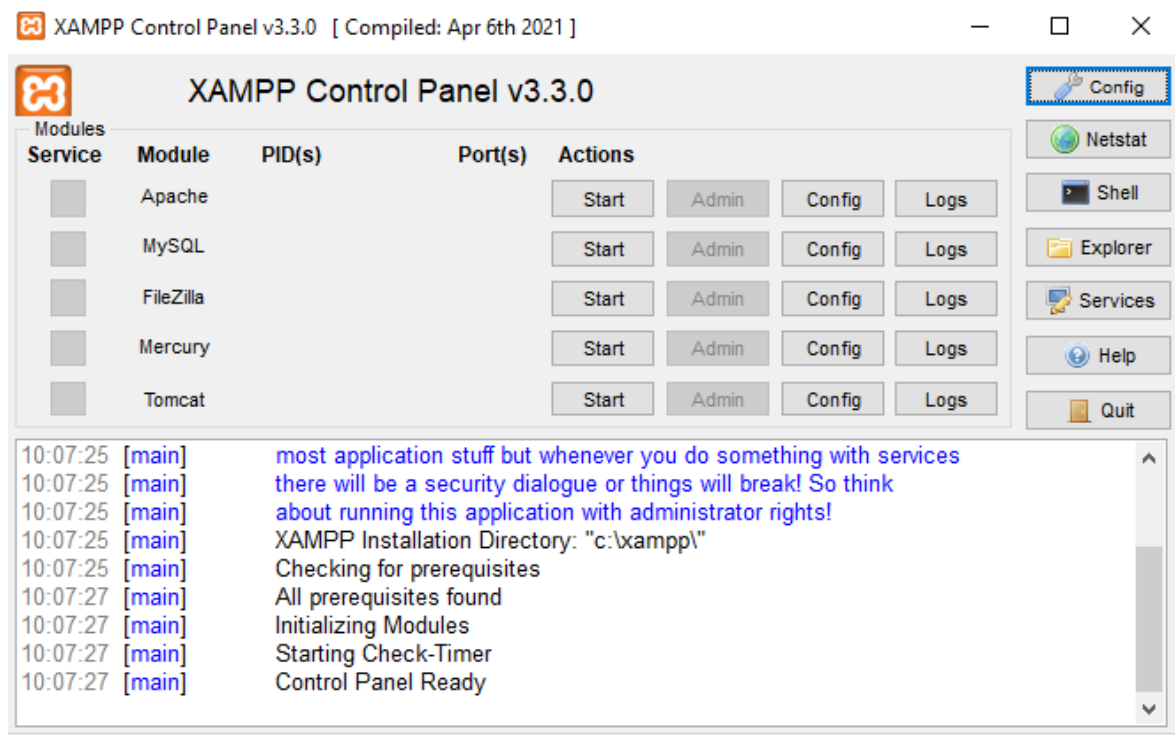
2.3 Hardver és Szoftverkövetelmény

A programoknak elegendő egy minimális hardverkörnyezet, hiszen ezek nem túl nagy erőforrást igényelnek. Az asztali alkalmazás és a webalkalmazás esetében ajánlott bármilyen olyan számítógép, amely megfelel a Microsoft Windows 7 operációs rendszer minimális rendszerkövetelményének. A mobil alkalmazás esetében minimum 512 mb memória, 480x800-as felbontással rendelkező mobileszköz ajánlott.

A kliens gép számára böngésző és folytonos internetkapcsolat szükséges az alkalmazások megtekintéséhez. Az internetkapcsolat sebessége nagyban befolyásolja az alkalmazások működését, mert a szerver felé irányuló kérések válasz ideje, egy gyengébb internetkapcsolat esetén a többszörösére nőhet. A webes felület platformfüggetlen, működik bármely frissebb böngésző segítségével. Részünkről Opera, Firefox, Google Chrome böngészővel teszteltük a használatát. Mobil alkalmazás esetében Android 8.0 vagy nagyobb verziójú operációs rendszer szükséges. Átlagos belső hálózat alkalmas a rendszer működtetésére. Az adatbázis használatához szükséges a XAMPP nevű program telepítése.

3. ADATBÁZIS (Imrő-Bárkányi Erika)

Az adatbázis elkészítéséhez a tanulmányaink során megismert **XAMPP** nevű programot használtuk, mely platformfüggetlen, szabad és nyílt forráskódú webservert - szoftvercsomag. Több alkotóeleme is van, így például az Apache webservert, a MariaDB(MySql) adatbázis-kezelő, PHP szerveroldali szkriptnyelv, Perl általános célú szkriptnyelv. A szoftvercsomag egy integrált rendszert alkot, ebben a csomagban minden megtalálható, amely a webes alkalmazások készítéséhez, teszteléséhez, futtatásához szükséges. A vizsgaremekben található alkalmazások működéséhez elengedhetetlen egy jól megtervezett adatbázis, ebben a MySQL volt a segítségünkre, mert ennek köszönhetően értük el a phpMyAdmin felületet. Itt készült el az adatbázis, az ehhez tartozó táblák, valamint ezen a felületen töltöttük fel a táblákban található adatokkal az SQL nyelv segítségével.



2. ábra - Xampp Control Panel

3.1 Adatbázis tervezése

Az adatbázis egy élelmiszerbolt online értékesítési lehetőségét, információ áramlását és készletnyilvántartását támogatja, tehát mindenképp tartalmaznia kell a termékeket, valamint a vásárláshoz és karbantartáshoz szükséges táblákat is. Az adatbázis tervezésekor az alábbi szempontokat vettük figyelembe:

- Rendszer elemzése: Melyek azok a termékek és azokhoz tartozó információk amelyből az adatbázist összeállítjuk, ebben segítségünkre volt a megrendelő.
- Rendszer tervezése: Csökkentjük a redundanciát, azaz az ismétlést, például nem tárolunk kiszámítható mezőket. A kapott adatokat 3. normál formára hozzuk, kialakítjuk az elsődleges és idegen kulcsokat, megszüntetjük a tranzitív függőségeket.
- Megvalósítás: Tudjuk a tábláink összetételét, azaz, hogy milyen oszlopokból és mezőkből fog állni, valamint milyen kapcsolatokat kell kialakítanunk.

Webes alkalmazás

Az adatbázis lehetőséget ad arra, hogy a weboldalon keresztül online rendelést leadó felhasználók, keresést tudjanak végrehajtani a termékek között, valamint kiválasztás után azt a kosárba helyezni és megvásárolni. A keresés gomb lenyomásával az adatbázis „termék” táblája biztosítja, hogy a vásárló válogatni tudjon a kínálatból. A kiválasztott termék „Kosárba” gombjára klikkelve átjut kosár oldalára, ahol a vásárló által rendelt mennyiség, cikknév az adatbázis „rendelési_tételek” táblájában tárolódik el. A „vásárló” tábla a vásárlók adatait tartalmazza, úgy mint cím, email cím, felhasználónév, jelszó stb formájában, melynek segítségével online rendelést tudnak leadni.

Asztali alkalmazás

Az asztali alkalmazáson keresztül a termékek karbantartása történik, ezt 2 admin végzi, miután a megfelelő felhasználónevet és jelszót megadva belépnek a készletnyilvántartóba. Ezek tárolását az „admin” elnevezésű táblában találjuk. Belépés után a „termék” tábla töltődik be, melyen lehetőség van törlésre és új termék felvitelére is. Ezekben az esetekben, az adatbázisban is megtalálható lesz az új termék vagy törlődik.

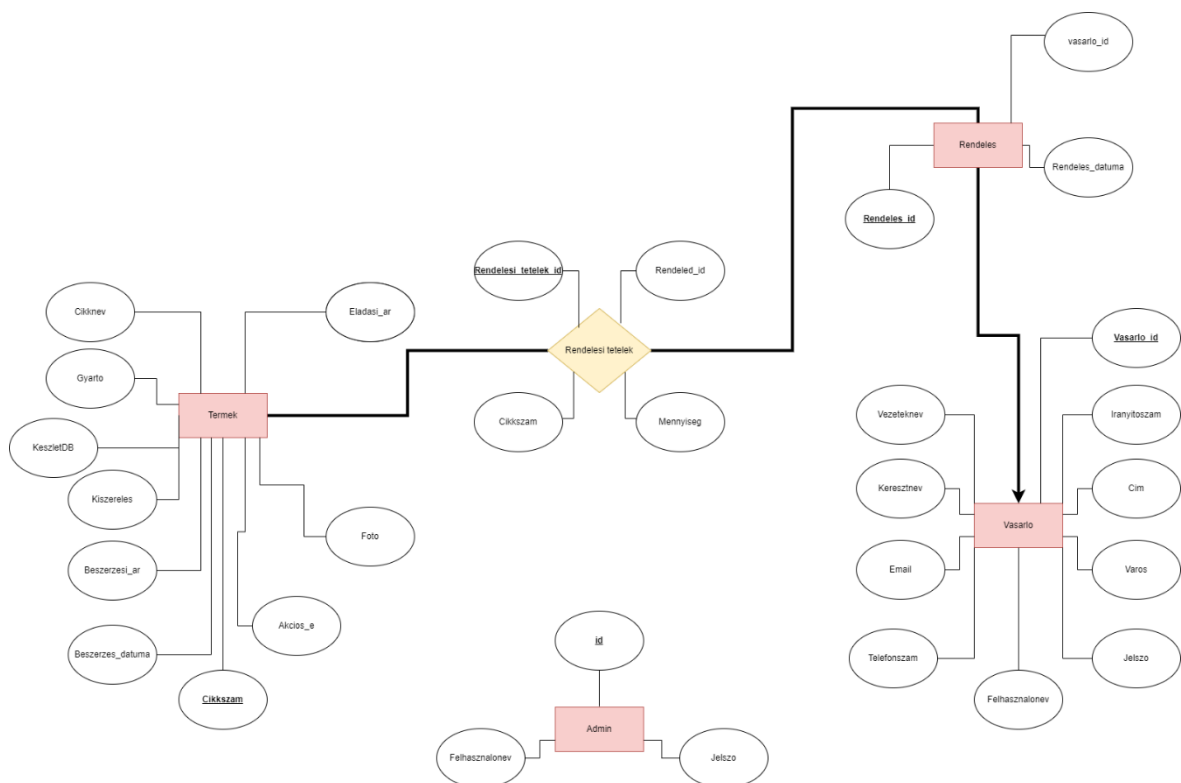
Mobil alkalmazás

A mobil applikáció lehetőséget nyújt a felhasználónak, hogy termékeket nézzon meg, valamint kiválaszthassa mint olyan terméket, amelyről értesítést kér ha a boltban akciós lesz. A termék az adatbázis „termék” táblájából kerül az alkalmazásba, majd kiválasztás után a „rendelési tételek” táblába fog megjeleníteni.

Backend

A backend az a háttérprogram, amely egy webes, asztali alkalmazás vagy mobilalkalmazás adatait kezeli. Minden logikát tartalmaz az adatok eléréséhez és kezeléséhez, amelyekhez a hétköznapi felhasználók nem férhetnek hozzá. A háttérrendszer felelős a webes kérések és webes válaszok kezelésért is. Az adatbázisban megtalálható táblák közötti kapcsolat kialakítása ebben nagy szerepet játszik.

Megvalósítás folyamatánál nagy hangsúlyt fektettünk a megrendelő által kért, kinyerhető adatokra.



3. ábra – E-K Diagram

3.2 Adatbázis bemutatása

Egyedek: Termék, Vásárló, Rendelés, Rendelési tételek, Admin

Tulajdonságok:

- Termék (Cikkszám, Cikknev, Gyarto, KeszletDB, Beszerzesi_ar, Eladasi_ar, Kiszereles, Beszerzes_datuma, Foto, Akciós_e)
- Vásárló (Vasarlo_id, Vezeteknev, Keresztnev, Email, Telefonszam, Felhasznalonev, Jelszo, Varos, Cím, Iranyitoszam)
- Rendelés (Rendeles_id, Vasarlo_id, Rendeles_datuma)
- Rendelési tételek (Rendelesi_tetelek_id, Rendeles_id, Cikkszám, Mennyiség)
- Admin (id, Felhasznalonev, Jelszo)

3.3 Az adatbázis tábláinak felépítése

Termek tábla, ebben a táblában található a termékekről az összes információ.

- Cikkszám: int(10) PRIMARY KEY – Szám, Elsődleges kulcs
- Cikknev: varchar(50) – Szöveg típusú, a termék neve
- Gyarto: varchar(50) – Szöveg típusú, a termék gyártójának neve
- KeszletDB: int(11) – Szám, a készleten lévő darabszámot jelzi
- Beszerzesi_ar: int(11) – Szám, a termék beszerzési értékét mutatja
- Eladasi_ar: int(11) – Szám, a termék eladási ára
- Kiszereles: varchar(15) – Szöveg, a termék kiszerezésére vonatkozó adat
- Beszerzes_datuma: date – Dátum típusú, beszerzés dátumát mutatja
- Foto: varchar(255) – Szöveg, termék fotó neve
- Akcios_e: boolean – logikai típusú, jelenleg akciós-e az áru

Vasarlo tábla, itt találhatóak a vásárlók adatai

- Vasarlo_id: int(11) PRIMARY KEY – Szám, Elsődleges kulcs
- Vezeteknev: varchar(50) – Szöveg típusú, a vásárló vezetéknéve
- Keresztnev: varchar(50) – Szöveg típusú, a vásárló keresztnéve
- Email: varchar(50) – Szöveg, a vásárló email címe
- Telefonszam: varchar(20) – Szöveg, a vásárló telefonszáma
- Felhasznalonev: varchar(50) – Szöveg, a vásárló felhasználó neve
- Jelszo: varchar(50) – Szöveg, a megadott jelszó

- Varos: varchar(50) – Szöveg, a vásárló város lakhelye
- Cim: varchar(50) – Szöveg, a vásárló utca, házszám, emelet szerinti lakcíme
- Irányitoszam: int(11) – Szám, a vásárló lakcímének irányítószáma

Rendeles tábla, itt található, hogy mikor és ki rendelt.

- Rendeles_id: int(11) PRIMARY KEY, AUTO INCREMENT – Szám, Elsődleges kulcs
- Vasarlo_id: int(11) – idegen kulcs
- Rendeles_datuma: date – Dátum típusú, a rendelés dátumát rögzíti

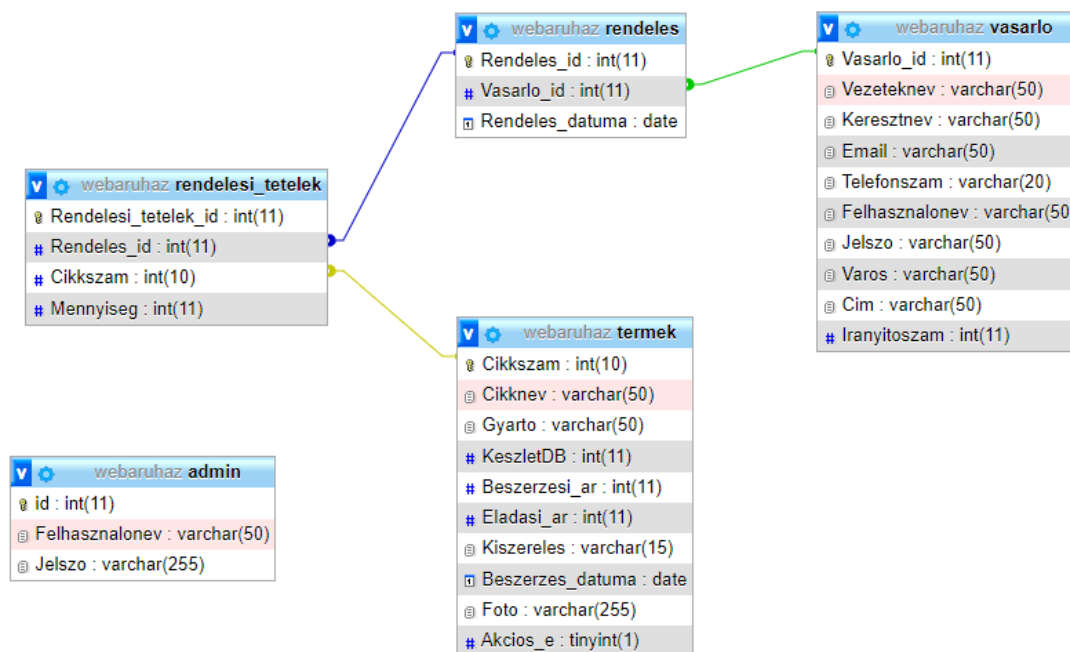
Rendelesi_tetelek tábla, tartalmazza, hogy mely cikkekből mennyit rendeltek

- Mennyiség: int(11) – Szám, rendelt termék darabszámát tartalmazza
- Rendelesi_tetelek_id: int(10) – PRIMARY KEY – Szám, Elsődleges kulcs
- Cikkszám: int(10) – Szám, idegen kulcs
- Rendeles_id: int(11) – idegen kulcs

Admin tábla, ebben a táblában a két admin felhasználóneve és jelszava van eltárolva a készletnyilvántartóba való belépéshez.

- id: int(11) – PRIMARY KEY – Szám, elsődleges kulcs
- Felhasznalonev: varchar(50) – Szöveg, admin belépéséhez szükséges név
- Jelszo: varchar(255) – Szöveg, admin belépéséhez szükséges jelszó

Az adatbázis öt táblájából egyetlen olyan tábla van amely nem kapcsolódik másik táblához, ennek oka, hogy itt az admin-ok vannak akik csak az asztali alkalmazásnál játszanak szerepet. A táblákban elsődleges kulcsként ID mezők lettek definiálva. A `rendelés` nevű táblában egy idegen kulcs is található a `vasarlo_id`, a tábla ezáltal a `vasarlo` táblához kapcsolódik. A `rendelesi_tetelek` tábla kapcsoló táblaként funkcionál, mert ezen keresztül tudunk a `termek` és a `rendeles` tábla adataival dolgozni. Ebben a táblában egy a többhöz kapcsolat van a másik két táblával, hiszen nem csak egy rendelési tétellel rendelkezünk hanem több rendelési tételünk is lehet.



4. ábra – Bachmann ábra

4. FEJLESZTŐI DOKUMENTÁCIÓ

4.1 Asztali alkalmazás (Imrő-Bárkányi Erika)

4.1.1 Alkalmazott fejlesztői eszközök, nyelvek

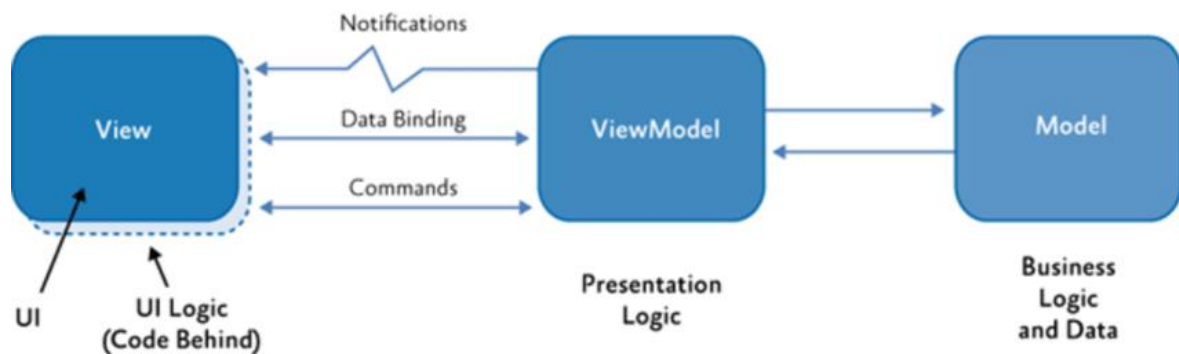
Az asztali alkalmazás elkészítéséhez a Microsoft Visual Studio Community 2019 intelligens fejlesztőkörnyezetet használtam, mely számos programnyelvet is támogat. Rengeteg bővítménnyel rendelkezik, melyek segítik a fejlesztés hatékonyságát. Ez az integrált fejlesztői környezet nagyban segítette a munkámat, hiszen vannak beépített eszközei, valamint verziókezelési lehetőségeket is biztosít. Az alkalmazásuk a WPF (Windows Presentation Foundation mozaikszavak rövidítése), keretrendszerrel készült, mely grafikus felhasználói felületek (UI) készítésére szolgál és a .NET keretrendszerrel használható. A WPF lehetővé teszi a felhasználói felület jelölőnyelvvvel történő kialakítását. A WPF jelölőnyelvének neve XAML (eXtensible Application Markup Language), mely nagyban segítik a programozók és a látványért felelősök munkásságát. Ez egy XML alapú leírónyelv, amit a HTML inspirált, de sokkal többre képes és lényegesen egységesebb.

Az alkalmazás C# (C sharp) nyelven íródott, mely egy objektumorientált nyelv, kényelmes és gyors lehetőséget biztosítva, hogy .NET keretrendszer alá alkalmazásokat

készítsünk. A C# szorosan kötődik a .NET keretrendszerhez, amitől megkapja a futtató osztályait és függvényeit, mert önálló osztály vagy függvénykönyvtárakkal nem rendelkezik. Az iskolai keretek között is ezt a programnyelvet használtuk, ezért egyértelmű volt ennek a nyelvnek a választása.

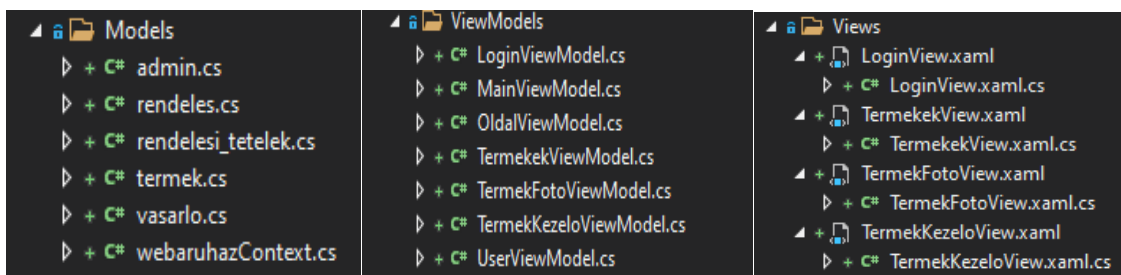
4.1.2 Asztali alkalmazás megvalósítása

Az alkalmazás objektum-orientált módon készült az MVVM (Model – View – ViewModel azaz Model – Nézet - Nézetmodel) tervezési minta segítségével. Ennek az architektúrális mintának az a célja, hogy a megjelenítést és a mögötte lévő tevékenységeket elválassza egymástól. Az alkalmazásunkat három, jól elkülöníthető részre bontja, hogy átláthatóbbá, szerkeszthetőbbé váljon.



5. ábra - MVVM struktúra (Forrás: [https://docs.microsoft.com/en-us/previous-versions/msp-n-p/gg405484\(v=pandp.40\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/msp-n-p/gg405484(v=pandp.40)?redirectedfrom=MSDN))

A **Model** tartalmazza a deklarációját az adataegységeknek (osztályok, avagy class-ok), hogy pontosan milyen adattípusokkal is fog dolgozni az alkalmazásunk. A **View** fogja tartalmazni a felületeinket, itt határozzuk meg a különböző képernyők megjelenését. Nem látja a Modelt és Model sem látja a View-t, ezért ők DataBindinggal kommunikálnak a ViewModel által. Ebből adódik, hogy a **ViewModel** a kettő összekötését oldja meg. Kapcsolatba lép a Modellel és az adatokat olyan formára hozza, amelyet a View már használni tud. Itt kerülnek példányosításra az objektumok, itt határozzuk meg a gyűjteményeket, melyeket valamilyen formában szeretnénk a felületen megjeleníteni. Az alkalmazás kódolt elemeit mappákba tettem, a könnyebb átláthatóság és a későbbi fejlesztések segítése miatt.



6. ábra - MVVM mappák és tartalmaik

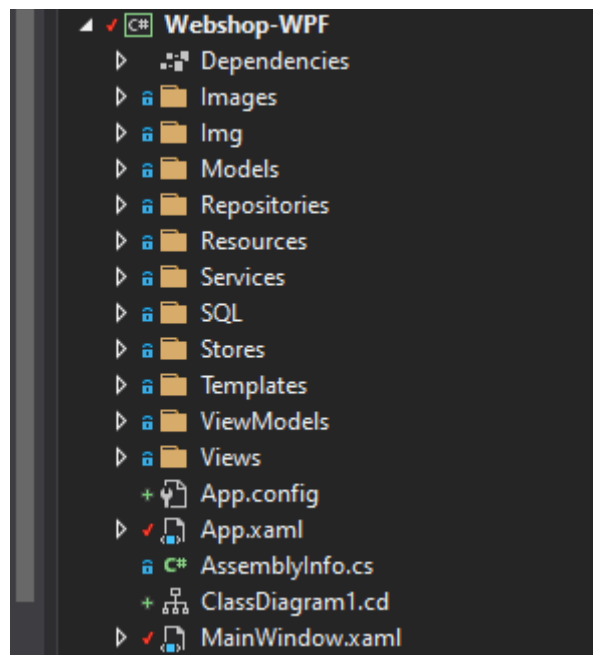
A korábban említett DataBinding vagyis adatkötésnek a célja, hogy a ViewModel-ben található példányosított objektumokat valamilyen formában megjelenítsük a felhasználó részére. Az alapértelmezett forrása az adatkötésnek a DataContext tulajdonsága. Ez nem jelenti azt, hogy minden elemnek egy DataContext-et kell használni, hanem megadhatunk más DataContext-et is, mint ahogyan a lenti ábrán is látható. Az alkalmazásban több adatkötés is létrejött, ebből mutatok egy példát. Model mappában megtalálható a termék osztály, melynek adatait szeretném megjeleníteni a felületen illetve majd módosítani. Ahhoz, hogy tükrözze az aktuális forrás állapotát, követni kell az abban történő változásokat. A forrásnak ezért meg kell valósítania az INotifyPropertyChanged interfészt, amely jelzi a felületnek, hogy a kötések frissíteni kell. A ViewModelben létrehoztam az ObservableCollection típust, amely alkalmas, változó tartalmú gyűjtemények követésére, mert ez a típus már tartalmazza az INotifyCollectionChanged interfész megvalósítását. Ez egy dinamikus adatgyűjtemény, amely értesítést küld, ha elemeket adnak hozzá vagy távolítanak el, a változás a felületen is reflektálva lesz. Ezután létrehoztam a felületet, ahol megjelenítem az adatokat (View). A View kódjában megadtam, hogy hol tároljuk a felülethez tartozó adattagokat, majd a felületi oldalán (xaml) összekötöttem.

```
<Grid Grid.Row="2">
  <Grid.ColumnDefinitions>
  </Grid.ColumnDefinitions>
  <StackPanel Grid.Column="0" Orientation="Horizontal">
    <TextBlock Text="Összesen: " FontFamily="Arial" FontSize="14" Foreground="#8e3210"/>
    <TextBlock Text="{Binding TotalItems}" FontFamily="Arial" FontSize="14" Foreground="#8e3210"/>
  </StackPanel>
</Grid>
```

7. ábra - Adatkötés (Data Binding)

Ezen felül van még más mappa és azon belül kódolt osztályok az alkalmazás megfelelő futtatásához. A **Repository** mappában azok az osztályok vannak, amelyek az adatbázisból kialakított osztályokkal közvetlen kapcsolatban állnak. Ez egy közvetítő réteg az

adathozzáférési és üzleti logika között. **Resources** mappában olyan adatok vannak amelyek erőforrásként tárolódnak lokálisan egy vezérlőhöz vagy akár a teljes alkalmazáshoz, így lehetővé téve, hogy többször is felhasználható legyen, akár több helyről. Ezek külső objektumokat jelentenek, gyakran stílusokhoz vagy sablonokhoz használják fel. Minden erőforrásunkhoz rendelhetünk egyedi azonosítót (x:Key) , amely kulccsal tudunk rá hivatkozni a StaticResource elem segítségével. Az **App.xaml** fájl ugyanúgy tartalmazhat ilyen erőforrásokat, melyek globálisan elérhetőek. A mi esetünkben itt állítottuk be, hogy melyik felület legyen a kezdőablak. A **Templates** mappában a vezérlők tetszőleges testreszabására van lehetőség. Ebben az alkalmazásban a dupla kattintás vezérlésének testreszabása található. A **Services** mappában található osztály a belépéskor eltárolt felhasználónév és jelszó tárolására szolgál. **Stores** mappa tartalma a tesztelési környezethez szükséges adatokat tartalmazza.

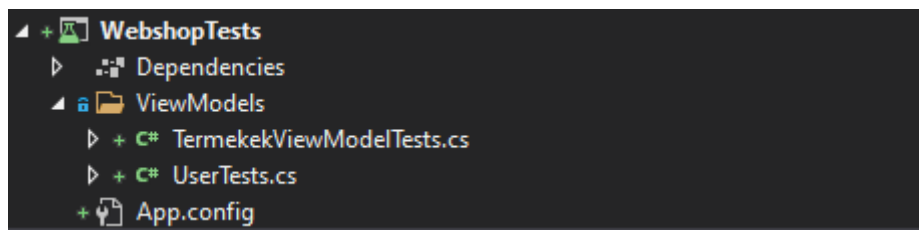


8. ábra - Asztali alkalmazás mappái

4.1.3 Tesztek

A tesztek a Visual Studio 2019-es program segítségével készültek. Teszt projektet hoztam létre, majd a projekten belül készítettem egy ViewModel mappát és ide készítettük el a teszthez szükséges osztályokat. A tesztek futtatásához a Test Explorert használtuk. A **Test Explorer** több tesztprojektből is futtathat teszteseteket, valamint olyan tesztosztályokból, amelyek az éles kódprojektek részét képezik. A tesztprojektek különböző egységteszt-

keretrendszereket használhatnak. Ha a tesztelés alatt álló kód .NET-hez van írva, a tesztprojekt bármely olyan nyelven írható, amely a .NET-et is megcélozza, függetlenül a célkód nyelvétől.



9. ábra - Teszt Osztályok

Az alkalmazás fontos részét képezi a tesztelés, mert ezekkel lehetőség van olyan hibák elkerülésére, melyek a későbbi fejlesztés során felmerülhetnek, és jelentősebb gondot okozhatnak, ezért olyan metódusok megírása volt a releváns amely a teszt vezérelt szemléletet képezi. Termékekhez és a bejelentkezéshez tartozó teszteket vettem elsőnek fontosnak.

CikkszámOK

Ebben az esetben azt várjuk, hogy amennyiben van ilyen cikkszám a terméklistában, akkor zöld jelzést adjon vissza a teszt, abban az esetben, ha nem talál a megadott cikkszámmal egyezőt a terméklistában, akkor adjon egy piros x-et a hibaüzenet kíséretével.

TermekekViewModelTest_DeleteInsertTermekTestDatabase()

A tesztet azt vizsgáljuk, hogy egy termék törlése után, a terméklistából is törlődik-e a tétel. Ez esetben a tesztnek úgy kell működni, hogy megnézi a terméklista darabszámát, majd a törlés megtörténte után, ismét megszámlolja a termékek számát. Amennyiben ugyanannyi termék maradt a listában, mint eddig, vagy az elvárthoz képest eltérés történt, akkor hibával térjen vissza. Ellenkező esetben a zöld jelzést várjuk.

TermekekViewModelTest_GetAllDataFromTestDatabase()

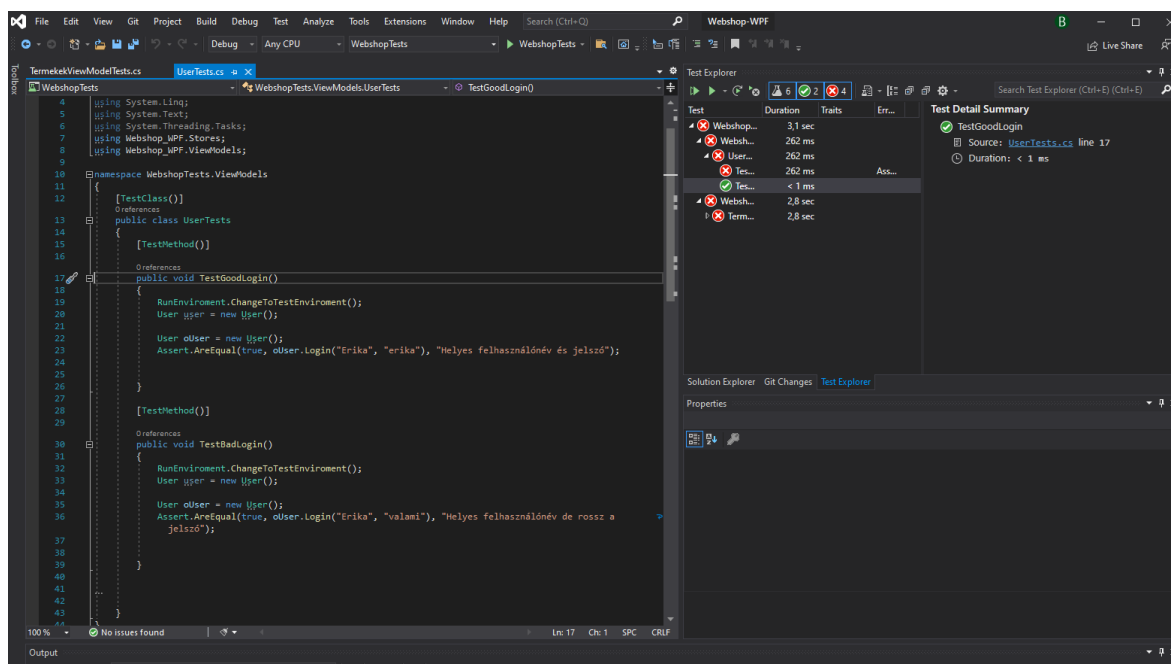
Amikor a rendszerbe belépünk, akkor a terméklista beolvasásra kerül az adatbázisból. Viszont előfordulhat, hogy nem kerül az összes beolvasásra egyéb hiba miatt. A tesztnek úgy kell működni, hogy megszámlolja a terméklistában található termékek számát majd megnézi a betöltött termékek számával, ha ez egyezik akkor sikeres eredményt mutat.

Amennyiben eltérés van a listában található és a betöltött adatok között, akkor dobjon hibát, hogy nem került minden termék beolvasásra.

TestGoodLogin()

Az adminok belépéséhez is készítettünk tesztet, amely arra ad választ, hogy a felhasználónév és az ahhoz megadott jelszó helyes-e. A megadott adatok összehasonlítása utáni eredménynek megfelelő zöld, vagy piros jelzést várunk.

A tesztjegyzőkönyvet az **1.sz. melléklet** tartalmazza.



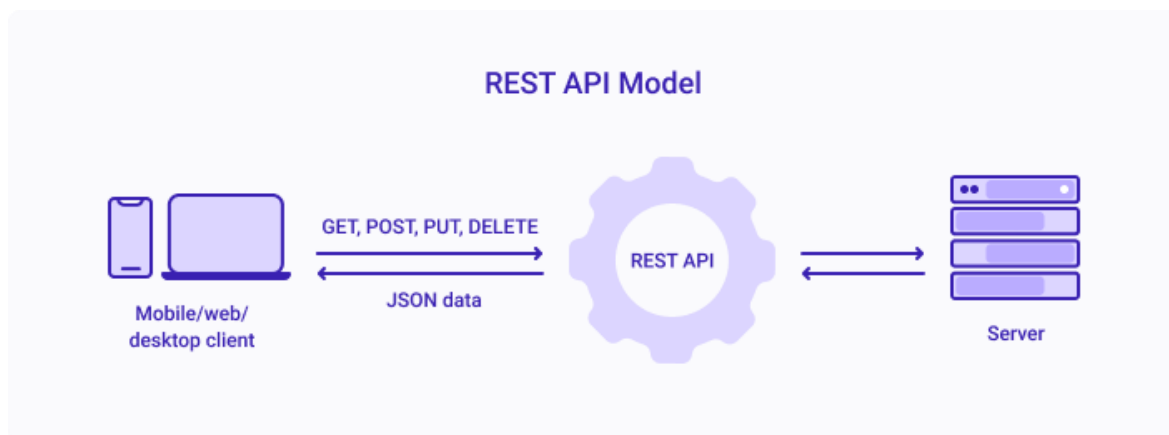
10. ábra - Tesztelési folyamatok

4.2 WebApi (Imrő-Bárkányi Erika)

4.2.1 Alkalmazott fejlesztői eszközök, nyelvek

A WebApi elkészítéséhez szintén a Microsoft Visual Studio Community 2019 intelligens fejlesztőkörnyezetet használtam, azon belül pedig az ASP.NET Core Web API projektet választottam. Az ASP.NET az Active Server Pages .NET rövidítése, mely leginkább weboldalak és webes technológiák létrehozásában játszik szerepet. Fontos eszköz egy fejlesztő számára például dinamikus weboldalak létrehozásánál. Segít a szolgáltatások felépítésében azáltal, hogy rendkívül széles körben lehet alkalmazni, mint például mobilon,

laptopon, böngészőn. Továbbá használtam az Entity Framework-öt is, mely egy objektum-relációs térképező (ORM), amely lehetővé teszi a .NET fejlesztők számára, hogy adatbázisokkal dolgozzanak .NET objektumok felhasználásával. Ez kiküszöböli a legtöbb adathozzáférési kód szükségességét, amelyet a fejlesztőknek általában meg kell írniuk. Továbbra is a C# nyelv volt számomra az evidens. Mivel az előző fejezetben már kifejtettem ezekről a főbb tudnivalókat, így most ezt nem részletezem ismét.

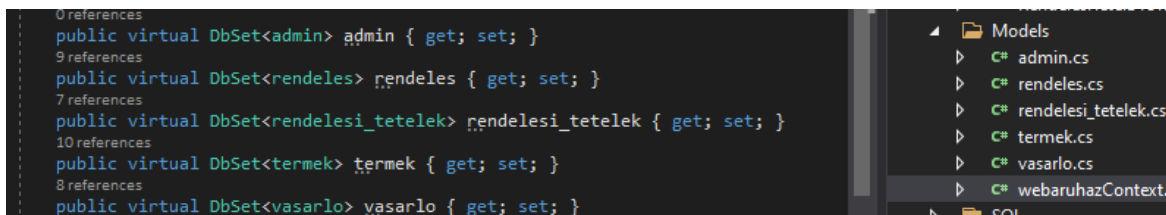


11. ábra - WebAPI kommunikáció (Forrás: <https://hevodata.com/learn/rest-api-best-practices/>)

4.2.2 WebApi megvalósítása

Az API (Application Programming Interface) egy alkalmazásprogramozási interfész azaz alkalmazások fejlesztését segítő, előre megírt függvények gyűjteménye. A frontend ezen keresztül kommunikál a backenddel. A WebShopApi fő feladata, hogy tudjon kommunikálni a weboldallal és a mobil alkalmazással. Ehhez szükséges osztályok a **Models** mappában található entity osztályok, amelyek az adatbázis táblákat reprezentálják. Az adatbázis ugyanaz, mint amit az asztali alkalmazásnál használtam. A másik fontos osztály a **DTO** (Data Transfer Object). Ezek az osztályok szemléltetik az adatbázis rekordokat az alkalmazás backend részén belül. Ezek az objektumok azok, amelyekkel a különböző rétegek metódusai dolgoznak. Az entity osztályokat azonban át kell alakítanunk DTO osztályokká, a könnyebb kezelhetőség érdekében. A legfelső rétegünk a jól ismert **Controller** réteg lesz. A Controller a felhasználói interakciókat dolgozza fel olyan formában, hogy tovább irányítja a modellt felé a kérést. A választ értelmezi, majd a megfelelő formában visszaadja a nézetnek. Itt találhatóak az úgynevezett endpointok vagy másképp végpontok, amikre az alkalmazás frontend része kapcsolódni fog. Ezekkel a végpontokkal tudunk backend tesztelést végezni a Postman alkalmazással. Az egyik legfontosabb osztály

a DbContext osztály, amely olyan munkamenetet reprezentál, amely az alatta lévő adatbázis segítségével alapvető adatkezelési műveleteket hajt végre. Ezek az osztályok az adatok lekérdezésére, az adatbázisba történő mentésre is szolgál.



12. ábra - Adatbázis táblák leképezése

Az alapvető adatkezelési műveletek vagy másképp CRUD műveletek a webszolgáltatások felhasználásához tartoznak. Ezek a létrehozás (*Create*), olvasás (*Read*), módosítás (*Update*), törlés (*Delete*). A műveleteknek adott a http megfelelője, azaz a létrehozás (*Post*), olvasás (*Get*), módosítás (*Put*), törlés (*Delete*). A metódusok felhasználásával az erőforrásokon végrehajtott műveleteket támogatja az API.

- GET - A megadott erőforrás letöltését kezdeményezi
- POST - Feldolgozandó adatot küld fel a szerverre, például egy űrlapot.
- PUT – Feltölti a megadott erőforrást
- DELETE – Törli a megadott erőforrást

A válasz kódja létrehozás esetén a *created* (201), többi művelet esetén *ok* (200), vagy *no content* (204). A műveleteknek ehhez a sémához kell alkalmazkodnia.

A kliens oldalon adatkezelést kétféleképpen tudjuk megvalósítani:

- Szinkron módon: a kliens és a szerver állapota megegyezik
- Aszinkron módon: a kliens és a szerver állapota eltér, és manuálisan szinkronizálható.

4.2.3 Tesztek

A WebApi – ban készítettem pár egységtesztet, hogy megtudjam, hogy a metódusok, amelyeket használunk jól működnek-e. Ebben az esetben is a Test Explorer segítségével futtattam le a teszteket. Készítettem egy *TermekTesztek* nevű osztályt, ahol kiválasztottam egy terméket az adatbázisból, valamint, hogy *termékController* legyen használva. A termék controller került tesztelésre. A tesztelés során arra voltam kíváncsi, hogy a termékek

lekérésekor a megfelelő darabszámú terméket adja-e vissza a lekérés valamint, hogy cikkszám alapján a helyes terméknev kerül-e feltüntetésre a GET használata során.

```
[TestClass]
References
public class TermekTesztek
{
    private termékController controller = new termékController(new webaruhazContext());

    3 references
    private termék GetPeldaTermek()
    {
        return new termék()
        {
            Cikkszám = 88326,
            Cikksnev = "Ammerland trappista 500g",
            Gyarto = "Ammerland"
        };
    }

    // GET: api/Termek, Darabszám ellenőrzése
    [TestMethod]
    References
    public async Task GetTermek_DarabszamMegfelel()
    {
        // Http Response üzenet
        var response = await controller.Gettermek();
        // IEnumerable a válasz
        var result = response.Value;
        // Listává alakítjuk
        var result2 = response.Value as List<termek>;

        Assert.IsNotNull(result);
        int szamlalo = 78;
        Assert.AreEqual(result.Count(), szamlalo);
        Assert.AreEqual(result2.Count, szamlalo);
    }
}
```

13. ábra - Teszt Osztály és teszt metódus

A POST esetben is készült teszt, mégpedig, hogy amikor a POST metóduson keresztül megpróbálnak létrehozni egy terméket amely már létezik, akkor 409-es ütközés hibát ad-e vissza StatusCode eredményként, jelezve, hogy a hozzáadás duplikáltnak minősül. A PUT kérés esetében azt vizsgáltam, hogy ha egy módosítást rossz ID-ra próbálok küldeni, akkor visszaadja-e a BadRequestResult választ, azaz jelzi, hogy rossz kérés történt.

Test	Duration	Traits	Error Message
WebS...	1,5 sec		
Web...	1,5 sec		
Te...	1,5 sec		
G..	1,3 sec		
G..	161 ms		
P..	28 ms		
P..	36 ms		Assert.IsInstanceOfTy...

Group Summary

WebShopAPI_Teszt

Tests in group: 4

Total Duration: 1,5 sec

Outcomes

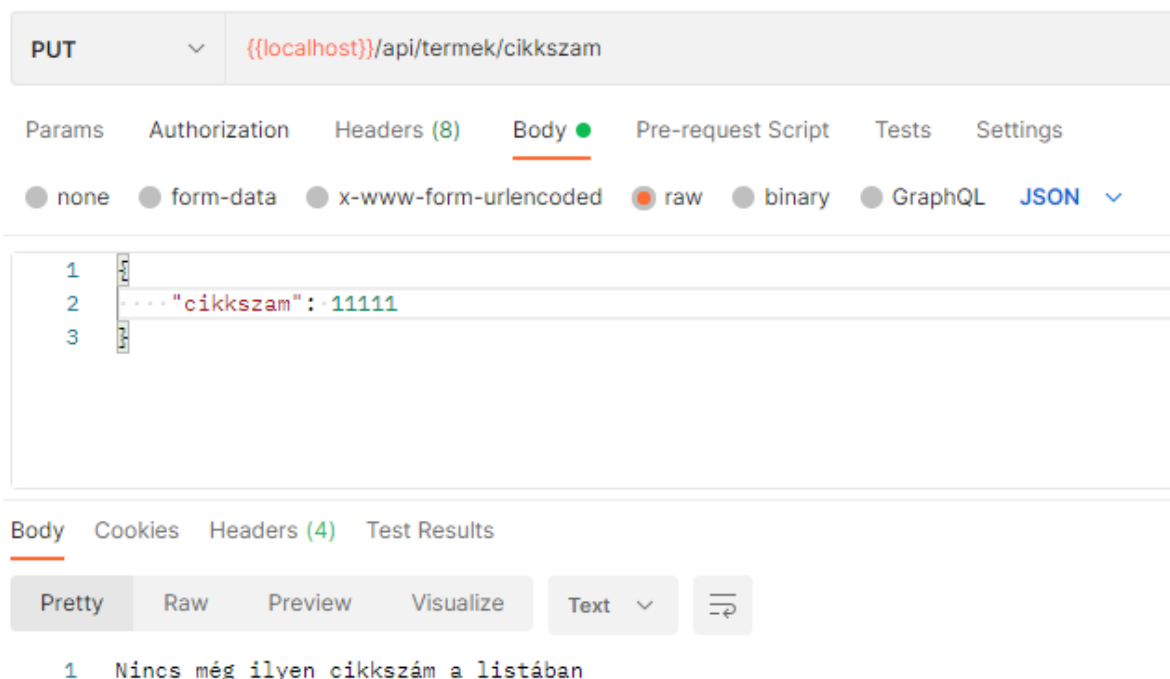
3 Passed

1 Failed

14. ábra - Teszt futtatás

Természetesen a kliens oldali tesztelés sem maradhatott el. Itt arra voltam kíváncsi, hogy a megfelelő válaszokat kapom-e a különböző hívások esetén. Ezt a tesztelést már a **Postman** program segítségével végeztem. A program rendkívül egyszerűen kezelhető, kéréseket készíthetünk vele és kipróbálhatjuk a visszajelzéseket. Ezeket egy gyűjteménybe kiexportálva később is felhasználhatjuk, megtekinthetjük.

Ezekben a tesztekben többféle controllert is vizsgáltam, hogy a weboldal által küldött kérésekre milyen választ fog kapni. Például a TermekControllernél azt vizsgáltam, hogy mielőtt létrehozunk egy terméket, van-e már azzal a cikkszámmal másik termék.



15. ábra - PUT - Cikkmódosítás

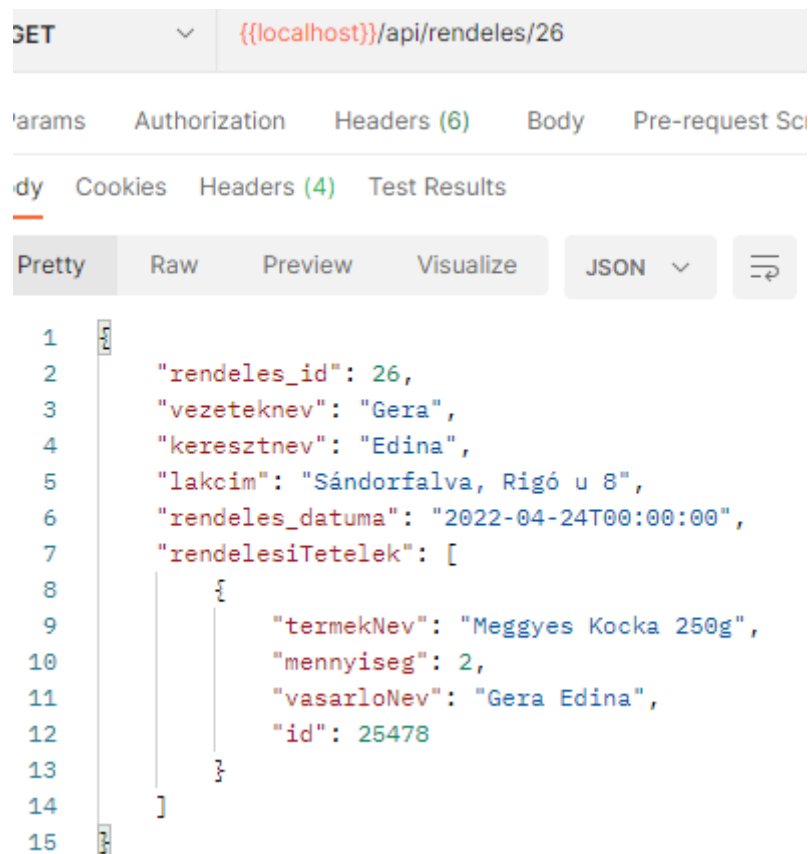
A POST kérések esetén az adatbázisba tárolódnak az adatok. Ez több esetben is előfordul, hiszen ha egy megrendelés beérkezik, akkor tárolásra kerül a rendelés_id-ja valamint az azon található tételek is. Alább látható egy POST kérés, mely egy rendelést küld:

```
{
  "vasarlo_id": 7,
  "rendelesi_tetelek" : [
    { "cikkszam": 25478,
      "mennyiseg" : 2 }
  ]
}
```


Majd az erre kapott json válasz:

```
{
  "rendeles_id": 26,
  "vasarlo_id": 7,
  "rendeles_datuma": "2022-04-24T17:33:05.8911739+02:00",
  "rendelesi_tetelek": [
    {
      "rendelesi_tetelek_id": 12,
      "rendeles_id": 26,
      "cikkszam": 25478,
      "mennyiseg": 2,
      "termek": null
    }
  ]
}
```

Egy GET-el lekérem a fenti rendelést és az alábbi kapom:



16. ábra - GET lekérés

További tesztek a WebshopApi.postman_collection –ben találhatóak a vizsgaremekhez csatolva.

4.3 Mobil alkalmazás (Szabácsi Noémi)

4.3.1 Alkalmazott fejlesztői eszközök, nyelvek

A mobil applikáció elkészítéséhez Microsoft Visual Studio Code fejlesztőkörnyezetet használtam, mely egy nyílt forráskódú kódszerkesztő, beépített eszközökkel. Lehetőséget nyújt, hogy több különböző, felhő alapú szolgáltatást, mint például a GitHub vagy Azure, azonnal elérjünk, valamint a bővítmények segítségével folyamatosan kiegészítsük készülő programunkat.

A felhőprojektjeink létrehozását pillanatok alatt futtathatjuk, azonnali hibakereséssel, mely lehetőséget nyújt a különböző eszközökre való konfiguráláshoz, valamint tesztek is futtathatunk bennük.

A jelenleg létrehozott natív mobil alkalmazás **TypeScript** nyelven íródott, Ionic React keretrendszerben. A TypeScript egy olyan JavaScript nyelv, amely megegyező funkciókkal rendelkezik, azonban a gépelési rendszer miatt fejlesztőbarát. A különbség legelragadóbb része, hogy míg egy JavaScript kódban a „string” vagy „number” típusú adattagokat a nyelv nem ellenőrzi, hogy megfelelően hozzárendeltük-e egymáshoz, addig a TypeScript mindent átvizsgál, azonnal visszajelez, ha valamit nem megfelelően definiáltunk, ezzel egyidőben a lehetséges bugokat (hibákat) kikerülhetjük.

Az **Ionic** egy olyan nyílt forráskódú eszköztár, amely a webes technológiákat használva kombinálja ismert keretrendszerek integrációit. Az alkalmazás telepítésekor egy olyan natív projekteket szolgáló futási időt (Run Time) is telepítünk, mely lehetőséget nyújt arra, hogy a készített applikáció több eszközön is futatható legyen, ami nem más mint a Capacitor. Ennek segítségével jelentősen egyszerűbb a lehetőségünk, hogy egy API-ra csatlakozzunk. A **React** keretrendszer lehetőséget nyújt, hogy az ott elkészített komponenseinket felhasználjuk és rugalmasan kezeljük őket, valamint módosítsuk és bővítsük. A hibák keresésére az ionic-react compilert, valamint Opera DevToolst használtam.

4.3.2 Mobil alkalmazás megvalósítása

Az alkalmazás Web API kérések mellett lokális adatokkal is dolgozik, melyet egy termék részletes adatainak kiírására használtam. Az aszinkron kérései között GET és POST kapja a főszerepet, amit a RESTful alkalmazás szolgáltat a projektnek a „localhost:5000/api/” címnek.

```
export const byCikkszam = async (cikkszam: string): Promise<IProductData | undefined> => {
  const url: string = `${baseUrl}termek/${cikkszam}`;
  const termekData: IProductData | undefined = await GET<IProductData>(url);
  return termekData;
}

export const getAll = async (): Promise<IProductData[]> => {
  const url: string = `${baseUrl}termek`;
  const termekData: IProductData[] | undefined = await GET<IProductData[]>(url);
  return termekData ? termekData : [];
}
```

17. ábra - API kérés

A termékek listája úgy lett kialakítva, hogy ne töltsen be az összes adatot, hanem csak egy meghatározott számot. Ha a jelenlévő termékek meghaladnák a maximális számot, amit kiírhatnak egy oldalra, akkor rövid betöltést követően az oldal folytatná a sort, azonban ezt teljes mértékben tiltottam, és ebben az esetben csak tíz sort írtam ki. A feltöltés lehetőségét egy kijelölőnégyzet adja, amit lokálisan tárol, mindaddig, amíg a felhasználó nem lép be egy részletes leírás oldalára. A tárolóban a rendszer a termék cikkszámát helyezi el egy üres tömbbe, valamint a mennyiség mindig egy marad, ez sosem növelhető.

```
const TermekComponent: React.FC<IProps> = (props: IProps): JSX.Element => {
  return (
    <><IonCheckbox className="checkbox" key={props.termek.cikkszam} checked={false} onIonChange={() => {
      props.onChange(props.termek.cikkszam);
    }}></IonCheckbox>
    <IonItem className="listcontent" routerLink={` /termek/${props.termek.cikkszam}`}>
      <IonAvatar slot="start">
        <img src={props.foto} alt={props.termek.cikknev} />
      </IonAvatar>
      <IonLabel>
        <h2>{props.termek.cikknev}</h2>
      </IonLabel>
    </IonItem></>
  );
}

export default TermekComponent;
```

18. ábra - Lokális adatok kérése

A modelben elhelyezett adatokat, amit helyből ér el az alkalmazás a „cikkszám” jelenlétével tudja beazonosítani, hogy az adott leírás mely termékhez is tartozik. Ha a cikkszám megegyezik a listában szereplő számmal, akkor a rendszer hozzárendeli a leírást, illetve az árat, amit külön kéréssel, az API-ból szolgáltat a termék nevével együtt. A képi megjelenés és a termékleírás helyileg töltődik be, így ha a Restful alkalmazásunk nem fut, ezek akkor is megjelennek.

```
export const termekdetail: IProductDetail[] =
[
  {
    "cikkszam": 2467,
    "sourceUrl": "https://cdn1.interspar.at/cachablesevents/articleImage.dam/hu/59063001/mb_sub.jpg",
    "description": "Étkezési só, Zöldségszármányok 15,5% (sárgarépa, paszternák, burgonya, vöröshagyma), 8 db friss tyúktorzs/1 kg késztermék (36%)"
  },
  {
    "cikkszam": 4197,
    "sourceUrl": "https://secure.ce-tescoassets.com/assets/HU/023/5997132504023/ShotType1_540x540.jpg",
    "description": "Tésztaipari búzaliszt (glutént tartalmaz), 8 db friss tyúktorzs/1 kg késztermék (36%)"
  },
  {
    "cikkszam": 4908,
    "sourceUrl": "https://secure.ce-tescoassets.com/assets/HU/023/5997132504023/ShotType1_540x540.jpg",
    "description": "100% búzaliszt (glutén)",
  },
  {
    "cikkszam": 5583,
    "sourceUrl": "https://cashmap.hu/storage/product/617989623292d.jpg",
    "description": "Só, Keményítő, Növényi zsírok (pálma, shea, sal), Élesztő kivonat, Cukor, 5,9% füstöl"
  },
  {
    "cikkszam": 8287,
    "sourceUrl": "https://secure.ce-tescoassets.com/assets/HU/229/5948935000229/ShotType1_540x540.jpg",
    "description": "Gluténmentes tészta kukoricából. Kagyló (Conchilietto)",
  },
  {
    "cikkszam": 8289,
    "sourceUrl": "https://static.egeszsegbolt.hu/images/upload/product/image/original/c0/ba/pasta_doro_ki",
    "description": "Gluténmentes tészta kukoricából.",
  },
]
```

19. ábra - Lokális adatok

4.3.3 Tesztek

A tesztelésben elsősorban a TypeScript és az **Opera DevTools** segített az egész projektet végig követve és segítve. A lekérések ellenőrzését a böngésző konzol paneljában, valamint a hálózat visszajelzése alapján figyeltem. Ezeket megelőzően pedig, a **Postman** alkalmazásban vizsgáltam meg, hogy a lekérések megfelelően működnek-e egy letisztult környezetben.

```
import { render, screen } from '@testing-library/react';
import HomePage from "../home.page"

test('HomePage should have a title of „Termékeink listája„', async () =>
{
  render(<HomePage />);
  const cardTitleText: HTMLElement = screen.getByText("Termékeink listája");
  expect(cardTitleText).toBeInTheDocument;
});
```

20. ábra - Mobil Teszt

A lista képernyőjének szalagcímét külön terminálos tesztben is elkészítettem, mely során az alkalmazás kivetíti a megadott képernyőt, és megkeresi az említett szöveget. Amennyiben ez megjelenik, a teszt sikeres, azonban hiba esetén a konzol panelben jeleníti meg a hibaüzenetet.

4.4 Webes alkalmazás (Szabácsi Noémi)

4.4.1 Alkalmazott fejlesztői eszközök, nyelvek

A webes alkalmazás elkészítéséhez, hasonlóan a mobilhoz, Microsoft Visual Studio Code fejlesztőkörnyezetet használtam, valamint Vue.js keretrendszert. A webes klienseket kezelő rendszerben akár komponenseket is használhatunk, mely jelentősen segíti a fejlesztők hatékony munkáját egy komplex vagy egyszerű weboldal elkészítésében. A Vue.js lehetőséget nyújt arra, hogy a frontendben elsajátított, statikus HTML kódokat beépítsük ebbe a JavaScript frameworkbe. Ezen felül akár egylapos megjelenítendő oldalakat, mobil applikációkat, sőt, akár terminálokat is létrehozhatunk a Vue.js segítségével.

Ezen kiválasztott rendszerünkbe különböző csomagokat telepíthetünk, többek között a HTML-ből már ismert Bootstrapet, vagy a népszerű Vuetifyt. A bővítmények segítségünkre lehetnek a programozásunk során, hogy akár a megjelenítési kivitelezéseket sokkal egyszerűbben és színesebben valósítsuk meg, rövid idő alatt.

Web API kommunikációra több módszert is alkalmazhatunk, mely lehet többek között a „fetch” valamint az Axios bővítmény. A komponensek összefűzésével, hasonlóan a Reacttal, egyszerűen hivatkozhatunk már előre megírt metódusokra, és importálhatjuk be őket, vagy helyben adjuk le kérésünket a RESTful alkalmazásunkhoz.

4.4.2 Webes alkalmazás megvalósítása

Az alkalmazás két fő oldalból épül fel, mely regisztráció nélkül is megtekinthető. A reszponzivitást a dizájnelemek segítségével valósítottam meg, bővítmények segítségével, a HTML kódokból megismert „@media” használatával. Az oldalt telefonos képernyőmérethez, valamint táblagépek átlagos kijelző méretéhez igazítottam, így megadva a lehetőséget a sokoldalú elérhetőséghez. Alapesetben nem szükséges teljeskörűen átszerkeszteni az elemeinket, egyedül az elhelyezkedésük és a méretezésük fontos, ha manuálisan kívánjuk elérni a reszponzív megjelenítést.

```
@media (max-width: 768px) {  
  header .fa-bars {  
    display: block;  
  }  
  
  header .navbar {  
    position: absolute;  
    top: 100%;  
    left: 0;  
    right: 0;  
    background: #1a1a1a;   
    border-top: 0.1rem solid #1a1a1a;   
    clip-path: polygon(0% 0%, 100% 0%, 100% 100%, 0% 100%);  
    padding: 50px;  
  }  
}
```

21. ábra - Reszponzivitás kódja

Az oldal sajátossága a menü változása, ahogy a képernyőméret módosul. Elérve a kiadott értéket (esetemben ez 768 pixel szélesség), a menüsor hamburger-menübe vált, és legördülő oszlopba rendeződik. A sor rendelkezik egy, gombnyomással aktiválódó elemmel is, mely két újabb menüt jelenít meg, ami új oldalra viszi el felhasználót, jelen esetben a projektben használt termékhez.



22. ábra - Lenyíló menü

Az applikáció fő funkciója a webshopokban szereplő Kosár elérhetősége. Jelenleg egy terméken tudjuk alkalmazni a vásárlás lehetőségét, de a jövőbeni tervek közé tartozik, hogy ezt az adatbázisban több, vagy akár az összes megjelenő termékre is kiterjesszük. A kiválasztott áru egy gomb lenyomását követően külön tárolja, és jeleníti meg a kívánt mennyiséget, mely kattintás után folyamatosan összeadódik, törlés esetén pedig eltűnik. A regisztráció hiánya miatt jelenleg a kívánt terméket és mennyiségét a rendszer eltárolja, rögzíti, a Web API pedig visszajelez a felhasználónak, hogy felrögzítette a rendelést, az adatok pedig a „fetch” Post metódusával rögzülnének az adatbázisban.

```
addToCart(cikkszam, darabszam) {  
  console.log(cikkszam);  
  this.rendeles.push({  
    cikkszam: cikkszam.cikkszam,  
    mennyiseg: darabszam,  
    megnevezes: cikkszam.cikknev,  
    beszerzesi_ar: cikkszam.beszerzesi_ar,  
  });  
  this.osszertek += cikkszam.beszerzesi_ar * cikkszam.mennyiseg;  
},  
removeFromCart(cikkszam) {  
  this.rendeles.splice(this.rendeles.indexOf(cikkszam), 1);  
  this.osszertek -= cikkszam.beszerzesi_ar;  
},
```

23. ábra - Kosárba tesz - töröl funkciók

Az oldalon szereplő termék neve és egységnyi ára a RESTful alkalmazásból érkezik, így enélkül rendelést leadni lehetetlen. Hasonlóan a mobil alkalmazáshoz, a leírást lokálisan tároltan jelenítettem meg, így az API nélkül is megjelenik. Az oldalak közti navigációt a Vue.js Router tette lehetővé, mely egy beépített JavaScripten fájlban belül specifikált irányadó kódokat tartalmaz, amit később egy „router-link” paranccsal hívtam meg

különböző elemekhez, többek között a „Főoldal” menüponthoz, vagy a főoldal alján található kiemelt termék nevéhez.

```
const routes = [
  {
    path: "/",
    name: "Home",
    component: Home,
  },
  {
    path: "/Home",
    name: "Home",
    component: Home,
  },
  {
    path: "/WebS",
    name: "WebS",
    component: WebS,
  },
];
```

24. ábra - Router

4.4.3 Tesztek

Az Opera DevTools segítségével ellenőriztem, hogy az alkalmazás és a Web API között megfelelően épült-e fel a kapcsolat, valamint azt, hogy a gombnyomások után megtörténnek-e a szükséges funkciók. A Postmanben megírt metódusok lehetőséget nyújtottak ahhoz, hogy a lekérések, valamint az adatfelküldések segítségével szolgáljanak, így megkönnyebbítve a fetch metódusok megírását, és az adatok definiálását.

```
describe('Post tesztelés', () => {
  test('Termékek elhelyezése a kosár tartalmába', async () => {
    const wrapper = mount(Post)
    const select = wrapper.find('select[name=megnevezes]')
    await select.setValue('Málnalekváros fánk')
    expect(wrapper.vm.megnevezes).toBe('Málnalekváros fánk')
  })
})
```

25. ábra - Post teszt

Manuális tesztben azt vizsgáltam meg, hogy az általam kiválasztott termék a kiválasztás során, az API-ban használt cikknév szerint azonosí

tja-e be az árut. Az általam megírt „fetch” metódusban külön azonosítóval ellátott „megnevezes” megtalálását követően az oldal felismeri a terméket, így leellenőrizheti, hogy az adatbázisban történt-e változás.

5. FELHASZNÁLÓI DOKUMENTÁCIÓ

5.1 Asztali alkalmazás bemutatása (Imrő-Bárkányi Erika)


Az asztali alkalmazás futtatásához elengedhetetlen az adatbázis használata. Ehhez a XAMPP nevű program feltelepítése majd az Apache és MySQL modul indítása szükséges. A webáruház adatbázisát a *webaruhaz.sql* importálásával tudjuk létrehozni. Az asztali alkalmazás elindítható a *Futtatható* elnevezésű mappában található *Webshop-WPF* alkalmazással, vagy ha rendelkezik gépünk a Visual Studio programmal, akkor a *Webshop-WPF.sln* kiterjesztésű fájlra klikkelve.

Elindítva az alkalmazást, a bejelentkező felülettel találkozunk. Itt meg kell adni a dolgozó felhasználónevét és jelszavát. Amennyiben helytelen adatot adott meg, vagy esetleg az adatbázis nem került elindításra, abban az esetben egy hibaüzenetet kapunk, helyes adat megadásakor és működő adatbázis esetén pedig beléphetünk a következő felületre.



26. ábra - Belépési felület hibaüzenettel

A következő oldalon a terméklistát találjuk, mely az adatbázisból kerül meghívásra. Ezek a termékek az Ica Bolt termékei. A bal felső sarokban látható, hogy jelenleg hány darab terméket tartalmaz a lista valamint, hogy hány darab termék jelenik meg egy oldalon. Ez utóbbi módosítható. A lap alján lapozó gombokat találunk, amellyel lehetőség van a következő oldalra vagy egyből az oldal végére jutni. A felületen megtalálható az *Új termék felvétele* gomb, amelyre ráklikkelve, egy új ablakot kapunk. Ebben a megnyíló ablakban lehetőségünk van új termék felvételére, melyhez meg kell adni az áru néhány adatát, majd a mentés gombra kattintva, megtalálható lesz a terméklistában. Amennyiben már meglévő termék tulajdonságát szeretnénk módosítani, abban az esetben a kiválasztott termék sorára klickelve kapjuk meg az új ablakot, amely már tartalmazza a kérdéses termék adatait. A változtatandó tulajdonság átírása után, a mentés gombra klickelve, módosulni fog a termék adata. A *Keresés* gomb segítségével hamarabb meg tudjuk találni a cikket. Az oszlopokra kattintva rendezhető táblázatot kapunk. A sorba rendezést úgy ajánlatos megtenni, hogy mi szerint szeretnénk látni az adatainkat. Ha például a beszerzés dátumát szeretnénk növekvő sorrendben látni, akkor a beszerzési dátum oszlop fejlécére kattintva tudjuk megtenni. Ismételt kattintással pedig csökkenő sorrendbe rendeződik az adat.



Cikkszám	Cikknév	Darabszám	Akció	Beszerzés Dátuma
2467	Podravka vegeta 1kg	15	<input type="checkbox"/>	2021-10-25
4197	Gyermelyi eperlevél 8 tojasos 250g	45	<input type="checkbox"/>	2021-10-28
4908	Cba finomliszt 1kg	78	<input type="checkbox"/>	2021-11-07
5583	Knorr leveskocka 80g	22	<input type="checkbox"/>	2021-11-16
8287	Gluténmentes tészta kagyló 500g	25	<input type="checkbox"/>	2021-10-25
8289	Gluténmentes tészta kiskocka 500g	17	<input type="checkbox"/>	2021-10-28
10643	Knorr fűszervázás fasírt 40g	16	<input type="checkbox"/>	2021-10-29
10644	Knorr fűszervázás sertéssült 30g	31	<input type="checkbox"/>	2021-11-15
11702	Újházi csigatészta 8 tojasos 200g	36	<input type="checkbox"/>	2021-11-18
12571	Szatmári finomliszt 1kg	52	<input checked="" type="checkbox"/>	2021-10-27
14909	Vitamill finomliszt 1kg	58	<input type="checkbox"/>	2021-11-04
14910	Vitamill grahmliszt 1kg	41	<input type="checkbox"/>	2021-11-09
15955	Biopont rizsliszt 500g	18	<input type="checkbox"/>	2021-11-05
16649	Lisztkeverék teljes kiőrlésű 5kg	4	<input type="checkbox"/>	2021-11-09
16657	Lisztkeverék sötét magvas 2kg	9	<input type="checkbox"/>	2021-11-05
16659	Lisztkeverék rozsos 5kg	7	<input type="checkbox"/>	2021-11-02
21033	Farmer tej 1,5% 1l	25	<input type="checkbox"/>	2021-11-09
22619	Lucullus gastro feketebors őrölt 250g	6	<input type="checkbox"/>	2021-10-30
25314	Izsáki Durum Szarvacská 500g	38	<input checked="" type="checkbox"/>	2021-10-31
25467	Eszterházi Szelet 250g	40	<input type="checkbox"/>	2021-10-25

27. ábra - Készletkezelő felület

A *Törlés* gombbal a kiválasztott termék sora kerül kitörlésre, ez értelemszerűen az adatbázisból is kitörlődik. A *Termékfotók* gomb egy újabb ablakot nyit meg, amelyben egyelőre azok a fotók láthatóak, amelyek a bolt szórólapján megtalálhatóak. Ezek sajnos

többször kerültek megsemmisülésre a promóciós lap készítésekor és így okozott némi problémát az ismételt kép elkészítése. A képek egy mappában vannak (Image) eltárolva, ez természetesen bővíthető a kérésnek megfelelően. Jelenleg nem tölt be más funkciót a bolt életében.



28. ábra - Termékfotók

Készletkezelés

Mentés

Cikkszám:

5583

Cikknév:

Knorr leveskocka 80g

Darabszám:

22

Akciós termék:

☐

Beszerezés dátuma:

2021. 11. 16.

15

Bezárás

29. ábra - Termék módosítása

A Készletkezelés felületen található *Bezárás* gomb megnyomásával a kezdőfelületre jutunk, míg a *Kijelentkezés* gombra klikkelve vissza tudunk térni a bejelentkező felületre.

5.2 WebApi bemutatása (Imrő-Bárkányi Erika)

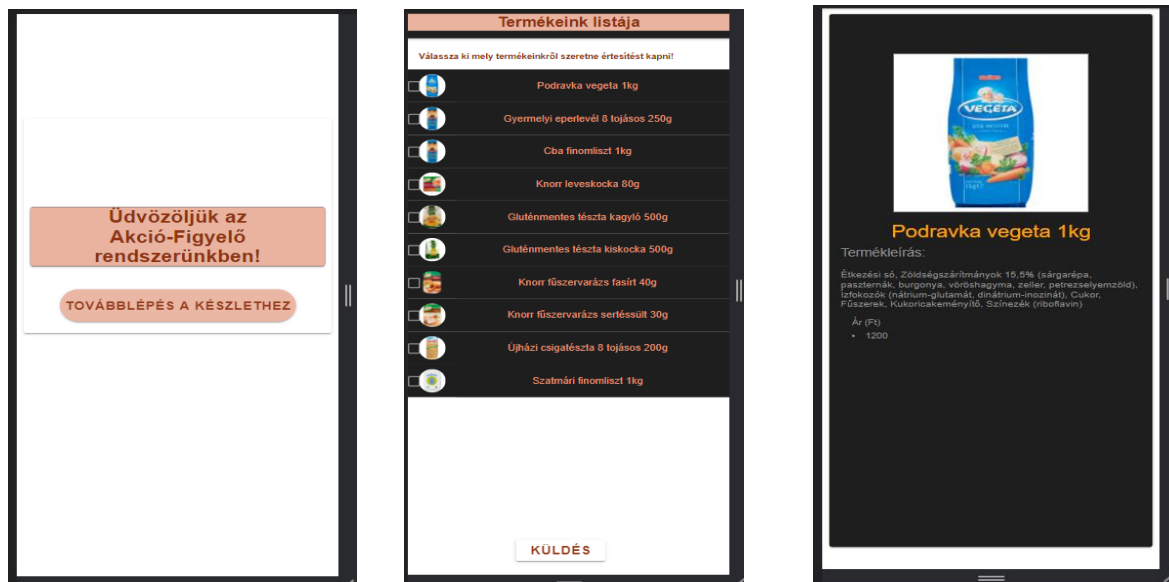
A WebApi elindítása szükséges a weboldal és a mobil alkalmazás használatához. A számítógépre telepített Visual Studio megnyitása, majd a WebShopAPI mappában található WebShopAPI.sln fájlra való dupla kattintás után tudjuk elindítani. Mivel az alkalmazások az adatbázist is igénylik, ezért a korábban már leírt módon azt is futtatni kell. A böngészőből vagy mobilról elindítunk egy POST vagy GET kérést, mint amit bármely más szerveroldali platform képes kezelni. Ezt a kérést a túloldalon a WebApi fogadja, feldolgozza és visszaad egy Json (kis méretű, szöveg alapú szabvány) választ, amit a kliens oldal feldolgoz.

5.3 Mobil alkalmazás bemutatása (Szabácsi Noémi)

Az applikáció elsőrangú célja, hogy a vásárló számára lehetőséget adjunk arra, hogy a listában szereplő termékek között kedvére válogasson, és ha úgy gondolja, hogy a kiválasztott cikk érdekli, visszajelzést kérhet a bolttól, hogy az mikor válik akcióssá.

Alkalmazásomban azonban nincs szükség regisztrációra, ugyanis ezt a lehetőséget kifejezetten törzskártyás vásárlóink eszközeire telepítem, így az azonosítás megtörténne a felkerülés pillanatában, azonban ez a szakasz még egy fejlesztés alatti elképzelés.

A főoldal letisztult jellege egyértelművé teszi a felhasználónak, hogy a megjelent gomb megnyomásával lehetősége van megtekinteni az Ica bolt által szolgáltatott termékeket.



30. ábra - Mobil oldalak

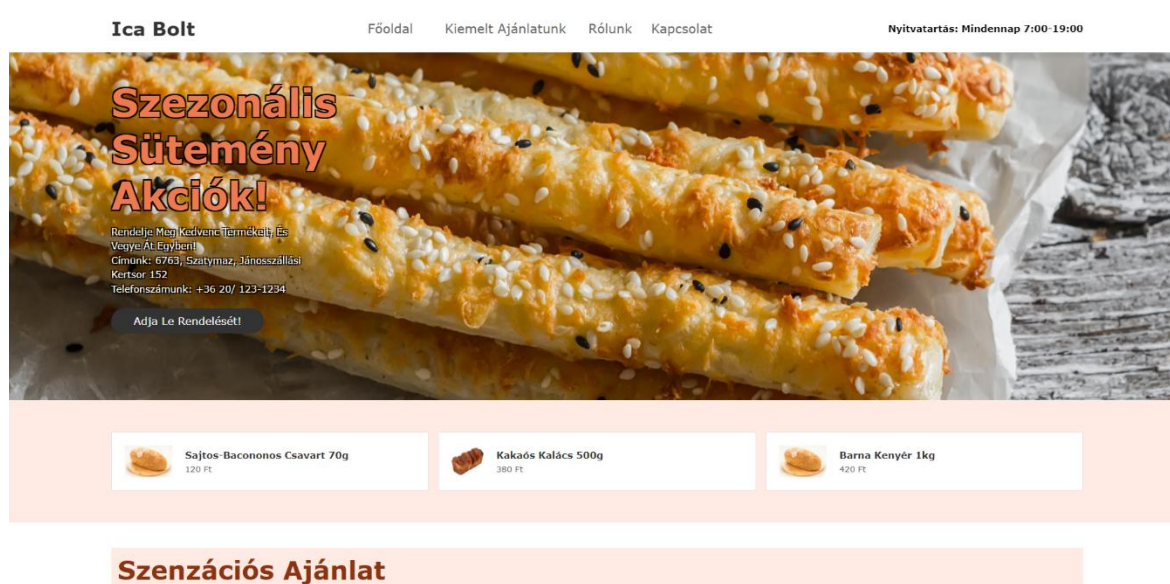
A lista egyelőre tíz elemet tartalmaz, mindegyik egyéni képpel, névvel, valamint leírással és értékkel rendelkezik. A funkciók függenek attól, hogy a vásárló az áru nevére kattint, vagy a termék képe előtt található kijelölőnégyzetre. A négyzetre klikkelve megjelenik egy pipa az adott termék előtt, amit természetesen akár az összes cikk esetén is megtehet a vásárló. Ha végzett a számára megfelelő termékek kiválasztásával, a „Küldés” gomb lenyomásával kérést küldhet a bolt felé, hogy a kijelölt termékekről szeretne értesítést kapni a jövőben. A Web API azonnali visszajelzést ad a felhasználónak egy felugró ablakban a sikeres rögzítésről. Ha a felhasználó nem jelöl ki semmit, és így ráklikkel a küldés gombra, akkor a rendszer egy hibajelzést küld neki, hiszen előfordulhat, hogy a kiválasztás esetleg véletlenül nem történt meg.

Amennyiben a termék nevére történik a kattintás, akkor a választott áru részletes leírása, képe, és ára jelenik meg. A termékismertető több célt is szolgál, mint például a vásárló előtt még ismeretlen termék alaposabb megismerése, a későbbi vásárlás céljából.

5.4 Webes alkalmazás bemutatása (Szabácsi Noémi)

A világjárvány kitörésekor létrejött kezdetleges, ám szükséges társadalmi lépések adtak inspirációt ahhoz, hogy egy olyan online bevásárlásra alkalmas oldalt hozzak létre, amely egy alig ötezer fős falu számára megkönnyébbítené és rugalmasabbá tenné a mindennapi szükséges ételárak beszerzését. A bejelentkezés nélküli webes alkalmazás biztosítja, hogy bárki összeválogassa a számára fontosnak gondolt élelmiszereket, majd egy e-mail cím megadását követően leadja rendelését a bolt számára. Ezt követően a vásárló idővel visszajelzést is kapna arról, hogy az általa összeállított csomag elkészült, és elmehet érte, így megspórolva a sorban állást.

Az egyszerű dizájnnal rendelkező főoldalon a vásárló könnyen eligazodhat, a betűméretek nagyságának köszönhetően kormegkötés nélkül elérhetőek számukra az aktuális ajánlatok és az állandó, népszerű termékek.



31. ábra – Főoldal

A dinamikus mozgások egyértelművé teszik a felhasználót, hogy abban a pillanatban épp mely terméket kívánja megnézni, a nevükre kattintva pedig az oldal a részletes leírást jeleníti meg. Ekkor már a kosár tartalma üresen várja, hogy a vásárló elkezdje összeállítani a számára szükséges élelmiszereket.

Szenzációs Ajánlat

<p>Akció!</p>  <p>Szatmári Finomliszt 1kg 120 Ft <small>137 Ft</small></p>	<p>Akció!</p>  <p>Família Durum Orsó 400g 380 Ft <small>434 Ft</small></p>	<p>Akció!</p>  <p>Tolle Tejföl 20 % 325g 156 Ft <small>176 Ft</small></p>	<p>Akció!</p>  <p>Málnalekváros Fánk 75g 39 Ft <small>45 Ft</small></p>
--	--	--	---

Készítette: Szabócsi Nóra és Imri-Bárány Erika

32. ábra - Főoldal

A „Kosárba tesz” gomb lenyomásával a tétel megjelenik a jobb oldali sávban, névvel, egységárral, valamint egy összegzővel, amely újabb kattintást követve mindig növekedik, függően attól, hogy hányszor szerepel a tétel a kosárban. A termék eltávolításakor, ha a felhasználó úgy dönt, többet választott a kelletténél, alulról kiválasztva törölheti az utolsó tételt, a legfelső gombot megnyomva a teljes kosár tartalma kiürül. Ha úgy dönt, végzett a vásárlással, e-mail címe megadását követően a „Vásárlás”-t megnyomva a rendszer üzenetet küld, miszerint mindent rögzített a bolt. Abban az esetben, ha üres kosarat próbál elküldeni a vásárló, az API hibaüzenetet ír.

<p>Akció!</p> <p>Málnalekváros Fánk</p> <p><small>Leírás: Búzaliszt, Málna Lekvár (25%) (Málna (50%), Fruktózsirup, Cukor, Ivóvíz, Sűrítőanyag (Pektinek), Sav (Citromsav), Aroma, Színezék (Kapszantin)), Ivóvíz, Fánk Keverék (Búzaliszt, Cukor, Tejfehérjék, Pálmazsír Por [Pálmazsír, Glükózsirup, Tejfehérjék], Szárlított Tojásfelhérje, Tojáspor, Só, Emulgeálószer (Lecitinek (Glutén, Zsírsavak Mono- És Digliceridjei, Zsírsavak Mono- És Digliceridjeinek Mono- És Diacetil Borkősav Estere)), Búzaglutén, Aroma, Savasságot Szabályozó Anyag (Kalcium-Foszfátok), Színezék (Karotinok), Sűrítőanyag (Guargumi), Lisztkezelő Szer (Aszkorbinsav, L-Cisztein)), Sütőolaj (Teljes Mértékben Finomított Növényi Olajok (Szójababolaj, Pálmazsír)), Aroma, Habosodásgátló Anyag (Dimetil-Polisziloxán)), Dekor Porcukor (Dextróz, Búzakeményítő, Teljes Tejpor, Aroma), Margarin (Pálmazsír, Repceolaj, Napraforgóolaj, Ivóvíz, Só, Emulgeálószer (Zsírsavak Mono- És Digliceridjei), Sav (Citromsav), Színezék (Karotinok), Aroma), Élesztő, Burgonya Dextrin, Cékla Koncentrátum (Maltodextrin, Cékla Koncentrátum, Citromsav)</small></p> <p>39 Ft</p> <p>Kosárba Tesz</p>		<p>Kosár Tartalma Név: Málnalekváros Fánk Ár: 39 Ft</p> <p>Tétel Eltávolítása</p> <p>Összérték: 39 Adja Meg E-Mail Címét</p> <p>Vásárlás</p>
---	--	---

33. ábra - Kosár

6. FEJLESZTÉSI LEHETŐSÉGEK

6.1 Asztali alkalmazás (Imrő-Bárkányi Erika)

Az asztali alkalmazás esetében fejlesztési tartalék részét képezi, hogy az új termék felvételénél több mező is legyen, hogy minél több tulajdonságot tudjunk felvinni. Ezeket újabb textblock-ok és textbox-okkal kell pótolni, majd adatkötéssel hozzátenni az adatbázis adatait. Validáció szintén szükséges több esetben is. A Termékfotók esetében a terméknev mellett a szórólapon megjelenő leírása is helyet kap a későbbiekben. Az egyik legfontosabb fejlesztés viszont a rendelési tételek importálása az adatbázisból. Ez egy teljesen új felület megalkotását fogja jelenteni, ahol a rendelési tételekben szereplő termékek és annak mennyiségét figyeljük. Mivel ezek az adatok az adatbázisba íródnak, ugyanúgy jeleníteném meg az adatokat, mint a termékeket a kezdőfelületen.

6.2 WebApi (Imrő-Bárkányi Erika)

A weboldal illetve a mobil alkalmazás esetében a regisztráció és bejelentkezés fejlesztése után, a WebApi esetében a felhasználók adatainak lekérése lenne az egyik fontos fejlesztés. Ebben az esetben részletesebben lehetne lekérni az adatokat, hiszen a regisztráció esetében több adat is meg lesz adva.

6.3 Mobil alkalmazás (Szabácsi Noémi)

A mobilalkalmazás jelenleg kezdetleges funkciókkal van ellátva, amely megfelelően reprezentálja az adatok közötti kapcsolatot, helyi, illetve WebAPI lekérésekkel együtt. A későbbi fejlesztések kapcsán a regisztráció és bejelentkezési felület kialakítása lenne a cél. A dolgozóknak külön felülete lenne. A mobil projekt bővítésének részét képezné a továbbiakban, hogy azok a termékek, amelyek már akcióban vannak, kijelölt termékként, egyfajta figyelem felhívásként mutatná a vevőnek. Az Ica boltban törzsvásárlói kártya készítését is tervezik, ennek kapcsán a kártyához tartozó pontgyűjtő applikációt is lehetne fejleszteni, amely egy teljesen új projekt lenne.

6.4 Webes alkalmazás (Szabácsi Noémi)

Mindazonáltal, hogy a későbbiekben egy belépést szeretnék építeni, fontosnak tartom, hogy egy űrlap is kerüljön az oldalra, így lehetőséget teremtve akár a házhozszállításra is. Az űrlap egy Vue.js Form által lehetne megvalósítani, mely hasonlóan a Vásárlás POST parancsához, az API-ban rögzítené az adatokat, és tárolná el. Megvalósíthatónak tartom azt

is, hogy ezek későbbiekben menthetők legyenek, így nem kellene minden alkalommal újra kitöltenie a felhasználónak, vagy csak apróbb módosításokat kellene véghez vinnie.

A kuponok létrehozása is kecsegtető lehet a vásárlók számára, melyet akár minden ötödik vásárlásuk után egy QR-kód formájában egy egyedi azonosítót rendel a regisztrált felhasználóhoz. Ebben az esetben szükséges lenne az adminnak egy webes felületre is, ahol az ott dolgozók a kód leolvasását követően beazonosíthatják nem csak a leértékelés százalékát, hanem azt is, hogy a jelenlévő vásárló milyen adatokkal rendelkezik (Név, elérhetőség).

7. ÖSSZEGZÉS (Imrő-Bárkányi Erika, Szabácsi Noémi)

A szakdolgozatunk célja az Ica bolt fejlesztése volt, amely több alkalmazás megalkotását jelentette. A webes alkalmazással a vásárlóknak könnyebb, kényelmesebb vásárlási felület kialakítása volt a terv. Egy egyszerű, letisztult oldal, melyen hasznos információkhoz tudnak hozzáférni, és lehetőséget kapnak rendelést leadni. Nem tartalmaz még egy teljes komplex webshop tudást az oldal, hiszen rengeteget kell még fejleszteni, de azt sikerként könyvelhetjük el, hogy azokban a járvány sújtotta időkben egy kicsit megkönnyítettük a vásárlást a vevőknek. Az asztali alkalmazás szintén sok fejlesztendő területtel rendelkezik még, de úgy gondoljuk, hogy a boltban dolgozók hasznára vált. Az volt a cél, hogy a megrendelő az eladót is segítse a munkavégzésben azáltal, hogy a készletkezelő program segítségével gyorsabban és könnyebben tudjanak a készlettel foglalkozni, akár az otthonukból is. A mobil alkalmazás fejlesztése volt számunkra a legnehezebb, újra is kezdtük, mert sajnos nem mindig tudtuk megoldani azt, amit szerettünk volna, így többször újra terveztünk és egyeztettünk. Nem megoldhatatlan problémákba ütköztünk, hanem kevés időbe, mint amit a feladat megoldása megkívánt volna. Úgy gondoljuk, hogy az akadályokhoz képest, sikerült egy olyan szolgáltatást megalkotni, amellyel tudtunk segíteni a vásárlóknak is, hiszen az akciók figyelése a mai világban már elengedhetetlen. Egy ilyen jelzés azokról a termékekről, amelyek az embereknek fontosak, nem csak egy szép gesztus a bolt részéről, de egyértelműen üzleti értékkel is bír. Mi örültünk neki, hogy a felmerült problémák ellenére mindenkinek tudtunk adni némi megoldást, a mi fejlődő tudásunkkal. A csapatmunka számunkra nem okozott akadályt, hiszen napi szinten tudtunk kommunikálni. Sajnos mi csak ketten voltunk a feladatra a három fős csapatokkal szemben és bizony ennek éreztük a hátrányát is. Eltérő időbeosztással rendelkezünk, viszont

úgy gondoljuk, hogy a tudásunk, egymást segítve gyarapodott. A fejlesztések során felmerülő hibák megoldásait keresve, az iskolában megszerzett tudás és az internet adta lehetőségek nagy segítséget nyújtottak. A legtöbb szakirodalom angol nyelven található meg, így a szakmai angol használatát is gyakorolhattuk. A későbbiekben folytatjuk az alkalmazások tökéletesítését.

Köszönetnyilvánítás

Szeretnénk elsősorban megköszönni egymásnak a hasznos munkát, bízva abban, hogy a jövőben lesznek még közös projektjeink. Hálásak vagyunk tanárainknak, akik az órákon a legjobb tudásuk szerint próbáltak terelgetni egy új szakma elsajátításában, megismerésében. Továbbá hatalmas köszönet Zolinak és Pedrónak akik a többéves tapasztalatuk alapján tanácsokkal segítettek bennünket. Végül, de nem utolsó sorban köszönet jár Áronnak, aki hatalmas türelemmel viselte a kettőnk között zajló rendszeres meetingeket.

Ábrajegyzék:

1. ábra - Use-Case diagram	8
2. ábra - Xampp Control Panel	9
3. ábra – E-K Diagram	11
4. ábra – Bachmann ábra	14
5. ábra - MVVM struktúra(Forrás: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/gg405484(v=pandp.40)?redirectedfrom=MSDN).....	15
6. ábra - MVVM mappák és tartalmaik.....	16
7. ábra - Adatkötés (Data Binding)	16
8. ábra - Asztali alkalmazás mappái.....	17
9. ábra - Teszt Osztályok.....	18
10. ábra - Tesztelési folyamatok	19
11. ábra - WebAPI kommunikáció (Forrás: https://hevo.com/learn/rest-api-best-practices/	20
12. ábra - Adatbázis táblák leképezése.....	21
13. ábra - Teszt Osztály és teszt metódus.....	22
14. ábra - Teszt futtatás	22
15. ábra - PUT - Cikkmódosítás.....	23
16. ábra - GET lekérés.....	24
17. ábra - API kérés.....	26
18. ábra - Lokális adatok kérése.....	26
19. ábra - Lokális adatok	27
20. ábra - Mobil Teszt	28
21. ábra - Reszponzivitás kódja.....	29
22. ábra - Lenyíló menü	30
23. ábra - Kosárba tesz - töröl funkciók.....	30
24. ábra - Router.....	31
25. ábra - Post teszt	31
26. ábra - Belépési felület hibaüzenettel	32
27. ábra - Készletkezelő felület.....	33
28. ábra - Termékfotók.....	34
29. ábra - Termék módosítása	35
30. ábra - Mobil oldalak	36
31. ábra – Főoldal.....	37
32. ábra - Főoldal2	38
33. ábra - Kosár	38

Forrásjegyzék:

<https://hu.wikipedia.org/wiki/Modell-n%C3%A9zet-n%C3%A9zetmodell>

<https://csharp tutorial.hu/zenelejatszo-program-wpf-alapok/>

[https://docs.microsoft.com/en-us/previous-versions/msp-n-p/gg405484\(v=pandp.40\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/msp-n-p/gg405484(v=pandp.40)?redirectedfrom=MSDN)

<https://hevodata.com/learn/rest-api-best-practices/>

https://moodle.ms.sapientia.ro/pluginfile.php/14309/mod_resource/content/1/MVVM.pdf

<https://ionicframework.com/docs/>

<https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>

<https://azure.microsoft.com/hu-hu/products/visual-studio-code/>

<https://vuejs.org/guide/introduction.html>

Boros Bence óraadó tanár órai jegyzetei

MELLÉKLET

1.sz. melléklet

Asztali alkalmazás tesztjegyzőkönyv

TESZTELÉSI JEGYZŐKÖNYV	
A tesztet leírása és célja:	Belépés modul tesztelése. Célja, hogy a belépéshez szükséges adatok helyesek legyenek és a teszt zöld jelzést adjon. Helytelen tesztadattal a teszt piros jelzést adjon vissza.
A tesztelt folyamat / funkció leírása:	Felhasználónév és jelszó tesztelés. Program indítása után, egy bejelentkező felület jelenik meg, itt meg kell adni a belépéshez szükséges felhasználónevet és jelszót. Hibás beviteli adat esetében kapjunk hibaüzenetet.
A tesztelés előfeltételei:	Az asztali alkalmazás telepítése, Xampp program elindítása. A program hiba nélküli indulása.
A tesztelés dátuma és ideje:	2022.04.01
A tesztadatok típusa:	string felhasználónév string jelszó
A tesztet végző személy(ek):	Imrő-Bárkányi Erika
A tesztelt rendszer beállításai:	Operációs rendszer: Windows 10 Home Hardver: 64-bit, Intel Core i5-2500 CPU @ 3.30 GHz, 3701 Mhz, 4 mag Szoftver: Microsoft Visual Studio Community 2019 Version 16.10.0
A tesztet elvárt eredménye:	Helyes felhasználónévvel és jelszóval a teszt zöld jelzést mutat, a belépés engedélyezve lesz. Helytelen jelszó használatával illetve helytelen felhasználónév megadással a teszt piros jelzéssel hibaüzenetet ad vissza és a beléptetés nem történik meg.
A tesztelés eredménye:	<input type="checkbox"/> <u>megfelelt</u> <input type="checkbox"/> nem felelt meg <input type="checkbox"/> megfelelt megjegyzésekkel
Megjegyzések:	
A tesztelési jegyzőkönyvet készítette (név, aláírás):	Imrő-Bárkányi Erika

TESZTELÉSI JEGYZŐKÖNYV	
A tesztet leírása és célja:	A terméklista megjelenítése az adatbázisból történik. A tesztet célja, hogy a megjelenő terméklista tartalmazza-e az összes adatot az adatbázisból és ennek megfelelően jelez-e vissza.
A tesztelt folyamat / funkció leírása:	Az adatbázis terméklistában megszámolja a termékek számát majd megnézi a betöltött termékek számát, ha ez egyezik akkor sikeres eredményt mutat. Amennyiben eltérés van a listában található és a betöltött adatok között, akkor dobjon hibát, hogy nem került minden termék beolvasásra.
A tesztelés előfeltételei:	Az asztali alkalmazás telepítése, Xampp program elindítása. Adatbázisok használata. Sikeres belépés
A tesztelés dátuma és ideje:	2022.04.01
A tesztadatok típusa:	<input type="checkbox"/> elvárt <input type="checkbox"/> aktuális
A tesztet végző személy(ek):	Imrő-Bárkányi Erika
A tesztelt rendszer beállításai:	Operációs rendszer: Windows 10 Home Hardver: 64-bit, Intel Core i5-2500 CPU @ 3.30 GHz, 3701 Mhz, 4 mag Szoftver: Microsoft Visual Studio Community 2019 Version 16.10.0
A tesztet elvárt eredménye:	Az aktuális termékszám 30, ehhez elvárt adatként 10-et adunk. Lefuttatva a tesztet hibajelzést kapunk, hiszen nem töltődött be az aktuális termékmennyiség csak 10. Ha elvárt adatként azt szimuláljuk, hogy 30, akkor a teszt zöld jelzést ad, az elvártaknak megfelelően.
A tesztelés eredménye:	<input type="checkbox"/> <u>megfelelt</u> <input type="checkbox"/> nem felelt meg <input type="checkbox"/> megfelelt megjegyzésekkel
Megjegyzések:	
A tesztelési jegyzőkönyvet készítette (név, aláírás):	Imrő-Bárkányi Erika 47

TESZTELÉSI JEGYZŐKÖNYV	
A tesztet leírása és célja:	A terméklistából történő törlés esete. Célja, hogy a kitörölt darabszámnak megfelelő számmal csökkent-e a terméklista.
A tesztelt folyamat / funkció leírása:	Termék törlése esetén az adatbázisban található termékek befrissülnek. A frissítéssel újra számolja a termékeket és a helyes darabszámot mutatják a táblázat tetején.
A tesztelés előfeltételei:	Az asztali alkalmazás telepítése, Xampp program elindítása. Adatbázisok használata. Sikeres belépés
A tesztelés dátuma és ideje:	2022.04.01
A tesztadatok típusa:	int elvart int aktualis
A tesztet végző személy(ek):	Imrő-Bárkányi Erika
A tesztelt rendszer beállításai:	Operációs rendszer: Windows 10 Home Hardver: 64-bit, Intel Core i5-2500 CPU @ 3.30 GHz, 3701 Mhz, 4 mag Szoftver: Microsoft Visual Studio Community 2019 Version 16.10.0
A tesztet elvart eredménye:	Az aktuális termékadat 30, majd adunk egy elvart adatot, amely szintén legyen 30, ez az az adatszám amely törlés után marad. Futtatás után zöld jelzéssel tért vissza a teszt. Az elvart adatot 5-re módosítjuk, az aktuális marad 30. Ebben az esetben a teszt visszatér egy hibaüzenettel, amely jelzi, hogy nem törlődött minden termék a listánkból.
A tesztelés eredménye:	<input type="checkbox"/> <u>megfelelt</u> <input type="checkbox"/> nem felelt meg <input type="checkbox"/> megfelelt megjegyzésekkel
Megjegyzések:	
A tesztelési jegyzőkönyvet készítette (név, aláírás):	Imrő-Bárkányi Erika

