



VAS VÁRMEGYEI SZAKKÉPZÉSI CENTRUM
HORVÁTH BOLDIZSÁR
KÖZGAZDASÁGI ÉS INFORMATIKAI
TECHNIKUM

A 5 0613 12 03 számú Szoftverfejlesztő és –tesztelő vizsgaremek

A szoftveralkalmazás dokumentációja

Készítették:

CSERNI GERGŐ

KARDOS DÁNIEL

SZAKÁLY MÁTÉ BOTOND

SZOMBATHELY

2025

Tartalom

1. Bevezetés.....	2
1.1 A program célja.....	2
1.2 Célközönség	2
1.3 Rendeltetésszerű használat	2
2. A program általános leírása.....	3
2.1 A felhasználó igényei	3
2.2 Követelmények teljesítése.....	4
2.3 A tervezés főbb fázisai	5
2.3.1 Feladatok felosztása	5
2.3.2 A felület vázlatos megtervezése	5
2.4. Adatbázis	6
2.4.1 Adatbázis bemutatása.....	6
2.4.2 ER-modell	7
2.4.3 Relációs modell.....	9
3. Komponentseinek technikai leírása	10
3.1 Vásárlás oldal	10
3.2 Termék oldal	10
3.3 Saját termék.....	10
3.4 Eladás oldal	10
3.5 Chat oldal	10
3.6 Profil.....	11
3.7 Bejelentkezés oldal.....	11
3.8 Regisztráció oldal.....	11
3.9 Menü oldal	11
4. Működésének műszaki feltételei	12
5. Monitorképek	13
6. Regisztráció és Bejelentkezés	20
7. Használatának rövid bemutatása	21
8. Továbbifejlesztési lehetőségek.....	23
9. Tesztelés	24
9.1 Backend Tesztelés	24
9.2 Frontend Tesztelés.....	30
10. Ábrajegyzék	33
11. Források.....	34

1. Bevezetés

1.1 A program célja

Célunk, az olvasóközösség számára egy kulturált, felhasználóbarát felület létrehozása, ahol szűrt tartalmak között kereshetik a számukra megfelelő könyvet. Nem kell időt pazarolni azon hirdetések átgörgetésére, amik nem is könyveket tartalmaznak. Csevegőfelületünk segítségével a felhasználók más szoftverek használata nélkül le tudnak bonyolítani akár egy csereüzletet is személyes találkozóval.

1.2 Célközönség

Voltunk már mi is olyan helyzetben, hogy egy kötelező olvasmányt egy adott időpontra be kellett szerezni, de a környéken nem találtuk meg egy könyvesboltban sem. Ha az interneten rendeltük volna meg, akkor a hosszas kiszállítási idő miatt kifutottunk volna a határidőből. Egy alternatív megoldást kellett keresni: egy magánembertől beszerezni. Ez nem egyszerű folyamat, ugyanis az interneten rengeteg a több éves, elavult hirdetés. Miután a könyvet elolvastuk, arra gondoltunk, hogy bárcsak segíthetnénk valakinek ezzel a könyvvel, aki hasonló cipőben jár, mint mi. Kerestünk olyan alkalmazást, weboldalt, ami kifejezetten a könyvek hirdetésére specializálódott, de nem találtunk ilyen, vagy ehhez hasonló weboldalt és alkalmazást sem. Ezért hoztuk létre a Bookie-t, ahol szerzőkre, címre, akár még távolságra is lehet szűrni.

1.3 Rendeltetésszerű használat

A weboldalunk betöltéséhez több különböző program használatát is igénybe kell venni. A XAMPP Control Panel-ben az Apache és a MySQL szervereket el kell indítani. Az elindított phpMyAdmin felületen az adatbázisunk (bookie.sql) importálása szükséges. A sikeres betöltés után a következő lépés a bookieAPI mappában lévő Visual Studio Solution megnyitása és futtatása. Figyeljünk arra, hogy a Web.config fájlban a connectionString-ben és a XAMPP kezelő felületen lévő MySQL portszám megegyezzen (alaphelyzetben: 3306). A sikeres backend elindítás után a Parancssorban a bookie mappába lépve a következő parancs használata: „npm start”. Győződjön meg róla, hogy a weboldal a 3000-es porton fut!

Ezek után a felhasználó a bejelentkezési felületen találja magát, ahol regisztrálhat vagy használhatja a következő bejelentkezési adatokat:

Felhasználónév: testprofil

Jelszó: test

2. A program általános leírása

2.1 A felhasználó igényei

1. Felület

- Reszponzív megjelenés, bármilyen eszközről
- Könnyen átlátható

2. Kapcsolat velünk

- A felhasználók üzenetet tudnak küldeni nekünk hibákról és fel tudják tenni kérdéseiket, ha segítségre szorulnak
- Felhasználói visszajelzések gyűjtése

3. Gyors és egyszerű hirdetés feladás

- A könyv címét kiegészíti a program
- Összes magyar települést tartalmazza az adatbázisunk

4. Termékek közötti szűrés

- A könyvek között tud a felhasználó különböző szűrőket használni (pl. ár szerint növekvő/csökkenő)

5. Könyvcseré

- A felhasználók nem csak eladni szeretnék, hanem könyvet cserélni is.
- Kölcsönösen elfogadható cserék szervezésének támogatása (chat funkció)

6. Felhasználói profil és hirdetéskezelés

- Saját profiloldal, ahol a felhasználó szerkesztheti adatait és kezelheti aktív hirdetéseit.

2.2 Követelmények teljesítése

A projekt során elvárt követelményeket teljesítettük. Az alábbiakban részletezném ennek részeit:

- A szoftver forráskódja:
 - A teljes forráskód elérhető a leadott GitHub linken.
- Adatbázis export (dump)
 - A GitHub linken az adatbázis komponensen belül megtalálható és importálható.
 - Figyeltük arra, hogy az .sql kiterjesztésű fájl létrehozza az adatbázist is.
- Adatbázismodell-diagram
 - A jelenlegi PDF dokumentum, valamint a hozzá tartozó csatolt állományok tartalmazzák.
- Program dokumentációja
 - A dokumentáció tartalmazza a szoftver céljának leírását, működésének technikai részleteit, a főbb komponensek áttekintését, valamint a használat bemutatását.
- Tesztesetek dokumentációja
 - A jelenlegi PDF fájl tartalmazza a tesztek dokumentációját.
 - Ezek alátámasztják a szoftver megbízható működését.

2.3 A tervezés főbb fázisai

2.3.1 Feladatok felosztása

témaválasztás, ötletelés	Cserni Gergő, Kardos Dániel, Szakály Máté
célok megfogalmazása	Cserni Gergő, Kardos Dániel, Szakály Máté
adatbázis megtervezése	Cserni Gergő
adatbázis kezdeti feltöltése	Cserni Gergő, Kardos Dániel, Szakály Máté
kódolás	Cserni Gergő, Szakály Máté
dokumentálás	Cserni Gergő, Kardos Dániel, Szakály Máté
design	Cserni Gergő, Kardos Dániel, Szakály Máté
manuális tesztelés	Cserni Gergő, Kardos Dániel, Szakály Máté
tesztek írása	Cserni Gergő, Szakály Máté
információ gyűjtés	Cserni Gergő, Kardos Dániel, Szakály Máté
védési prezentáció tervezése	Cserni Gergő, Kardos Dániel, Szakály Máté
védési prezentáció létrehozása	Szakály Máté
védési prezentáció előadása	Cserni Gergő, Kardos Dániel, Szakály Máté

2.3.2 A felület vázlatos megtervezése

Már a kezdeti tervezéseknél is törekedtünk arra, hogy egy átlátható és könnyen kezelhető felületet biztosítsunk a felhasználóink számára. A program tervezési folyamatában készített monitorképek alapján hoztuk létre a weboldalunkat.

2.4 Adatbázis

2.4.1 Adatbázis bemutatása

Az adatbázisunk 8 különböző táblából áll. A **Profil** és az **Eladó termék** a két fő táblánk, a **Profil** a felhasználó adatait menti el, a **Település** és **Csevegés** táblák segítségével. Az **Eladó termék** tábla a felhasználók által feltöltött könyveket és annak adatait tárolja. A **Könyvek** továbbá egy másik tábla segítségével raktározza annak íróját a **Szerző**-be. A weboldalunkra feltöltött képeket a **Kép** tábla menti el egy felhő alapú szolgáltató segítségével, a Cloudinary-vel. (2. forrás)

Tábla	Művelet	Sorok	Tipus	Illesztés	Méret	Felülírás
<input type="checkbox"/> cseveges	★ Tartalom Szerkeszt Keresés Beszúrás Kiürítés Eldobás	6	InnoDB	utf8_hungarian_ci	48.0 KB	-
<input type="checkbox"/> eladotermek	★ Tartalom Szerkeszt Keresés Beszúrás Kiürítés Eldobás	14	InnoDB	utf8_hungarian_ci	48.0 KB	-
<input type="checkbox"/> kapcsolo	★ Tartalom Szerkeszt Keresés Beszúrás Kiürítés Eldobás	16	InnoDB	utf8_hungarian_ci	32.0 KB	-
<input type="checkbox"/> kep	★ Tartalom Szerkeszt Keresés Beszúrás Kiürítés Eldobás	14	InnoDB	utf8_hungarian_ci	32.0 KB	-
<input type="checkbox"/> konyv	★ Tartalom Szerkeszt Keresés Beszúrás Kiürítés Eldobás	16	InnoDB	utf8_hungarian_ci	16.0 KB	-
<input type="checkbox"/> profil	★ Tartalom Szerkeszt Keresés Beszúrás Kiürítés Eldobás	9	InnoDB	utf8_hungarian_ci	32.0 KB	-
<input type="checkbox"/> szerzo	★ Tartalom Szerkeszt Keresés Beszúrás Kiürítés Eldobás	15	InnoDB	utf8_hungarian_ci	16.0 KB	-
<input type="checkbox"/> telepules	★ Tartalom Szerkeszt Keresés Beszúrás Kiürítés Eldobás	3,145	InnoDB	utf8_hungarian_ci	192.0 KB	-
8 tábla	Összesen	3,235	InnoDB	utf8_hungarian_ci	416.0 KB	0 B

1. ábra: Adatbázisunk táblái

A **Települések** tábla tartalmazza az összes magyar település KSH kódját, nevét (3. forrás) és koordinátáját (4. forrás). A táblázatunkba a *SzélességiFok* és *HosszúságiFok* adatokat megszerzéséhez egy külön programot fejlesztettünk ki, ami fokperceket alakít át, egy képlet segítségével, ami: szélességfok + fokperc / 60. Például: 41°24'12.2"N => 41+0,24122/60

```
var sorok2 = File.ReadAllLines("telepulesek.txt");
var sorok = File.ReadAllLines("tables.helyseg_hu.csv").Skip(1);

HashSet<string> list = new HashSet<string>();
foreach (var line in sorok)
{
    list.Add(line.Split(';')[0]);
}
foreach (var line in sorok2)
{
    list.Add(line.Split(';')[0]);
}
var sorok2telepules = sorok2.Select(s => s.Split(';')[0]).ToList();
var soroktelepules = sorok.Select(s => s.Split(';')[0]).ToList();
```

2. ábra: Települések adatainak kinyerése, tárolása

A tábla tartalmát két különböző fájlból nyertük ki és alakítottuk át a megfelelő formátumra. A kinyert adatokat egyből adatbázisba felöltésre alkalmazható fájlba írtuk ki.

```
StreamWriter sw = new StreamWriter("telepulesektwo.txt");
sw.WriteLine("INSERT INTO `telepules` (`KSH`, `TelepulesNev`, `SzelessegiFok`, `HosszusagiFok`) VALUES");
foreach (var item in sorok)
{
    string ksh = null;
    if (item.Split(';')[0] == "Kehidakustány")
    {
        ksh = sorok2.Where(x => x.Split(';')[0] == "Kehidakustány").FirstOrDefault();
        soroktelepules.Add("Kehidakustány");
    }
    else
    {
        if (item.Split(';')[0] == "Rákócziújfalú")
        {
            ksh = sorok2.Where(x => x.Split(';')[0] == "Rákócziújfalú").FirstOrDefault();
            soroktelepules.Add("Rákócziújfalú");
        }
        else
        {
            ksh = sorok2.Where(x => x.Split(';')[0] == item.Split(';')[0]).FirstOrDefault();
        }
    }
    if (ksh == null)
    {
        ksh = $"{item.Split(';')[0]};0";
    }
    sw.WriteLine($"({ksh.ToString().Split(';')[1]}, '{ksh.Split(';')[0]}', " +
        $"{(int.Parse(item.Split(';')[2].Split(':')[0]) + double.Parse(item.Split(';')[2].Split(':')[1]) / 60).ToString()}", " +
        $"{(int.Parse(item.Split(';')[1].Split(':')[0]) + double.Parse(item.Split(';')[1].Split(':')[1]) / 60).ToString()})");
}
```

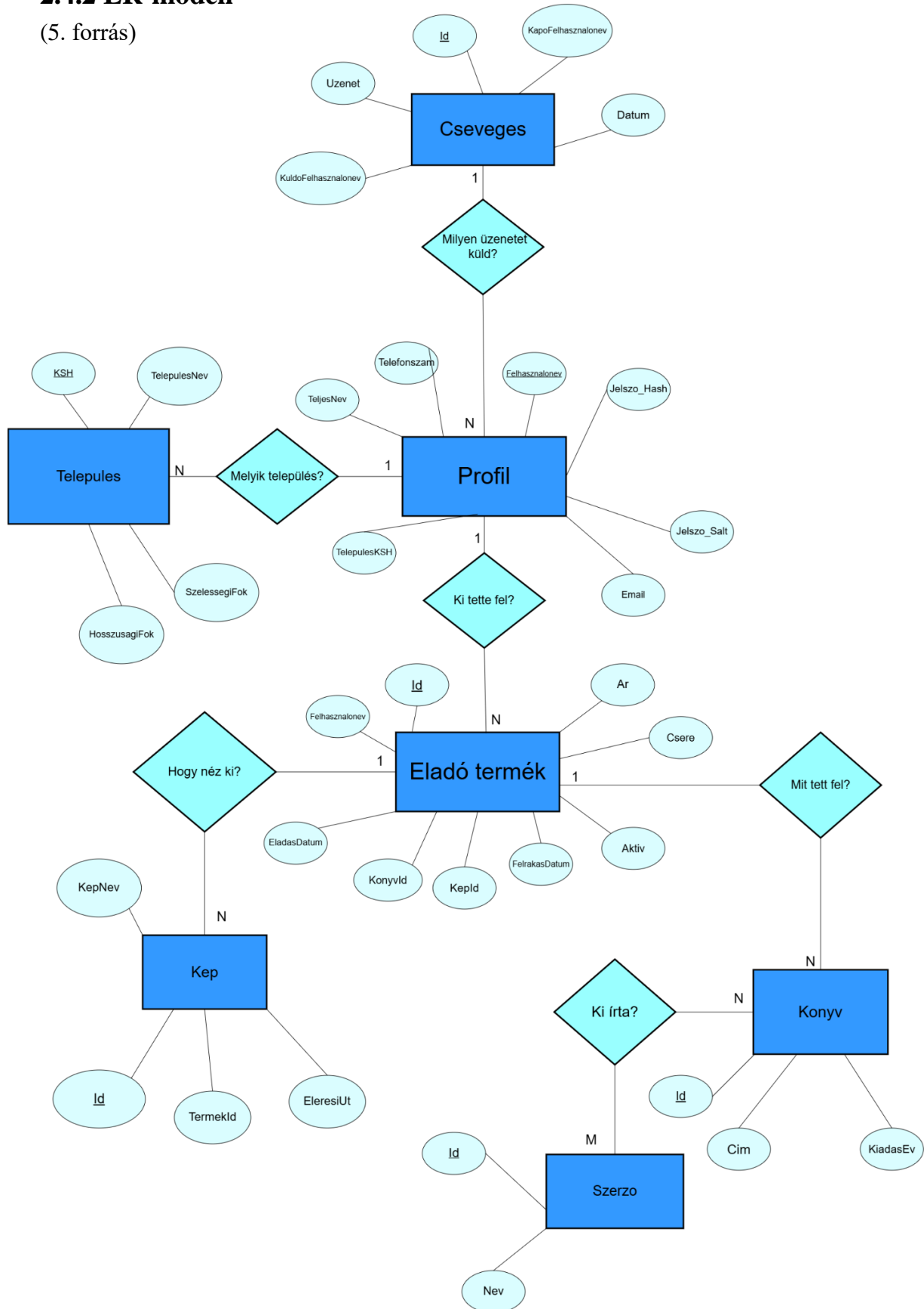
3. ábra: Kinyert adatok átalakítása és fájlba írása

```
foreach (var item in sorok2)
{
    if (!soroktelepules.Contains(item.Split(';')[0]))
    {
        sw.WriteLine($"({item.ToString().Split(';')[1]}, '{item.Split(';')[0]}', ),");
    }
}
sw.Close();
```

4. ábra: Kinyert adatok fájlba írása

2.4.2 ER-modell

(5. forrás)



5. ábra: ER-modell

2.4.3 Relációs modell

Szerzo (Id, Nev)

Konyv (Id, Cim, KiadasEv) Kapcsolatok: *Kapcsolo*

Kep (Id, Cim, EleresiUt, KepNev, TermekId) Kapcsolatok: *EladoTermek*

EladoTermek (Id, Ar, Csere, Aktiv, FelrakasDatum, EladasDatum, FelhasznaloNev, KonyvId) Kapcsolatok: *Profil*, *Konyv*

Profil (Felhasznalonev, Jelszo, Email, Telefonszam, TeljesNev, TelepulesKSH) Kapcsolatok: *Telepules*

Telepules (KSH, TelepulesNev, SzelessegiFok, HosszusagiFok)

Cseveges (Id, Uzenet, Datum, KuldoFelhasznalonev, KapoFelhasznalonev) Kapcsolatok: *Profil*

Kapcsolo(KonyvId, SzerzoId) Kapcsolatok: *Szerzo*

3. Komponenseinek technikai leírása

Minden oldalon kivéve a Bejelentkezés, Regisztráció és Menü oldalon, ha a felhasználó nincs belépve akkor átlesz irányítva a bejelentkezés oldalra.

3.1 Vásárlás oldal

Oldal megnyitása után, ha volt előzetes beállított rendezés akkor azt az állapotot lementjük. Oldal legenerálásához *fetch* függvény segítségével az adatbázisunkból lekérjük az összes aktív eladó terméket és a lementett rendezés alapján rendezzük. A rendezéshez a sort függvényt használjuk. Azon belül a távolság kiszámításához a *DistanceTo* (6. forrás) függvényt vesszük igénybe. A termékek kiválasztásakor a React-Router-Domból importált *useNavigate* csomagban lévő *navigate* segítségével lépünk át és tudjuk bővebben megnézni a termékeket.

3.2 Termék oldal

Az oldal megnyitásakor a React-Routerből importált *useParams* csomagban lévő *useParams* függvény segítségével az URL-ből kiolvassuk az termék id-t majd *fetch* segítségével lekérjük az adatbázisból a megfelelő id-val rendelkező terméket és közben legeneráljuk a terméket. A *@react-google-maps/api* csomagból a *GoogleMap*, *LoadScript*, *Marker* függvények és a Google Maps API (7. forrás) kulcsunk segítségével a térképet létrehozuk.

3.3 Saját termék

Ezen az oldalon a megjelenítés ugyanúgy működik, mint a termék oldalon csak itt a saját termékedet tudod kezelni, itt lehet törölni vagy megjelölni eladottként. Mindkettő funkció működését különböző *fetch*-ek biztosítják.

3.4 Eladás oldal

Az oldal betöltése után, ha vannak, az elmentett adatokat betölti a megfelelő helyre. A könyvkeresés működéséhez egy *fetch*-et használtunk, ami két másik fájl segítségével felkínálja a már feltöltött könyveket, ha a felhasználó itt nem találta meg a keresett könyvet, akkor egy másik oldal segítségével feltöltheti. Kép feltöltéskor *FileReader* osztály használatával a képet elmentjük ideiglenes *URL*-ként és ha minden adat jó akkor *fetch* segítségével feltölti az adatbázisba.

3.5 Chat oldal

A chat lista legenerálásához egy *fetch*-et használunk, amivel lekérjük a bejelentkezett felhasználó korábbi beszélgetéseit, generálás közben a megadott szempont szerint rendezzük. Valamelyik chat kiválasztásakor egy másik *fetch* segítségével legeneráljuk az összes üzenetet a kiválasztott beszélgetésben.

3.6 Profil

Az oldal megjelenítése után *update* és *validation* függvényeink használatával tudjuk módosítani a felhasználó adatait. A termékeim fölön egy *fetch* segítségével megjelennek a bejelentkezett felhasználó által feltöltött termékek. Egyik kiválasztásánál *navigate* segítségével átirányítjuk a felhasználót a Saját termék oldalra. A kapcsolat fölön egy *fetch* segítségével a felhasználó tud nekünk üzenetet küldeni, amit mi Discordon kapunk meg. A Kijelentkezés föltre kattintva a felhasználót kijelentkezteti és átirányítja a Bejelentkezés oldalra.

3.7 Bejelentkezés oldal

Abban az esetben, ha a felhasználó megfelelő adatokat ad meg, be tud jelentkezni, ha nincs fiókja akkor a „Regisztrálok” gombra kattintva tud a felhasználó fiókot létrehozni.

3.8 Regisztráció oldal

Abban az esetben, ha a felhasználó minden adatot helyesen kitölt, akkor egy *fetch* segítségével létrehozza a fiókját és át lesz irányítva a bejelentkezés oldalra.

3.9 Menü oldal

Ezen az oldalon, ha a felhasználó nincs bejelentkezve más oldalváltó opciókat ad a felhasználóknak.

4. Működésének műszaki feltételei

1. Szerveroldali követelmények

- Backend technológia: ASP.NET
- Adatbázis: MySQL
- Tárhely igény: Legalább 1 GB szabad tárhely
- Operációs rendszer: Windows

2. Kliensoldali feltételek

- Internetkapcsolat
- Böngészőtámogatás (Google Chrome, Opera, Microsoft Edge)
- JavaScript engedélyezés

3. Fejlesztői környezet

- Forráskódkezelés: GitHub
- Fejlesztői eszközök: Visual Studio Code, Visual Studio 2019, XAMPP

5. Monitorképek

The image shows a registration form with a dark blue background. It contains seven input fields, each with a label on the left and a text box on the right. The labels are: 'E-mail:', 'Felhasználónév:', 'Telefonszám:', 'Teljes Név:', 'Település:', 'Jelszó:', and 'Jelszó újra:'. Below these fields is a large button labeled 'Regisztrálás'.

E-mail:	<input type="text"/>
Felhasználónév:	<input type="text"/>
Telefonszám:	<input type="text"/>
Teljes Név:	<input type="text"/>
Település:	<input type="text"/>
Jelszó:	<input type="password"/>
Jelszó újra:	<input type="password"/>
<input type="button" value="Regisztrálás"/>	

6. ábra: Regisztráció oldal

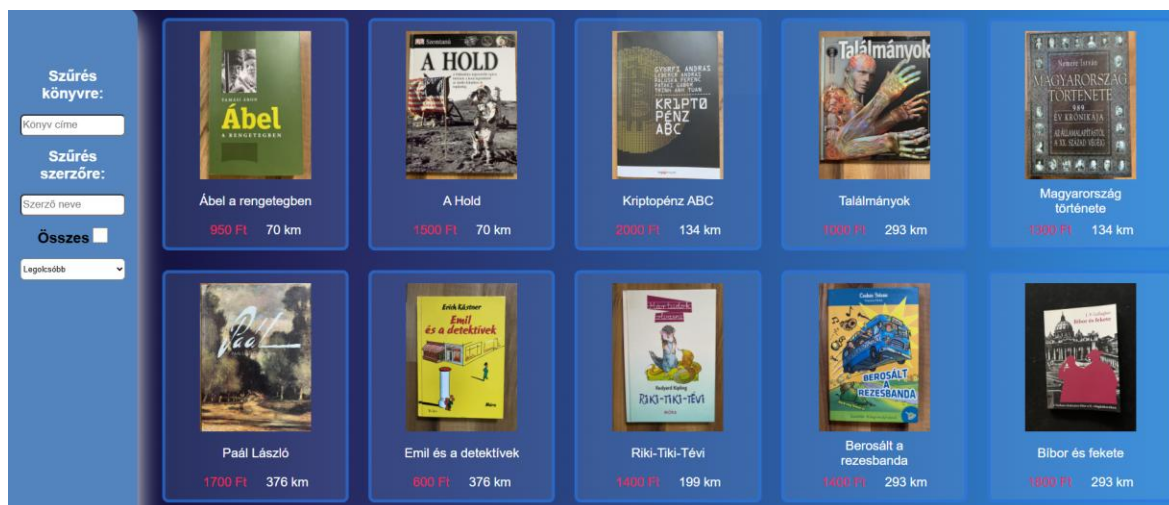
E-mail:	abc
Nem megfelelő az e-mail cím formátuma!	
Felhasználónév:	
Telefonszám:	654321
Rossz telefonszám	
Teljes Név:	
Település:	
Jelszó:
A jelszó túl rövid!	
A jelszóban nem szerepel nagybetű!	
A jelszóban nem szerepel szám!	
A jelszóban nem szerepel különleges karakter!	
Jelszó újra:	
Regisztrálás	

7. ábra: Regisztráció oldal – rossz/hibás adatokkal

E-mail / Felhasználónév:	testprofil
Jelszó:
Bejelentkezés	
Nincs fiókod? Regisztrálok	

8. ábra: Bejelentkezés oldal

9. ábra: Menüsor – oldalak közötti váltás



10. ábra: Vásárlás – összes termék megjelenítése, szűrőkkel



11. ábra: Termék oldal – egy kiválasztott termékről bővebb információk

Keresés

Ár (Ft):

Csere:

Kép (max 3 MB):

Fájl kiválasztása Nincs fájl kiválasztva

Nincs itt a könyved?

Eladás

12. ábra: Termék feltöltése – termék adatainak kitöltése megfelelően

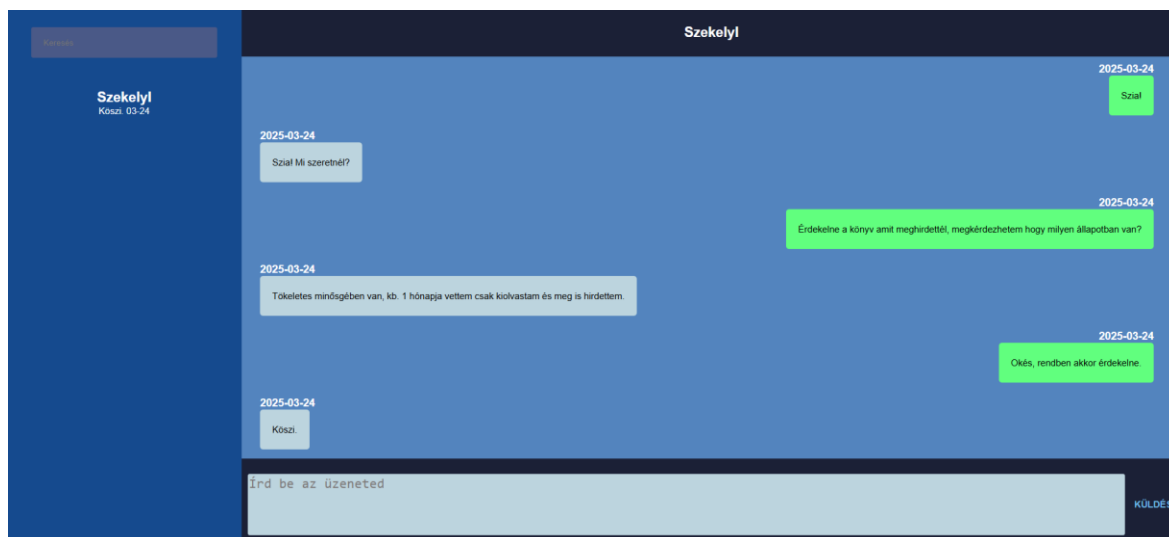
Könyv címe:

Kiadási éve:

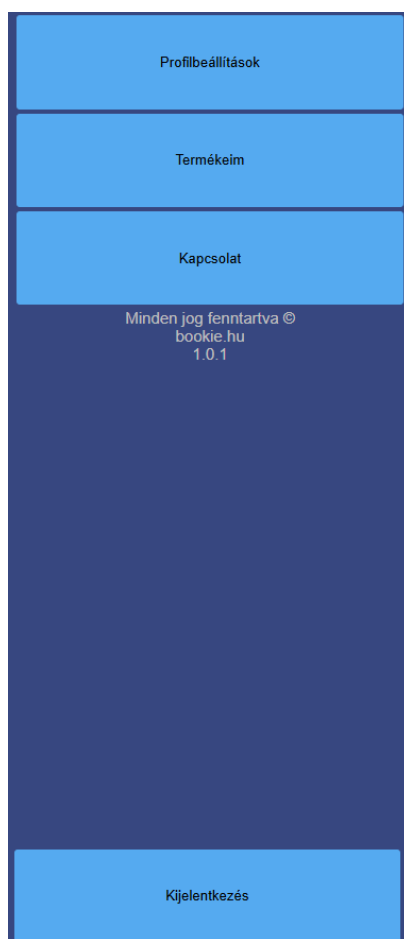
Szerző/Kiadó neve:

Felvétel

13. ábra: Könyv felvétel az adatbázisunkba



14. ábra: Chat felület (8. forrás) – aktuális beszélgetés, illetve bal oldalt a chat előzmények



15. ábra: Profil oldal menü


bookie.hu


Profilbeállítások
Termékeim
Kapcsolat
Kijelentkezés

Új Jelszó:
Módosítás

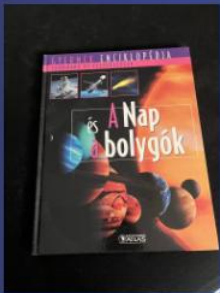
Teljes Név:
Test Nev
Módosítás

Telefonszám:
123456789
Módosítás

Település:
Kőszeg
Módosítás


Mentés

16. ábra: Profil adatok módosítása – Mentéskor jelszó megadása kötelező



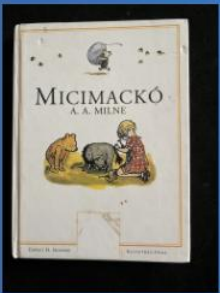
Gyermekenciklopédia
- A Nap és a bolygók

1500 Ft



Vágta a horizontig

1500 Ft



Micimackó

1300 Ft

17. ábra: Saját termékek – piros háttérszínű eladottnak jelölt



Edition Atlas:

Gyermekenciklopédia - A Nap és a bolygók

1500 Ft

Megjelölés
Eladottként

Hirdetés
Törlése

18. ábra: Saját termék konfigurálása

Ha fennakadtál, hibát észleltél vagy csak problémába ütköztél, akkor vedd fel velünk a kapcsolatot, mi pedig megpróbálunk minél hamarabb segíteni Neked!

Küldés

19. ábra: Kapcsolat felvétel – üzenet küldési lehetőség nekünk

6. Regisztráció és Bejelentkezés

A weboldalunk használata regisztrációhoz kötött, amit az oldal megnyitása után tudunk elkezdeni vagy egy meglévő fiókkal bejelentkezni. Egy új profil létrehozásához szükség van egy email címre, amivel természetesen még nincs Bookie fiók regisztrálva, felhasználónévre, aminek ugyancsak egyedinek kell lennie, telefonszámra, teljes névre, településre, ami segítségével később koordináták alapján tudjuk behatárolni más termékekhez képesti távolságunkat. Szükség van egy jelszóra, amit kétszer kell megadni a regisztráció során ezzel ellenőrizve, hogy a felhasználó jól adta meg és nem gépelt el semmit. A jelszóba szükség van legalább egy nagy betűre és egy speciális karakterre (Pl: @&#!).

Ha már rendelkezik fiókkal, akkor a bejelentkezési opciónál az email címet vagy a felhasználónevet és ahhoz a profilhoz kötött jelszót helyesen kell megadni az oldal használatához.

Tesztprofil:

Felhasználónév: testprofil

Jelszó: test

7. Használatának rövid bemutatása

Sikeres bejelentkezés után használhatjuk a fő menüsort (telefonon: hamburger menü), itt is tudunk különböző oldalak között lépkedni. Négy ilyen komponens található Vásárlás, ami egyben a főoldal is, Eladás, Chat és a Profil.

7.1 Vásárlás (Főoldal)

Itt tudunk böngészni különböző eladásra, illetve cserére bocsátott könyvek között. Ezen oldal bal oldalán más-más szűrők alapján tudjuk szortírozni a hirdetéseket. Ilyen például, hogy be tudjuk írni a címet, szerzőt, ezzel gyorsabban meg tudjuk találni az általunk keresett könyveket. Ki tudjuk szűrni, hogy valaki cserére, vagy csak értékesítésre szánja a műveket. Található az oldalon egy általános szűrő, amivel tudunk árra és időre is állítani a sorrendet. Minden termék egy keretbe épül fel, ahol megtalálható egy kép a könyvről, annak címe, ára, illetve a profilba megadott saját településünktől számított távolság. A saját „apróhirdetésünket” piros háttérszínnel látjuk, amire rákattintva tudjuk annak tulajdonságait itt is módosítani. Más feladott hirdetésére kattintva megnyílik az adott árucikk oldala. Itt bal oldalt látható egy nagyobb kép a termékről, jobbra megtalálhatók a könyv adatai, ilyen a szerző, cím, ár, cserélhetőség, eladó neve, települése, Google Map és egy gomb. Utóbbival át tudunk lépni a chat oldalra, és üzenetet tudunk küldeni az eladónak. A chat fülön tudunk kommunikálni a részletekről, az áru állapotáról, esetleg cseréről.

7.2 Eladás oldal

A névből következtetően a saját hirdetéseket tudunk létrehozni. Először is, meg kell adnunk egy könyv általános adatait, aminek a címének a gépelését elkezdve a program megtalálja azt az adatbázisunkban, és kiegészíti. Meg kell adni egy árat, egy dobozt kipipálni, hogy érdekel-e csere, illetve egy képet feltölteni. Ha nem egészíti ki a könyv címét, a „Nincs itt a könyved?” gombot kell használni, és itt fel lehet tölteni az adatbázisba.

7.3 Chat oldal

Üzenetet tudunk küldeni a többi felhasználónak. Bal oldalt lehet kiválasztani a régebbi üzenetváltásokat, felül pedig keresni felhasználókra. Új chatet itt nem tudunk kezdeményezni, azt csak egy termék oldalán a chat gomb megnyomásával lehet.

7.4 Profil oldal

A bal oldali menüsorban négy gomb található: a Profilbeállítások, Termékeim, Kapcsolat, ezek alatt a weboldal általános adatai, és teljesen alul pedig a Kijelentkezés gomb. A főoldalon közepén tudjuk a profil általános beállításait módosítani, ilyen a jelszó, az oldalhoz

kötött polgári teljes név, telefonszám, illetve település. A Termékeim gomb az adott felhasználó saját hirdetésre kínált árucikkeit mutatja, amit az előzőekben leírt módon lehet módosítani. A Kapcsolat komponensen pedig a weboldal szerkesztőinek tudunk üzenetet küldeni az oldalon észrevett hibákról, bugokról, illetve javításra szoruló dolgokról. A Kijelentkezés gombbal pedig ki tudunk jelentkezni az oldalról.

8. Továbbfejlesztési lehetőségek

1. Beépített bankkártyás fizetés

- A felhasználók a weboldalon keresztül tudnak fizetni bankkártyával, megelőzve a sok egyeztetést chaten
- Biztonságos

2. Support csevegés

- Ügyfélszolgálati segítség csevegésben
- Későbbiekben mesterséges intelligencia az általános kérdésekre

3. Értékelési rendszer

- A felhasználók értékelhetik egymást a tranzakciók után (csillag, komment)
- Növeli a biztonságot

4. Közösségi média

- Az oldal terjeszkedése
- Szeretnénk elérni, hogy egy gomb használatával a felhasználók könnyen megtudják osztani a hirdetéseiket a social platformjaikon

5. Ajánlórendszer

- Algoritmikus könyv ajánlások korábbi keresések szerint

6. Mobilalkalmazás

- Értesítések
- Több látogató az oldalon

7. Többnyelvűség

- Más országokba terjeszkedés

9. Tesztelés

9.1 Backend Tesztelés

Backend tesztek sikeres futtatásához futnia kell a backendnek.

```
[TestMethod]
| 0 references
public void PostSucceeds()
{
    var response = _client.PostAsJsonAsync("/api/Profil", new ProfilCreateModel
    {
        Felhasznalonev = "testfelhasznalonev",
        Email = "testemail",
        Jelszo = "testjelszo",
        Telefonszam = "testtelefonszam",
        TelepulesNev = "Szombathely",
        TeljesNev = "testteljesnev"
    })).Result;
    Assert.AreEqual(HttpStatusCode.OK, response.StatusCode);
}

[TestMethod]
| 0 references
public void PostReturnsNotFound()
{
    var response = _client.PostAsJsonAsync("/api/Profil", new ProfilCreateModel
    {
        Felhasznalonev = "testfelhasznalonev",
        Email = "testemail",
        Jelszo = "testjelszo",
        Telefonszam = "testtelefonszam",
        TelepulesNev = "nincsilyentelepules",
        TeljesNev = "testteljesnev"
    })).Result;
    Assert.AreEqual(HttpStatusCode.NotFound, response.StatusCode);
}

[TestMethod]
| 0 references
public void PostReturnsEmailExists()
{
    var response = _client.PostAsJsonAsync("/api/Profil", new ProfilCreateModel
    {
        Felhasznalonev = "testfelhasznalonev",
        Email = "test.email@gmail.com", //pelda felhasznalo emailje
        Jelszo = "testjelszo",
        Telefonszam = "testtelefonszam",
        TelepulesNev = "Szombathely",
        TeljesNev = "testteljesnev"
    })).Result;
    Assert.AreEqual(HttpStatusCode.Conflict, response.StatusCode);
}
```

20. ábra: Post metódus tesztjei

A „PostSucceeds” tesztmetódusban Ok státusz kódot várunk válasznak ezért megfelelő adatokat adunk meg így a „response”-nak a státusz kódja Ok lesz, sikeresen lefut a teszt.

A „PostReturnsNotFound” tesztmetódusban NotFound (település nem található) státusz kódot várunk válasznak ezért településnek rossz adatot adunk meg így a „response”-nak a státusz kódja NotFound lesz, sikeresen lefut a teszt.

A „PostReturnsEmailExists” tesztmetódusban Conflict (már létezik ilyen email) státusz kódot várunk válasznak ezért emailnek rossz adatot adunk meg így a „response”-nak a státusz kódja Conflict lesz, sikeresen lefut a teszt.

```
[TestMethod]
public void AuthenticateWorks()
{
    var response = _client.PostAsJsonAsync("/api/Profil", new ProfilCreateModel
    {
        Felhasznalonev = "testfelhasznalonev",
        Email = "testemail",
        Jelszo = "testjelszo",
        Telefonszam = "testtelefonszam",
        TelepulesNev = "Szombathely",
        TeljesNev = "testteljesnev"
    }).Result;

    //Ok
    var response3 = _client.PostAsJsonAsync("/api/Profil/authenticate", new AuthenticationModel
    {
        Email = "testemail",
        Jelszo = "testjelszo"
    }).Result;
    Assert.AreEqual(HttpStatusCode.OK, response3.StatusCode);

    //Nincs ilyen email
    var response4 = _client.PostAsJsonAsync("/api/Profil/authenticate", new AuthenticationModel
    {
        Email = "nincsilyenemail",
        Jelszo = "testjelszo"
    }).Result;
    Assert.AreEqual(HttpStatusCode.NotFound, response4.StatusCode);

    //Rossz jelszó
    var response5 = _client.PostAsJsonAsync("/api/Profil/authenticate", new AuthenticationModel
    {
        Email = "testemail",
        Jelszo = "nemezajelszo"
    }).Result;
    Assert.AreEqual(HttpStatusCode.Unauthorized, response5.StatusCode);
}
```

21. ábra: Authenticate metódus tesztjei

Az „AuthenticateWorks” tesztmetódusban az Authenticate metódusunk minden kimenetét teszteljük. Kezdeként feltöltünk egy példa tesztadatot, ezután az első tesztelési résznél Ok (sikeres bejelentkezés) státusz kódot várunk válasznak ezért megfelelő adatokat adunk meg így a „response3”-nak a státusz kódja Ok lesz. A második tesztelési résznél NotFound (nem található ilyen email/felhasználónév) státusz kódot várunk válasznak ezért rossz email adatot adunk meg így a „response4”-nek a státusz kódja NotFound lesz. A harmadik tesztelési résznél Unauthorized (rossz jelszó) státusz kódot várunk válasznak ezért rossz jelszó adatot adunk meg így a „response5”-nek a státusz kódja Unauthorized lesz.

```

[TestMethod]
0 references
public void PutWorks()
{
    var response = _client.PostAsJsonAsync("/api/Profil", new ProfilCreateModel
    {
        Felhasznalonev = "testfelhasznalonev",
        Email = "testemail",
        Jelszo = "testjelszo",
        Telefonszam = "testtelefonszam",
        TelepulesNev = "Szombathely",
        TeljesNev = "testteljesnev"
    }).Result;

    var adatok = response.Content.ReadAsStringAsync().Result;
    var adat = JsonConvert.DeserializeObject<ProfilResponseModel>(adatok);

    //Ok
    var response6 = _client.PutAsJsonAsync($"api/Profil/{adat.Felhasznalonev}", new ProfilUpdateModel
    {
        RegiJelszo = "testjelszo",
        UjJelszo = "tesztesebbjelszo",
        Email = "testemail",
        Telefonszam = "testtelefonszam",
        TelepulesNev = "Sopron",
        TeljesNev = "testteljesnev"
    }).Result;
    Assert.AreEqual(HttpStatusCode.OK, response6.StatusCode);

    //Rossz jelszo
    var response7 = _client.PutAsJsonAsync($"api/Profil/{adat.Felhasznalonev}", new ProfilUpdateModel
    {
        RegiJelszo = "nemezajelszavam",
        UjJelszo = "tesztesebbjelszo",
        Email = "testemail",
        Telefonszam = "testtelefonszam",
        TelepulesNev = "Sopron",
        TeljesNev = "testteljesnev"
    }).Result;
    Assert.AreEqual(HttpStatusCode.Unauthorized, response7.StatusCode);

    //Nem talal ilyen felhasznalot
    var response8 = _client.PutAsJsonAsync("/api/Profil/nincsilyenfelhasznalonev", new ProfilUpdateModel
    {
        RegiJelszo = "testjelszo",
        UjJelszo = "tesztesebbjelszo",
        Email = "testemail",
        Telefonszam = "testtelefonszam",
        TelepulesNev = "Sopron",
        TeljesNev = "testteljesnev"
    }).Result;
    Assert.AreEqual(HttpStatusCode.NotFound, response8.StatusCode);
}

```

22. ábra: Put metódus tesztjei

```

//Már van ilyen email
var response9 = _client.PutAsJsonAsync($"/api/Profil/{adat.Felhasznalonev}", new ProfilUpdateModel
{
    RegiJelszo = "tesztsebbjelszo",
    UjJelszo = "legtesztsebbjelszo",
    Email = "test.email@gmail.com", //pelda felhasznalo emailje
    Telefonszam = "testtelefonszam",
    TelepulesNev = "Sopron",
    TeljesNev = "testteljesnev"
}).Result;
Assert.AreEqual(HttpStatusCode.Conflict, response9.StatusCode);

//telepules nem talalhato
var response10 = _client.PutAsJsonAsync($"/api/Profil/{adat.Felhasznalonev}", new ProfilUpdateModel
{
    RegiJelszo = "tesztsebbjelszo",
    UjJelszo = "tesztjelszo",
    Email = "legtestsebbemail",
    Telefonszam = "testtelefonszam",
    TelepulesNev = "nincsilyentelepules",
    TeljesNev = "testteljesnev"
}).Result;
Assert.AreEqual(HttpStatusCode.NotFound, response10.StatusCode);
}

```

23. ábra: Put metódus tesztjei

A „PutWorks” tesztmetódusban a Put metódusunk minden kimenetét teszteljük. Kezdeként feltöltünk egy példa tesztadatot, ezután az első tesztelési résznél Ok (sikeres adatmódosítás) státusz kódot várunk válasznak ezért megfelelő adatokat adunk meg így a „response6”-nak a státusz kódja Ok lesz. A második tesztelési résznél Unauthorized (rossz jelszavat adott meg) státusz kódot várunk válasznak ezért rossz régi jelszó adatot adunk meg így a „response7”-nek a státusz kódja Unauthorized lesz. A harmadik tesztelési résznél NotFound (nem található ilyen felhasználó) státusz kódot várunk válasznak ezért rossz felhasználó adatot adunk meg így a „response8”-nak a státusz kódja NotFound lesz. A negyedik tesztelési résznél Conflict (már létezik ilyen email) státusz kódot várunk válasznak ezért rossz email adatot adunk meg így a „response9”-nek a státusz kódja Conflict lesz. Az ötödik tesztelési résznél NotFound (nem található ilyen település) státusz kódot várunk válasznak ezért rossz település adatot adunk meg így a „response10”-nek a státusz kódja NotFound lesz.

```

[TestMethod]
| 0 references
public void GetGets()
{
    var response = _client.GetAsync("/api/Profil").Result;
    var adatok = response.Content.ReadAsStringAsync().Result;
    List<ProfilResponseModel> lissta = new List<ProfilResponseModel>();
    lissta = JsonConvert.DeserializeObject<List<ProfilResponseModel>>(adatok);
    Assert.IsNotNull(lissta);
}

[TestMethod]
| 0 references
public void GetWithUsernameGets()
{
    var response = _client.PostAsJsonAsync("/api/Profil", new ProfilCreateModel
    {
        Felhasznalonev = "testfelhasznalonev",
        Email = "testemail",
        Jelszo = "testjelszo",
        Telefonszam = "testtelefonszam",
        TelepulesNev = "Szombathely",
        TeljesNev = "testteljesnev"
    }).Result;
    var result = response.Content.ReadAsStringAsync().Result;
    var adat = JsonConvert.DeserializeObject<ProfilResponseModel>(result);

    var response = _client.GetAsync($"api/Profil/{adat.Felhasznalonev}").Result;
    var adatok = response.Content.ReadAsStringAsync().Result;
    //List<ProfilResponseModel> lissta = new List<ProfilResponseModel>();
    var lissta = JsonConvert.DeserializeObject<ProfilResponseModel>(adatok);
    Assert.IsNotNull(lissta);
}

[TestMethod]
| 0 references
public void GetWithUsernameReturnsNotFound()
{
    var response = _client.GetAsync("/api/Profil/nincsilyenfelhasznalonev").Result;
    Assert.AreEqual(HttpStatusCode.NotFound, response.StatusCode);
}

```

24. ábra: Get metódusok tesztjei

A „GetGets” tesztmetódusban nem null értéket várunk válasznak és mivel az adatbázisban vannak alapból adatok így a „lissta” nem lesz null, sikeresen lefut a teszt.

A „GetWithUsernameGets” tesztmetódusban nem null értéket várunk válasznak ezért a tesztmetódus elején feltöltünk egy példa tesztadatot, aminek a felhasználónevét felhasználva a „lissta” nem null lesz, sikeresen lefut a teszt.

A „GetWithUsernameReturnsNotFound” tesztmetódusban NotFound (nem létezik ilyen felhasználó) státusz kódot várunk válasznak ezért felhasználónévnek rossz adatot adunk meg így a „response”-nak a státusz kódja NotFound lesz, sikeresen lefut a teszt.

```

[TestMethod]
0 references
public void DeleteWorks()
{
    var response = _client.PostAsJsonAsync("/api/Profil", new ProfilCreateModel
    {
        Felhasznalonev = "testfelhasznalonev",
        Email = "testemail",
        Jelszo = "testjelszo",
        Telefonszam = "testtelefonszam",
        TelepulesNev = "Szombathely",
        TeljesNev = "testteljesnev"
    }).Result;
    var adatok = response.Content.ReadAsStringAsync().Result;
    var adat = JsonConvert.DeserializeObject<ProfilResponseModel>(adatok);

    var response2 = _client.DeleteAsync($"/api/Profil/{adat.Felhasznalonev}").Result;
    Assert.AreEqual(HttpStatusCode.OK, response2.StatusCode);

    var response8 = _client.DeleteAsync($"/api/Profil/nincsilyenfelhasznalonev").Result;
    Assert.AreEqual(HttpStatusCode.NotFound, response8.StatusCode);
}

```

25. ábra: Delete metódus tesztjei

A „DeleteWorks” tesztmetódusban a Delete metódusunk minden kimenetét teszteljük. Kezdeként feltöltünk egy példa tesztadatot, ezután az első tesztelési résznél Ok (sikeres törlés) státusz kódot várunk válasznak ezért megfelelő adatokat adunk meg így a „response2”-nek a státusz kódja Ok lesz. A második tesztelési résznél NotFound (nincs ilyen felhasználó) státusz kódot várunk válasznak ezért rossz felhasználónév adatot adunk meg így a „response8”-nak a státusz kódja NotFound lesz.

9.2 Frontend Tesztelés

Frontend teszteléséhez a Katalon Studiót használtuk. (9. forrás)

Item	Object	Input	Output	Description
➔ 1 - Open Browser		""		
➔ 2 - Navigate To Url		"http://localhost:3000/Login"		
➔ 3 - Set Text	input_E-mail Felhasznlnv	"testprofil"		
➔ 4 - Set Encrypted Text	input_Jelsz_passwd1	"P9ET2sDE0SE="		
➔ 5 - Click	input_Jelsz_login			
➔ 6 - Accept Alert				
➔ 7 - Delay		5		
➔ 8 - Click	input_text			
➔ 9 - Set Text	input_text_1	"p"		
➔ 10 - Set Text	input_text_1_2	"á"		
➔ 11 - Set Text	input_text_1_2_3	"l"		
➔ 12 - Click	img			
➔ 13 - Click	button_zenet			
➔ 14 - Delay		2		
➔ 15 - Set Text	textarea_Szia, rdekelne ez	"Szia, érdekelne ez a könyv"		
➔ 16 - Click	a_Klds			
➔ 17 - Set Text	textarea_Szia, rdekelne ez	"Mikor, és hol tudnánk találkozni?"		
➔ 18 - Click	a_Klds			

26. ábra: Könyvvétel tesztje

Az első tesztben azt néztük meg, hogy bejelentkezés után a „pál”-ra rákeresve, a megfelelő terméket kiválasztva, „Üzenet” gombra kattintás után az üzenetek el lesznek-e küldve.

Test Cases/elsos (150.198s)
➔ 1 - openBrowser("") (1.879s)
➔ 2 - navigateToUrl("http://localhost:3000/Login") (0.546s)
➔ 3 - setText(findTestObject("Object Repository/Page_Bookie/input_E-mail_Felhasznlnv_email"), "testprofil") (0.769s)
➔ 4 - setEncryptedText(findTestObject("Object Repository/Page_Bookie/input_Jelsz_passwd1"), "P9ET2sDE0SE=") (0.343s)
➔ 5 - click(findTestObject("Object Repository/Page_Bookie/input_Jelsz_login")) (0.348s)
➔ 6 - acceptAlert() (0.042s)
➔ 7 - delay(2) (2.057s)
➔ 8 - click(findTestObject("Object Repository/Page_Bookie/input_text")) (30.098s)
➔ 9 - setText(findTestObject("Object Repository/Page_Bookie/input_text_1"), "p") (180.291s)
➔ 10 - setText(findTestObject("Object Repository/Page_Bookie/input_text_1_2"), "á") (180.334s)
➔ 11 - setText(findTestObject("Object Repository/Page_Bookie/input_text_1_2_3"), "l") (180.305s)
➔ 12 - click(findTestObject("Object Repository/Page_Bookie/img")) (30.088s)
➔ 13 - click(findTestObject("Object Repository/Page_Bookie/button_zenet")) (33.156s)
➔ 14 - setText(findTestObject("Object Repository/Page_Bookie/textarea_Szia_rdekelne_ez_a_knyv_uzenet"), "Szia, érdekelne ez a könyv") (6.900s)
➔ 15 - click(findTestObject("Object Repository/Page_Bookie/a_Klds")) (0.478s)
➔ 16 - setText(findTestObject("Object Repository/Page_Bookie/textarea_Szia_rdekelne_ez_a_knyv_uzenet"), "Mikor, és hol tudnánk találkozni?") (36.775s)
➔ 17 - click(findTestObject("Object Repository/Page_Bookie/a_Klds")) (0.376s)

27. ábra: Sikeres teszt

Item	Object	Input	Output	Description
→ 1 - Open Browser		""		
→ 2 - Navigate To Url		"http://localhost:3000/Lc		
→ 3 - Set Text	input_E-mail_Felhasznlnv	"testprofil"		
→ 4 - Set Encrypted T	input_Jelsz_passwd1	"P9ET2sDE0SE="		
→ 5 - Click	input_Jelsz_login			
→ 6 - Accept Alert				
→ 7 - Click	a_Elads			
→ 8 - Click	button_Nincs itt a knyvec			
→ 9 - Set Text	input_Knyv_cme_ujkonyv	"Vuk"		
→ 10 - Set Text	input_Kiadsi ve_kiadasev	"1965"		
→ 11 - Set Text	input_SzerzKiad neve_nev	"Fekete István"		
→ 12 - Click	button_Felvtel			
→ 13 - Delay		1		
→ 14 - Accept Alert				
→ 15 - Set Text	input_Profilom_konyvvv	"Vu"		
→ 16 - Click	div_Vuk			
→ 17 - Set Text	input_r (Ft)_prizeinput	"1200"		
→ 18 - Click	input_Csere_Csere			
→ 19 - Upload File	vuk	"D:\Programming\Katalo		
→ 20 - Click	button_Elads			
→ 21 - Delay		3		
→ 22 - Accept Alert				
→ 23 - Click	a_Vsrls			
→ 24 - Click	a_Profilom			
→ 25 - Click	input_button			
→ 26 - Click	img			
→ 27 - Click	button_Megjells Eladottk			
→ 28 - Click	a_Profilom			
→ 29 - Click	input_button			
→ 30 - Click	img			
→ 31 - Click	button_Hirdets Trlse			
→ 32 - Click	input_button			

28. ábra: Könyveladás tesztje

A második tesztben azt teszteljük, hogy bejelentkezés után feltudunk-e tölteni könyvet, amit később ki is választunk, árat adunk annak, cserét is engedélyezzük és képfeltöltés után megtudjuk-e jelölni eladottként és törölni is tudjuk az oldalról.


```

Test Cases/masodik (121.365s)
  1 - openBrowser("") (1.977s)
  2 - navigateToUrl("http://localhost:3000/Login") (0.542s)
  3 - setText(findTestObject("Object Repository/Page_Bookie/input_E-mail_Felhasznlnv_email"), "testprofil") (0.746s)
  4 - setEncryptedText(findTestObject("Object Repository/Page_Bookie/input_Jelsz_passwd1"), "P9ET2sDE0SE=") (0.321s)
  5 - click(findTestObject("Object Repository/Page_Bookie/input_Jelsz_login")) (0.299s)
  6 - acceptAlert() (0.029s)
  7 - click(findTestObject("Object Repository/Page_Bookie/a_Elads")) (30.238s)
  8 - click(findTestObject("Object Repository/Page_Bookie/button_Nincs itt a knyved")) (0.522s)
  9 - setText(findTestObject("Object Repository/Page_Bookie/input_Knyv_cme_ujkonyvvv"), "Vuk") (0.761s)
  10 - setText(findTestObject("Object Repository/Page_Bookie/input_Kiadsi_ve_kiadasev"), "1965") (0.531s)
  11 - setText(findTestObject("Object Repository/Page_Bookie/input_SzerzKiad_neve_newauthor"), "Fekete István") (0.513s)
  12 - click(findTestObject("Object Repository/Page_Bookie/button_Felvtel")) (0.275s)
  13 - delay(1) (1.022s)
  14 - acceptAlert() (0.020s)
  15 - setText(findTestObject("Object Repository/Page_Bookie/input_Profilom_konyvvv"), "Vu") (0.840s)
  16 - click(findTestObject("Object Repository/Page_Bookie/div_Vuk")) (0.594s)
  17 - setText(findTestObject("Object Repository/Page_Bookie/input_r_Ft_prizeinput"), "1200") (0.753s)
  18 - click(findTestObject("Object Repository/Page_Bookie/input_Csere_Csere")) (0.279s)
  19 - uploadFile(findTestObject("Page_Bookie/vuk"), "D:\\Programming\\KatalonTests\\BookieFrontendTest\\Object Repository\\Page_Bookie\\konyv1.jpg") (5.101s)
  20 - click(findTestObject("Object Repository/Page_Bookie/button_Elads")) (0.283s)
  21 - delay(3) (3.011s)
  22 - acceptAlert() (0.014s)
  23 - click(findTestObject("Object Repository/Page_Bookie/a_Vsrsls")) (0.294s)
  24 - click(findTestObject("Object Repository/Page_Bookie/a_Profilom")) (30.090s)
  25 - click(findTestObject("Object Repository/Page_Bookie/input_button")) (0.507s)
  26 - click(findTestObject("Object Repository/Page_Bookie/img")) (0.504s)
  27 - click(findTestObject("Object Repository/Page_Bookie/button_Megjells_Eladottknt")) (0.849s)
  28 - click(findTestObject("Object Repository/Page_Bookie/a_Profilom")) (0.976s)
  29 - click(findTestObject("Object Repository/Page_Bookie/input_button")) (0.490s)
  30 - click(findTestObject("Object Repository/Page_Bookie/img")) (0.503s)
  31 - click(findTestObject("Object Repository/Page_Bookie/button_Hirdets_Trise")) (22.344s)
  32 - click(findTestObject("Object Repository/Page_Bookie/input_button")) (0.895s)

```

29. ábra: Sikeres teszt

10. Ábrajegyzék

1. ábra: Adatbázisunk táblái	6
2. ábra: Települések adatainak kinyerése, tárolása.....	6
3. ábra: Kinyert adatok átalakítása és fájlba írása	7
4. ábra: Kinyert adatok fájlba írása.....	7
5. ábra: ER-modell.....	8
6. ábra: Regisztráció oldal	13
7. ábra: Regisztráció oldal – rossz/hibás adatokkal.....	14
8. ábra: Bejelentkezés oldal.....	14
9. ábra: Menüsor – oldalak közötti váltás.....	15
10. ábra: Vásárlás – összes termék megjelenítése, szűrőkkel.....	15
11. ábra: Termék oldal – egy kiválasztott termékről bővebb információk.....	15
12. ábra: Termék feltöltése – termék adatainak kitöltése megfelelően	16
13. ábra: Könyv felvétel az adatbázisunkba	16
14. ábra: Chat felület – aktuális beszélgetés, illetve bal oldalt a chat előzmények,.....	17
15. ábra: Profil oldal menü	17
16. ábra: Profil adatok módosítása – Mentéskor jelszó megadása kötelező.....	18
17. ábra: Saját termékek – piros háttérszínű eladottnak jelölt.....	18
18. ábra: Saját termék konfigurálása	19
19. ábra: Kapcsolat felvétel – üzenet küldési lehetőség nekünk	19
20. ábra: Post metódus tesztjei	24
21. ábra: Authenticate metódus tesztjei.....	25
22. ábra: Put metódus tesztjei	26
23. ábra: Put metódus tesztjei.....	27
24. ábra: Get metódusok tesztjei.....	28
25. ábra: Delete metódus tesztjei.....	29
26. ábra: Könyvvétel tesztje	30
27. ábra: Sikeres teszt	30
28. ábra: Könyveladás tesztje	31
29. ábra: Sikeres teszt	32

11. Források

1. Logo: <https://chatgpt.com/>
2. <https://cloudinary.com/>
3. https://www.ksh.hu/apps/hntr.egyeb?p_lang=HU&p_sablon=LETOLTES
4. https://www.kemitenpet.hu/letoltes/tables.helyseg_hu.xls
5. <https://app.diagrams.net/>
6. <https://stackoverflow.com/questions/18883601/function-to-calculate-distance-between-two-coordinates>
7. <https://console.cloud.google.com/>
8. <https://codepen.io/robinllopis/pen/mLrRRB>
9. <https://katalon.com/>