Budapesti Műszaki Szakképzési Centrum

Bláthy Ottó Titusz Informatikai Technikum

ZÁRÓDOLGOZAT

Tanuló neve:	Molnár Dániel
Osztálya:	**5/13S**
Elkészítés éve:	**2021/2022**

Budapesti Műszaki Szakképzési Centrum
Bláthy Ottó Titusz Informatikai Technikum
1032 Budapest, Bécsi út 134.
OM azonosító: 203058
Feladatellátási hely: 002
Tel: +3612507995,
Email: madarasz.peter@blathy.info

ZÁRÓDOLGOZAT ADATLAP

A záródolgozat készítőjének neve:

Molnár Dániel

e-mail címe:

molnardani002@gmail.com

A záródolgozat témája:

A fő feladata programnak egy könyvtári adatbázis grafikus felhasználói felülete, ami kilistázza az eltárolt rekordokat, illetve ezeket szerkeszti, törli és ad hozzá új elemeket, illetve ezek keresésére is alkalmas. A program képes a könyvtár ügyfeleinek kölcsönzéseit is kezelni.

A záródolgozat címe:

Könyvtári alkalmazás

Konzulens tanár:

Zaletnyik Péter Tibor

Kelt: Budapest, 2021. december 04.

....

a záródolgozat készítőjének aláírása a konzulens tanár aláírása

Budapesti Műszaki Szakképzési Centrum
Bláthy Ottó Titusz Informatikai Technikum
1032 Budapest, Bécsi út 134.
OM azonosító: 203058
Feladatellátási hely: 002
Tel: +3612507995,
Email: madarasz.peter@blathy.info

EREDETISÉG NYILATKOZAT

Alulírott tanuló kijelentem, hogy a záródolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült záródolgozatomban található eredményeket az intézmény egy példányban archiválhatja.

Budapest, 2022.04.10

.....

tanuló aláírása

Budapesti Műszaki Szakképzési Centrum
Bláthy Ottó Titusz Informatikai Technikum
1032 Budapest, Bécsi út 134.
OM azonosító: 203058
Feladatellátási hely: 002
Tel: +3612507995,
Email: madarasz.peter@blathy.info

ZÁRÓDOLGOZATI KONZULTÁCIÓ IGAZOLÓ LAP

Alulírott Zaletnyik Péter Tibor konzulens tanár aláírásommal igazolom

Molnár Dániel nevű tanuló konzultációkon való részvételét[1].

Dátum	Téma	Tanuló aláírása	Tanár aláírása
2021.12.16	Témamegjelölés, projektvezetés		
2022.01.26	Adatbázis kapcsolatok tisztázása		
2022.02.09	Backend optimalizálás		
2022.03.02	Frontend optimalizálás		
2022.04.06	Dokumentáció véglegesítés		

[1] *(A záródolgozat leadásig kötelező a három konzultációs alkalom, ennek hiányában a záródolgozat nem értékelhető!)*

Tartalom

1 Bevezetés

Az én záródolgozatom témája/feladata egy könyvtári adatbázis grafikus megjelenítésére szolgáló C# nyelven írt, Grafikus Felhasználói Felületű (GUI) asztali alkalmazás készítése. Ebben a programban egyszerűen megjeleníthetőek az adott könyvtári adatbázisban lévő könyvek, szerzők és ügyfelek, illetve a könyvtárban dolgozó alkalmazottak. Használni a programot csak a könyvtár dolgozói tudják, akik felvehetik az ügyfeleket, és nekik írhatnak be kölcsönzéseket. Az adatbázisban lévő könyveket szintén csak a könyvtár alkalmazottjai tudják állítani vagy hozzáadni. Mivel már sok ehhez hasonló program létezik a világon, ezért próbáltam valami egyedi kinézetű, és ha nem is teljesen egyedi funkcionalitású, de a legtöbb egyéb program legjobb tulajdonságait egybevonni. Ha például egy nagyobb cég tervezte volna a programot, akkor figyelembe lehetne venni a modularitását. Ha esetleg más célra szeretnénk hasonló programot készíteni, például egy webshop raktárának grafikus vezérlője, ne kelljen teljesen új programot tervezni, hanem ennek a szükséges moduljait, például az adminisztrátori felület, könnyítve a továbbfejlesztést.

A program egy a háttérben, egy szerveren futó adatbázisban tárolt adatokat módosítják, törlik és adnak hozzá új sorokat/rekordokat a megfelelő táblákba. Szerencsére, ha valami probléma adódik az adatok bevitele közben könnyen lehet szerkeszteni őket, ugyan is az adott rekord kiválasztása után csak egy gomb nyomása segítségével elérhetőek az adott rekord adatai, és rögtön lehet szerkeszteni az elemeket, és ha esetleg az adat egy kapcsolatban van másik rekorddal, például egy könyv és egy kölcsönzés, és esetleg szerkesztődnek a könyv adatai, akkor a kölcsönzésnél is módosul az adott könyv, és ezért nem kell módosítani a kölcsönzést is. Előnye a programnak, hogy könnyen lehet benne a rekordokat keresni, az embereket a nevük alapján, illetve a könyveket a címük alapján, így is lerövidítve a várakozási időket egy kölcsönzés során.

A felület könnyen megtanulható, stílusos, egyértelmű, és egy viszonylag egyedi front-end dizájnnal rendelkezik, ezért felhasználó barát és igényes, de egyszerre egyértelmű is. Az adott könyvek háttérszíne alapján meg lehet állapítani, hogy szükséges lehet e további példányokat rendelni belőle. A program automatikusan jelzi, ha egy kölcsönzés lejárt, így könnyen lehet az ezekre hatályos büntetéseket, például késedelmi díj kiszabását elrendelni. Amint az ügyfél visszahozta a kikölcsönzött könyvet, lehet is törölni az adatbázisból a kölcsönzést, és az adott könyv darabszáma automatikusan megnő, ezzel jelezve, hogy visszahozták a könyvet, és egyel több elérhető darab van belőle.

2 Irodalomkutatás

2.1 Hasonló rendszerek

Már több erre a célra kifejlesztett programmal találkoztam a programom megírása előtt, amik segítettek átgondolni, hogy mi az, amit egy könyvtári programnak tudnia kell. Az egyik első, és legfontosabb program az a Magyarországon is használt Szirén Integrált Könyvtári rendszer volt, melynek a fejlesztése már 1985-ben elkezdődött, és jelenleg hét országban használják körülbelül 1600 könyvtárban. A Szirén rendszere sokkal komplexebb, mint az én programom, mert az több, és nagyobb adatbázisokkal tud kommunikálni, így akár még olyan helyekről is tud információt lekérni, ahol nincs teljesen bevezetve maga a program. Az én rendszerem abban különbözik ettől, hogy én minden különböző helyen elhelyezkedő könyvtárnak külön adatbázist készítenék, így könnyebben ellenőrizhető, hogy melyik könyvből hány darab elérhető. A szirénnek a keresési funkciója is sokkal bővebb, nem csak szavakat tud keresni a címekben, így sokkal összetettebb. A Szirén programnak az egyik legnagyobb „hátránya" az az, hogy mivel egy sokkal nagyobb adatbázist használ, így többe is kerül a fenttartása annak, viszont az én programomnak az árába nem lenne fixen belekalkulálva az adatbázis szerverének az ára, és a szerver tárhelyétől függően változna az ára. [1]

Amíg a Szirén csak a könyveket tárolja el, az én programom a kölcsönzéseket is eltárolja, így megvan minden programnak az előnyei és a hátrányai is, és én ezeket a hátrányokat szeretném kiküszöbölni az én programomban is.

Külföldön sok egyéb könyvtári szoftver létezik például a Destiny Library Manager, Alexandria, Alma és Aleph, LibAnswers. Van például a Cloud9, ami inkább csak iskolák könyvtáraival foglalkozik, amire szerintem az én programom is tökéletesen használható, mert egy iskolai könyvtár nincs összecsatlakoztatva se más iskolák könyvtáraival, se nagyobb könyvtárok adatbázisaival, mint például az Országos Széchényi Könyvtár, vagy az Oxford University szerverével. [2]

3 Használt keretrendszer

A program maga egy WPF program, ami a Windows Presentation Foundation rövid megfelelője. Ez a keretrendszer a .NET keretrendszer egyik fontosabb grafikus UI keretrendszere, amely a front-end rendszernek az XAML nyelvet használja, illetve back-end nyelvként a C# nyelvet használja. Az XAML az XML-re hasonlító úgynevezett jelölőnyelv, mivel minden jelölést úgy helyez el, hogy az eleje és a vége egy azonos jelű tag például a grid amiben a megjelenő kód helyezkedik el, egy <Grid> nyitó és egy </Grid> záró tag-el jelöli, illetve ha nincs benne tartalom, akkor rögtön le is lehet zárni csak egy <Grid/> használatával. Attribútumokat a nyitó tag-be lehet írni, például egy háttérszín beállítást is.

A program az MVVM technológiára épült, ami a Model View ViewModel / Modell Nézet NézetModell rövidítést jelenti. Az MVVM egy programvezetési minta, ami a nevéből eredő három részre bontja szét a programot. Ennek a mintának az az előnye, hogy szinte mindenre fel lehet használni, illetve a kód esetleg még újra felhasználható is lehet. A tervezői úgy készítették, hogy a már a WPF-ben meglévő adatkötési módot használja ki. A három különböző réteget több fő külön is tudja fejleszteni úgy, hogy ne akadályozzák egymást a munkában.

Model: A modell a back-end adatbázis kódon belüli modell-je, ezen keresztül tudjuk létrehozni a programban az egyes példányokat, minden tábla egy külön osztályként valósul meg, és ezeket a példányokat egy adatbázis kontextusban tárolja el a program. Ennek a használatával lehet új adatot hozzáadni, de módosítani, vagy törölni is, amennyiben már benne van a modellben. Ameddig viszont ezek a programban futó példányok nincsenek elmentve, addig nem éles adatok, és az adatbázisban nincs semmi módosítva.

View: Ez a réteg lesz az, amit a felhasználó fog használni saját maga. Ez a megjelenéssel foglalkozik, és itt jelennek meg az adatai a modellben eltárolt példányoknak. Ezek az oldalak, amik a programba vannak beépítve, itt lehet minden kinézetet változtatni.

ViewModel: Ez a réteg felel a két előző réteg szétválasztásáért. Minden oldalnak van egy saját ViewModell-je, amikre lehet adatkötés használatával illeszteni a View-ben megjelenített adatokat, és ide lehet beilleszteni a Model-ben lévő adatokat, és itt fognak létrejönni a fő változások a példányban.

4 Fejlesztői dokumentáció

4.1 Rendszerterv

Ahhoz, hogy a program futtatható lehessen, ellenőrizni kell néhány dolgot a programot futtató számítógépen. Első az, hogy mivel egy .EXE fájlt futtatunk, ezért a számítógép operációjának egy Windows típusnak kell lenni, mert más operációk nem tudják futtatni ezt a fájl kiterjesztést modifikációk nélkül. Igaz, hogy nem a leg erőforrás igényesebb a program, de azért így is jobb, ha például nem a processzor beintegrált videókártyáját használjuk, illetve ajánlott legalább egy két gigabájt memóriát használni, hogy a program és a XAMPP szerverei is simán akadályok és akadozások nélkül tudjanak együtt futni. XAMPP helyett lehet más adatbázis motort is használni, de én ezt választottam, mivel ezt tanultuk használni, és egyszerű is használni és gyorsan is indítható.

Ha szerkeszteni, vagy továbbfejleszteni kívánjuk a programot, ajánlott egy fejlesztői környezetet kiválasztani. Mivel én a Microsoft által fejlesztett Visual Studio programot használtam eddig, és ezt is tanultuk használni, ezért én ezt is ajánlom. Ebben könnyen lehet például az ablakok kinézetét grafikusan szerkeszteni, könnyítve a továbbfejlesztést, és magát a háttérben futó kódot is módosítani, így nem kell külön programokat használni minden feladatra. A program fejlesztéséhez szükséges bizonyos NuGet csomagok letöltésére a Visual Studion belül, melyek:

- Adatbázis műveletek:
    - Microsoft.EntityFrameworkCore (.Net 5.0-ra való kiadás)
    - Microsoft.EntityFrameworkCore.Tools (.Net 5.0-ra való kiadás)
    - Microsoft.EntityFrameworkCore.Relational (.Net 5.0-ra való kiadás)
    - MySQL.EntityFrameworkCore (.Net 5.0-ra való kiadás)
- MVVM technológia:
    - Microsoft.Toolkit.MVVM (legfrissebb verzió)
- Könyvek elérhetőségének szerkesztésénél használt mező:
    - Gu.WPF.NumericInput (legfrissebb verzió)

4.2 Kényes információk

Mivel a program az alkalmazottakról valamennyire kényes információkat tárol, gondolva az adott alkalmazott jelszójára, amivel be tud lépni, ezért fontos az, hogy az adatbázisban ezek az adatok titkosítva legyenek eltárolva, és csak az adott kulccsal lehessen megtörni a titkosított elemeket. Fontos, hogy titkosítva legyenek a jelszavak, ugyanis, ha valamilyen külső féltől származó támadás történne az adatbázisra, ne tudják ellopni az alkalmazottak jelszavát, és a program segítségével esetleg törölni, vagy módosítani kölcsönzéseket. A jelszavakat a program titkosítja egy a programkódban eltárolt kulccsal, így, ha valaki magában az adatbázisban eltárolt rekordot módosítja, az oda beírt jelszóval nem fog tudni belépni a programba, mivel az adatbázisban eltárolt „Titkosított” jelszót a program megpróbálja visszafejteni, de mivel az különbözik a beírt jelszóval, nem fogja tovább engedni a támadót. [3]

Fontos, hogy legyenek eltárolva az ügyfelek olyan információi is, amivel könnyen lehet azonosítani őket, például egy igazolványszám, illetve értesítési lehetőségek, például telefonszám, email cím is, ha esetleg késedelmi problémák merülnének fel. A születési dátummal pedig meg lehet akadályozni a felnőtt tartalmú jelzéssel ellátott könyvek kiskorúak kezébe való eljutását is.

4.3 Adatbázis

Jelenleg öt tábla van az adatbázisban eltárolva, melyekről elkészül a modell a programba. Van egy tábla, ami az adott könyv írójának adatait tartalmazza, ez össze van kötve magával a könyveket tartalmazó táblával egy egy a többhöz kapcsolattal, mivel egy írónak lehet több könyve, viszont a könyveknek általánosan csak egy írója/szerzője van. Ez kapcsolatban van a kölcsönzések táblájával szintén egy a többhöz kapcsolattal, mert egy könyvtípust többször ki lehet kölcsönözni, viszont egy kölcsönzésben csak egy könyvet lehet kölcsönözni. A kölcsönzések össze vannak kapcsolva az ügyfelek táblájával, szintén egy a többhöz kapcsolással, egy ügyfél többször kölcsönözhet, de egy kölcsönzést egyszerre csak egy fél tudja végrehajtani. Van külön még a programba való belépés érdekében egy tábla a könyvtár alkalmazottjainak is, ami nincs kapcsolatban mással, ebben tároljuk el a belépési információkat.

4.4 Author táblázat

Az author tábla célja a szerzők adatainak eltárolása, melynek elemei:

Id:

A szerző egyértelmű azonosítására szolgál, AutoIncrement, így az adatbázis rendszere automatikusan generálja az értékeket. (Pozitív egész szám)

Name:

A szerző teljes neve, az egyszerű felhasználói azonosításra szolgál. (Szöveg)

Dob.:

A szerző születési ideje. (Dátum)

Pob.:

A szerző születési helye. (Szöveg)

Nationality:

A szerző nemzetisége. (Szöveg)

4.5 Book táblázat

A book tábla célja a könyvek adatainak eltárolása, melynek elemei:

Id:

A könyv egyértelmű azonosítására szolgál, AutoIncrement, így az adatbázis rendszere automatikusan generálja az értékeket. (Pozitív egész szám)

Title:

A könyv címe, segít a megfelelő könyv megkeresésében, így is könnyen azonosítható (Szöveg)

Description:

A könyv leírása, a programban a kilistázáskor rövid maximum 50 karakteres szövegben jelenik meg, de szerkesztéskor az egész átírható. (Szöveg)

Releasedate:

A könyv kiadásának dátuma (Dátum)

Genre:

A könyv műfaja, a programban előre beírt műfajokból lehet választani. (Szöveg)

IMG:

A könyv borítóképének képe eltárolva az adatbázisban. (Medium Blob)

Language:

A könyv elérhető nyelve, csak két karakter hosszú lehet az ISO 639-1 es szabványa alapján. (Szöveg)

Publisher:

A könyv kiadója. (Szöveg)

Amount:

A könyv jelenlegi elérhető mennyisége. Ha nulla, akkor nem jelenik meg kölcsönzés hozzáadásánál, csak ha módosítjuk, ha valami probléma történne. (Pozitív egész szám)

ISBN:

A könyv 13 karakteres azonosító kódja, amivel nyilván van tartva az összes a Földön létező könyv adatbázisában (Szöveg a kötőjelek miatt)

Authorid:

A könyv szerzőjéhez való egyértelmű azonosítására szolgál. (Pozitív egész szám)

4.6 Booking táblázat

A booking tábla célja a kölcsönzések adatainak eltárolása, melynek elemei:

Id:

A kölcsönzés egyértelmű azonosítására szolgál, AutoIncrement, így az adatbázis rendszere automatikusan generálja az értékeket. (Pozitív egész szám)

Startdate:

A kölcsönzés kezdetének időpontja. (Dátum)

Enddate:

A kölcsönzés végének időpontja. Ebből állítja meg a program, hogy még legálisan az ügyfélnél van-e a könyv (Dátum)

Bookid:

A kölcsönzésben lévő könyv egyértelmű azonosítására szolgál. (Pozitív egész szám)

Userid:

A kölcsönző fél egyértelmű azonosítására szolgál. (Pozitív egész szám)

4.7 User táblázat

A user tábla célja az ügyfelek adatainak eltárolása, melynek elemei:

Id:

Az ügyfél egyértelmű azonosítására szolgál, AutoIncrement, így az adatbázis rendszere automatikusan generálja az értékeket. (Pozitív egész szám)

Name:

Az ügyfél teljes neve, segít a keresésében, így is könnyen azonosítható (Szöveg)

Dob:

Az ügyfél születési dátuma (Dátum)

Address, Email, Phone:

Az ügyfél értesítési lehetőségei (Szöveg)

Idcard:

Az ügyfél fizikai azonosítására szolgál (Szöveg)

4.8 Employee táblázat

Az employee tábla célja az alkalmazottak adatainak eltárolása, melynek elemei:

Id:

Az alkalmazott egyértelmű azonosítására szolgál, AutoIncrement, így az adatbázis rendszere automatikusan generálja az értékeket. (Pozitív egész szám)

Name:

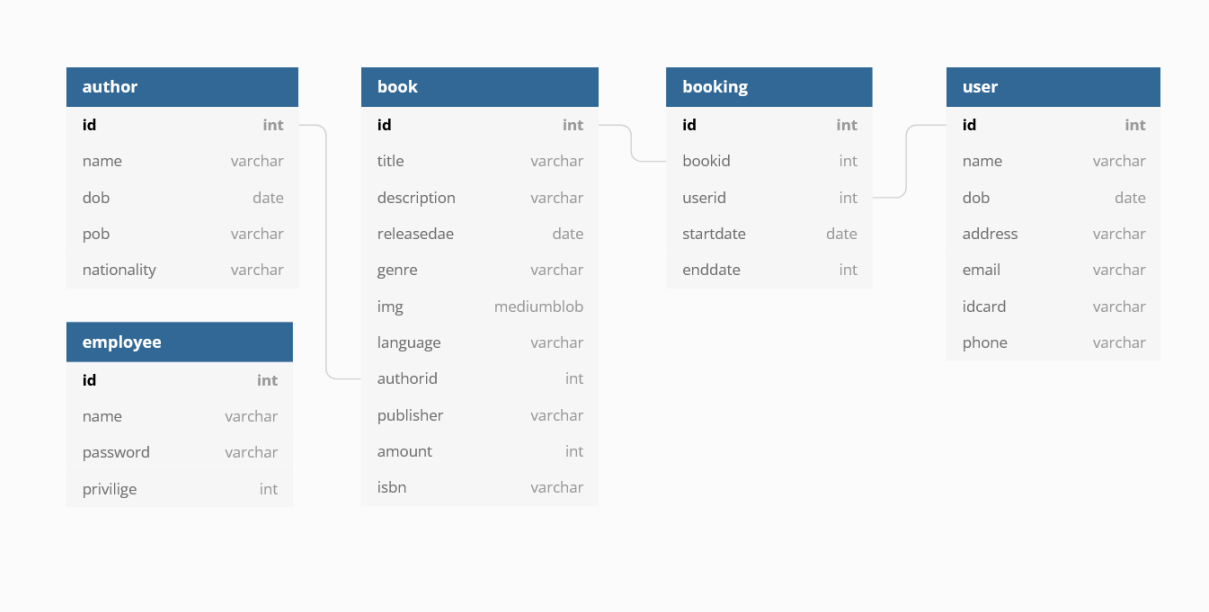
Az alkalmazott teljes neve, segít a keresésében, így is könnyen azonosítható. (Szöveg)

Password:

Az alkalmazott jelszava már titkosítva. (Szöveg)

Privilige:

Az alkalmazott joga. Amennyiben nulla, eladói jogokkal rendelkezik, amennyiben egy, adminisztrátori jogokkal rendelkezik. (Igaz/Hamis érték Pozitív egész számra vetítve)



4.1. ÁBRA AZ ADATBÁZIS GRAFIKUSAN

4.9 Kezelőfelület / UI

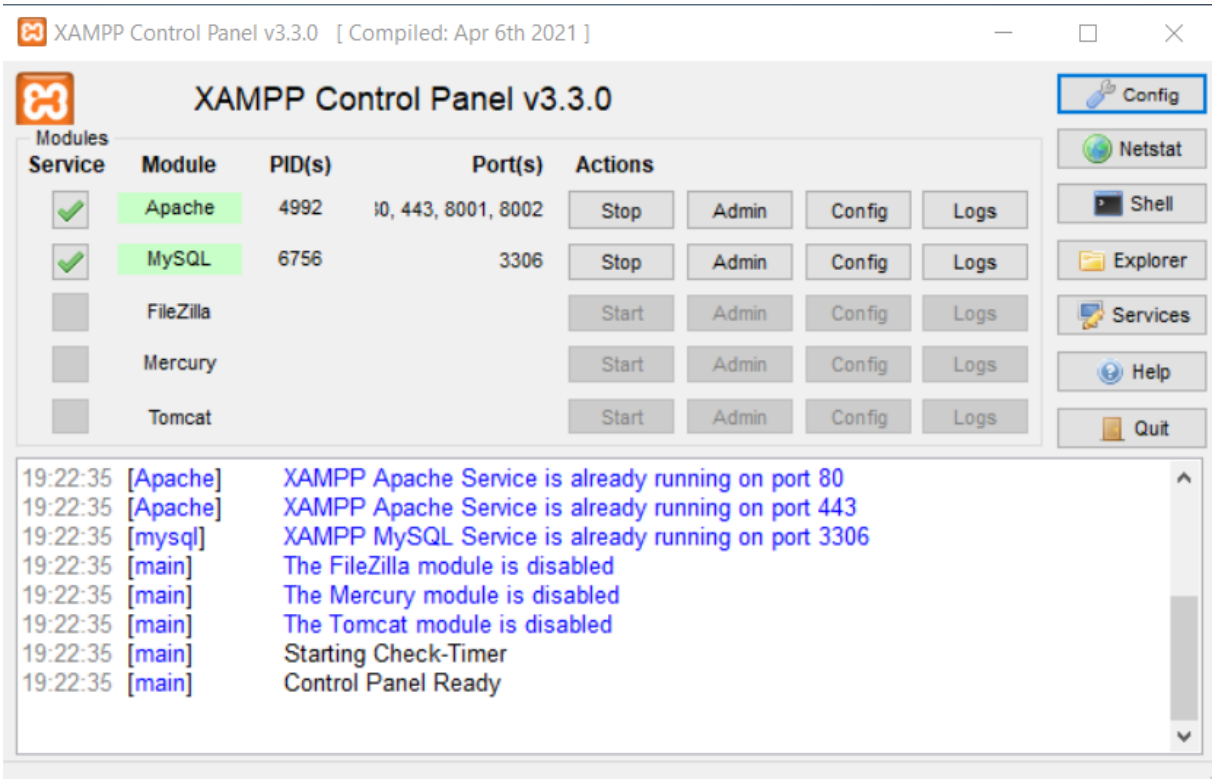
Az adatbázis tábláinak sorai a programban általában egy Listbox-ban vannak elhelyezve, illetve egy adat mintával vannak ábrázolva, amikben címkék vannak elhelyezve. Ezekbe vannak belekötve a ViewModelbe kötött adatnak az adatai, ha valami mást szeretnénk kijelezni a felhasználó velé, akkor ezt kell szerkeszteni. Ameddig a gombok nem változtatnak az oldal belső tartalmán, például a könyvek szerkesztésénél, igyekeztem az MVVM szabályai szerint a ViewModel-be írt ICommand-hoz kötöttem a működésüket, amik a program Business Logikájával valósítják meg a változásokat. Amennyiben változik a belső tartalom, a gombok egy a xaml.cs fájlba elhelyezett függvényt hajtanak végre. Amennyiben adatbevitelnél nincs beírva adat, addig a szövegdoboz mögött megjelenik, hogy milyen adatot kell bevinni egy szerintem oda illő képpel, majd, ha elkezdünk beleírni, a szövegdoboz mögött eltűnik a mögötte lévő kép és az írás is.

A program végső front-end dizájnján sok időt gondolkoztam, ugyanis számomra fontos, hogy a lehető legoptimálisabban használható, de még úgy is igényesen kinéző legyen. Inspirációnak sok más egyéb célra használt program kinézetét próbáltam venni, és szerencsére sok igényesen kinéző programot találtam erre a célra. Több egyedi kinézetet próbáltam készíteni, de ez volt szerintem a legmegfelelőbb erre a célra, és akiket kérdeztem, nagyrészt ez tetszett számukra is. Magára a felépítésre nem használtam semmi speciális szoftvert, a Visual Studio saját front-end tervező kinézete tökéletes volt számomra, habár léteznek erre a célra gyártott programok is, például a Blend for Visual Studio is, amit kipróbáltam, de nem találtam nagy előnyét az alapból a Visual Studioba beintegrált felülettel szemben. Először papíron próbáltam megtervezni, aztán a program segítségével létrehoztam. [4]

Legnagyobb problémát számomra a kinézet elkészítése közben a háttérben lévő címkék és képek eltűntetése volt, sok különböző móddal próbáltam elérni, hogy működővé váljon a program, de vagy a képek eltűnése nem volt lehetséges, vagy például a jelszó beviteli mezője nem volt működőképes. Először még maga a jelszó beviteli mező is probléma volt, mivel passwordbox-ra, amit használ a program, nem lehet az MVVM szabályai szerint adatot kötni. Ezért megpróbáltam egy tulajdonságot beleilleszteni a jelszó mező backend kódjába, de mivel ez nem volt működőképes, ezért a program háttér kódjában egy a lenyomott billentyű felemelkedésénél elinduló függvény segítségével ellenőrzi le a program a jelszót. Ez után, hogy létre tudjam hozni a watermark-ot, ami eltűnik, megpróbáltam az XAML kódban egy saját stílus bevezetésével elérni, ami le ellenőrzi, hogy a fő szöveg doboz, amibe beírjuk az adatot és amire kötve van a ViewModel-en belüli példány, üres-e, és aszerint teszi láthatatlanná a szövegét a benne található elemeknek, de a jelszó bevitelnél ez a funkció se működik biztonsági szempontokból. A végső megoldás egy az összes beviteli mező által használt függvény lett, ami ellenőrzi a függvényt lefuttató grafikai objektumnak az egyedi azonosító nevét miután átkonvertálta szövegdobozra a típusát, majd az ahhoz a névhez illő egyedi azonosítójú képet és címkét láthatóvá, vagy láthatatlanná teszi. Ennek a függvénynek egy változata minden beviteli mezőre van hatása, amennyiben megadjuk az azonosítókat, és adunk egy új ágat az elágazásnak, ami ezt végrehajtja.

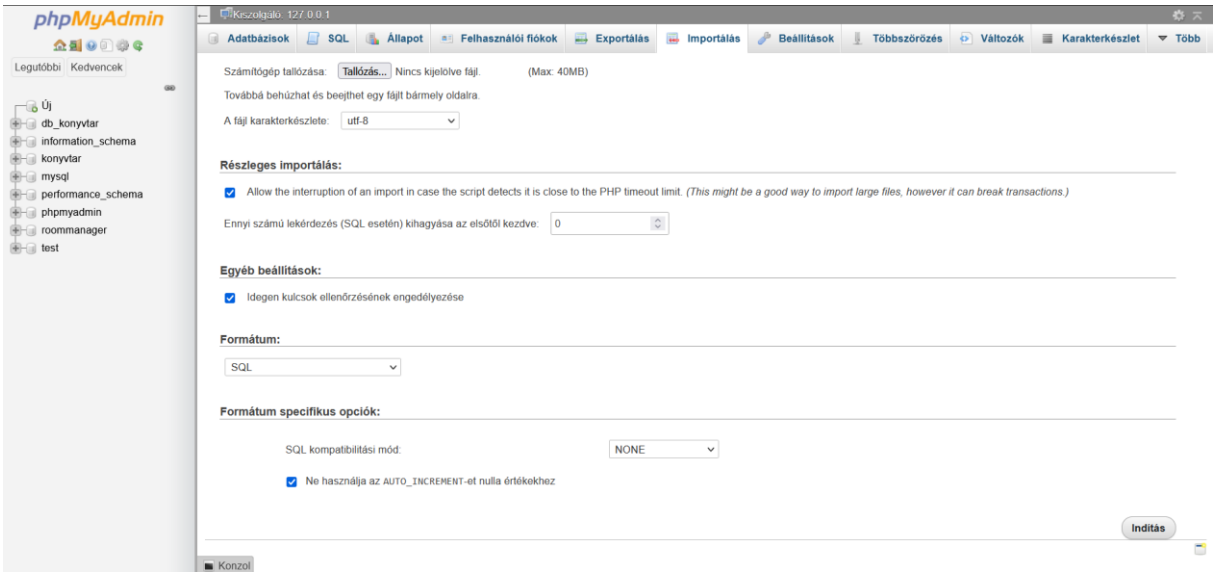
5 Felhasználói dokumentáció

Mielőtt el tudnánk indítani az alkalmazást, elérhetővé kell tenni az adatbázist. Én erre a célra a XAMPP [5] nevű webszerver-szoftvercsomagot használom, amire fel lehet tölteni egy SQL kódból az adatbázist, és a benne lévő rekordokat. Telepítésnél ki kell választani a MySQL és az Apache szerver opciót. A XAMPP indítsa után el kell indítani ezt a két szervert.

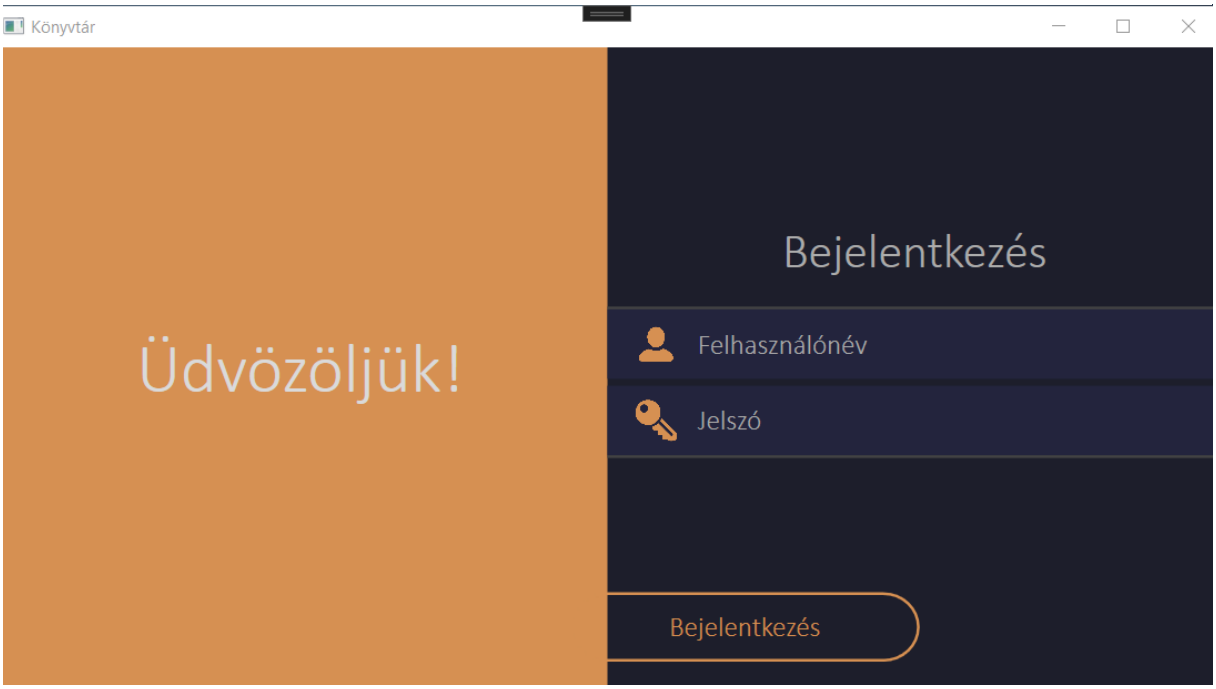


5.1. ÁBRA XAMPP FUTTATÁS

Ez után a MySQL admin gombjára kattintva az alapból beállított böngészőben megjelenik a phpMyAdmin felülete, amin keresztül lehet importálni az .SQL fájlt, ami tartalmazza az adatbázist. Fontos, hogy a létrehozott adatbázis neve „könyvtár” legyen, vagy át kell írni a kapcsolódási paramétereket a létrehozott adatbázis nevére. Importálás után az adatbázis elérhetővé válik.



5.3. ÁBRA PHPMYADMIN



5.2. ÁBRA BELÉPÉSI FELÜLET

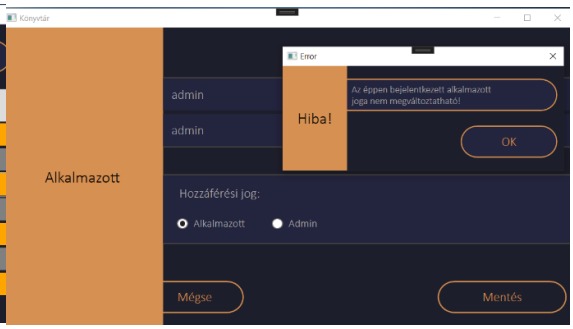
Miután elérhetővé vált az adatbázis, a program egy bejelentkező felülettel indul, ahonnan a megfelelő alkalmazott a megfelelő oldalra tudja átirányítani magát. Felhasználónevet és jelszót egy alkalmazott egy a könyvtárban dolgozó adminisztrátortól tud kérni, amennyibben nincsen, vagy szeretné megváltoztatni. Ezeket az adatokat az adatbázisban automatikusan elmenti a program, a jelszókat természetesen titkosítva, hiszen kényes információ.

5.1 Adminisztrátori belépés

Amennyiben adminisztrátori jogosultsága van a belépett félnek, új alkalmazottat hozzáadni, módosítani vagy törölni a megfelelő gombok megnyomásával lehet. Ha módosítani, vagy törölni szeretnénk, akkor ki kell jelölni az adott alkalmazottat, e nélkül a program nem enged módosítást. Adatok módosítását a megfelelő mezőkbe bevitt értékkel lehet végrehajtani, majd a mentés gombra való kattintással a program meg is teszi a beírt módosításokat.

Ha minden felhasználó törlődne az adatbázisból, akkor viszont magába az adatbázisba kell beírni az új adatokat, mivel nincsen alapértelmezetten elfogadott bejelentkezési lehetőség. Amennyiben a program segítségével kellene törölni az alkalmazottak adatait, a program nem engedi, hogy minden alkalmazott jogosultságát törölje, egy adminisztrátori belépés mindig meg marad, mert nem engedi meg azt, hogy az éppen bejelentkezett adminisztrátor adatait töröljék, se az ő belépési jogának a módosítását.

5.4. ÁBRA HIBA 1



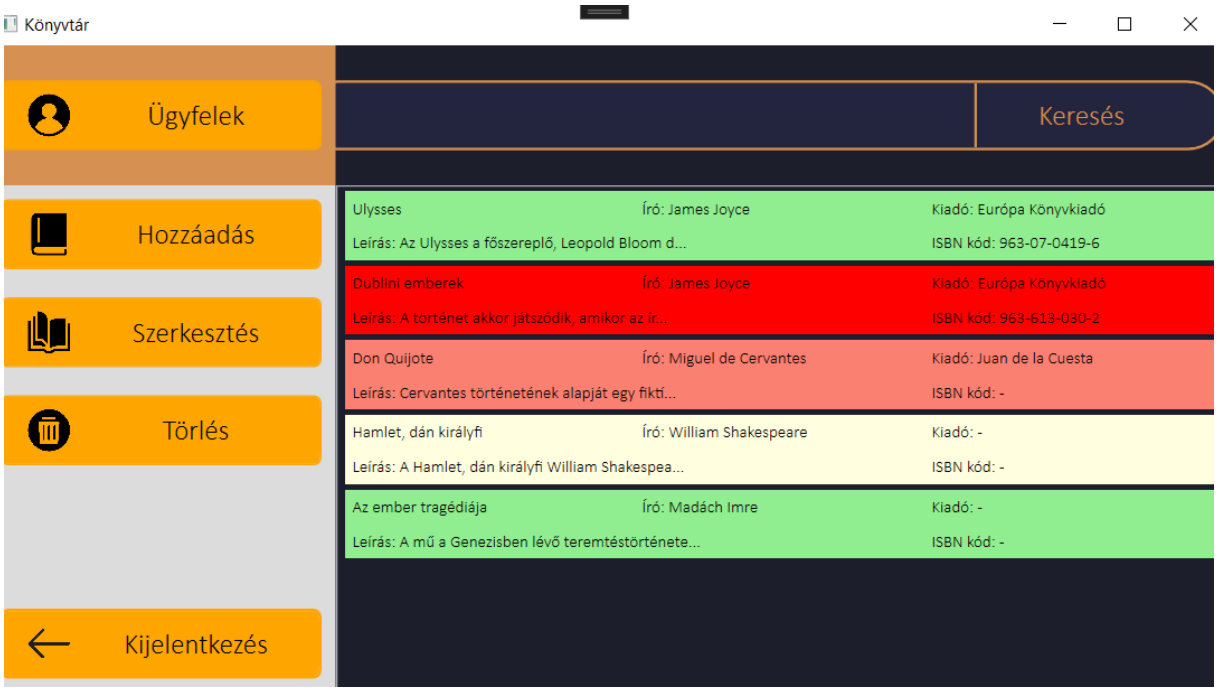
5.5. ÁBRA HIBA 2

5.2 Könyvtári alkalmazotti belépés

Amennyiben normális könyvtári alkalmazotti joga van a belépni kívánó dolgozónak, akkor tovább irányítja arra a felületre a program őt, amely kijelzi a számára az adatbázisban szereplő könyveket, és az eltárolt kölcsönző ügyfeleket. A program először a könyveket jeleníti meg, melyeknek a háttérszíne az adott könyvtípus elérhetőségéből adódóan változik, melyek:

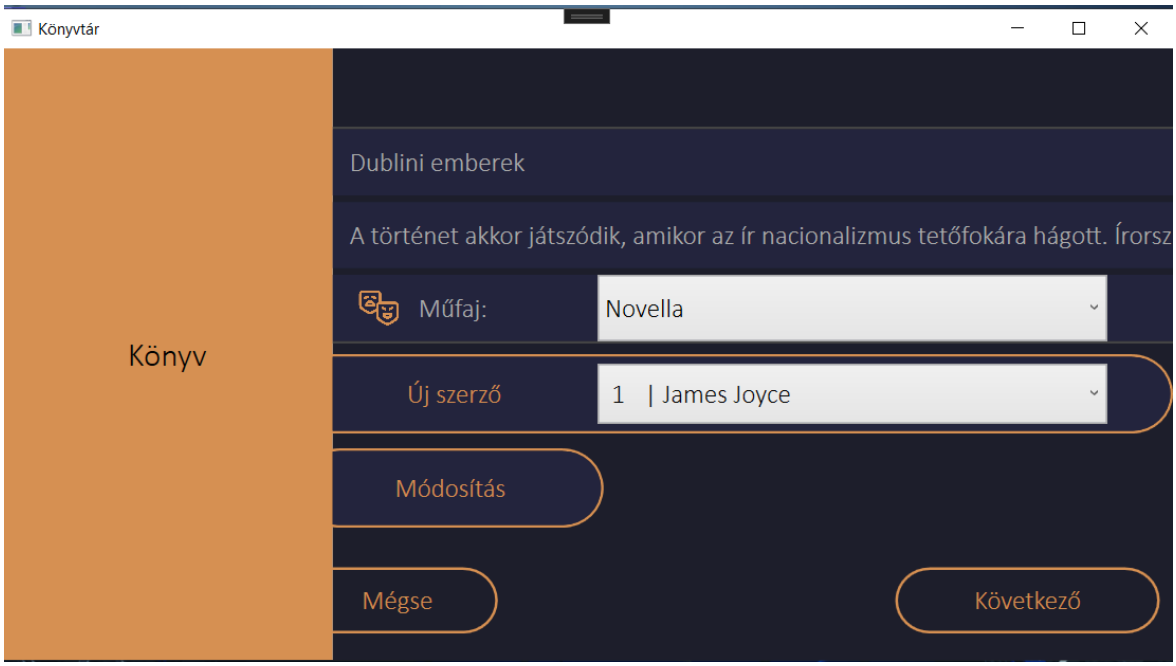
- 0 => Piros, ezeket a könyveket nem lehet kikölcsönözni már,
- 1-5 => Halvány piros, későbbi verziókban tervezem, hogy a program automatikusan jelezze az alkalmazott felé, hogy érdemes lenne új példányokat beszerezni.
- 5-10 => Halványsárga,
- 10+ => Zöld, van még elegendő mennyiség eltárolva.

Új könyvet hozzáadni, módosítani vagy törölni a bal oldalon elhelyezkedő gombok megnyomásával lehet, illetve törlésnél és szerkesztésnél kiválasztva kell lennie egy rekordnak, amin a munkát végezzük. A program felső sorában található egy keresési sor, amibe a könyv címének, vagy a cím egy részletének a beírásával ki lehet keresni az adatbázisban lévő leszűkített elemeket.



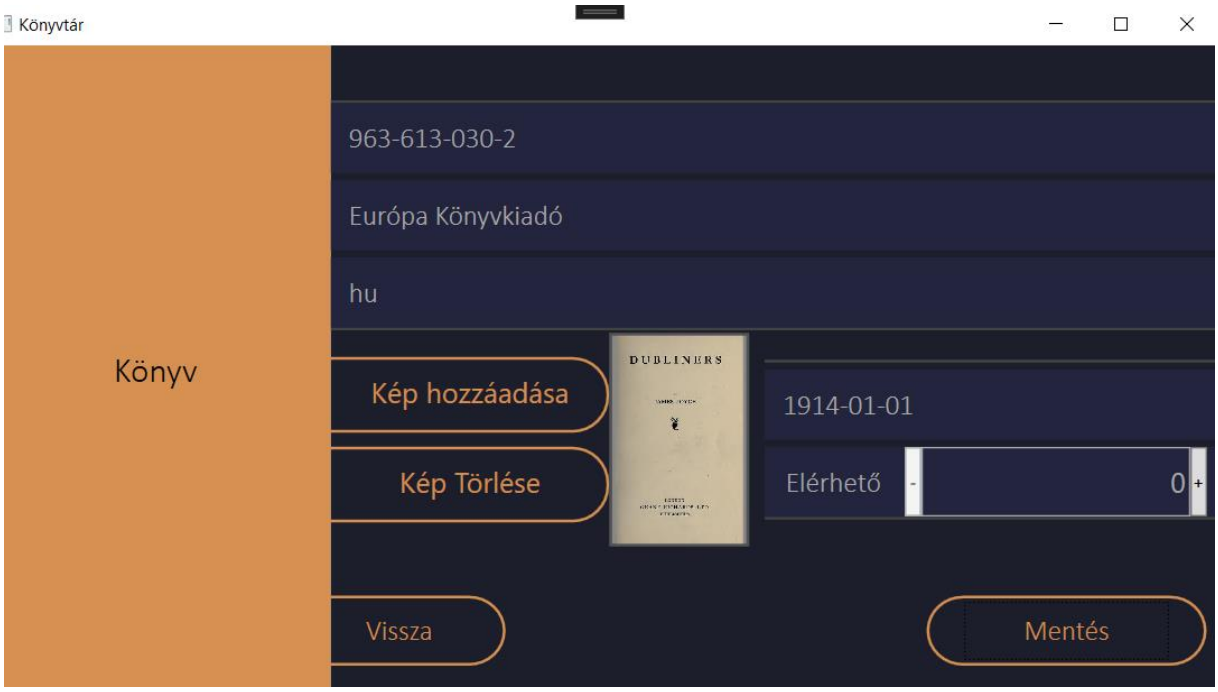
5.6. ÁBRA KÖNYVEK OLDAL

Amennyiben adatot szeretnénk beírni, vagy módosítani, a programon belül megnyílik egy új oldal, aminek használatával lehet ezt megtenni. Amennyiben szerkesztünk, a program automatikusan beilleszti az adott példány adatait, ha pedig új, jelzi azt, hogy melyik mezőbe milyen adatot kell beírni. Általában ezt egy szöveges beviteli mezőn keresztül lehet megtenni, például a könyv címe, de van olyan eset, ahol legördülő menüvel lehet kiválasztani adatokat, például a szerzők adatbázisából, vagy a programkódban eltárolt műfajokból. Ha szerzőt szeretnénk hozzáadni, vagy eltávolítani, akkor azt a megfelelő gombbal el lehet érni. Törlést nem gondoltam szükségesnek, mert általában egy szerző több művet is tud írni, és ezzel véletlenül törölhetjük a többiművét is.



5.7. ÁBRA KÖNYV SZERKESZTÉSE 1

Miután az adatokat beillesztettük, a következő oldalra jutásnál további értékeket lehet beilleszteni. Itt lehet megadni a könyv borítóját is, ám amennyiben nincs elérhető kép, mentéskor a program automatikusan felajánlja, hogy beilleszt egy alapból beállított képet helyfoglalás céljából. A vissza gombbal át lehet menni az előző oldalra, de a már beírt módosításokat a program megjegyzi, így nem kell attól tartani, hogy ezeket újra be kell vinni.



5.8. ÁBRA KÖNYV SZERKESZTÉSE 2

A mentés gombra való kattintással pedig az adatbázisban létrejönnek a módosítások.

Amennyiben az ügyfelek listáját szeretnénk lekérni, a bal felső sarokban lévő gombbal lehet változtatni a két listán belül. Új ügyfelet hozzáadni, módosítani, vagy törölni a már ismert módon lehet. Itt is elérhető a keresési fül, itt lehet az ügyfelek között keresni. Az adatok háttérszíne a kikölcsönzött könyvektől függ, amennyiben van kölcsönzött könyve piros, egyébként zöld színnel jelenik meg.

5.9. ÁBRA ÜGYFELEK OLDAL

Ha kijelölünk egy ügyfelet, akkor a kölcsönzések gombra kattintva ellenőrizhetjük az összes elmentett kölcsönzését, illetve itt is tudunk új rekordot hozzáadni, szerkeszteni, vagy törölni, amennyiben ezek ki vannak jelölve szükség szerint. A háttérszínnek itt is jelentős értéke van, ezzel jelzi a program, hogy az adott könyvet vissza kellett volna már szolgáltatni piros színnel, vagy még az ügyfélnél lehet a kikölcsönzött könyv szabályosan, és nem járt le a határidő zöld színnel.

5.10. ÁBRA ADOTT ÜGYFÉL KÖLCSÖNZÉSEI

5.11. ÁBRA KÖLCSÖNZÉS KIVÁLASZTÁSA

A módosítási oldalon lehet beírni a kölcsönzés kezdeti és végső dátumát, illetve legördülő menüvel lehet kiválasztani a könyvet, amit az ügyfél szeretne kikölcsönözni. Magát az ügyfelet már nem kell kiválasztani, mivel az előbb kiválasztott ügyfélhez adjuk hozzá alapból a kölcsönzést. Amennyiben az adott könyvből elfogyott az elérhető mennyiség, a program nem jelzi ki, mint választási lehetőség. Kijelentkezni a két főoldalon lévő kijelentkezés gombbal lehetséges.

6 Összegzés és további tervek

Összegzésül a program egy könyvtári adatbázis kezelését végzi el, amely a XAMPP nevű program segítségével kommunikál egy MYSQL adatbázissal, és képes az abban eltárolt rekordokat megjeleníteni, törölni és szerkeszteni, illetve képes új rekordokat létrehozni. A program funkcióit a grafikus interfészre ráillesztett gombokkal tudjuk véghez vinni. A programnak szerintem az egyik legnagyobb előnye az az, hogy kis módosítással más adatokat is meg lehetne jeleníteni, a jelenlegi munkakörén belül is, de ha más adatbázist kellene használni, akkor az MVVM technológiája szerint módosításokkal teljesen más célú raktári programot is lehetne készíteni, vagy esetleg egy hotel szoba és bérlés menedzsment szoftvere is lehetne.

Későbbi továbbfejlesztett verziókba tervezek további funkciókat adni, mint például további adatokból való keresés és szűrés, például műfajok alapján, vagy szerzőket szűrve, ezzel is megkönnyebbítve az alkalmazottak munkáját. Lehetne csinálni egy külön felületet a könyvtárba járó embereknek, akik a könyvtárban lévő számítógépek segítségével saját maguknak is ki tudnának kölcsönözni könyveket, egy bolti önkiszolgáló pulthoz hasonlóan. Lehetne készíteni a könyvtárnak egy egyedi weboldalt, amivel az ügyfelek ellenőrizhetik a saját kölcsönzéseiknek a lejáratát, ezzel is elkerülve a késedelmeket. Ha pedig túl sokat késnének, a program magától blokkolná őket a további kölcsönzésektől, amíg vissza nem hozzák a kikölcsönzött könyveket.

Mivel az ügyfeleknek eltároljuk az elérhetőségeiket, ezért lehetne a fő alkalmazotti programnak egy olyan további funkciót beépíteni, ami levelet, vagy emailt küld az ügyfeleknek, ha valami probléma történne, vagy esetleg azzal is emlékeztetőket küldeni. Tudna küldeni személyre szabott ajánló leveleket, ezzel is ösztönözve embereket, hogy többször látogassák a könyvtárat. Ahhoz, hogy ez létre jöhessen, az ügyfeleknek fel kell iratkozniuk a hírlevélre, ami kiküldi a leveleket a preferált módon, postai úton, vagy emailben, esetleg telefonos módon keresztül is, ha van olyan alkalmazottja a könyvtárnak, aki ezzel foglalkozna. Ahhoz, hogy ez működőképes legyen, két különböző fajta módszere gondoltam, amikben az egyiknél vagy az adminisztrátorok vagy más egyéb alkalmazottak írnak emailt a programban, vagy a program induláskor ellenőrizné, hogy milyen leveleket kell írnia, és ezeket automatikusan elküldi az ügyfeleknek, szintén könnyítve a munkát, és segítve az alkalmazottakat.

Természetesen lehetne még fejleszteni a program felhasználói felületén, esetleg egy alkalmazott vagy később egy ügyfél által kiválasztható további grafikus kinézeteket készíteni, mint például egy sötét és világos mód közötti váltás. A program tudna reagálni az ablak méretéhez, vagy ha például egy látássérült ember nem tudja tökéletesen elolvasni a betűket, akkor bele tudjon nagyítani, vagy kicsinyíteni a programba, ezzel is segítve és gyorsítva az automatikus, önkiszolgáló kölcsönzéseket. Fontos lenne még az, hogy különböző nyelvekre át lehessen váltani a programot, ha egy ügyfél lép be, mivel nem csak magyar nyelvű könyveket tartalmaz a rendszer, ezért külföldi emberek is tudják használni a programot úgy, hogy nem kell zavarni alkalmazottakat.

Beépítenék még egy automatikus élő frissítés funkciót is, ha esetleg egy alkalmazott változtatna egy elemen az adatbázisban, azt más gépeken is frissítse rögtön, nem kelljen manuálisan frissítenie a programnak az adatbázisból.

Mivel egy könyvtárban nem kötelező csak könyveket tárolni, hanem lehet esetleg CD-re, vagy DVD-re írt hangoskönyveket, más iratokat, önéletrajzokat is kölcsönözni, ezért szeretnék egy azokhoz is működő adatbázis fejlesztést is végrehajtani, és ehhez hasonlóan a programot is továbbfejleszteni.

7 Felhasznált irodalmak

[1] SZIRÉN könyvtári rendszer saját weboldala: http://www.sziren.com/index_sziren.htm

[2] Külföldi könyvtár management rendszerek: https://www.g2.com/categories/library-management

[3] Jelszó titkosítás: http://curlybrackets.com/posts/43017/how-to-encrypt-and-decrypt-a-string-in-c-sharp

[4] WPF dizájn ötletek: https://dribbble.com/tags/wpf

[5] XAMPP a backendhez: https://www.apachefriends.org/hu/index.html

8 Ábrajegyzék