



ZÁRÓDOLGOZAT

Készítették:

Ambrus Kristóf-Oláh Márk-Szesztai Péter

Konzulensek:

Kerényi Róbert Nándor, Kiss Roland, Németh Bence, Négyesi Gábor, Farkas Zoltán

Miskolc

2024.

Miskolci SZC Kandó Kálmán Informatikai Technikum

Miskolci Szakképzési Centrum

SZOFTVERFEJLESZTŐ- ÉS TESZTELŐ SZAK

KaKöNyir

Kandó Könyvtárnyilvántartó rendszer

Készítették:

Ambrus Kristóf-Oláh Márk-Szesztai Péter

Miskolc

2023-2024

Tartalomjegyzék

1	BEVEZETÉS	1
2	FEJLESZTŐI ESZKÖZÖK.....	2
2.1	ADATBÁZIS:.....	2
2.1.1	WampServer:.....	2
2.1.1.1	MySQL:	3
2.1.1.2	phpMyAdmin:	3
2.2	BACKEND:.....	3
2.2.1	ASP.NET:.....	4
2.2.2	Web Api:	6
2.2.3	Entity Framework:	6
2.2.4	CORS Policy.....	6
2.2.5	A C#	7
2.2.6	JWT Token	8
2.2.7	CRUD (Create, Read, Update, Delete)	9
2.2.8	Az ORM.....	10
2.2.9	Postman:	11
2.3	FRONTEND:	12
2.3.1	React:	15
2.3.1.1	Előnyei:	16
2.3.1.2	Hátrányai/Hátrányosságai:	16
2.3.1.3	React felület telepítése:.....	17
2.3.2	Node.js:.....	18
2.3.2.1	A Node.js története:	18
2.3.2.2	Node.js architektúra:	18
2.3.2.3	Node.js modul-jai:	19
2.3.2.4	Node csomagkezelő:.....	19
2.3.3	Visual Studio Code:.....	19
2.3.3.1	Visual Studio vs Visual Studio Code:	20
2.3.3.2	Különbségek:	21
2.4	A WPF:	21
2.4.1	Asztali Alkalmazásunk:	22
3	TÁRHELYEK:	25
3.1	FTP (FILE TRANSFER PROTOCOL):.....	25
3.1.1	FTP működése:	25
3.1.2	FTP fontossága és használhatósága:	26
3.1.3	Képfájlok:.....	26
4	CSAPAT KOMMUNIKÁCIÓS ESZKÖZÖK:	27
4.1	TRELLO:	27
4.2	GITHUB:	27
4.3	GOOGLE DRIVE:	27
4.4	MESSANGER:	28
4.5	DISCORD:	28
4.6	TEAMSPEAK 3:.....	28
5	TESZTELÉS:	29
6	ÖSSZEFOGLALÁS:	29
6.1	SAJÁT UTÓÉRTÉK:.....	29
6.1.1	Megvalósított elképzelések:	30
6.1.1.1	Adatbázis:	30

6.1.1.2	BackEnd:	30
6.1.1.3	FrontEnd:	31
6.1.1.4	Asztali Alkalmazás (WPF):	31
6.1.2	<i>Továbbfejleszthetőségek:</i>	32
7	KÖSZÖNETNYILVÁNÍTÁS	32
8	IRODALOMJEGYZÉK:	33



1 Bevezetés

A könyvtárak az adat, továbbá a szakértelem döntő forrásai, a digitalizáció korában pedig rendkívül fontossá vált a hatékony, valamint korszerű könyvtári nyilvántartási rendszerek kialakítása. Ez a dolgozat a mi általunk kialakított könyvtári nyilvántartási rendszerre összpontosít, a napi könyvtári működés nehézségeinek és követelményeinek megértése alapján.

Ebben a dolgozatban részletesen ismertetjük a szoftver létrehozásának processzusát, jellemzőit, továbbá előnyeit. Azt is megvitatjuk, hogy ez a program hogyan javítja a könyvtári szolgáltatások hatékonyságát és felhasználói élményét.

A dolgozatunk célja, hogy átfogó képet mutassunk arról, hogyan optimalizálhatók a könyvtárak a mai technológia segítségével, mi az előnye, hátránya és választ adunk arra is, hogy milyen kihívásokkal kellett szembesülnünk készítése közben.

Programunk az iskoláknak szeretne segíteni. Észrevettük, hogy az iskolai könyvtárunknak még nincs digitalizálva az adattárolása, ennek megteremtésével nemcsak a könyvtáros számára lesz átláthatóbb, nyomonkövethetőbb, hanem egyszerűbben, gyorsabban tanulóitársainknak is segítségére lesz bármely könyv választásában.

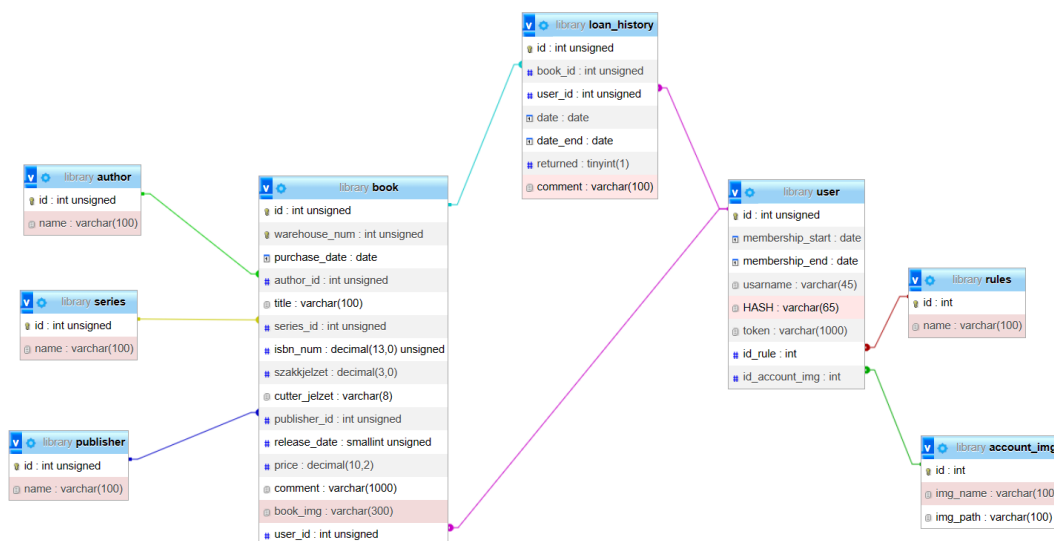
Jelmondatunk: *“Egy pár szavas intelem egy órára szól, egy könyv egy egész életre.”*



2 Fejlesztői eszközök

2.1 Adatbázis:

Adatbázisunk 8 relációból(táblából) áll. Látható a (1. ábra Adatbázis_szerkezet) képen. Egyik fő táblánk a **book** reláció, amely 15 mezőt tartalmaz, a másik fő táblánk az **user**, aminek 8 mezője van. Ennek a két táblának vannak al-táblái, amelyek név szerint a: **rules**, **account_img**, **loan_history**, **publisher**, **series** és az **author**. Az utolsó három reláció az a book táblának az al-táblái. Az author az adott könyv íróját, a series a könyv szériáját, a publisher pedig a kiadókat tárolja. A user-nek a rules al-táblájában tároljuk a felhasználók rangjait, az account_img-be pedig az oldalunkon választható profil képeket. A book és user táblának közös al-táblája pedig, a loan_history. Itt tároljuk a felhasználók által kikölcsönzött könyveket.



1. ábra (Adatbázis szerkezet)

2.1.1 WampServer:

A WampServer egy Windows webfejlesztő platform dinamikus webes alkalmazásokhoz, amelyek Apache2 szerver, PHP szkriptnyelvet, MySQL adatbázist és MariaDB-t használnak. A PhpMyAdmin segítségével könnyebben kezelhetjük az adatbázisokat, valamint a MySQL Workbench GUI-t a MySQL és MariaDB szervereket.

Az évek során kipróbált localhost szerverkörnyezetek közül a Wampserver a legjobb. Könnyen használható, több lehetőséget kínál, könnyen konfigurálható. Szerintünk a legjobban kiforrt és a leghatékonyabb helyi webszerver.

Romain Bourdon 2005-ben adta ki először a nagyvilágnak a programját. Azóta már nyílt forráskódú lett, és több fejlesztő is segítkezik a kódbázis fejlesztésében és karbantartásában. A WampServer a közösség által eléggé támogatott projekt, amely folyamatosan frissül és új funkciókkal bővül.

Amit jó tudni, a WAMP-ról, hogy a LAMP-ból származik (az L a **Linux**ot jelenti). A kettő között az egyetlen különbség az, hogy a WAMP-ot Windowshoz, míg a LAMP-ot Linux alapú operációs rendszerekhez használják.



A "WAMP" jelentése:

- „W” a Windows rövidítése, van még LAMP (Linux)-hoz és MAMP (Mac)-hez.
- „A” az Apache rövidítése. Az Apache a weblapok kiszolgálásáért felelős szerverszoftver. Amikor jön egy kérés felőlünk, hogy láss egy oldalt, az Apache HTTP-n keresztül teljesíti a kérést, és megmutatja a webhelyet.
- „M” a MySQL rövidítése. A MySQL feladata, hogy a szerver adatbázis-kezelő rendszere legyen. Tárolja az összes releváns információt, például a webhely tartalmát, felhasználói profiljait stb.
- „P” a PHP rövidítése. Ez az a programozási nyelv, amelyet a WordPress írásához használnak. Úgy működik, mint az egész szoftvercsomag ragasztója. A PHP az Apache-val együtt fut, és a MySQL-el kommunikál.

2.1.1.1 MySQL:

A MySQL az Oracle által kifejlesztett relációs adatbázis-kezelő rendszer (RDBMS), amely strukturált lekérdezési nyelven (SQL-en) alapul. Az adatbázis az adatok strukturált gyűjteménye. A MySQL-t egy svéd cég, a MySQL AB hozta létre, amelyet a svédek **David Axmark, Allan Larsson** és a finn **Michael "Monty" Widenius** alapítottak. A MySQL első verziója 1995. május 23-án jelent meg. Ezt a Sun Microsystems 2008-ban vásárolta meg, és aztán az Oracle 2010-ben felvásárolta a Sun-t.

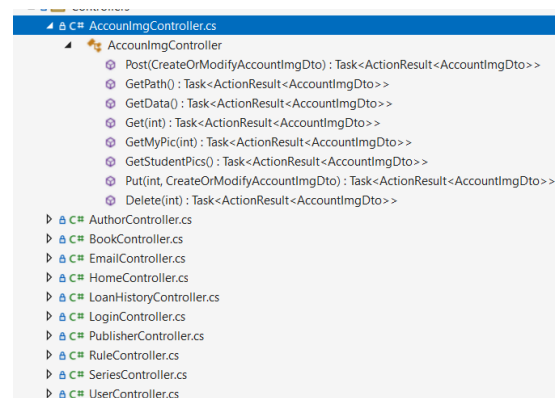
2.1.1.2 phpMyAdmin:

A **phpMyAdmin** egy PHP nyelven írt ingyenes szoftvereszköz, amely a MySQL webes adminisztrációjának kezelésére szolgál. A phpMyAdmin a **MySQL** és a **MariaDB** műveletek széles skáláját támogatja. A gyakran használt műveletek (adatbázisok, táblák, oszlopok, relációk, indexek, felhasználók, engedélyek stb. kezelése) végrehajthatók a felhasználói felületen keresztül, miközben továbbra is lehetősége van bármilyen SQL utasítás közvetlen végrehajtására.

A phpMyAdmin projekt a **Software Freedom Conservancy** tagja. Az SFC egy non-profit szervezet, amely segít népszerűsíteni, javítani, fejleszteni és megvédeni a szabad, ingyenes és nyílt forráskódú szoftvereket (**FLOSS**) tartalmazó projekteket.

2.2 BackEnd:

A backend elkészítéséhez WEB API-t használtunk, ami egy olyan keretrendszer, amely megkönnyíti az ügyfelek széles körét elérő HTTP-szolgáltatások létrehozását, beleértve a böngészőket és a mobileszközöket is. A WEB API ideális platform Restfull alkalmazások .NET-keretrendszeren történő létrehozásához. A projekt készítése során .NET 7-es verziót használtunk, mert amikor belekezdünk a projektbe, a 7. verzió volt a legkiforrottabb, és ahhoz volt elérhető a legjobb és legújabb, de még releváns biztonsági eszközök, a .NET 8 pedig még béta állapotban volt, és a legtöbb kiegészítő meg eszköz még nem volt frissítve erre a verzióra. A WEB API a HTTP kéréseket úgynevezett controllerekkel kezeli, és mikor az API egy kérést fogad, átirányítja a megfelelő controllerhez (lásd 2.ábra).



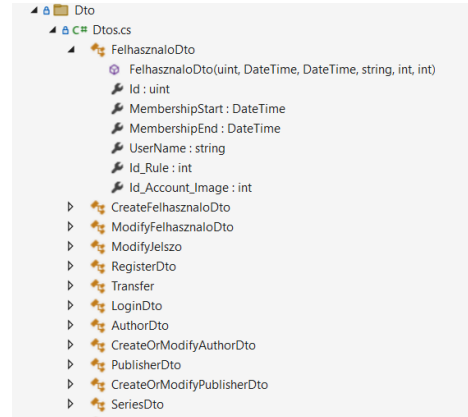
2. ábra(Contollerek)



A controllerek mindegyike tartalmazza a négy alap CRUD műveletet:

- **Create** : új információ felvitele az adatbázisba.
- **Read (vagy Get)** : az adat, vagy adatok lekérdezése és továbbítása
- **Update** : egy kiválasztott adat információinak megváltoztatása
- **Delete** : egy adat eltávolítása az adatbázisból

Ezen felül vannak olyan controllerek, amelyek speciális adatmegjelenítést, vagy adatmódosítást végeznek, ennek elősegítésére DTO-kat (Data Transfer Object) hoztunk létre (Lásd 3.ábra), ami egy olyan osztály, ahol megszabjuk milyen adatok kerülnek továbbításra, így gyorsítani és egyszerűsíteni tudjuk a kommunikációt és a felesleges adatok utaztatását, elrejtjük azokat a felhasználó elől, amelyre vagy nincs szüksége, vagy nem szabad hozzáférnie és megvédi az adatbázist attól, hogy a kellőnél több adat kerüljön elküldésre.



3. ábra (DTO-k)

A backend controllereinek egy része le van védve, így nem minden felhasználó férhet hozzá, csak azok, akiknek van jogosultságuk hozzá, például van admin joguk, vagy egyszerűen csak van felhasználói fiókjuk. Az azonosításra pedig JWT tokenet használtunk. Minden bejelentkezésnél generálunk egy tokenet, ami egy hosszú karakterlánc formájában titkosítva tartalmazza az azonosításhoz szükséges információkat, ami a kéréssel együtt elküldésre kerül a szervernek. Amennyiben az adott felhasználó nem rendelkezik megfelelő jogosultsággal, a szerver 401-es kóddal egy figyelmeztetést küld vissza a felhasználónak.



4. ábra (Swagger)

A controllerek tesztelésére Swaggetrt használtunk, ami egy nyílt forráskódú, az OpenAPI-on alapuló tool-ok összessége, aminek a felhasználói felülete elősegíti a controllerek átláthatóságát, dokumentálását és tesztelését. A biztonsági azonosítás teszteléséhez Postman-t használtunk, mivel a Swaggerben van egy hiba, ami sok esetben akkor is hibás azonosítást jelez vissza, amikor egyébként sikeres lenne. A Postman egy olyan szoftver alkalmazás, amely lehetővé teszi a fejlesztők számára az API-ok tesztelését és dokumentálását, rengeteg fajta lehetőséggel rendelkezik a Postman, így sok különböző környezet tesztelésére alkalmas.

2.2.1 ASP.NET:

Az **ASP.NET** egy nyílt forráskódú webes keretrendszer, amelyet a Microsoft hozott létre modern webalkalmazások és -szolgáltatások a .NET segítségével történő létrehozására, az 1.0-s verzió pedig 2002-ben jelent meg, hogy a fejlesztők dinamikus webalkalmazásokat, szolgáltatásokat és webhelyeket készíthessenek. A keretrendszer a szabványos HTTP protokollal működik, amely az összes webalkalmazásban használt szabványos protokoll.

A .NET pedig egy olyan fejlesztői platform, amely széles eszköz kínálattal, sok programozási nyelv támogatással rendelkezik, és számos különböző típusú alkalmazás létrehozására alkalmas. Az alapplatform olyan komponenseket tartalmaz, amelyek több különböző típusú alkalmazás létrehozására alkalmasak. A további keretrendszerek, például az ASP.NET, kiterjesztik a .NET-et bizonyos típusú alkalmazások létrehozásához szükséges összetevőkkel.

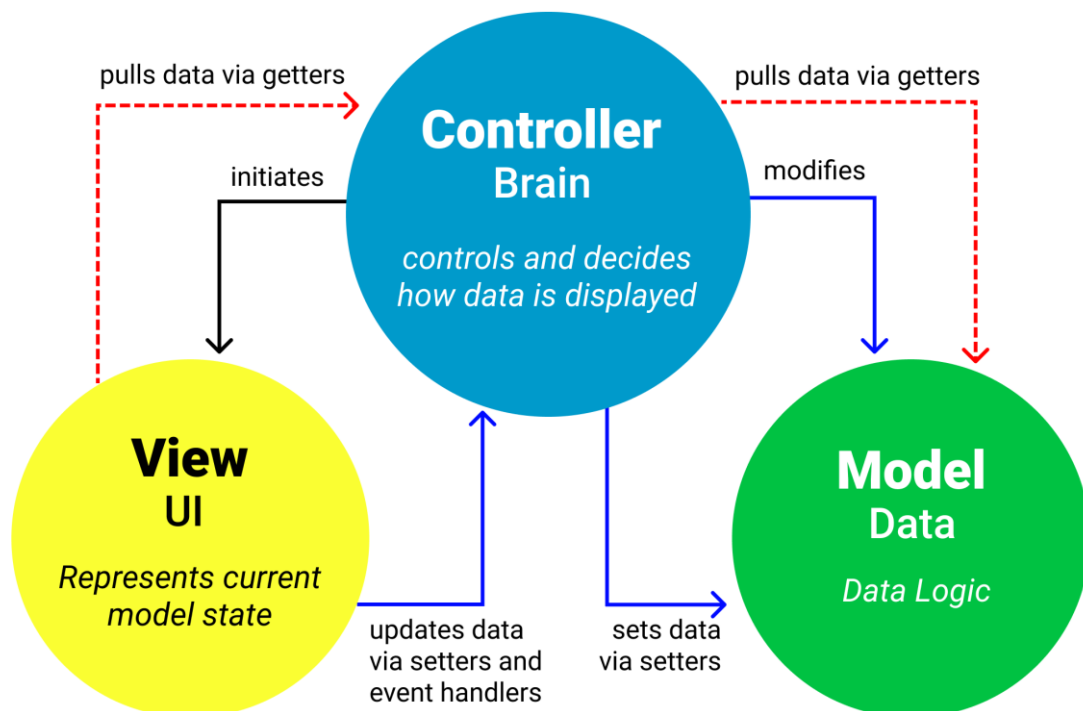


Az ASP.NET az ASP (**Active Server Pages**) technológia utódja, és jelentős fejlesztést jelentett a rugalmasság és a teljesítmény tekintetében, kiterjeszti a .NET-et, így használható a csomagok és könyvtárak nagy ökoszisztémája, amely minden .NET-fejlesztő számára elérhető. Saját könyvtárak is létrehozhatóak, amelyek megoszthatók a .NET platformon írt alkalmazások között. A keretrendszer tartalmazza a .NET keretrendszer alapkönyvtárait, de a gyakori webes minták könyvtárait is.

Az egyik ilyen könyvtár a **Model View Controller** (MVC), amely lehetővé teszi az MVC tervezési minta használatát webalkalmazások és webhelyek tervezéséhez, amelyet a felhasználói felület (nézet, vagy angolul view), az adatok (modell) és az alkalmazáslogika (vezérlő, vagy angolul controller) szétválasztására használnak. A webhelyekhez használt MVC-mintát használva a kérések egy vezérlőhöz kerülnek továbbításra, amely felelős a Modellel való együttműködésért a műveletek végrehajtása és/vagy az adatok lekéréséért.

A Vezérlő kiválasztja a megjeleníteni kívánt nézetet. A Nézet a végső oldalt jeleníti meg a Modell adatai alapján. Megkönnyíti a “tisztá” modell osztályok létrehozását, és leegyszerűsíti a kapcsolódást az adatbázissal. Deklaratív módon határozza meg az érvényesítési szabályokat a C# attribútumok használatával, amelyeket a kliensen és a kiszolgálón alkalmaznak.

MVC Architecture Pattern



5. ábra (MVC_Model)

Az ASP.NET továbbra is támogatott és frissített, ezért a legújabb fejlesztési eszközökkel lehet dolgozni benne, ami az egyik oka volt, hogy ezt a keretrendszert használjuk, a másik pedig, hogy az asztali alkalmazás programozási nyelve a C# volt, és mivel a keretrendszer támogatja azt, egyszerűbb megoldás volt egy programnyelvet használni a két projektrészhez, mint megtanulni egy teljesen újat.



2.2.2 Web Api:

Az **API** (application programming interface) vagy alkalmazásprogramozási felület olyan szabályok vagy protokollok összessége, amelyek lehetővé teszik a szoftveralkalmazások számára, hogy kommunikáljanak egymással az adatok, szolgáltatások és funkciók cseréje érdekében. Az API-k leegyszerűsítik és felgyorsítják az alkalmazás- és szoftverfejlesztést azáltal, hogy lehetővé teszik a fejlesztők számára, hogy más alkalmazások adatait, szolgáltatásait integrálják, ahelyett, hogy a semmiből fejlesztenék azokat.

Az API-k az alkalmazástulajdonosok számára is egyszerű és biztonságos módot kínálnak arra, hogy alkalmazásadataikat és funkcióikat elérhetővé tegyék szervezetük részlegei számára. Az alkalmazástulajdonosok adatokat és funkciókat is megoszthatnak vagy forgalmazhatnak üzleti partnerekkel vagy harmadik felekkel.

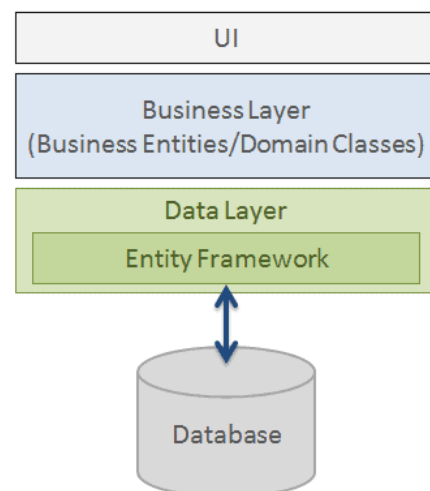
Az API-k csak a szükséges információk megosztását teszik lehetővé, más belső rendszerrészleteket rejtve, amik segítik a rendszer biztonságában tartását. A kiszolgálóknak vagy eszközöknek nem kell teljes mértékben felfedniük az adatokat – az API-k lehetővé teszik kis adatsomagok megosztását az adott kérés szempontjából.

Manapság a legtöbb API már WEB API. A webes API-k a Remote API egy fajtája (ami azt jelenti, hogy az API protokollokat használ a külső erőforrások manipulálására), amelyek az alkalmazás adatait és funkcióit teszik közzé az interneten keresztül.

2.2.3 Entity Framework:

Az Entity Framework egy nyílt forráskódú ORM-keretrendszer a Microsoft által támogatott .NET-alkalmazásokhoz. Lehetővé teszi a fejlesztők számára, hogy tartományspecifikus osztályok objektumai segítségével dolgozzanak az adatokkal anélkül, hogy a mögöttes adatbázistáblákra és oszlopokra kellene összpontosítaniuk, ahol ezeket az adatokat tárolják. Az Entity Framework segítségével a fejlesztők magasabb absztrakciós szinten dolgozhatnak, amikor adatokkal foglalkoznak, és adatorientált alkalmazásokat hozhatnak létre és karbantarthatnak kevesebb kóddal a hagyományos alkalmazásokhoz képest.

Az Entity Framework API (EF6 és EF Core) magában foglalja a tartomány (entitás) osztályok adatbázissémához való leképezését, a LINQ-lekérdezések SQL-be való lefordítását és végrehajtását, az entitásokon az élettartamuk során bekövetkezett változások nyomon követését és a változtatások mentését az adatbázisban.



© EntityFrameworkTutorial.net

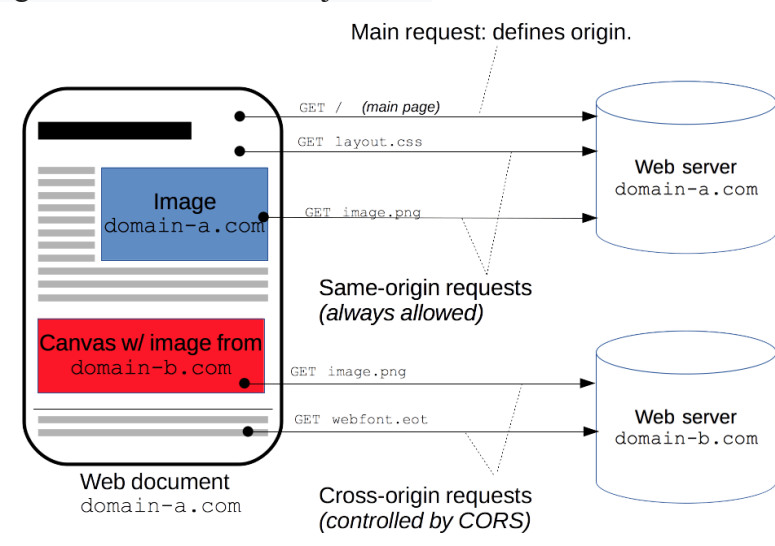
6. ábra (Entity Framework)

2.2.4 CORS Policy

Cross-Origin Resource Sharing (CORS) egy HTTP-fejlécen alapuló mechanizmus, amely lehetővé teszi a kiszolgáló számára, hogy jelezze a sajátjától eltérő eredetet (tartományt, sémát vagy portot), amelyből a böngészőnek engedélyeznie kell az erőforrások betöltését. A CORS egy olyan mechanizmusra is támaszkodik, amellyel a böngészők "elővizsgálati" kérést küldenek az eredetközi



erőforrást tároló kiszolgálónak annak ellenőrzése érdekében, hogy a kiszolgáló engedélyezi-e a tényleges kérést. Ebben az elővizsgálatban a böngésző fejléceket küld, amelyek jelzik a HTTP-metódust és a tényleges kérésben használt fejléceket.



7. ábra (CORS Policy)

Biztonsági okokból a böngészők korlátozzák a parancsfájlokból kezdeményezett eltérő eredetű HTTP-kérélmeket. A `fetch()` és az `XMLHttpRequest` például az azonos eredetre vonatkozó szabályzatot követi. Ez azt jelenti, hogy az API-kat használó webalkalmazások csak ugyanabból a forrásból kérhetnek erőforrásokat, ahonnan az alkalmazás be lett töltve, kivéve, ha a más forrásokból érkező válasz tartalmazza a megfelelő CORS-fejléceket.

A CORS mechanizmus támogatja a biztonságos eredetközi kéréseket, valamint a böngészők és szerverek közötti adatátvitelt. A böngészők CORS-t használnak olyan API-kban, mint a `fetch()` vagy az `XMLHttpRequest` az eredetközi HTTP-kérések kockázatának csökkentésére.

Az eredetközi erőforrás-megosztási szabvány új HTTP-fejlécek hozzáadásával működik, amelyek lehetővé teszik a kiszolgálók számára, hogy leírják, mely eredetek olvashatják be ezeket az információkat egy webböngészőből. Ezenkívül olyan HTTP-kérési módszerek esetén, amelyek mellékhatásokat okozhatnak a kiszolgáló adataira (különösen a GET-től vagy bizonyos MIME-típusoknál a POST-tól eltérő HTTP-metódusok), a specifikáció előírja, hogy a böngészőknek "ellenőrizniük" kell a kérést, támogatott metódusokat kell kérniük a kiszolgálótól a HTTP OPTIONS kérés módszerrel, majd a kiszolgáló "jóváhagyása" után el kell küldeniük a tényleges kérést. A kiszolgálók arról is tájékoztathatják az ügyfeleket, hogy a "hitelesítő adatokat" (például a cookie-kat és a HTTP-hitelesítést) el kell-e küldeni kérésekkel.

A CORS-hibák hibákat eredményeznek, de biztonsági okokból a JavaScript nem fér hozzá a hibával kapcsolatos információkhoz. A kód csak annyit tud, hogy hiba történt. Az egyetlen módja annak, hogy meghatározzuk, mi történt konkrétan, ha megnézi a böngésző konzolját a részletekért. A következő szakaszok ismertetik a forgatókönyveket, valamint részletezik a használt HTTP-fejléceket.

2.2.5A C#

A C# nyelv a legnépszerűbb nyelv a .NET platformon, egy ingyenes, többplatformos, nyílt forráskódú fejlesztői környezetben. A C#-programok számos különböző eszközön futtathatók, az eszközök internetes hálózata (IoT) eszközöktől a felhőig és bárhol a kettő között. Írhat alkalmazásokat telefonra, asztali számítógépre és laptopra, valamint kiszolgálókra. A C# egy platformfüggetlen



általános célú nyelv, amely a fejlesztőket produktívvá teszi, miközben nagy teljesítményű kódot ír. Több millió fejlesztővel a C# a legnépszerűbb .NET nyelv.

A C# széles körű támogatást nyújt az ökoszisztémában és az összes .NET számítási feladatban. Az objektumorientált elvek alapján más paradigmák számos jellemzőjét magában foglalja, nem utolsósorban a funkcionális programozást. Az alacsony szintű funkciók támogatják a nagy hatékonyságú forgatókönyveket nem biztonságos kód írása nélkül. A legtöbb .NET futtatókörnyezet és kódtár C# nyelven íródott, és a C# fejlesztései gyakran minden .NET fejlesztő számára előnyösek.

A C#-alkalmazások kihasználják a .NET Runtime automatikus memóriakezelését. A C#-alkalmazások a .NET SDK által biztosított kiterjedt futásidejű kódtárakat is használják. Egyes összetevők platformfüggetlenek, például a fájlrendszerkönyvtárak, az adatgyűjtések és a matematikai könyvtárak. Mások egyetlen számítási feladatra vonatkoznak, például a ASP.NET Core webkódtárak vagy a .NET MAUI UI kódtár. A NuGet gazdag nyílt forráskódú ökoszisztémája kiegészíti a futtatókörnyezet részét képező kódtárakat. Ezek a könyvtárak még több használható összetevőt biztosítanak.

A C# egy erősen típusos nyelv. Minden deklarált változónak van egy, a fordításkor ismert típusa. A fordító vagy a szerkesztőeszközök jelzik, ha helytelenül használja az adott típust. Ezeket a hibákat a program futtatása előtt kijavíthatja. Az alapvető adattípusok be vannak építve a nyelvbe és a futásidőbe: olyan értéktípusok, mint az **int**, **double**, **char**, referenciatípusok, például string, tömbök és egyéb gyűjtemények. A programok írása közben saját típusokat hoz létre.

Ezek a típusok lehetnek értékek strukturáltsági típusai vagy objektumorientált viselkedést definiáló osztálytípusok. A rekordmódosítót hozzáadhatja a struct vagy az osztály típusokhoz, így a fordító szintetizálja a kódot az egyenlőségi összehasonlításokhoz. Létrehozhat felületdefiniciókat is, amelyek meghatározzák azt a szerződést vagy tagok készletét, amelyet az interfészt megvalósító típusnak biztosítania kell. Általános típusokat és módszereket is definiálhat. Az általános szabályok típusparaméterekkel biztosítanak helyőrzőt egy tényleges típushoz, amikor használják. A kód írásakor függvényeket, más néven metódusokat definiálhat a struct és az osztály típusok tagjaiként.

Ezek a metódusok határozzák meg a típusok viselkedését. A módszerek túlterhelhetők, különböző számú vagy típusú paraméterekkel. A metódusok igény szerint értéket is visszaadhatnak. A metódusok mellett a C#-típusok tulajdonságokkal is rendelkezhetnek, amelyek accessoroknak nevezett függvények által támogatott adatelemek.

A C#-típusok eseményeket határozhatnak meg, amelyek lehetővé teszik, hogy egy típus értesítse az előfizetőket a fontos műveletekről. A C# támogatja az objektumorientált technikákat, például az öröklődést és a polimorfizmust az osztálytípusokhoz.

2.2.6 JWT Token

A JSON webes jogkivonat (JWT) JSON-objektum, amely az információk biztonságos átvitelére szolgál a weben keresztül (két fél között). Használható hitelesítési rendszerhez, és információcserére is használható. A token főleg fejlécből, hasznos teherből és aláírásból áll. Ezt a három részt pontok (.) választják el egymástól.

A JWT meghatározza az egyik féltől a másiknak küldött információk szerkezetét, és két formában jelenik meg - szerializált, deszerializált. A szerializált megközelítést elsősorban arra használják, hogy



az adatokat a hálózaton keresztül továbbítsák minden egyes kéréssel és válasszal. Míg a deszerializált megközelítés az adatok webes jogkivonatba való olvasására és írására szolgál.

Deserialized : A JWT deszerializált formában csak a fejléct és a hasznos terhet tartalmazza. Mindkettő egyszerű JSON-objektum.

Header : A JWT-ben lévő fejléct leginkább a JWT-re alkalmazott titkosítási műveletek, például a rajta használt aláírási/visszafejtési technika leírására használják. Tartalmazhatja az általunk küldött információk média/tartalom típusára vonatkozó adatokat is. Ez az információ JSON-objektumként van jelen, majd ez a JSON-objektum **BASE64URL** kódolva lesz. A fejléctben található titkosítási műveletek határozzák meg, hogy a JWT aláírt/aláíratlan vagy titkosított-e, és így milyen algoritmustechnikákat kell használni.

Payload : A payload a JWT azon része, ahol az összes felhasználói adat ténylegesen hozzá van adva. Ezeket az adatokat a JWT "állításainak" is nevezik. Ezeket az információkat bárki elolvashatja, ezért mindig tanácsos bizalmas információkat ide tenni. Ez a rész általában felhasználói adatokat tartalmaz. Ez az információ JSON-objektumként van jelen, majd ez a JSON-objektum **BASE64URL** kódolva lesz. Annyi jogcímet helyezhetünk el egy hasznos teherben, amennyit csak akarunk, bár a fejlécttel ellentétben a hasznos adatokban nem kötelező jogcím.

Signature : Ez a JWT harmadik része, és a jogkivonat hitelességének ellenőrzésére szolgál. **BASE64URL** kódolt fejléct és hasznos adat össze van kapcsolva a dot(.) függvénnyel, majd kivonatolva van a fejléctben definiált kivonatoló algoritmussal egy titkos kulccsal. Ezt az aláírást ezután hozzáfűzi a fejlécthez és a hasznos adatokhoz a dot(.) használatával.

2.2.7 CRUD (Create, Read, Update, Delete)

Az API-k létrehozásakor azt szeretnénk, hogy modelljeink négy alapvető funkciótypust biztosítsanak. A modellnek képesnek kell lennie erőforrások létrehozására, olvasására, frissítésére és törlésére. Az informatikusok gyakran hivatkoznak ezekre a funkciókra a CRUD rövidítéssel. Ahhoz, hogy egy modell teljes legyen, képesnek kell lennie legfeljebb e négy funkció végrehajtására. Ha egy művelet nem írható le e négy művelet egyikével, akkor potenciálisan saját modellnek kell lennie.

A CRUD paradigma gyakori a webalkalmazások létrehozásában, mert emlékeztető keretet biztosít a fejlesztők emlékeztetésére a teljes, használható modellek felépítésére. Képzeljünk el például egy rendszert a könyvtári könyvek nyomon követésére. Ebben a hipotetikus könyvtári adatbázisban el tudjuk képzelni, hogy lenne egy könyvforrás, amely könyvobjektumokat tárolna.

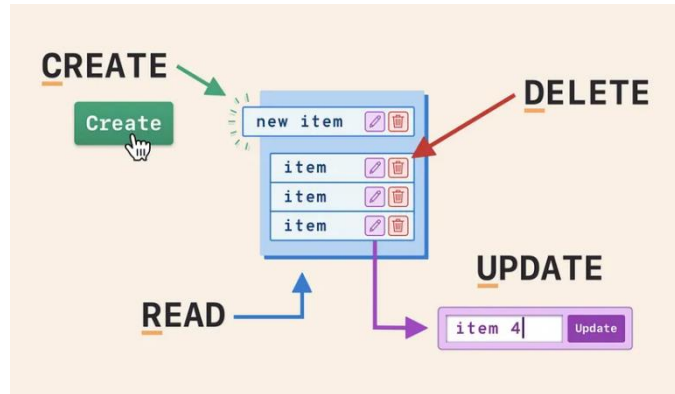
Létrehozás (Create) — Ez egy olyan függvényből állna, amelyet akkor hívnánk meg, amikor egy új könyvtári könyvet adunk hozzá a katalógushoz. A függvényt hívó program megadja a "title", "author" és "isbn" értékeket. A függvény meghívása után egy új bejegyzésnek kell lennie a könyvek erőforrásában, amely megfelel ennek az új könyvnek. Emellett az új bejegyzéshez egyedi azonosító van rendelve, amellyel később hozzáférhet ehhez az erőforráshoz.

Olvasás (Read) — Ez egy olyan függvényből állna, amely a katalógusban jelenleg szereplő összes könyv megtekintésére szolgál. Ez a függvényhívás nem változtatja meg a katalógusban lévő könyveket - egyszerűen lekéri az erőforrást, és megjeleníti az eredményeket. Funkciónk lenne egyetlen könyv visszakeresésére is, amelyhez megadhatnánk a címet, a szerzőt vagy az ISBN-t. Ismétlem, ezt a könyvet nem módosítanák, csak visszakeresnék.

Frissítés (Update) — Lennie kell egy függvénynek, amelyet meg kell hívni, ha egy könyvvel kapcsolatos információt meg kell változtatni. A függvényt hívó program megadja a "title", "author" és "isbn" új értékeit. A függvényhívás után a könyvek erőforrás megfelelő bejegyzése tartalmazza a megadott új mezőket.



Törlés (Delete) — Lennie kell egy függvénynek, amellyel el lehet távolítani egy könyvtári könyvet a katalógusból. A függvényt hívó program egy vagy több értéket ("title", "author" és/vagy "isbn") ad meg a könyv azonosításához, majd ez a könyv törlődik a könyvek erőforrásból. A függvény meghívása után a books erőforrásnak tartalmaznia kell az összes korábbi könyvet, kivéve az imént töröltet.



8. ábra (CRUD modell)

2.2.8 Az ORM

Az ORM vagy az **Object Relational Mapper** egy olyan szoftver, amelyet az adatbázisok által használt adatrepresentációk és az objektumorientált programozásban használt adatábrázolások közötti fordításra terveztek. Alapvetően az adatokkal való munka e két módja természetesen nem illeszkedik egymáshoz, ezért az ORM megpróbálja áthidalni a két rendszer adattervei közötti szakadékot.

A fejlesztő szempontjából az ORM lehetővé teszi, hogy adatbázis-alapú adatokkal dolgozzon ugyanazokkal az objektumorientált struktúrákkal és mechanizmusokkal, amelyeket bármilyen típusú belső adathoz használna. Az ORM-ek ígérete az, hogy nem kell speciális technikákra támaszkodnia, vagy feltétlenül meg kell tanulnia egy új lekérdezési nyelvet, például az SQL-t ahhoz, hogy produktív legyen az adataival.

Általában az ORM-ek absztrakciós rétegeként szolgálnak az alkalmazás és az adatbázis között. Megpróbálják növelni a fejlesztői termelékenységet azáltal, hogy kiküszöbölik a sablonkód szükségességét, és elkerülik a káros technikák használatát, amelyek megtörhetik a választott nyelvtől elvárt idiómákat és ergonómiát.

Bár az ORM-ek hasznosak lehetnek, fontos, hogy eszközként tekintsünk rájuk. Nem minden esetben lesznek hasznosak, és előfordulhat, hogy kompromisszumokat kell figyelembe vennie. Általánosságban elmondható, hogy az ORM jó választás lehet, ha a nyelv számos objektumorientált funkcióját használja sok állapot kezelésére.

Az összetett öröklési kapcsolatokkal rendelkező objektumokba ágyazott állapot kezelésének következményeit például nehéz lehet manuálisan figyelembe venni. Emellett segíthetnek a projekt könnyebb elindításában, és olyan funkciókkal kezelhetik az adatstruktúra változásait, mint a sémaáttelepítés.

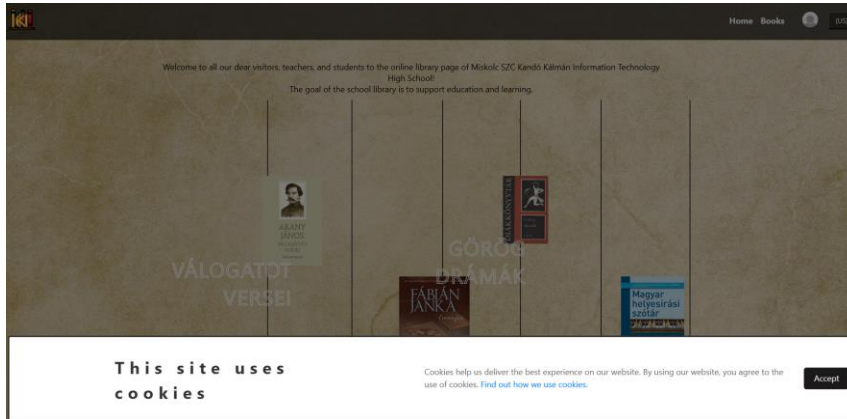
Bár az ORM-ek gyakran hasznosak, nem tökéletesek. Néha az ORM által bevezetett absztrakciós szint megnehezítheti a hibakeresést. Előfordulhat, hogy az ORM által az adatbázis és az alkalmazás közötti fordításhoz használt ábrázolás nem teljesen pontos, vagy kiszivárogtathatja a belső megvalósítás részleteit. Ezek bizonyos esetekben problémákat okozhatnak.

Fontos megérteni, hogy mik a projekt követelményei, és hogyan szeretné elkölteni az erőforrásokat a szoftver létrehozásakor. Az ORM egy olyan eszköz, amely segíthet az adatbázis-alapú alkalmazások könnyebb felépítésében, de magának kell eldöntenie, hogy hozzáadott értéket ad-e a projekthez.

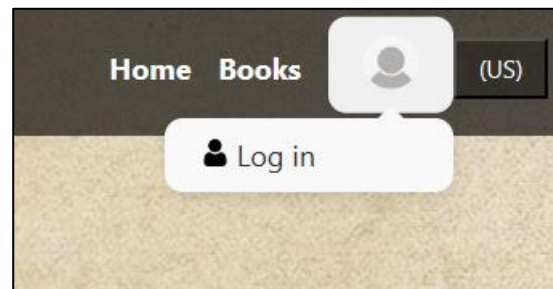


2.3 Frontend:

Mi frontendünk elkészítéséhez **React**, **Node.js** és **Visual Studio Code**-ot használtunk. Weblapunkat próbáltuk minél jobban felhasználóbarátabbá tenni. Megnyitásakor egy Sütí elfogadására készítő Modal ugrik fel, amelyekkel a felhasználó hozzájárul oldalunk sütiéhez, majd jobb felül a felhasználó a bejelentkezetlen ikonra nyomva be tud lépni kkszi-s email címmel. Például: ”gyakorlas@kkszki.hu” (Lásd 11. kép)

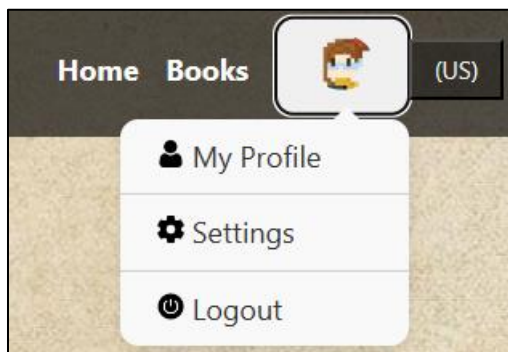


10. ábra (Frontend kezdőlap)



11. ábra (Bejelentkezési fül)

Amiután sikeresen bejelentkezett a felhasználó, azután ez a menüpont megváltozik és már 3 menüt láthatunk rajta. “My Profil”, “Settings”, “Logout”.



12. ábra (Bejelentkezett menüpontok)

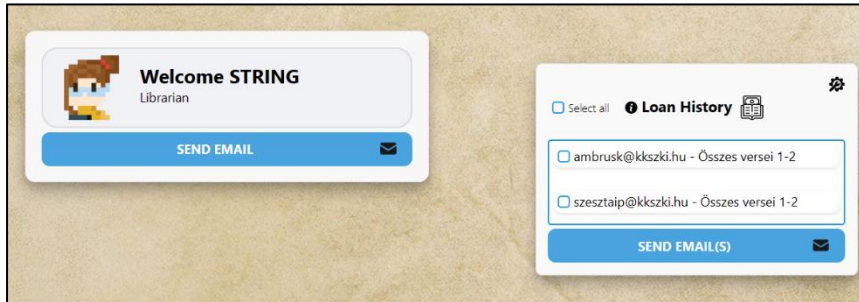
Kétféleképpen különböztetjük meg felhasználóinkat. “Admin” és “User”.

Mindkét felhasználó más dolgokat lát a weblapon. Más dolgokhoz van hozzáférése.

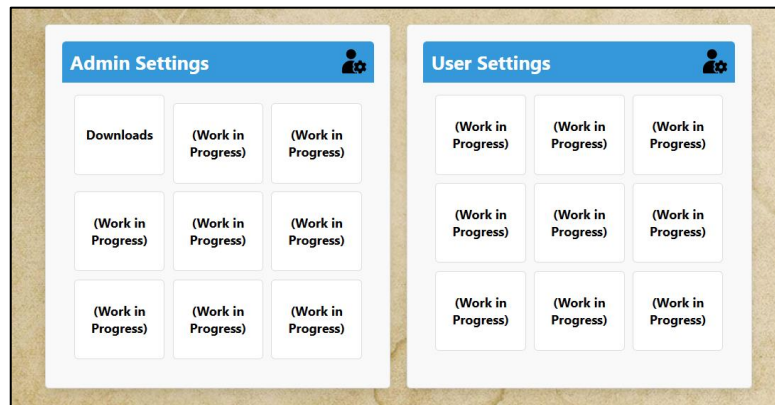


Admin felhasználó:

- Minden olyan dolgot tud, amit a *User* felhasználó, kivéve ő nem tud könyvet kölcsönözni.
- Tud email-t küldeni olyan felhasználóknak, akik kölcsönöztek könyvet.
- Tud email-t küldeni egyesével bármilyen felhasználónak.
- 2-vel több profilkép közül tud választani, mint az alap felhasználó.
- A “Settings” menüpontban látszódik neki egy másik settings komponens, ahol letudja tölteni a “WPF”-et.



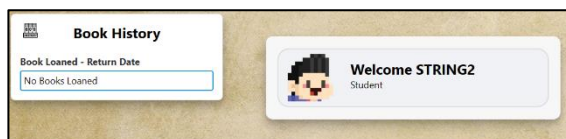
13. ábra (Admin profil)



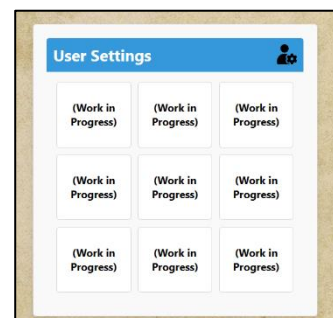
14. ábra (Admin settings)

User felhasználó:

- Látja az adatbázisban eltárolt könyveket a Navbar “Books” menüpontjában.
- A könyvek menüpontban tudja használni a könyv kereső komponenset.
- A “My Profil”-ban látja a saját könyv kölcsönzéseit.
- Csak 2 profilkép közül tud választani.
- A “Settings”-ben pedig látja a “User Settings” komponenset.



16. ábra (User profil)



15. ábra (User settings)

A Book menüpontban lehet megnézni az összes olyan könyvet, ami el van tárolva az adatbázisban.



17. ábra (Books menüpont kinézete)

Amint legörgetünk ennek az oldalnak az aljára, akkor észrevehetjük, hogy a lap alján látható a következő oldalra lépés és a jobb oldalt egy felfele mutató nyíl jelenik meg.



18. ábra (Books weboldal alsó része)

A bal oldali keretezésnél láthatóak a: “First”, “Prev”, “Next”, “Last”.

Ezek használata:

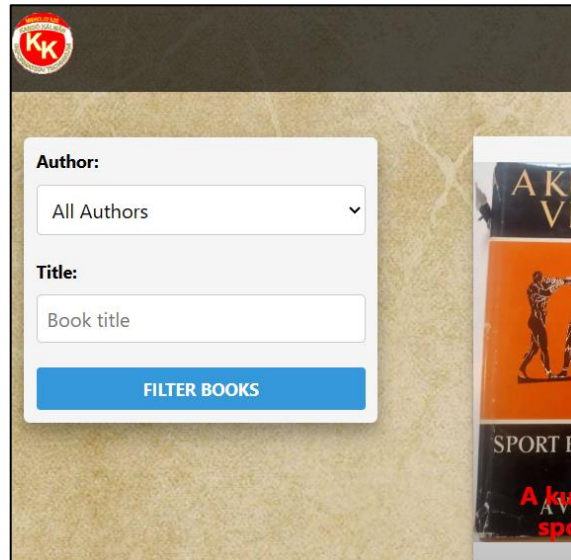
First: A legelső oldalra vissza.

Prev: Az előző oldalra vissza visz.

Next: A következő oldalra visz.

Last: Az utolsó oldalra visz.

A jobb oldalon látható piros keretben lévő gomb funkciója, hogyha a felhasználó nem szeretne felgörgetni, akkor arra nyomva felvisz minket az oldal tetejére.



19. ábra (Könyv kereső ablak)

Utoljára a könyveknel, pedig lehet szűrni a könyv írójára és a könyv nevére. A weblap bal fenti oldalán.

A fenti egy legördülő menüpont, ahol a könyv szerzőjére tudunk szűrni, alatta pedig a könyv nevére való szűrés lehetőség látható.

2.3.1 React:

A **React** egy olyan keretrendszer, amely a Webpack segítségével automatikusan lefordítja a React, a JSX és az ES6 kódot a CSS-fájl előtagjainak kezelése közben. A React egy JavaScript-alapú **UI fejlesztői könyvtár**. Bár a React inkább könyvtár, mint nyelv, széles körben használják a webfejlesztésben. A könyvtár először 2013 májusában jelent meg, és mára az egyik leggyakrabban használt frontend könyvtár a webfejlesztéshez. A React különféle bővítményeket kínál a teljes alkalmazás-architektúra támogatásához, mint például a **Flux** és a **React Native**, a pusztán felhasználói felületen túl.

A React népszerűsége manapság felülmúlta az összes többi frontend fejlesztési keretet. Íme, miért:

- **Dinamikus alkalmazások egyszerű létrehozása:** A React megkönnyíti a dinamikus webalkalmazások létrehozását, mivel kevesebb kódolást igényel, és több funkcionalitást kínál, szemben a JavaScripttel, ahol a kódolás gyakran nagyon gyorsan bonyolulttá válik.
- **Jobb teljesítmény:** A React Virtuális DOM-ot használ, ezáltal gyorsabban hoz létre webalkalmazásokat. A Virtual DOM összehasonlítja az összetevők korábbi állapotait, és csak a Real DOM megváltoztatott elemeit frissíti, ahelyett, hogy újra frissítené az összes összetevőt, ahogyan azt a hagyományos webalkalmazások teszik.
- **Újra felhasználható összetevők:** Az összetevők bármely React alkalmazás építőkövei, és egyetlen alkalmazás általában több összetevőből áll. Ezeknek az összetevőknek megvan a saját logikájuk és vezérlőelemeik, és az alkalmazás során újra felhasználhatók, ami viszont drámaian csökkenti az alkalmazás fejlesztési idejét.



- **Egyirányú adatfolyam:** A React egy egyirányú adatfolyamot követ. Ez azt jelenti, hogy a React-alkalmazások tervezésekor a fejlesztők gyakran a szülőkomponensekbe ágyazzák be az alárendelt összetevőket. Mivel az adatok egyetlen irányban áramlanak, könnyebbé válik a hibák hibakeresése.
- **Kis tanulási görbe:** A React könnyen megtanulható, mivel többnyire az alapvető HTML és JavaScript fogalmakat egyesíti néhány előnyös kiegészítéssel. Ennek ellenére, ahogy az más eszközök és keretrendszerek esetében is, el kell szánnia egy kis időt a React könyvtárának megfelelő megértésére.
- **Használható webes és mobilalkalmazások fejlesztésére is:** Azt már tudjuk, hogy a Reactot webes alkalmazások fejlesztésére használják, de ez nem minden. Létezik egy React Native nevű keretrendszer, amely magából a React-ból származik, és amely rendkívül népszerű mobilalkalmazások létrehozására használják. Tehát a valóságban a React webes és mobilalkalmazások készítésére is használható.
- **Dedikált eszközök az egyszerű hibakereséshez:** a Facebook kiadott egy Chrome-bővítményt, amellyel a React alkalmazások hibakeresésére használhatók. Ez gyorsabbá és egyszerűbbé teszi a React webalkalmazások hibakeresési folyamatát.

2.3.1.1 Előnyei:

- A React.js testreszabott virtuális DOM-ot épít fel. Mivel a JavaScript virtuális DOM gyorsabb, mint a hagyományos DOM, ez javítja az alkalmazások teljesítményét.
- A ReactJS csodálatos felhasználói felületet tesz lehetővé.
- A React különféle architektúrákat integrál.
- A React egyszerűbbé teszi a teljes szkriptelési környezet folyamatát.
- Megkönnyíti a fejlett karbantartást és növeli a teljesítményt.
- Garantálja a gyorsabb megjelenítést
- A mobilalkalmazások fejlesztésére szolgáló szkript elérhetősége a React legjobb tulajdonsága.
- A ReactJS-t egy nagy közösség támogatja.

2.3.1.2 Hátrányai/Hátrányosságai:

- Csak az alkalmazás fő részével foglalkozik, ennek eredményeként további script-eket kell kiválasztani, ha a fejlesztői eszközök teljes gyűjteményére vágyunk.
- Beépített szkriptet és JSX-et alkalmaz, amit egyes programozók kényelmetlennek találhatnak.
- Számos eszköz és könyvtár létezik, például a Redux és a Reflux, amelyek növelhetik a React teljesítményét. Még magát a Reactot is rendszeresen frissítik. Sajnos ennek van egy árnyoldala is. Egyes fejlesztők úgy gondolják, hogy a React technológiák olyan gyorsan frissülnek és gyorsan fejlődnek, hogy nincs idő dokumentálni vagy megfelelő utasításokat írni. A fejlesztőknek csak ritka szöveges útmutatók maradnak, amelyek nem fedik le a részleteket.
- Míg egyesek azzal érvelnek, hogy a Reactot folyamatosan fejlesztik, és megkönnyítik a munkájukat, mégis előfordulhat, hogy ezt negatívumként fogják fel. Ennek az az oka, hogy minden folyamatosan fejlődik, ami azt eredményezi, hogy a fejlesztők nem elégedettek a folyamatok vagy az új mechanikák folyamatos újratanulásának



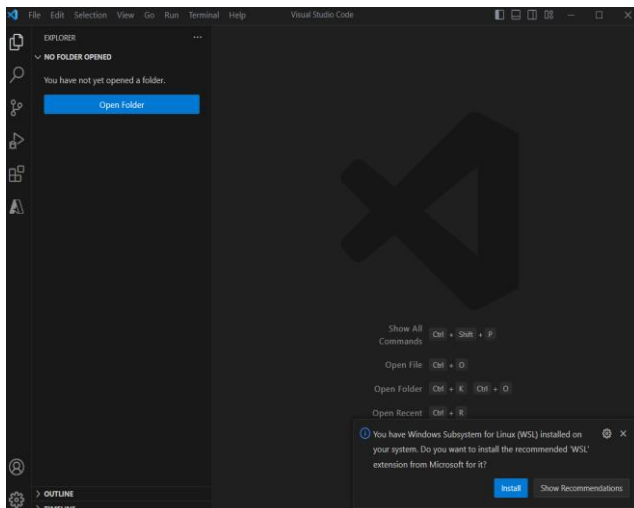
szükségességével, és előfordulhat, hogy egyes fejlesztők nem érzik jól magukat, ha lépést akarnak tartani ezzel a tempóval.

- A JSX-et a legtöbb ember előnynek tekintené, nem pedig hátránynak, és mégis, botránynak is tekinthető. Ez zavart okozhat azoknak, akik nem nagyon ismerik ezt. A JSX egyszerű szavakkal egy JavaScript-bővítmény, amely olvashatóbbá és tisztábbá teheti a kódot. A HTML és a JavaScript keveréke a JSX-ben sokkal bonyolultabbá teszi a React JS elsajátítását, és ez nem lehet vonzó a törekvő fejlesztők számára. A fejlesztők és a tervezők panaszkodnak a JSX tanulásának bonyodalmaira és az ebből következő nehéz tanulási görbére.

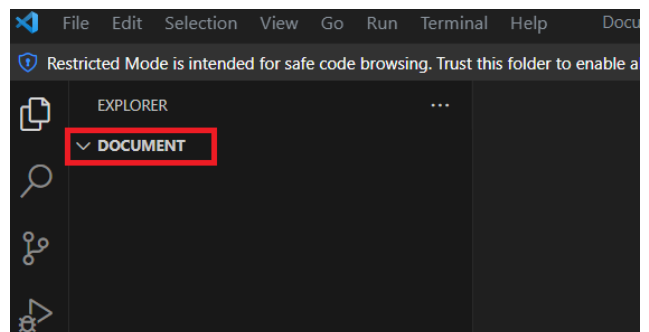
2.3.1.3 React felület telepítése:

Első lépésként telepítenünk kell a [Visual Studio Code](#)-ot.

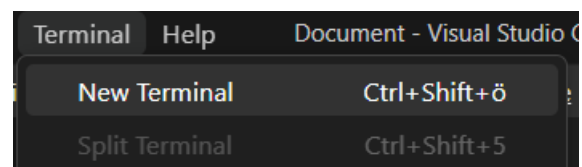
Ezután hozzunk létre egy mappát, ahol szeretnénk tárolni az applikációnkat, majd indítsuk el a Visual Studio Code-ot és keressük ki azt a mappát, amit létrehoztunk neki. (Lásd: 22. kép)



22. ábra (Visual Studio Code mappa keresése)



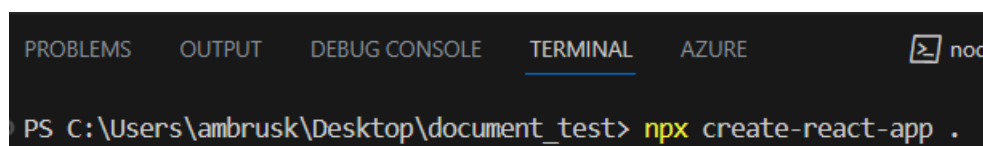
21. ábra (Visual Studio Code megnyitott mappa)



20. ábra (Visual Studio Code Terminal)

Aztán, ha megnyitottuk, akkor ezt láthatjuk (Lásd 21. kép). Amint ez megtörtént nyomjunk a *“Terminal”* fülre, majd a *“New Terminal”*-ra (20. kép).

Azután a terminálba, amit megnyitottuk írjuk be *“npm create-react-app .”*



23. ábra (React telepítése)

A terminál, ha kiírja, hogy *“Happy Hacking”*, akkor tudhatjuk, hogy sikeresen letöltöttük a React-ot.

Ezután a terminálba írjuk be, hogy *“npm start”* és az alap weboldalunk el is indult.



2.3.2 Node.js:

A Node.js egy nyílt forráskódú, többplatformos futási környezet és könyvtár, amelyet webalkalmazások futtatására használnak a felhasználó böngészőjén kívül.

Szerveroldali programozásra és elsősorban nem blokkoló, esemény vezérelt kiszolgálókhoz, például hagyományos webhelyekhez és háttér API-szolgáltatásokhoz használják, de eredetileg a valós idejű, push-alapú architektúrákat szem előtt tartva tervezték. Minden böngészőnek megvan a saját JS-motor verziója, a node.js pedig a Google Chrome V8 JavaScript motorjára épül.

Leegyszerűsítve ez azt jelenti, hogy teljes webhelyek futtathatók egységes “**verem**” segítségével, ami gyorsrá és egyszerűvé teszi a fejlesztést és a karbantartást, lehetővé téve, hogy a projekt üzleti céljainak elérésére összpontosítson.

Az a tény, hogy a Node.js nyílt forráskódú, azt jelenti, hogy ingyenesen használható, és a fejlesztők globális közössége folyamatosan módosítja és fejleszti. Fontos megérteni a Node.js-t, hogy valójában nem keretrendszer vagy könyvtár - mint a hagyományos alkalmazásszoftvereknél -, hanem a JavaScript egy futási környezet. A futatókörnyezet (néha RTE-re rövidítve) tartalmaz olyan webes API-kat, amelyekhez a fejlesztő hozzáférhet kód létrehozásához, és egy JavaScript-motort, amely elemzi a kódot. Ez könnyűvé, rugalmassá és könnyen telepíthetővé teszi, minden olyan funkció, amely segít optimalizálni és felgyorsítani az alkalmazásprojektet.

2.3.2.1 A Node.js története:

A Node.js-t 2009-ben hozta létre **Ryan Dahl**. Korábban Dahl bírálta a meglévő népszerű webszerverek és a közös kódolási módszerek által kínált korlátozott lehetőségeket.

Abban az időben a szerverek nehezen kezelték a nagy mennyiségű párhuzamos kapcsolatokat, és a kódok vagy blokkolták a teljes folyamatot, vagy azt sugallták, hogy több veremre volt szükség. Ezek mind olyan problémák voltak, amelyek hátráltatták a vállalkozásokat abban, hogy olyan sokoldalú termékeket hozzanak létre, amelyek megfelelnek a nagy mennyiségű felhasználói igényeknek.

A Node.js kezdeti kiadása csak Linux és Mac OS X operációs rendszereket támogatott. Fejlesztését és karbantartását eleinte a Dahl vezette, később pedig a Joyent, egy szoftver- és szolgáltató cég szponzorálta.

2.3.2.2 Node.js architektúra:

A Node.js mechanikája hozzájárul a fejlesztők körében való népszerűségéhez. Míg a legtöbb alternatív futási környezet többszálú feldolgozási modelleket használ, a Node.js mindezt egyetlen szálon belül végzi el.

A többszálú feldolgozási beállításokban minden kiszolgálónak korlátozott számkészlete van, amelyhez hozzáférhet. Tehát minden alkalommal, amikor egy szerver kérést kap, kihúz egy szálat a készletből, és hozzárendeli a kéréshez, hogy gondoskodjon a feldolgozásról. Ebben az esetben a feldolgozás szinkron és szekvenciális, ami azt jelenti, hogy egyszerre egy műveletet hajtanak végre.

A többszálas feldolgozás során a rendszer minden kéréskor kiválaszt egy szálat, amíg az összes korlátozott szál fel nem használják. Amikor ez megtörténik, a kiszolgálónak meg kell



várnia, amíg egy foglalt szál újra felszabadul. Ez lassú és nem hatékony alkalmazásokat eredményezhet, ami az ügyfélélménytől a potenciális ügyfelek konverziójáig mindenre átütő hatásokhoz vezethet. Ez különösen akkor jelenthet problémát, ha az alkalmazásnak nagyszámú egyidejű ügyfélkéréssel kell foglalkoznia.

A Node.js azonban egyszálú feldolgozást használ. A kettő közötti különbség az, amilyenek képzele: az egyszálú architektúrák minden kérést egyetlen főszálon dolgoznak fel, eseményhurkokat használva a blokkoló bemeneti/kimeneti műveletek nem blokkoló módon történő futtatására.

Egy egyszálú architektúra elméletileg sokkal gyorsabban és hatékonyabban tud működni és skálázható, mint a többszálú beállítások. Ryan Dahl erre gondolt, amikor először megírta a Node.js-t, és ez nagyban hozzájárult ahhoz, hogy miért olyan népszerű a webalkalmazás-fejlesztők körében.

2.3.2.3 Node.js modul-jai:

A Node.js több „modult” tartalmaz, amelyeket egyedi kontextusokban tartanak, így nem zavarják a többi modult, és nem szennyezik a node.js globális hatókörét. Ez kulcsfontosságú a nyílt forráskódú szoftvereknél.

A Node.js modulja egy egyszerű vagy összetett funkció, amely JavaScript-fájlokba van rendezve, és újrafelhasználható a Node.js alkalmazásban. A Node.js-ben háromféle modul létezik: alapmodulok, helyi modulok és harmadik féltől származó modulok.

- Az **alapmodulok** a Node.js alapvető, csupasz funkcióit tartalmazzák. Ezek automatikusan betöltődnek, amikor egy csomóponti folyamat elindul, és a Node.js bináris terjesztésének részét képezik.
- A **helyi modulok** a Node.js alkalmazásban létrehozott modulok. Különböző és további funkciókat tartalmaznak külön fájlokban és mappákban az alapvető funkciócsomaghoz. A helyi modulok csomagolhatók és terjeszthetők a szélesebb Node.js közösség számára is.
- A **harmadik féltől származó modul** egy meglévő kód, amelyet egy harmadik fél írt, és importálható a Node.js alkalmazásba különböző szolgáltatások és funkciók bővítéséhez vagy hozzáadásához.

2.3.2.4 Node csomagkezelő:

A Node csomagkezelő (vagy röviden 'npm') tesz néhány dolgot; Először is, online tárhelyként működik a nyílt forráskódú Node.js projektek közzétételéhez. Másodszor, parancssori lehetőségként használják az adott tárhellyel való interakcióhoz, segítve a csomagtelepítést, a verziókezelést és a függőségek kezelését.

Leggyakrabban programok közzétételére, megtalálására, telepítésére és fejlesztésére használják. Lényegében egy hasznos felülettel segíti a fejlesztőket a Node.js eszközök és csomagok legjobb kihasználásában.

2.3.3 Visual Studio Code:

A Visual Studio Code vagy a VS Code egy szövegszerkesztő, amely több testreszabható funkciót kínál beépülő modulok formájában a fejlesztők számára, hogy egy meglehetősen



szelektív fejlesztői környezetet tudjon kialakítani. A VS Code könnyű, és könnyen telepíthető bármilyen platformra.

Miért a **Visual Studio Code**?:

- ❖ A VS Code egy **többplatformos kódszerkesztő**, amely könnyen futtatható macOS, Windows és Linux rendszeren. A VS Codes összes webtechnológiája végül Electron Framework-et használ, ami azt jelenti, hogy a VS Code-on keresztül épített alkalmazások karcsúak és zökkenőmentesek frissítésükkor.
- ❖ A VS Code viszonylag sokkal gyorsabb, mint a Visual Studio. Az IDE-hez képest gyorsabban tud elindulni.
- ❖ A Visual Studio Code **hihetetlenül rugalmas**. Nagyjából bármire képes, amit egy fejlesztő akar. Annak ellenére, hogy kódszerkesztőnek tervezték, megismételheti, amit egy integrált fejlesztői környezet csinál ugyanazon.
- ❖ A VS Code a Visual Studióval összehasonlítva meglehetősen **egyszerű** és **zökkenőmentes** a fejlesztési tapasztalatok szempontjából. A VS Code meglehetősen áramvonalas és egyszerű, így a fejlesztők nem keverednek bele semmilyen bonyolultságba.
- ❖ A Visual Studio Code a legjobb választás webfejlesztéshez. Hihetetlen támogatást kínál ezernyi eszközzel és bővítményével.

2.3.3.1 Visual Studio vs Visual Studio Code:

A Visual Studio segítséget nyújt számítógépes programok, webhelyek, webes alkalmazások, mobilalkalmazások és webszolgáltatások fejlesztésében. A Visual Studio vagy az IDE a Microsoft szoftverfejlesztési platformját, azaz a Windows API-t, a Windows Presentation Foundation-t, a Windows Forms-t, a Microsoft Silverlightot és a Windows Store-t veszi igénybe a natív kód előállításához és kezeléséhez.

Míg a Visual Studio Code használható a kód írására, szerkesztésére és hibakeresésére, mindezt egy helyen. A VS Code számos programozási nyelvet támogat, amelyekhez a fejlesztőnek nincs szüksége webes támogatásra. Minden megtalálható a beépített többnyelvű támogatásban.

A fejlesztők mindenféle fejlesztéshez támaszkodhatnak a Visual Studio Code-ban, amennyiben azt a megfelelő eszközökkel párosítják. A VS Code beépített JavaScript, Node.js és TypeScript támogatással érkezik. Ha ez nem elég, egyszerűen hozzáadhatja a szükséges támogatást olyan nyelvekhez, mint a Python, C#, PHP, Java és még sok más a bővítmények telepítésével.

Az elsődleges ok, amiért a VS Code jobbnak tűnik a használat szempontjából a Visual Studio vs. Visual Studio Code vitájában, az az, hogy hihetetlen támogatást kínál, minden platformon működik, valamint könnyű és robusztus. Mindenre képes egy helyen.



2.3.3.2 Különbségek:

Visual Studio	Visual Studio Code
A Visual Studio egy integrált fejlesztői környezet, más néven IDE	A Visual Studio Code egy kódszerkesztő. A fejlesztő könnyen szerkesztheti a kódját.
A VS lassabb, ha a különböző platformokon való teljesítményről van szó. A feldolgozási sebesség lassabb.	A VS Code összehasonlítva gyorsabb
Visual Studio-nak van egy ingyenes szerkesztője fejlesztőknek, de van egy jobb és fizetett IDE verziója is.	VS Code teljesen ingyenes és nyílt forráskódú.
A VS a legjobb és legfejlettebb IntelliSense-t használja.	VS Code-ban nincs IntelliSense
A teljes letöltési méret elég nagy	A Visual Studiohoz képest a Visual Studio Code meglehetősen kevés helyet foglal. Nem igényel sok tárhelyet.
A VS több helyet igényel a jobb és gördülékenyebb működéshez.	A VS Code viszonylag nem igényel sok helyet a futtatáshoz. Könnyen fut 300 MB rammal.
A Visual Studio csak macOS és Windows rendszeren fut.	A VS Code futhat macOS, Windows és Linux alatt is.
Nem sok professzionális fejlesztésű bővítmény érhető el a Visual Studio számára.	A VS Code professzionálisan összeállított bővítmények és bővítmények széles skáláját tartalmazza, hogy megfeleljen mindenféle szerkesztési és fordítási igénynek.

2.4 A WPF:

Windows megjelenítési alaprendszer (WPF), egy felbontásfüggetlen felhasználói felületi keretrendszer, amely vektoralapú renderelő motort használ, és a modern grafikus hardverek előnyeinek kihasználására készült. A GUI keretrendszer lehetővé teszi, hogy olyan alkalmazást hozzon létre, amely a GUI elemek széles skáláját tartalmazza, például címkéket, szövegdozokat és más jól ismert elemeket.

GUI keretrendszer nélkül ezeket az elemeket manuálisan kellene megrajzolni, és kezelnie kellene az összes felhasználói interakciós forgatókönyvet, például a szöveg- és egérbevitelt. Ez sok munka, így helyett a legtöbb fejlesztő egy GUI keretrendszert fog használni, amely elvégzi az összes alapvető munkát, és lehetővé teszi a fejlesztők számára, hogy nagyszerű alkalmazások készítésére összpontosítsanak.

A WPF átfogó alkalmazásfejlesztési szolgáltatásokat kínál, beleértve az XAML (Extensible Application Markup Language) nyelvet, vezérlőket, adatkötést, elrendezést, 2D és 3D grafikákat, animációkat, stílusokat, sablonokat, dokumentumokat, médiát, szöveget és tipográfiát. A WPF a .NET része, így olyan alkalmazásokat hozhat létre, amelyek a .NET API más elemeit is tartalmazzák.



2.4.1 Asztali Alkalmazásunk:

Az asztali alkalmazás készítéséhez WPF-et használtunk, mert egyszerű kezelhetősége és széles tool-szetek elérhetősége miatt könnyebb volt vele dolgozni, mint más applikációkkal. Fejlesztés során törekedtünk arra, hogy a lehető legfelhasználóbarátabb felületet és szerkezetet hozzuk létre, és ezzel minél könnyebbé és egyszerűbbé tegyük a munkát a felhasználó számára.

Első lépésként a felhasználónak be kell jelentkeznie az alkalmazásba (A bejelentkezési ablak az 24.ábrán látható). A bejelentkezéshez szükséges adatok az emailcím és a felhasználó által megadott jelszó. Ha az adatok megfelelőek, és az adott fiók rendelkezik "Admin" jogosultsággal, sikeres lesz a bejelentkezés és a felhasználó igénybe tudja venni az alkalmazást. Amennyiben a bejelentkezés sikeres, de a fiók nem rendelkezik "Admin" jogosultsággal, az alkalmazás nem fogja továbbengedni a felhasználót.

24. ábra (WPF bejelentkezési fül)

25. ábra (WPF karbantartó elosztó)

Sikeres bejelentkezés és azonosítás után a felhasználó továbbításra kerül egy másik ablakra (25.ábra), ahol lehetősége van kiválasztani, melyik táblán szeretne műveleteket végezni.

A kiválasztott menüpont átirányítja a felhasználót egy másik ablakra, ahol egy Datagrid fogadja majd (példa az 26.ábrán). A datagrid egy nagy ablak, ami tartalmazza az összes elérhető adatot az adatbázisból, és egy kattintással ki lehet választani belőle egy elemet, amivel műveletet kívánunk végezni. Több elem kiválasztása esetén csak és kizárólag az első kijelölt elemet fogja felhasználni a program, biztonsági okokból. A legtöbb menüpont tartalmazza az alap CRUD művelethez szükséges mezőket és lehetőségeket, de különleges igények révén például a kölcsönzési adatokat vagy a felhasználókat nem lehet törölni, mert előfordulhat, hogy később egy kölcsönzést vissza kell nézni, különböző okokból. Illetve mindegyik ablak tartalmaz egy "Vissza" gombot, ami a főoldalra navigálja a felhasználót.

26. ábra (Felhasználó szerkesztő fül)



A felhasználó szerkesztés menüpont alatti ablak az alap CRUD műveleteken kívül tartalmaz egy jelszómegváltoztató gombot, aminek segítségével egy felhasználó jelszavát az admin megváltoztathatja, ami elfelejtett jelszó esetén lehet hasznos. A gomb megjelenít egy új ablakot (27.ábra) ami egy "Új jelszó" mezőt tartalmaz. Ha az admin kitölti a mezőt, és rákattint a "Visszaállítás" gombra, a felhasználó jelszava megváltozik. Azonban a jelszó csak akkor változik meg, ha az admin rákattint a gombra, így ha időközben kiderül, hogy mégsem kell változtatni akkor elég csupán bezárni az ablakot, és nem fog történni változás.

Új felhasználó létrehozása esetén a "Létrehozás" gombra kattintva felugrik egy új ablak (28.ábra) ahol, ha kitöltik az "Email cím" és "Jelszó" mezőket majd a "Felhasználó Regisztrálása" gombra kattintanak, egy új felhasználó kerül feltöltésre az adatbázisba. Több felhasználó egyszerre való regisztrálása esetén a "File-ből Beolvasás" gombra kattintva ki lehet választani egy szöveges dokumentumot (.txt) az admin számítógépéről, ami, ha jó struktúrával rendelkezik, a program beolvassa az adatait, és feltölti az adatbázisba.

A "Megváltoztatás" menüpontra kattintva a kiválasztott felhasználó adatait lehet megváltoztatni egy új ablakon (29.ábra), ahova alaphoz átviszi a felhasználó jelenlegi adatait, át lehet írni az Email címet a megfelelő szövegdobozban, ki lehet választani egy új dátumot év, hónap, napra szétbontva, meg lehet változtatni az adott felhasználó jogosultságát, és be lehet állítani egy másik felhasználó képet (ami egy előre feltöltött profilkép). A "Szerkesztés" gombra kattintva véglegesíthetőek a változtatások.

A könyvek adatainak megváltoztatása hasonlóképpen működik (30.ábra), és ha valamilyen hiba lép fel az adatbevitel során, a hibás adattal ellátott mező értéke egy alapértéket fog beállítani. A "Változtatás" gombbal lehet véglegesíteni a megváltoztatni kívánt értékeket.



Új jelszó :

Visszaállítás

27. ábra (Felhasználó jelszó változtatása fül)



Az email cím mezőbe csak a "kkszi.hu" előtti részt kell beírni! (Például : nagyk)

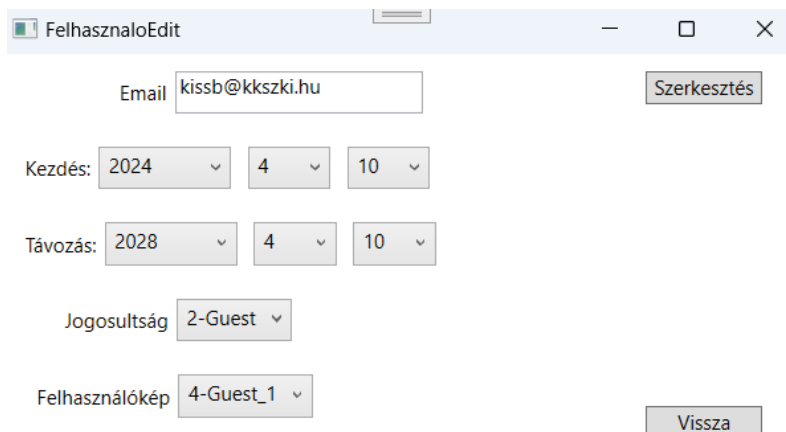
Email cím :

Jelszó :

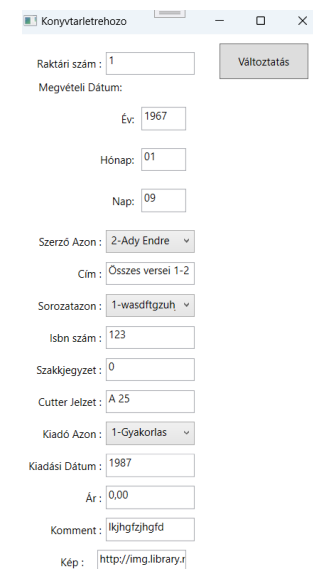
Felhasználó Regisztrálása

File-ből beolvasás

28. ábra (Új felhasználó hozzáadása fül)



29. ábra (Felhasználó szerkesztő fül)



30. ábra (Könyv szerkesztése fül)



Létrehozás esetén egy nagyon hasonló ablak nyílik meg (32.ábra), a lényeges különbség a kép feltöltésével kapcsolatos. A “Kép hozzáadása” gombbal ki lehet választani az admin gépről egy képet, aminek az elérési útvonalát a program átformálja, hogy FTP szerverről töltsse be a képet feltöltés után, majd automatikusan beilleszti a “Kép” mezőbe. Ha a “Kép feltöltése” be van jelölve, akkor a program feltölti a képet az FTP szerverre. Ezen technológia felhasználásáról [itt olvashatnak bővebben](#).

A Kölcsönzések manipulálása egy kicsivel speciálisabb a többi szerkesztőhöz képest (31.ábra). Minden rögzített kölcsönzésnek megjelenik a kezdete, a határideje, ki kölcsönözte ki és melyik könyvet, illetve, hogy az adott kölcsönzés még folyamatban van-e vagy már befejeződött, illetve ha szükséges, egyedi megjegyzéseket is hozzáadhat az admin.

Id	BookId	UserId	Date	DateEnd	Returned	Comment
1	1	2	4/18/2024 12:00:00 AM	6/16/2029 12:00:00 AM	<input checked="" type="checkbox"/>	
2	1	2	4/18/2024 12:00:00 AM	6/16/2029 12:00:00 AM	<input type="checkbox"/>	
3	2	2	4/18/2024 12:00:00 AM	6/16/2029 12:00:00 AM	<input checked="" type="checkbox"/>	:D

Buttons: Frissít, Kölcsönzés hozzáadása, Váloztat, Befejezés, Vissza

31. ábra (Köny kölcsönzés szerkesztő fül)

automatikusan be van állítva arra az időpontra, de tetszés szerint megváltoztatható, ugyan úgy, ahogyan a kölcsönzés kezdete is, aminek az alap értéke a használatának napja. Ahogyan a többi ablaknál is, a “Hozzáadás” gomb véglegesíti és tölti fel adatbázisba az adatokat.

Fields: Azon (nem kötelező), KönyvAzon: 1, Összes versei 1-2, Felhasználó: 2, string2@kkszki.hu, Kezdet: 2024, 4, 24, Határidő: 2029, 6, 16, Megjegyzés (opcionális). Button: Hozzáad

34. ábra (Kölcsönzés hozzáadás fül)

Fields: Raktári szám, Megvétel dátum: 2024, 4, 24, Szerző Azon: 1-No informa, Cím, Sorozatazon: 1-wasdfgtzuh, Isbn szám, Szakkjegyzet, Cutter Jelzet, Kiadó Azon: 1-Gyakorlas, Kiadási Dátum, Ár, Komment, Kép, Kép feltöltése. Button: Hozzáadás

32. ábra (Új könyv hozzáadása)

Új kölcsönzés hozzáadása esetén az adminnak ki kell választania, melyik felhasználó, melyik könyvet, mikortól meddig szeretné kikölcsönözni.

Mivel kérés volt, hogy egy bizonyos időpont legyen a határidő, ezért az

Fields: Azon (nem kötelező), KönyvAzon: 1, Összes versei 1-2, Felhasználó: 2, string2@kkszki.hu, Kezdet: 2024, 4, 18, Határidő: 2029, 6, 16, Megjegyzés (opcionális). Button: Megváltoztat

33. ábra (Adott kölcsönzés szerkesztő fül)

Véglegesítés után pedig a kiválasztott könyv egy kikölcsönzött állapotba kerül, és addig nem lehet ismételt kikölcsönözni, amíg az adott könyv nincs visszahozva, amit a kölcsönzés főoldalon található “Befejezés” gombbal lehet jelezni az adatbázisban. Ilyenkor a kölcsönzés rubrika kipipálódik, és a könyv újra kikölcsönözhetővé válik, azonban, ha a kölcsönzés befejeződött, azt már nem lehet újra megnyitni, még a megváltoztató fülnél sem (ami az 31.ábrán látható)!



3 Tárhelyek:

3.1 FTP (File Transfer Protocol):

Az FTP (**File Transfer Protocol**) egy hálózati protokoll fájlok számítógépek közötti átvitelére **Transmission Control Protocol/Internet Protocol** (TCP/IP) kapcsolatokon keresztül. A TCP/IP csomagon belül az FTP alkalmazási réteg protokollnak számít.

Az FTP-tranzakciók során a végfelhasználó számítógépét általában helyi gazdagépnek nevezik. Az FTP-ben részt vevő második számítógép egy távoli gazdagép, amely általában egy szerver. Mindkét számítógépet hálózaton keresztül kell csatlakoztatni, és megfelelően be kell állítani a fájlok FTP-n keresztüli átviteléhez. A kiszolgálókat be kell állítani az FTP-szolgáltatások futtatására, és az ügyfélnek telepített FTP-szoftverrel kell rendelkeznie a szolgáltatások eléréséhez.

Bár sok fájlátvitel végrehajtható a Hypertext Transfer Protocol (HTTP) használatával – a TCP/IP csomag egy másik protokolljával –, az FTP-t továbbra is gyakran használják fájlok átvitelére a színtfalak mögött más alkalmazásokhoz, például banki szolgáltatásokhoz. Időnként új alkalmazások webböngészőn keresztüli letöltésére is használják.

3.1.1 FTP működése:

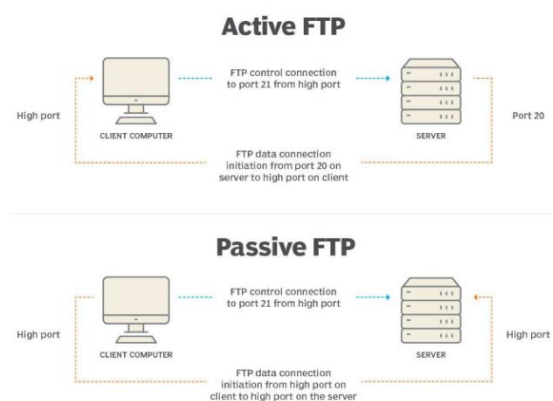
Az FTP egy kliens-szerver protokoll, amely két kommunikációs csatornára támaszkodik az ügyfél és a szerver között: egy parancscsatorna a beszélgetés vezérlésére és egy adatcsatorna a fájl tartalom továbbítására.

Így működik egy tipikus FTP átvitel:

- 1) A felhasználónak általában be kell jelentkeznie az FTP-kiszolgálóra, bár egyes kiszolgálók a tartalom egy részét vagy egészét bejelentkezés nélkül is elérhetővé teszik, ez a modell névtelen FTP-ként ismert.
- 2) A kliens beszélgetést kezdeményez a szerverrel, amikor a felhasználó fájl letöltését kéri.
- 3) Az FTP használatával az ügyfelek feltölthetnek, letölthetnek, törölhetnek, átnevezhetnek, áthelyezhetnek és másolhatnak fájlokat a szerveren.

Az FTP-munkamenetek aktív vagy passzív módban működnek:

- **Aktív mód:** Miután egy kliens munkamenetet kezdeményez egy parancscsatorna-kéréssel, a szerver adatkapcsolatot hoz létre vissza az ügyféllel, és megkezdik az adatok átvitelét.
- **Passzív mód:** A szerver a parancscsatorna segítségével küldi el a kliensnek az adatcsatorna megnyitásához szükséges információkat. Mivel a passzív módban az ügyfél kezdeményez minden kapcsolatot, jól működik a tűzfalakon és a hálózati címfordító átjárókon keresztül.



35. ábra (FTP aktív_passzív mód)

A felhasználók egy egyszerű parancssori felületen keresztül dolgozhatnak az FTP-vel



Microsoft Windows, Apple macOS vagy Linux rendszeren konzolból vagy terminálablakból vagy egy dedikált grafikus felhasználói felülettel. A webböngészők FTP-kliensként is szolgálhatnak.

3.1.2 FTP fontossága és használhatósága:

Az FTP egy szabványos hálózati protokoll, amely kiterjedt fájlátviteli lehetőségeket tesz lehetővé az IP-hálózatokon keresztül. FTP nélkül a fájl- és adatátvitel más mechanizmusokkal is kezelhető, például emailekkel vagy HTTP webszolgáltatással, de ezekből a többi opcióból hiányzik az FTP által biztosított fókusz, pontosság és vezérlés.

Az FTP-t az egyik rendszer és a másik rendszer közötti fájlátvitelre használják, és számos gyakori felhasználási esete van, beleértve a következőket:

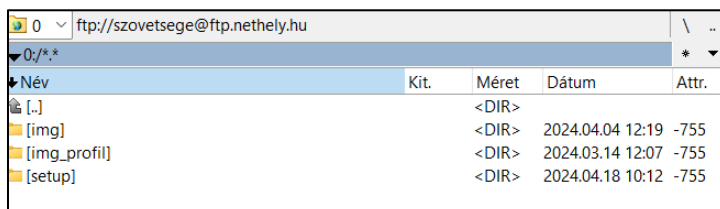
- **Biztonsági mentés:** Az FTP-t biztonsági mentési szolgáltatások vagy egyéni felhasználók használhatják az adatok biztonsági mentésére egy helyről egy FTP-szolgáltatásokat futtató biztonságos biztonsági mentési kiszolgálóra.
- **Replikáció:** A biztonsági mentéshez hasonlóan a replikáció magában foglalja az adatok megkettőzését egyik rendszerről a másikra, de átfogóbb megközelítést alkalmaz a magasabb rendelkezésre állás és rugalmasság biztosítása érdekében. Ennek elősegítésére az FTP is használható.
- **Hozzáférés és adatbetöltés** Az FTP-t gyakran használják a megosztott webtárhely- és felhőszolgáltatások elérésére is, mint az adatok távoli rendszerre való betöltésének mechanizmusát.

3.1.3 Képfájlok:

Mi képek tárolására FTP szervert használtunk. Név szerint a Nethelynél, mivel nem szeretnénk adatbázisunkat, backend-ünket lassítani azzal, hogy adatbázisba BLOB-ként tároljuk azt. Sokkal egyszerűbb egy elérési útvonalat letárolni adatbázisban például: “http://img.library.nhely.hu/img/default_book_img.png” és ezt az adatot küldözgetni adatbázis, backend és frontend között.

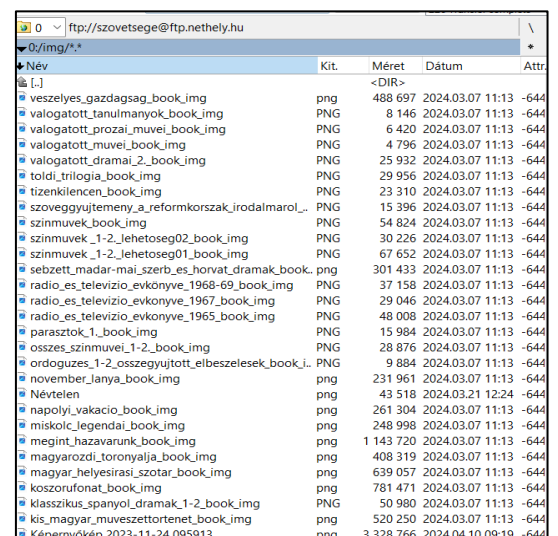
Ezzel a módszerrel sok adatnál nagyon meg lehet gyorsítani frontend-ünk válaszidejét.

A képen az látható, ahogy mi FTP szerverrel feltöltöttük a képeket és ott tároljuk azokat, de mi például az profil képeket is, és az adminisztrációs(WPF) setuppert is ott tároljuk.



Név	Kit.	Méret	Dátum	Attr.
[.]		<DIR>		
[img]		<DIR>	2024.04.04 12:19	-755
[img_profil]		<DIR>	2024.03.14 12:07	-755
[setup]		<DIR>	2024.04.18 10:12	-755

36. ábra (FTP szerveren található mappák)



Név	Kit.	Méret	Dátum	Attr.
[.]		<DIR>		
veszelyes_gazdagsag_book_img	png	488 697	2024.03.07 11:13	-644
valogatott_tanulmanyok_book_img	PNG	8 146	2024.03.07 11:13	-644
valogatott_prozai_muvei_book_img	PNG	6 420	2024.03.07 11:13	-644
valogatott_muvei_book_img	PNG	4 796	2024.03.07 11:13	-644
valogatott_dramai_2_book_img	PNG	25 932	2024.03.07 11:13	-644
tolid_trilogia_book_img	PNG	29 956	2024.03.07 11:13	-644
titzenklencen_book_img	PNG	23 310	2024.03.07 11:13	-644
szovegyujtemeny_a_reformkorszak_irodalmarol...	PNG	15 396	2024.03.07 11:13	-644
szinmuvek_book_img	PNG	54 824	2024.03.07 11:13	-644
szinmuvek_1-2_lehetoseg02_book_img	PNG	30 226	2024.03.07 11:13	-644
szinmuvek_1-2_lehetoseg01_book_img	PNG	67 652	2024.03.07 11:13	-644
sebzett_madar_mai_szerb_es_horvat_dramak_book...	png	301 433	2024.03.07 11:13	-644
radio_es_televizio_ekonyve_1968-69_book_img	PNG	37 158	2024.03.07 11:13	-644
radio_es_televizio_ekonyve_1967_book_img	PNG	29 046	2024.03.07 11:13	-644
radio_es_televizio_ekonyve_1965_book_img	PNG	48 008	2024.03.07 11:13	-644
parasztok_1_book_img	PNG	15 984	2024.03.07 11:13	-644
osszes_szinmuvei_1-2_book_img	PNG	28 876	2024.03.07 11:13	-644
ordoguzes_1-2_osszegujtott_elbeszelesek_book_i...	PNG	9 884	2024.03.07 11:13	-644
november_lanya_book_img	png	231 961	2024.03.07 11:13	-644
Névtelen	png	43 518	2024.03.21 12:24	-644
napolyi_vakacio_book_img	png	261 304	2024.03.07 11:13	-644
miskolc_legendai_book_img	png	248 998	2024.03.07 11:13	-644
megint_hazavarunk_book_img	png	1 143 720	2024.03.07 11:13	-644
magyarozdi_toronyaljja_book_img	png	408 319	2024.03.07 11:13	-644
magyar_helyesirasi_szotar_book_img	png	639 057	2024.03.07 11:13	-644
koszorufonat_book_img	png	781 471	2024.03.07 11:13	-644
klasszikus_spanyol_dramak_1-2_book_img	png	50 980	2024.03.07 11:13	-644
kis_magyar_muveszettortenet_book_img	png	520 250	2024.03.07 11:13	-644
Képernyőkép 2023-11-24 095913	png	3 328 766	2024.04.10 09:19	-644

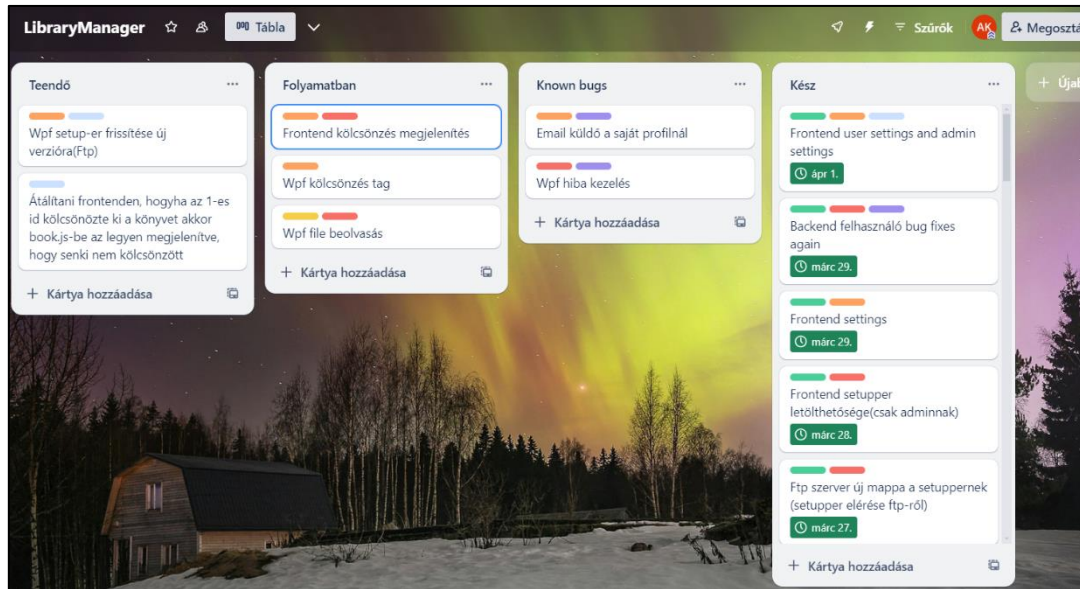
37. ábra (FTP szerveren eltárolt képek)



4 Csapat kommunikációs eszközök:

4.1 Trello:

A Trello egy együttműködési eszköz, amely táblákba rendezi a projektünket. A Trello egy pillantással megmutatja, hogy ki mivel dolgozik épp, és hol tart azzal kapcsolatban. Természetesen, ha frissítve van napi szinten.



38. ábra (A mi Trello-nk)

Mi is használtuk a Trello-t (Lásd [Trello](#)), ahogy az a 38. képen is látható. Nagyon megkönnyítette a csapatunk közötti kommunikációt és munkát, mikor nem suliban voltunk. Hasznos és egyben egyszerű weblap. Aki projekten dolgozik csapatban vagy akár egyedül, mind a kettő opcióhoz tökéletes ez az oldal. Mikor csapattal elkezdtek használni a Trello-t, akkor érezhetően kevesebb lett a téves kommunikáció.

4.2 Github:

A GitHub egy Git repository hosting szolgáltatás, amely webalapú grafikus felületet biztosít. Itt található a világ legnagyobb kódoló közössége. A programozók számos különböző nyelven találhatnak forráskódokat, és a parancssori felületet, a Git-et használhatják a változtatások végrehajtására és nyomon követésére.

A GitHub segít minden csapattagnak együtt dolgozni egy projekten bárhol, miközben megkönnyíti az együttműködést. Vissza lehet tölteni régebbi projekt verziókat is, ha valaki elrontott valamit vagy hibakeresésre kerül a sor.

Mi is a Github-ot használtuk **tárolásra**, és **verziókezelésre** egyaránt. (Lásd: [Github](#)).

4.3 Google Drive:

A Google Drive egy **felhőalapú tárhelyszolgáltatás**, amely lehetővé teszi a felhasználók számára, hogy fájlokat tároljanak és elérjenek online. A szolgáltatás szinkronizálja a tárolt dokumentumokat, fényképeket és egyébeket a felhasználó összes eszközén, beleértve a mobileszközöket, táblagépeket és PC-ket is.



Projektünk elején Github helyett Google Drive-ot használtunk programunk tárolására és át küldésére. Így tárolni és küldözgetni az adatokat sokkal nehezebb volt, mint a Github-os verzió. Szóval kisebb projektekhez tudom ajánlani a Google Drive használatát, de közepes és nagyobb programokhoz nem.

4.4 Messenger:

A Facebook Messenger egy **ingyenes** mobil üzenetküldő alkalmazás, amely azonnali üzenetküldésre, fényképek, videók, hangfelvételek megosztására és csoportos csevegésre használható.

A Messenger-t mi üzenet küldésre és információ megosztására használtuk.

4.5 Discord:

A Discord egy platform valós idejű szöveges, video, és hangcsevegés fogadására. Míg más közösségi platformok egy központi közösség köré épülnek, a Discord szerverekre vagy sok kisebb közösségre oszlik.

A Discord-ot megbeszélésekre, munkamegosztásra, és a csapattagjainknak való segítségre használtuk, mivel Discord-on lehetőség van arra, hogy megosszuk, mi is történik a mi gépünkön.

4.6 TeamSpeak 3:

A TeamSpeak 3 egy voice-over-internet protokollalkalmazás, amely lehetővé teszi a felhasználók számára, hogy valós időben kommunikáljanak és működjenek együtt hang használatával. Ez egy megbízható és könnyű eszköz egyedi funkciókkal, például AES-256 titkosítással, minimális késleltetéssel és kiváló hangminőséggel.

Erre a programra akkor volt szükségünk, amikor régebbi laptopokon kellett folytatnunk a projektünket és a Discord-ot nem futatta a gép.



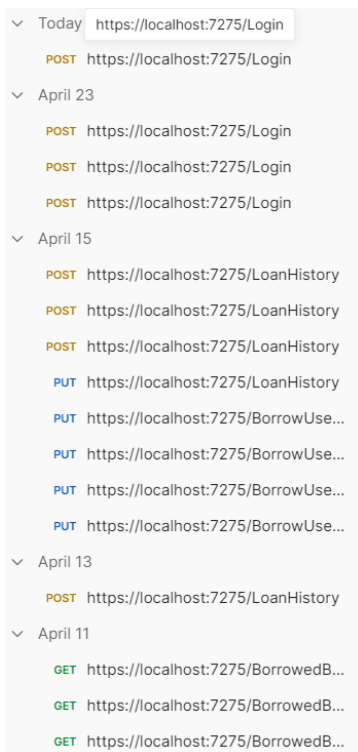
5 Tesztelés:

A tesztelés során számos alkalmazást használtunk, hogy meggyőződjünk a program biztos működéséről:

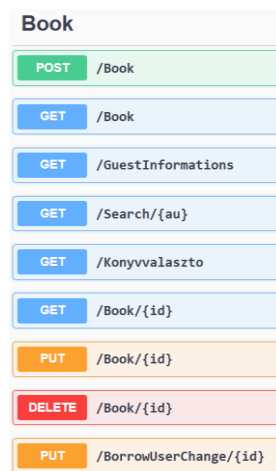
- **xUnit:** Az xUnit tesztelési keretrendszer a .Net és C#-ban írt programok tesztelésére szolgáló keretrendszer. A végpontok műveleteinek szabályos működésének tesztelése érdekében ezt a tesztelési módot használtuk (41.ábra).

- **Swagger:** A végpontok tesztelésére a beépített Swagger-t (40.-ábra) használtuk, mert egyszerűen és könnyen lehet alkalmazni.

- **Postman:** Az azonosítás tesztelésére Postman-t használtunk (39.-ábra), mert a Swagger időnként nem működött rendesen, működő bejelentkezési információkkal együtt is hibás visszajelzést adott, ha be is jelentkeztünk, a végpontok akkor sem tudták azonosítani a felhasználót, a Postman pedig hiba nélkül megcsinálta a bejelentkezést is, és az azonosítást is. A Postman technológiáról [itt olvashatnak bővebben](#).



40. ábra (Postman tesztek)



39. ábra (Swagger végpontok)

ControllerTester (15)	4,8 sec
ControllerTester (15)	4,8 sec
AuthorTest (5)	1,6 sec
Delete	314 ms
Get	5 ms
GetAll	8 ms
Post	9 ms
Put	1,3 sec
Publisher (5)	1,6 sec
Delete	271 ms
Get	1,3 sec
GetAll	32 ms
Post	7 ms
Put	3 ms
SeriesTest (5)	1,6 sec
Delete	40 ms
Get	4 ms
GetAll	15 ms
Post	1,3 sec
Put	263 ms

41. ábra (xUnit teszt)

6 Összefoglalás:

6.1 Saját utóérzés:

Dolgozatunkban egy olyan alkalmazás elkészítésével, megvalósításával és használhatóságával foglalkoztunk, mely nemcsak az iskolánk részére teszi lehetővé a könnyebb, hatékonyabb és gördülékenyebb digitális könyvtár használatát, hanem diáktársaink részére is.



A program elkészült, melyet magabiztosan tudunk átadni iskolánk részére. Büszkék vagyunk a **KaKöNyír** projektre.

Sok nehézséggel kellett szembenéznünk a project készítése során: **React** és **Web Api** megismerése és használása, megfelelő időbeosztás, új -és már ismert- programok még mélyebben való megismerése illetve, hogy hogyan dolgozzunk együtt csapatként. Ezek közül is a legnehezebb számunkra az utolsó volt, hisz eddig minden feladatot/munkát egyedül kellett megoldanunk. Ez a projekt megmutatta nekünk a csapatmunka előnyeit, megtanított bennünket egy célért küzdeni, kitartani és a legjobbat kihozni egymásból, de legfőképpen önmagunkból. Az előző pontokban felsorolt alkalmazások (Lásd [Csapat kommunikációs eszközök](#)) megkönnyítették munkánkat.

6.1.1 Megvalósított elképzelések:

6.1.1.1 Adatbázis:

- [Bővíthetőség könnyedsége](#)
- [Adateltávolítás leegyszerűsítése](#)
- [Adatbázis átláthatósága](#)
- [Adatbázis adatbetöltés gyorsítása](#)

6.1.1.1.1 Bővíthetőség könnyedsége:

Adatbázisunkat úgy hoztuk létre, hogy könnyen lehessen új táblákat és mezőket létrehozni benne. Leegyszerűsítettük és a hibákat lekezeljük. Mostmár könnyen lehet adatbázisunkat bővíteni.

6.1.1.1.2 Adateltávolítás leegyszerűsítése:

Az egész adatbázisban alkalmaztuk a CASCADE kapcsolatot vagyis, ha egy adat törlésre kerül, akkor amivel az az adat kapcsolatban állt az szintűgy törlésre kerül.

6.1.1.1.3 Adatbázis átláthatósága:

Úgy terveztük adatbázisunkat, hogy könnyen átlátható legyen. Így, ha less egy új csapattársunk, akkor ő is sokkal gyorsabban megtudja érteni adatbázisunk felépítését.

6.1.1.1.4 Adatbázis adatbetöltés gyorsítása

Adatbázisunk nem használ olyan erőforrásokat, amelyek nagyobb adattárolásnál nagyon lassíthatná programunkat például mi nem használtunk BLOB-ot adatbázisban, mivel az nagyon letudja lassítani az adatbetöltést. Mi FTP-t használtunk, amiről megtudhat többet a [FTP](#)-re kattintva.

Adatbázisunkról megtudhat többet itt: [Adatbázisunk](#)

6.1.1.2 Backend:

- [JWT tokennel történő védelem/azonosítás](#)
- [Email küldés](#)
- [A kód átláthatósága](#)
- [Speciális controllerek](#)

6.1.1.2.1 JWT tokennel történő védelem/azonosítás:

Programunk a legbiztonságosabb JWT kódolást használja. 512 karakterre kódolja le a felhasználó adatait, így megóvva felhasználónk értékes adatait.



6.1.1.2.2 Email küldés:

Backendünk kezeli az email küldés lehetőségét is. Az admin felhasználó bárkinek tud email-t küldeni bármilyen szöveggel.

6.1.1.2.3 Kód átláthatósága:

Programunkat úgy terveztük, hogy minél jobban megfeleljen a “*Clean Code*” elveinek. Így ugyan úgy, mint adatbázisunknál, ha esetlegesen új csapattársunk lenne, akkor könnyebben tudna illeszkedni a fejlesztésbe, hisz gyorsabban megérti a kódunkat.

6.1.1.2.4 Speciális Controllerek:

Többféle “speciális controller-je” is van backendünknek, mint például: “*LoanHistory*”, “*AccountImgController*”, “*BookController*”. Mindegyik több és nehéz függvényekkel rendelkezik.

Megtudhat többet a: [Backend](#)-re kattintva.

6.1.1.3 FrontEnd:

- [Leegyszerűsített felhasználói felület](#)
- [Modern kinézet](#)
- [Program élesbe állíthatósága](#)

6.1.1.3.1 Leegyszerűsített felhasználói felület:

Frontendünket úgy terveztük, hogy a leginkább felhasználóbarátibb legyen. Úgy díszítettük, helyeztük el az oldalon a dolgokat, hogy minden korosztálynak könnyen és egyszerűen egy gombnyomással teljesítsék kéréseiket.

6.1.1.3.2 Modern kinézet:

A komponenseket az oldalon úgy alakítottuk, díszítettük, hogy a 2024-es elvárásoknak megfeleljen. Próbáltuk oldalunkat a mai tinédzsereknek tetsző módon elkészíteni, hisz oldalunkat az iskolák tanulóinak terveztük, hogy minél több ember megragadjon az oldal kinézete.

6.1.1.3.3 Program élesbe állíthatósága:

Programunkat úgy fejlesztettük, hogy iskolánk könyvtárosának és a projekt kiadójának mindig megmutattuk haladásunkat és úgy fejlesztettük az oldalt, ahogy ők javasolták.

6.1.1.4 Asztali Alkalmazás (WPF):

- [Bejelentkezéssel történő védelem](#)
- [Adatbázis teljeskörű manipulálása](#)
- [Speciális igények kielégítése](#) (Több felhasználó feltöltése egyszerre)

6.1.1.4.1 Bejelentkezéssel történő védelem:

A felhasználó miután letöltötte az asztali alkalmazást, akkor elindítása után egy bejelentkezési fül jelenik meg, ahol mást nem fog át engedni csak az Admin jogosultságúakat. Többet megtudhat itt: [WPF](#)-ünk.

6.1.1.4.2 Adatbázis teljeskörű manipulálása:

Az asztali alkalmazás segítségével az adatbázis minden adatát könnyen és egyszerűen lehet módosítani, törölni, hozzáadni.



6.1.1.4.3 Speciális igények kielégítése:

Asztali alkalmazásunk tud txt fájlból felhasználó adatokat feltölteni. Egyszerre többet is. Ezeket a txt fájlban “;”-vel kell elválasztani.

6.1.2 Továbbfejleszthetőségek:

A követelményeket kielégítő rendszer megvalósult, de a tesztelés után folytatott beszélgetések során több új ötlettel gazdagodott az igények halmaza, melyek egy része a munka gyorsítását, más részük az univerzálisabb működést célozzák.

- A könyvek megjelenítésénél az admin tudjon az adott könyv adatain változtatni
- Szezonális Frontend megjelenítés például: *télen téli díszítés a weblapon*
- Asztali alkalmazásnál könyveket is fel lehessen vinni txt-ből. Egyszerre többet is
- Több profilkép minden felhasználónak.
- Szezonális eventek
- Season Pass
 - Új keretek feloldhatósága
 - Új profilképek feloldhatósága
 - A könyvek megjelenítő keretek más feloldhatósága
- Egyedi profilkeretek
- Home oldal szépítése
- Prémium előfizethetőség megoldása
 - Új funkciók feloldása ezzel

Ezekre a változtatásokra azért lenne szüksége az oldalnak, mivel a mai generáció már nem nagyon foglalkozik ilyen oldalakkal, ezért csapatommal gondolkoztunk új dolgokon, amik egy tinédzser érdeklődését is megmozgatja.

7 Köszönetnyilvánítás

Szeretnénk köszönetet mondani mentorainknak: Kerényi Róbert Nándornak, Kiss Rolandnak, Németh Bencének, Négyesi Gábornak, Farkas Zoltánnak, akik nélkülözhetetlen támogatást és útmutatást nyújtottak számunkra a projekt során.

Külön köszönet illeti, Kerényi Róbert Nándort, aki szakértelmével és türelmével mindvégig mellettünk állt.

Köszönjük Óvári Enikő könyvtáros együttműködését, kedvességét, a tesztidőszakban való részvételét.

Köszönjük, hogy hittek bennünk...



8 Irodalomjegyzék:

1. Téma: [Adatbázis](#) (wamp), webcím: <https://www.hostinger.com/tutorials/what-is-wamp>, letöltés dátuma: 2024. 04. 18.
2. Téma: [MVC modell](#), webcím: <https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/>, letöltés dátuma: 2024. 04. 19.
3. Téma: [Entity Framework](#), webcím: <https://www.entityframeworktutorial.net/entityframework6/what-is-entityframework.aspx>, letöltés dátuma: 2024. 04. 19.
4. Téma: [CORS Policy](#), webcím: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>, letöltés dátuma: 2024. 04. 22.
5. Téma: [C#](#), webcím: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>, letöltés dátuma: 2024. 04. 22.
6. Téma: [CRUD](#), webcím: <https://www.codecademy.com/article/what-is-crud>, letöltés dátuma: 2024. 04. 22.
7. Téma: [React](#), webcím: <https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs>, letöltés dátuma: 2024. 04. 22.
8. Téma: [Node.js](#), webcím: <https://www.netguru.com/glossary/node-js>, letöltés dátuma: 2024. 04. 23.
9. Téma: [Visual Studio Code](#), webcím: <https://www.turing.com/kb/ultimate-guide-visual-studio-vs-visual-studio-code>, letöltés dátuma: 2024. 04. 24.
10. Téma: [FTP](#), webcím: <https://www.techtarget.com/searchnetworking/definition/File-Transfer-Protocol-FTP>, letöltés dátuma: 2024. 04. 24.

Az összes Sources megtalálható [itt](#).