SZAKDOLGOZAT

Verovszki Dominika

NYÍREGYHÁZA, 2025

Mobil Könyvtár

Verovszki Dominika

Programtervező informatikus BSc

Konzulens: Andrikó Imre

ügyvivő szakértő, informatikus

2025

1 Tartalomjegyzék

1. Bevezető

1.1 Előszó

Az általam tervezett mobilalkalmazás célja, hogy segítséget nyújtson a könyveim rendszerezésében és nyilvántartásában. Az évek során jelentős mennyiségű könyvet vásároltam, és mára már kihívást jelent számomra nyomon követni, hogy mely kötetek találhatók meg a gyűjteményemben. Gyakran előfordul, hogy amikor új könyveket szeretnék venni, már nem emlékszem, mely kiadványok vannak meg és melyek hiányoznak.
Ez az alkalmazás lehetővé tenné számomra, hogy egyszerűen katalogizáljam a könyveimet, ISBN számok beolvasásával. Az alkalmazás segítene abban is, hogy gyorsan és könnyen rákereshessek egy adott könyvre, legyen szó egy szerző vagy cím alapján történő keresésről. A rendszer egyszerűsítené a könyvgyűjtemény kezelését, és megszüntetné azokat a bizonytalanságokat, amelyek a nyilvántartás hiányából fakadnak.

Nem csupán időt takarítanék meg, hanem könnyebben is követhetném nyomon a könyveim aktuális állapotát – például azt, hogy mely könyveket adtam kölcsön másoknak, vagy melyek azok, amelyeket még nem olvastam el. A digitális katalógus átláthatóbbá tenné a gyűjteményemet, és bármikor, bárhol hozzáférhetnék, így a jövőbeli könyvvásárlások is egyszerűbbé válnának, hiszen a kezemben lenne egy naprakész lista az összes könyvemről.

Ezzel az alkalmazással tehát egy modern, könnyen használható megoldást szeretnék létrehozni, amely nemcsak az életemet tenné könnyebbé, hanem lehetőséget adna arra is, hogy bármikor átlássam és menedzselhessem a könyveimet.

Ezen túlmenően, az alkalmazás elkészítése nemcsak a könyveim kezelését tenné könnyebbé, hanem egy kiváló lehetőséget biztosítana számomra, hogy tanuljak a mobilalkalmazás-fejlesztésről. Az elkészítés során rengeteget tanulhatnék arról, hogyan működik egy mobil app a háttérben, a teljes fejlesztési folyamatról, a frontend és backend integrációról, valamint arról, hogyan lehet hatékony és felhasználóbarát megoldásokat létrehozni. Ez egy olyan kihívás, amely során nemcsak technikai képességeimet fejleszthetem, hanem egy alkalmazást is létrehozhatok, amely az én igényeimet szolgálja.

Az is motivál, hogy bár léteznek hasonló alkalmazások a Google Play Áruházban, ezek közül egyik sem felel meg teljes mértékben az elvárásaimnak. Sok esetben hiányoznak azok a funkciók, amelyekre nekem szükségem lenne, vagy a dizájn nem felel meg az

elképzeléseimnek. Ezért döntöttem úgy, hogy egy saját alkalmazást készítek, amely teljesen testre szabható, így minden olyan funkciót és megjelenést biztosíthatok, amely számomra ideális. Így a saját igényeimnek megfelelően alakíthatom ki a könyvtározó alkalmazást, és nem kell kompromisszumokat kötnöm más megoldásokkal.

Ez az alkalmazás tehát nemcsak a mindennapi életemet könnyítené meg a könyveim kezelésében, hanem lehetőséget adna arra is, hogy értékes tapasztalatokat szerezzek a mobilfejlesztés világában, és egy valóban személyre szabott megoldást hozzak létre.

1.2 Elemzés

Ebben a rövid fejezetben szeretnék kitérni arra, hogy miért döntöttem saját app készítése mellett, ahelyett, hogy meglévőt választottam volna.
A fejlesztés előkészítő szakaszában irodalmi kutatást végeztem annak érdekében, hogy feltérképezzem a témához kapcsolódó, hasonló célú alkalmazásokat. A vizsgálat során kiderült, hogy magyar nyelven nem áll rendelkezésre olyan mobilalkalmazás, amely kifejezetten egyéni könyvgyűjtemény kezelésére, kategorizálására és ISBN-alapú adatlekérdezésre lenne optimalizálva.
Angol nyelvű környezetben ugyan léteznek hasonló alkalmazások (pl. Handy Library, Bookshelf-Your virtual library vagy My Library), azonban ezek közül van amelyik elsősorban közösségi platformként működik, nem pedig személyes, offline használatra optimalizált nyilvántartó alkalmazásként. Ezeknél az appoknál gyakran szükséges fiók létrehozása, internetkapcsolat, és sok esetben a felhasználói felület is más logika szerint épül fel, mint amit én célszerűnek találtam, illetve nem tartalmaz minden olyan funkciót, amire nekem szükségem van.
Ezek alapján egyértelművé vált számomra, hogy bár léteznek alternatívák, egyik sem tudja teljes mértékben kielégíteni az egyéni igényeimet. A célom egy olyan letisztult, egyszerűen használható, magyar nyelvű alkalmazás létrehozása volt, amely nem igényel külső regisztrációt, nem tárol adatokat felhőben, és lehetőséget biztosít a testre szabásra, például háttérkép kiválasztására.
Az általam készített alkalmazás tehát a létező megoldásokhoz képest egyszerűbb, célzottabb és személyesebb megközelítést alkalmaz. A fejlesztés egyik legfőbb motivációja éppen az volt, hogy olyan eszközt hozzak létre, amely kifejezetten az én elvárásaimnak felel meg, és nem kell kompromisszumokat kötnöm a használat során.

1.3 A szakdolgozat rövid bemutatása

Szakdolgozatom első részében bemutatom a mobil applikációm telepítését, felépítését és működését felhasználói szempontból.

Ezt követően bemutatom a felhasznált technológiákat.

Ezután megvizsgálom a szakdolgozatom felépítésére fejlesztői szemszögből, mind frontend mind backend oldalról, és kitérek a fejlesztés menetére.

Majd összefoglalom a munkám eredményét a megszerzett ismeretek fényében és megemlítem a jövőbeni fejlesztési terveimet is.

A szakdolgozat végén padig a Felhasznált források rész kap helyet.

2. Felhasználói dokumentáció

Ez a rész az alábbi témákra fog kitérni, részletes leírást adva a felhasználó(k)nak:

- Mi az alkalmazás célja.
- Mik a fő funkciók.
- Hogyan kell telepíteni és elindítani az alkalmazást.
- Hogyan kell használni az alkalmazást.

2.1 Célja

Az alkalmazásom célja, a könyvek nyilvántartása és kezelése, könnyen és egyszerűen. Egy mobiltelefonra van szükség hozzá, így lehetővé téve a felhasználónak, hogy bárhonnan tudja kezelni a könyveit, ami nagyon kényelmessé teszi az alkalmazást.
Saját példából kiindulva, akár a könyvesbolt kellős közepén elővehető és használható, elkerülve ezzel annak az esélyét, hogy már meglévő könyvet vásároljon a felhasználó.

2.2 Fő funkciók

Az alkalmazás funkciói a következők:

- Scan ISBN
- Kereső mező
- Szűrés
- Rendezés
- Megjelenítés
- Új könyv hozzáadása
- Háttérkép beállítása
- Meglévő könyv szerkesztése
- Meglévő könyv törlése

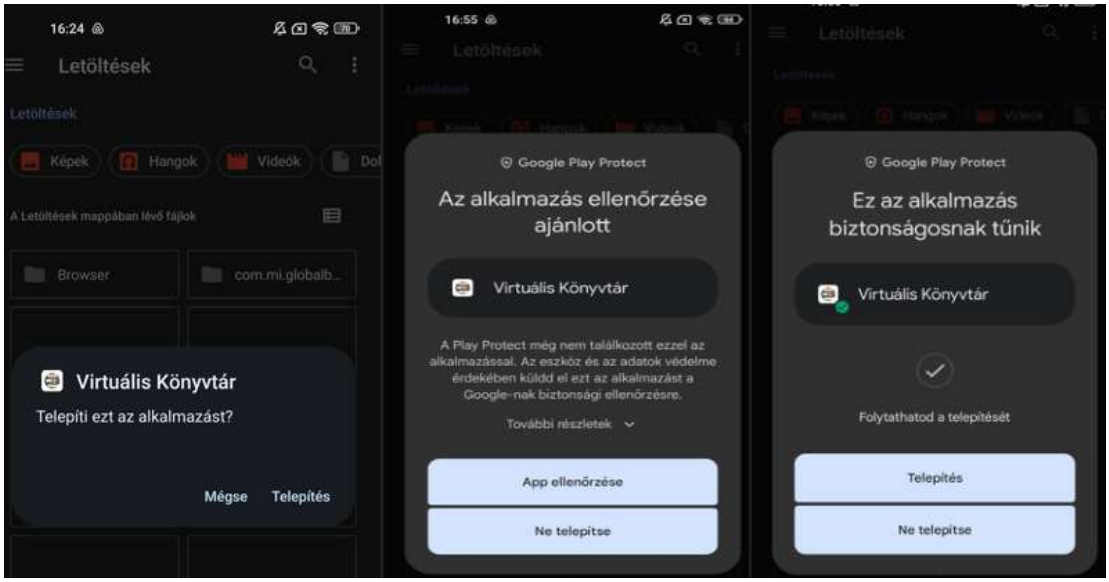
2.3 Telepítés és indítás

Ebben a fejezetben bemutatom azokat a lépéseket, amiket a felhasználónak kell eszközölnie ahhoz, hogy saját eszközén telepítse és használni tudja az alkalmazást.

2.3.1 Lépés: telepítés

Az Android Studio által készített .apk állományt letöltöm a Google Drive-ról. A telefonomban a letöltések alatt megkeresem a .apk fájlt, majd kiválasztom a telepítés opciót, ahogy a kép mutatja.
Telepítés során megjelenik egy popup ablak, miszerint ajánlott ellenőrizni az alkalmazást. Itt kiválasztom az App ellenőrzése gombot, majd, amikor lefutott az ellenőrzés, akkor pedig a Telepítés gombot.

A képek ezt mutatják be sorrendben.



1. ábra Telepítés

2.3.2 Lépés: app indítása

Miután a telepítés befejeződött, megnyitom az alkalmazást.
Az alkalmazást egy általam testreszabott ikonnal lehet elindítani:

2. ábra Az alkalmazás ikonja

Feltöltöttem pár könyvet mintának, hogy amikor még üres az adatbázis, akkor is legyen mit megjeleníteni.

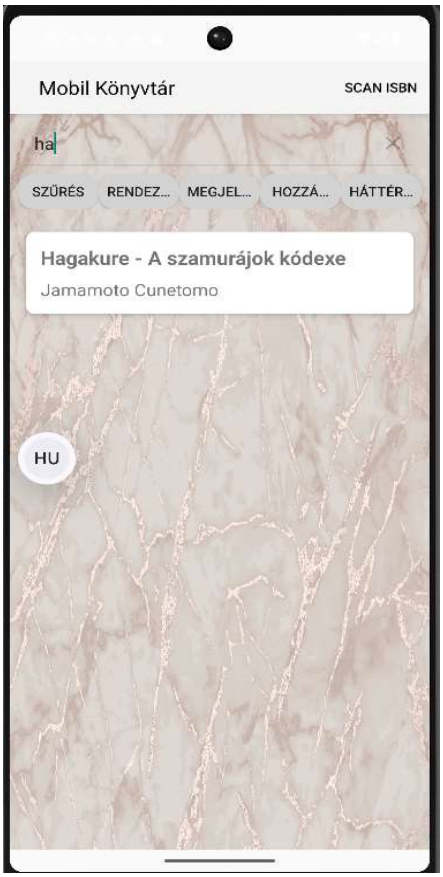
3. ábra Az alkalmazás főoldala

2.4 Használata

2.4.1 Főoldal

Az applikáció indulásakor ez az oldal jelenik meg, itt találhatóak a könyvek és az egyes funkciók (keresés, Scan ISBN, szűrés, rendezés, megjelenítés, hozzáadás, háttérkép beállítása).
A Scan ISBN arra szolgál, hogy ellenőrizzem beolvasható-e az általam kiválasztott könyv ISBN-je.

A keresés mező, pedig cím vagy szerző alapján keres az adatbázisban.



4. ábra Kereső mező használata

2.4.2 Szűrés

Az első funkció a szűrés, itt lehet adott tulajdonságok alapján szűrni a könyveket.
Van, ahol radio gombot használtam, ott csak egy opciót lehet kiválasztani (nyelv, állapot, státusz, kinél van és típus) és van, ahol check box-ot, ott több dolgot is ki lehet választani.

Miután kiválasztottam minden szükséges szűrési lehetőséget, a lap alján található Mentés gombra kattintva tudom megtekinteni az eredményét.

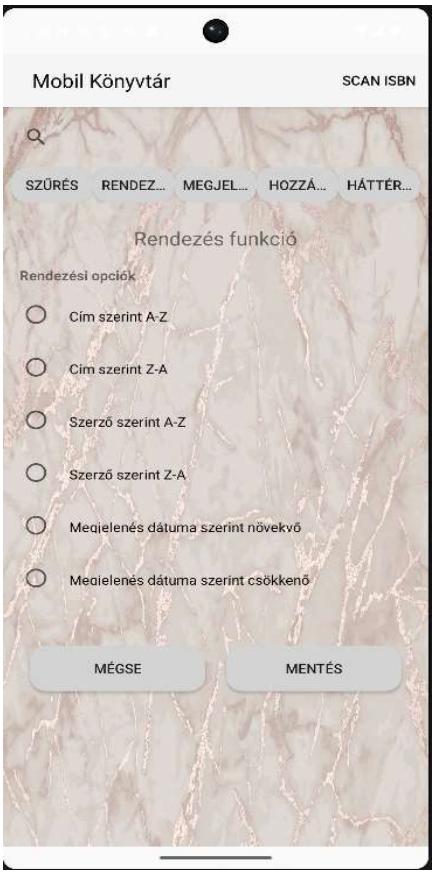
A Virtuális Könyvtárra kattintva lehet visszakerülni a főoldalra.

5. ábra Szűrő használata

2.4.3 Rendezés

Itt tudom bizonyos kritériumok szerint rendezni az adatbázisban megtalálható könyveimet. Ilyen kritérium a cím szerint, szerző szerint és a dátum szerint növekvő, illetve csökkenő rendezés.
Szintén a Mentés gomb az, ami menti a módosításaimat.

A Virtuális Könyvtárra kattintva lehet visszakerülni a főoldalra.

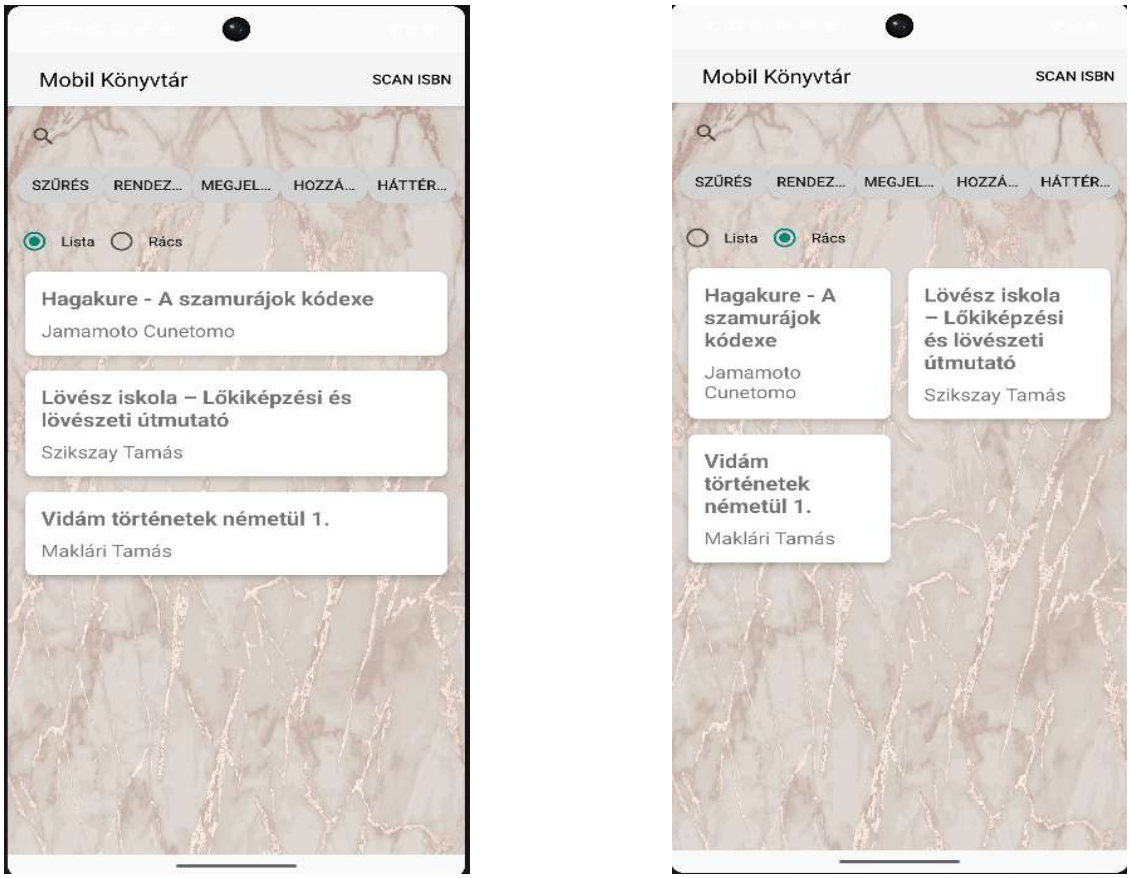


6. ábra Rendezés használata

2.4.4 Megjelenítés

Itt lehet kiválasztani a megjelenítés módját, ami lehet lista vagy rács. Viszont ez a beállítás csak ezen az oldalon érvényes és arra szolgál, hogy ha sok könyvem van az adatbázisban akkor gyorsabban végig lehessen nézni a listát.
A lista nézet az alapértelmezett.

A Virtuális Könyvtárra kattintva lehet visszakerülni a főoldalra.



7. ábra Rendezés használata

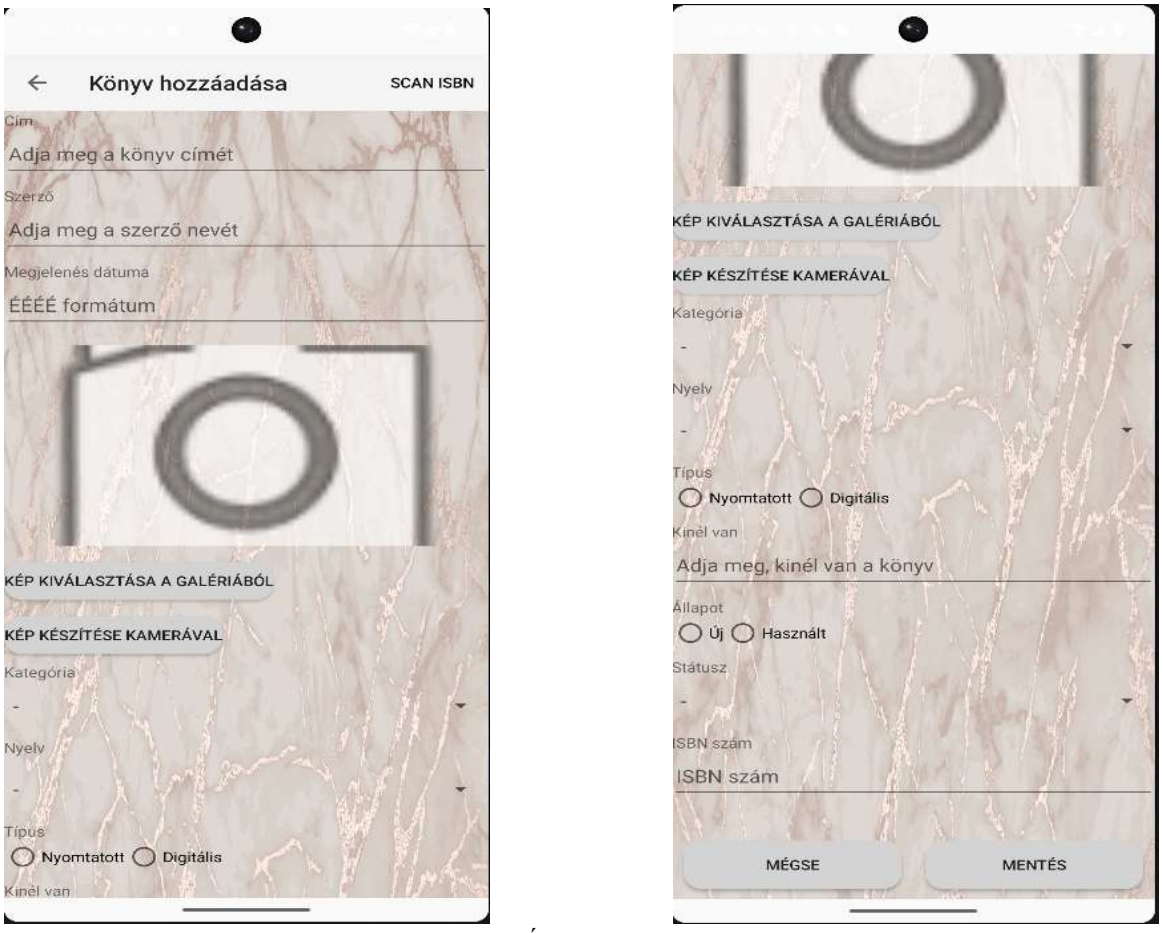
2.4.5 Új könyv hozzáadása

Ezen az oldalon lehet új könyvet hozzáadni az adatbázishoz abban az esetben, amennyiben az adott könyv még ne szerepel az adatbázisban.
A következő mezőkből áll: cím, szerző, kiadás éve, fénykép készítés vagy kiválasztás galériából, kategória, nyelv, típus, kinél van, állapot, státusz és ISBN szám.

Ezek közül a cím és a szerző megadása kötelező, illetve a dátumnál tettem olyan megkötést, hogy 4 számból kell állnia abban az esetben, ha meg van adva.

Itt van lehetőség arra, hogy úgy adjak hozzá az adatbázishoz könyvet, hogy az ISBN számát beszkenneljem.

A Virtuális Könyvtárra kattintva lehet visszakerülni a főoldalra.

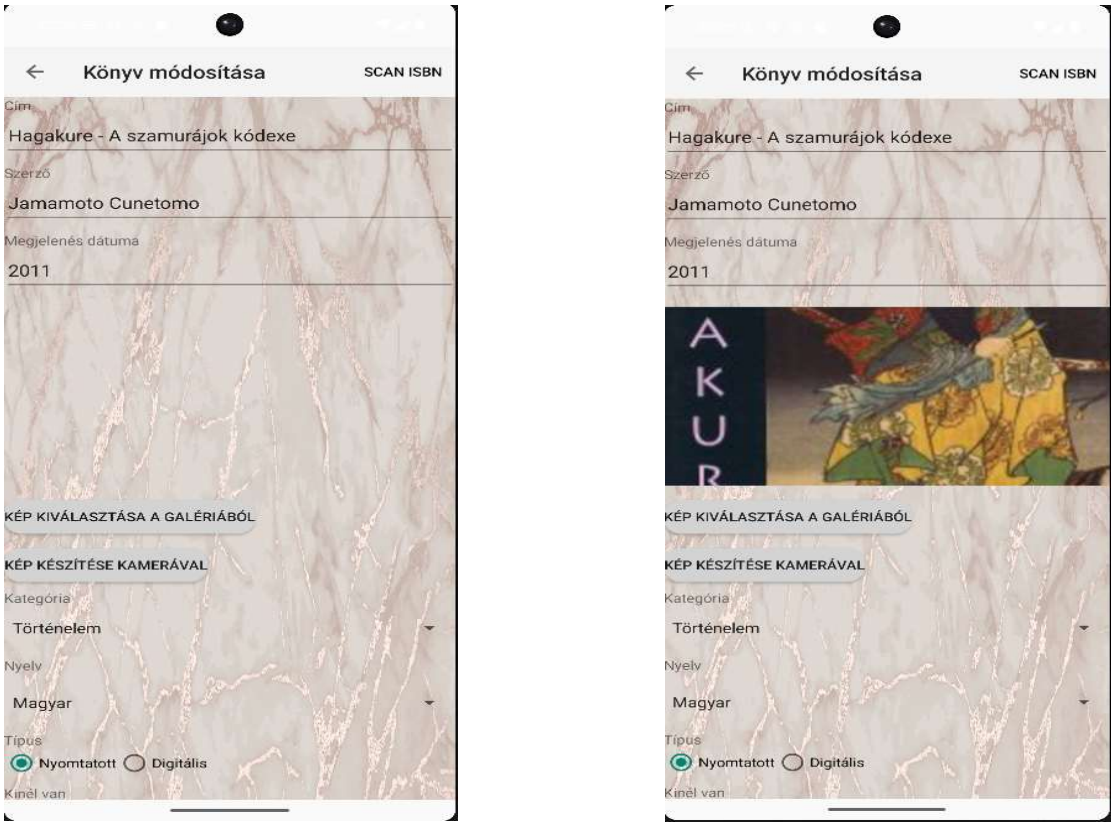


8. ábra Új könyv hozzáadása

2.4.6 Meglévő könyv szerkesztése

Itt már meglévő könyv adatait lehet szerkeszteni, miután kiválasztom az adott könyvet betölti az alkalmazás az adatbázisból a hozzá tartozó adatokat és ugyanazon a felületen jeleníti meg, mint az új könyv hozzáadást.
Ha úgy módosítom, hogy a cím és a szerző megegyezik egy meglévő könyvével, akkor hibaüzenet jelenik meg és nem kerül mentésre a módosítási kísérletem.

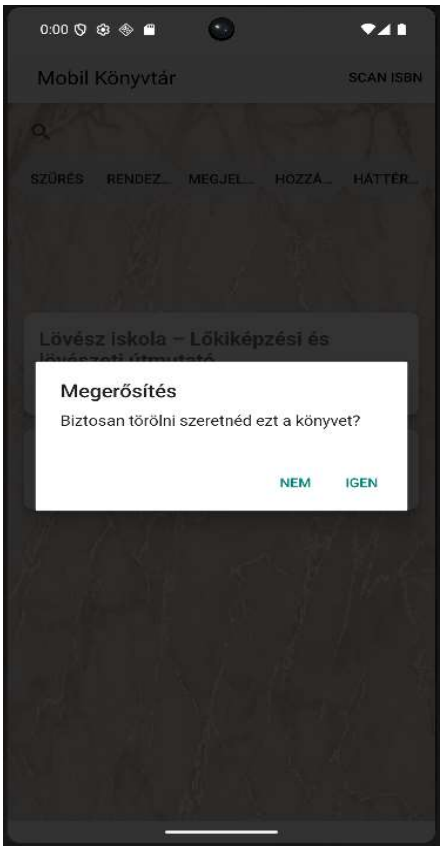
A Virtuális Könyvtárra kattintva lehet visszakerülni a főoldalra.



9. ábra Meglévő könyv szerkesztése

2.4.7 Könyv törlése

Ha a kezdő oldalon megjelenített könyvek közül egyet jobbra vagy balra húz a felhasználó, megjelenik egy figyelmeztető üzenet, amelyben megkérdezem a felhasználót, hogy valóban törölni szeretné az adott könyvet, igen válasz esetén a könyv törlésre kerül, nem válasz esetén visszalép az alkalmazás.

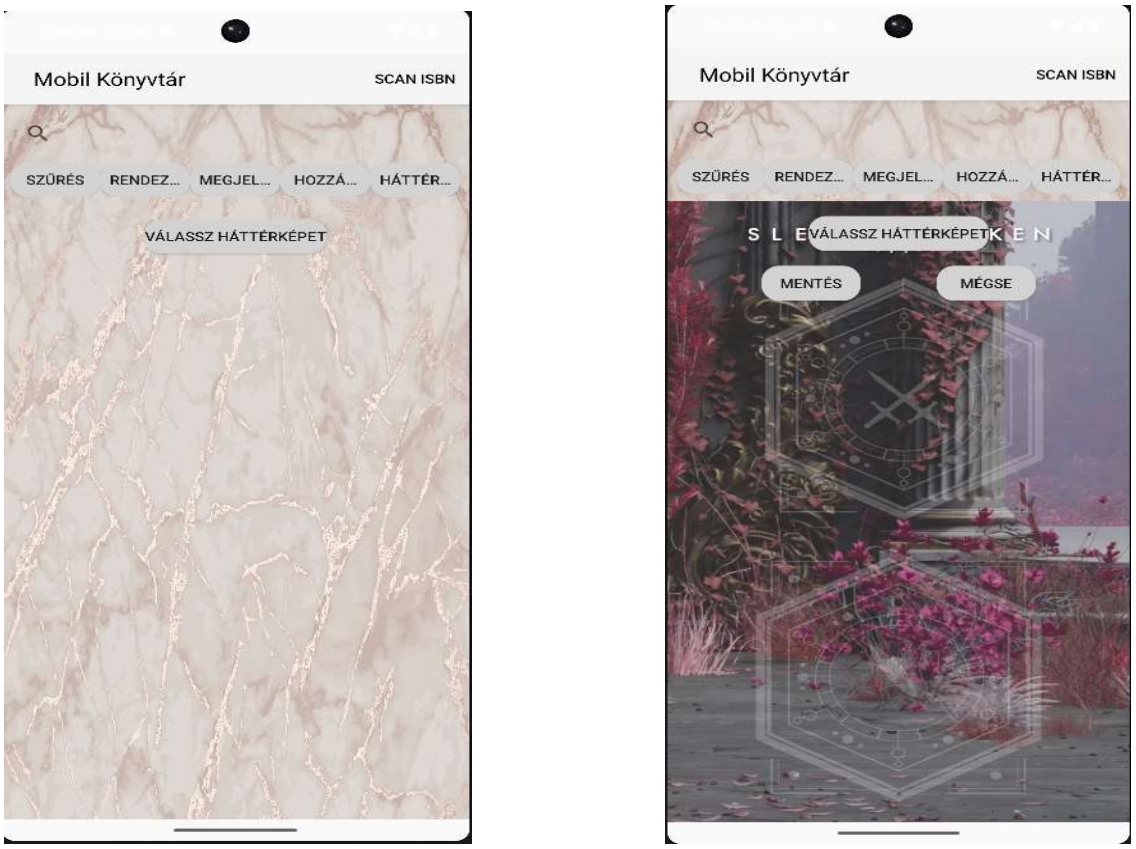


10. ábra Könyv törlése

2.4.8 Háttérkép beállítása

Az alkalmazás egy előre beállított háttérképpel települ, viszont a felhasználónak lehetősége van ezt a későbbiekben módosítani a saját készülékén található képek segítségével.

A Virtuális Könyvtárra kattintva lehet visszakerülni a főoldalra.



11. ábra Háttér beállítása

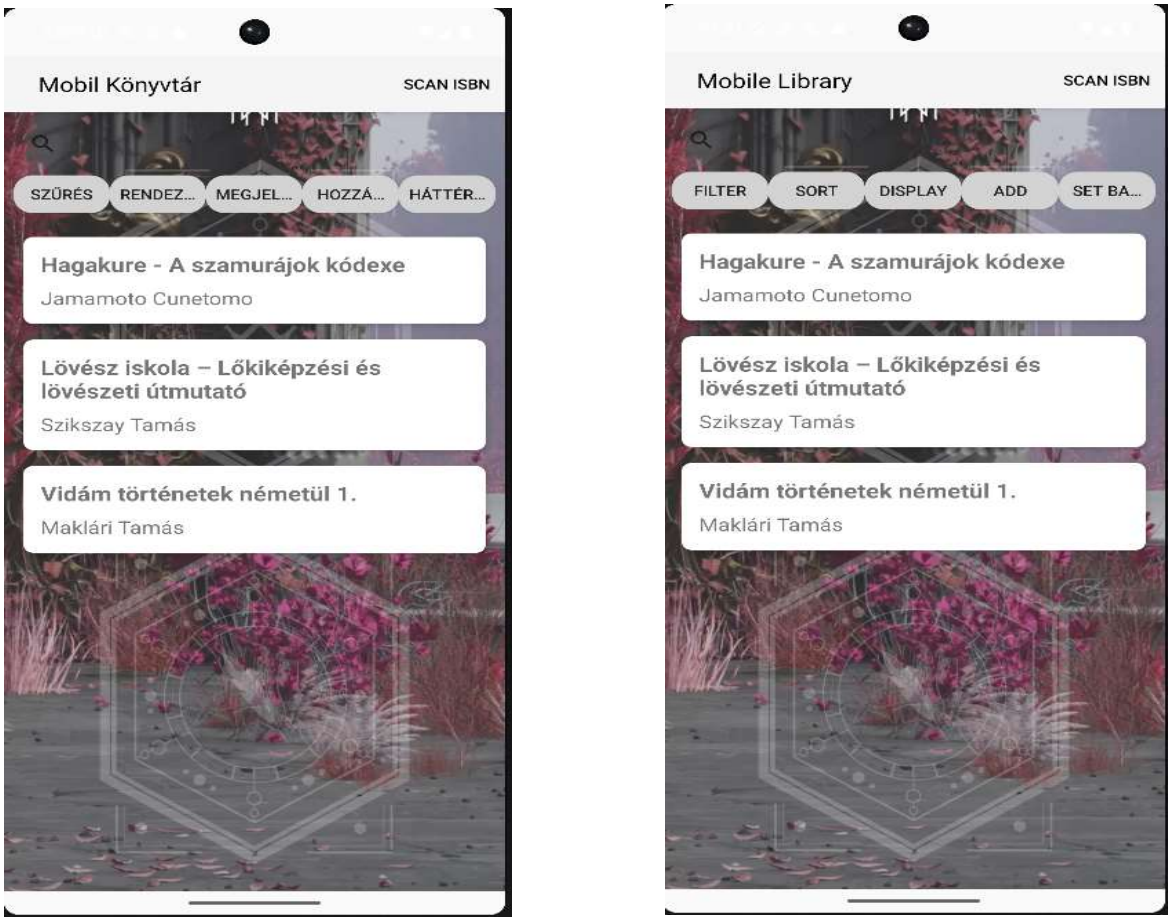
Mikor kiválasztok egy képet, akkor megjelenik előnézetben, ha rányomok a Mentés gombra, akkor történik a tényleges beállítás az egész alkalmazáson beül, és megjelenik egy üzenet is, ami a beállítás sikerességét közli.

12. ábra Újonnan beállított háttérkép

2.4.9 Nyelvek

2 nyelven érhető el jelenleg az alkalmazás, angolul és magyarul. Az, hogy melyik nyelv van használatban a telefon nyelvi beállításától függ.

A cím és a szerző nem változik, olyan nyelven jelenik meg, amilyenen felvételére került.



13. ábra Az alkalmazás magyar és angol nyelven

3. Felhasznált technológiák

3.1 Android Studio

Az Android Studio a Google Android operációs rendszerének hivatalos integrált fejlesztői környezete (IDE), amely a JetBrains IntelliJ IDEA szoftverére épül, és kifejezetten android-fejlesztéshez készült.[i]

Mire való az Android Studio?

- Android alkalmazások fejlesztésére:

Lehetővé teszi androidos telefonokra, tabletekre, okosórákra szánt appok készítését.

- Kódírásra:

Támogatja a java és kotlin (az ajánlott nyelv) nyelveket.

Felhasználói felület (UI) tervezésére:
Drag-and-drop felülettervező (Layout Editor) segíti a vizuális elemek gyors összeállítását.

- Emulátor használatára:

Beépített emulator segítségével tesztelhető az alkalmazás, fizikai készülék nélkül is.

- Hibakeresésre és tesztelésre:

Beépített eszközökkel figyelhető a teljesítményt, egység- és UI-teszteket lehet futtatni.

- APK generálásra:

Lehetővé teszi az alkalmazás csomagolását és aláírását, hogy feltölthető legyen a Google Play Áruházba.

14. ábra Android Studio

3.2 Kotlin

A kotlin egy modern, statikusan típusos programozási nyelv, amelyet a JetBrains fejlesztett ki, és elsősorban android alkalmazások fejlesztésére használnak.[ii]

15. ábra Kotlin

3.3 Room

A Room perzisztencia könyvtár egy absztrakciós réteget biztosít az SQLite felett, hogy gördülékeny adatbázis-hozzáférést tegyen lehetővé, miközben az SQLite teljes erejét kihasználja.

A következő előnyöket nyújtja:

- SQL lekérdezések fordítási idejű ellenőrzése.
- Annotációk, amelyek minimalizálják az ismétlődő és hibára hajlamos sablonkódot.
- Egyszerűsített adatbázis-migrációs útvonalak.[iii]

16. ábra Room

3.4 Google Books API

A Google Books API lehetővé teszi az alkalmazás számára, hogy teljes szöveges keresést végezzen, és lekérje a könyvadatokat, és az e-könyvek elérhetőségét.[iv]

Mire jó a Google Books API?

- Könyvkeresés:

Lehetőség van kulcsszavak, cím, szerző, ISBN stb. alapján könyveket keresni.

- Könyvinformációk lekérése:

Le lehet kérni egy adott könyv részletes adatait.

- Beépíthető könyvajánlók:

Weboldalakon vagy mobilappokban lehet megjeleníteni könyveket.

17. ábra Google Books API

3.5 ZXing

A ZXing egy nyílt forráskódú könyvtár, amelyet vonalkódok és QR-kódok beolvasására és generálására fejlesztettek. Főként java nyelven íródott, de több platformra is léteznek implementációi (pl. Android).

Mire való a ZXing?

- QR-kódok beolvasása kamerával vagy képfájlból.
- Vonalkódok dekódolása.
- QR- vagy vonalkód generálása.
- Android alkalmazásokban való felhasználás beépített kamerás szkennelőként.[v]

18. ábra ZXing

3.6 ChatGPT

A ChatGPT (Generative Pre-trained Transformer) az OpenAI mesterséges intelligencia (MI) kutató laboratórium által kifejlesztett chatbot, mely a felhasználókkal való folyamatos kommunikáció automatizálása során értelmezőmodelleket használ, melyek segítségével a bevitt információkat azonnal interaktívan kezeli.[vi]

19. ábra ChatGPT

3.7 Udemy

Az Udemy a világ egyik legnagyobb online tanulási platformja, ahol bárki hozzáférhet videós tanfolyamokhoz különféle témákban — például programozás, nyelvtanulás, üzlet, design, marketing, fotózás, és még sok más.

Mire jó az Udemy?

- Tanulás otthonról:

Bárhonnan, bármikor hozzáférhető anyagok (weben, mobilon, akár offline is).

- Gyakorlati tudás:

A tanfolyamokat szakemberek készítik, sokszor valós projekteken keresztül mutatják be a tananyagot.

- Kedvező árak:

Gyakori akciók vannak, amikor a tanfolyamokat akár pár ezer forintért is meg lehet vásárolni.

- Nincs előfizetés:

Egy tanfolyamért egyszer kell fizetni, és örökös hozzáférést biztosít.

- Tanúsítvány:

A legtöbb tanfolyam elvégzése után egy nem hivatalos oklevelet lehet szerezni.

20. ábra Udemy

4. Fejlesztői dokumentáció

Az alkalmazásom kotlin nyelven készült Android Studio fejlesztői környezetben. A projekt egy könyvtárkezelő alkalmazás, amely lehetőséget nyújt könyvek listázására, hozzáadására, szkennelésére és szerkesztésére.
Az alkalmazás minimum Android 7.0 (API 24) verziótól támogatott, és az Android 15 (API 35) verzióra van optimalizálva. A fordításhoz az Android 15 SDK-t használja.

21. ábra Részlet a build.gradle.kts fájlból

Értelmezése:

- compileSdk:

Ez az az SDK verzió, amivel az app fordul. Ennek legalább olyannak kell lennie, mint a targetSdk.

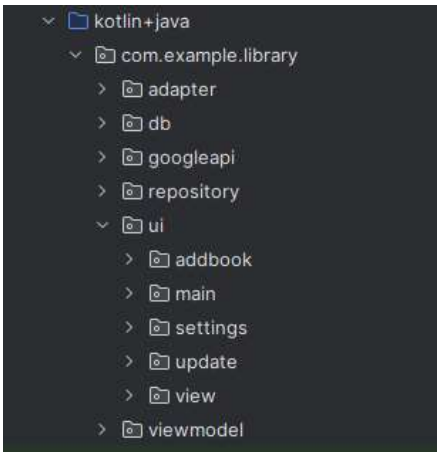
- minSdk:

Ez a minimum Android verzió, amin az app elindul.

- targetSdk:

Az a verzió, amelyhez az app optimalizálva van. Ezt érdemes mindig a legfrissebbre állítani.

Projekt struktúrája:



22. ábra Részlet a projekt struktúrájáról

Az alkalmazás a com.example.library nevű csomagban helyezkedik el, és modulárisan lett felépítve.

A csomagok a következők:

- adapter:

Az adatok megjelenítéséhez használt adaptereket tartalmazza, például RecyclerView adaptereket.

- db:

Az adatbázis-kezelésért felelős osztályokat tartalmazza.

- googleapi:

A Google API-khoz kapcsolódó kommunikációs logikát kezeli.

- repository:

A repository réteg biztosítja az adatok központi kezelését.

- ui:

Az alkalmazás felhasználói felületéhez kapcsolódó aktivitások és fragmentek, további almappákra bontja.

- addbook:

Könyv hozzáadásához kapcsolódó képernyők találhatóak itt (AddBookActivity, BookScanner).

- main:

A fő képernyő (MainActivity).

- settings:

Háttérbeállítások kezelése (BackgroundSelectorFragment).

- update:

Könyvfrissítési funkciók (UpdateActivity).

- view:

A könyvlista megjelenítéséhez használt szűrési és rendezési fragmentek (FilterFragment, SortFragment, ViewFragment).

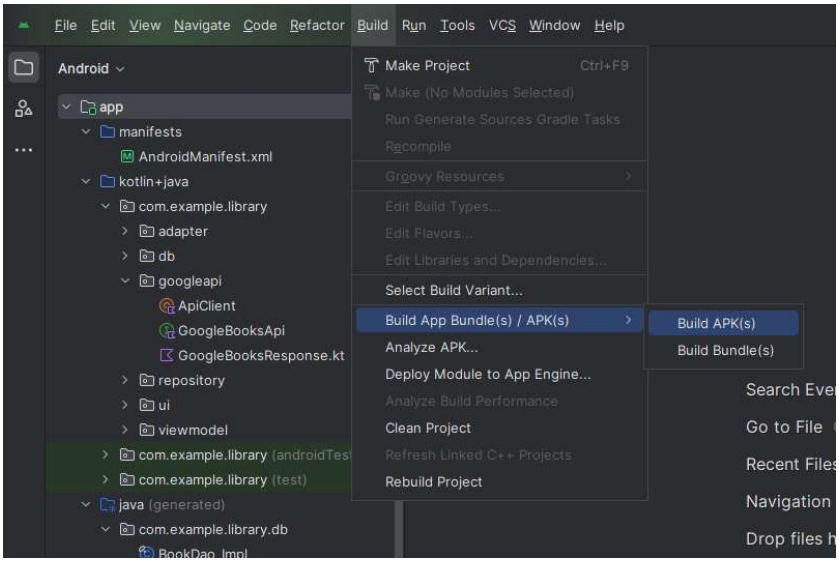
- viewmodel:

A ViewModel osztályt tartalmazza.

4.1 Build

Mielőtt bemutatom az alkalmazás felépítését és a fejlesztés menetét, előtte ismertetem az .apk készítésének lépéseit.
Mivel szinte csak saját felhasználásra készült az applikációm, és nem tervezem elérhetővé tenni a Play Áruházban, ezért az általam a build során generált .apk fájl használatával telepíthető. Ez a folyamat készülékenként eltérő lehet, én Xiaomi 11 Lite 5G NE, valamint Samsung Galaxy S22 telefonjaimra telepítettem.

A Menü sávban megkeresem a Build opciót, azon belül pedig a Build App Bundles(s) / APK(s), és itt a Build APK(s)-t választom, ahogy a képen is látszik.



23. ábra .apk fájl készítésének menete

4.2 Backend

4.2.1 Belépési pont

Az Android alkalmazásokban a belépési pontot az android rendszer határozza meg, és ez általában az az Activity, amit az AndroidManifest.xml fájlban MAIN és LAUNCHER intent-filterrel megjelölünk.

Az én alkalmazásomban a belépési pont a MainActivity, amely az alábbi módon van definiálva a Manifest fájlban:

```
<activity
    android:name="com.example.library.ui.main.MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

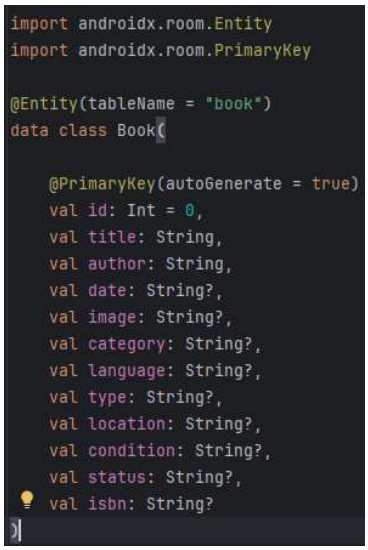
24. ábra Részlet a AndroidManifest.xml fájlból

Ez az Activity felel a fő felület megjelenítéséért, ahonnan a többi funkció (könyv hozzáadás, nézetek szűrése, beállítások stb.) elérhető.

4.2.2 Adatbázis-kezelés

Az alkalmazás az Android Room könyvtárát használja az adatok helyi tárolására. A Room egy ORM (Object Relational Mapping) megoldás, amely lehetővé teszi, hogy az adatbázis-kezelést kotlin osztályokon és annotációkon keresztül valósítsuk meg.

Book:
A Book osztály reprezentálja az adatbázisban tárolt könyveket. Ez az osztály a @Entity annotációval van ellátva, ami jelzi, hogy egy táblát reprezentál a Room adatbázisban.



25. ábra Book.kt osztály

@PrimaryKey(autoGenerate = true): automatikusan növekvő azonosító.

Az id, title és author mezők kötelezőek, a többi opcionális.

BookDao (Data Access Object):

A BookDao interfész tartalmazza az adatbázis-műveletekhez szükséges metódusokat, mint például beszúrás, lekérdezés, frissítés és törlés. A Room ezeket az annotációk alapján implementálja.

A lekérdezések LiveData típusban adnak vissza adatokat, így az UI automatikusan frissül.

26. ábra Részlet a BookDao.kt osztályból

BookDB:
A BookDB egy absztrakt osztály, amely örökli a RoomDatabase osztályt, és meghatározza a DAO elérhetőségét. Singleton mintával biztosított az egy példányban történő elérés.

Itt szúrok be néhány előre definiált könyvet az adatbázisba, hogy amikor telepítés után először indul el az alkalmazás, ne legyen teljesen üres.

```
val executorService = Executors.newSingleThreadExecutor()
executorService.execute {
    bookDao?.insertBook(
        Book(
            title = "Hagakure - A szamurájok kódexe",
            author = "Jamamoto Cunetomo",
            date = "2011",
            image = null,
            category = "Történelem",
            language = "Magyar",
            type = null,
            location = null,
            condition = null,
            status = null,
            isbn = "9786156702197"
        )
    )
```

27. ábra Részlet a BookDB.kt osztályból

4.2.3 BookRepo

A Repository célja, hogy elválassza az adatok forrását az alkalmazás többi részétől. Ez lehetővé teszi, hogy az UI komponensek (pl. ViewModel, Activity) ne ismerjék közvetlenül az adatbázis vagy más adatforrás működését. Így a kód tisztább, könnyebben tesztelhető és bővíthető lesz.

A BookRepo osztály az egyetlen kapcsolódási pont az adatbázis és a ViewModel között. A Room DAO metódusait wrapeli.

getAllBooks:
A könyvek teljes listáját figyeli LiveData segítségével. Ez automatikusan frissül, amikor változás történik az adatbázisban.

```
fun getAllBooks(): LiveData<List<Book>> {
    return books // Az összes könyv adatainak visszaadása
}
```

28. ábra getAllBooks metódus

insert(), delete(), update():

Aszinkron műveletek, így nem blokkolják a fő szálat, és kotlin suspend kulcsszóval lettek megvalósítva.

```
fun insert(book: Book) {
    executorService.execute {
        bookDao.insertBook(book) // Könyv hozzáadása az adatbázishoz
    }
}

// Könyv frissítése az adatbázisban (háttérszálon)
fun update(book: Book) {
    executorService.execute {
        bookDao.updateBook(book) // Könyv adatainak frissítése
    }
}

// Könyv törlése az adatbázisból (háttérszálon)
fun delete(book: Book) {
    executorService.execute {
        bookDao.deleteBook(book) // Könyv törlése
    }
}
```

29. ábra insert, update és delete metódusok

Továbbá itt vannak még a különféle kritériumok alapján történő keresések.

```
// Könyvek keresése cím és/vagy szerző alapján
fun searchBooks(query: String): LiveData<List<Book>> {
    val formattedQuery = "%$query%" // A keresési kifejezés formázása
    return bookDao.searchBooks(formattedQuery) // Keresés a DAO-ban
}

// Könyv keresése ISBN alapján
suspend fun findBookByIsbn(isbn: String): Book? {
    return bookDao.findBookByIsbn(isbn) // Könyv keresése az ISBN alapján
}
```

30. ábra searchBooks és findBookByIsbn metódusok a BookRepo.kt osztályból

4.2.4 BookViewModel

A ViewModel felelős a UI és az adatforrás közötti közvetítésért. A ViewModel nem tartalmaz UI logikát, viszont túléli az Activity/Fragment újralétrehozását (pl. képernyőforgatáskor), és biztosítja az adatokat a UI számára.

A BookViewModel a ViewModel osztályból származik, és egy BookRepo példány segítségével éri el az adatokat. A LiveData típus biztosítja, hogy a UI figyelni tudja az adatváltozásokat.

4.2.5 BookAdapter

Az Adapter az android RecyclerView komponensének egyik kulcsfontosságú része, amely az adatokat jeleníti meg listanézet formájában. Az alkalmazásban a BookAdapter felelős a könyv adatok megjelenítéséért a felhasználói felületen. A BookAdapter a RecyclerView.Adapter leszármazottja, amely egyéni nézeteket tölt fel könyv adatokkal. Gondoskodik arról, hogy az adatok hatékonyan és újrahasznosítható módon jelenjenek meg, valamint lehetőséget biztosít az egyes elemekre való kattintás kezelésére is. Az adaptert a RecyclerView komponenshez kell hozzárendelni, általában egy Fragment vagy Activity belsejében. A setBooks() hívással frissíthető a megjelenített adatlista, míg a setOnItemClickListener() segítségével az egyes könyvekre való kattintások kezelhetők.

4.2.6 Fragmentek

A com.example.library.ui.view csomag az alkalmazás megjelenítési logikájának egy részét valósítja meg. Ebben a csomagban található három kulcsfontosságú fragment, amelyek lehetővé teszik a könyvek listájának vizuális testre szabását és megtekintését.

FilterFragment:
Ez a fragment felelős a könyvlista szűrési lehetőségeinek megjelenítéséért. A felhasználó különböző szempontok alapján (például kategória, szerző, kiadási év) tudja szűkíteni a megjelenített könyvek körét.

Főbb funkciói:

- Szűrési opciók lekérdezése a ViewModel-től.
- Felhasználó által kiválasztott szűrési feltételek továbbítása a könyvlistát megjelenítő komponensnek.
- UI elemek kezelése (pl. checkboxok, dropdown menük).

```
// Nyelv szerinti szűrés kezelése
val languageRadioGroup: RadioGroup = view.findViewById(R.id.language_radio_group)
languageRadioGroup.setOnCheckedChangeListener { _, checkedId ->
    val selectedLanguage = when (checkedId) {
        R.id.language_hu -> "Hungarian"
        R.id.language_en -> "English"
        R.id.language_de -> "German"
        R.id.language_jp -> "Japanese"
        R.id.language_cn -> "Chinese"
        R.id.language_sp -> "Spanish"
        R.id.language_sg -> "Scottish Gaelic"
        else -> "None selected"
    }
    Toast.makeText(context, "Selected language: {selectedLanguage}", Toast.LENGTH_SHORT).show()
}
```

31. ábra Példa a nyelv szerinti szűrésre

SortFragment:

A SortFragment lehetőséget biztosít a könyvek rendezésére különféle paraméterek szerint, például cím, szerző neve vagy dátum alapján.

Főbb funkciói:

- Rendezési szempontok megjelenítése.
- Kiválasztott rendezés leküldése a ViewModel-nek.
- Kommunikáció a ViewFragment-tel az adatok újrarendezése érdekében.

```
sortRadioGroup.setOnCheckedChangeListener { _, checkedId ->
    val selectedOption = when (checkedId) {
        R.id.sort_title_az -> "Title A-Z"
        R.id.sort_title_za -> "Title Z-A"
        R.id.sort_author_az -> "Author A-Z"
        R.id.sort_author_za -> "Author Z-A"
        R.id.sort_date_ascending -> "Publication date ascending"
        R.id.sort_date_descending -> "Publication date descending"
        else -> "None selected"
    }

    // Toast üzenet a választás megjelenítéséhez (opcionálisan itt hívhatod meg a rendezési logi
    Toast.makeText(context, "Selected sort option: {selectedOption}", Toast.LENGTH_SHORT).show()
}
```

32. ábra A felhasználó által választott opció figyelése

ViewFragment:

Ez a fragment jeleníti meg a könyvek listáját a kiválasztott opció alapján. A könyvek megjelenítése RecyclerView segítségével történik, amely az adapteren keresztül tölti be az adatokat.

Főbb funkciói:

- Könyvadatok megjelenítése különböző nézetek alapján.

```
    viewTypeRadioGroup.setOnCheckedChangeListener { _, checkedId ->
        when (checkedId) {
            R.id.view_type_list -> setupRecyclerViewLayout(LinearLayoutManager(requireContext()))
            R.id.view_type_grid -> setupRecyclerViewLayout(GridLayoutManager(requireContext(), spanCount: 2))
        }
    }
}
```

33. ábra Ez a kódrészlet felelős a nézetváltásért

4.2.7 UpdateActivity

A com.example.library.ui.update csomag az alkalmazás azon funkcióját valósítja meg, amely lehetővé teszi a meglévő könyvek adatainak módosítását.

Az UpdateActivity osztály célja, hogy lehetővé tegye a meglévő könyvadatok módosítását egy külön képernyőn, amely valójában az AddBookActivity újrahasznosításával történik, de "frissítés" módban.

Főbb funkciók:

- Launcher regisztrációja az eredmény fogadásához.

Az ActivityResultLauncher segítségével az AddBookActivity elindítása történik, és visszatérés után az új adatok kezelése.

Ellenőrzi, hogy az eredmény sikeres volt-e.

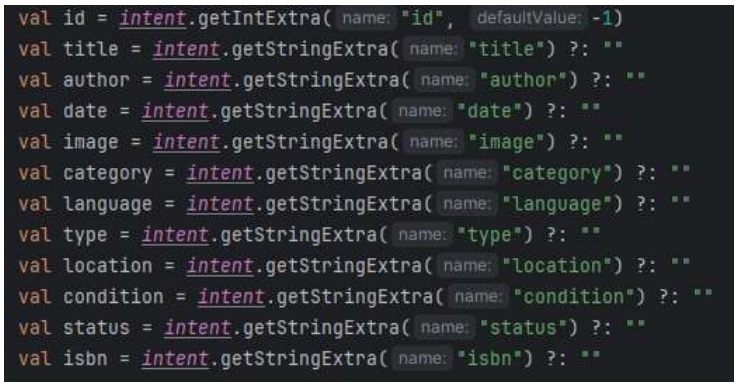
Kinyeri az összes módosított adatot az Intent-ből.

Visszaküldi az eredményt a hívó komponensnek.

34. ábra Részlet a Launcher-ből

- Előző adatok betöltése az Intent-ből.

Az UpdateActivity az indításkor kapott adatokból (könyv azonosító, cím, szerző stb.) összeállít egy új Intent-et, amellyel elindítja az AddBookActivity-t frissítés módban.



35. ábra Részlet az új Intent összeállításából

- Kommunikáció az AddBookActivity-val.

A putExtra() hívások révén az AddBookActivity minden szükséges könyvadatot megkap, így az űrlapot előre kitöltve jeleníti meg. Az isUpdate flag alapján tudja, hogy nem új könyvet, hanem meglévőt kell szerkeszteni.

```
val updateIntent = Intent( packageContext: this, AddBookActivity::class.java).apply {
    putExtra( name: "id", id)
    putExtra( name: "title", title)
    putExtra( name: "author", author)
    putExtra( name: "date", date)
    putExtra( name: "image", image)
    putExtra( name: "category", category)
    putExtra( name: "language", language)
    putExtra( name: "type", type)
    putExtra( name: "location", location)
    putExtra( name: "condition", condition)
    putExtra( name: "status", status)
    putExtra( name: "isbn", isbn)
    putExtra( name: "isUpdate",  value: true)
}
```

36. ábra putExtra hívások

- Felhasználói élmény szempontjából.

A felhasználó számára teljesen egyértelmű, hogy mi történik, az adatok előre kitöltve jelennek meg, és csak a kívánt mezőket kell módosítania.

4.2.8 BackgroundSelectorFragment

A BackgroundSelectorFragment lehetőséget nyújt a felhasználónak, hogy az alkalmazás háttérképét egyéni képpel személyre szabja. A kiválasztott kép megjelenik előnézetként, majd el is menthető és visszatölthető később.

Főbb funkciók:

- Kép kiválasztása a galériából.

Az ActivityResultLauncher (pickImageLauncher) segítségével a felhasználó képet választhat a galériából.
A kiválasztott kép URI-ja átmenetileg elmentésre kerül, és előnézetben megjelenik.

```
pickImageLauncher = registerForActivityResult(ActivityResultContracts.GetContent()) { uri: Uri? ->
    uri?.let {
        handleImageSelection(it)
    }
}
```

37. ábra Részlet a kódból, ahol a háttérkép kerül kiválasztásra

- Háttér előnézet.

A previewBackground(uri: Uri) metódus betölti és megjeleníti az előnézeti hátteret a backgroundLayout komponensben.

```
private fun previewBackground(uri: Uri) {
    try {
        val inputStream = requireContext().contentResolver.openInputStream(uri)
        val drawable = Drawable.createFromStream(inputStream, uri.toString()) //

        drawable?.let {
            binding.backgroundLayout.background = it
        }
    } catch (e: Exception) {
        e.printStackTrace()
    }
}
```

38. ábra Részlet a kódból, ahol az előnézet megjelenítése történik

- Kép végleges mentése.

A mentés gomb (saveButton) használatával a kép elmentésre kerül a SharedPreferences segítségével.

A kiválasztott URI-hoz olvasási engedély is biztosítva van (takePersistableUriPermission).

A háttér beállítása megtörténik az activity szintjén (MainActivity.setAppBackground(drawable).

```
private fun saveUriToPrefs(uri: Uri) {
    val prefs = requireContext().getSharedPreferences( name: "wallpaper_prefs", Context.MODE_PRIVATE)
    prefs.edit().putString("wallpaper_uri", uri.toString()).apply() // URI mentése
}
```

39. ábra Részlet a háttérkép mentésének megvalósításából

- Mentett háttér automatikus betöltése.

A loadSavedBackground() metódus betölti az előzőleg elmentett háttér URI-t, és beállítja a megfelelő képet.

```
private fun loadSavedBackground() {
    val prefs = requireContext().getSharedPreferences( name: "wallpaper_prefs", Context.MODE_PRIVATE)
    val uriString = prefs.getString( key: "wallpaper_uri",  defValue: null) // Előzőleg mentett URI

    uriString?.let {
        val uri = Uri.parse(it)
        setBackgroundFromUri(uri)
    }
}
```

40. ábra LoadSaveBackgroung metódus

- Felhasználói élmény.

Gombok (Mentés, Mégse) csak akkor jelennek meg, ha egy új kép lett kiválasztva.
Sikeres háttérbeállítás vagy hiba esetén visszajelző Toast üzenet jelenik meg a felhasználónak.

4.2.9 AddBookActivity

Ez az osztály egy Android Activity, amely lehetővé teszi a felhasználóknak, hogy könyveket adjanak hozzá egy adatbázishoz, vagy frissítsenek meglévő könyv adatokat. Ez az Activity felelős egy új könyv hozzáadásáért vagy meglévő szerkesztéséért az alkalmazásban. Több mezőt tartalmaz a könyvadatok rögzítéséhez: cím, szerző, megjelenési dátum, kép, kategória, nyelv, típus, állapot, ISBN stb. Az Activity többféle funkcióval támogatja a felhasználót, beleértve a képfeltöltést, ISBN szkennelést, illetve az űrlapadatok előtöltését frissítés esetén. Ez a rész a könyv mentéséért/frissítéséért felelős logikát, valamint a kamera- és galériahasználatot, valamint a Google Books API integrációját tartalmazza. Tartalmaz továbbá segédfüggvényeket, amelyek megkönnyítik az űrlap adatainak kezelését.

Főbb komponensek és működésük:

- UI elemek inicializálása.

A findViewById() segítségével minden fontos mező és vezérlőelem (pl. EditText, Spinner, ImageView, RadioGroup, Button) példányosítva van.

```
title = findViewById(R.id.book_title)
author = findViewById(R.id.book_author)
date = findViewById(R.id.book_date)
image = findViewById(R.id.book_image)
category = findViewById(R.id.book_category)
language = findViewById(R.id.book_language)
type = findViewById(R.id.book_type)
location = findViewById(R.id.book_location)
condition = findViewById(R.id.book_condition)
status = findViewById(R.id.book_status)
isbn = findViewById(R.id.book_isbn)
```

41. ábra Részlet a mezők inicializálásáról

- Spinner adapterek.

A Spinner típusú elemekhez (kategória, nyelv, státusz) listák kapcsolódnak, amelyeket az array.xml fájl határoz meg.

- Új könyv mentése vagy meglévő frissítése.

A Intent paraméterek alapján eldönti, hogy új könyvet veszünk-e fel, vagy meglévőt frissítünk.
Ha frissítünk, betölti a meglévő adatokat az űrlapra.
Mindkét esetben ellenőrzi, hogy a könyv már szerepel-e az adatbázisban (cím + szerző alapján).

- Hátterek beállítása.

A korábban kiválasztott háttérkép (SharedPreferences-ből) be van töltve és alkalmazva az Activity-re – ez összekapcsolódik a BackgroundSelectorFragment működésével.

- Kép kiválasztása vagy készítése.

Kép rögzítése kamerával (CAMERA_REQUEST_CODE): Amennyiben a kamera használatához szükséges engedélyek megvannak, megnyílik a kamera. A létrejött képet URI alapján beállítja az ImageView-ba.
Kép kiválasztása galériából (GALLERY_REQUEST_CODE): A kiválasztott képet az alkalmazás átmásolja a belső tárhelyre, majd megjeleníti.

- ISBN szkennelés kezelése.

Az onActivityResult() metódus kezeli a ZXing könyvtár által visszaadott ISBN-t.
Sikeres szkennelés esetén a scannedIsbn változó frissül, és a fetchBookDataAndSave() metódus meghívásra kerül, amely az ISBN alapján lekéri az adatokat egy külső forrásból.
Megszakított szkennelés esetén egyszerűen csak egy Toast üzenet jelenik meg a felhasználónak.

```
private fun fetchBookDataAndSave(isbn: String) {

    Toast.makeText(
        applicationContext,
        "Fetch started with ISBN: {isbn}",
        Toast.LENGTH_SHORT
    ).show()
```

42. ábra Részlet a fetchBookDataAndSave metódusból

- Toolbar menü kezelése.

Az Activity rendelkezik egy Toolbar-ral, ahol elérhető az ISBN szkennelés gomb (action_scan_isbn). Erre kattintva elindul a könyvszkenner (BookScanner.startScan()), amely elindítja az ISBN kód beolvasását.

- Engedélykérések kezelése.

Az onRequestPermissionsResult() biztosítja, hogy a szükséges engedélyek (kamera, galéria elérés) megléte után a funkciók használhatók legyenek. Elutasítás esetén figyelmeztető üzenetet kap a felhasználó.

```
if (requestCode == 0) {
    if (grantResults.isNotEmpty()) {
        val granted = grantResults.all { it == PackageManager.PERMISSION_GRANTED }
        if (granted) {
            // Ha minden szükséges engedély meg van adva
            Toast.makeText(
                context: this,
                "Permissions granted",
                Toast.LENGTH_SHORT
            ).show()
        } else {
            // Ha valamelyik engedélyt megtagadták
            Toast.makeText( context: this, "Permissions denied", Toast.LENGTH_SHORT)
                .show()
        }
    }
}
```

43. ábra Részlet a onRequestPermissionsResult metódusból

4.2.10 BookScanner

A BookScanner osztály arra szolgál, hogy lehetővé tegye az ISBN kódok beolvasását okostelefonok kamerájával. Az osztály a vonalkódok beolvasására specializálódik, és a beolvasott adatokat az alkalmazás számára elérhetővé teszi. Az ZXing könyvtár (IntentIntegrator) segítségével integrálja a beolvasási funkciót, így egyszerűen elindítható és kezelhető a vonalkódok szkennelése android környezetben. Lehetővé teszi a felhasználónak, hogy beolvassa egy könyv ISBN-jét, majd feldolgozza az eredményt.

Főbb funkciói:

- startScan()

A metódus elindítja a vonalkód szkennelési folyamatot, és beállítja a szükséges paramétereket az IntentIntegrator objektumban.

```
class BookScanner(private val activity: Activity) {

    // A startScan metódus indítja el a szkennelést
    fun startScan() {
        // Inicializáljuk az IntentIntegrator-t, amely a szkennelési műveletet végzi
        val integrator = IntentIntegrator(activity)

        // Beállítjuk a prompt szöveget, amely a szkennelés előtt jelenik meg
        integrator.setPrompt("Scan ISBN")

        // Bekapcsoljuk a hangjelzést, hogy a szkennelés eredményét jelezzük
        integrator.setBeepEnabled(true)

        // Lehetővé tesszük, hogy a tájolás ne legyen lezárva a szkennelés során
        integrator.setOrientationLocked(false)

        // Csak vonalkód típusokat szeretnénk olvasni (nem QR kódot, hanem az ISBN típusú vonalkódot
        integrator.setDesiredBarcodeFormats(IntentIntegrator.ONE_D_CODE_TYPES)

        // Elindítjuk a szkennelést
        integrator.initiateScan()
    }
```

44. ábra startScan metódus

- handleResult(…)

Ez a metódus kezeli a szkennelési eredményeket. Ha sikeres volt a szkennelés, akkor az ISBN kódot visszaadja egy callback függvénynek.

4.2.11 A Google Books API integrációja

A szakdolgozat során az alkalmazás egyik fő funkciója a könyvinformációk lekérdezése az interneten keresztül. Ezt a célt a Google Books API segítségével valósítottam meg. Ez biztosítja az API-val való kapcsolatot, az adatok strukturált kezelését és a lekérdezések egyszerű kezelését.

ApiClient:
Az ApiClient egy Kotlin objektum, amely egy Retrofit alapú HTTP kliens konfigurációját valósítja meg. Ez a komponens felelős a Google Books API-val történő kommunikációhoz szükséges alapbeállításokért. A BASE_URL konstans a következő alap URL-t definiálja:

```
9       // Az alap URL, amelyhez az endpointok csatlakoznak
10      private const val BASE_URL = "https://www.googleapis.com/books/v1/"
```

45. ábra Részlet az ApiClient.kt osztályból

A googleBooksApi property egy lazy inicializálással ellátott példány, amely a Retrofit könyvtár segítségével egy GoogleBooksApi interfészt implementáló objektumot hoz létre. Az API hívások során a JSON válaszokat a GsonConverterFactory konvertálja kotlin objektumokká, biztosítva a típusbiztonságot.

GoogleBooksApi:

A GoogleBooksApi egy Retrofit-kompatibilis interfész, amely definiálja az API végpontokat. Jelenleg egyetlen metódust tartalmaz, amely ISBN alapú könyvkeresést tesz lehetővé:

```
12      @GET("volumes")
13      fun getBookByIsbn(
14          @Query("q") query: String // A lekérdezés paramétere, pl.: "isbn:9780140449136"
15      ): Call<GoogleBooksResponse> // A válasz egy GoogleBooksResponse objektum lesz
16  }
```

46. ábra Részlet a GoogleBooksApi.kt osztályból

Ez a metódus egy GET típusú HTTP-kérést küld a volumes végpontra. A q paraméterként megadott lekérdezés határozza meg, hogy milyen feltételek szerint történjen a keresés (például: isbn). A válasz GoogleBooksResponse formájában érkezik vissza, ami a válasz JSON struktúráját írja le kotlin nyelven.

VolumeInfo:

A VolumeInfo adatosztály a lekérdezett könyvek metaadatait reprezentálja. A következő mezőket tartalmazza:

```
// A könyv metaadatait tartalmazó osztály
data class VolumeInfo(
    val title: String,
    val authors: List<String>?,
    val date: String?,
    val image: String?,
    val category: String?,
    val language: String?,
    val type: String?,
    val location: String?,
    val condition: String?,
    val status: String?,
    val isbn: String?
)
```

47. ábra VolumeInfo adatosztály

Az osztályban minden mező opcionális (?), kivéve a title mezőt, így rugalmasan kezelhető a válasz, még hiányzó adatok esetén is.

4.2.12 A MainActivity működése

A MainActivity az alkalmazás fő képernyője, amely a legtöbb interakciót kezeli a felhasználóval. Többek között itt történik meg a könyvek megjelenítése, szűrése, keresése, hozzáadása, frissítése és törlése. Emellett ez az Activity felel a háttérkép kezeléséért, a fragmentek közötti navigációért, valamint az ISBN szkennelés integrációjáért is.

- Felület inicializálása.

Az onCreate() metódus során inicializálásra kerül:
Toolbar: Egyéni címsor, amelyre kattintva újraindul az Activity.
RecyclerView: A könyvlista megjelenítésére szolgáló komponens.
Gombok: A különböző műveletek (szűrés, rendezés, nézetváltás, hozzáadás, háttérválasztás) vezérlésére.

- Fragmentkezelés.

A felhasználó különböző gombok segítségével válthat a különböző Fragment nézetek között (FilterFragment, SortFragment, ViewFragment, BackgroundSelectorFragment). Ezek a fragmentek a fragment_container View elemben jelennek meg, míg a könyvlista ilyenkor elrejtésre kerül.

- Könyvek megjelenítése és frissítése.

A ViewModel segítségével figyeljük az adatbázisban tárolt könyveket. Bármilyen módosítás esetén az adapter frissíti a megjelenített listát.

```
        // Megfigyeljük a könyvek változásait a ViewModel-ben, és frissítjük az UI-t
        bookViewModel.getAllBooks().observe( owner: this) { books ->
            if (books != null) {
                adapter.setBooks(books)
                Log.d( tag: "MainActivity",  msg: "UI frissítve a könyvekkel: $books")
            } else {
                Log.d( tag: "MainActivity",  msg: "Nincs könyv adat")
            }
        }
```

48. ábra Részlet a MainActivity.kt osztályból

- Keresés.

A keresőmező (SearchView) lehetőséget biztosít arra, hogy a felhasználó azonnal szűrje a könyvlistát, miközben gépel.

```
        // Keresőmező beállítása és eseménykezelők
        val searchView: SearchView = findViewById(R.id.search_view)
        searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {
            override fun onQueryTextSubmit(query: String?): Boolean {
                query?.let {
                    Toast.makeText( context: this@MainActivity, "Search: {it}", Toast.LENGTH_SHORT).show()
                }
                return true
            }
```

49. ábra Részlet a MainActivity.kt osztályból

- Könyv törlése.

A RecyclerView elemek jobbra vagy balra húzásával lehetőség van a könyv törlésére. A törlés előtt egy megerősítő dialógus jelenik meg.

```
        // Swipe (balra/jobbra húzás) eseménykezelő a könyvek törlésére
        ItemTouchHelper(object :
            ItemTouchHelper.SimpleCallback( dragDirs: 0,  swipeDirs: ItemTouchHelper.LEFT or ItemTouchHelper.RIGHT) {
            override fun onMove(
                recyclerView: RecyclerView,
                viewHolder: RecyclerView.ViewHolder,
                target: RecyclerView.ViewHolder
            ) = false
```

50. ábra Részlet a MainActivity.kt osztályból

- Könyv hozzáadása és szerkesztése.

Két külön ActivityResultLauncher felelős az új könyv hozzáadásáért (AddBookActivity) és a meglévő könyv frissítéséért (UpdateActivity). A MainActivity felel az adatok fogadásáért és a ViewModel frissítéséért.

- Háttérkép kiválasztása és mentése.

A felhasználó beállíthat egyéni háttérképet, amely az alkalmazás újraindítása után is megmarad. Az adatok tárolása SharedPreferences segítségével történik, a kép betöltése Drawable objektummá konvertálva valósul meg.

```
365     // Háttérkép betöltése SharedPreferences-ből
366     private fun loadSavedBackground() {
367         val prefs = getSharedPreferences( name: "wallpaper_prefs", Context.MODE_PRIVATE)
368         val uriString = prefs.getString( key: "wallpaper_uri",  defValue: null)
369
370         uriString?.let { uriStringValue ->
371             try {
372                 val uri = Uri.parse(uriStringValue)
373                 val inputStream = contentResolver.openInputStream(uri)
374                 val drawable = Drawable.createFromStream(inputStream, uri.toString())
375                 drawable?.let { drawableValue ->
376                     setAppBackground(drawableValue)
377                 }
378             } catch (e: Exception) {
379                 e.printStackTrace()
380             }
381         }
382     }
```

51. ábra loadSavedBackground metódus

- ISBN szkennelés.

A menüből elérhető szkennelési funkció lehetővé teszi az ISBN kód beolvasását, például egy könyv hátlapjáról. A szkennelés eredményét egy BookScanner nevű segédosztály dolgozza fel.

```
238     // Activity eredmények kezelése
239     override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
240         val handled = bookScanner.handleResult(requestCode, resultCode, data) { isbn ->
241             Toast.makeText( context: this, "Successful scan: {isbn}", Toast.LENGTH_SHORT).show()
242             // ISBN alapján lehet keresni, vagy új könyvet hozzáadni
243         }
244
245         if (!handled) {
246             super.onActivityResult(requestCode, resultCode, data)
247         }
248     }
```

52. ábra onActivityResult metódus

- Navigációs logika – visszalépés fragmentből.

A visszagomb kezelése során a rendszer figyeli, hogy van-e visszalépési lehetőség a fragment stack-ben. Ha igen, akkor visszalép az előző fragmentre, ha nem, akkor kilép az Activity-ből.

4.3 Frontend

4.3.1 MainActivity

Ez az XML fájl az alkalmazás főképernyőjének UI-t határozza meg. CoordinatorLayout alapstruktúrára épül, és egy jól strukturált felületet biztosít, amely tartalmaz egy dinamikus RecyclerView listát, vezérlő gombokat, keresőt, toolbar-t és lehetőséget fragmentek megjelenítésére.

Ez az XML layout egy Android főképernyőt definiál, amely vizuálisan jól tagolt, reszponzív, és többféle felhasználói interakcióra ad lehetőséget.

A teljes felületet egy CoordinatorLayout fogja össze. A háttér egy FrameLayout-ban kap helyet, ahol egy háttérkép (@drawable/wallpaper1) jelenik meg.

A képernyő tetején egy Toolbar található egyedi címkével, amely egy TextView segítségével jelenik meg – ez adja meg az alkalmazás vizuális identitását.

A tartalom egy ConstraintLayout-ban helyezkedik el, amelyen belül:

- Van egy SearchView, ahol a felhasználó kereshet a listában.
- Alatta elhelyezkedik egy vízszintes LinearLayout, amely öt darab gombot tartalmaz. Ezek különböző műveleteket kínálnak (pl. szűrés, rendezés, nézetváltás, új elem hozzáadása, háttérkép váltás).
- Egy rejtett FrameLayout szolgál fragmentek fogadására, ha dinamikusan szeretnénk tartalmat cserélni a képernyőn.
- Egy RecyclerView tölti ki az alsó szekciót, amely a könyvek listázásáért felel – ehhez tartozik egy külön item layout (@layout/book).

4.3.2 Könyv listaelem

Ez a komponens egy CardView-ra épül, ami Material Design-os kártya megjelenítést biztosít. A kártya enyhe árnyékot (cardElevation) és lekerekített sarkokat (cardCornerRadius) kapott, ezzel is kiemelve a többi elem közül.

A tartalom egy függőleges LinearLayout-ban helyezkedik el, amely két szöveges mezőt (TextView) jelenít meg.

Ez a két mező a következő:

- az egyik a könyv címét,
- a másik a szerző nevét tartalmazza.

A két mező különböző betűmérettel és stílussal van formázva, így vizuálisan jól elkülönülnek egymástól, és az információ könnyen áttekinthető. A padding és margin értékek biztosítják az átlátható elrendezést.

4.3.3 Szűrés

Ez a layout egy komplex szűrőfelületet valósít meg, amely lehetővé teszi a felhasználók számára, hogy többféle szempont alapján szűrjék a könyveket. Egy görgethető ScrollView-ban helyezkedik el, így nagy mennyiségű tartalom is kényelmesen kezelhető.

Ez az oldal a következő szűrési lehetőségeket biztosítja:

- Nyelv szerint: rádiógombok segítségével lehet kiválasztani, hogy milyen nyelvű könyveket szeretnénk (pl. magyar, angol, német stb.).
- Kategóriák szerint: checkbox-okkal választható több kategória is, mint pl. pszichológia, történelem, tudomány, hobbi stb.
- Állapot szerint: új vagy használt könyvek szűrése.
- Olvasási státusz szerint: elolvasott, olvasás alatt vagy még nem olvasott könyvek szűrése.
- Tulajdonos szerint: hogy kinél van éppen a könyv (pl. "M", "D").
- Típus szerint: e-book vagy nyomtatott könyv.

Az alján két gomb található:

- Mégse (Cancel) – kilépés a szűrőből változtatás nélkül.
- Mentés (Save) – a kiválasztott szűrők alkalmazása.

4.3.4 Rendezés

Ez a második layout XML egy rendezési beállításokat kínáló képernyő, amely vizuálisan és funkcionálisan is illeszkedik az előzőleg bemutatott szűrőfelülethez. Itt kizárólag rendezési logikák választhatók, hogy a könyvlista megjelenítésének sorrendjét lehessen befolyásolni.

A felhasználó az alábbi szempontok szerint választhat rendezési sorrendet:

Cím szerint:

- A → Z
- Z → A

Szerző szerint:

- A → Z
- Z → A

Megjelenési dátum szerint:

- Növekvő
- Csökkenő

Mindez egyetlen RadioGroup-ba van szervezve, ami biztosítja, hogy egyszerre csak egy opció legyen kiválasztva.

Megjelenés és használat:
A felület tetején szerepel egy cím, amely egyértelművé teszi a képernyő célját.

A gombok alul helyezkednek el.

4.3.5 Megjelnítés

Ez a layout egy könyvlista megjelenítő képernyőt ír le, amely kétféle nézetmódot támogat: lista és rács (grid) formátumot.

Funkció:

- Nézetváltás – RadioGroup

Felül két rádiógombból álló sáv:

- Lista nézet
- Rács nézet

Ez lehetőséget ad a felhasználónak, hogy kedve szerint váltogasson a megjelenési formák között.

Könyvlista:
Az alsó rész egy RecyclerView, amely a könyvek megjelenítéséért felel. A

tools:listitem="@layout/book" jelzi, hogy a listaelemekhez egy book.xml layout sablon van rendelve.

4.3.6 Háttér beállítás

Ez a layout egy háttérkép kiválasztó felületet valósít meg, ahol a felhasználó képet választhat az alkalmazás háttérképeként, majd elmentheti vagy visszavonhatja a választását.

Fő funkciók:

- Kép kiválasztása gomb:

Felül, középre igazítva helyezkedik el.

Címkéje: "select background" (@string/select_background)

Megnyomására képgaléria nyílik meg a kép kiválasztásához.

- Mentés gomb:

Csak akkor látható (visibility="gone" alapból), ha ki lett választva egy kép.

Jobb oldalon, a törlés mellett helyezkedik el.

Címkéje: "save"

- Mégse gomb:

Szintén csak kiválasztás után jelenik meg.

Balra a mentés gomb mellett helyezkedik el.

Címkéje: "cancel"

UI jellemzők:

- ConstraintLayout:

Rugalmas elhelyezés, könnyen adaptálható különböző képernyőméretekhez.

- Kerekített stílus:

Minden gomb a @drawable/rounded_button háttérrel rendelkezik, ami vizuálisan egységes

és modern megjelenést kölcsönöz.

4.3.7 Könyv hozzáadása

Ez az XML fájl egy új könyv hozzáadására szolgáló űrlapot ír le az alkalmazásban, amelyet egy teljesen görgethető felületként valósítottam meg. A célja, hogy a felhasználó egyszerűen hozzáadhasson új könyvet a rendszerhez, megadva a szükséges metaadatokat (pl. cím, szerző, nyelv, kép stb.). A felület intuitív, űrlapszerű elrendezésre épül.

AddBookActivity áttekintése:
A teljes tartalom egy ScrollView-ba van ágyazva, így kis képernyőn is gördíthető marad.

A vizuális háttérként egy FrameLayout szolgál, amely mögé egy háttérkép (@drawable/wallpaper1) van beállítva.

Az űrlapelemeket egy függőleges LinearLayout-on belül helyezkednek el, ami logikus és könnyen áttekinthető sorrendet biztosít.

5. A fejlesztés menete

A dolgozat tárgyát képező alkalmazás fejlesztését egy sor tervezési és implementációs lépés előzte meg, amelyek célja a felhasználói és saját igényekhez illeszkedő mobilalkalmazás elkészítése volt. Az alkalmazás fejlesztése a specifikációk meghatározásával kezdődött, majd a különböző tervezési szakaszok és implementációs lépések következtek.
A fejlesztés menetében és a hibakeresésekben nagy hasznomra voltak a Felhasznált forrásnál megjelölt linkek, illetve a ChatGPT is.

5.1 Fejlesztési specifikáció

A fejlesztés során az első lépés a specifikációk meghatározása volt. Az alkalmazás célja, hogy egy személyes könyvgyűjtemény kezelésére szolgáló eszközt biztosítson, amely lehetővé teszi a könyvek hozzáadását, kategorizálását és megjelenítését. A specifikációk az alábbiakban foglalhatók össze:

- Alapvető funkciók:

Könyv hozzáadása ISBN kód alapján, és manuálisan, könyvinformációk lekérése az API-ból, könyvek kategorizálása, könyvek keresése, könyvek szűrése és rendezése.

- Felhasználói élmény:

Testre szabható vizuális megjelenés, kategóriák és könyvgyűjtemények rendszerezése.

- Platform:

Android operációs rendszer, esetlegesen későbbiekben iOS platformra való portolás terve.

- Technológia:

Kotlin programozási nyelv, Android Studio, SQLite adatbázis (Room), Google Books API.

5.2 Tervezési fázis

A fejlesztés második lépése a rendszerterv és az adatbázis-struktúra megtervezése volt.

5.2.1 Rendszerterv

A rendszerterv a következő főbb komponenseket tartalmazza:

- Felhasználói felület (UI):

Az alkalmazás felhasználói felülete egyszerű és intuitív, lehetővé téve a felhasználók számára, hogy könnyedén hozzáadhassanak, szerkeszthessenek és kezelhessenek könyveket, keresgélhessenek a gyűjteményükben és testre szabják az alkalmazás vizuális megjelenését.

- Backend logika:

Az alkalmazás a könyvhozzáadás során az ISBN kódot felhasználva kapcsolatba lép a Google Books API-val, és az API által visszaadott adatokat felhasználja a könyv információinak megjelenítésére. Amennyiben ezt manuálisan tesszük, úgy ez a rész kimarad. Duplikáció nem megengedett, így minden könyv csak egyszer szerepelhet az adatbázisban. Nem csak hozzáadni tudunk könyvet, de szerkeszteni, illetve törölni is.

- Rendszerezés:

Az alkalmazásban a könyvek különböző kategóriákba sorolhatók, például műfajok, nyelvek, állapot szerint (például elolvasott, kölcsönzött), így segítve a felhasználókat a könyveik rendszerezésében.

5.3 Adatbázis-séma

Az alkalmazás adatbázisát SQLite használatával készítettem el, mivel az ideális választás volt egy egyszerű mobilalkalmazás számára, és egy book nevű táblát tartalmaz.

- id (INT, PRIMARY KEY): egyedi azonosító.
- title (TEXT): könyv címe, kötelező.
- author (TEXT): könyv szerzője, kötelező.
- date (TEXT): könyv kiadásának éve.
- image (TEXT): a könyv borítójának képe.
- category (TEXT): könyv kategóriája.
- language (TEXT): a könyv nyelve.

- type (TEXT): a könyv típusa.
- location (TEXT): a könyv helye.
- condition (TEXT): a könyv állapota (pl. új, használt).
- status (TEXT): könyv státusza (pl. elolvasott, kölcsönzött).
- isbn (TEXT): a könyv ISBN kódja.

5.4 Képernyők

A felhasználói interakciók megtervezése során az alábbi képernyőket készítettem:

- Főoldal:

Itt jelennek meg a hozzáadott könyvek és a funkciók, mint például a szűrés, rendezés, könyv hozzáadása és keresés.

- Könyv keresése:

Egy egyszerű kereső, amely lehetővé teszi a felhasználó számára, hogy a könyveit cím vagy szerző alapján keresse meg.

- Könyv szűrése:

Ezen az oldalon tud a felhasználó elvégezni különböző paraméterek alapján szűréseket.

- Könyv rendezése:

Itt tudja a felhasználó különböző paraméterek alapján rendezni a könyveit.

- Könyv megjelenítés:

Ezen a képernyőn van lehetőség arra, hogy a felhasználó kiválassza a könyvlista nézetét.

- Könyv hozzáadása:

Egy űrlap, ahol a felhasználó mentheti a könyv adatokat, mint a cím, szerző, kiadás éve stb. Az alkalmazás ezeket az adatokat jeleníti meg, és lehetőséget ad a felhasználónak a könyv kategorizálására.

- Könyv szerkesztése:

Az egyes könyvek adatainak módosítását lehetővé tevő űrlap, ahol a felhasználó megváltoztathatja a könyv adatait, például a státuszát vagy a kategóriáját. Eredetileg külön Activityben terveztem megoldani a könyvek szerkesztését, de menet közben rájöttem, hogy javarészt csak duplikációja lenne a könyv hozzáadás Activity-nek, és ezt szerettem volna elkerülni, ezért végül úgy döntöttem, hogy az új könyv hozzáadása, valamint a meglévő szerkesztése is ugyanazon az Activity-n történik.

- Háttér beállítás:

Lehetősége van a felhasználónak arra, hogy saját galériából állítson be háttérképet az alkalmazásnak.
Először arra gondoltam, hogy megadok pár előre definiált képet, amik közül lehet választani, de mikor a fejlesztésben oda jutottam, átgondoltam, és jobbnak láttam, ha inkább a készülék galériájához biztosítok hozzáférést, így mindenki, aki használja az appot a kedve szerint tudja módosítani a hátteret, megkötések nélkül.

5.5 Fejlesztés és implementáció

A szakdolgozat keretében elkészült mobilalkalmazás célja egy olyan könyvnyilvántartó rendszer létrehozása volt, amely lehetővé teszi a felhasználó számára saját könyvgyűjteményének hatékony kezelését. A fejlesztés során elsődleges szempont volt a használhatóság, az áttekinthetőség, valamint az, hogy az alkalmazás személyre szabható legyen.
A fejlesztés első lépése a funkcionális követelmények meghatározása volt, amely során meghatároztam azokat az alapvető funkciókat, amelyeket az alkalmazásnak feltétlenül tudnia kell. Ide tartozott a könyvek felvétele, szerkesztése, törlése, kategorizálása, valamint egy keresési lehetőség biztosítása. Ezen kívül fontosnak tartottam, hogy az alkalmazás támogassa az ISBN-alapú könyvfelismerést is, melyhez a Google Books API-t vettem igénybe.
A tervezési fázisban elkészült a rendszer felépítésének logikai vázlata, beleértve az adatbázis-sémát is. Az alkalmazás SQLite adatbázist használ, ahol minden könyv külön rekordként szerepel a hozzá tartozó adatokkal (cím, szerző, kiadás éve, kategória stb.). A felhasználói felületek (űrlapok, listák, részletező nézetek) vázlatait előre megterveztem, ezek segítették a fejlesztés következetes és átlátható lebonyolítását.

A fejlesztés során kotlin nyelvet használtam, az Android Studio fejlesztőkörnyezetben. A képernyőváltások és különböző felhasználói felületek kezelésére fragmenteket alkalmaztam. Az egyik technikai kihívás az volt, hogy a fragmentek helyes megjelenítését és az azok közötti navigációt megfelelően kezeljem. A fejlesztés elején előfordult, hogy a fragment tartalma alatt jelent meg a könyvek listája, ami zavarta a felhasználói élményt, ezért ezen a részen jelentősebb módosításokat kellett végeznem.
Egy másik kulcsfontosságú döntés a könyvszerkesztő felület megvalósítása kapcsán született. Eredetileg egy külön Activity-ben szerettem volna kezelni a szerkesztést, azonban a fejlesztés előrehaladtával világossá vált, hogy ezzel feleslegesen duplikálnám az AddBookActivity tartalmát. Ezért a szerkesztést ugyanazon képernyő újrafelhasználásával oldottam meg.
A megjelenés testreszabhatóságát is kiemelten fontosnak tartottam. Eredetileg előre definiált háttérképek közül lehetett választani, de végül lehetővé tettem, hogy a felhasználó saját képet választhasson a galériájából, így jobban igazodik az egyéni ízléshez.
Az egyik legnagyobb kihívás a Google Books API használatából fakadt. Bár az API lehetővé teszi a könyvek adatainak lekérését ISBN alapján, magyar nyelvű könyvek esetén jelentős hiányosságokat tapasztaltam. Sok esetben nem sikerült lekérni a könyv adatait, ezért is hasznos, hogy manuális adatbeviteli lehetőséget is biztosítottam. Ez a jövőbeni fejlesztések egyik kiemelt területe lehet, akár egy alternatív adatforrás integrálásával, akár saját adatbázis építésével.

Az adatbázis szerkezetét, a repositoryt, a view modelt és az adaptert javarészt egy Udemy-n található kurzusban bemutatott módon oldottam meg.[vii] Ezen kívül utána olvastam a hivatalos dokumentációban.[viii] [ix] [x]

A Google Books API-val kapcsolatban is elolvastam a hivatalos dokumentációt, de itt igénybe vettem a ChatGPT segítségét is, mivel beleütköztem néhány nehézségbe.[xi] [xii] [xiii] [xiv]

A MainActivity, AddBookActivity és UpdateActivity bizonyos részeihez is igénybe vettem a ChatGPT-t, és az Udemy-s tanfolyamot, amit igyekeztem pontosan követni, mivel megbízhatónak tartom. De természetesen belefutottam jónéhány hibába, amik kijavításához hasznos eszköznek bizonyult a ChatGPT.

A fragmentek kialakításában szintén az Udemy-s tanfolyam volt nagy hasznomra.

A BookScanner osztály létrehozása okozott némi fejtörést, de szerencsére találtam egy weboldalt, amit tudtam használni és itt is igénybe vettem helyenként a ChatGPT ötleteit. [xv]

A BackgroundSelectorFragment osztálynál ChatGPT segített elindulni és az elakadásaimnál is számíthattam nagyrészt a hibamegoldási javaslataira, amiket végül sikerült megoldanom.

Végül szeretném megemlíteni az AndroidManifest.xml-hez használt forrásaimat.[xvi] [xvii] [xviii] [xix] [xx] [xxi] [xxii]

A fejlesztés során törekedtem a karbantartható, jól strukturált kód létrehozására. Az alkalmazás elsődlegesen saját használatra készült, így a kategóriák, nyelvek, és egyéb paraméterek is az én könyvtáramhoz lettek igazítva. Ugyanakkor az architektúra lehetővé teszi a későbbi bővítést és személyre szabást más felhasználók számára is.

5.6 Tesztelés és hibajavítás

A fejlesztési szakasz végén az alkalmazást alaposan teszteltem, hogy biztosítsam a különböző funkciók megfelelő működését. A hibák javítása, a felhasználói felület finomhangolása és az adatbázis optimalizálása után az alkalmazás már teljes mértékben működőképes volt.

6. Összegzés

A dolgozatban bemutatott alkalmazás célja, hogy segítséget nyújtson a személyes könyvgyűjtemények kezelésében, lehetővé téve a könyvek egyszerű nyilvántartását, kategorizálását és megjelenítését. A fejlesztés központi célja az volt, hogy a felhasználó elkerülje a már meglévő könyvek újbóli megvásárlását, és átlátható módon rendszerezhesse saját könyvgyűjteményét. Az alkalmazás fő funkciói közé tartozik a könyvek hozzáadása ISBN kód alapján, a könyvinformációk lekérése a Google Books API-ból, valamint a könyvek különböző kategóriákba sorolása és keresése. A felhasználói élmény érdekében a megjelenés testre szabható, a háttérképek szabadon módosíthatók, így a felhasználó saját ízléséhez igazíthatja az alkalmazást.

A fejlesztés során a kotlin programozási nyelvet és az Android Studio fejlesztői környezetet használtam. Az alkalmazás adatbázis-kezelésére SQLite-t alkalmaztam, amely lehetővé teszi a könyvgyűjtemények tárolását és kezelését. Az ISBN alapján történő könyvhozzáadás során a Google Books API-t használtam a könyvadatok automatikus lekérésére. Az alkalmazásban beépített keresőfunkciók is szerepelnek, amelyek a felhasználó számára gyors keresést és kategorizálást biztosítanak, így a könyvek gyorsan megtalálhatók és könnyedén hozzáadhatók.

Bár az alkalmazás alapvető funkciói teljes mértékben megvalósultak, a fejlesztés során egy jelentős problémával találkoztam. A Google Books API nem tartalmazza a magyar nyelven megjelent könyveket, vagy legalábbis az én tapasztalataim szerint nem képes felismerni a magyar ISBN kódokat. Ezért az alkalmazásban a könyvek jelentős részét manuálisan kell hozzáadni az adatbázishoz, mivel az API nem képes a magyar könyveket automatikusan azonosítani. Ez a korlátozás fontos kihívást jelent, és bár az alkalmazás jelenleg is használható, hosszú távon szeretném megtalálni a megfelelő megoldást a magyar nyelvű könyvek kezelésére. Alternatív API-k vagy saját adatbázis kialakítása lehetőséget adhat arra, hogy a magyar piacra készült könyveket is támogassa az alkalmazás, habár ilyen API-t azóta sem sikerült találnom.

A jövőben a legfontosabb fejlesztési irányok közé tartozik a könyvadatbázis bővítése és frissítése, hogy az alkalmazás minél több könyvet és információt tartalmazzon. Továbbá, szeretném lecserélni azokat a könyvtárakat, amelyek már elavultak, és biztosítani, hogy az alkalmazás naprakész legyen minden technológiai szempontból. A felhasználói visszajelzések alapján, amennyiben igény lesz rá, további funkciók beépítését is elvégzem

majd. Mivel az alkalmazás saját használatra készült, az egyes funkciók és beállítások az én személyes könyvtáramhoz igazodnak, de az univerzális felhasználás érdekében a későbbiekben szeretném az alkalmazást további funkciókkal és kategóriákkal bővíteni.

Az alkalmazás fejlesztése során egy másik fontos jövőbeli lehetőség az iOS platformra történő fejlesztés. Jelenleg az alkalmazás csak Android operációs rendszeren érhető el, de a jövőben, ha a felhasználói igények indokolják, akkor az iOS verzió elkészítése is fontos lépés lehet. Ezzel új technológiai kihívásokkal is szembesülhetnék, amelyek lehetőséget adnak arra, hogy új ismeretekre tegyek szert és fejlesszem a programozási készségeimet.

Az alkalmazás kategóriái, nyelvei és beállításai az én gyűjteményemhez illeszkednek, azonban a jövőbeli tervek között szerepel az alkalmazás univerzálisabbá tétele. Ez magában foglalja az új kategóriák hozzáadását, az adatbázis struktúrájának fejlesztését és a felhasználói élmény javítását. A célom, hogy az alkalmazás olyan széleskörűen alkalmazható és könnyen testre szabható legyen, hogy megfelelő módon rendszerezhessem a gyűjteményem, akár különböző nyelveken, akár különböző könyvkategóriák szerint.

7. Felhasznált források

Itt sorolom fel azokat a forrásokat, amik közelebb vittek egyes hibák kijavításához, illetve elláttak az alkalmazásom elkészítéséhez szükséges tudással.

https://developer.android.com/develop/ui/views/components/radiobutton

https://www.youtube.com/watch?v=N9-6Feu93lc&ab_channel=MaskedProgrammer

https://developer.android.com/jetpack/androidx/releases/lifecycle

https://medium.com/@pritam.karmahapatra/retrofit-in-android-with-kotlin-9af9f66a54a8

https://developer.android.com/jetpack/androidx/releases/recyclerview

https://www.youtube.com/watch?v=SkH1lMnu-TI&ab_channel=IJApps

https://developer.android.com/build/migrate-to-ksp

https://developer.android.com/media/camera/camera-intents

https://developer.android.com/reference/androidx/core/content/FileProvider

https://www.youtube.com/watch?v=QpxOw_h10rQ&ab_channel=AvinashPrasad

https://www.geeksforgeeks.org/difference-between-mvc-mvp-and-mvvm-architecture-pattern-in-android/

https://stackoverflow.com/questions/2487145/difference-between-the-repository-pattern-and-the-view-model-pattern

https://stackoverflow.com/questions/50285503/what-is-the-difference-between-adapters-such-as-recyclerviewadapter-or-listadapt

Itt azokat a forrásokat sorolom fel, amikre valamilyen formában hivatkozok a kódomban, illetve ebben a dokumentációban.

[i] https://developer.android.com/get-started/overview

[ii] https://developer.android.com/kotlin

[iii] https://developer.android.com/training/data-storage/room

[iv] https://rachelaemmer.medium.com/how-to-use-the-google-books-api-in-your-application-17a0ed7fa857

[v] https://github.com/zxing/zxing

[vi] https://hu.wikipedia.org/wiki/ChatGPT

[vii]https://www.udemy.com/course/android-development-android-app-developer-course-with-pie/

[viii] https://developer.android.com/training/data-storage/room

[ix] https://developer.android.com/topic/libraries/architecture/viewmodel

[x]https://www.geeksforgeeks.org/how-to-perform-crud-operations-in-room-database-in-android/

[xi] https://chatgpt.com/

[xii] https://developers.google.com/books/docs/v1/getting_started

[xiii] https://developers.google.com/books/docs/v1/reference/volumes/get

[xiv] https://developers.google.com/books/docs/v1/using

[xv]https://blair49.medium.com/how-to-add-qr-code-generation-and-scanning-in-your-android-app-7baff15bd7c6

[xvi] https://developer.android.com/guide/topics/manifest/uses-permission-element

[xvii] https://developer.android.com/guide/topics/manifest/uses-feature-element

[xviii] https://developer.android.com/guide/topics/manifest/queries-element

[xix] https://developer.android.com/reference/android/Manifest.permission

[xx]https://stackoverflow.com/questions/37832604/androidrequired-false-is-being-ignored-in-uses-feature-tag

[xxi] https://stackoverflow.com/questions/62782648/android-11-scoped-storage-permissions

[xxii] https://developer.android.com/training/data-storage/manage-all-files