ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS

INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK	Hallgató neve:	**Nagy Bálint**
2025	Hallgató törzskönyvi száma:	**T/008665/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Szoftvertervezés és -fejlesztés Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Nagy Bálint**
Törzskönyvi száma: T/008665/EI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

Kártya katalógus digitalizáló android alkalmazás képfelismerő algoritmusok és felhő szolgáltatások alkalmazásával
Card catalog digitizing android application using image recognition algorithms and cloud services

Intézményi konzulens: Sipos Miklós László
Külső konzulens:

Beadási határidő: 2023. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
Felhőszolgáltatási technológiák és IT biztonság specializáció

A feladat

A feladat egy olyan mobil alkalmazás készítése, mely segítségével online egyszerűen rendszerezhető és listázható Magic: The Gathering kollekciót tudunk felhőben tárolni. Nagy kollekciókkal rendelkező gyűjtőknek nehéz nyomon követni a lapok értékét és fejben tartani esetleg, milyen lapjait cserélné mással. Az applikáció legyen képes valós időben frissíteni az árait a kártyáknak, ezáltal biztosítani a naprakész adatokat. A kamerát használva az applikáción belül legyen képes beszkennelni a kártyákat és megfelelő verzióikat ezzel felgyorsítva a manuális bevitelt. Vizsgálja meg milyen egyéb funkciók lehetnek hasznosak a felhasználónak (adatok felhőben való tárolása, egy lista exportálása más szoftverek formátumába). Az exportálási funkcióval kész paklikat tudjon a felhasználó a játék legnépszerűbb digitális verziójába a Magic: The Gathering Arena-ba importálni. Végezzen továbbá elemzést a tesztelési lehetőségekről és készítse is el az egyes teszt típusokat és teszteseteket.

A dolgozatnak tartalmaznia kell:

- a megoldandó feladat pontos leírását és részletes elemzését,
- magas szintű szerkezeti mintát az alkalmazásról,
- az egyes funkciók megvalósításához felhasználható lehetséges technológiákat,
- az elkészített rendszer tervét, blokkdiagramját,
- a megvalósítás menetét, felmerülő problémákat és az azokra adott megoldást,
- a tesztelési tervet, a tesztelés módszertanát és a tesztelés eredményeit,
- az elkészült szoftver felhasználási lehetőségeit,
- az eredmények bemutatását és értékelését a teljes munkára vonatkozóan.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * 1. Budapest

Ph.

.....
Dr. Vámossy Zoltán
intézetigazgató

A szakdolgozat elévülésének határideje: **2025. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

.....
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

HALLGATÓI NYILATKOZAT

Alulírott Nagy Bálint [redacted] hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, ÉÉÉÉ. hónap nap.

.....

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: NAGY BÁLINT

Neptun Kód: [redacted]

Tagozat: NAPPALI TAGOZAT

Telefon: [redacted]

Levelezési cím (pl.: lakcím): [redacted]

Szakdolgozat címe magyarul:

KÁRTYA KATALÓGUS DIGITALIZÁLÓ ANDROID ALKALMAZÁS KÉPFELISMERŐ ALGORITMUSOK ÉS FELHŐ SZOLGÁLTATÁSOK ALKALMAZÁSÁVAL

Szakdolgozat címe angolul:

CARD CATALOG DIGITIZING ANDROID APPLICATION USING IMAGE RECOGNITION ALGORITHMS AND CLOUD SERVICES

Intézményi konzulens: SIPOS MIKLÓS LÁSZLÓ

Külső konzulens: -

Alk.	Dátum	Tartalom	Aláírás
1.	2023.03.06.	Félév teendőinek, menetrendjének és mérföldköveinek ismertetése.	
2.	2023.03.20.	Hasonló rendszerek elemzése.	
3.	2023.04.10.	Irodalomkutatás, annak összegzése valamint konzekvenciák meghatározása.	
4.	2023.05.08.	Tervezés, követelmények specifikációja. Véglegesítés. Leadással kapcsolatos teendők egyeztetése.	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. tantárgy követelményét teljesítette, beszámolóra bocsátható.

.....

Intézményi konzulens

Budapest, 2023. 05. 08.

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: NAGY BÁLINT

Neptun Kód: [redacted]

Tagozat: NAPPALI TAGOZAT

Telefon: [redacted]

Levelezési cím (pl.: lakcím): [redacted]

Szakdolgozat címe magyarul:

KÁRTYA KATALÓGUS DIGITALIZÁLÓ ANDROID ALKALMAZÁS KÉPFELISMERŐ ALGORITMUSOK ÉS FELHŐ SZOLGÁLTATÁSOK ALKALMAZÁSÁVAL

Szakdolgozat címe angolul:

CARD CATALOG DIGITIZING ANDROID APPLICATION USING IMAGE RECOGNITION ALGORITHMS AND CLOUD SERVICES

Intézményi konzulens: SIPOS MIKLÓS LÁSZLÓ

Külső konzulens: -

Alk.	Dátum	Tartalom	Aláírás
1.	2024.09.29.	Fejlesztés tervezett menetének revíziója, szakmai egyeztetés, fejlesztési teendők megkezdése.	
2.	2024.10.20.	Fejlesztés állapotának és haladásának áttekintése. Technikai problémák és módosítások egyeztetése.	
3.	2024.11.17.	Felmerülő problémák egyeztetése és megoldása. Tesztelési fázis megindítása.	
4.	2024.12.01.	Dokumentáció véglegesítése. Leadással kapcsolatos teendők egyeztetése.	

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat II. tantárgy követelményét teljesítette, védésre bocsájtható.

A konzulens által javasolt érdemjegy: 5

Sipos S.

.....

Intézményi konzulens

Budapest, 2024. 12. 06.

Tartalomjegyzék

ABSZTRAKT

Szakdolgozatom célja egy Magic: The Gathering kártyák digitalizálásához használható Androidos alkalmazás megtervezése és megvalósítása. A Magic: The Gathering egy világhírű gyűjtögetős kártyajáték, melynek során kettő vagy több játékos próbálja kiejteni ellenfeleit különleges lapok segítségével. Az alkalmazás feladata a játékosok kártyáinak beszkennelése, ezzel a kollekciójának digitalizálása, megjelenítése. A digitalizált kollekció nagyban megkönnyíti a játékosok közötti cserék lebonyolítását verseny és más rendezvények alatt.

Előzetesen alapos kutatást végeztem a már elérhető Magic: The Gathering kollekció digitalizáló platformokról és alkalmazásokról. Megvizsgáltam ezek előnyeit, hátrányait és mérlegeltem az implementálható lehetőségek között.

Ezután a tervezés következett, ahol meghatároztam az elkészítendő modulokat és elemeket. Ezt a legfontosabb modullal kezdtem, a kártya szkennerrel, majd eljutottam a főalkalmazás kisebb elemeihez.

ABSTRACT

The aim of my thesis is to design and implement an android application for digitizing Magic: The Gathering cards. Magic: The Gathering is a world-famous collectible card game, where two or more players try to eliminate their opponents using special cards. The application can scan the cards of the players, digitizing and displaying their collection. The digitized collection greatly facilitates the exchange of cards between players in tournaments.

Firstly, I did thorough research on the already available Magic: The Gathering collection digitizing platforms and applications. I examined their advantages and disadvantages and weighed up the options that could be implemented.

Next came the design phase, where I determined the modules and elements to be created. I started this with the most important module, the card scanner, and then moved on to the smaller elements of the main application.

1. BEVEZETÉS

A Magic: The Gathering az első TCG (Trading Card Game), magyarul gyűjtögetős kártyajáték a világon. 1993-as megjelenése óta több mint 80.000 különböző kártyát adtak ki és világszerte népszerű mind versenyszinten, mind hobbiként. Idővel megjelent ennek a digitális változata is 2002-ben, valamint egy újabb, felhasználóbarátabb verzió 2018-ban.

A játék során 2 vagy több játékos körökre osztva próbálja meg minden ellenfelének életerőpontjait 0-ra redukálni, ezzel megnyerve a játékot. Bizonyos lapok direktben tudják csökkenteni ezt, míg más lapok erőforrásként szolgálnak a fontosabbak kijátszásához.

Kártya lapokat 15 lapos kártyacsomagokban lehet venni, amiben véletlenszerűen kapunk bizonyos ritkaságú lapokat. Egy ilyen csomagban 10 gyakori lap van, viszont csak 1 ritka, aminek 1/8 esélye van, hogy mitikus ritka legyen. Természetesen a ritkább lapok általában az értékesebbek, mivel ezek erősebbek, mint a gyakoribb lapok. A kártyák ára ennek hatására nagyon különböző: a legolcsóbb lapok értéktelenek, míg a legdrágább lap nemrég 540.000 dollárért lett elárverezve. [1]

Világszerte több millióan játsszák a játékot, így a piaca is ezeknek a lapoknak nagy. Egy átlagos játékosnak több ezer lapja is lehet. Ezeknek az ára rendszeresen változik, attól függően melyik lapok teljesítenek a legjobban versenyeken. Egy jól átlátható rendszer nélkül nagyon nehéz követni milyen lapok vannak meg, melyek kellenek még a gyűjteménybe.

1. ábra Magic kollekció:
Értékesebb Magic: The Gathering kártyák tárolása ilyen mappákban történik általában.

1.1. A cél

A cél egy olyan alkalmazás elkészítése, ami a kamerát használva egyszerűen beszkenneli és digitalizálja a kártyákat. A szkennelés során meghatározza az alkalmazás a kártya megfelelő verzióját, tehát melyik kiadásból van és rendelkezik-e alternatív grafikával. Egy publikus adatbázisból kikeresve a pontos lapot, elmenti naprakész adatokkal a digitális kollekcióba. Ezután az alkalmazás felületén kijelzi ezeknek a képekként megjelenő kártyáknak az árát és más karakterisztikáját, ami fontos lehet egy felhasználónak. A kártyákat listákba lehet rendezni tetszés szerint ár, név és típus alapján. A felhasználó paklikat is tud készíteni a lapjaiból, amit exportálni tud más-más formátumba, akár a játék digitális verziójába vagy más embernek elküldeni cserélgetés szempontjából.

2. IRODALOMKUTATÁS

Ez a fejezet során pár mobiltelefonos alkalmazást és weboldalt mutatnék be részletesen. Ezek a felületek kifejezetten erre a játékra lettek létrehozva, hogy egyszerűen lehessen digitalizálni fizikai kollekcióikat a gyűjtőknek és játékosoknak.

2.1. Hasonló rendszerek

A következő rendszereket a következő szempontok alapján fogom megvizsgálni:

- Kártya kollekció digitalizálásának módja
- Kártyák rendezése a rendszeren belül
- Hasznos információk a felhasználónak
- Felhasználóbarát felület és hátrányok

2.1.1. Digitális módszer mellőzése

Ez a megoldást csak azért mutatnám be, hogy jobban lehessen látni a többi módszer előnyeit. Mivel maguknak a fizikai kártyáknak van ára és jelen helyzetünkben ezeket is gyűjtjük így ezek rendszerezése nélkülözhetetlen. Sajnos a játék nem kínál lehetőséget arra, hogy ezeket egytől egyig importáljuk az online verzióba, ott vagy napi játékkal szép lassan lehet megszerezni bizonyos lapokat, vagy online vásárlással.

A játékos reális fizikai kollekcióját érdemes rendben tartani attól függetlenül melyik alternatívát használja online katalogizálásra. Több ezer lapnál, ami egy teljesen általános mennyiség egy játékos számára, nem egyszerű ezt rendszerezve tartani. Ehhez lehet vásárolni dobozokat vagy bármi mást újra felhasználni például cipősdobozokat.

Ahhoz, hogy ne vesszen el az ember teljesen, sok alkategória alapján be lehet rendszerezni a lapokat. Első sorban a lapok színe alapján ajánlom tapasztalatból és ehhez kapcsolódó cikkek alapján. Azután ritkaság alapján érdemes új részlegeket kialakítani, majd ABC sorrendbe rendezni a maradékot. Ez alapján más kiadások ugyanolyan lapjai nem fognak teljesen külön helyen elhelyezkedni, mely bezavarhat keresgélés közben. [2]

Amikor elkészült a rendezett kollekcióm rájöttem, hogy Magic: The Gathering versenyeknél és más rendezvényeknél nem nyújt segítséget egy mobilos verzió nélkül. Többször is, amikor láttam egy lapot, ami érdekelt már egy jó ideje, nem tudtam eldönteni, hogy megszereztem-e azóta vagy sem. Emiatt több cseréről és számomra érdekes lapokról csúsztam le és ekkor kerestem fel egy lehetőséget kollekcióm online követésére.

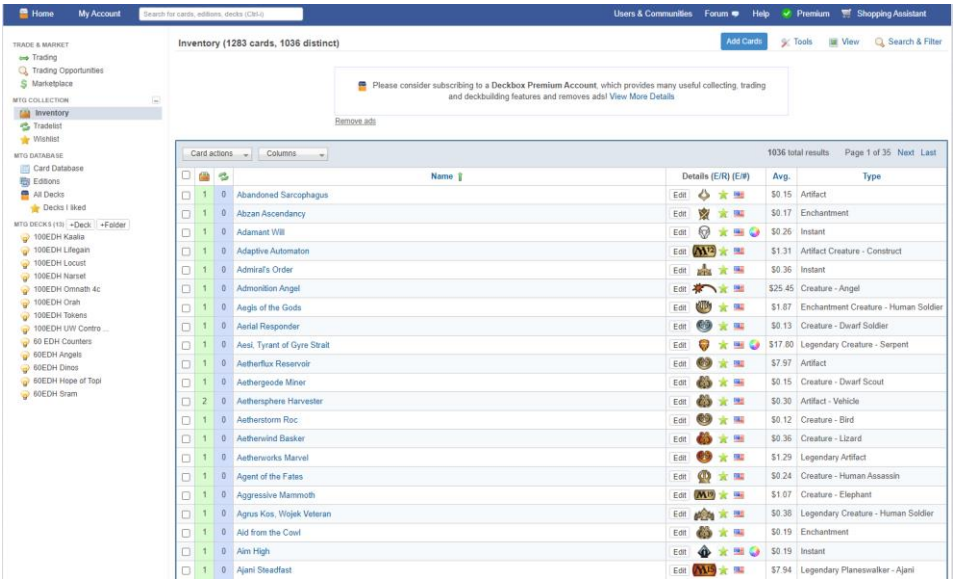
2.1.2. Deckbox weboldal

Legelső rendszer, amit bemutatnék az úgynevezett Deckbox weboldal. Ez a felület üzemel a legrégebb óta a három közül, 2008-tól volt elérhető. Mivel ez egy weboldal, így nagyban eltér a másik két tárgyalt alkalmazástól. Regisztrálás szükséges a felület használatához, ezután tudunk saját kollekciót tárolni az oldalon.

Lapjainkat itt egy online kereső segítségével tudjuk egyenként kikeresni, kiválasztani a megfelelő verziót és elmenteni azt a gyűjteményünkbe. Nincs lehetőség kamerával laponként gyorsan bevinni az adatokat. Importálni és exportálni lehet más alkalmazásokból, amik megkönnyítik és felgyorsítják nagyban a kollekciók felvitelét. Importálni Excelből és más adatbáziskezelő alkalmazásokból lehet CSV (Comma Separated Values – pontosvesszővel tagolt értékek) formátumú fájlokat. Legegyszerűbben ilyen adatbázisokat a másik 2 alternatíva tud produkálni.

A lapjainkat digitalizálás után a rendszer egy készletbe helyezi az összes eddigi lap mellé. Ezeket tudjuk szortírozni név, kiadás, ár és típus alapján csökkenő és növekvő módon. Feltűnteti ezenfelül az oldal, hogy egy ugyanolyan lapból hány darab van a gyűjteményen belül, így ezeknek nem kapnak külön sort. Viszont, ha két lap teljesen ugyanaz, viszont az egyik egy ritkább úgynevezett csillogós verzió akkor külön sorba kerül. Ugyanez történik, ha más nyelvű, állapotú és kiadású a lap.

Alternatív rendszerezési opciókként tudjuk a lapokat paklikba rakni, ilyenkor mindkét helyen feltűnik a kártya, hiszen a kollekciónk és egy kész paklink része is a lap. Egy lap belehelyezésekor a pakliba az oldal úgy tekinti mintha egy teljesen új lapot adnánk hozzá, ami eddig nem volt birtokunkba, ennek elkerülésére belehet állítani, hogy a lapot csak a saját készletünkből keresse ki.



2. ábra Deckbox:
A weboldalon található kollekcióm, amit már több éve nem itt vezetek, így nem naprakész

Egy kártyának kijelzi oldal a következő hasznos információit: jelenlegi árat pár weboldalon, hány felhasználónak van ez a fajta lap a paklijában és cserére kínálva, valamint, hogy milyen játékmódokban használható a lap.

A weboldalon érződik, hogy nem a legújabb dizájn és bizonyos cserével kapcsolatos funkciók egy havonta megújuló prémium előfizetéshez vannak kötve.

2.1.3. Delver Lens mobilalkalmazás

Delver Lens Android alkalmazás a második felület, ami nagyon népszerű lett játékos körökben, 2020ban jelent meg egy teljesen új módszerrel az emberek kártyakollekciójának tárolására. IOS platformra a weboldal alapján nem tervezik kiadni az alkalmazást.

Ezen az alkalmazáson belül lehetett legelőször mobiltelefon kamerájának használatával könnyedén, gyorsan és pontosan leszkennelni akár több száz lapot is egyenként.

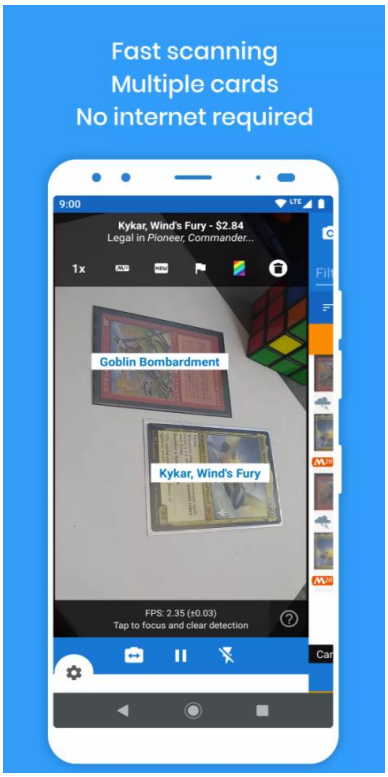
Természetesen nem volt hibamentes, ezért a felhasználó képes manuálisan alakítani, ha egy rossz kiadást, verziót, ritkaságot ismert fel a rendszer szkenneléskor. Mint az elsőleg vizsgált rendszernél ez is képes importálni és exportálni kártyalistákat más alkalmazásokhoz. Lehetőséget kínál Dropbox és más szolgáltatások párosításával biztonsági mentést is oda helyezni, ezzel több helyen elérhetővé tenni a játékos saját lapjait.

Szkenneléskor az alkalmazás automatikusan egy listába pakolja a lapokat, amit elnevezhetünk bárminek. A lista neve mellett megjelenik hány lapot tartalmaz és ezek értékének összege. Ingyenes verzióban a lista 100 laponként limitálva van így, ha több lapot szeretnénk egyszerre felvinni a rendszerbe, azt több külön listában tudjuk csak megtenni.

A listán belül feltünteti nekünk az alkalmazás a képet, ami alapján felismerte a lapunkat és ezt használja, mint ikon ahhoz a megadott laphoz. Ennek segítségével később is kitudjuk javítani az applikáció hibáit, tehát nem kell rögtön a kép elkészítése után módosítani azokat. Egyenként tudjuk mozgatni és másolni a lapokat más mappákba és/vagy listákba.

Lapok mellett feltünteti a jelenlegi árát értékesítő oldalakról szerzett információk alapján. Mivel ez egy világ minden pontján játszott játék így opciót is kínál a beállításokban az árazáshoz milyen weboldalról szerezze az adatait: Európában népszerű Cardmarket-et vagy USA-ban használt TCGplayer-t.

Ez az Android alkalmazás a legújabb a megvizsgáltak közül, viszont bizonyos menüpontoknál nehezen navigálható, zsúfolt a felület. A menüben túl sok opció van ezzel elrejtve a fontosabbakat. További limitációkat képez a prémium havonta megújuló előfizetés a mögé rejtett funkciókkal. Ezeket ki lehet próbálni egy 15 napos ingyenes próbaidőszak alatt.



3. ábra Delver Lens:
Az applikáció szkennere képes egyszerre több lapot is felismerni egy képről

2.1.4. ManaBox mobilalkalmazás

Az úgynevezett ManaBox Android és IOS platformokon egyaránt elérhető alkalmazás, ami 2018ban jelent meg. Funkciói nagyban megegyeznek az eddig említettekével, viszont az évek során sok fejlesztés után ez vált a legnépszerűbbé a három közül.

Megjelenésekor még nem volt képes kamerával szkennelni lapokat. Legtöbben az előbb vizsgált Delver Lens segítségével importálták lapjaikat ebbe az alkalmazásba. Nagy előnye a kinézete és leegyszerűsített felülete, mely sokkal felhasználóbarátabb megoldás volt, mint eddig bármelyik említett rendszernek. Természetesen innen is lehetett exportálni is digitális alkalmazásokba már készen összeállított paklikat. 2022ben kiadták ők is a saját kártya szkennelő lehetőséget, mint egy ingyenes frissítés.

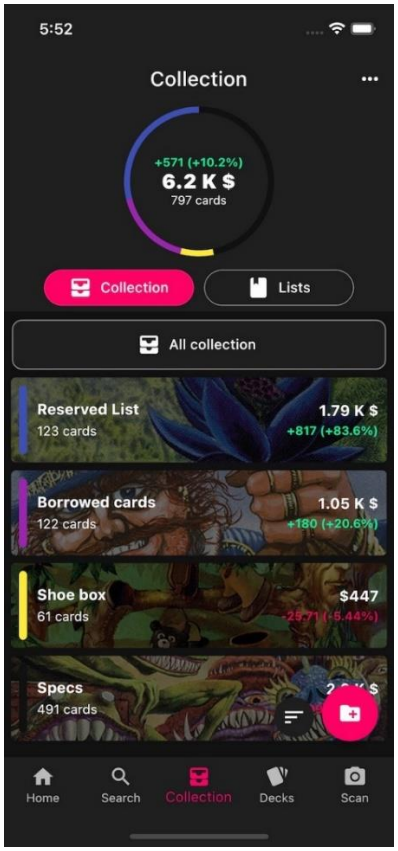
Szkenneléskor felismeri a lapot és egy ideiglenes listába pakolja azt. Itt manuálisa korrigálni tudjuk annak esetleg hibás aspektusait. Új opciókat is kínál a felhasználónak azzal, hogy miután bevitte teljes pakliját, beállíthatja milyen játékmódra készült és elnevezheti bárhogy. Egy ilyen pakli bevitele után az egészet egybe hozzáadhatjuk a kollekciónkhoz, ahol minden lap egyedileg szerepel. Mivel ez egy opcionális funkció, akár félkész paklikat is létrehozhatunk úgy is, hogy hiányoznak bizonyos lapok. Ezeket kijelzi nekünk az applikáció és egyenként akár átmozgathatjuk a keresett lapok listájára.

Egy paklinak részletesen meg lehet tekinteni az árát, a külön fajta lapok elosztását hasznos grafikonok segítségével. Más hasznos észrevételeket is feljegyezhetünk rajta egy-egy mérkőzés lebonyolítása után, hogyan is kellene tovább fejleszteni paklinkat. Egyedi lapok megtekintésekor kijelzi az alkalmazás a lappal kapcsolatos szabályokat és melyik játékmódban legális a lap használata.

A fő menüpontban pedig a teljes szabálykönyv megtalálható bonyolultabb interakciók helyes levezetéséhez. Felkínál ezenkívül egy kifejezetten cseréhez használt felületet, melyen belül két térfélre pakolhatunk kártyákat és ezek árának összegét valós időben kijelzi. Ennek használatával elkerülhetőek esetleg az olyan esetek, amikor valaki be akar minket csapni azzal, hogy más árnak hazudja be a cserére kívánt kártyáját.

Annak ellenére, hogy ez a legnépszerűbb a megvizsgált opciók közül, hiányoznak fontos funkciói ennek is. Exportálásnál nem ad lehetőséget a játék online kiadásához szükséges formátumra, így ahhoz külön weboldalt kell felkeresni. Legtöbb menüpontot csak ikonokkal jeleznek, így némelyik használata nem egyértelmű mit fog tenni a listáinkkal.

Kártyaolvasója sajnos a legutóbbi kiadásként menti el a beolvasott lapot minden esetben, a tesztelt kártya kiadásától függetlenül. Sötét témája az alkalmazásnak egy egyszeri vásárlás után használható csak. Mivel ingyenes az applikáció így reklámokat helyeztek el a felület különböző helyein.



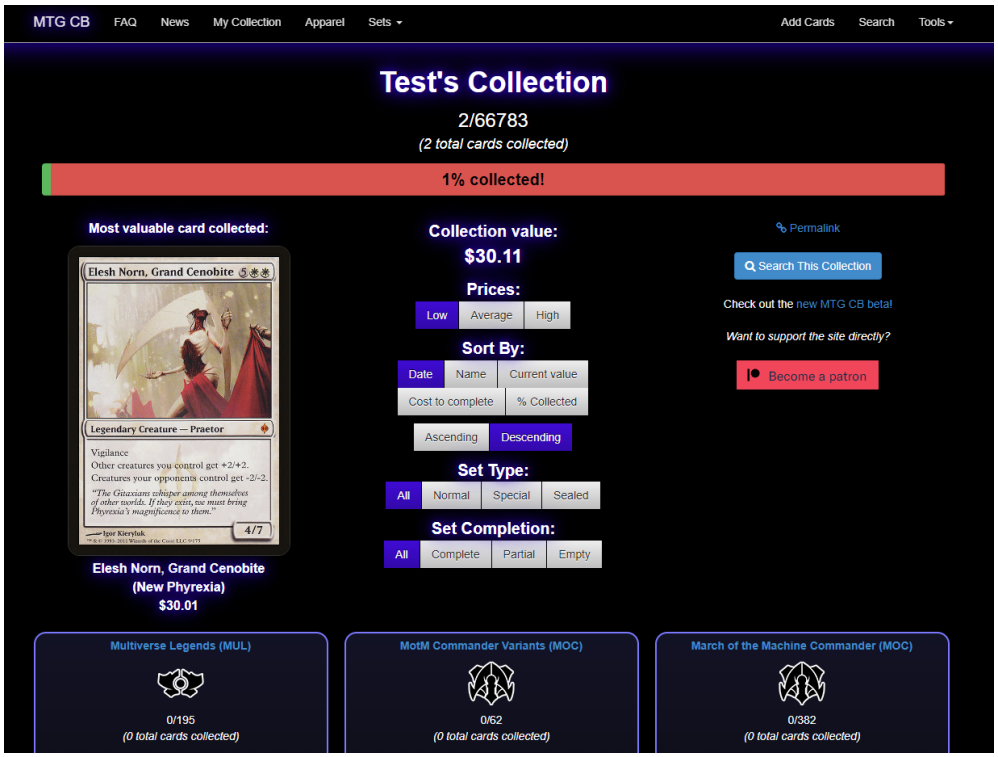
4. ábra Manabox:
Gyűjteményünk teljes árát kiírja és azt is mennyit változott a közelmúltban

2.1.5. MTG Collection Builder weboldal

Ez a weboldal 2013 óta kínál lehetőséget lapjaink leltározására. Digitalizálni manuális keresőből van lehetőség. Ez egyszerű és könnyen használható. A legnagyobb különbsége az oldalnak, hogy itt kiadásonként lehet nyomon követni hány lap hiányzik melyikből. A bizonyos szettnek, amit éppen beírunk kiírja hány százaléka van így tulajdonunkba. Itt könnyen átnézhető egy teljes kiadás, átfutva mi hiányozhat esetleg egy számunkra még ismeretlen lapra is lelhetünk. Egy gombnyomásra a lap egy közösség által futtatott amerikai boltból akár meg is lehet rendelni azt.

A kártyáinkat különböző opciók alapján tudjuk rendezni. Lapok ára, abc-sorrendben név alapján és egész kiadásokat az alapján például, hogy hány lap hiányzik belőle. Az oldalon az összes valaha kiadott kártya és annak bármelyik verziója is szerepel, így nagyon precízen beállíthatjuk, ha egy különleges lapunkat szeretnénk bevinni a rendszerbe. Lehetőség van itt is importálni hasonló alkalmazásokból és exportálni a játék számítógépes verziójába.

Gyűjteményünk főoldalán legértékesebb lapunk látható, mellette pedig a lapok összértéke. Ezek árát az oldal leírása alapján naponta frissítik, ami egy elég jó időtáv, viszont néha, ha egy lapkombináció felfedezése miatt megnövekedik hirtelen egy-két lap ára és ez nem fogja tudni időben kijelezni annak mozgását.



5. ábra MTG Collection Builder:
Az oldalon létrehozott teszt kollekcióm, ami csak pár lapot tartalmaz

A rendszer 2013 óta nem lett frissítve, kivéve az új kiadások implementációját. Viszont egy béta verziót készítenek teljesen új felülettel és funkciókkal. Mivel a nem végleges béta weboldalon még minden megváltozhat, így ezt nem fogom analizálni, mint hasonló rendszer. Felületes átfutás után az előbb említett újabb rendszerekre hasonlít, implementálva a saját teljes kiadás gyűjtésére orientált nézőpontját.

2.1.6. Összefoglalás

A megvizsgált hasonló rendszereket egy 1től 5-ig terjedő skálán értékelem a következő táblázatban.

	Fizikai	Deckbox	Delver Lens	ManaBox	Collection Builder
Digitalizálás	***0/5*** Nincs rá lehetőség	***3/5*** Manuális keresővel	***5/5*** Offline mobil szkenner	***4/5*** Online mobil szkenner	***3/5*** Manuális keresővel
Kártyák rendezése	***2/5*** Címkék alapján lehet	***4/5*** Minden fontos alapján	***4/5*** Minden fontos alapján	***5/5*** Minden lehetőség alapján	***4/5*** Minden fontos alapján
Hasznos információk	***0/5*** Nem nyújt ilyet	***4/5*** Ár, kiadás, verzió	***4/5*** Ár, kiadás, verzió	***5/5*** Ár, kiadás, verzió, szabályok	***2/5*** Ár, kiadás
Felhasználó-barát felület	***0/5*** Nem szempont	***2/5*** Nehezen átlátható, régi	***3/5*** Követhetetlen listák, rejtett funkciók	***4/5*** Letisztult, rendszerezett	***2/5*** Üres, régi, kihasználatlan felület
Összegezve	2/20	13/20	16/20	18/20	10/20

2.2. Fejlesztői környezetek

Első és egyben legfontosabb döntés, hogy milyen környezetben fogom elkészíteni az Android alkalmazást. Itt a három legnépszerűbb környezetet Android fejlesztésben taglalnám a következő pontok alapján:

- Fontos funkciók
- Támogatott programozási nyelvek
- Népszerűség
- Árazás

2.2.1. Android Studio

Az Android Studio talán a legnépszerűbb a vizsgált opciók közül, arra majd még visszatérek, hogy ez miért fontos nézőpont. Mivel ez a Google által támogatott és ajánlott környezet így okkal az első opciója sok embernek. A programomat csak Android operációs rendszerre tervezem elkészíteni, így nem kell esetleges portolási problémákkal később foglalkoznom bármelyik opciót választom. [3]

A rendszer kifejezetten Android applikációk fejlesztésére van kialakítva, így rengetek funkció van, amit hibamentesen lehet használni és kipróbálni. Könnyedén lehet egy emulátorként elindítani a programot ezzel tesztelni a már készülődő szoftvert. Ugyanez az indok miatt optimalizált is fog maradni a program az erre használatos fórumokon rengeteg megoldás megtalálható. [4]

Könnyen integrálható GitHub-al, amely megkönnyíti esetleges hibák visszavonását egy verziókövetés segítségével. Ezenkívül beépített felület alakító funkciókkal egyszerűbb a program olyan elemeit felépíteni.

Az Android Studio által támogatott programozási nyelvek a következők: Java, C, C++, Kotlin.

Ez a környezet nagyon népszerű és könnyen megtanulható. Ezen kívül rengetek videó és cikk található hibaelhárítás vagy ötletmerítés gyanánt. Egy kezdő Android fejlesztőnek is ezt ajánlják a rendkívül egyszerű felülete és költségmentes használata miatt.

2.2.2. Eclipse

Az Eclipse volt a legnépszerűbb környezet az Android Studio megjelenése előtt. Emiatt rengetek anyag van feltöltve és érhető el publikusan.

Nagyon sok plug-in érhető el erre a környezetre, ami megkönnyíti bizonyos részeit a fejlesztésnek. Ezt is könnyen össze lehet kötni GitHub és más verzió követést és programozást könnyítő oldalakkal. Negatívjai, hogy a legújabb Android verzióknál már egyre kevesebben használták. Ennek választása ilyen szempontból egy visszalépés lenne, mivel van már egy népszerűbb és támogatottabb alternatíva.

Az Eclipse rengetek programozási nyelvet támogat, ezek közül párat említenék meg: Java, C, C++, C#, JavaScript, Python.

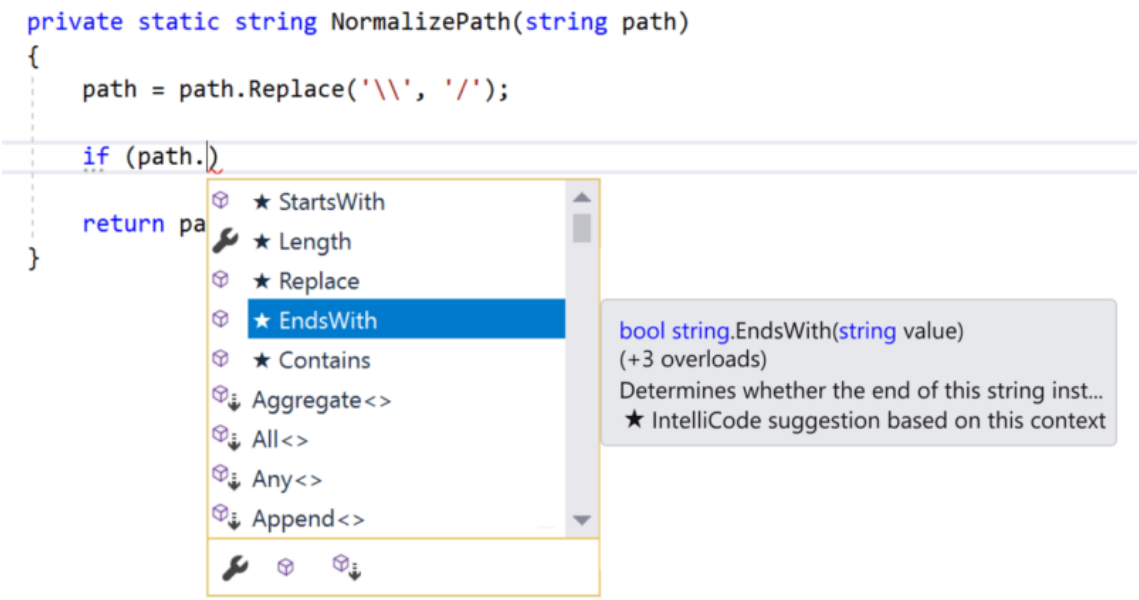
Szerencsére ez is, mint a legutóbb megvizsgált lehetőség ingyenes, így ez nem befolyásolhatta későbbi döntésemet és tudtam tisztán a környezet előnyei és hátrányai alapján választani.

Kutatómunkám során kiderült, hogy önmagában nem lehet benne Androidra fejleszteni és külön szükségünk van az Android Studio-ra is. A kettőt egy plug-in segítségével köti össze. Mivel nem vagyok egyikben sem jártas ez egy fontos információ volt ahhoz, hogy sokkal inkább az önállóan működő és naprakészebb platform mellett döntsek a kettő közül. [5]

2.2.3. Visual Studio

A Visual Studio egy mai napig nagyon népszerű és széles körűen használt környezet rengetek különböző programozási nyelven való fejlesztésre. Saját programozási tanulásom is ezzel kezdődött és még egyetemen is ezzel folytatódott. Sokáig nem is jött szóba Android fejlesztéshez a Visual Studio. Ez megváltozott amikor az integráltak Xamarin-nal, ami lehetővé teszi a cross-platform fejlesztést.

Az integrált tesztelési funkciók miatt zökkenőmentesen lehet fejleszteni. Ezenkívül a beépített úgynevezett IntelliSense segítségével gyorsan és könnyedén lehet dolgozni. Az IntelliSense automatikusan felkínálja a szerinte odaillő opciót miközben írjuk a kódot ezzel felgyorsítva a munkafolyamat ezen részét.



6. ábra IntelliSense

Az eddig megvizsgált lehetőségek közül a Visual Studio támogatja a legtöbb nyelvet.

Ezek közül a legfontosabbak a C#, JavaScript, Python és a HTML.

A Visual Studio a világon legnépszerűbb környezet, ha nem egy bizonyos kategóriában való fejlesztést nézünk. Viszont, mivel itt az Android egy integráció segítségével elérhető csak, így beépített segítség nem sok van a fejlesztéshez.

Az eddig említett érvek és mivel egy új nyelvvel is szeretnék megismerkedni az applikáció elkészítéséhez így választásom a következő:

1. A fejlesztői környezet az Android Studio lesz, ennek hatására rengetek anyag fog tudni segíteni az alkalmazás megfelelő elkészítésében.
2. Programozási nyelvként pedig a Kotlin-t választom, mivel sok cikk és ajánlások alapján ez a legjobb választás egy Android alkalmazásra, ha új nyelvben gondolkodik az ember. [6]

2.3. Felhő megoldások

A programom következő eleme a kártyák felhőben való elmentésére és esetlegesen ott elvégzendő képfelismerő algoritmusokra alkalmas tároló. A világon két legismertebb felhő alapú tároló megoldásokat fogom taglalni, mint opciók.

2.3.1. AWS S3

Az amazon volt világelső abban, hogy kihasználatlan szervertelepeit mások számára is megnyitotta, ezzel beindítva ezt a piacot. Mai napra már sok egyéni felhasználó számára is használható megoldással rendelkezik. Ezek közül talán a legszimpatikusabb az S3 (Simple Storage Service) [7]

Ez az opció 12 hónap ingyenes próbaidőszakot kínál, mely bőven elegendő a szakdolgozatom elkészítéséhez és bemutatásához szükséges időre. Mivel sok funkcióját nem használnám ki ezért bőven elegendő a kínált 5GB tárhely. Ezen kívül 2000 kérést tudok majd felhasználni, amely a demonstrációra és az előtte lévő tesztelésre tökéletesen megfelel.

7. ábra S3 működése

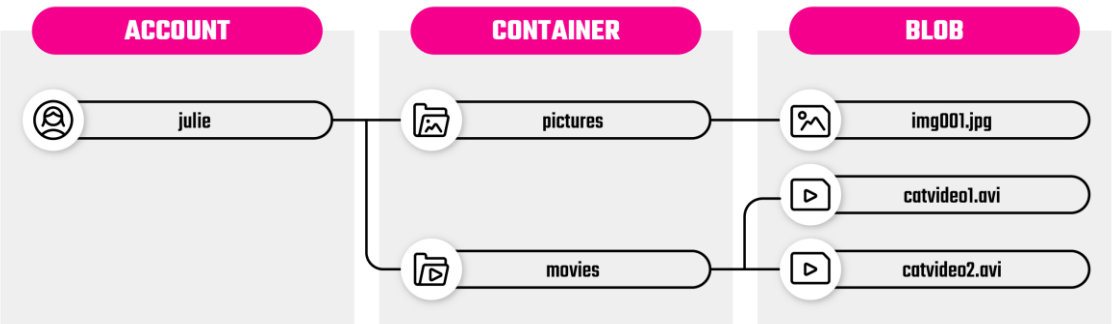
Mivel az alkalmazást nem több száz felhasználó adatbázisának egységet tárolására tervezem, így ez a megadott tárhely is kellő forgalmat kínál. Alkalmazásokhoz is könnyen használható, így az én tervezett alkalmazásomhoz tökéletes megfelel. Feladata a kártyákról készült képek tárolása lenne.

Ezenkívül még az úgynevezett AWS Rekognition-t találtam, ami nagy plusz lenne, ha ezt az opciót választanám. Ezt később fogom részletesen körül járni a Képfeldolgozási szekcióban.

2.3.2. Azure

A másik megoldás az úgynevezett Azure Blobs. Az Azure weboldala alapján a programomhoz legfelelőbb opciót választottam. [8]

Az Azure Blobs a különféle lehetőségek közül a legmegfelelőbb egy alkalmazáshoz, amely képeket szeretne tárolni és lekérni. Ezenkívül az esetleges mellék adatok, amelyek segítségével a szkenner képes felismerni a lapokat, is itt tárolható. Természetesen ezt a variánst is sokkal nagyobb adatokra tervezik, viszont ezért bőven belefér a megengedett kreditmennyiségbe az alkalmazás használata alatt felhasznált tárhely.



8. ábra Azure Blobs felépítése

Ez a megoldás is 12 hónap ingyenességet biztosít egy bizonyos havi kvóta átlépéséig. Akkor viszont ki kell fizetni az extrát, amivel meghaladtuk azt. Ezen kívül 200$ dollár mennyiségű kreditet is kapunk. Itt kevésbé van leírva egy egyszerű felhasználó számára elsőre, hogy ez a kredit pontosan mire is használható, de végül az AWS-hez hasonlóan ezt a mennyiséget kell beosztanunk az ingyenes próbaidőszak alatt kvótáktól függetlenül. [9]

Árazása ezenkívül még komplikáltabb, mivel itt adatmennyiségek mozgatása után és más operációk és kérések lefutása után is csökken ez a kreditmennyiség. Természetesen ezek az árak sokkal nagyobb mennyiségekre vannak kalkulálva, így bármennyire is akarnám túllépni az alkalmazás tesztelésével ezt a határt realisztikusan nem fogom tudni soha

AWS Rekognition megtalálása után részletesebben átfutva a lehetőségeket találtam egy Azure-on belüli alternatívát is. Így mindkét megoldás kínál felhőben lefutó képanalizáló funkciókat.

2.4. Képfelismerő lehetőségek

A program fő feladata egy lap beszkennelése és a róla készült fotó alapján beazonosítani azt. Ez majd lefut egy adatbázison és visszaadja a bizonyos értékeket, amiket keresünk. Ezzel kapcsolatban két lehetőségem van.

Az első az, hogy online a felhő tárolókon belüli opciókat használom, mint például az AWS Rekognition vagy az Azure Cognitive Service for Vision. Ezekkel az opciókkal internet eléréshez lesz szüksége végig az alkalmazásnak, hogy lefusson a felismerő rendszer és kinyerje a képről a nevét és kiadását jelző karakter sorozatot. Ilyenkor a legfrissebb árakkal tud visszatérni a felhasználóhoz a digitalizált lap, amit API-on keresztül szerez meg.

A második az, hogy egy offline megoldást választunk. Ilyenkor a programon belül lenne megvalósítva ez a szkenner. Erre a legjobb megoldást az OpenCV adja. Ebben az esetben elkészül a kép melyen lefut ez az algoritmus. Utána kell raktározni egy bizonyos adatbázist az alkalmazáson belül is, annak a legutóbbi verzióját amikor még volt internet elérés. Ilyenkor ezeket az adatokat manuálisan le kell futtatni API segítsége nélkül és megtalálni a passzoló lapot, majd kijelezni azt.

2.4.1. AWS Rekognition

Amazon Rekognition egy felhő alapú megoldás, mely során a használt algoritmus már előre felismer nagyon sok különféle tárgyat és írást. Jelen helyzetünkben elég csak a kártyának a nevét beolvasnia és 3 betűs meghatározóját. Ezen belül is lehet több variáns egy lapból, de ezeket még finomításokkal meg lehet oldani, ha tényleg képes minden felismerni és kiszűrni, amit szeretnénk egy fotóról. Ezt természetesen külön tesztelni kell mielőtt alkalmazásba építeném.

9. ábra AWS Rekognition felismerő rendszere

A Rekognition ezen kívül temérdek más funkcióra is képes, mint például arcfelismerés, logók és más jelek észrevétele és bármiféle tárgy és táj analizálása.

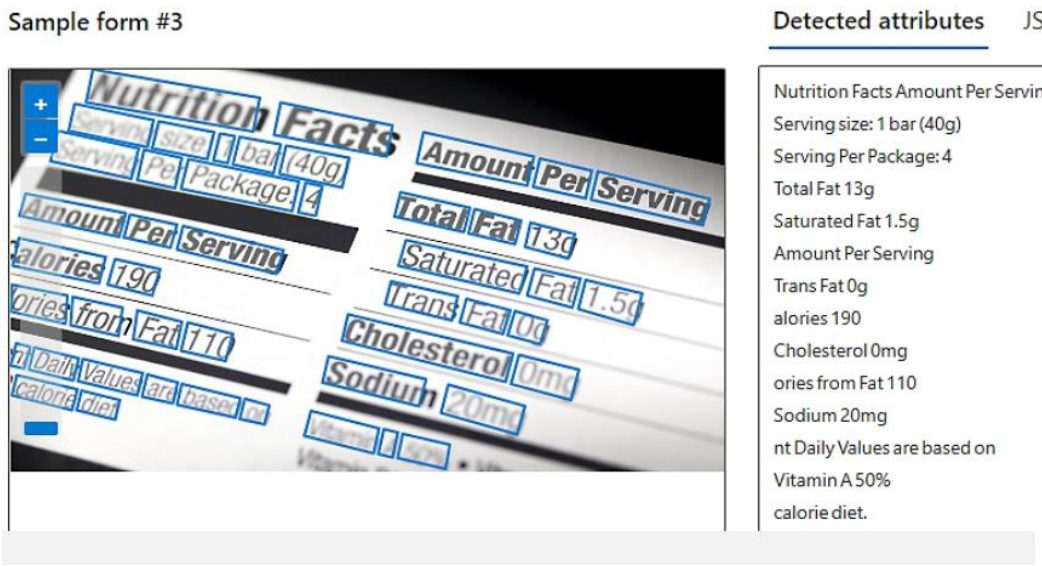
Ezekkel a lehetőségekkel akár a kártyalap rajza alapján is kereshetnénk a tökéletes megfelelőt, viszont ez is több lapon ugyan az így nem lenne nagyobb sikere a csak szöveg alapján keresőhöz képest.

Egy ingyenes próbaidőszak keretein belül 12 hónap alatt 5000 fotón tudna lefutni, ami bőven elegendő az alkalmazás tesztelgetésére és finomítására. [10]

2.4.2. Azure Cognitive Service for Vision

Egyszerűség kedvéért ACSV-nek fogom rövidíteni mostantól ezt a megoldást. A Rekognition-hez hasonlóan ez is egy felhő alapú megoldás és képes felismerni rengeteg dolgot. [11]

Az úgynevezett optikai karakterfelismerés (OCR) segítségével egy képről rengeteg szöveget tud az ACSV felismerni és beszkennelni. Ezek a szöveget lehetnek különböző betűtípusúak és nyelvűek is. A kinyert értékeket futtatnánk le itt is az API-n keresztül az adatbázison és találnák meg a megfelelő kártyalapot.



10. ábra ASCV szövegfelismerő rendszere

A szövegeket a képen is látható különféle dőlésszögben is felismeri, ezzel megkönnyítve bizonyos különleges kártyák felismerésének lehetőségét.

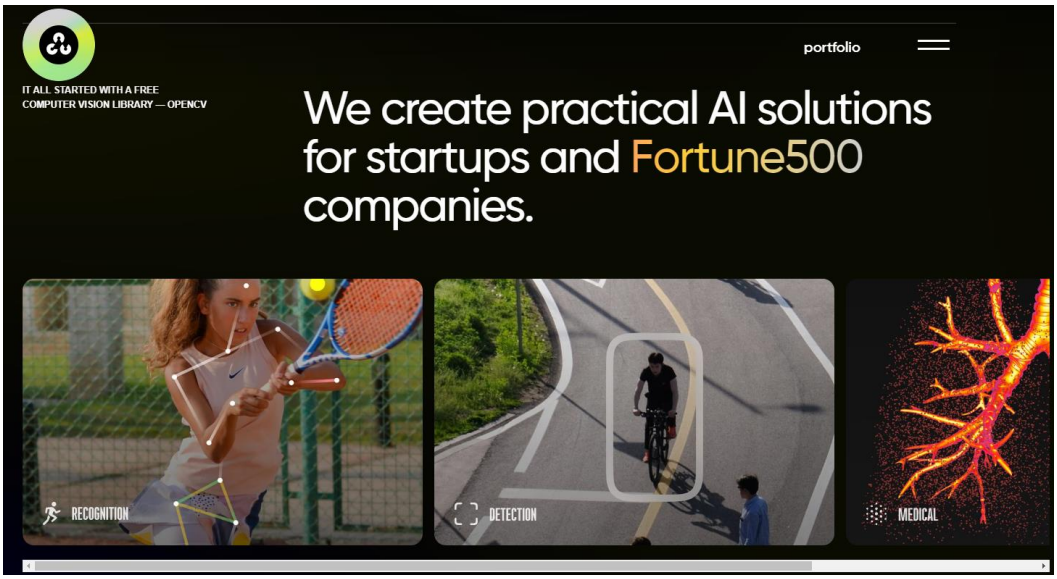
Másik opciónk a teljes lapot egy képként tekinteni és nagyon precízen lefuttatni a teljes adatbázison, hogy pontosan ugyanolyanokat tud e találni. Ezzel az a probléma, hogy nagyon sok ugyanolyan verzió esetén olyan minimális különbségeket kellene az algoritmusan találnia, amit sokszor két ugyanolyannak tekintene.

2.4.3. OpenCV

Ez a megoldás az úgynevezett Open-source computer vision (OpenCV)-t használná. Egy nagyon elterjedt nyílt forráskódú megoldás lenne ez, amelyet világszerte használnak pont, mint az előzőekben taglalt felhő alapú rendszereket. Rengetek programozási funkcióval rendelkező machine learning könyvtár, ami több mint 2500 algoritmust tartalmaz. Közülük a számunkra egy lehetséges megoldás, az optikai karakterfelismerésre tervezett úgynevezett Keras-OCR és egy másik lehetőség az EasyOCR. Az OpenCV megoldás közvetlenül a program része lenne és felhőben lévő folyamatra való várakozás nélkül lefut és kifejezné a pontos adatokat. [12]

Ez a megoldás a következő programozási nyelveket támogatja és Android kompatibilis: C++, Python, Java, MATLAB.

Mivel ez egy felhő nélküli megoldás, így manuálisan kell a konfigurációs fájlt a számunkra legmegfelelőbb beállításokkal ellátni, míg a felhő variánsoknál az ott lévő beállításokat és opciókat kell finomítgatni. Az előbb említett két OCR variáns közül számunkra a legoptimálisabb az Keras-OCR, mivel ez sokkal inkább vegyes képekre van kitalálva, amik nem rögtön kivehetőek. Ezzel ellentétben az EasyOCR inkább nagy kontraszttal rendelkező fehér papíron lévő szövegekre van kitalálva, de azokról sok fajta betűtípust és nyelvet is könnyedén felismer. [13]



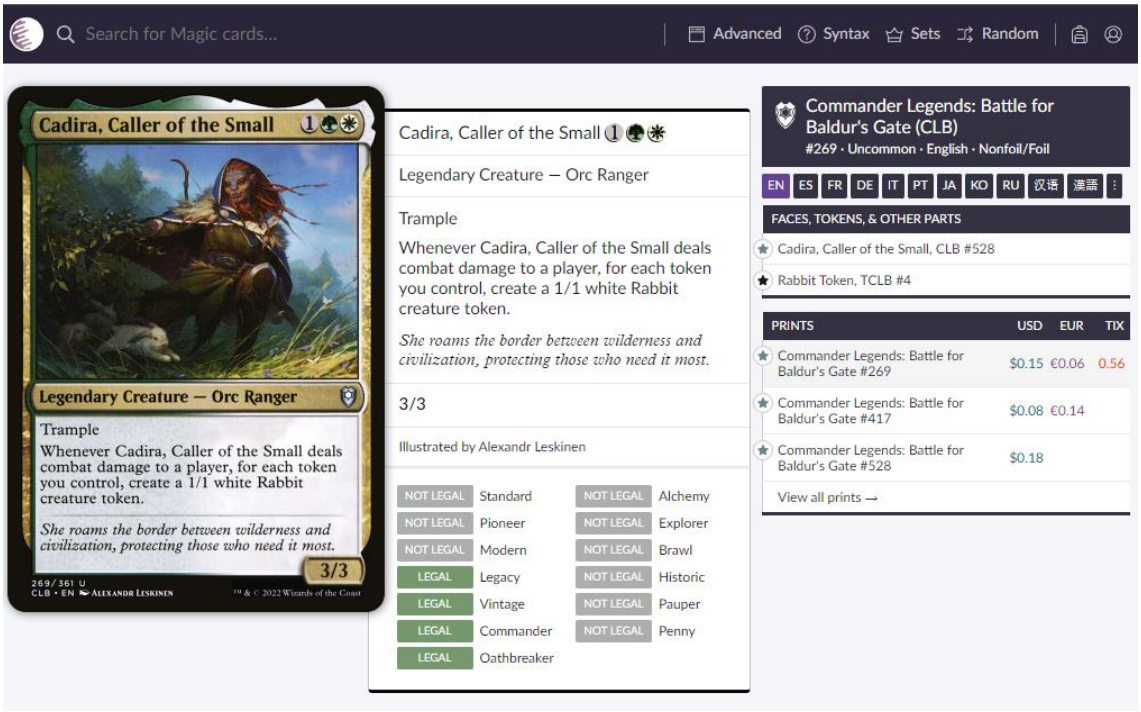
11. ábra OpenCV

2.5. Magic kártya adatbázis és API

Legutolsó fontosabb része a programunknak a nagy adatbázis és az onnan szükséges adatok kinyeréséhez szükséges API. Itt nincs sok választási lehetőségünk, viszont ami van az tökéletesen megfelel az alkalmazás számára.

A Scryfall a legnagyobb és legalaposabb adatbázisa a Magic: The Gathering kártyajátéknak. Ezen az oldalon minden lap és annak verziója megtalálható. Egy kiadás megjelenésekor már elérhető az összes új lap.

Maga a weboldal kínál nekünk egy API-t, ami teljesen ingyenesen használható. Leírásában egyedül arra tér ki, hogy ha túl sok kéréssel bombázzuk a szervert akkor egy idő után letilt minket. Szerencsére azt is megadják, hogy mekkora intervallumot állítsunk be két kérés közötti minimumnak mellyel ez elkerülhető.



12. ábra Scryfall:
Minden adat ami fontos lehet egy lapról az megtalálható ezen a felületen

Az oldal eltárolja minden fontos információját egy kártyának. Ezeket felhasználva könnyedén megtalálhatjuk a pontos verzióját egy lapnak és így annak további adatait. [14]

A legfontosabb adatok, amik alapján az API-n keresztül fogunk keresni:

- A kártyalap neve
- A bal alsó sarokban található kiadást jelző 3 betűs kód
- A kiadástól jobbra lévő jel

A jel különbözteti meg, hogy ugyan az a lap csillogós vagy más alternatív verzió, ami a névből és a kiadásból nem derülne ki. Ez régebbi lapoknál nem egységes viszont, mivel ott a kiadást jelző kód máshol helyezkedik el és kevesebb információ nyerhető összesítve a lapról. Ezért, hogy a program rendesen működjön régi és új kártyákra is egyaránt csak bizonyos adatokból megkeresi a lapot és onnan a felhasználó fogja tudni kiválasztani egy listából a saját lápjának megfelelőt.

2.6. GitHub

Legutolsó sorban a már eddig is rendszeresen általam választott GitHubot fogom, mint verziókövető használni.

Ez egy Git alapú tárhelyszolgáltatás, amellyel csapatok egyszerűen tudnak együttműködni egy programon. Mindezt egy kizárólag felhő alapú, weboldalon keresztül elérhető felület segítségével.

Vannak más alternatívák is, mint például a GitLab és a Bitbucket, de mivel egész eddig ezt használtam tanulmányaim során és jól megismerkedtem a felülettel, ezért nincs indokom egy másik alternatívára váltani.

2.7. Összegzés

Fejlesztői környezetnek az Android Studio-t választottam. Mivel kizárólag Android alkalmazást tervezek elkészíteni, így maga a Google által támogatott és legfrissebb funkciókkal ellátott környezet mellett döntöttem. Ezenkívül rengetek erőforrás is elérhető kezdőbb Android fejlesztők számára, mint jómagam.

Felhő alapú tárolási módszerek közül az AWS S3 felel meg legjobban a program számára. A fejlesztés teljes ideje alatt garantáltan költésmentesen fogom tudni használni minden funkcióját. Ezenkívül ez az opció lehetővé teszi az AWS Rekognition-t, melyről más opció választásakor le kellene mondanom.

A képfelismerő opciók közül még nem tudtam választani, mivel ezeket tesztelés alapján fogom tudni csak biztosra kizárni. A megmaradt két opcióm az OpenCV és az AWS Rekognition. Mindkettő kompatibilis a választott felhő technológiával, viszont az OpenCV internetelérés nélkül is képes kiadni az adatokat. A Rekognition pedig jobban kihasználná a felhőben tárolt adatbázist, ezzel frissebb adatokat nyújtva bármikor.

Kártya adatbázisnál nincs nagy választási lehetőség és nincs is szükség rá. A Scryfall és annak API lehetőségével az összes valaha nyomtatott lapot lekérhetjük és akár a legkisebb tulajdonság alapján is kereshetünk közöttük.

Verziókövetőnek pedig a GitHubot választottam személyes preferencia és már jól bevált tapasztalat alapján.

3. KÖVETELMÉNY SPECIFIKÁCIÓ

Ebben a fejezetben a programom legfontosabb elemeinek elvárásait fogalmaznám meg.

3.1. Adattárolás

Az alkalmazás egyik legfontosabb feladata az, hogy gyorsan és egyszerűen elérhetővé váljon a felhasználó gyűjteménye bárhol és bármikor. Ehhez szükséges egy, a legutóbbi internet eléréskor lementett állapot, mely alapján könnyedén átfutható a kollekció bármelyik része az akkori adatokkal. Ilyen esetben természetesen a változékony értékek, mint például egy népszerű kártyának az ára nem lesz naprakész, ezt érdemes fel is tüntetni a felhasználó számára.

Amikor internet eléréssel indítjuk el a programot automatikusan frissíti a legújabb adatokkal lapjainkat, ezzel biztosítva azok valós értékét. Ezen felül sajtóhiba esetén az új tulajdonságokat is fel kell tűntetni a kártyánál, mivel bizonyos pakliknál és kártyáknál sok változtatást hozhatnak ezek módosítása. Ezek a sajtóhiba javítások új kiadásoknál jelennek meg, melyben egy régebbi kártyát újra nyomtatnak bizonyos eltérésekkel.

3.2. Szkennelés

Az applikációnak szüksége lesz a telefon kamerájához a lapok szkenneléséhez, így annak az engedélynek a megvonása esetén jelezni kell, hogy a szkennelő funkció nem elérhető. Ezen felül a teljes szkennelés folyamatának a kép készítésétől, a passzoló digitális kártya gyűjteménybe adásáig aránylag gyorsnak kell lennie. Ez felettébb fontos, mivel a gyűjtők és játékosok általában nagy kollekciókkal rendelkeznek.

Maga a szkennelés folyamatának különösen precíznek kell lennie, mivel ugyanolyan lapok között csak a kiadás jele, vagy más apró különbség vehető csak észre. Ilyen esetben az alkalmazásnak érdemes kilistázni a potenciális lehetőségeket, hogy a felhasználó tudjon választani.

3.3. Rendszerezés

A felhasználó kollekciójának átláthatónak és egyénileg módosítható szabályok alapján rendezhetőnek kell lennie. Ezen felül saját listákat és paklikat is tudjon készíteni, az utóbbinál meghatározva, hogy milyen módú játékra lett az építve. A listákból vagy paklikból való törlés vagy módosítás esetén a felhasználó tudjon választani, hogy minden előfordulását szeretné-e változtatni.

3.4. Exportálás

Az applikációból exportálandó listákat és paklikat képes legyen az alkalmazás az előre beállított játékmód alapján kinyerni. Ezen felül, ha az exportálás célpontja egy olyan alkalmazás, ami bármilyen kiadású lapokat tud kezelni, akkor a felhasználó által beállított kiadás és verzió jelenjen meg, ne pedig a legújabb nyomtatású.

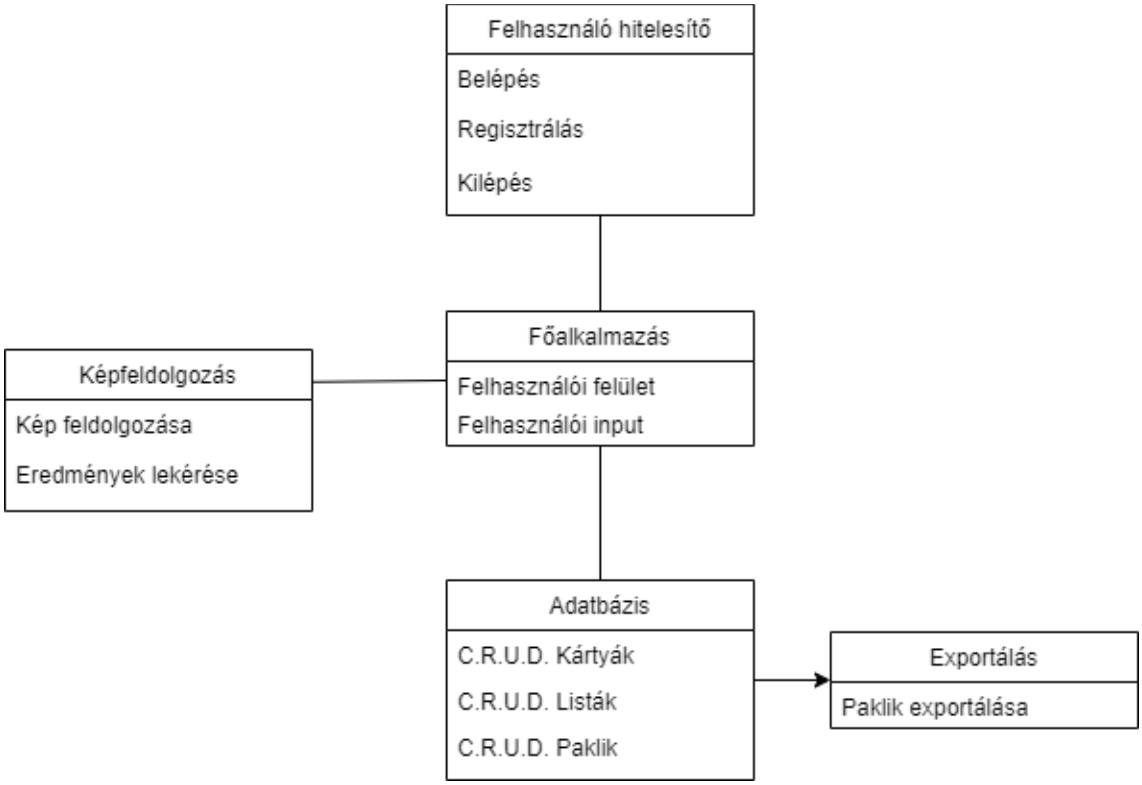
```
Deck
4 Beanstalk Giant
4 Bonecrusher Giant
4 Brazen Borrower
4 Breeding Pool
4 Edgewall Innkeeper
3 Escape to the Wilds
4 Fae of Wishes
6 Forest
3 Island
4 Lovestruck Beast
4 Lucky Clover
2 Mountain
2 Nissa, Who Shakes the World
3 Steam Vents
4 Stomping Ground
1 Temple of Abandon
2 Temple of Epiphany
2 Temple of Mystery
```

13. ábra Pakli lista:
Egy standard játékmódú pakli az MTG: Aréna játék importálásának megfelelő formátumban.

4. TERVEZÉS

4.1. Rendszerterv

A következő képen látható az alkalmazás teljes rendszere és a különféle modulok kapcsolata.

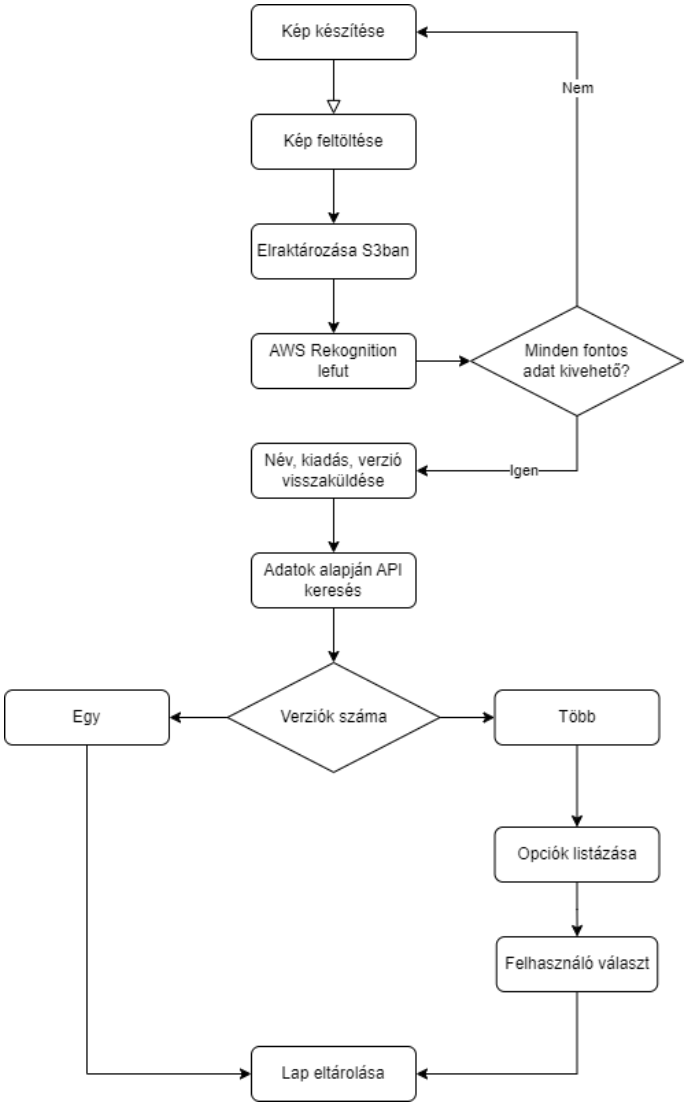


14. ábra Rendszerterv

A hitelesítő modul felel arról, hogy a felhasználó a saját kollekciójába bármilyen Androidos eszközről képes legyen belépni.

A főalkalmazásból lesz elérhető minden fontos funkció és beállítás, ami szükséges a játékos egyéni rendszerezéséhez szükséges.

A képfeldolgozás modul lesz képes egy kép alapján beazonosítani és megkeresni a megfelelő kártyalapot. Ennek működését a következő ábrán szemléltettem.



15. ábra Képfeldolgozás diagram

Az adatbázis modul felel a kártyák lokális tárolásáért internetelérés hiányában, valamint azok módosításáért is.

Az exportáló modul segítségével a leggyakoribb formátumokba lehet exportálni a szükséges paklik és listák adatait.

4.2. Funkciólista

4.2.1. Felhasználó hitelesítő komponens

Név	*Leírás*
Belépés	A felhasználó megadja saját adatait és ezzel eléri a saját kollekcióját.
Regisztrálás	Első használat esetén egy felhasználót létre kell hozni, amivel később elérhető a kollekció.
Kijelentkezés	Kilépés után más felhasználó is be tud lépni a rendszerre.

4.2.2. Főmenü komponens

Név	*Leírás*
Lapkeresés menüpont	Egy bizonyos kártya kikeresése a felhasználó saját gyűjteményéből.
Kész listák menüpont	Kijelzi a kész listáit a felhasználónak pl: cserelista.
Kész paklik menüpont	Kijelzi a kész paklijait bizonyos játékmódok szerint csoportosítva.
Új lap szkennelése menüpont	Előhozza a kamerát, ami a szkennelés első lépése.
Opciók menüpont	Fontos beállításokat itt lehet módosítani pl: pénznem.

4.2.3. Képfeldolgozó komponens

Név	*Leírás*
Képfeldolgozás	Egy képről kinyeri a fontos adatokat, amivel precízen lehet azonosítani.
Eredmények lekérése	Szkennelés adatai alapján a Scryfall API-ból megkeresi a pontos kártyának képét.

4.2.4. Adatbázis komponens

Név	*Leírás*
CRUD Kártyák	Már rendszerezett kártyák módosítása, törlése.
CRUD Listák	Listák létrehozása, feltöltése, módosítása, törlése.
CRUD Paklik	Paklik létrehozása, feltöltése, módosítása, törlése.

4.2.5. Adatbázis komponens

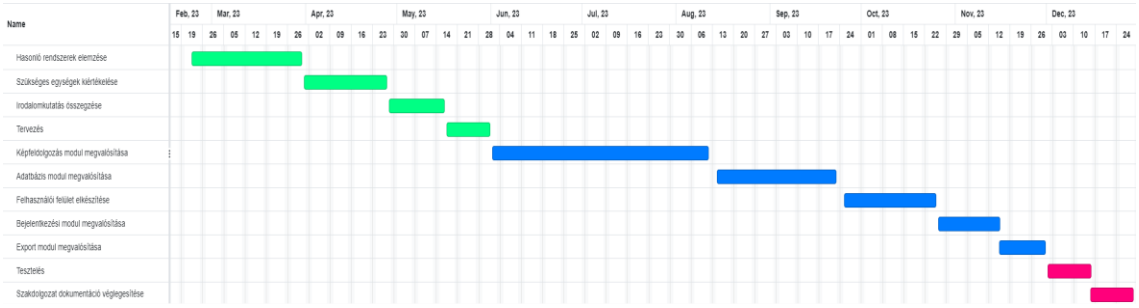
Név	*Leírás*
Exportálás	Videójátékba való pakli, erre specifikus formátumban exportálás

4.3. Fejlesztés tervezett menete

A fejlesztés legfontosabb elemével fogom kezdeni, mivel ez az alkalmazás legkomplikáltabb része. Ennek véglegesítése után elkészítem az adatbázist, ezzel megkönnyítve a fejlesztés közben lényeges teszteléseket.

A következő modul a felhasználói felület modul lesz. Ennek véglegesítése után az alkalmazás már funkcionálisan képes lesz a belső folyamatokra, viszont még nem tud több felhasználó esetén külön kollekciókat vezetni és azokat egymástól függetlenül is tárolni.

Az alkalmazás legutolsó két modulja ígérkezik a legrövidebb folyamatnak és strukturális szempontból is érdemes a végére hagyni. Ezek elkészítése után kezdődik a tesztelési fázis, mely során az előző modulokba fejlesztés alatt nem észrevett hibákat fogom kijavítani.



16. ábra Fejlesztés menete

5. FEJLESZTÉS

5.1. A váz elkészítése

A kártya digitalizáló alkalmazásom fejlesztésének első lépése az volt, hogy létrehoztam egy GitHub-könyvtárat, amelyben elkezdtem az alkalmazás fejlesztését. A Kotlin nyelv mellett döntöttem, és az sokak által ajánlott Android Stúdióban kezdtem el a munkát. Az első lépés az volt, hogy létrehozzak egy kártya objektumot, valamint egy adatbázist, amelyben tárolom a több mint százezer különféle lapot. [15]

A Scryfall API-ról letöltött JSON fájl több mint 400 MB-os volt, és idővel, ahogy egyre több kártya fog megjelenni a játékhoz, ez csak tovább fog növekedni. Sajnos sok problémába ütköztem ilyen nagy adatmennyiség rendszerezésekor és beolvasásakor.

Legnagyobb probléma az magával a kártyákat tartalmazó fájlal volt, mivel általánosságban nem szoktak egy alkalmazásban ilyen méretű JSON fájlt tárolni, így tanácsok és tippek se voltak sok az interneten.

Ideiglenesen egy tesztfájlt használtam, amely mindössze 50 darab kártyát tartalmazott. Az 50 lapos tesztfájl beolvasása viszont sikeresen és gyorsan működött, így ennek segítségével fel tudtam építeni az adatbázis struktúráját és tesztelni a működését. Ez stabil alapot biztosított a további fejlesztésekhez.

5.2. Navigáció

Ezután, mivel már rendelkezésre állt a háttér-adattároló, vizuális elemeket is létre akartam hozni, hogy a későbbi fejlesztések jobban átláthatók legyenek. Ennek érdekében előre elkészítettem Android Studióban vázlatosan az összes tervezett képernyőt: a kollekció, a kereső, a szkennelő és a pakli képernyőt.

Ezekhez létrehoztam az alkalmazás alján található alsó navigációs menüt, amelynek valamelyik ikonjára kattintva az alkalmazás a megfelelő képernyőre vált. Úgy éreztem így már megvan az alap struktúra és rátértem a keresőképernyőnek részletes kidolgozását.

A keresőképernyőre beillesztettem az adatbázisban tárolt 50 tesztlapot, és egy egyszerű, de hatékony keresőfunkciót hoztam létre. A keresőt úgy alakítottam ki, hogy ne legyen szükség a kártyák pontos nevének beírására, mivel elég, ha a felhasználó csak egy részletet ír be, így a keresés gyorsabb és praktikusabb lesz.

5.3. Szkenner

Ezek után elkezdtem dolgozni az alkalmazás talán legfontosabb elemén, a szkenneren. Az eredeti tervek szerint az AWS Rekognition szoftvert használtam volna, azonban időközben a Google által fejlesztett ML Kit 2 és azon belül a Text Recognition V2 jobb választásnak bizonyult. [16]

Ez leginkább az Android Stúdióval való kompatibilitásban és a szkenner implementálásának egyszerűségében mutatkozott meg, valamint a kínált funkciók is sokkal jobban illeszkedtek az alkalmazás megtervezett funkcióinak.

A jelenleg használt szkenner csak a szöveg beolvasására képes, ennek köszönhetően egyszerűen tudom a releváns elemeket kinyerni, és így pontosan azonosítani, hogy melyik kártya van éppen a kamera előtt.

5.3.1 Váltás Google ML Kitre

Az eredeti kutatómunka során már találkoztam a Google ML Kit nevével, de akkoriban még nem volt elterjedt, voltak ismert hibái. Sok fejlesztő nem ajánlotta kezdő Android-fejlesztők számára, mivel nehézkes volt a használata és hiányosságai miatt nem számított megbízhatónak.

Azóta azonban jelentős fejlesztéseket kapott. Mára kifejezetten ideális választás kisebb alkalmazásokhoz, hiszen gyorsan integrálható.

Az első implementáció után az alkalmazás már képes volt a szöveget beolvasni és megjeleníteni a képernyőn. Jelenleg ugyan még nem tudtam semmilyen műveletet végezni ezzel a szöveggel, de ez egy fontos lépés volt annak bizonyítására, hogy a szkenner működik.

A következő lépés innen az volt, hogy kinyerjem a szövegben található releváns elemeket, és ezeket összekapcsoljam az adatbázisban tárolt kártyákkal. Ez a folyamat alapozza meg a későbbi funkciók fejlesztését, mint például a kártyák felismerését és kezelését.

5.3.2 Finomítások

Továbbra is a szkenner fejlesztésére koncentráltam, és a következő lépés az volt, hogy az alkalmazás már képes legyen az adatbázisból megtalálni a kamera előtt lévő kártyát. Egyelőre azonban nem volt szükséges a kártya adatait menteni vagy más funkcióhoz kapcsolni – a cél csupán a felismerés volt.

Ehhez a beolvasott szöveget elemezve olyan kulcsfontosságú elemekre kerestem rá, amelyek minden kártyán megtalálhatók, például nevek vagy egyedi azonosítók.

Elsősorban a teljes szövegben keresek bizonyos kulcsszavakat, amik közül egy garantáltan szerepelnie kell minden kártyán és elég nagy méretű a szövege is, hogy rögtön kivehető legyen. Sajnos az sem garantált, hogy a sorokra osztott szövegmező első sora mindig a kártya nevét tartalmazza. Bizonyos kártyákon megtalálhatóak különleges jelek, amiket felismer a szkenner, mint egy külön sor.

Ezután, ha megvan a kártyalap neve az alapján már ki tudjuk szűrni melyik lapokról lehet szó. Viszont, mivel egy kártyának több verziója, kiadása és kinézete is van, csupán a név alapján még akár több száz opció is lehet. Ezekre a további finomításokra később tértem vissza, mivel a teszt adatbázisban nem voltak duplikált névvel kártyák.

5.4 Kártyapaklik megalkotása

Most, hogy az alkalmazás képes felismerni a kis adatbázisból egy-két lapot, ami fizikailag kamera előtt van így úgy gondoltam itt az ideje, hogy ki létrehozzam a pakli képernyőt. Ehhez pont, mint a kártyákhoz egy saját objektumot kellett létrehoznom, ami a paklikat tárolja és egy teljesen új adatbázist csak ezeknek.

Miután bebizonyosodott, hogy működik az adatbázis egy-két teszteset hozzáadásával elkezdtem még jobban kibővíteni a saját paklik képernyőjét.

Első lépésként hozzáadtam néhány design elemet a pakli oldalhoz, hogy vizuálisan is átlátható legyen. Ezek az elemek megmutatják, hány kártya van az adott pakliban, és a kártyák neveit is megjelenítik. Ezután létrehoztam egy menüpontot, amely lehetővé teszi, hogy néhány kattintással új paklit hozzanak létre a felhasználók.

17. ábra Paklik képernyő

A funkció implementálása közben folyamatosan teszteltem az új paklik létrehozását. Egy újonnan létrehozott paklit elmentettem az adatbázisba, majd manuálisan hozzáadtam néhány lapot. Az eredmény működött: az új paklik adatai pontosan megjelentek, és a rendszer gyorsan beolvasta a hozzájuk tartozó kártyákat is.

5.4.1 Pakli funkciók bővítése

A következő nagy lépés a fejlesztésben az a szkenner még optimálisabbá tétele volt és egy olyan képernyő létrehozása, hogy a szkennerről rögtön bele tudjuk rakni egy pakliba a talált kártyát. Tehát amikor szkenner megtalál egy kártyát és megjeleníteni a felhasználó számára oda létrehoztam egy gombot, ami hatására előhozza, hogy milyen paklik vannak jelenleg elkészítve és melyikbe szeretnénk belerakni ezt a lapot akárcsak egybe vagy egyenként az összesbe.

Miután a szkennerrel való kártya hozzáadás funkció elkészült, hasonló módon implementáltam ezt a kereső képernyőre is. Így a felhasználók már nemcsak a szkenner segítségével, hanem a keresett kártyák listájából is könnyedén hozzáadhatják a kártyákat a paklikhoz.

A keresőképernyőn megjelenő kártyák mellett egy új gombot helyeztem el, amely lehetővé teszi a kártya hozzáadását a paklihoz.

Bár a kártyák hozzáadása mind a szkenner, mind a kereső képernyőről sikeresen megvalósult, kezdetben vizuálisan nem voltak elérhetők a kártyák részletes információi, mint a képek vagy a szövegek. Ezt a problémát a felhasználói élmény javítása érdekében orvosolni kívántam.

Az alkalmazás adatbázisában tárolom a kártyák egyedi online azonosítóit (ID), amelyek lehetővé teszik, hogy a rendszer megjelenítse a kiválasztott kártya képét. Ez a funkció azonban csak internetkapcsolat mellett működik. Amennyiben az alkalmazás nem rendelkezik elérhető internetkapcsolattal, a rendszer korábban hibát jelzett, és az alkalmazás leállt. Ennek megakadályozására bevezettem egy olyan lehetőséget, amely offline üzemmód esetén az univerzális kártya hátlapot jeleníti meg, és egy értesítő üzenetet ír ki, amely tájékoztatja a felhasználót arról, hogy az alkalmazás jelenleg offline módban működik.

5.5 Szkenner továbbfejlesztése

A fejlesztés során eddig viszonylag kevés problémával találkoztam, az első nagyobb nehézség a JSON fájlok méretével kapcsolatos volt. Azonban, amikor bevezettem a szkenner funkciót, amely az online kártyák képeit is megjeleníti, jelentős teljesítménybeli problémák léptek fel. A probléma oka az volt, hogy amikor az új képernyő megjelent a kártya és a paklikhoz adás lehetőségével, a kamera továbbra is aktív maradt a háttérben, és csak pár másodperc után állt le.

Ez a problémás működés jelentős teljesítménycsökkenést eredményezett, és az alkalmazás válaszideje lassult. Mivel nem találtam gyors megoldást erre, a problémát átmenetileg háttérbe helyeztem, és inkább tovább haladtam a fejlesztés más részeivel.

Miután sikerült megoldanom a JSON fájlok teljes beolvasását, és bár a fájlok betöltése sikeresen megtörtént, az időigényessége problémát okozott. Ennek elrejtése érdekében bevezettem egy úgynevezett Splash-Screent, ami egy bizonyos ideig tart az alkalmazás indulásakor és ez alatt a háttér elemeket az alkalmazás már képes felépíteni és beolvasni. [17]

5.5.1 Navigáció a szkennerhez

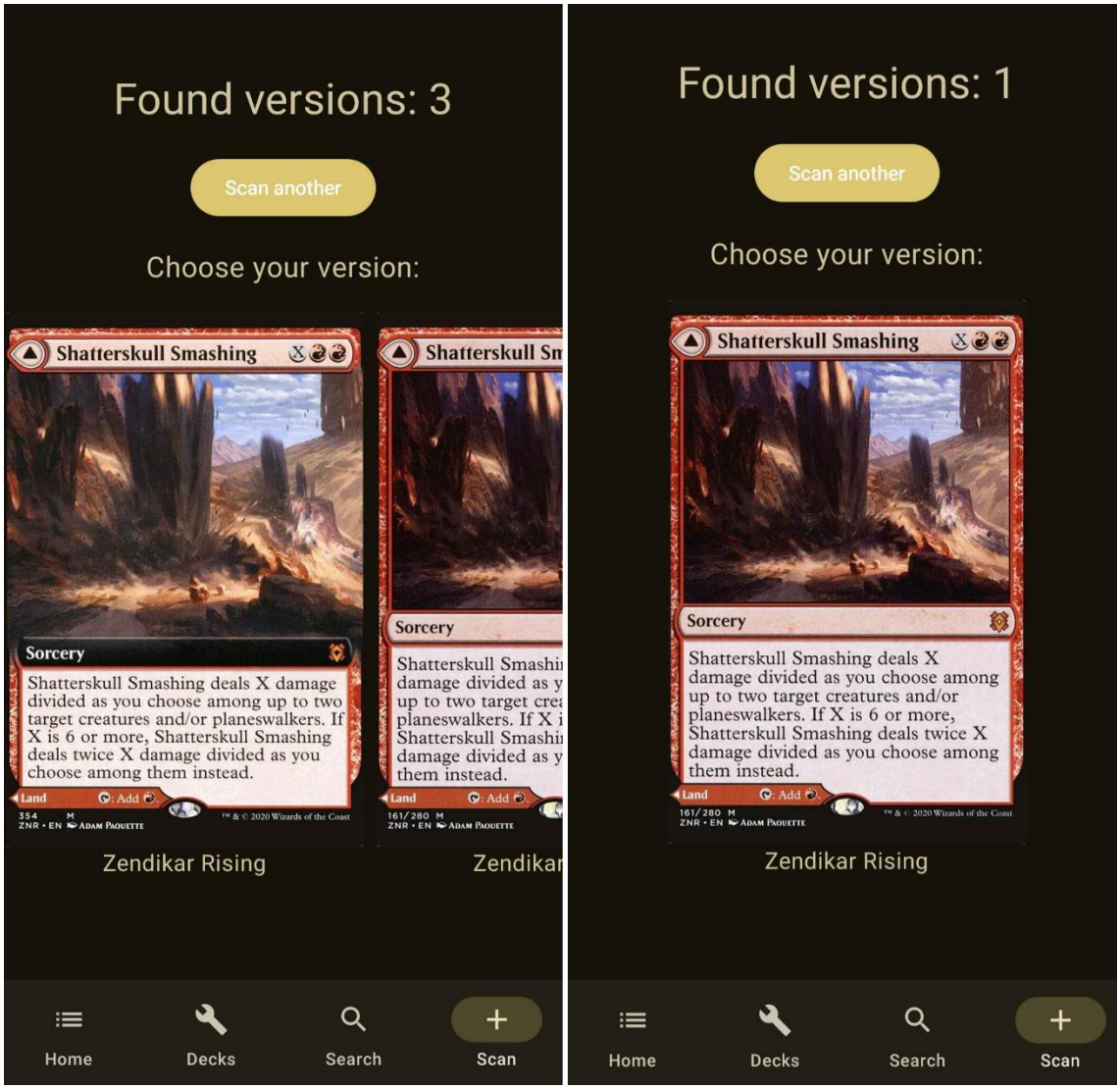
A sikeres adatbetöltés után visszatértem a szkenner teljesítményének javításához, hogy orvosoljam a korábbi problémákat. Ennek érdekében bevezettem egy új navigációs elemet, a NavController-t, amely lehetővé tette a képernyők közötti váltogatást. A két képernyő, amelyeket ezen a ponton kezeltem, a szkenner képernyő és a talált kártya képernyője voltak. [18]

A NavController implementálásával sikerült megoldanom, hogy a két képernyő között oda-vissza tudjak váltani. Amikor elhagytam az egyik képernyőt, a rendszer automatikusan leállította a háttérben futó folyamatokat, így megszakította a nem szükséges funkciókat. Bár a megoldás alapvetően működött, hosszú ideig finomítanom kellett, mivel előfordult, hogy az adatok nem mindig kerültek megfelelően átadásra, és olykor üres kártyák vagy korábban szkennelt lapok jelentek meg. A további finomításoknak köszönhetően sikerült a szkennert még sikeresebbé tennem.

5.5.2 Szkenner befejezése

A következő lépésben a szkenner továbbfejlesztésére összpontosítottam. Módosítottam a kártya kijelző oldalát úgy, hogy az most már az adott kártya összes elérhető verzióját megjelenítse. Ennek hatására a felhasználó könnyedén kiválaszthatja, melyik verziót szeretné tárolni, így biztosítva, hogy minden variáns elérhető legyen a paklikban.

A szkenner továbbfejlesztése során bevezettem egy új funkciót, amely lehetővé teszi, hogy a szkenner a kártyán található specifikus kollekció számot azonosítva csak az adott verziót jelenítse meg. Ez különösen fontos olyan lapok esetében, amelyek több száz különböző variánssal rendelkezhetnek. A megfelelő fényviszonyok és a tiszta kép mellett a szkenner így képes a pontos verziót megjeleníteni, ezáltal még precízebbé téve a kártyák felismerését és kezelését.



18. ábra Különbség kollekció szám hiányakor a kamerán

5.6 API-on keresztül adatbázis

Eddig a fejlesztés során egy nagy lokális fájllal dolgoztam, amelynek az volt a hátránya, hogy nem tartalmazza a legfrissebb adatokat, mint például az árakat és a kártyákat. Ennek orvoslására visszatértem a Scryfall weboldalra, és átállítottam az alkalmazást úgy, hogy az hetente frissítse az adatbázist az új árakkal, lapokkal és egyéb adatokkal.

Ez az új rendszer implementálása némi módosítást igényelt a fájlbeolvasás logikájában. Továbbá, el kellett tárolnom az utolsó frissítés időpontját is, hogy az alkalmazás meghatározhassa, szükséges-e az adatbázis frissítése.

További manuális tesztelés során felmerült, hogy offline módban az alkalmazás nem képes frissíteni az adatokat, illetve nem tudja betölteni a kártyák képeit, valamint az API hívások sem működnek. Ezt a problémát megoldottam úgy, hogy offline módban az alkalmazás a már letöltött adatokat és képeket használja, és nem próbál frissíteni az API-ról.

5.7 Maradék funkciók elkészítése

5.7.1 Publikált paklik

A következő lépés a fejlesztésben az volt, hogy online felületet hozzak létre, amely lehetővé teszi a paklik megjelenítését és tárolását. Az eredetileg tervezett AWS S3 és Rekognition helyett most a Google által fejlesztett Firebase megoldást használtam, mivel jobban illeszkedett a szkennelési rendszerhez. A Firebase integrálása viszonylag egyszerű volt, és sikeresen összekapcsoltam egy weboldalt, amely az alkalmazásomban tárolt paklikat jeleníti meg. [19]

A webes felület kialakítása során megjelenítettem a paklik neveit, valamint a bennük található kártyák képeit, így biztosítva az adatok könnyű elérhetőségét és vizualizálását az online tárolásban. Az online tárolás lehetősége lehetővé tette a dinamikus adatkezelést, amely folyamatosan frissíthető az alkalmazáson keresztül.

5.7.2 Árazás bevezetése

Bár a kártyák árait tároltam már az alkalmazásban, ezeket nem jelenítem meg sehol a felhasználó számára. Ennek orvoslására továbbfejlesztettem a paklik menüt, így most már az összes kártya árát is megjeleníti, és egy összesített árat is mutat a pakli teljes értékéről.

19. ábra Példa egy kész paklira

Ezen kívül bevezettem egy funkciót, amely lehetővé teszi a felhasználók számára, hogy egyetlen kattintással feltöltsék paklijukat a weboldali adatbázisba. Ez a gomb mutatja, hogy a paklit már régebben feltöltöttük, vagy még szükség van erre a lépésre. Az új paklik létrehozásakor biztosítottam, hogy a név egyedisége garantált legyen: nem lehet olyan paklit létrehozni, amely már létezik a lokális adatbázisban vagy a felhőben tárolt paklik között, ezzel megakadályozva azt, hogy egy másik felhasználó az véletlen törölje vagy felülírja.

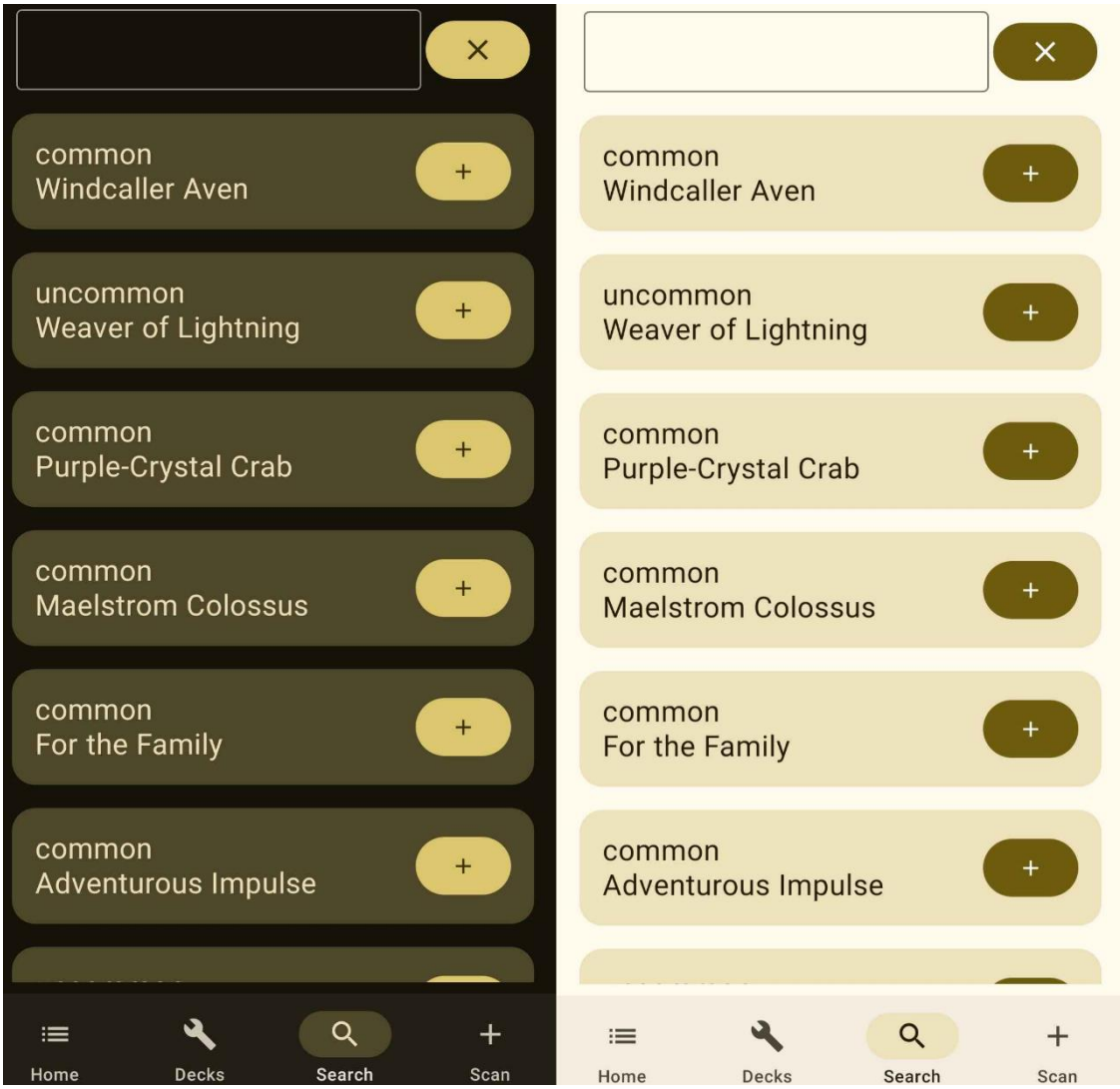
5.7.3 Exportálás

A pakli képernyőjére létrehoztam egy exportálás gombot melynek hatására a teljes listát benne minden megtalálható kártyával adtunk ehhez a paklihoz egy text fájlba kimenti és onnan importálhatja a Magic: The Gathering Arena alkalmazásba, így az saját alkalmazásban tárolt pakli adatok egyszerűen átvihetők a játékplatformram, ha ott már elérhető lapokat tartunk a paklinkban.

5.7.4 Vizuális dizájn

Ilyenkorra voltam úgy az alkalmazással, hogy az előre tervezett funkciók már be vannak építve és itt az ideje, egy vizuális témát is bevezessek. [20]

Ha a felhasználó telefonján a sötét téma van bekapcsolva, az alkalmazás automatikusan sötét módra vált, így biztosítva a kényelmesebb használatot.



20. ábra Sötét és világos témák

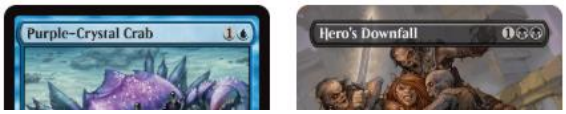
5.7.5 Alkalmazás elnevezése

A Firebase weboldalhoz már egy nevet kellett adnom, így ezt a kitalált nevet a Methodical Magicet bevezettem az alkalmazásba, elneveztem bizonyos elemeit és egy szabadfelhasználású ikont is beállítottam.

- Name: Methodical Magic
- Cards:

- Name: Test deck
- Cards:

- Name: Teszt pakli
- Cards:



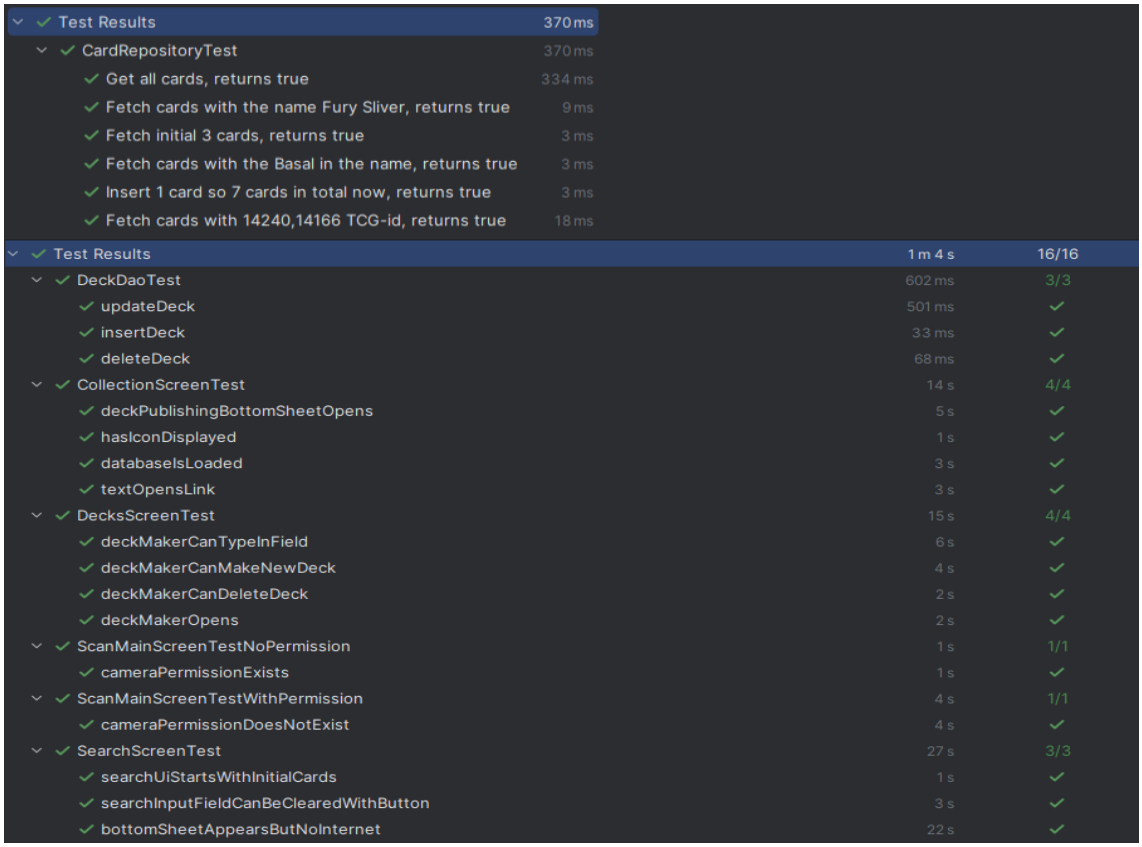
21. ábra Weboldalon a paklik megjelenítése

Egy utolsó körös probléma keresés jött ezután saját telefonomon, emulátoron keresztül, fizikai lapokkal, alkalmazáson belül tárolt lapokkal. Szerencsére ezek során nem sok hibát találtam és publikáltam a Github-ra a kész alkalmazást, tesztelésre készen.

6. TESZTELÉS

A Methodical Magic-re keresztelt kártyadigitalizációs alkalmazásomhoz alapos teszteket készítettem melyek a funkcionalitás, a megbízhatóság és a felhasználói élmény biztosítása érdekében végig mennek a legfontosabb funkciókon és vizuális elemeken. [21]

Különböző teszteseteket terveztem az alkalmazás teljesítményének kulcsfontosságú szempontjaira, az adatbázis-műveletekre, a felhasználói felület összetevőire és az engedélyek kezelésére kiterjedően. Az alábbiakban a tesztelési megközelítés és az eredmények ismertetése következik:



22. ábra Teszteredmények

6.1. Adatbázis és tároló tesztelése

A CardRepositoryTest során olyan kritikus műveleteket vizsgálok, mint a kártyák név szerinti lekérése, az összes kártya lekérése és új kártyák beszúrása. Olyan speciális műveleteket is teszteltek ezzel, mint a kártyák lekérdezése meghatározott azonosítókkal vagy nevekkel, amelyek során megbizonyosodtam a lekérdezési tesztek teljeskörűségéről.

A DeckDaoTest során a paklik kezelésére összpontsítottam, például a frissítési, beszúrási és törlési műveletekre. Minden művelet sikeresen lezajlott; a legintenzívebb művelet, a paklik frissítése, mindössze 501 ms alatt befejeződött. Ezek a tesztek igazolják az alkalmazásban a paklik tárolására és kezelésére szolgáló mechanizmusok megbízhatóságát.

6.2. Felhasználói felület tesztelése

A felhasználói felület tesztelését olyan komponenseken keresztül végezem, mint a képen látható Képernyő-Tesztek (ScreenTest) során olyan interaktív elemeket vizsgáltam, mint az alsó menü lap megnyitása és a szöveges hivatkozások működése, ezzel biztosítva a zökkenőmentes élményt a paklik böngészése és közzététele közben. A DecksScreenTest keretében ellenőriztem, hogy a felhasználók problémamentesen tudnak paklikat létrehozni, törölni és megnyitni; minden eset kevesebb mint 15 másodperc alatt sikeresen lezajlott.

A SearchScreenTest során validálom a keresési funkciókat, például az induló kártyák helyes megjelenítését és a beviteli mezők törlését érintő felhasználói műveleteket. Ezek az eredmények alátámasztják a keresési rendszer megbízhatóságát és gyors reakcióidejét.

6.3. Engedélykezelés

Az alkalmazás engedélykezelési funkcióit a ScanMainScreenTestWithPermission és a ScanMainScreenTestNoPermission keretében teszteltem. Mivel az alkalmazásnak szüksége van a kamerához a szkenner használatához, így, ha a felhasználó ezt nem engedélyezi, akkor ezt a funkciót nem fogja tudni használni. Az eredményekből kiderült, hogy az alkalmazás megfelelően ismeri fel a kamerák engedélyezési állapotait, és azokat az adatvédelmi és működési követelményeknek megfelelően kezeli és szükséges esetén kérvényezi őket az alkalmazás megfelelő működése érdekében.

6.4. Hatékonyság és összegzés

Az összes tesztsorozat során következetesen pozitív eredményeket kaptam, minden eset sikeresen teljesült. Az átfogó tesztelés során emulátoros és nem emulátoros teszteket is elvégzek. Emulátoros tesztekre akkor van szükség, amikor olyan funkciókat akarunk tesztelni, amikhez már maga az Android operációs rendszer elemeire van szükség, például alkalmazás kontextus.

A tesztelési folyamat során biztosítva van a funkciók stabil működése, esetleges hibák/változások a későbbiekben könnyen észrevehetőek Az egységtesztek, integrációs tesztek és felhasználói felület tesztjeinek kombinációjával biztosítottam, hogy az alkalmazás megfeleljen a Magic: The Gathering játékosok digitális igényeinek, és magas színvonalú felhasználói élményt nyújtson.

Tesztcsomag	Tesztesetek	Futási Idő (ms)
CardRepositoryTest	Minden kártya lekérése	334 ms
	Kártyák lekérése a "Fury Sliver" névvel	9 ms
	Kezdeti 3 kártya lekérése	3 ms
	Kártyák lekérése "Basal" névvel	3 ms
	1 kártya beszúrása, így összesen 7 kártya van	3 ms
	Kártyák lekérése a 14240, 14166 TCG-azonosítóval	18 ms
DeckDaoTest	Pakli frissítése	501 ms
	Pakli beszúrása	33 ms
	Pakli törlése	68 ms
CollectionScreenTest	Alsó képernyőlap megjelenítése	5 ms
	Ikon megjelenítése	1 ms
	Adatbázisok betöltése	3 ms
	Szöveges hivatkozás megjelenítése	3 ms
DecksScreenTest	Pakli készítő mezőbe lehet gépelni	6 ms
	Pakli készítő tud új paklit létrehozni	4 ms
	Pakli készítő tud paklit törölni	2 ms
	Pakli készítő tudja megnyitni a paklit	2 ms
ScanMainScreenTest	Kamera engedély létezik	1 ms
	Kamera engedély nem létezik	4 ms
SearchScreenTest	SearchScreen UI indul kezdeti kártyákkal	1 ms
	Keresőmező törölhető gombbal	3 ms
	Alsó lap megjelenik, de nincs internet	22 ms
Összegzés	Összes teszt	1 m 4 s

7. ÉRTÉKELÉS

7.1 Mit tennék másképp

A fejlesztési folyamat során számos olyan tapasztalatot szereztem, amelyek segíthetnek a jövőbeli projektek hatékonyabb tervezésében és kivitelezésében. Bár úgy vélem, összességében jó ütemben haladtam, van néhány terület, ahol utólag visszatekintve más megközelítést választottam volna.

7.1.1 Kihívások a kezdetben

Az egyik legnagyobb kihívást az adatbázis kezdeti beállítása jelentette. Ez a lépés jelentős mennyiségű időt és energiát emésztett fel, így kevesebb lehetőség maradt más fontos funkciók fejlesztésére. Ha újrakezdeném, valószínűleg egy ideiglenes, helyi adatbázist használnék az elején. Egy egyszerűbb, kisebb méretű adatbázis lehetővé tette volna, hogy gyorsabban az alapvető funkciók fejlesztésére összpontosítsak, és így a projekt ütemezésén belül maradt volna elég idő további funkciók beépítésére. Ez a megközelítés lehetővé tette volna az alkalmazás korai tesztelését is, miközben fokozatosan finomítottam volna a végleges adatbázist.

7.1.2 Nehézségek a szkenneroptimalizálással kapcsolatban

Egy másik, bár kevésbé kritikus probléma a szkenner optimalizálása volt. A fejlesztés korai szakaszában sok energiát fektettem abba, hogy a szkenner a lehető leghatékonyabban működjön. Visszatekintve azonban ez nem volt a leghatékonyabb stratégia, mivel a háttéradatok és az alkalmazás struktúrája akkoriban gyakran - néha hetente - változott. Ezek a változások gyakran megkövetelték, hogy újraírjam a szkenner funkcionalitásának egyes részeit, így a korábbi erőfeszítések nagy része elavulttá vált. Egy ilyen dinamikus környezetben célszerűbb lett volna a szkenner optimalizálását az alkalmazás szerkezetének véglegesítése utánra halasztani.

7.1.3 Tanulságok és jövőbeli irányok

Ezek a tapasztalatok alapján a jövőbeni projekteknél nagyobb hangsúlyt fektetnék a fejlesztési lépések sorrendjének rugalmasabb megtervezésére. Az adatbázis kezdeti beállításánál egy ideiglenes, gyorsan megvalósítható megoldással kezdeném, és a végleges struktúra kialakítását az alkalmazás életciklusának későbbi szakaszára halasztanám. Hasonlóképpen, a kritikus komponensek, például a szkenner finomhangolását addig halasztanám, amíg a mögöttes rendszerek stabilak és nem valószínű, hogy változnak.

8. FEJLESZTÉSI LEHETŐSÉGEK

8.1 Paklik importálása

Az alkalmazás kiegészíthető egy olyan funkcióval, amely lehetővé teszi külső forrásból származó paklik importálását. Ez történhet különböző fájlformátumok (például CSV vagy JSON) feldolgozásával, valamint más digitális platformokkal való integráció megvalósításával. Jelenlegi verziója az alkalmazásnak csak exportálni tud viszont hasznos funkció lehet az oda-vissza történő adatcsere tehát az importálás is.

8.2 Bővített felületinformációk

A felhasználói felület további hasznos adatokat is megjeleníthetne, például a kártyák részletes jellemzőit, piaci áruk alakulását az elmúlt pár napban, vagy egyéni megjegyzéseket. Ezek az információk segíthetnék a felhasználókat a kártyák közötti eligazodásban, valamint a gyűjteményük kezelésében.

8.3 Webes felület és azonosítás

Az alkalmazás biztonságának növelése érdekében célszerű egy felhasználói azonosítási rendszer bevezetése. Ez biztosítaná, hogy a felhasználók személyre szabott hozzáféréssel rendelkezzenek, miközben adataik védelme is garantált.

A webes felület vizuális megjelenésének és funkcionalitásának javítása szintén indokolt. Az átláthatóbb és modernebb dizájn megkönnyíteni a másik emberekkel való cseréket és beszélgetéseket a játékról.

Ide tartozóan hasznos lenne az alkalmazásban ideiglenes listák létrehozásának lehetősége is, amelyek cserék lebonyolítását támogatják. Ezek a listák tartalmazhatnák a kártyák aktuális piaci értékét, ezzel segítve a felhasználókat a tárgyalások során.

9. ÖSSZEGZÉS

Úgy gondolom, hogy elégedett lehetek egy számomra új nyelvel és környezett megbirkózva sikeresen elkészítettem egy jó digitalizáló kártyajáték alkalmazást. Ez lehetővé teszi a felhasználók számára, hogy gyorsan és egyszerűen digitalizálják kártyáikat egy szkenner segítségével, így hatékonyabban kezelhetik gyűjteményeiket.

Bár a Magic: The Gathering kártyák személyesen is sokat jelentenek számomra, ezzel kapcsolatos applikációs területen jelentős innovációs lehetőség kínálkozott. Sikeresen kezeltem egy valós igényt, mivel sok gyűjtő számára nehézkes volt kézzel rögzíteni a kártyáik adatait. Erős igény van az ilyen jellegű megoldásokra, és az általam kifejlesztett rendszer hatékony módot biztosít a gyűjtemények digitalizálására, integrált kártyafelismerő rendszerrel és szkennelési funkcióval.

A rendszer fejlesztése során a tanulmányaim szinte minden aspektusát alkalmazni tudtam és egy új programozási nyelv ismeretével és technikákkal lettem gazdagabb.

Az alkalmazás a következő központi összetevőkre épül: egy szkennerfunkció, amely lehetővé teszi a felhasználók számára a kártyák beolvasását és az adatok kinyerését, egy jól strukturált adatbázis, amelyből ugyancsak hozzáadhatják paklikhoz a keresett kártyákat. Az adatbázis pedig olyan információkat tárol, mint a kártyák neve, típusa, állapota és árai. Fontos megemlíteni még a webes felületet, amely lehetővé teszi a felhasználók számára a gyűjtemények egyszerű bemutatását másik játékosok számára.

A szkennelési folyamat egyszerű és gyors, mivel a rendszer képes felismerni a kártyák adatait a beolvasott képekről, és a felhasználó eldöntheti, hogy melyik paklihoz szeretné hozzáadni.

A szakdolgozat során a Magic: The Gathering kártyagyűjtemények digitalizálásának új és izgalmas módját ismerhetik meg. Emellett a szakdolgozat felvázolja a teljes tervezési folyamatot, a kutatástól és a tervezéstől a megvalósításig és az alkalmazásig. Az elkészült rendszer egyszerűségében hatékony, és egy nagyon jó alap és kiinduló verzió egy ilyen program számára.

Irodalomjegyzék

[1] Black Lotus auction - Polygon (2023. március 17.) Forrás: Polygon: https://www.polygon.com/23644519/magic-the-gathering-black-lotus-auction price-2023.

[2] Magic collection organizing - Channel Fireball (2022. január 31.) Forrás: https://strategy.channelfireball.com/channelfireball-library/how-to-organize-store your-magic-collection-best-method.

[3] Android Studio cikk - Developer (2017. szeptember 15.) Forrás: https://www.developer.com/mobile/android/android-studio-android-programming ide-overview.

[4] Optimize options - Android (2023. április 12.) Forrás: https://developer.android.com/build/optimize-your-build#kts.

[5] Eclipse setup - Geeksforgeeks (2022. november 07.) Forrás: https://www.geeksforgeeks.org/how-to-install-and-setup-eclipse-ide-for-android app-development.

[6] Kotlin - Instabug (2023. január 15.) Forrás: https://www.instabug.com/blog/why you-should-learn-kotlin.

[7] Amazon S3 Forrás: https://aws.amazon.com/s3.

[8] Types of Azure storage - Stealthbits (2020. augusztus 25.) Forrás: https://stealthbits.com/blog/types-of-azure-storage.

[9] Azure Blobs Forrás: https://azure.microsoft.com/en ca/pricing/details/storage/blobs/.

[10] AWS pricing Forrás: https://aws.amazon.com/rekognition/pricing/?loc=ft#Free_Tier.

[11] Azure ASCV Forrás: https://azure.microsoft.com/en-us/products/cognitive services/vision-services.

[12] OpenCV Forrás: https://opencv.org/about.

[13] OCR variants - Medium (2021. július 15.) Forrás: https://medium.com/mlearning ai/tesseract-vs-keras-ocr-vs-easyocr-ec8500b9455b.

[14] Scryfall API Forrás: https://scryfall.com/docs/api.

[15] Android Compose Forrás: https://developer.android.com/courses/android-basics-compose/course.

[16] Google ML Kit Forrás: https://developers.google.com/ml-kit/vision/text-recognition/v2.

[17] Splash Screen Forrás: https://developer.android.com/develop/ui/views/launch/splash-screen.

[18] NavController Forrás: https://developer.android.com/guide/navigation/navcontroller.

[19] Firebase Forrás: https://firebase.google.com/docs/android/setup.

[20] Material Theme Builder Forrás: https://material-foundation.github.io/material-theme-builder/.

[21] Android Testing Forrás: https://developer.android.com/kotlin/flow/test.

Ábrajegyzék

[1. ábra] Magic kollekció - Forrás: https://www.mtggoldfish.com/articles/collection-buying-101-where-to-find-collections-rules

[3. ábra] Delver Lens szkenner - Forrás: https://www.delverlab.com/

[4. ábra] Manabox - Forrás: https://manabox.app/

[6. ábra] IntelliSense - Forrás: https://visualstudio.microsoft.com/de/services/intellicode/

[7. ábra] AWS S3 működése - Forrás: https://aws.amazon.com/s3/

[8. ábra] Azure Blobs felépítése - Forrás: https://stealthbits.com/blog/types-of-azure-storage/

[9. ábra] AWS Rekognition felismerő rendszere - Forrás: https://aws.amazon.com/rekognition/

[10. ábra] ASCV szövegfelismerő rendszere - Forrás: https://azure.microsoft.com/en-us/products/cognitive-services/vision-services

[11. ábra] OpenCV - Forrás: https://www.opencv.ai/

[12. ábra] Scryfall - Forrás: https://scryfall.com/

[13. ábra] Pakli lista - Forrás: https://draftsim.com/mtg-arena-import-deck/