

Budapesti Műszaki Szakképzési Centrum Petrik Lajos Két Tanítási Nyelvű Vegyipari,
Környezetvédelmi és Informatikai Szakgimnáziuma

Szakképesítés megnevezése: **Szoftverfejlesztő**
OKJ száma: **54 213 05**

ZÁRÓDOLGOZAT

Light Airlines – Android applikáció

Merényi Miklós
témavezető

Nyíró Levente Gyula
14.S

Budapest, 2020

Nyilatkozat

Alulírott Nyíró Levente Gyula kijelentem, hogy ez a záródolgozat saját tudásom, önálló munkám terméke.

Kijelentem, hogy a záródolgozat bekötött és elektronikus formában leadott példányai mind formátumban, mind tartalomban egyezők, eltérést nem tartalmaznak.

Dátum:

hallgató aláírása

Tartalom

1. Bevezetés	4
2. Témaválasztás	6
3. Fejlesztői dokumentáció	8
3.1. Az alkalmazott fejlesztői eszközök	8
3.2. Adatmodell leírása	9
3.3. Részletes feladatspecifikáció	12
3.3.1. segedOsztaly package	12
3.3.2. kezdoActivity package	13
3.3.3. userActivity package	14
3.3.4. adminActivity package	16
3.3.5. globalActivity package	16
3.4. Tesztelési dokumentáció	17
3.5. Továbbfejlesztési lehetőségek	19
4. Felhasználói dokumentáció	21
4.1. A program általános specifikációja	21
4.2. Rendszerkövetelmények	22
4.3. A program telepítése és konfigurálása	23
4.4. A program használata	23
5. Összegzés	31
Forrásjegyzék	33
Ábrajegyzék	33

1. Bevezetés

A 2 éves szoftverfejlesztői képzésem keretében, az utolsó évben be kell mutatnom egy általam szabadon választott témát, ezt megtervezni és megvalósítani a rendelkezésre álló eszközökkel. Ennek a produktuma olvasható itt most, ahol leírom részletesen a témaválasztásomat, fejlesztésem kitűzött céljait, a megvalósított programom jellemzőit, a felhasználóknak szóló dokumentációt, ahol a belső funkcionalitásokra reflektálok, illetve a fejlesztőknek szánt részletes bemutatást, amelyben kiderül, hogyan is készült az applikáció minden egyes funkciója, ezen belül bemutatva a metódusokat és a programkód részleteit.

A választásom a mobilapplikáció fejlesztésre esett a többi lehetőség közül, ugyanis kíváncsi vagyok erre a technológiára, szeretném magam minél jobban kiismerni, illetve ezen az irányvonalon kívánok továbbhaladni a jövőben. Ezekkel a feltételekkel adott volt a kérdés, hogy Android platformra kívánom készíteni a záródolgozatom.

Vessünk egy pillantást a mobilapplikáció-fejlesztés hasznosságára. A mai felgyorsult világunkban a mobiltelefon egyre inkább, sőt már szerves részünké vált. Ott van velünk a legrövidebb útjaink során, de hosszútávú útitársnak is kiváló. Az ember nem minden környezetben tudja használni a számítógépét, hogy egy szimpla adatra rátaláljon az interneten, ezért volt szükség egy olyan hordozható számítógépre, ami viszonylag kicsi, kompakt, elfér a zsebünkben és a lehetőségek végtelen tárházát tudjuk vele végrehajtani. Ezért is látok nagy potenciált a mobilapplikációkban, ugyanis ezek a legtöbbet használt programok, amik mindig velünk vannak.

A szakdolgozati mobilapplikációm a fentieket figyelembe véve készült. Ez az egész projekt programnyelvet tekintve Java és XML ötvözete, mindez az Android Studio IDE segítségével kialakítva. A design is fontos része az applikációnak, ami főképp az Adobe termékek, azaz Illustrator, Photoshop és XD felhasználásával valósult meg. Az adatbázis részénél az SQLite-ot választottam. Ennek a kezelésére, a kapcsolatok vázolására és a benne szereplő egyes adatok ellenőrzésére a JetBrains által fejlesztett DataGrip nevű programra támaszkodtam, illetve a PhpMyAdmin is sokat segített a munkámban.

A programom lényege egy fiktív légitársaság menetrend keresési, illetve helyfoglalási felület, amelyben lehetőség nyílik a saját profil kezelésére, belépés előtt,

valamint azután is. Járatok között lehet böngészni és információkat megtekinteni, majd az információk alapján foglalást tenni, a felhasználó által kiszemelt ülőhelyre. Az előző lépések után elkészül a „Boarding pass”, azaz a beszállókártya. Ezen a felületen lehet az adatokat megtekinteni, illetve a foglalást lemondani.

Rengeteg kitűzött cél volt a fejlesztésem megkezdése előtt. Mindenképp szerettem volna bizonyítani magamnak, hogy képes vagyok felépíteni egy olyan mobilapplikációt, amelyben nagy hangsúly van a kinézetbe, illetve a funkcionalitásba fektetve a legkifinomultabb módon, és ehhez a jelen szakdolgozat egy kiváló lehetőség volt. Célnak tekintettem az Android Studio rutinos használatát, amelyet a fejlesztés során sikerült elérnem, valamint a Java programnyelvben is sok tapasztalatra tehettem szert. A fejlesztés során nagyon sok problémára bukkantam, majd ezekre sikerült megoldást találnom, amivel folyamatosan nőtt az önbizalmam a programozással kapcsolatban. Mivel egy általam frissen tanult programozási környezethez volt szerencsém, ezért nagyon sok sikerélmény és érdekes kutatás tudott engem érni a fejlesztés során.

2. Témaválasztás

Sok gondolkodás és ötletelés során már az idei tanév októberi hónapjában előkerült egy tervem, hogy egy légitársasággal kapcsolatos applikáció lenne a hozzám illő projekt. Persze volt nagyon sok más ötlet is. Példákat nem említve biztos voltam abban, hogy valamilyen menetrenddel kapcsolatos applikáció megvalósítása volna a célom.

Mivel közel áll hozzám a repülés és az utazás világa, ezért sokszor van szerencsém repülőjegy foglalásokkal foglalkozni, így megihletett a terv, hogy lemodellezem egy légitársaság applikációjának belső rendszerét. Érdekel az ilyen cégen belüli szervezések és adatáramlások. Az a cél is szintén vezérelt, hogy ez az applikáció minél átláthatóbb legyen a rendelkezésemre álló UI design és backend segítségével. El is vállaltam a feladatot, miután benyújtottam az ötletemet.



1. ábra Light Airlines logó

Mérvadó volt számomra a döntés során, hogy minél jobban ki tudjam fejteni mire is vagyok képes az arculattervezés világában. Nagyon sokat kutattam és vizsgálódтам ebben a témában. Megvalósítottam a program arculatát, az activity-kben a háttereket és hozzá az építőelemet, azaz a view-ok kinézetét, hogy minél könnyebben átlátható és használatra is letisztult alkalmazást sikerüljön készítenem. A szakdolgozatomnál lényeges volt, hogy minden elem a saját elképzelésem és alkotásom alapján készüljön el. Így lett az általam kreált fiktív légitársaság neve Light Airlines. Az arculathoz szorosan hozzátartozik egy logó, aminek az végleges verziója az ábrán látható.

Fontos kihangsúlyozni egy ilyen projekt keretében az applikáció funkcionalitását is, ugyanis ez fontos összetevője a programnak, ami nagy mértékben befolyásolja a minőséget is. Figyelmet kell szentelni az alkalmazás logikai felépítésére és az activity-k közötti kohézióra is. Ezzel az elhatározással készítettem el az oldalakat és a kapcsolatokat, hogy minél átláthatóbb és egyszerűbb legyen használni az applikációt egy kezdő felhasználó számára is. Ez is közrejátszott a témaválasztásomban, ugyanis célom

volt megtanulni, miképp lehet egy Android alkalmazás backend funkcióit megvalósítani a legkifinomultabb eszközökkel.

Az eddigi tanulmányaimban webprogramozásban volt szerencsém találkozni a frontend, adatbázis és háttérbeli funkciók megvalósításával, így egy kicsit szkeptikusan, valamint ügyet álltam ehhez a feladathoz Androidon. Szerencsére az elhivatottságom sokat segített a szakdolgozatnál, illetve az ehhez kapcsolódó tudás elsajátításánál.

3. Fejlesztői dokumentáció

3.1. Az alkalmazott fejlesztői eszközök

A program fejlesztésénél az egyik alapja mindennek, a rendelkezésre álló eszközök meghatározása. Tudnunk kell milyen fejlesztői környezet, programnyelv, illetve program kell. Ez a fontos tényező már a programom megtervezése elején eldöntött kérdésnek minősült.

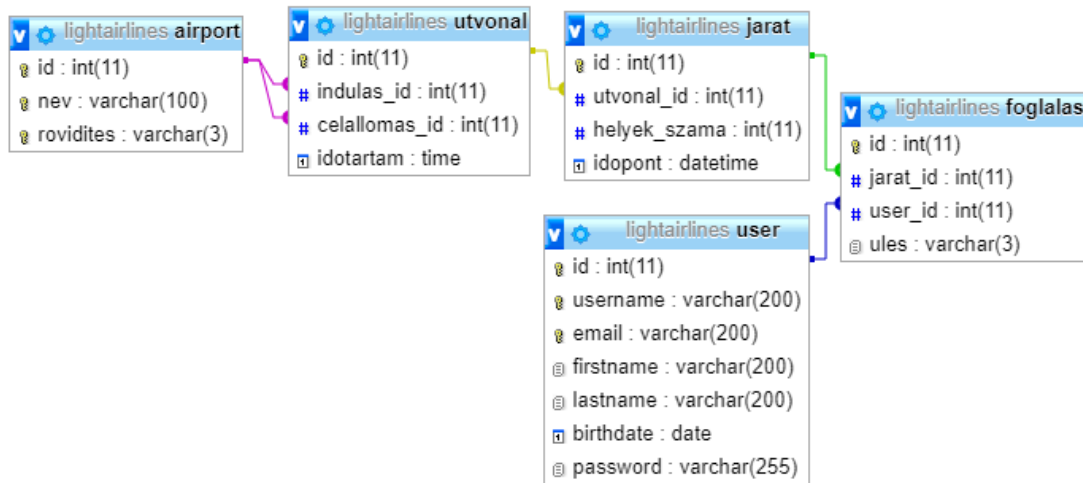
Programnyelv tekintetében a Java-ra támaszkodtam, ugyanis ez volt az a programnyelv, amit a mobilapplikáció fejlesztésben tanultam, valamint a tudásomat is biztosnak érzem ebben a programnyelvben. Az Android programozáshoz tartozik egy leírónyelv, amely főként a frontend elemekre összpontosul, ez ebben az esetben XML.

Fejlesztői környezetnek Android Studio legfrissebb, 3.5-ös verzióját használtam, ugyanis a tanulmányaim során ez volt az egyetlen meghatározó IDE, amivel mobilapplikációkat készítettem.

Az applikáció adatbázisa SQLite alapú, ami lokális szerverként funkcionál az eszközön. Magát az adatbázist Android Studio-ban alakítottam ki, azonban támaszkodtam más programokra is az elkészítéskor és a kapcsolatok meghatározásakor. Egész pontosan 2 szoftverrel éltem: az egyik a JetBrains által fejlesztett DataGrip, a másik a PhpMyAdmin nevezetű program. Itt készítettem el a táblákat, a kapcsolatokat, beillesztéseket, illetve a lekérdezéseket is, majd az SQL kódokat bemásoltam az Android Studio-ba, amivel ki tudtam alakítani a lokális adatbázist, valamint a listázásokat is, ahol erre szükség volt.

Ezekén kívül nagy szükség volt a dizájn elkészítésére, amihez kép-, illetve vektorgrafikus szerkesztőkre kellett támaszkodnom. Ezekhez az Adobe termékeket használtam, így a Photoshop, Illustrator és az XD segítségével sikerült kialakítanom az elvárt eredményt. Az előbb felsorolt programok egy részét tanulmányaim során, egy részét pedig önálló módon sajátítottam el.

3.2. Adatmodell leírása



2. ábra Az adatbázis szerkezete

Ebben az alfejezetben kifejezetten az adatbázisról, annak tábláiról és a köztük lévő kapcsolatokról lesz szó, valamint arról, hogyan lehetett mindezt minél jobban normalizálni.

Két nagy csoportja volt az adatbázisnak a fejlesztés elején: az egyik a *user* tábla, ahol a felhasználók adatai tárolódnak, a másik fő tábla a járatok adatait tartalmazó „járat” tábla. A két tábla közül az utóbbinak a normalizálására volt szükség az adatredundanciák kizárására. Ebből a célból hoztam létre további 2 táblát *utvonal* és *airport* néven. Ezeken kívül létezik még egy tábla, amely a foglalásokat tartalmazza, benne a felhasználó és a járat hivatkozásával. Minden tábla azonosítására egy integer típusú *id* nevű primary key tulajdonságú elem szolgál.

A két alaptábla közül a felhasználók adataival foglalkozó táblának a *user* nevet adtam. Ez a tábla rendkívül fontos a regisztrációnál, belépésnél, azonosításnál, illetve személyre szabásnál. Ide tartozik két unique tulajdonságú elem, és ennek köszönhető, hogy az adott adatú elemből kizárólag csak egy szerepelhet a táblán. Ez a két adatmező a *username* és az *email* nevet kapta. Ezek az adatok a belépésnél kardinális kérdésnek számítanak, ugyanis a felhasználó nevet, vagy az e-mail címet feltétlenül tudni kell a belépéshez. A tábla rendelkezik továbbá a *firstname* és a *lastname* nevű adatmezőkkel, amelyeknek célja, hogy az applikáció személyesebbé legyen téve. A keresztnév például visszaköszön a bejelentkezés utáni activityben, ahol az „Üdvözljük!” üzenet jelenik meg. Az utóbb felsorolt négy adatmező mindegyike varchar típusú és 200-as hosszal

rendelkezik. Létezik egy születési dátumot is letároló adattag *birthdate* néven, amelynek azért van fontos tulajdonsága, mert a regisztrációnál ellenőrzésre kerül, hogy a felhasználó idősebb-e a meghatározott korhatárnál, azaz 13 évnél. Erre az adattagra *date* tulajdonságot állítottam be, ezzel csak a dátumot eltárolva YYYY-MM-DD formátumban. Az utolsó adatmező ebben a táblában a jelszó eltárolására szolgál *password* név alatt. Erre egy 255 hosszú varchar tulajdonságot szántam. Fontos tudni, hogy a jelszó nem eredeti formájában tárolódik el ebben a mezőben, hanem titkosított formában a *salting*-al összevegyítve, így megakadályozva a jelszavak eltulajdonítását.

A másik alap a *jarat* tábla, amely minden egyes járat adatait hívatott leképezni és eltárolni. A normalizálás következménye miatt ez egy viszonylag kevés adatmezővel rendelkező tábla, azonban közvetlenül és közvetetten is kapcsolódik hozzá tábla. Mit is nevezünk járatnak az applikációban? Egy repülőjárat az, amely egy adott útvonalat teljesít egy előre meghatározott időpontban adott számú férőhellyel. Így lett hát az első adattag az *utvonal_id*, ami az előre meghatározott repülési útvonalat határozza meg „A” pontból „B” pontba. Ez az adatmező kapcsolódva van az *utvonal* tábla *id* azonosítójához, ezzel importálva az adott rekord adatait. Ezen kívül van egy másik integer típusú elem, amely a *helyek_szama* nevet kapta. Ez mutatja meg, hogy az adott járaton indított repülőgépen, mennyi a maximálisan elfoglalható helyek száma (ez a szám alapesetben 120). Ez akkor jelenik meg az applikációban, amikor kivonjuk ebből a számból az eddig foglalt helyek összegét és visszaadjuk, hogy mennyi foglalható hely létezik még az adott járat repülőgépén. Az utolsó adattag ebben a táblában egy *datetime* típusú *idopont*, ami megmutatja, hogy a gép pontosan mikor indul. Az adatbázisban ez YYYY-MM-DD HH:MM:SS formátumban van eltárolva, azonban az applikációban való megjelenéskor le van vágva róla a másodperc, ugyanis nincs szükség ekkora pontosságra.

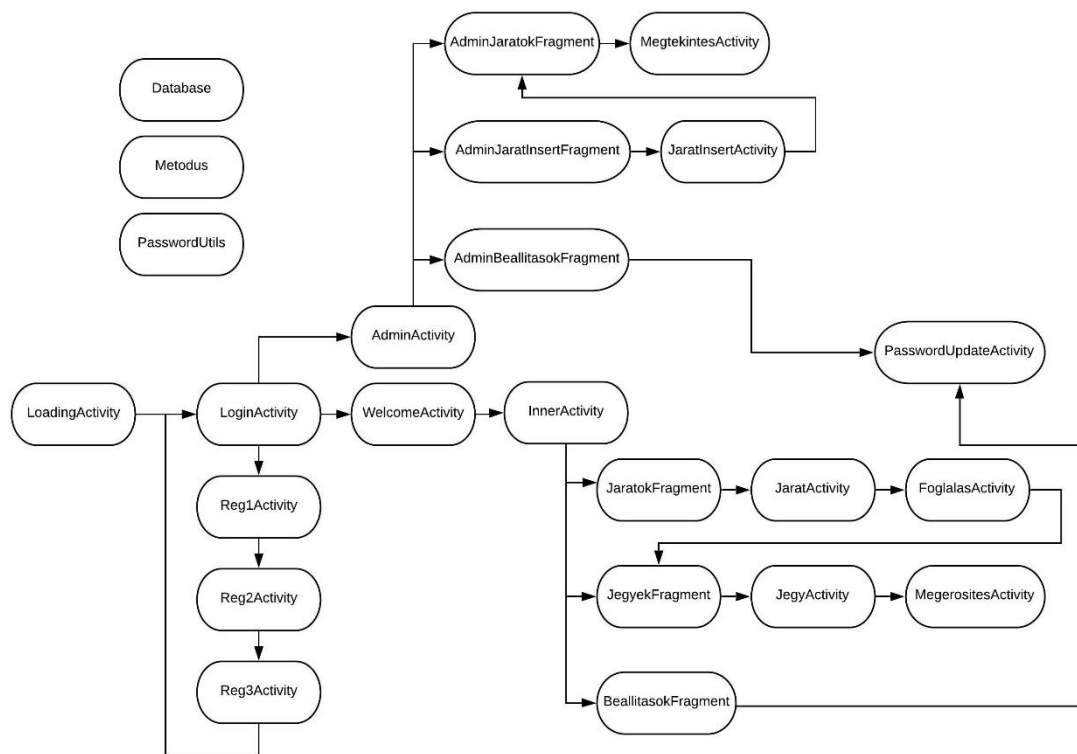
Az előző táblát követi két normalizált tábla, amelyből az első az *utvonal* nevet kapta. Egy útvonal megmutatja, hogy honnan-hova tarthat egy járat és ezt az időt mennyi idő alatt teljesíti egy előzetes becslés szerint. Itt két integer tulajdonsággal rendelkező adatmező is hivatkozik ugyanarra az *airport* tábla azonosítójára. Ez a két adatmező az *indulas_id*, ami az indulási repülőteret, és a *celallomas_id*, ami az érkezési repülőteret azonosítja és határozza meg. Ez egy alapinformációja egy járatnak. Létezik még egy *time* típusú adatmező *idotartam* névvel, amely megmutatja az applikációban, hogy mennyi idő alatt teljesíti az utat a repülőgép. A formátuma HH:MM:SS, azonban az applikációban eltűnik a másodperc, mert itt sem egy mérvadó információról van szó.

A másik normalizált tábla *airport* névvel szerepel. Ez a tábla csak azt mutatja meg, hogy egy repülőtérnek mi a neve és a kódja. Mind a két adatmező unique tulajdonsággal rendelkezik és varchar. Az első adatmező *nev* alatt fut 100-as hosszúsággal, míg a másik *rovidites* névvel szerepel, amelynek 3 karaktert kell tartalmaznia.

Az adatbázisban szereplő utolsó tábla nagyon fontos feladatokat lát el. Ez foglalás névvel rendelkezik és a feladata, hogy megmutassa, melyik felhasználó melyik járatra rendelkezik ülőhellyel. Az első adatmező a *jarat_id*, amely hivatkozik a *jaratok* tábla azonosítójára ezzel megtudva, hogy melyik járatra érvényes a foglalás. Ez akkor lesz fontos, amikor kilistázzuk, hogy hol vannak foglalt ülőhelyek a repülőgépen egy adott járaton. A másik adatmező a *user_id*, amely a *user* tábla adataira hivatkozik annak *id*-ján keresztül. Ennek segítségével kilistázhadjuk egy adott felhasználó jegyeit a járatokra. Az admin felhasználónak is nagy segítség, hogy ki vett jegyet egy adott járaton, bizonyos ülőhelyekre, ezáltal összeállítva egy utaslistát, ami a légitársaságoknak egy rendkívül fontos dokumentum. Az utóbb felsorolt két adatmező mindegyike integer típusú. A harmadik és egyben utolsó adatmező az *ules*, amely megmutatja, hogy az adott foglalás melyik kiválasztott ülőhelyre szól. Ez egy varchar típusú és maximum három karakterrel rendelkező adatmező, ugyanis az adat lehet „8D”, de akár „15B” is. Az utóbbi példa azt jelenti, hogy a 15. sor 2. üléséről van szó.

Minden előbb felsorolt tábla nagyon fontos komponense az applikációnak. Nincs olyan tábla az adatbázisban, amelyre nem lenne még minimálisan se szükség. Mindez teljes mértékben normalizált, így az alkalmazás is bökkenőmentesen és gyorsan képes az adattranzakciókat kezelni.

3.3. Részletes feladatspecifikáció



3. ábra Activity-k közötti kapcsolatok

Ez az alfejezet nagyrészt az applikációban található osztályokról, valamint activity-kről szól, amik ténylegesen meghatározzák, hogyan is nézzen ki a szakdolgozat egésze, ebben szót ejtve mind a backend, mind a frontend témájáról. Mindezt érdemes olyan sorrendben nézni, ahogy haladunk végig az applikációban felhasználóként.

Az osztályok több package-be, azaz csomagba vannak szervezve. Ennek célja az átláthatóság programozás közben. Öt package van: adminActivity, ahol az admin felület osztályai tárolódnak, globalActivity, itt az az osztály található, amely több felületen is használatos, kezdActivity, ahol a regisztráció és a bejelentkezés működik, segedOsztaly, amelyben azok az osztályok találhatóak, amelyek kiszolgálják az activity-ket, illetve a userActivity, ahol az átlagfelhasználói felület activity-jei szerepelnek.

3.3.1. segedOsztaly package

A tervezésnél és a fejlesztésnél szükség volt egy olyan osztályra, amely a kommunikációt és az adattranzakciót végzi az adatbázis és az applikáció között. Erre szolgál a Database osztály, amely SQLiteOpenHelper osztályt hívja meg, hiszen az applikáció SQLite adatbázist használ minden tekintetben. Konstruktora egy Context

tiípusú paramétert vár, amely létrehozza az adatbázist lightairlines.db néven. A meghívott osztály metódusai az onCreate, ahol a táblák adatai valósulnak meg, illetve a másik az onUpgrade: itt a táblák létezése kerül ellenőrzésre. A többi metódus select, insert, update és delete parancsokra vonatkozik, amelyek az activity-kben kerülnek meghívásra.

A második segédosztály a Metodus class. Ennek az elsődleges célja, hogy magába foglaljon, minden olyan függvényt és eljárást, amelyek több helyen is szerepelnek, ezzel is megóvva a programot a kódismétlődéstől. A konstruktorban két osztály kerül példányosításra: Database és Context, azonban csak a Context paraméterét kéri be a program. Sok metódus van ebbe az osztályba belefoglalva, ilyen a dpToPx, ami az osztály által várt pixeles értéket ad vissza a dp megadásával. Ezen kívül még található felhasználóhoz kapcsolódó ellenőrzés, a hosszával, valamint, hogy nem található-e benne whitespace, azaz üres karakter. E-mail címmel kapcsolatban a helyessége kerül ellenőrzésre. Jelszónál itt található maga a jelszóellenőrzés folyamata, illetve a regisztrációnál található feladat, amely a jelszó erősségét hívatott vizsgálni. A keresztnév, vezetéknévvel kapcsolatban az elsoNagybetu metódus szabja formára a helytelenül írt neveket. Található itt még idő átalakítás, valamint üléssel és foglalással kapcsolatos metódus is.

Az utolsó osztály ebben a packageben a PasswordUtils osztály. Itt történik a jelszó titkosítás a getSalt, hash és generateSecurePassword segítségével, illetve belépéskor a visszafejtés, ezzel validálva a jelszó érvényességét a verifyUserPassword felhasználásával.

3.3.2. kezdActivity package

Az activity-k csoportosítása úgy történt, hogy bizonyos funkciócsoportok, el legyenek egymástól szeparálva. Így meg kellett határozni, hogy a belépéskor megjelenő activity-k hova kerüljenek besorolásra, illetve meddig tart ez a felületcsoport. Így lett végül pontosítva, hogy a kezdő funkciócsoport a betöltő képernyőtől egészen a bejelentkezés véglegesítéséig tart. Ezekhez az activity-khez már XML fájl is tartozik, amelyek a UI-t szolgáltatják.

Sorrendben az első, amivel találkozunk az a betöltő képernyő, ami a LoadingActivity nevet kapta. Ezzel az activity-vel kezd maga az applikáció, majd 1,3 másodperc elteltével, amit a Thread osztály késleltet, Intent osztály segítségével átléptet

a LoginActivitybe. Az átléptetésig egy animációt látunk a képernyőn, ami a LottieAnimation felhasználásával történt meg.

Ezután a LoginActivityben találjuk magunkat. Itt 4 fontosabb komponens található: a két EditText, valamint két Button. Az egyik gomb segítségével elindítjuk a bejelentkezési procedúrát, ellenőrzéseket, ha minden adat egyezik átkerülünk vagy az adminActivity packagebe, vagy a WelcomeActivitybe, ami az átlagfelhasználók felülete. A másik gombbal a regisztrációs részben találjuk magunkat.

Három RegActivity osztály létezik. Itt lehet lépésenként az adatokat megadni a sikeres regisztrációhoz. Minden ilyen felületnél 3 lehetőség létezik: a továbblépés, a visszalépés, illetve a van már profilom, aminek a segítségével visszatérhetünk a LoginActivitybe. Az elsőnél a felhasználónevet és az e-mail címet kell megadni. A tovább gombbal a metódusok ellenőrzik, hogy biztos nem létezik még egy ilyen adat, valamint, a Metodus osztály segítségével biztos elég hosszú-e a felhasználónév és helyes-e az e-mail cím. Csak igaz esetekben léptet tovább a program a Reg2Activitybe.

A második regisztrációs felületen a keresztnév, vezetéknév, illetve születési dátum megadása kell. A program figyel arra, hogy 13 éven aluliak ne legyenek képesek a regisztrációt elvégezni. A továbblépésnél a nevek átalakulnak úgy, hogy az első betű minden esetben nagy legyen.

Az utolsó regisztrációs osztályban a jelszó kerül bekérésre, mégpedig kétszer. A metódusok ellenőrzik, hogy egyeznek-e, valamint a hosszúság és a benne lévő karakterek is fontos körülmények a kontroll szempontjából.

A legutolsó osztály a packageben egy WelcomeActivity, ami akkor kerül elindításra, ha a felhasználó sikeres azonosítást tett. Itt megjelenik egy üdvözlő üzenet a felhasználó keresztnévvel, amely kapott egy fade animációt, majd 1 másodperc elteltével átkerülünk a userActivity packagebe.

3.3.3. userActivity package

Valaki akkor kerül be ezekbe a felületekbe, ha az azonosítás alapján átlagfelhasználónak minősül. Itt található rengeteg olyan felület, amelyeknek a lényege, hogy a felhasználó elérje a céljait az applikáció használatával. Tulajdonképpen ez a csoport adja a mobilalkalmazás nélkülözhetetlen részét.

Ebben a packageben az első felület az InnerActivity, azonban ez önmagában csak egy üres felület, erre ugyanis rájönnek a fragment-ek, mivel ez egy alsó navigációs területtel (bottom navigation) rendelkező felület. Az InnerActivity szolgál a kilépés megerősítésére, amire a választ a felhasználótól várja, valamint ez kezeli, hogy éppen melyik fragment váljon láthatóvá. Három fragment van: Jaratok-, Jegyek-, BeallitasokFragment. Az elsőben az olyan járatok láthatók, amelyre a felhasználónak még nincs jegye és a jövőben fog indulni a járat (a múltbéli járatok nem kerülnek kilistázásra). Érdekeség, hogy itt a kártyákat az osztályban hozza létre a program és nem az XML fájlban, erre azért volt szükség, mert nem tudjuk pontosan hány járatot fog kilistázni az adatbázis. Átlépve a JegyekFragment-ekbe láthatók az eddig foglalt helyek adott járatokon. Ez is szintén kártyás stílusban lett elkészítve az osztályban. Az utolsó fragment, azaz a beállításoknál személyre szabhatók a felhasználó adatok. Egy gomb megnyomásával szerkeszteni lehet az EditText-eket, majd a megfelelő ellenőrzések mellett menteni lehet, ha minden helyes. A „Mégse” gombra kattintva elvetjük a változtatásokat. Egy metódus segítségével meg kellett oldani, hogy a felhasználónév és az e-mail menthető legyen, ha nem létezik, illetve, ha eddig a felhasználó birtokában állt. Itt található még egy gomb, ami a PasswordUpdate osztályba vezet át, azonban ez egy másik packageben található, így erről később lesz szó.

A fragment-eken kívül léteznek még bőven felületek, amelyek adatokat képesek közvetíteni. Az első ilyen a JaratActivity, ami akkor jelenik meg, ha a JaratFragment-ben kiválasztunk egy adott járatot. Itt megjelennek részletesebb információk a járatról, valamint a foglalás is elérhetővé válik. Innen továbblépve a FoglalasActivity található, ahol az üléseket az osztályban volt célszerű létrehozni. Egy ülést ki tudunk jelölni, ha az nincs lefoglalva. Amennyiben nem választottunk ülést az ellenőrzés gondoskodik róla, hogy figyelmeztetést tegyen. Ez az osztály főként a Metodus osztály elemeit használja, azaz lekéri a lefoglalt helyeket és listában visszaadja azokat, illetve fordítja is a helyek számát, ugyanis a program int-ként kezeli, míg az adatbázisban, más formátumban, azaz String-ként jelenik meg (8-as hely: 2B – második sor, második ülése). A foglalás véglegesítésével az insertFoglalas metódus rögzíti az adatokat az adatbázisba.

A JegyActivity segítségével megtudhatjuk hova és mikorra szól a járatunk, illetve egy QR kód is található ott, ami funkciókat nem szolgál, jelenleg ez még esztétikai elem. Amikor ez megnyitásra kerül a képernyő felvilágosodik és így a későbbi leolvasást is megkönnyíti ez a folyamat. A foglalás lemondására is képesek vagyunk: az erre

hivatkozó gomb megnyomásával az applikáció áthelyez a MegerositesActivity-be, ami azt a célt szolgálja, hogy jelszóval megerősítsük a jegylemondási szándékunkat. Ez megvéd attól, hogy jogosulatlan felhasználó foglalást töröljön, illetve a figyelmetlen felhasználót is segíti. Ezután kerül sor a jegytörlésre, majd a felhasználó visszakérül az InnerActivity-be.

3.3.4. adminActivity package

Ha az azonosítás után kiderül, hogy a felhasználó valójában admin jogokkal rendelkezik, akkor megjelenik előtte az AdminActivity, ami hasonlóan épül fel, mint a userActivity package, így ez is bottom navigation tulajdonságokkal bíró activity, amelyben fragment-ek találhatók. Az AdminJaratokFragment-ben a járatok listázódnak ki, hasonlóan a felhasználóknál megjelenő fragment-nél. Itt viszont egy járat kiválasztásakor a MegtekintesActivity-t nyitjuk meg, amiben a foglalt helyek és rányomáskor a foglaló adatai jelennek meg, ezzel elkészítve egy utaslistát a légitársaságnak. Innen már csak vissza tudunk menni az előző fragment-re. A másik fragment neve AdminJaratInsertActivity, ahol megjelennek a lehetséges útvonalak, amelyre a légitársaság járatot szervez. Ennek kiválasztásával továbblépünk a JaratInsertActivity-be, ahol ki tudjuk tölteni, hogy a leendő járat melyik nap, melyik időpontban induljon. Ezt a timeToString metódus átalakítja, majd az insertJarat eljárás rögzíti. Az utolsó itt található fragment az AdminBeallitasokFragment, ahol csak a jelszót lehetséges változtatni, a „Jelszómódosítás” gomb megnyomásával. A másik lehetőség a „Kijelentkezés” gomb, ami előhívja az AdminActivity AlertDialog-ját, hogy az admin biztos-e a kilépésében.

3.3.5. globalActivity package

Ebben a package-ben szerepel az az activity, amely több helyről is meg lett hívva. Innen bármelyik package bármelyik osztálya meg tudja hívni ezt a felületet. Itt csak egy activity szerepel, mégpedig a PasswordUpdateActivity, amely a jelszó megváltoztatására szolgál. Több helyen is meghívásra került: a userActivity és az adminActivity package beállítási részéről is elérhető. A régi jelszó ellenőrzésének a folyamata a Metodus osztályban történik. Itt valójában, csak az új jelszó titkosítására és rögzítésére kerül sor. Majd visszalépünk arra a felületre, ahonnan jöttünk, ugyanis az admin felhasználónak is itt módosul a jelszava, az alap felhasználó mellett.

3.4. Tesztelési dokumentáció

A program írása közben és után is rengeteg tesztelés történt, illetve az esetlegesen felmerülő hibákat azonnal orvosolni is kellett. A végső tesztelésnél a program megfelelt minden követelménynek a kezdeti specifikációban foglaltak szerint. Ezen kívül már csak az volt hátra, hogy mindezt egy fejezetbe foglaljam.

Az első fontos funkció, amit tesztelni kell az a regisztrációnak a lépései. Először a felhasználó nevet és az e-mail címet szükséges megadni. A rendszer kiszűri az 5 karakternél rövidebb felhasználó neveket, valamint ellenőrzi a felhasználónév eddigi létezését. E-mail cím esetében a maszkot ellenőrzi: „@” előtt és mögött szöveg, egy pont karakter követi mindezt, ami után egy domain következik. Természetesen ebben az esetben is ellenőrzésre kerül az egyedi adat elérhetősége az adatbázisban. Szélsőséges vagy hibás esetben a program vörössé változtatja a hibás mezőt és kiírja mit vett hibának. Tovább lépve a következő mezőkre a keresztnévet, vezetéknévet, illetve a születési dátumot kell megadni. A hibalehetőségek kizárása érdekében bármilyen nevet beírhat a felhasználó, azonban a lezárást követően az első betű kapitálissá válik. Mivel 13 éven aluliak nem regisztrálhatnak, így értelemszerűen erre hibát fog dobni a program és a következő regisztrációs felületre való lépés semmissé válik. A harmadik, egyben utolsó felületen a jelszó megadására van szükség. Ahhoz, hogy a felhasználó biztos legyen a jelszavában, így ezt kétszer kell bekérni. Ha valami nem egyezik, vagy a jelszó túl rövid, illetve nem tartalmaz minimum egy nagybetűt és egy számot, a rendszer hibát dob, a jelszó eltűnik a mezőből, maga a mező vörössé válik, illetve a regisztráció véglegesítése lehetetlen lesz mindaddig, míg a két jelszó a feltételeknek meg nem felelnek.

A belépésnél is meg van oldva, hogy semmilyen hiba ne léphessen fel. Az első mezőnél meg lehet adni felhasználónevet, illetve e-mail címet egyaránt. Ha ezek az adatok az adatbázisban nem léteznek, teljesen mindegy, hogy milyen jelszót adott meg a felhasználó, az azonosítás sikertelenné válik. A jelszó helytelen megadása is ugyanazt a hibát dobja fel, mintha az első mező lenne helytelen. Mind a két hiba esetében a két mező vörössé válik, a jelszó mezőben a szöveg kitörlődik, illetve a hibaüzenet csak a sikertelen belépésről ad tájékoztatást. Ennek egy célja van: ne lehessen tudni, hogy milyen felhasználónévvel, illetve e-mail címmel vannak regisztrálva felhasználók egy esetleges kibertámadás esetén. Maga a jelszó is egy sajátos titkosítással van ellátva és így eltárolva

az adatbázisban, belépésnél ez visszafejthetetlen, csupán a szózás és a titkosított forma ad utalást arra, hogy a jelszó egyezik a mezőben megadott string-gel.

A belső felhasználói felületeken a helyfoglalásnál kellett megoldani, hogy egy foglalható járat csak abban az esetben jelenjen meg, amennyiben az még nem múltbeli, illetve a felhasználó nem tett rá még foglalást (ugyanis minden felhasználó egy helyre foglalhat). Ennek a lekezelése megtörtént és a program a kívánságnak megfelel. A keresés is kiválóan visszaadja azokat az eredményeket, amikre a felhasználó számíthat. A helyfoglalásnál a felhasználónak kötelező kiválasztani egy ülőhelyet és oda nem ülhet, ahol már más megtette a foglalását.

A tesztelés során sikerült egy problémát is kijavítanom. Ezt a hibát a honnanhova szűrésnél vettem észre. A lényeg annyiból áll, hogy a lekérdezés ezen szakaszába az adatbázis műveletet összezavarja a félidézőjel, így ezt ki kellett szűrni a gépelés közben. Miközben a program folyamatosan figyelte, hogy a felhasználó mit ír be, innentől kezdve tiltva lesz számára ennek a karakternek a begépelése, ezzel megoldva a problémát.

A jegyekhez térve a felhasználó hiba nélkül megtalálja az összes lefoglalt jegyét, amikre rányomva megtekinthet minden részletet hibátlanul a legmegfelelőbb felbontásban.

A beállításoknál a felhasználó az adatmódosításnál minden mezőt átírhat. Azonban kritériumokra figyelnie kell, amit a program hibaüzenetekkel azonnal jelez egy mentés során. A felhasználónévre és az e-mail címre megvannak a fentebb említett protokollok, azonban arra ügyelni kellett, hogy ugyanazt a felhasználónevet és/vagy e-mail címet megadhassa, amivel eddig rendelkezett. A jelszómódosításba lépve pedig elsősorban a régi jelszónak kell érvényt szerezni, majd az újat kétszer megadni, hogy minden hibalehetőség kizárásra kerüljön.

Egy-két helyen a program a megerősítés felületre lép. Ennek azért van értelme, hogyha a készülék feloldva marad, illetve a felhasználó bejelentkezve maradt, például egy esetleges jegytörlésnél a jelszával véglegesítse a döntését. Ennek köszönhetően kiszűrhetők az akaratlan módosítások és műveletek.

A program készítésénél ügyelni kellett, hogy minden elem az alkalmazásban rezponzív legyen, azaz bármilyen eszközön minden adat ugyanúgy megtalálható legyen. A tesztelésnél főként kétféle eszközt használtam. Az egyik egy Samsung A50 volt, amely 19.5:9-es felbontással rendelkezik. Miközben fejlesztettem erre a mobiltelefonra, fontos

volt odafigyelni, hogy mindez egy 16:9-es készüléken is tökéletesen jelenjen meg. Ezen a készüléken minden tökéletesen működött, főleg erre a teljes képernyős verzióra készítettem az applikációt és ez adott nekem egyfajta iránymutatást, hogy a dizájnt tekintve hogyan is haladjak tovább.

A tesztelés egy Samsung Note 10 táblagépen is megtörtént. Azt tudni kell, hogy ez a készülék egy alacsonyabb API szinttel rendelkezik. Alapból a program minimum elvárt szintje a 17-es, de a lényeges elemek inkább az új készülékekre vannak szánva. Az elemek kiválóan megjelennek, azonban megjelent az applikáció egyetlen hibája ezen a készüléken. Maga a Bottom Navigation Bar egy ismeretlen hiba miatt használhatatlan. A sikeres belépés után megjelennek a kártyák, azonban az alsó navigációs panel a kártyák alá kerül. Amikor ez a panel előbukkan a fragment-ek közötti váltás lehetetlenné válik, ugyanis az ikonok semmit nem reagálnak a megnyomásukra. Próbáltam kijavítani ezt a hibát, de sem önerőből, sem pedig az internet használatával nem sikerült mindezt megoldani, így elkönyveltem ezt az alkalmazásom egyetlen súlyos hibájaként.

A harmadik tesztelési felület egy emulátor volt, amely Android Nougat-al volt ellátva és 16:9-es képernyővel rendelkezett. Ez nagy segítség volt abban, hogy biztos legyek abban, hogy erre a felületre is kiválóan alkalmazkodik a megvalósított alkalmazásom. Részletesen teszteltem minden lehetőségét a programnak ezzel a régebbi Android verzióval és az előbb említett táblagéphez képest még az alsó navigációs panel is tökéletesen működött.

Ezekkel a módszerekkel sikerült kijavítani a hibalehetőségek számát, bár a program még így se 100%-os, de ez a feladatrész az applikációnak már a karbantartási szakaszára vonatkozik.

3.5. Továbbfejlesztési lehetőségek

Az elkészített program önmagában elégnak minősülhet, de valójában rászorul még egy-két továbbfejlesztés, hogy a mai világban egy releváns és modern applikációnak minősüljön. Értem itt például egy globális adatbázis felállítását a jelenlegi, lokális SQLite-al szemben.

A globális adatbázisra több koncepció is volt. Az első egy MySQL csatlakozás, azonban ehhez API hívások kellenek, amik részben bonyolultabbnak számítanak, mint a másik lehetőség, amit Firebase-nek hívnak. Ez egy Google által üzemeltetett online

adatbázis, amiben lehet kezelni az autentikációkat (felhasználók azonosítását), illetve az azonnali idejű adatbázist. A program specifikációban szereplő biztos verziója után, elkezdtem GitHub-on egy külön branch készítésével a FireBase kiépítését, az SQLite jövőbeli leváltására. Ez valameddig sikerült is. Az autentikáció és a bejelentkeztetés készen is áll, azonban a probléma a járatok listázásánál kezdődött. A lekérdezés tökéletesen megy, kereséssel együttműködve, azonban több kereséshez fűződő feltételt nem lehet már megadni, ami a nehézséget okozta ebben a feladatban. A határidő közeledte miatt így ezt a feladatot félretettem a későbbi alkalmakra, egyfajta potenciálnak.

Egy másik lehetőség, amit a későbbiekben be lehetne illeszteni az az új ülésrend hozzáadása repülőgéptől függően, ugyanis jelenleg csak egy 120 ülőhellyel rendelkező jármű van forgalomban. Azért készült csak ez a típus, mert más esetben újra kell tervezni az osztályt, amennyiben felületről tápláljuk be, hogy az adott járat mennyi eladható hellyel rendelkezik.

Ott volt továbbá az első- és a turistaosztály lehetősége, azonban, hogy ez szembetűnő lehessen egy bizonyos pénzért kell árusítani a helyeket. A rendszernek ehhez ki kell építeni egy olyan lehetőséget, hogy pénzt kérhessen be és azt később a felhasználó a saját keretéből megvásárolhassa.

Volt egy vízió, a több helyre való foglalás. Ez egy azért nehéz dolog, mert ha több helyet szeretne egy felhasználó foglalni, abban az esetben meg kell adnia a többi felhasználó adatait. A többi felhasználó adatainak a valóságát viszont vissza kell igazolni és ez is egy plusz fejlesztés lenne ebben a projektben.

Az utolsó lehetséges továbbfejlesztési lehetőség az az volna, ha az admin felületen, nem csak előre meghatározott útvonalon lehetne járatokat meghatározni, hanem új útvonalakat, ezzel együtt új desztinációkat létrehozva. Ehhez egy nagyon komoly adatbázis kell minden repülőtérről, illetve az ellenőrzést is meg kell határozni a programban, hogy ne lehessen két ugyanazon repülőteret megadni.

A fent említett fejlesztések nem telnének rengeteg időbe, azonban nem volna elég ahhoz, hogy a programot határidőn belül tudjam teljesíteni. Az egyértelmű azonban, hogy a szoftver tele van végtelen továbbfejlesztési lehetőségekkel és valószínű, hogy érdemes is lenne a jövőben vele foglalkoznom, ugyanis ez is egy kulcsa a fejlődésemnek ebben a programozási ágban.

4. Felhasználói dokumentáció

A jelen fejezet lényege, hogy egy útmutatást, illetve segítséget nyújtson a program leendő felhasználójának a programban jelenlévő összes funkció elsajátításában. Szó lesz itt a program általános leírásáról, az elvárt rendszerkövetelményekről, a program telepítéséről, illetve annak használatáról több oldalon taglalva mindezt.

4.1. A program általános specifikációja

Fontos feladatnak számít a fejlesztés előtt, alatt és után is gondolkodni azon, hogy az alkalmazás kinek is készül igazán és miképp lehet elmagyarázni egy átlagos felhasználónak, hogy neki ezt érdemes, sőt feltétlen használnia kell. Nos ebben a részben pontosan erről lesz szó.

Mivel egy légitársaságról beszélünk, így szükség van egy felületre, ahol a jegyfoglalások, illetve az egyéb műveletek megtörténnek. A mai világban több lehetőség van egy felhasználónak egy repülőjegyet megvennie. Weboldalon keresztül, illetve mobilapplikáción is meg tudja ezt tenni. Azonban a fizikai érintkezés, azaz jegyirodákban való vásárlás egyre jobban szorul ki a trendből. Ez az ok, amiért feltétlenül meg kell teremteni a lehetőséget a felhasználónak, hogy mindezt okostelefonon is megtehesse. Amennyiben mindezt tényleg az utóbbi lehetőségen tenné meg, arra az esetre szól neki ez a fejezet, hogyan is használja ezt az applikációt. Maga az applikáció régiótól függően elérhető magyar, angol, valamint német nyelven is.

Ez az alkalmazás elsődlegesen a légitársaság járatainak böngészéséről, foglalásról és a jegy felhasználásáról szól. A regisztráció, illetve bejelentkezés után a felhasználó átkerül egy olyan felületre, ahol a járatokat tekintheti meg, ebből szűrhet indulási és érkezési repülőtér alapján, majd kiválasztva valamelyik kártyát megtekintheti annak részletes információit. Továblépéssel helyet tud foglalni magának, majd a program minden adatot összevegyítve generál egy virtuális beszállókártyát. Mindezt a felhasználó a jegyek között böngészve megtalálja, majd kiválasztva az egyik jegyet, betekintést nyerhet a felületre, ahol egy QR kóddal van azonosítva maga a foglalás. Ezt a jegyet le is tudja mondani egészen az indulásig. A jegy alapból eltűnik az indulás után, ahogy az adott járat is. A felhasználó képes továbbá felhasználói beállításokat is

végrehajtani. Be tudja állítani az új felhasználónevét, e-mail címét, vezeték- és keresztnévét, illetve jelszót is tud cserélni.

Admin típusú felhasználó esetében másféle funkciók vannak értelemszerűen. Itt meg lehet tekinteni a jelenleg meglévő járatokat, illetve az utaslistát a helyet foglaló utasokról és alapadataikról egy adott járaton. Ezen kívül van még egy járat hozzáadás funkció, amivel az előre meghatározott útvonalak közül egyet kiválasztva lehet létrehozni az adatbázisba, ezáltal megjelenítve új járatként a felhasználóknak.

4.2. Rendszerkövetelmények

Sok vizsgálatra és némi tesztelésre volt szükség, hogy körül lehessen határozni milyen hardveres, illetve szoftveres rendszerkövetelményekre van szükség a program tökéletes működéséhez. Szerencsére mindezt sikerült végrehajtani: A program már a minimum rendszerkövetelményekkel is képes hozni azt az áttekinthetőséget, mint amit az ajánlott követelmény nyújt.

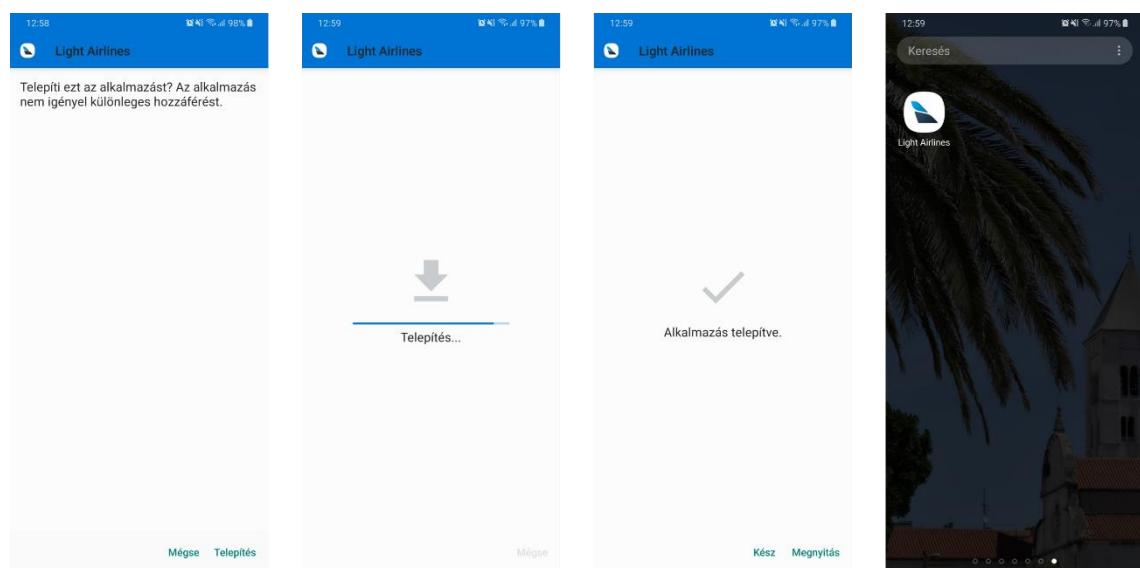
	Minimum	Ajánlott
Hardverkövetelmények	600 MHz-es processzor	1.5 GHz
	512 MB RAM	2 GB RAM
	20 MB szabad tárhely	
	1080x1920-as felbontás	
Operációs rendszer	Android 4.2 vagy újabb	Android 8.0 vagy újabb

4. ábra Rendszerkövetelmények

4.3. A program telepítése és konfigurálása

A program első lépéseként rendelkezni kell a telepítőfájllal, aminek a kiterjesztése .apk. Ezt a letöltések mappájába találjuk meg az Android fájlböngészőjében. A fájl megnyitásával egy kérdést tesz fel a program, hogy kívánjuk-e telepíteni a programot. Ezután elindul egy folyamat, ami alatt a fájl telepíti a programot a telefonra, parancsikont hoz létre, majd minden szükséges kelléket kialakít hozzá. Az egész folyamat maximum fél percet vehet igénybe, természetesen ez függ a készülék sebességétől. A telepítés végén kiléphetünk a telepítőprogramból, illetve más esetben megnyithatjuk a kész alkalmazást, amennyiben nem akarjuk keresgélni a programot a telefon ikonfelületén.

Belépve az applikációba találkozunk az első felülettel a betöltés után. Ez a bejelentkezés. Mivel új a program a készüléken, így még nem rendelkezünk saját profillal, ezért kell létrehozni a regisztrációnál egy saját felhasználói profilt. Ennek a lépéseit a 4.4-es fejezetben lehet megtalálni.



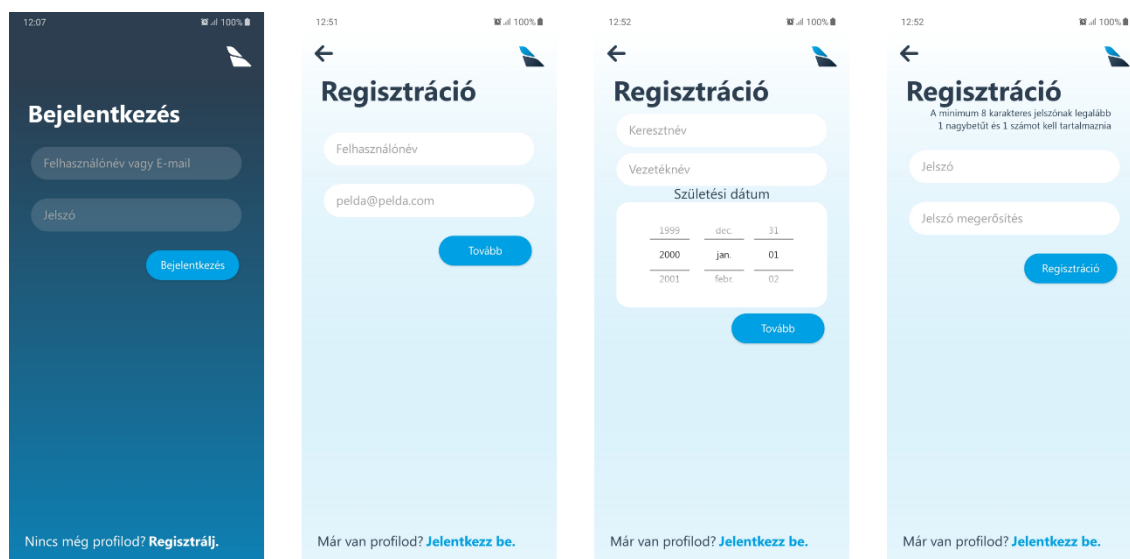
5. ábra Telepítés menete

4.4. A program használata

Ebben a fejezetben kifejezetten a program használatát fogom leírni lépésről-lépésre. Szó lesz arról, hogy melyik lapon, milyen lehetőségei vannak a felhasználónak, illetve az adminnak.

Amikor a készüléken megnyitjuk az applikációt, meg fog jelenni egy betöltőképernyő, ahol maximum 2 másodpercig tart a betöltés. Itt csak várakozni kell, majd ezután máris megjelenik a következő oldal: a bejelentkezési felület.

A bejelentkezési felületen rengeteg lehetőséget láttunk, ám mégis egyszerű összeszedni mindegyiknek a funkcióját. A beépített vissza gombra kattintva az applikáció rákérdez, hogy tényleg kíván-e a felhasználó kilépni. Igenleges válasszal megerősíthető ez a döntés. Amennyiben még nincs felhasználói fiókunk az alul lévő félkövér „Regisztrálj” gombra nyomjunk.



6. ábra Bejelentkezés és regisztráció

A regisztrációs felületen a leendő felhasználónevet és e-mail címet lehet megadni. Fontos tudni, hogy mindkét adatnak kritériumoknak kell megfelelnie. Egyik mező se maradhat üresen. A felhasználó név nem lehet 5 karakternél rövidebb, nem tartalmazhat üres karaktert, illetve egyedinek kell lennie, azaz, ha valaki más is használja ezt a felhasználónevet, ki kell találni egy újat. Az e-mail címmel hasonló a helyzet – egyedinek kell lennie – ezen kívül meg kell felelnie a sablon e-mail címeknek: „@”-t és domain címet (például: .hu) és a kettő között, valamint a „@” előtt szöveget kell, hogy tartalmazzon. A domain címnek meg kell felelnie a nemzetközi normáknak, így az minimum 2 hosszúságú lehet. Segítségként sablont is lehet találni ebben a mezőben (pelda@pelda.com). Amennyiben mindent kitöltöttünk a „Tovább” gombra kattintva minden kitöltött mező ellenőrzésre kerül. Ha valami nem felel meg a fent említett elvárásoknak, azt a program egy üzenettel és a hibás mező vörösre váltásával jelzi. Ezen a felületen, ahogy az összes többi regisztrációs felületen találunk egy vissza nyilat, illetve

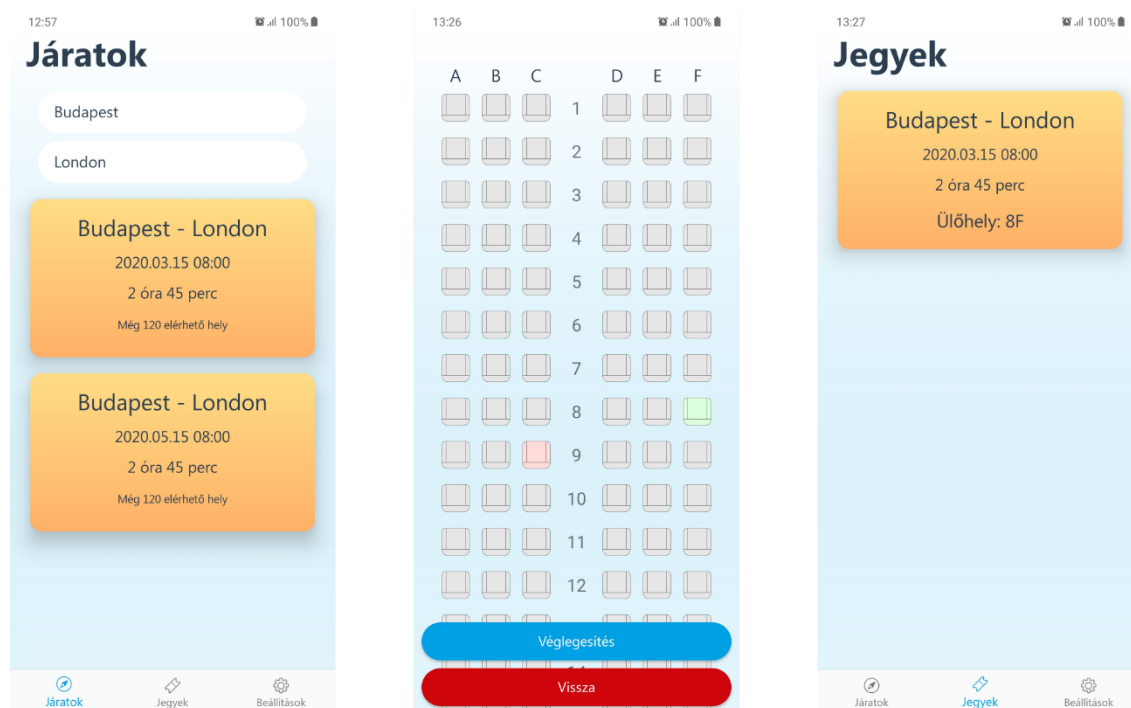
egy „Jelentkezz be” gombot alul, valamint a beépített vissza gombot is. Mindhárom lehetőséggel visszatérünk a bejelentkezési felületre.

Ha az első regisztrációs felületen minden elem sikeres volt, áttérünk a második regisztrációs oldalra. Itt egyszerűbb a feladat. A keresztnévet, vezetéknévet, illetve a születési dátumot kell megadni. Fontos, hogy minden mező ki legyen töltve. Nem kell attól aggódni, ha a nevünket kis betűvel írjuk, az applikáció gondoskodik róla, hogy a név helyesen (első betű nagy, a többi kisbetű) legyen eltárolva. A születési dátumnál a választó segítségével tudjuk kiválasztani a helyes dátumot. Fontos tudni, hogy az applikációt 13 éven aluliak nem használhatják az internetes biztonságpolitika miatt. Egy ilyen kitöltésnél erre a program figyelmeztetést is fog adni, és nem lehet a következő regisztrációs felületre térni. Az itt található nyíllal, vagy a beépített vissza gombbal az előző felülettel ellentétben ez az előző regisztrációra dob vissza és ezután jelennek meg ott az előbb megadott adatok (felhasználónév és e-mail cím).

Ha minden jó volt az előző esetben, lehetőség nyílik a harmadik regisztrációs oldalon a jelszót létrehozni. Erre két mező áll rendelkezésre: kétszer kell megadni ugyanazt a jelszót. Minderre azért van szükség, hogy a jelszót pontosan lehessen beírni. Véletlenül se történjen meg, hogy elgépelje a felhasználó. A jelszónak fontos kritériumai vannak. Tartalmaznia kell 1 nagybetűt, 1 számot és minimum 8 karakter hosszúnak kell lennie. A program a „Tovább” gomb megnyomásával ellenőrizni fogja a mezőket. Jelezni fog, ha nincs kitöltve valamelyik mező, ha a két jelszó nem egyezik, illetve, ha gyengének ítéli meg a jelszót. A hiba kiírása során a program kiüríti a mezőket, hogy tiszta lappal lehessen kezdeni egy új jelszó megadását, ezzel kizárva a hibalehetőségeket. Ha minden helyes a program eltárolja az adatokat, a jelszót pedig szigorúan titkosítva, majd a program áthelyez a bejelentkezési felületre. A vissza nyilat és a „Jelentkezz be” gomb itt is használható. A vissza nyíl esetében a második regisztrációs felületre helyez a program az előbb beírt adatokat megjelenítve.

A bejelentkezési felületre visszatérve már csak a bejelentkezés vár ránk. Meg kell adni azt a felhasználónevet vagy e-mail címet, amivel regisztráltunk és be kell írni a másik mezőre a hozzá tartozó jelszót. Ha a felhasználónév/e-mail cím és a jelszó nem helyes azt a program jelzi, kiüríti a jelszó mezőt és mindkét mező vörösre vált. Amennyiben sikerült az azonosítás két továbblépési lehetőség van. Ha admin jogosultsággal jelentkeztünk be, akkor az admin felületre fog átrakni a program, de elsősorban a normál felhasználói esetet kell átnézni.

Miután megtörtént a felhasználói azonosítás, a program egy személyre szabott üzenettel üdvözlí a felhasználót majd, ha ez kész áthelyez a járatok böngészőbe. Itt lehet keresni indulási és érkezési repülőterekre városnév és a repülőtér kódja alapján (pl.: Budapest – BUD). A mező módosítása után azonnal megjelennek azok a járatok, amelyek a keresési feltételnek megfelelnek. Ha rossz a város neve, vagy ilyen járat nem létezik, a program azt egyértelműsíti. Egy járat kártyán látható, hogy honnan hova tart a járat, mikor indul, mennyi a repülőút hossza, valamint mennyi elérhető hely van még rajta. Egy járat akkor nem jelenik meg, ha már elindult, van rá foglalásunk, valamint, ha tele van.



7. ábra Jegyvásárlás

Ha rányomunk egy kártyára, megjelennek a benne rejlő foglalási lehetőség, és az összes hozzá tartozó információ. Itt több gomb is megjelenik, a vissza nyíl és a „Mégse” gomb ugyanazt a funkciót tölti be: visszatérnek arra az oldalra, ahol a járatok listája látható. A harmadik lehetőség a „Helyfoglalás” gomb. Ez szolgál arra, hogy továbblépjünk a repülőgép helyfoglalási oldalára.

A helyfoglalási felületen egy 6 oszlopból és 20 sorból álló repülőgép ülésrendszerét láthatjuk. Pirossal jelennek meg a lefoglalt, szürkével a szabad, illetve zölddel jelölhetjük ki az általunk választott helyet, amit változtatni lehet egészen a „Véglegesítés” gomb megnyomásáig. Ha nem választottunk még helyet és úgy nyomjuk meg a gombot, akkor a program hibát ír, ugyanis ez a lépés kötelező. Ha piros helyet nyomunk meg, akkor a program arra fog figyelmeztetni, hogy erre a helyre valaki már

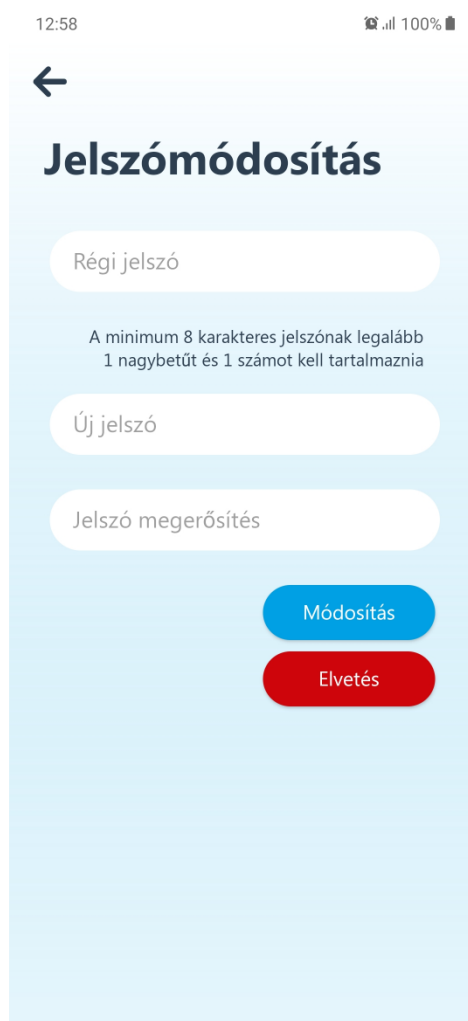
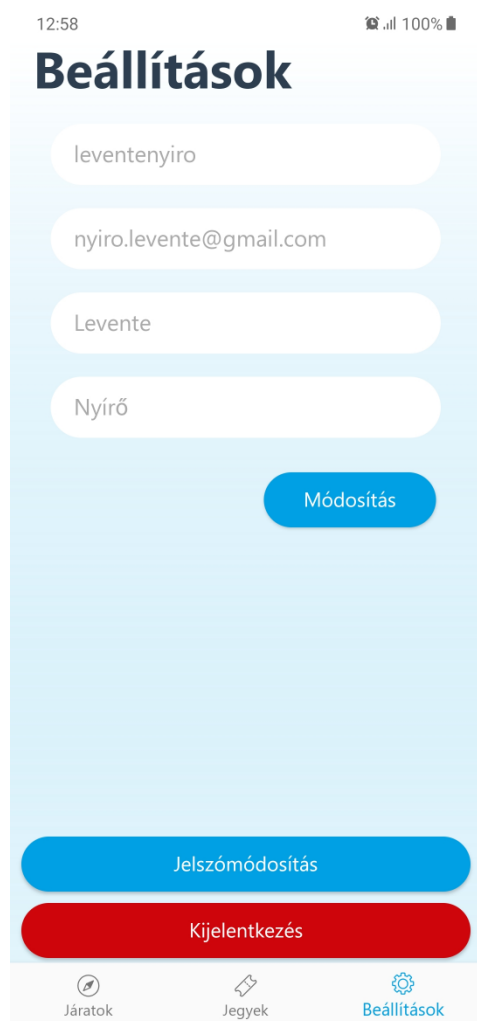
rendelkezik foglalással. Alul található egy „Vissza” gomb, amivel az előző, azaz a járatinformációs felületre térünk vissza. Amennyiben minden helyes és a véglegesítünk, akkor a program feldob egy ablakot, hogy biztosak vagyunk-e az ülés lefoglalásában. Az igenre nyomva áttérünk arra a felületre, ahol a lefoglalt jegyek találhatóak. Ez a felhasználói felület második ikonja az alul található menüben a „Járatok” és a „Beállítások” opció mellett. Az itt fellelhető elemek mindegyike egy-egy beszállókártya. Egy ilyen kártyán megjelenik, hogy honnan-hova szól a járat, mikor indul, mennyi idő a repülőút, illetve, hogy melyik ülőhelyre szól a jegyünk.

Egy beszállókártyára rányomva megjelenik az összes információ a járatról és a foglalásunkról a repülőterek 3 jegyű kódját is beleértve. Ha először tértünk ide az applikáció azonnal átdob a beállításokba, hogy engedélyt adhassunk neki a képernyő felvilágosítására, ami hasznos azért, hogy a beszállókártya sokkal inkább leolvashatóbb legyen, mint gyenge képernyőfény mellett. Ahogy ezzel végeztünk, vissza gombbal újra a beszállókártyán találjuk magunkat. Természetesen ezt a procedúrát nem kell minden alkalommal megtenni, csupán az első elindulás alkalmával. Ahogy látjuk a beszállókártyán az információk mellett egy QR kód is található a beszállókártya azonosítására a későbbi beléptetőkapuknál. Van egy vissza nyíl, amivel természetesen a fő felületre juthatunk, illetve egy „Lemondás” gomb, amivel a foglalást lehet véglegesen lemondani.

Amennyiben tényleg szándékozunk lemondani egy jegyet, abban az esetben átkerülünk egy jóváhagyás felülethez, ahol a helyes jelszót kell megadni a véglegesítéshez. A véglegesítés megszakítására a vissza nyíl, illetve a „Mégse” gomb szolgál. A harmadik gomb a „Tovább” gomb, ami ellenőrzi a megadott jelszó érvényességét, elvégzi a törlést a jegyen és visszahelyezi a felhasználót a járatok oldalra a főoldalon.

Egy harmadik egyben utolsó lehetőség, ami az alsó menüsorban található az a „Beállítások” menü. Erre az ikonra rányomva sok lehetőség található, ami a felhasználói fiókadatok módosítására irányul. Az itt megjelenő négy mezőben található a felhasználónév, az e-mail cím, kereszt- és vezetéknév. A „Módosítás” gombra nyomva az eddig letiltott mezők módosíthatóvá válnak. Innentől kezdve a regisztrációs felületeken is elvárt kritériumoknak megfelelően lehet módosítani az adatokat. Egyik mező sem lehet üres. Amikor a módosításban vagyunk, 2 lehetőség jelenik meg. Az egyik az „Elvetés” ezt megnyomva minden módosítást elfelejt a program és megtartja az eddigi

adatokat. A másik lehetőség a „Mentés”, aminek a megnyomása esetén a program ellenőrzés alá von minden mezőt majd, ahol nem felelnek meg az adatok, abban az esetben az adott mező vörössé válik és hibaüzenet formájában láthatóvá válik, miért nem módosítható az adott elem. Egészen addig, amíg hiba van a Mentés gomb sose fogja a módosított adatokat rögzíteni, azonban, ha minden sikeres a Mentés gombbal visszaáll a régi kinézete a lapnak és az új adatokat láthatjuk a blokkolt mezőkben. Ezen az oldalon még további két lehetőség van. Az egyik a „Kijelentkezés”. Ez a gomb ugyanazt a funkciót tölti be, mint alaphól a főoldal esetében a beépített vissza gomb. Megjelenik egy ablak, amivel a program a biztosításunkat kéri a tényleges kijelentkezéshez. Az igen gomb megnyomásával a program visszaléptet a bejelentkezési felületre, természetesen minden mentett adatot elmentve. A beállításoknál a másik lehetőség a „Jelszómódosítás” gomb.



8. ábra Felhasználói beállítások

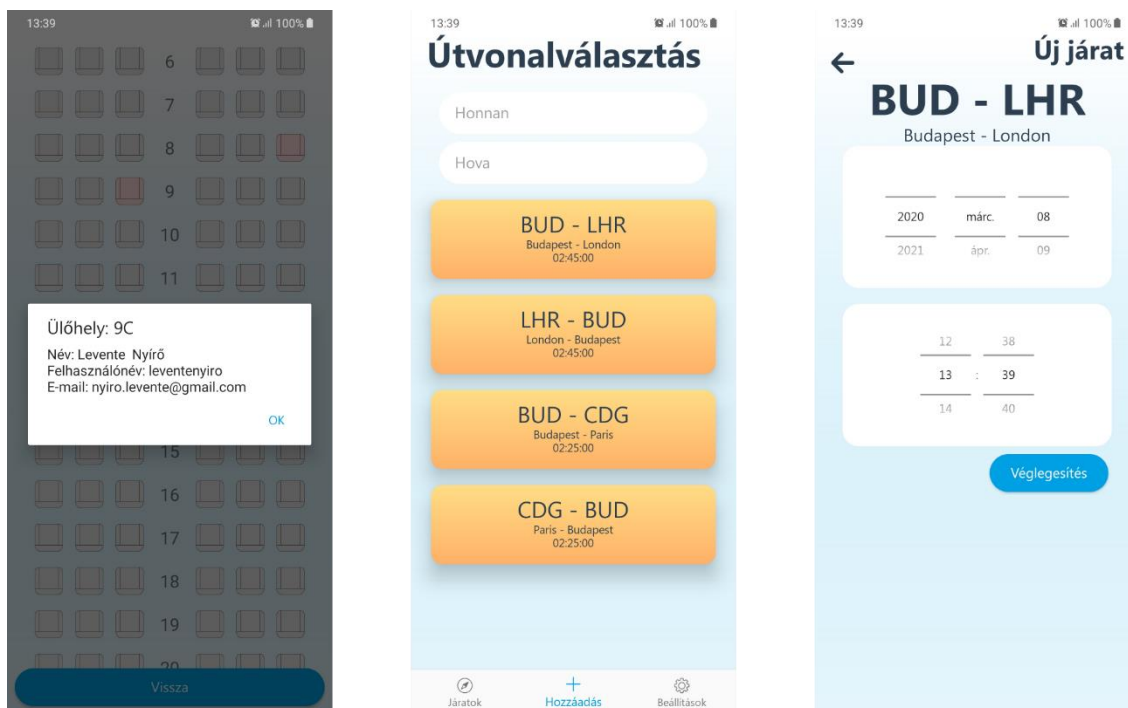
A jelszómódosítás felületen a 3 mezőt találunk. Az elsőbe meg kell adni a régi jelszót, ezzel azonosítva a saját fiókunkat, valamint az alsó két mezőbe az általunk választott új jelszót kell megadni. Az új jelszó kritériumai itt is ugyanazok, mint a regisztrációnál: minimum 8 karakter, ebben minimum 1 nagybetű és egy szám. Ha valami nincs kitöltve, a régi jelszó helytelen, az új jelszavak nem hasonlítanak egymásra, illetve az új jelszó gyenge, akkor a „Módosítás” gombra nyomva ezt a program jelezni fogja hibaüzenettel, illetve üríti a mezőket az újrapróbálkozáshoz, azonban, ha a jelszómódosítás sikeres, az applikáció rögzíteni fogja az új jelszót az adatbázisban titkosítva eltárolva, majd visszaléptet a beállítások oldalhoz. Az „Elvetés”, a vissza gomb, illetve a vissza nyíl is szintén visszavezet az előbb említett oldalhoz, azonban ebben az esetben nem történik ellenőrzés, valamint módosítás se.

A felhasználói funkciókon kívül létezik még egy lehetőség bejelentkezéskor, ha valaki rendelkezik az admin jogosultságú felhasználó adataival. Szintén a belépés felületen kell megadni az admin felhasználónevét vagy e-mail címét, illetve a jelszavát. Ha minden adat helyes, ezután a belső admin felületen találjuk magunkat.

Az első oldal, amivel szembesülünk az a járatok lista. Hasonlóan a felhasználói felülethez, itt is lehet böngészni keresővel a járatok között. Megjelenik egy kártyán minden információ az adott járatról. Egy járatra rányomva azonban nem az adatokat tudjuk meg a járatról, hanem a repülőgépen foglalt helyeket láthatjuk. Egy ilyen foglalt helyre rányomva megjelenik egy ablak, aminek a segítségével megtudjuk, hogy melyik felhasználó foglalt helyet, illetve megtudjuk jeleníteni az adatait. Mindez azért fontos, mert automatikusan létrejön egy utaslista az adott járatról. Itt egy gomb található viszont, ez a „Vissza” gomb, ami természetesen a járatok oldalhoz terel vissza minket.

Az alsó menüsorban továbblépve a „Hozzáadás” gombhoz találunk útvonalakat, amikből a keresővel böngészhetünk. Ezek előre meghatározott útvonalak, amelyekből ki kell választani egyet, ha új járatot szeretnénk létrehozni. Egy ilyen kártyán megjelenik az indulási, illetve érkezési repülőterek kódja, a városok neve, valamint a repülőút hossza.

Az egyik kártyára rányomva áttérünk az új járat adatai oldalra, ahol meg kell adni, hogy az adott útvonalon melyik időpontban induljon járat. Egy járatot minimum egy héttel és maximum egy évre tudunk ütemezni. A véglegesítés gombbal nyílik lehetőségünk az új járatot a választék közé helyezni. Ha meggondolnánk magunkat a vissza nyíllal, illetve a beépített vissza gombbal tudunk visszatérni a főoldalra.



9. ábra Admin felhasználó funkciói

A harmadik, egyben utolsó menüpont a „Beállítások”. Itt két lehetőségünk adódik admin felhasználóként. Az egyik a „Jelszómódosítás” a másik a „Kijelentkezés”. Fontos tudni, hogy az admin felhasználó adatait nem lehet módosítani biztonsági okok miatt. Ennek a felhasználónak csak a jelszavát lehetséges szerkeszteni. A „Kijelentkezés” gombnak tulajdonképpen ugyanaz a funkciója, mint a sima felhasználói kilépésnek. Egy ablak megkérdezi, hogy biztosak vagyunk-e a kijelentkezésben. Ha ez egy igenleges választ kap, kis idő után a bejelentkezési oldalon találjuk magunkat.

A „Jelszómódosítás” gombra nyomva, ugyanazt az űrlapot találhatjuk, mint a felhasználói jelszómódosításnál. Itt is meg kell adni a régi jelszót, illetve kétszer az újat. A vissza nyíllal, gombbal, az „Elvetés” gombbal, illetve sikeres módosítás esetében a „Módosítás” gombbal visszatérhetünk az admin felületre.

Nagyon fontos, hogy a regisztrációnál, illetve a módosításoknál jegyezzünk meg minden adatot, valamint vigyázzunk ezeknek a birtoklásával, ugyanis bárki visszaélhet a rendelkezésre álló adatainkkal.

5. Összegzés

Elérkeztünk a szakdolgozat utolsó fejezetéhez. Ez a rész kifejezetten arról szól, hogyan éltem meg a mögöttem álló hónapokat fejlesztés tekintetében, illetve milyen eredményeket értem el, miket vártam, mivel tapasztalatokat szereztem, hogyan tudnám ezt kamatoztatni a jövőben.

Az elsődleges cél kifejezetten a légitársaságokhoz közel lévő applikáció készítése egy saját keretbe belehelyezve az előre megtervezett funkciókkal volt. Úgy érzem ezt minden feltétel nélkül sikeresen teljesítettem. Sok minden terv volt a fejemben, amivel még összetettebbé, még kompaktabbá lehetne tenni az alkalmazást, de az elvárt szintet így is sikerült elérnem az amúgy meglepően gyors fejlesztés során. A bejelentkezés-regisztráció páros, a belső járatböngészés, helyfoglalás is tökéletesen működik, csakúgy, mint a jegy kigenerálása. A felhasználói beállítások is úgy sikerültek, ahogy elvártam, illetve az admin lehetőségei a járat hozzáadásával, illetve az utaslista összeállításával. Az alapelképzeléshez sikerült hozzáraknom újdonságokat, illetve plusz dolgokat, azonban volt egy-két olyan feladatrészt, amibe belebonyolódtam és utólag nehéznek ítélem meg, vagy épp nem működött az elvárthoz képest. Volt olyan új plusz lehetőségem is, amit kifejezetten elkezdtem, azonban a működése nem úgy volt meg, ahogy az normál esetben kéne, hogy működjön. A cél viszont, hogy ezeket a szakdolgozat beadása után meg tudjam valósítani a későbbiekben.

Mint minden fejlesztésnél, itt is adódtak nehézségek, valamint problémák. Fontos célokat adott a programozás közben, hogy mindezt megtudjam oldani. Egy kisebb probléma esetén az internetes fórumok segítségével jutottam a megoldásra, azonban egy komplexebb nehézséggel találkozva, már egy külön projekt létrehozásával kellett illusztrálnom magamnak a probléma megoldását. Sok része a szakdolgozatnak például ilyen elemekből áll, amit egy-egy projektből emeltem át, miután a problémát fixáltam és minden az elvárt módon működött. Fontos volt sok esetben „átlapoznom” az Android hivatalos dokumentációját a hibafeltárással kapcsolatban, de az Android Studio IDE is sokat segített a helyes megoldás rávezetésében a debuggolás közben. Mindezeknek köszönheti az applikáció a jelenlegi hibamentes működését, és természetesen az én fejlődésemet is nagy mértékben elősegítette.

A szakdolgozatom készítése közben rengeteg pozitív impulzus és ugyanannyi szakmai fejlődés is ért engem, amit egy nagyon fontos hasznának tartottam és jelenleg is tartok a szakdolgozatoknak. Mivel egy ambiciózus, céltudatos személyként ismerem magam, így egy csomó szorgalmat és kutatómunkát sikerült beleáldoznom a programba, amit pozitív élményként írnék le, ugyanis nagyon sok tapasztalattal gazdagodtam az Android és a mögötte lévő komplex adatbázis területén. A fejlesztés alatt nagyrészt az Android hivatalos dokumentációja, illetve bizonyos fórumok vezettek rá a legmegfelelőbb válaszra. Megtanultam azt a fontos dolgot is, hogy a fejlesztésnél a megoldásokat hol kell keresni, hogyan vezessem rá magam a sikeres válaszra. Alapból ezt tartom az szoftverfejlesztői szakma alapjának. Erre még ráépül a kreativitás és a türelem, ami egy nagyon fontos tényező egy-egy hasonló fejlesztésnél.

Összességében nagy örömmel szolgált számomra, hogy ezt az egészet végig csinálhattam, ugyanis kiélhettem a kreativitásom mind témaválasztásban, mind az elkészítésben. Határtalan mozgástérrel rendelkeztem a fejlesztést tekintve és úgy érzem mindent sikerült a programba belehelyeznem, amiről ennek az alkalmazásnak szólnia kell. Sokat köszönhetek a támogató szavaknak, az építő jellegű kritikáknak, illetve a pozitív visszajelzéseknek egyaránt a fejlesztés során, mind tanároktól, osztálytársaimtól, ismerőseimtől, családtagoktól, de legfőképp mentortanáromtól, aki segített meghatározni a fejlesztés ütemtervét és minden egyéb szakmai kérdésre választ tudott adni.

Forrásjegyzék

- Android Studio Documentation - <https://developer.android.com/docs>
- Stack Overflow - <https://stackoverflow.com/>

Ábrajegyzék

1. ábra Light Airlines logó	6
2. ábra Az adatbázis szerkezete	9
3. ábra Activity-k közötti kapcsolatok	12
4. ábra Rendszerkövetelmények	22
5. ábra Telepítés menete	23
6. ábra Bejelentkezés és regisztráció	24
7. ábra Jegyvásárlás	26
8. ábra Felhasználói beállítások	28
9. ábra Admin felhasználó funkciói	30