Szakképesítés megnevezése: **Szoftverfejlesztő**

OKJ száma: **54 213 05**

ZÁRÓDOLGOZAT

PizzApp – Android applikáció

Merényi Miklós

konzulens Dóczy Kristóf László

14/S

Budapest, 2020

Nyilatkozat

Alulírott Dóczy Kristóf László kijelentem, hogy ez a záródolgozat saját tudásom, önálló munkám terméke.

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

2020. 03. 29.

hallgató aláírása

**Tartalom**

[1. Bevezetés 4](#_Toc36579989)

[2. témaválasztás 5](#_Toc36579990)

[3. fejlesztői dokumentáció 6](#_Toc36579991)

[3.1 Alkalmazott fejlesztői eszközök 6](#_Toc36579992)

[3.1.1 Android története és főbb verziói 6](#_Toc36579993)

[3.1.2 Android Studio 10](#_Toc36579994)

[3.1.3 Firebase 11](#_Toc36579995)

[3.1.4 Git, Github 12](#_Toc36579996)

[3.2 Adatmodell 13](#_Toc36579997)

[3.3 Részletes feladat specifikáció 16](#_Toc36579998)

[3.3.1 adminActivityClasses 16](#_Toc36579999)

[3.3.2 curActivityClasses 17](#_Toc36580000)

[3.3.3 misc 18](#_Toc36580001)

[3.3.4 segedClassok 19](#_Toc36580002)

[3.4 Tesztelési dokumentáció 20](#_Toc36580003)

[3.5 Továbbfejlesztési lehetőségek 20](#_Toc36580004)

[4. Felhasználói dokumentáció 21](#_Toc36580005)

[4.1 Rendszer követelmények 21](#_Toc36580006)

[4.2 Program telepítése és indítása 21](#_Toc36580007)

[4.3 Admin felület 22](#_Toc36580008)

[4.4 Futár felület 25](#_Toc36580009)

[5. Összegzés 27](#_Toc36580010)

[6. Jegyzékek 28](#_Toc36580011)

[6.1 Ábrajegyzék 28](#_Toc36580012)

[6.2 Forrásjegyzék 28](#_Toc36580013)

# Bevezetés

Két éves szoftver- és webfejlesztői képzésem végéhez közeledve azt a feladatot kaptam, hogy egy szabadon választott témában, saját ötleteim alapján egy nagyobb projektmunkát valósítsak meg és készítsem el mellé a hozzátartozó fejlesztői és felhasználói dokumentációt. Több hetes fejlesztés és kemény munka után olvasható most itt tevékenységem eredménye. A programom nem tartom késznek, ahogyan egyszer egy Facebook kommentelőtől olvastam:

*„Egy program sosincs kész, csak a fejlesztését fejezed be.”*

Több ötlet közül végül egy Android alkalmazásra esett a választás, amelyet legfőképp az indokol, hogy mind az eszköz, mind az eszközre való fejlesztés közel áll hozzám. Ez a munkám közben csak megerősödött bennem. Úgy gondolom, hogy a jövőben az okos telefonok még a jelenleginél is fontosabb szerepet kapnak, hiszen rendkívül kompaktak és napról napra, frissítésről frissítésre egyre közelebb kerülnek a laptopokhoz és az asztali számítógépekkel is kezdik felvenni a versenyt.

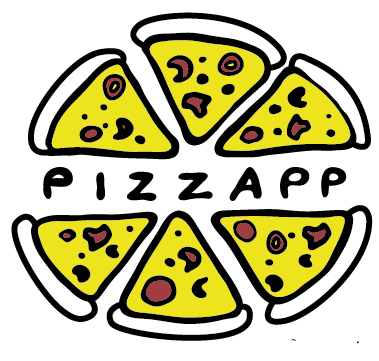
A fejlesztést magát Java és XML nyelvek felhasználásával végeztem, melynek okai, hogy applikációm külsője, illetve a tovább fejlesztés is rendkívül fontos számomra és a Java-ban, mint program nyelvben rengeteg lehetőség rejlik.

Applikációm lényegében egy pizzázó mindennapi munkáját könnyítheti meg, és teheti környezettudatosabbá. Minimális admin felület mellett inkább a futároknak szól az alkalmazás. Bejövő rendeléseket gombnyomással futárhoz tudnak rendelni, aki az app segítségével könnyedén tudja hova kell mennie, mennyi pizzát szállít és mekkora összeggel kell visszaérkeznie. Programom célja, hogy helyettesítse a papírt, amit legfeljebb egyszer használnak majd hulladékba kerül.

# témaválasztás

Az ötlet maga egy barátomtól származik eredetileg, aki az egyik nyáron pizza futárként dolgozott egy vidéki pizzázónál és amikor egy beszélgetésünk során kiderült, hogy Android alkalmazás fejlesztést is tanulok, panaszkodott, hogy nem érti miért nincs még futároknak szóló alkalmazás, ami észben tartja mit hová kell vinnie és mennyi pénzt kell kérnie, pedig elég kézen-fekvő, hiszen

*„úgyis mindig mindenkinek a kezében van a telefon”*

Kisebb kutató munka után arra jutottam, hogy az ilyenfajta program valóban meglepően ritka. Általam megkérdezett tíz különböző, közöttük kettő budapesti pizzázó közül, mint nemrég tudomásomra jutott, egyetlen egy vidéki étterem rendelkezik hasonlóval, amit sajnos nem állt módomban megtekinteni. Eredeti elképzelésem alapján az admin felület egy Windows Form lett volna, amihez társult volna az Android alkalmazás, viszont rá kellett jönnöm, hogy ez rengeteg munka lenne, ami a minőség romlását eredményezné, ezért úgy döntöttem, hogy ötletem felét elajándékozom osztálytársamnak, Tolnai Dominiknak, aki rögtön elvetette a Windows Form-ot és webes applikáció fejlesztésébe kezdett. Elmondása szerint sokkal logikusabb, mert nem kötött operációs rendszerhez és a verzió frissítések is könnyedén megoldhatóak web alkalmazás esetében. Hosszabb ideig párhuzamosan ment a két szoftver fejlesztése, viszont sajnos végül különböző adatbázis kezelő eszközök mellett döntöttünk, ami azt eredményezte, hogy az alkalmazásaink jelen állapotukban nem tudnak kommunikálni egymással. Természetesen rengeteg ötletet cseréltünk és egymás véleményére adva fejeztük be munkáink.

1. ábra: PizzApp logó

# fejlesztői dokumentáció

## Alkalmazott fejlesztői eszközök

### Android története és főbb verziói

Az Android nem más, mint egy rendkívül elterjedt és híres, Linux kernelen alapuló operációs rendszer, amely eszközök széles skáláján (mobiltelefonokon, tableteken, TV-ken, okos órákon, hűtőkön stb.) fellelhető. Népszerűségét többek között látványos felhasználói felületének, kis hardverigényének és nyílt forrás kódjának (ezáltali ingyenes használatnak) köszönheti. Továbbá sokat segített a Google sikeres marketingje, mely lehetővé tette, hogy az évek során háttérbe szorítsa versenytársait és kimagaslóan uralja az okos telefonok operációs rendszerének piacát. Jelenlegi adatok (2020. február) szerint közel 75%-os részesedéssel bír, mögötte az Apple által fejlesztett IOS 25%-kal rendelkezik és minden más operációs rendszer (KaiOS, Samsung, Windows például), okos telefonokra, a maradék 1%-on osztozik. Érdekesség, hogy az elmúlt öt évben 10%-ot nyert az IOS az Androidtól, viszont a különbség még így is grandiózus.

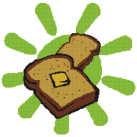
Fejlesztését az Android Inc kezdte melyet 2005-ben felvásárolt a Google, majd az Open Handset Alliance (amely egy 84 cégből álló konzorcium, vezetője a Google​) 2007-ben bejelentette az Android platformot, 2008 októberében pedig forgalmazásra került az első Androidot használó eszköz, a HTC Dream (ismerhető még HTC G1, T-Mobile G1 vagy Era G1 néven).

Mielőtt rátérnék a különböző kiadásokra kronológiai sorrendben érdemes megemlítenem, hogy az eltérő változatok visszafele kompatibilisek​, viszont elfordulhatnak nagyobb váltások ebből kifolyólag észszerű lehet alaposan átgondolni, hogy mire kezdjük meg a fejlesztést. Érdekesség, hogy tíz éven át, azaz 10 verzión keresztül kódnévként valamilyen édességet kapott az adott variáns. Ezt a sorozatot a jelenlegi legújabb kiadás törte meg.

Android 1.0: Apple Pie

Az első kereskedelmi forgalomba hozott verzió, amely 2008 szeptemberében került piacra a korábbiakban említett HTC G1 operációs rendszereként. Őt tekintjük az első API szintnek. Felhasználói felülete nem nyerte el a legtöbbek tetszését és nehezen volt használható. Megtalálható volt benne a kamera, média lejátszó, WiFi és a Bluetooth. Ezeken kívül tartalmazta már az Android Marketet, egy böngészőt, Gmail-t, YouTube applikációt, a Google Calendar, Contacts, Search, Talk szolgáltatásait és ezek szinkronizációs lehetőségét, azonban hiányzott belőle a szoftveres billentyűzet, ami a HTC Dream esetében nem volt hatalmas probléma, hiszen az eszköz rendelkezett külső, dedikált klaviatúrával, viszont az érintő képernyő fejlődésével és méretben való növekedésével ez eltűnt, ezáltal a későbbiekben szükségletté vált a szoftveres billentyűzet.

2. ábra: HTC Dream

Android 1.1: Banana Bread

2009 februárjában jelent meg, frissítésként megkapta a HTC G1. Második szintű API. Rengetek hibát javítottak az előző kiadásból, azonban említésre méltó újdonság nem került bemutatásra.

Android 1.5: Cupcake

Két hónappal az előző verzió után került kiadásra. Harmadik szintű API. Megjelent a szoftveres billentyűzet méghozzá automatikus kiegészítéssel, a widgetek (angol szójáték: window gadget, ablak kütyü), animációk és automatikus forgatás. Ezeken kívül Bluetooth automata párosítás, másolás-beillesztés funkció páros a böngészőben, videó rögzítés, lejátszás és YouTube-ra való feltöltés.

Android 1.6: Donut

A 2009 szeptemberében kiadott Donut esetében szintén inkább a hibák, főként az Android Market javítására fektették a hangsúlyt és csak néhány új funkció került be, mint például teljes platformos keresés, WVGA kijelző támogatása, hangalapú keresés és tárcsázás, valamint a beépített szöveg felolvasó. Mindemellett meghozta a negyedik szintű API-t.

Android 2.1: Eclair

Az Eclair verzió uralkodása idején több szintet is lépett az Android API (5-7). A mindössze egy hónappal a Donut után kiadott variánsban megjelent a Multitouch, beszéd szöveggé alakítása, NetBook és tablet támogatás, az élőhátterek és a több felhasználói fiók kezelése. Ezeken kívül a kamera szoftverét javították jelentősen.

Android 2.2: Froyo

2010 májusában jelent meg, többnyire teljesítmény és a stabilitás javítása történt meg. Emellett kiadásra került a nyolcadik szintű API.

Android 2.3: Gingerbread

A 2010 decemberében megjelent Gingerbread a következő két API szinten kívül egy teljesen új, letisztult felhasználói felülettel és számos új funkcióval érkezett. Ezek közé tartozik például a giroszkóp, NFC vagy pedig a letöltéskezelő.

Android 3.2: Honeycomb

2011 elején került az eszközökre. Legfőképp a tablet PC támogatás köré épült ez a frissítés. API szint 11-13.

Android 4.0: Ice Cream Sandwich

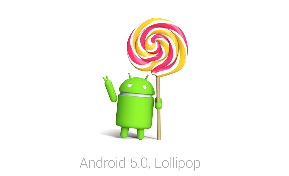
Ebben a kiadásban jelent meg a mai nap is rengeteget használt arc felismerés és a „swipe” („elhúzás”) mozdulat. Mindemellett a fényképszerkesztő applikáció kapott nagyobb figyelmet. 2011 októberében jelent meg és hozta az API 14-15. szintjét.

Android 4.1: Jellybean

A 2012 júliusában piacra dobott Jellybean számos új animációval és WiFi alapú helymeghatározással érkezett. Ezek mellett nagy mértékben nőtt az operációs rendszer hatékonysága is. Szokásos API szint növelés 16-18 között.

Android 4.4: KitKat

API szint 19-20. 2013 októberében jelent meg eredetileg „Key Lime Pie” néven. Az új tárcsázó és a teljes képernyős appok mellett a wireless nyomtatás és a lépésszámláló is bekerült.

Android 5.0: Lollipop

A 2014 novemberében napvilágot látó verzió tizenöt új nyelvet, új futtatókörnyezetet (ART) és designt, továbbá megújult értesítéseket és a 21-22 API szintet hozta magával.

Android 6.0: Marshmallow

2015 októberében kiadták az API 23-at és vele együtt a Marshmallow kódnevű Android verziót, melyben megvalósult egy újfajta permisson (engedély) rendszer és a mobil fizetés. Ezeken kívül új emojik kerültek be és fejlesztettek az ujjlenyomat támogatáson.

Android 7.0: Nougat

2016 augusztusában publikálták a 24-es API szinttel együtt. Többek közt a multi window támogatás, VR interfész, Unicode 9 emojik, alap billentyűzet GIF funkciója és annak többnyelvűsége voltak az újdonságok, emellett az operációs rendszert stabilizálták, ezáltal gyorsabb és energiatakarékosabb lett.

Android 8.0: Oreo

A 2017 augusztusában megjelent verzióban újragondolták az értesítéseket és a „Beállítások” menüt. Mindemellett megszületett a „Project Treble” fogalma, ami jelentősen megkönnyíti a telefon gyártóknak az Android szoftver frissítés folyamatát, ezáltal az gyorsabban juthat el a felhasználókhoz. API 26. szintje.

Android 9.0: Pie

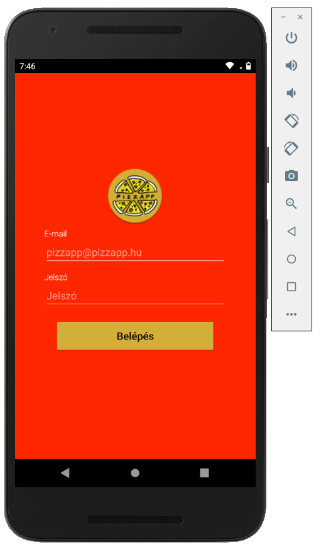
Az Android kilencedik jelentősebb változata 2018 augusztusában került publikálásra. Leginkább a felhasználói felület kapott több figyelmet. Többek közt a komponensek lekerekített sarkokat, még a kereső sáv és a hangerő csúszka teljesen új kinézetet kapott, ezzel modernebbé tették az UI-t. Ezek mellett bekerült egy választható, gesztusokon alapuló rendszer, mely a hagyományos UI-hoz képest egyszerűbb navigációt eredményez „swipe” mozdulatok esetén. Az API 28-as szintje.

Android 10

Az első olyan verzió, ami nem kapott kódnevet 2019 őszén került piacra. Főbb újdonságai közé tartoznak például az értesítés buborékok, az egész rendszerre kiterjedő, opcionális sötét (vagy éjszakai) mód, az összecsukható telefonok támogatása, a feljavított biometrikus hitelesítés vagy pedig a „Project Mainline”, ami lehetővé teszi, hogy kisebb alkotó elemek frissüljenek a Google Play Store-ból anélkül, hogy egy teljes rendszer frissítést végre kelljen hajtani. 29. szintű API.

### Android Studio

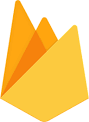
Az Android Studio egy ingyenes platform, integrált fejlesztőkörnyezet (IDE), melynek stabil verzióit 2014. decemberétől számítjuk. Java és Kotlin nyelvű programozás a legelterjedtebb az esetében, viszont lehetőség van C és C++ használatára is akár. Rendszerkövetelményeit tekintve Windows, Mac OS és Linux gépeken is elfut, ha azok rendelkeznek minimum négy, de inkább nyolc GB rammal és ajánlott esetben négy GB szabad tárhellyel, amiból másfél az emulátoré. A felettébb hasznos, beépített emulator viszont csak a 64-bites Windowst támogatja.

 Számomra nem volt kérdéses, hogy ezt a fejlesztői környezetet használjam, hiszen egyrészt ezen kezdtem meg Android fejlesztő tanulmányaim és a fórumok többsége is ezt ajánlja, ugyanis számomra minden elérhető benne amire szükségem volt a PizzApp fejlesztése során. Több hasznos funkciója között ott van például a Control + Space billentyűzet kombináció, ami javaslatot tesz, vagy ha egyértelmű, hogy mi következik automatikusan kiegészíti a szavakat vagy sorokat. Ezenkívül legtöbb hiba esetében a hibánk erősségétől függően különböző színű villanykörte ikonok jelennek meg javaslattal a hiba elhárítására, néha még használható is a tipp. Viszont talán a leghasznosabb funkciója a fentebb említett beépített emulator, mely egy fizikai eszközt emulál, utánoz, a Studiora kötve, így lehetőségünk van tesztelni az applikációnk különböző tesztesetekben egyetlen eszközről, nem is beszelve a tényről, hogy mennyivel gyorsabb, mint saját, fizikai eszközön. Lehetőségünk van kódunkat formázni és a felesleges, nem használt kód részleteket törölni akár az egész projektben egyszerre a Control + Alt + L billentyűzet hármas egyidejű lenyomásával. Ezeken kívül ezernyi beépített funkcióval rendelkezik az Android Studio és ha mindez nem lenne elég megannyi pluginnal egészíthetjük ki a fejlesztő környezetet.

3. ábra: beépített emulátor

### Firebase

Firebase egy mobil és web alkalmazás fejlesztését kisegítő termékekkel rendelkező platform, melyet a Firebase Inc. fejleszt 2011 óta, amit 2014-ben magába olvasztott a Google. Több, mint másfél millió app használja, köztük például a 9GAG.

Kezdetekben James Tamplin és Andrew Lee támogatásával az Envolve fejlesztett ki egy API-t, ami lehetőséget nyújtott web fejlesztőknek, hogy könnyedén beintegráljanak egy chat funkciót weboldalukba, viszont chat helyett adat szinkronizálásra használták fel, felhasználok közt, valós időben. Ilyen valós idejű adat szinkronizáció például a játék állapot frissítése. Végül úgy döntöttek, hogy külön választják ezt a chat szolgáltatástól és 2011-ben megállapítottak a Firebaset, melyet 2014-ben felvásárolt a Google.

Főbb, általam is használt szolgáltattatásai közé tartozik például a Firebase Authentication, ami a felhasználók regisztrálását, jogainak ellenőrzését és beléptetését végzi többek közt. Regisztrálásnál automatikusan titkosítja a jelszavakat és különböző formokkal e-mailen keresztül végrehajtható például egy e-mail cím visszaigazolás, vagy egy elfelejtett jelszó cseréje. Következő fontosabb szolgáltatása az adatbázis kezelés. Firebase esetében két féle adatbázis közül választhatunk: Cloud Firestore, vagy pedig a Realtime Database. A Firestore közelebb áll a hagyományos értelemben vett adatbázishoz. Dokumentumokkal és cellákkal dolgozik. Viszonylag új termék, 2018-ban vált elérhetővé a méltán sikeres Realtime Database utódjaként. A Realtime Database, vagy valós idejű adatbázis, a platform egyik első terméke, felhő alapú NoSQL adatbázis, amivel egyszerűen tudjuk tárolni és szinkronizálni az adatokat felhasználok közt. Ezeken kívül 16 termékkel rendelkezik még a Firebase, amelyek lehetőséget adnak programunk statisztikának követésére vagy fájljaink egyszerű és biztonságos tárolására.

Használata felettébb egyszerű. Az Android Stuidoban beépített Firebase asszisztens segítségével összekötjük adatbázisunk és egy gombnyomásra implementálja az úgy nevezett dependency-ket, melyek olyan könyvtárak, amikben megtalálhatóak a felhasználandó metódusok. Mindezek után a létrehozott CollectionReference segítségével végezhetünk adatbázis műveleteket: adatfelvétel, adatmódosítás, adattörlés. A művelet eredményét vizsgálhatjuk a megadott feladat sikeressége alapján (task.isSuccessful()), vagy pedig a lekérdezés állapota alapján (onSuccess-, onFailureListener).

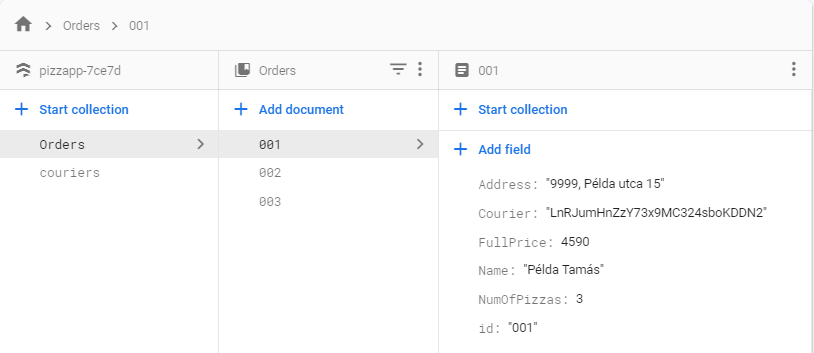
### Git, Github

Git és Github. Nem összekeverendő. Még az első egy nyílt forrás kódú verzió követő rendszer, amely a kis és nagy projekteket egyaránt nagy sebességgel és hatékonysággal képes kezelni, addig a Github egy Git gyűjtemény tárolására alkalmas felhő alapú, host szolgáltatás, mely a tárolás mellett rengeteg funkcióval rendelkezik. A Git parancssori eszköz, a Github pedig ehhez gondoskodik egy web alapú grafikus felhasználói felületről.

Felhasználását saját munkám alapján szemléltetném. A PizzApp esetében a Git letöltése és telepítése után az Android Studio eszköztára segítségével létrehoztam egy lokális gyűjteményt, úgy nevezett respository-t, amely szükséges esetleges offline munka alkalmával. A helyi respository képes tárolni a mentett változásokat (commit), amiket utána feltolhatunk (push) Githubra. Github gyűjtemény létrehozásánál lehetőségünk van választani publikus megosztás vagy privát tárolás között. Mindkét esetben lehetőség van csapatmunkára, viszont privát respository esetében csak az láthatja, akivel manuálisan megosztjuk. Nagyobb előnye az ilyenfajta verzió kezelésnek, hogy esetleges hiba esetén visszatérhetünk az előző, működőképes commitok egyikéhez és folytathatjuk onnan a fejlesztést, nagyobb csapat esetében pedig visszanézhető, hogy ki, milyen változásokat hajtott végre így egyszerűen megtalálható a hiba elkövetője. Természetesen ez a funkció az én esetemben nem volt felettébb hasznos, mert egyértelmű volt, hogy bármiféle komplikáció merült fel az teljes mértékben az én kezem munkája.

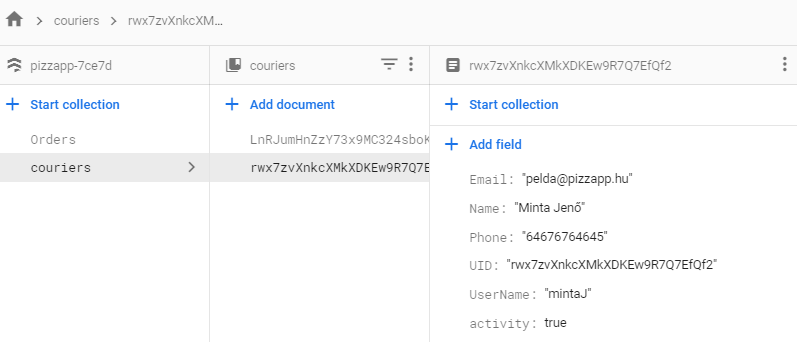
## Adatmodell

Alkalmazásom alapja két Firebase FireStore tábla, melyek a rendeléseket és a futárokat tartalmazza.

Az első tábla az *Orders* (4. ábra), mely a bejövő rendeléseket tárolja. Példának felvettem három rendelést. Dokumentum sorszámának (ID) egymást követő számokat határoztam meg, melyek minden esetben három számjegyűek és az egy rendelésre való hivatkozást segítik. Egy rendelés dokumentum a következőket tartalmazza:

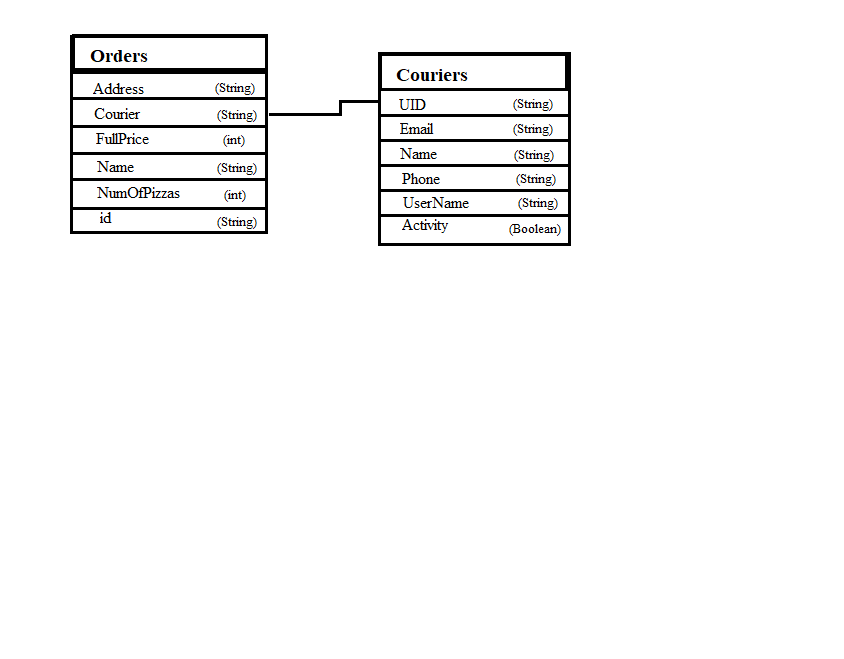
4. ábra Orders tábla

* *Address*
  + A szállítási címet tartalmazza, szövegként elmentve. Lekérdezésével a futár egy útvonaltervet kap a Google Maps használatával.
* *Courier*
  + A futár egyedi, a Firebase által generált ID-je, amely az egyetlen kapcsolatot biztosítja a két tábla között. Feladata, hogy a futár az eszközén csak azokat a rendeléseket kapja meg, amiket neki kell kiszállítania.
* *FullPrice*
  + A rendelés végösszege, szám, azaz integer típusban tárolva. Segítségével könnyedén meghatározható, hogy melyik címről, mekkora összeggel kell visszatérnie a futárnak.
* *numOfPizzas*
  + A szállítandó pizzák száma természetesen integerként tárolva. Lekérdezésével a futár ellenőrizheti, hogy meg van-e a szükséges mennyiségű termék. Fejlesztés közben jutott eszembe a lehetőség, hogy ne csak a pizzák számát, hanem nevét és méretét is eltárolhatnám. A jövőben előfordulhat, hogy bekerül ez a funkció.
* *Id*
  + Végül a dokumentum id-je szöveg, azaz String típusban tárolva. Szükségesnek éreztem, hogy elmentsem ilyen formában is, mert valamiért a Firebase nem szereti, ha a dokumentum id-ivel dolgozunk, szóval kénytelen voltam ezt valahogy megkerülni, hogy egyszerűen tudjak hivatkozni egyes objektumokra.

A második tábla a *couriers*, ami a futárok adatait tárolja, melyek elengedhetetlenek a program működéséhez. Ezeket az adatokat az alkalmazás admin felületén fel tudjuk venni, ezáltal regisztráljuk a futárokat, majd megadott e-mail címével vagy felhasználó névvel tud csak bejelentkezni. Egy futár objektum a következőket tartalmazza:

5. ábra couriers tábla

* *Email*
  + A futár e-mail címe szöveggént elmentve. A Firebase autentikációhoz szükséges, hogy létrehozza a felhasználót.
* *Name*
  + A futár neve String-ként elmentve a futárra való hivatkozáshoz.
* *Phone*
  + A futár telefon száma maximum 11 karakter hosszú szövegként bekérve és eltárolva. Az applikációm nem használja az adatot, de úgy gondoltam fontos adat lehet a munkáltató számára.
* *UID*
  + Egy futár egyedi, a Firebase által generált ID-je, amelyet szintén elmentettem cellaként, hogy egyszerűen tudjak vele dolgozni és egy adott futárt megkeresni lekérdezéssel.
* *UserName*
  + A futár által választott felhasználónév azt a célt szolgálja, hogy ne kelljen minden alkalommal az e-mail címmel bejelentkeznie, hanem lehetősége legyen egy rövidebb becenevet használnia. Szövegként kerül elmentésre.
* *Activity*
  + Logikai érték, azaz Boolean-ként kerül tárolásra. Két értéket vehet fel értelemszerűen: igaz vagy hamis. Feladata, hogy a törlést helyettesítse. Azt a javaslatot kaptam, hogy adatbázisból ritkán töröljek és inkább adott objektum állapotát frissítsem. Ha ez a cella igaz akkor a futár be tud jelentkezni és végezheti munkáját, viszont, ha hamis, akkor felhasználója fagyasztva van, akár átmenetileg, akár véglegesen.

 Ahogy azt már említettem a két tábla közt az egyetlen kapcsolat a futár automatikusan generált UID-je, melynek segítségével le tudom kérdezni azokat a rendeléseket, amit az adott futárhoz csatoltak.

6. ábra: táblák kapcsolata

## Részletes feladat specifikáció

Fejlesztés közben a jobb átláthatóság érdekében különböző csomagokba, package-be helyeztem Java osztályaim a könnyebb átláthatóság és logikusabb elrendezés érdekében. Ebben a fejezetben a következő csomagokon fogok végig menni és elemezni adott osztály metódusait és feladatait.

### adminActivityClasses

Ez első csomag az admin felület elemeit tartalmazza. Négy activityből áll maga az admin rész, mindegyik activityhez egy-egy activity.class, melyek elvégzik a megadott feladatokat.

A csomagban lévő első osztály az *AdminMenuActivity*, melynek fő célja az elhelyezett gombok onClick eseményének figyelése, és gombtól függően adott activityre való eljuttatás egy Intent segítségével. Induláskor az elhelyezett négy gombot az *inito* metódus inicializálja. A *btn\_regCur* és a *btn\_manageCur* megnyomása esetén átkerülünk a *RegCurActivity*-re vagy pedig a *ManageCourierActivity*-re, a *btn\_exit* és a *btn\_logout* esetében viszont egy *alertDialog* kerül a képernyőre, hogy biztosan végre hajtsa-e a gomb az utasítást, azaz kijelentkezzen-e vagy bezárja-e az alkalmazást. A különbség annyi, hogy még a *btn\_exit* bezárja az alkalmazást, addig a *btn\_logout* csak az admin menüt zárja be, majd visszadob a bejelentkező felületre.

A *ManageCourierActivity* feladata, hogy egy felületet biztosítson, amin kiválaszthatunk egy futárt, majd az adott futárt vagy töröljük (inaktívvá állítjuk) vagy módosíthatjuk. Indításkor az *inito* metódusinicializálja a komponenseket és feltölti a *spinner*-t a Firebase-ből betöltött adatokkal. A *getSelectedName* feladata, hogy Stringként visszaadja a *spinner*-ben kiválasztott nevet. Az *errorAlert* metódus egy Stringet kér be, amit kiír egy *alertDialog* formájában. A *setCurInactive* egy olyan metódus, mely a bekért nevet megkeresi a táblában és ha megvan módosítja az adott futár aktivitását. Az *addToGlobal* egy Stringet vár, amit, ha megkap inicializál egy osztályt, melynek feladatáról a későbbiekben lesz szó, majd a megkapott szöveget értekként átadja a globális változónak. Ezeken kívül megtalálható két gomb. A *btn\_delcur* megnyomása esetén egy *alertDialog* ugrik fel, ami megkérdezi, biztosan szeretnénk-e törölni az adott futárt, ha igen akkor meghívja a *setCurInactive* eljárást és *Toast* formájában jelez vissza az eredményről. A *btn\_modCur-*ra nyomva meghívásra kerül a *getSelectedName* és a visszakapott nevet megkeresi a táblában, majd, ha megtalálta, adott futár UID-jét átadja az *addToGlobal*-nak és elindítja a *UpdateCurActivity-*t.

A *RegCurActivity*-ben van lehetőség futárok adatainak felvételére és regisztrációjára. A felhasználó a szöveget EditTextek segítségével vihet fel az adatbázisba, ezeket az komponenseket az *inito* eljárás inicializálja. A mezők kitöltése után a *btn\_regCur* megnyomásával elindul a *vrfInput* metódus, amely végignézi, hogy ki vannak-e töltve a beviteli mezők, hat karakternél hosszabb-e a jelszó és a jelszó ellenőrzés sikeres volt-e, ami jelen esetben a jelszó bekérését jelenti még egyszer és abban az esetben enged tovább, ha a két mezőben a szöveg megegyezik. Természetesen, ha a gomb úgy kerül lenyomásra, hogy a mezők nincsenek kitöltve, egy hiba üzenet jelenik meg és nem kezdődik meg a mezők ellenőrzése. Abban az esetben, ha a *vrfInput* mindent rendben talál a megadott adatokkal létrejön a futár objektum az adatbázisban.

Az *UpdateCurActivity* két helyről érhető el: a fentebb említett *ManageCourierActivity-*ből, vagy pedig a futár profiljából. Ebben az activityben a kiválasztott futár adatait módosíthatjuk. Az *inito* elindítja a komponenseket, majd a globális változó segítségével a *loadData* metódus megkeresi a módosítandó futárt UID alapján, ezek után a lekérdezett értékeket beilleszti az EditTextekbe. A *btn\_modCur* gombra nyomva meghívásra kerül a *vrfInput,* ami leellenőrzi, hogy szükséges mezők ki vannak-e töltve, ha rendben találja meghívja az *updateData* eljárást, ami a szükséges mezőket módosítja a bevitt adatok alapján.

### curActivityClasses

Ez a csomag tartalmazza azokat az osztályokat, amelyek összessége teszi ki a futár felületet. Első ilyen osztály a *CurMenuActivity*, az *AdminMenuActivity-*hez hasonlóan négy gombot tartalmaz. Ezeket a gombokat a megszokott *inito* metódus inicializálja. Szintén az admin menühöz identikusan a felső két gomb (*btn\_curProf* és a *btn\_manageDeliveries*) egy-egy activityre dobnak át, az alulsó két gomb pedig a kijelentkezésért és az alkalmazás bezárásáért felelős. Az ablak indulásakor egy lekérdezés történik, amely megkeresi a bejelentkezett futár megadott felhasználónevét és a menü felett megjeleníti azt.

A *btn\_curProf-*ra nyomva átkerülünk a *CurMenuActivity*-re, ahol a megadott adatokat tudjuk megnézni. A komponenseket az *inito* metódus indítja el, majd a bejelentkezett futár UID-je alapján lekérdezésre kerülnek az adatok, amik beillesztésre kerülnek a megfelelő TextView-ban. A *btn\_mod* lenyomása esetén a jelenlegi UID értéke átadásra kerül a globális változónak, ami a másik oldalt lekérdezésre kerül, miután egy Intent segítségével átkerültünk a *UpdateCurActivity-*re.

A *CurDeliveriesActivity* legfőbb feladata a futárhoz rendelt fuvarok megjelenítése CardView és RecyclerView segítségével. Miután az *inito* metódus inicializálta a komponenseket a *setUpRecyclerView* nevű eljárás lép életbe. Egy query használatával leszűkíti a keresést azokra a rendelésekre, amelyek a bejelentkezett futárhoz vannak kiosztva és ezalapján felépíti a *recyclerView* *adaptert.* A felepitett adapter folyamatosan figyeli az adatbázis változásait ezáltal valós időjűvé teve a program ezen részét. Ezt a folyamatos figyelést, úgy értem el, hogy az onStart és az onStop metódusok felül írtam az *adapter.startListening* és az *adapter.stopListening,* ezáltal az alkalmazás csak abban az esetben figyeli az adatbázist valós időben, ha az előtérben van. Ezeken kívül az adapter rendelkezik egy clickListenerrel, ami figyeli, hogy melyik cardView lett megnyomva és az információ alapján az adott kártyán levő címet nyitja meg egy alertDialog után a navigációban.

### misc

A *misc* csomagban azok az osztályok helyezkednek el, melyek nem tartoznak sem a futárokhoz sem az adminhoz. Ilyen class például a belépést intéző *LoginActivity* vagy az alkalmazás betöltését végző *fakeLoad.* A *fakeLoad* az alkalmazás elindításakor az első, ami megjelenik. Eredetileg nem volt különösebb szerepe, szimplán vizuális szempontból került be, ezért is kapta a „fake” elnevezést. A későbbiekben viszont bekerült a *randoPing* metódus, aminek feladata egy lekérdezés, amivel azt tesztelem, hogy az app képes-e elérni az adatbázist, emellett azt vettem észre, hogy ez a lekérdezés észrevehetően felgyorsítja a programban következő későbbi lekérdezéseket. Ezen kívül megtalálható benne egy *fakeloading* eljárás, ami a logó animációjáért felelős.

A másik osztály, ami a *misc* csomagban található az a *LoginActivity*. A *LoginActivity* feladata a beleptetés felülvizsgálata és végrehajtása. Miután az *inito* eljárás inicializálta a komponenseket lefut egy *isNetworkConnected* metódus, ami ellenőrzi, hogy kapcsolódva van-e az eszköz az internethez. Ezt minden egyes alkalommal megvizsgálja, mikor a *btn\_login* megnyomásra kerül. Ha az eszköz csatlakoztatva van a hálózathoz megkezdődhet a bejelentkezés. Természetesen, ha üres adatokkal kíséreljük meg a bejutást a *vrfInput* eljárás azonnal megállít és jelez, hogy üresek a mezők. A program felismeri, hogy ha e-mail címmel jelentkezünk be, azesetben azonnal a meghívja *perfromLogin*-t, ami a megadott adatokkal megpróbálkozik belépni és annak függvényében megy tovább, hogy mi lett az eredmény. Ha nem találta az adatbázisban a megadott e-mail címet vagy megtalálta, viszont hibás a megadott jelszó, hiba üzenettel értesít, ha megtalálta és megfelelő a jelszó akkor pedig végrehajtja a bejelentkezést. Lehetőségünk van felhasználó névvel is bejelentkezni. Ezt az *if (Patterns.EMAIL\_ADDRESS.matcher(userName).matches())* parancs felügyeli, ami annyit tesz, hogy megvizsgálja a megadott szöveg formáját és az eredménytől függően lép tovább.

### segedClassok

A *segedClassok* csoportjába azok az osztályok tartoznak, melyek a többi class munkáját segítik. Ilyen segéd osztály például az *Order*, aminek a dolga, hogy a lekérdezett rendelést objektummá alakítsa, ami alapján a program tud vele tovább dolgozni. Maga az osztály egy üres és egy populált konstruktorból, egy rendelés tulajdonságaival megegyező adattagokból, getterekből, setterekből és egy toString metódusból áll.

Második segéd osztályom a többször is említett *Globals*, melynek feladata egyes adatokat osztályok között mozgatni. Tartalmaz egy *setValue* eljárást, amivel értéket kap és egy *getValue* függvényt, ami visszaadja a beállított értéket.

Végül az utolsó a segéd osztályom a *FireStoreAdapter,* melynek feladata, hogy feltöltse a recyclerView-t, kiegészítve egy clickListenerrel, aminek segítségével elindítható az adott címre való navigáció a Google Maps használatával

## Tesztelési dokumentáció

Fejlesztés alatt a programom folyamatosan teszteltem a beépített emulátoron és a saját eszközömön, és az így felmerülő problémákat, bugokat menet közben azonnal javítottam. A fejlesztés végeztével egy átfogóbb tesztelésbe kezdtem. Először az emulátoron hajtottam végre a főpróbát, amelynek eredménye nem volt meglepő, minden tervezett szerint működött, hiszen ahogy említettem a fejlesztést ezen az eszközön követtem figyelemmel. Saját eszközömön a funkciók megfelelően működtek, viszont a vizuális effektek és a komponensek minimálisan elcsúsztak a képernyőm mérete miatt. A családom régi tabletjén folytatott tesztelés során megfigyelhettem, hogy a nagyobb képernyő nem okoz problémát, meglepő módon nem csúsztak szét az alkotóelemek, viszont több helyen előfordult, hogy egyes komponensek, például a *spinner* nem jelent meg megfelelően vagy pedig nem rendeltetés szerint működött. Ennek oka, hogy az általam fejlesztett program legkisebb támogatott API-ja a 16-os Jellybean, az ajánlott pedig a 20-as KitKat, viszont a tablet ennél jóval régebbi Android verzióval rendelkezett.

## Továbbfejlesztési lehetőségek

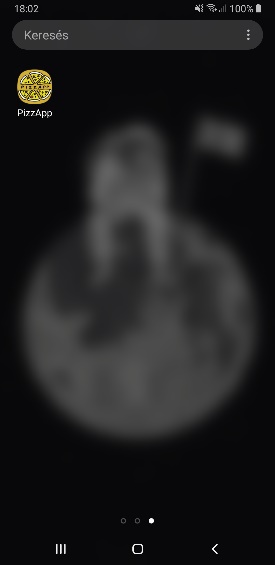
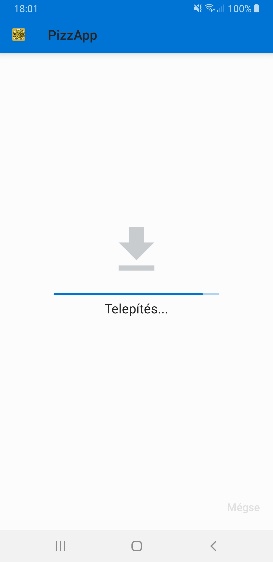
Fejlesztés közben rengeteg új ötletem támadt, melyek sajnos a jelenlegi verzióba nem fértek be. Ilyen ötlet például az általam tesztelésnél használt felület, ahol az admin ki tud adni rendeléseket futároknak. A későbbiekben bekerülhetne, hogy amint a futárhoz kiadnak egy rendelést, ő értesítést (push notification) kapjon róla és a „maradjak bejelentkezve funkció”.

# Felhasználói dokumentáció

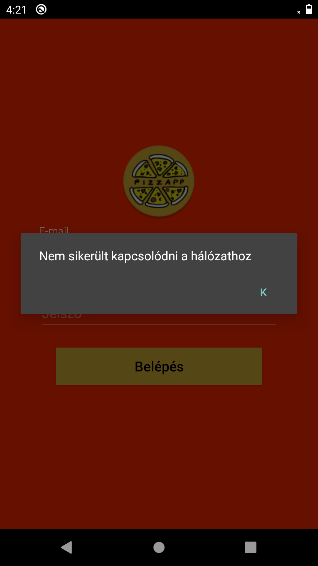
## Rendszer követelmények

Ahhoz, hogy a PizzApp megfélően működőjén minimum az Android JellyBean verziója szükséges, de az ajánlott az inkább a KitKat vagy az annál újabb kiadások. Tárhelyet illetve maximum nyolc MB szabad területet foglal, ezen kívül pedig stabil hálózati kapcsolat szükségszerű.

## Program telepítése és indítása

A kapott telepítőt, mely .apk kiterjesztéssel rendelkezik elindítja, majd a „Telepítés” gombra nyom. A telepítés nagyjából egy percet vesz igénybe, természetesen ez az eszköz sebességétől is függ. Ha kész a telepítés az ikonra nyomva indítható el az alkalmazást.

7. ábra: A telepítés folyamata

Az alkalmazás betöltése után a bejelentkező felület jelenik meg. A felületen megtalálható két mező, ahova értelemszerű sorrendbe a felhasználó név vagy az e-mail cím és a jelszó kerül. Amennyiben a következő ábra jobb oldalán látható „Nem sikerült kapcsolódni a hálózathoz” hiba üzenetet kapja, mennyen rá az X-re és győződjön meg róla, hogy be van-e kapcsolva a mobil hálózat vagy a WiFi elérése.

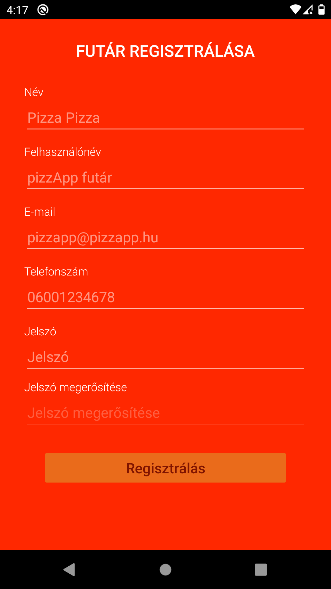
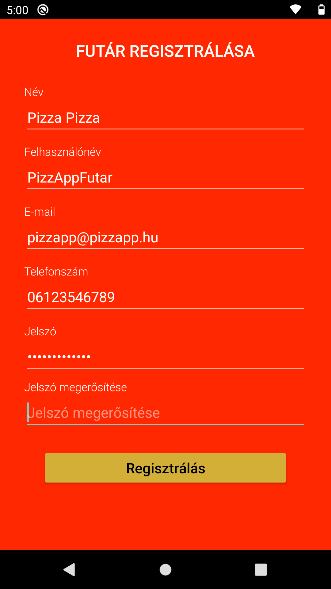
8. ábra: bejelentkezési felület

## Admin felület

Attól függően, hogy futárként dolgozik vagy sem kell bejelentkeznie. Első esetként az admin bejelentkezését nézzük meg. Alapesetben a felhasználó név „*admin*” és a jelszó szintén „*admin*”. Ha sikeres a bejelentkezés a következő menüt fogja látni.

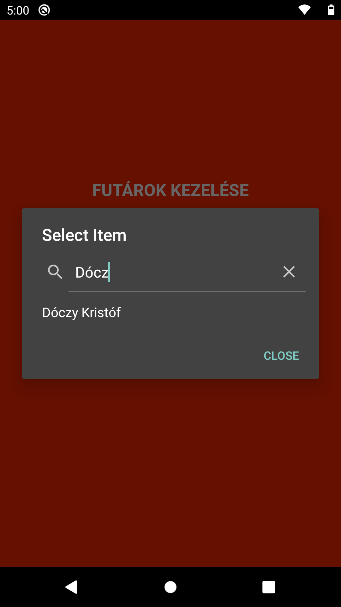


9. ábra: admin menü

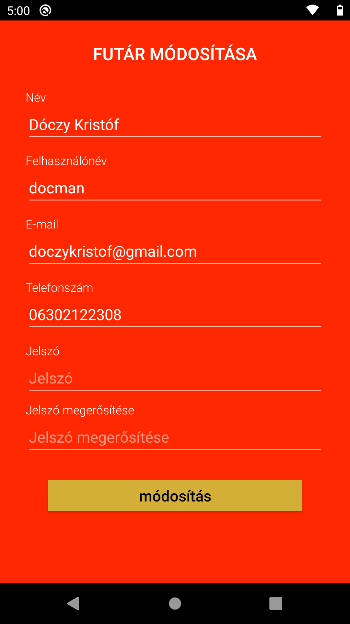
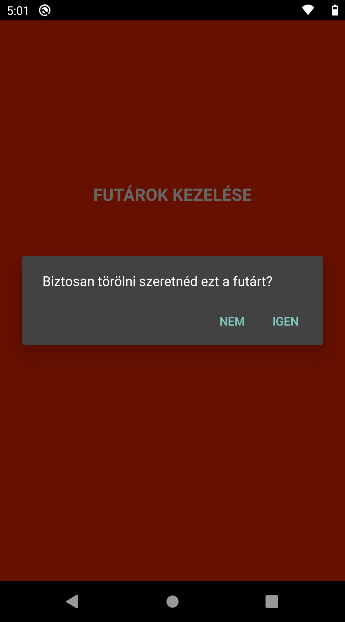
A felületen négy gomb látható. Az első a futárok felvétele melyben felvehetőek a futárok adatai és regisztrálásra kerülnek az e-mail címek.

10. ábra: Futár regisztrálása

Az ábrán jól látható, hogy a regisztrálás gomb és a jelszó ellenőrzés csak abban az esetben lesz elérhető, ha a jelszó megadásra került. A gomb megnyomásával lefut egy ellenőrzés, ami megvizsgálja, hogy megfelelőek az adatok, majd, ha minden rendben, visszalép a fő menübe.

 A második gomb a futárok kezelése. Amint megnyomja egy újabb felületre dob, melyen a „válassz futárt…” szövegre nyomva felugrik egy ablak, melyben az összes aktív futár fel van sorolva és lehetőség van kereséssel szűkíteni a találatok listáját.

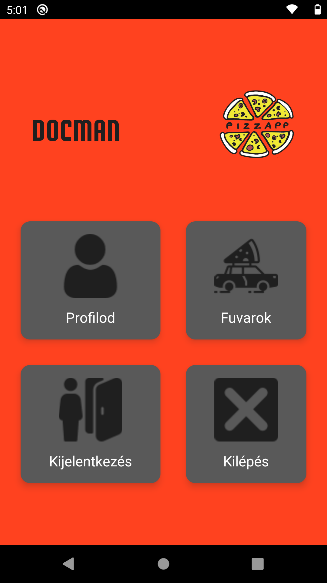
11. ábra: futár kiválasztása

Amint kiválasztotta a futárt, elérhetővé válik két gomb, a törlés és a módosítás. A törlés esetében egy ablak ugrik fel, mely megkérdezi biztosan végrehajtja-e a műveletet. Nemre nyomva bezáródik az ablak és nem történik semmi, viszont az igenre nyomva a kiválasztott futár törlésre kerül az aktív futárok közül és fagyasztva lesz a profilja, ezáltal nem lesz képes belépni felhasználó nevével.

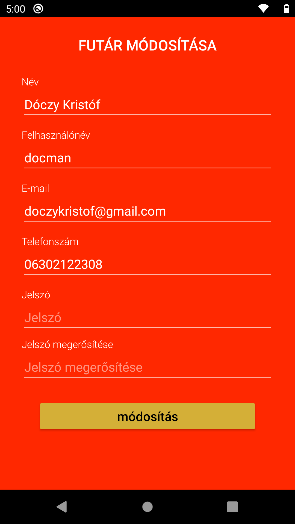
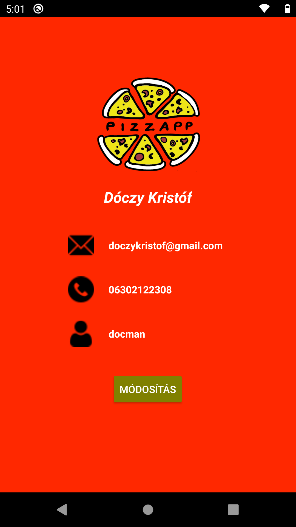
12. ábra: Futár törlése és módosítása

A művelet végén a program visszadob a menübe. Az alsó két gomb a kijelentkezésért és az app bezárásáért felelős. Mindkét gomb esetében egy ablak ugrik fel, ami megkérdezi biztosan végrehajtsa-e a műveletet. A különbség annyi köztük, hogy még a kilepés bezárja az appot, addig a kijelentkezés csak az admin felületet zárja be és dob a bejelentkező felületre.

## Futár felület

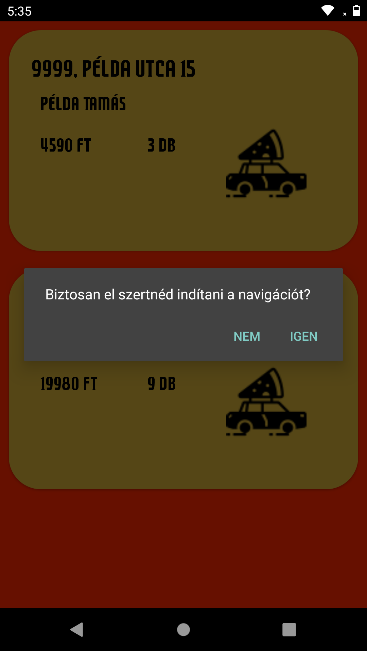
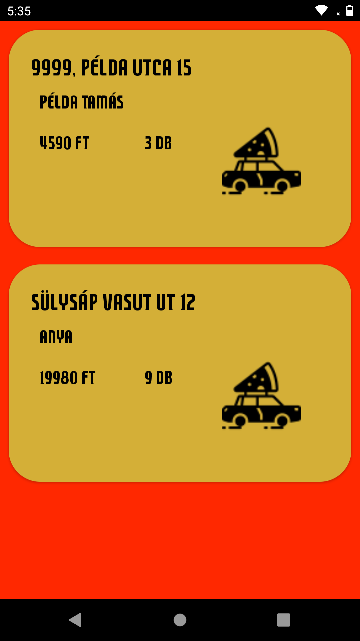
Abban az esetben, ha futárként szeretne bejelentkezni két féle lehetősége van: vagy a megadott e-mail címet vagy a feltételezhetően rövidebb felhasználó nevét használja. Mindkét esetben a megfelelő jelszóval párosítva sikeres lesz a bejelentkezés, ha mégsem a program tájékoztatja. Sikeres belepés esetén a következő ábrán látható menüt találja a saját felhasználó nevével.

13. ábra: Futár menü

A menü négy gombot tartalmaz. Az alsó két gomb a kijelentkezésért és az app bezárásáért felelős. Mindkét gomb esetében egy ablak ugrik fel, ami megkérdezi biztosan végrehajtsa-e a műveletet. A különbség annyi köztük, hogy még a kilepés bezárja az appot, addig a kijelentkezés csak az admin felületet zárja be és dob a bejelentkező felületre.

A *Profilod* gombra nyomva egy olyan felület, jelenik meg ahol áttekinthetőek a megadott adatok és ha valamely hibás, vagy unja a felhasználó nevét a módosítás gomb megnyomásával frissítheti azokat.

14. ábra: Futár profil és módosítás

Végezetül a negyedik gomb, melynek megnyomása esetén felsorolva láthatja a bejelentkezetett futár a rá kiosztott rendeléseket.

15. ábra: Fuvarok és egy cím elvállalása

Az ábrán látható példában a futárhoz két cím van rendelve melyek közül az alsót vállalta el elsőként. A címre nyomva egy ablak ugrik fel, amely megkérdezi biztosan elindítja-e a navigációt. Igen esetében pedig Google Maps alatt megnyílik az úti cél, ahol egyetlen egy gomb nyomassál elérhetjük a navigációt.

# Összegzés

Őszinte leszek még tegnap este sem hittem, hogy elkészülök, de be kell valljam végezetül felülmúltam a saját elvárásaim. Nem gondolom tökéletesnek az elkészült munkám, de ettől független büszke vagyok rá. Készítése ideje alatt rengeteget tanultam az Android fejlesztésről, a Java program nyelvről, az idő menedzsmentről és a tényről, hogy a Home Office nem azt jelenti, hogy mikor gép elé kerülök két e-sport meccs között haladok valamennyit. Pozitívan emlékszem vissza mind az elmúlt pár hónapra, ami a fejlesztéssel telt, mind az elmúlt két évre, ami alatt szerelembe estem ezzel a szakmával és barátokra találtam. Rengeteg nehézségen küzdöttem át magam és tanultam meg, hogy nem szabad feladni csak időt kell adni magamnak meg rájövök a megoldásra, vagy valahogy megoldóik magától. Izgalommal és kíváncsisággal várom, hogy mit hoz a jövő.

Hálás vagyok, hogy munkáim beadhattam, hálás vagyok, hogy megkaptam az esélyt, hogy elmondhassam magamról: Szoftverfejlesztő vagyok és hálás vagyok önnek, aki most ezt olvassa. Végezetül szeretném megköszönni konzulensemnek, Merényi Miklósnak, aki látott lehetőséget az ötletemben és elvállalt.

# Jegyzékek

## Ábrajegyzék

[1. ábra: PizzApp logó 5](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576950)

[2. ábra: HTC Dream 7](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576951)

[3. ábra: beépített emulátor 10](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576952)

[4. ábra Orders tábla 13](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576953)

[5. ábra couriers tábla 14](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576954)

[6. ábra: táblák kapcsolata 15](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576955)

[7. ábra: A telepítés folyamata 21](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576956)

[8. ábra: bejelentkezési felület 22](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576957)

[9. ábra: admin menü 22](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576958)

[10. ábra: Futár regisztrálása 23](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576959)

[11. ábra: futár kiválasztása 23](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576960)

[12. ábra: Futár törlése és módosítása 24](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576961)

[13. ábra: Futár menü 25](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576962)

[14. ábra: Futár profil és módosítás 25](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576963)

[15. ábra: Fuvarok és egy cím elvállalása 26](file:///E:\mellék%20projectek\Dokumentacio\Dokumentacio.docx#_Toc36576964)

## Forrásjegyzék

* Android Studio dokumentáció: <https://developer.android.com/docs>
* Stack Overflow: <https://stackoverflow.com>
* Firebase dokumentáció: <https://firebase.google.com/docs>
* PizzApp logó: Benkő Nóra