Munkahelyi beosztás-kezelő Android applikáció fejlesztése

**Készítette**: Ferenczik Judit

**Csapattagok:** Tisza - Kis Eleonóra, Szmolka László

Debrecen, 2022.

Debrecen Szakképzési Centrum  
DSZC Beregszászi Pál Technikum

Nyilatkozat

Alulírott **Ferenczik Judit** kijelentem, hogy ez a vizsgaremek saját tudásom, önálló munkám terméke.

A vizsgaremek közös részeit Tisza-Kis Eleonóra, Szmolka László és Jómagam készítettük, ezeket pontosan jelöltük.

Aláírás Aláírás Aláírás  
 Ferenczik Judit Tisza-Kis Eloenóra Szmolka László

Debrecen, 2022.

Tartalomjegyzék

[**1.** **Bevezető** 3](#_Toc100252140)

[**2.** **Megoldandó feladatok** 4](#_Toc100252141)

[**3.** **Fejlesztői dokumentáció** 6](#_Toc100252142)

[3.1. Alkalmazott fejlesztői eszközök 6](#_Toc100252143)

[3.2. Felületterv 6](#_Toc100252144)

[3.3. Adatmodell leírása 8](#_Toc100252145)

[**4.** **Részletes feladatspecifikáció** 12](#_Toc100252146)

[4.1. A program lényeges függvényeinek, eljárásainak, az osztályok metódusainak a specifikációja 12](#_Toc100252147)

[4.2. Az algoritmizálható részek leírása valamilyen algoritmus-leíró eszközzel 16](#_Toc100252148)

[**5.** **Tesztelési dokumentáció** 17](#_Toc100252149)

[**6.** **Továbbfejlesztési lehetőségek** 20](#_Toc100252150)

[**7.** **Felhasználói dokumentáció** 21](#_Toc100252151)

[7.1. Hardver követelmények 21](#_Toc100252152)

[7.2. Szoftver követelmények 21](#_Toc100252153)

[**8.** **A program telepítésének és konfigurálásának a leírása** 22](#_Toc100252154)

[8.1. A program használatának a részletes leírása 22](#_Toc100252155)

**[9.](#_Toc100252156)****[Összegzés](#_Toc100252156)** [26](#_Toc100252156)

[**10.** **Irodalomjegyzék, forrásjegyzék** 28](#_Toc100252157)

[**11.** **Ábrajegyzék** 29](#_Toc100252158)

# **Bevezető**

A dokumentáció elkészültét csoporttársaimmal Tisza-Kis Eleonóra és Szmolka Lászlóval készített beosztás kezelő és bérszámfejtő applikáció létrejötte hívta életre. Ötletet Szmolka László biztosította, aki könyvelőként dolgozik és közeli kapcsolatba került hasonló alkalmazások használatával. A jelenleg elérhető hasonló alkalmazások használata rendkívül bonyolult, így tervezés során fontosnak tartottuk, hogy az alkalmazásunk egy a való életben is használható, felhasználóbarát alkalmazás legyen.

A fő cél megvalósítása, hogy a munkáltató illetve munkavállaló kommunikációját megkönnyítsük, illetve kiküszöböljük az emberi mulasztásból adódó hibákat, automatizáljuk, rögzítsük a folyamatot.

A funkciók között többek között lehetősége lesz a munkavállalónak a szabadság, betegszabadság igényét jelezni, melyet a munkaadó értesítés formájában megkap és ezt vagy jóváhagyja, vagy elutasítja. Ezáltal biztosítja azt, hogy a ledolgozott munkaidő biztosan dokumentálva legyen, megkönnyítendő a bérszámfejtés feladatait. További funkciók között szerepel a munkavállaló adatainak módosítása, korábbi szabadságai időpontjának lekérdezése.

Az applikációt három különböző platformon terveztük megvalósítani szem előtt tartva azt a legfontosabb elvet, hogy ezek az applikációk összeköttetésben legyenek, azaz ugyanazt az adatbázist használják. A webes változatot Tisza-Kis Eleonóra, Androidos változatot pedig Ferenczik Judit készítette. A két felület elkészítésében hasonló elveket követtünk mind a használható funkciók, mind design tekintetében. Szerettük volna, ha nem csak, belülről de kívülről is szinkronban lenne a két felület egymással. Az asztali alkalmazás, melyet Szmolka László készített egy kicsit más funkciókkal rendelkezik. Ezt arra használjuk, hogy a hónap elején esedékes bérkalkulációs számításokat elvégezze a webes illetve androidos felületről kapott információkat felhasználva.

# **Megoldandó feladatok**

A projektmunka első fázisaként arról egyeztettünk, hogy milyen programot szeretnénk készíteni, és pontosan milyen funkciók legyenek benne elérhetők. Végeredményként egy letisztult, egyszerű felület megvalósítását terveztük, tudásunk, képességeink legjavát felhasználva, hogy egy stabilan működő alkalmazást készíthetsünk.

Ezt követően elosztottuk a feladatokat egymás között. Mivel hárman dolgozunk a projekten, adott volt, hogy három különböző felületre egy-egy személy készítse el az alkalmazást. Tisza-Kis Eleonóra a webes felületet vállalta, Szmolka László az asztali alkalmazást, én pedig a Android-os felület elkészítésére jelentkeztem.

A következő fázisban a felhasználók körét határoztuk meg. Végleges megoldásként kétféle felhasználói jelleget különböztettünk meg: alkalmazotti és vezetői felület. Lehetséges opcióként felmerült egy harmadik, „admin” felület létrehozása is, aki felügyeli a felhasználók tevékenységét, de végül ezt elvetettük, mivel csupán az adatbázis kezeléséhez lett volna szükséges.

Ezek után a felhasználók jogosultságait, az általuk elérhető funkciókat definiáltuk. Az alkalmazott részéről a legfontosabb a munkabeosztásának ellenőrzése, esetlegesen annak változtatásának lehetősége. Ehhez kapcsolódóan szükséges a „*Beosztásom”* menüpont, ahol a szabadság és táppénzt jelölhető/kérelmezhető, valamint egy *„Értesítéseim”* menüpont, ahol a még függőben lévő kérelmei láthatók. Mivel az alkalmazás egyik célja a jelenlétek rögzítése, ezért a már elfogadott szabadságok megtekintésére is szeretnénk egy felületet. Végül pedig a személyes adatainak módosítására is lehetőséget szerettünk volna biztosítani.

Az alkalmazotti felülethez hasonló a beosztást kezelő vezetői felület. Megtekintheti az alá tartozó beosztottak listáját és akár törölhet is őket. A „*Értesítéseim”* menüpontban a hozzá beérkező szabadság kérelmeket láthatja, és egyúttal jóvá is hagyhatja vagy el is utasíthatja. A személyes adatok módosítására a vezetői jogosultsággal is lehetőség van.

Mindezen feladatok megoldásához a következő lépés a közös adatbázis megtervezése volt. Hosszas egyeztetés után jött létre a végleges adatszerkezet, amely alkalmas az összes személy (akármilyen jogosultsággal rendelkezik is) adatainak tárolására, és az összes funkció működtetéséhez: egy naptár, amely a munkanapokat és munkaszüneti napokat elkülöníti, tábla a kérések kezeléséhez, valamint a jelenlétek tárolásához.

A program működéséhez tehát fontos egy bejelentkező felület, ahol e-mail cím és a megfelelő jelszó kombinációjával lehet az oldal tartalmát megtekinteni. Egyúttal itt dől el, hogy alkalmazotti vagy vezetői felületre kerül a felhasználó. Jogosultságtól függetlenül szükséges, hogy az adataiban történő változást azonnal végrehajtsa az adatbázisban is, például az alapértelmezett jelszót első bejelentkezés után a saját választott jelszóra cserélje a felület használója.

A beosztások megtekintéséhez egy naptár funkció bevezetésére van szükség: az adatbázissal való kapcsolat segítségével a hétvégék és munkaszüneti napok jól elkülönülnek, valamint felhívja a figyelmet a bedolgozandó pihenőnapokra, azaz a szombati munkavégzésre. A munkaszüneti és hétvégi napokra szabadság kérése sem lehetséges, így az algoritmus működésének is ehhez kell igazodnia.

A szabadság-kérelmek, valamint a táppénz/egyéb-bejelentés információit szintén az adatbázis megfelelő táblájába kell rögzíteni, hogy az majd az értesítésekben is megjelenhessen. Az eltárolt adatok alapján kap jelzést az adott munkavállaló vezetője, és a dolgozó is az elfogadás tényéről. Ezt követően kell eltárolni az adatokat a munkavégzésről, hogy a megtekinteni kívánt hónapban mely napokon dolgozott, és melyek azok a napok, amelyeken nem történt munkavégzés és ez esetben a munka hiányának a típusa.

Ezeknek a feladatoknak a megvalósítása volt a közös cél, ami alkalmazásonként más programozási megoldásokat kíván. Hasonló, egymással kompatibilis programok létrehozása a mobilos, webes és asztali felületre.

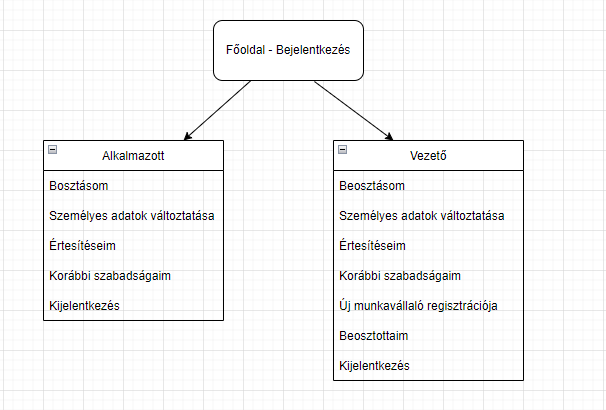
# **Fejlesztői dokumentáció**

## Alkalmazott fejlesztői eszközök

Az alkalmazás fejlesztése során főleg Kotlin nyelvet használtam égpedig Android Studio 2020.3.1-ben. A webes alkalmazás adatbázisa phpmyadminben készült, az androidos alkalmazás pedig lokális adatbázist használ, így mind mysql mind sqlite (verzió SQLite version 3.35.4) nyelvekkel is megismerkedtem mélyebben. Phpmyadminhoz XAMPP Control Panel v3.3.0-val segítségét vettem igénybe, mint webserver Environment. A különböző szerkesztési feladatok ellátására Microsoft Paintet használtam, illetve a feladatok megosztásának egyszerűsítésére Github-os repository hoztunk létre, melyhez én Git basht használtam.

## Felületterv

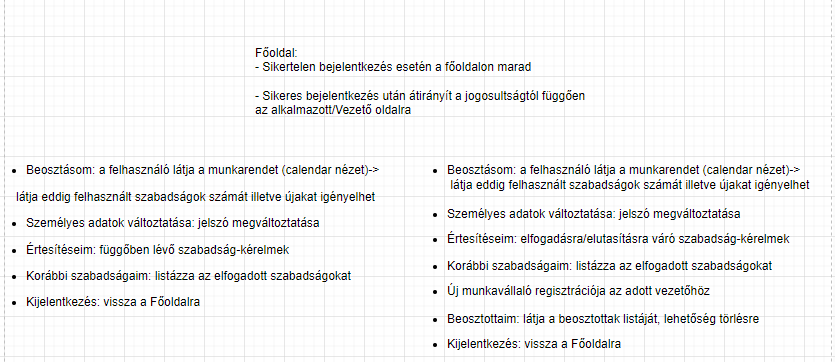
Az alkalmazás fejlesztése során első lépés az általunk kívánt funkciók megtervezése volt, mely szükséges a következő lépéshez azaz adatbázis megtervezéséhez. Az alábbi ábra mutatja, hogy milyen alap funkciókat terveztünk az applikációink szerves részévé tenni.



1. ábra BeosztasApp szerkezeti felépítés

A felhasználót legelőször egy bejelentkező felület fogja fogadni, melyen csak akkor tud tovább jutni, ha rendelkezik egy azonosításra szolgáló e-mail címmel és jelszóval. Az applikációban nem lesz lehetősége a felhasználónak önmagát regisztrálni, ez csak admin-i jogokkal lehetséges, hiszen össze kell kötni a főnököt az alkalmazottjaival, nem engedhetjük, hogy bárki regisztrálhasson, az biztonsági kockázatot jelentene. Ennek kiküszöbölésére egy előre választott jelszót kap minden regisztrált felhasználó (*Tesztjelszo*), melyet bejelentkezés után meg tud változtatni egy általa választottra., erre bejelentkezés után egy üzenet fel is hívja a figyelmét.

*Bejentekezéskor* a jogosultsági szint szerint az applikáció eldönti, hogy a felhasználó az alkalmazott vagy a vezetői csoportba tartozik, ez alapján irányítja tovább a megfelelő oldalra.



2. ábra BeosztasApp funkciói

Az alkalmazottak a *Home* felületről a következő opciók közül választhatnak:

* + - *Beosztásom* fül alatt megtekinthetik beosztásukat, ahol szín kódolva lesznek a napok az alapján, hogy az adott nap minek számít (munkaszüneti, munkanap, hétvége, vagy pihenőnap, betegszabadság, szabadság egyéb, illetve elfogadott e az adott nap vagy nem). Itt lehet új szabadságot igényelni, betegszabadságot felvinni. Ehhez Spinnert használtam, amiből csak ez a három lehetőség választható megakadályozva a felhasználói hibákat. Miután a Spinnerből kiválasztja, melyik típusú szabadságot szeretné igényelni, és kiválasztja a napot a naptárból a megfelelő napra kattintva a mentés gombra nyomva elküldi a szabadság kérelmet a vezetőjének. Itt még fel van tüntetve az eddig felhasznált szabadságok mennyisége, és a felhasználható szabadságok mennyisége.
    - *Értesítéseim* fül alatt lehetőség lesz a vezető által elfogadott változtatásokat megtekinteni, mint a függőben lévő szabadságok igénylésének a jelenlegi állapotát.
    - *Korábbi szabadságaim* alatt megtekinthetők az adott évben kivett szabadságok listája (mikor lett kivéve, típusa és hogy jelenleg elfogadott vagy sem).
    - Végül a *Személyes adatok változtatása* opció alatt lehetőség van a személyes adatok módosítására, azaz lecserélni a jelszót
    - *Kijelentkezés* fül alatt lehetőség van kijelentkezni

Az vezetők a *Home* felületről a következő opciók közül választhatnak:

* + - A vezetők ugyanúgy megtekinthetik a beosztásukat (amikor egy vezetői jogosultsággal rendelkező egyén igényel szabadságot az egyből elfogadásra kerül), korábbi szabadságaik listáját, módosíthatják személyes adataikat és kijelentkezhetnek.
    - Az *Értesítéseim* fül alatt láthatják a beosztottjaik által igényelt szabadnapokat és lehetőség van ezek jóváhagyására, elutasítására.
    - *Beosztottaim* fül alatt megtekinthető a jelenlegi munkavállalók listája és lehetőség van az adott vezetőhöz tartozó munkavállaló törlésére.
    - Az *Új munkavállaló regisztrációja* fül alatt lehetőség van új munkavállaló felvételére

A design megtervezésénél Tisza-Kis Eleonóra tervezetét követtem, aki a webes részt tervezte. Próbáltam ezt a stílust az Android-os platformra is átvinni.

## Adatmodell leírása

Első lépés az alkalmazás elkészítésében az adatbázis megtervezése volt, hiszen ez köti össze a három platformot, mindegyik platform igényeinek eleget kellett tennie, ennek elkészülte előtt nem lehetett a tervezést elkezdeni. Az applikációban 5 különböző táblázatot hoztunk létre, illetve egy eljárást, melyeket a következőkben részletesebben is szeretnék bemutatni. Az adatbázis megtervezése után egyből beleütköztem az első nagyobb akadályba, mégpedig hogy a Kotlin nem igazán köthető össze egyszerűen a localhost-tal. A modern alkalmazások többnyire felhő alapú adatbázisokat, illetve API-okat használnak az adatbázissal való összeköttetéshez. Mivel a mi adatbázisunk a localohost-on található, így a mi igényeinknek ez nem felelt meg. Végül arra a döntésre jutottunk, hogy egy Sqlite alapú helyi adatbázist hozunk létre, mely rendszeres időközönként szerviz frissítés keretében frissíti, összefésüli az adatbázisokat. Viszont maga az adatbázis szerkezetén nem változtattunk semmit, ugyanazt az alapot használtuk mind a webes, mind az Android-os alkalmazáshoz. A következőkben szeretném pár szóban részletezni az általunk használt különböző táblázatok tulajdonságait és a köztük lévő kapcsolatot.

A webes felület adatbázis terve, melyet az Android-os adatbázishoz is használtam a következőképpen néz ki:

* **Személyek** tábla

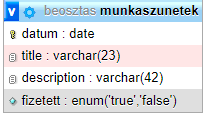
3. ábra személyek tábla

Első és egyben legfontosabb táblázatunk a személyek tábla. Ez fogja tartalmazni a munkavállaló, illetve munkáltató legfontosabb személyes adatait, melyek az azonosításhoz, illetve hitelesítéshez szükségesek. Számunkra, a későbbiekben fontos lesz a belépésnél a jelszó és az e-mail cím, melyek kombinációja egyértelműen azonosítja a felhasználót. A szemely\_id egy max 11 hosszú egész szám, mely *elsődleges* kulcsként szerepel a táblázatunkban, és *idegen* kulcsa a *jelenlétik* táblázatnak (szintén szemely\_id néven). A nev egy max 30 karakter hosszú String, mely a felhasználó nevét tárolja. A fonok (max 11 hosszú egész szám) a beosztott főnökének szemely\_id-ja, ha értéke nulla, akkor ő a vezető, nincs főnöke. Az adóazonosító egy 8 számjegyből álló egész szám. Fontos megemlíteni, hogy munkarend esetében az1 kötött a 0 pedig kötetlen munkaidőt jelöl. Ez *idegen* kulcsként szerepel a *munkarend* táblázat elsődleges kulcsának (munkarend\_id), itt szerepel az adott munkarend részletes leírása is. A belépés Date típusú, és azt jelöli mikor lépett a munkavállaló alkalmazásba. További adatok az email cím, mely egy 30 karaker hosszú illetve a jelszó, ami egy maximum 30 karakter hosszú String típusú adat. Szükségünk lesz még az adott évben a munkavállaló által kivehető szabadságok számára, illetve a heti munkaidő nagyságára órákban meghatározva (mindkettő max 11 karakter hosszú egész szám). Ezek az adatok szükségesek lesznek a jelenlétek precíz rögzítéséhez és a hivatalos bérszámfejtés elkészítéséhez

* **Munkarend** tábla

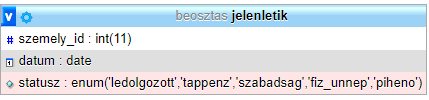
4. ábra munkarend tábla

Ebben a táblában van tárolva a különböző munkarendek szerint dolgozók időbeosztása. Egy csoportba, munkarendbe tartoznak azok, akik ugyanakkor kezdik és fejezik be a munkájukat, illetve a szünetek ideje és a ledolgozandó órák száma is azonos. Jelen állapotban 2 típust különböztetünk meg, munkáltatói igény szerint ezek száma bővíthető. A munkarend\_id a táblázat *elsődleges* kulcsa, mely a *személyek* tábla munkarendjének *idegen* kulcsa. Munkaközi szünet egész szám a szünetet percekben írja le(a munka törvénykönyvében meghatározott napi szünetet meg kell adni, viszont ez nem része a dolgozó munkaidejének). A kezdés egy time változó, mely megadja, hogy a fix munkarend szerint mikor kezdik a munkát, illetve a befejezés pedig, hogy mikor fejezik be (pl kezdes 8:00, befejezes 16:30). A szunet\_kezdete és szunet\_vege szintén time típusú és a munkaidőhöz hasonlóan a szünetek időpontját jelölik (szunet\_kezd 12:00, szunet\_vege 12:30). A ledolgozott óra egy maximum 11 karakter hosszú egész szám, mely megadja a munkában töltött időt a munkaközi szünet nélkül órákban.

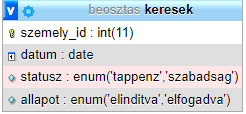
* **Munkaszunetek** tábla

5. ábra munkaszünetek tábla

Tárolja a különböző magyar nemzeti ünnepeket, dátum illetve leírás alapján. A datum jelöli, hogy mikor van a nemzeti ünnep és date típusú. A title egy maximum 23 karakter hosszú String, mely leírja, hogy melyik ünnepről (pl.: Nagypéntek) vagy pedig a hosszú hétvégéből származó pihenőnapról vagy az általa okozott szombatra áthelyezett munkanapról van szó. A descriptionba (max 42 hosszú karakterlánc) szerepel pár extra információ, mint hogy 4 napos hétvége, munkaszünet stb. . A fizetett opció hozzáadásának a szükségességét a indokolta, hogy a munkaszüneti nap eshet hétvégére is, mely esetében ez a nap a fizetésbe nem számolódik be, tehát szükséges tudnunk, hogy a havi bér kalkulációjába ezek a napok bele fognak e számítani.

* **Jelenletik** tábla

6. ábra jelenletik tábla

Rögzíti az adott munkavállaló adott napon teljesített munkaidöjét, ha történt munkavégzés. A szemely\_id (max 11 karakter hosszú egész) a táblázat *elsődleges* kulcsa, mely a szemelyek tábla szintén szemely\_id nevezetű kulcsával van összekötve. A statusba mentjük, hogy az adott keltezéshez, milyen esemény történt munkanap/táppénz/szabadság/egyéb/fizetett ünnep/ pihenő(hosszú hétvégéből fakadó). A datum date típusú és rögzíti, melyik napon történik az ****előbbi esemény.

* **Keresek** tábla

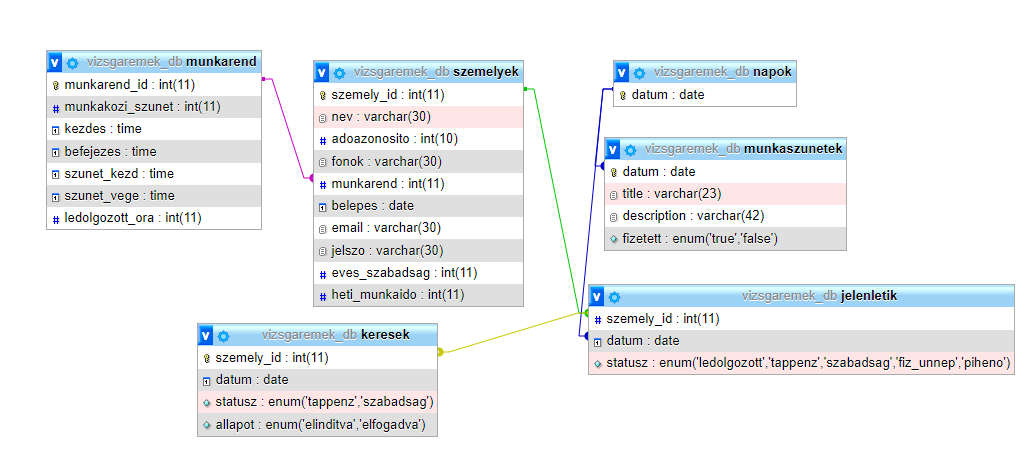
Annak tárolására szolgál, hogy egy adott személy a regisztráció óta milyen típusú szabadságokat, illetve mikor vett ki. A szemely\_id az *elsődleges* kulcs (max 11 karakter hosszú egész), mely a *személyek* tábla szemely\_id-jával van összekapcsolva. A datum date típusú és menti az adott szabadság időpontját. Ezek típusa (status) háromféle lehet táppénz, szabadság vagy egyéb, így enum változókba mentettük őket. Elfogadásuk a munkáltató feladata, állapotuk (allapot) lehet még csak „*elindított*” vagy már „*elfogadott*”, tárolásuk a statusban történik, mely egy enum típusú változó és csak ezt a két értéket veheti fel. Ez segít a szabadságok számának nyomon követésében és a bérkalkulációs számolásan is.

7. ábra keresek tábla

* **Napjelleg** eljárás

A Magyarországi szabályozás szerint 6 különböző típusú napot különítünk el: munkanap, ünnepnap, pihenőnap (az ünnepnap és hétvége között esedékes nap, mely egyedülálló módon, Magyarországon nem számít munkanapnak), szombatra áthelyezett munkanap, illetve a hétvége, azaz szombat és vasárnap. Fontos annak az eldöntése, hogy az adott naptári nap, melyik kategóriába esik, hiszen ez fogja meghatározni, hogy az adott naphoz tartozó fizetés, hogyan számolódik.

* **ER diagram**, azaz kapcsolat a táblák között

A következő táblázat összefoglalja a fentebb részletezett kapcsolatot a táblázataink között:

8. ábra Adatbázis diagram

# **Részletes feladatspecifikáció**

Az alkalmazást Android Studio 2020.3.1-ben írtam Kotlin nyelven. Választásom azért esett erre a nyelvre, mert 2019-től ez a Google első számú preferált nyelve az Android-os alkalmazások fejlesztése terén. Maga a nyelv statikus, objektum orientált nyelv, mely a JAVA Virtual Machinet használja fordítói környezetként.

Mivel a Kotlin nem képes közvetlenül a phpmyadminhoz hozzáférni ezért az első megoldandó problémát egy megfelelő API készítése volt.

Sajnálatos módon ezt a tervet el kellett vetni, mivel a Kotlin localhost-hoz való kapcsolása meddőnek bizonyult. A github-on számos megoldás található, de ezek közül még a frissebek is elavulttá, azaz használhatatlanná váltak, illetve amelyek nem bizonyultak egyből hibásnak azok tesztelés során futottak hibára. Emiatt arra a döntésre jutottunk, hogy az Android alkalmazás adatbázisát lokálisra (sqlite3) cseréljük, megtartva a webes alkalmazás adatbázisának szerkezetét. A két platform bizonyos időközönként szervizfrissítésre kerül, azaz az adatokat összefésüli.

Elsőként szükséges volt létrehozni a felhasználó azonosításának illetve authentikációjának (jogosultsági szintjének) eldöntésére egy querty, mely lekérdezi az adatbázis megfelelő táblájából a felhasználó adatait. Döntést kell hozni, hogy ha szerepel a felhasználók között, akkor milyen felhasználói jogokkal rendelkezik. Mi két különböző jogosultságot vezettünk be, egyik a munkavállalói, másik a munkáltatói jogkör. Ennek a típusa fogja eldönteni, miket tehet, illetve láthat a felhasználó, ez fogja meghatározni, hogy a bejelentkező oldal után, melyik oldalra lép tovább a felhasználó és miket engedünk meg neki, hogy csináljon.

## A program lényeges függvényeinek, eljárásainak, az osztályok metódusainak a specifikációja

Az *Első* nagyobb megoldandó akadályt jelentette egy olyan osztály létrehozása, mely kommunikálni fog az sqlite3 adatbázissal. SQLiteOpenHelpert használtam, aminek az alapját képezte az alap CRUD műveletek végrehajtásának megvalósítása. Akármikor az adatbázissal kommunikálunk, ezt az osztályt fogjuk meghívni. Ehhez körülbelül 20 darab függvényt írtam, mely a szükséges lekérdezéseket fogják végrehajtani. A checkUser függvény ellenőrzi, hogy az adott e-mail címmel szerepel e munkavállaló az adatbázisban, a checkSzemely pedig hogy a bejelentkezésnél megegyeik e az adott e-mail-jelszó kombináció. A getSzemely függvény pedig az adott id-hoz tartozó személyt adja vissza. A személyekre létrehoztam egy osztályt, amelyben a személy tábla adatait tároltam. A getSzabadsag függvény az összes eddigi szabadságból (keresek táblából képzett osztály) készített tömböt térít vissza, ezt használtam a Kalendárium eventjeinek a feltöltésére. Az addSzabiJelenletik, AddSzabiKeresek, InsertJelenletek függvények a megfelelő táblákba való beillesztést végzik. A beosztottaim függvénnyel az adott fonokID-hoz tartozó lekérdezést hajtjuk végre, melyre a főnök felület beosztottaim fülnél lesz szükség. Az ertesitesek az adott szemelyIDhoz tartozó szabadságokból képzett listát, az ertesitesekSzabi az adott szemelyIDhoz tartozo még nem elfogadott szabadságokból képzett listát ad vissza. A checkalreadyEnterSzabi függvény ellenőrzi, hogy az adott napon van e már igényelt szabadság, hiszen ha már van nem szeretnénk kétszer beilleszteni a táblázatunkba. Az ertesitesekF egy kicsit bonyolultabb, itt össze kellet kapcsolni a szemelyek táblát a keresek táblával, hiszen csak az adott vezetőhöz tartozó beosztottak szabadság kérelmeit szeretnénk listázni. Ennek a kezelésére szintén létrehoztam egy osztályt. A checkjelenletek az adott szemelyIDhoz tartozó jelenléteket ellenőrzi, erre a Calendarnál volt szükség, mivel ha ez hamis értéket ad vissza, akkor lefut egy fillUpJelenletik függvény, mely feltölti az adott személyhez tartozó jelenletik táblát az év minden napjára az alap beállításokkal. A checkSzabadsag visszaadja, hogy az adott szemely az évben eddig, hány szabadságot vett ki, ezt a beosztásom fülön használjuk, hogy jelezzük hány szabadság vehető még ki az évben. Az updateUser-t akkor hívjuk meg, mikor a felhasználó szeretné az e-mail címét és/vagy jelszavát megváltoztatni. Az elfogadSzabi és elutasitSzabi a főnök felület értesítéseim fülön van használva, mikor a szabadság elfogadása vagy törlése gombra nyomunk. A deleteUser szintén a főnöki felülethez tartozik, és akkor kerül használatba, mikor a vezető egy alkalmazottját szeretné törölni.

A *második* nagyobb megoldandó akadályt a recycler View Adapter létrehozása okozta, ezekből négy típust hoztam létre, kettőt a különbző felhasználói szintekhez tartozó értesítésekhez illetve egyet a vezetői felülethez tartozó munkavállalók listázásához és a korábbi szabadságok listázásához (ez mindkét szinten ugyanazt az Activity-t használja). A vezetői szinthez tartozó értesítések volt a legkomplexebb így a továbbiakban ennek az elkészültét ismertetném.

Először a layout xml file-okat hoztam létre, melyből az egyik maga a recycler view, másik a dinamikus elemszámú listának egy egysége, ami egy cardviewba épített hat darab linear layout. Ebben az utolsó sor két darab gombot tartalmaz, egyet az elfogadásra egyet pedig az elutasításra. Ezután szükségem volt egy recyclerAdapterre, mely ezeket a cardview-kat feltölti elemekkel. Létre kellett továbbá hozni egy új osztályt (ErtesitesF néven), a személyek és a keresek tábla összekapcsolásából, mely tartalmazza, a keresek\_id-t, szemely\_id-t, kérelmező nevét, a szabadság típusát(szabadság/táppénz/egyéb), és a szabadság kérelem dátumát. Ezek közül a nevet, dátumot, típust fogjuk az onBindViewHolder metódusban a cardview különböző mezőihez kötni. Mivel a lista dinamikus mindig a lista adott pozíciójában lévő objektum property-jeit fogjuk használni az adott cardview feltltésére.

A hozzá tartozó Activity-t össze kellet kötni a container recycler view layout xml-lel, majd az adaptert összekötni a korábban készített recyclerAdapterrel. Az adapternek bemenetként szüksége lesz egy ErtesitesF objektumokból álló tömbre. Ezt úgy értem el, hogy írtam egy lekérdezést:

**"SELECT azon, datum, statusz, szemelyek.szemely\_id, nev FROM szemelyek INNER JOIN keresek ON szemelyek.szemely\_id = keresek.szemely\_id WHERE allapot = 'elinditva' AND fonok = " + fonokID.toString() + ";"**

Ahol a fonokID a query függvényünk bemeneti értéke. Cursor használatával elkészítjük azon értesítésekből álló listát, melyek az adott id-jú főnökhöz érkeztek be.

Ezek után minden egyes megnyitásnál pontosan annyi cardview-t fog megjeleníteni az alkalmazásunk, amennyi az adott lekérdezés eredménye a táblázatunkból.

A következő megoldandó problémát jelentette, hogy a két gombhoz azaz az elutasításhoz, elfogadáshoz hozzá kellett szintén egy-egy query műveletet rendelni. Elfogadás esetén update-elni kell a recordot és az ’elinditva’ tulajdonságot az adott keres\_id-nál ’elfogadva’-ra állítani. Míg elutasítás esetén törölni kell a rekordot a táblából. A nehézséget az jelentette, hogy az adapterben nem lehet az sqlLiteHelpert példányosítani, így magát a lekérdezéseket nem tudtam meghívni. Erre azt a megoldást találtam, hogy létrehoztam a következő property-t illetve metódust:

private var onClickDeleteItem: ((szemelyek) -> Unit)? = null  
fun setOnClickDeleteItem(callback: (szemelyek) -> Unit) {  
 this.onClickDeleteItem = callback  
}

Ezen kívül még hozzá kellet adni a következő pár sort a onViewBindHolder-hez

viewHolder.elfogadButton.setOnClickListener **{** onClickElfogadItem?.invoke(tmpErtesites) **}**viewHolder.elutasitButton.setOnClickListener **{** onClickElutasitItem?.invoke(tmpErtesites) **}**

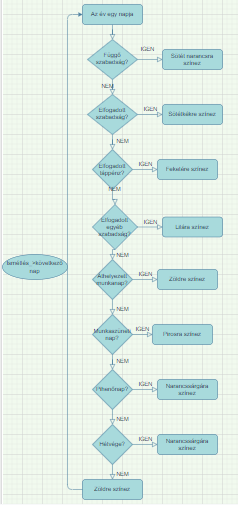
Majd a hozzá tartozó Activity-ben meg kellet hívni az alábbi módon

adapter?.setOnClickElfogadItem **{** elfogadErtesites(**it**.keresek\_id)  
 getErtesitesek()  
**}**adapter?.setOnClickElutasitItem **{** elutasitErtesites(**it**.keresek\_id)  
 getErtesitesek()  
**}**

Ezek után a gombokra kattintva végre tudjuk hajtani a kívánt műveleteket.

Még egy utolsó akadályt itt meg kellett oldani, mégpedig hogy mikor a felhasználó megnyomja a telefonján a vissza gombot (beépített, nem pedig az alkalmazáson belülin) akkor visszakerüljön a megfelelő kezdőlapra. Ehhez felül kellett írni az onBackPressed metódust, úgy hogy megnyomás esetén indítsa a főnök Home Activity-jét, ügyelve arra, hogy a szemely\_id-t, illetve a nevet az extrák közé mentsük, mielőtt elindítjuk, hogy ezek az értékek a későbbiekben is elérhetők legyenek.

## Az algoritmizálható részek leírása valamilyen algoritmus-leíró eszközzel

A *harmadik* nagyobb megoldandó akadályt az Android-os Kalendárium létrehozása jelentette. A naptár létrehozásánál próbáltam a webes alkalmazás designját követni, azaz hogy a napok hátterét a nap típusának megfelelően színezni. Ez nehezebb feladatnak bizonyult, mint elsőre tűnt. Végül Applandeo Material Calendar View-ját használtam (github), mely segítségével különböző színű ikonokat (saját magam által készített) tudtam a napokhoz rendelni, mely jelöli a nap típusát. Ehhez létrehoztam néhány függvényt (isAthelyezett, isMunkaszunet, isPiheno), mely az adott nap stringként való megadása után eldönti, hogy beletartozik e valamelyik csoportba (visszatérítése boolean). Az eldöntéshez mindegyik típushoz létre lett hozva egy lista, mely tartalmazza a kérdéses napokat. A hétvége eldöntésére a beépített Kotlin metódusokat, mint Calendar.SUNDAY és Calendar.SATURDAY használtam. Szükség volt a eddig igényelt szabadságok lekérdezésére is a keresek táblázatból. Ehhez egy queryt használtam, mely visszaad egy keresek osztályegyedekből álló tömböt. Az Activityben ezt a tömböt használtam az adott egyénre vonatkozó szabadság típusának (táppénz/szabadság/egyéb-elfogadott/elindított) meghatározására, illetve hogy melyik napra érkezett az igény. Az adott szabadság felvitelére Spinnert választottam, melyből kiválasztható annak típusa, azaz hogy táppénz, szabadság vagy egyéb. A választott nap változására létrehoztam egy DayClickListenert, mely a get methodus különböző paraméterei alapján létrehozza a kiválasztott napból létrehozott stringet. Ezt fogja használni, hogy visszaküldje az adatbázisnak, mint választott napot, mikor a Mentés gomb lenyomásra kerül.

9. ábra Folyamatábra a napok típusának eldöntésére

# **Tesztelési dokumentáció**

1. Kitöltésből adódó hibák kiküszöbölése

A Bejelentkező oldalnál, ha a felhasználó valamelyik mezőt nem tölti ki, akkor az alkalmazás üzenet formájában figyelmezteti. Szintén, ha a jelszó és e-mail páros nem egyezik egyik felhasználóéval sem akkor nem engedi tovább, üzenet formájában tájékoztatást közöl (ezekhez Toast-ot használtam).

A személyes adatok megváltoztatásánál mind a régi jelszó, mind az e-mail cím megadása kötelező a jelszó megváltoztatásához. Itt ellenőrzi, hogy a megadott e-mail jelszó megfelelő e (korábbi). Ha igen akkor kitöltött e az új jelszó két mezője és ezek megegyeznek e (Snackbarral jeleztem).

2. A szabadságok igénylése és elfogadása közben adódó hibák kezelése

A beosztásom fülnél csak akkor lehet szabadságot igényelni, ha a kalendáriumban kiválasztunk előbb egy dátumot (ez meg is van jelentetve a választott dátum sor alatt) majd a legördülő menüből a megfelelő típusú szabadságot (alapértelmezett a táppénz) Ezzel szerettem volna megakadályozni a különböző felhasználói hibák egy részét. Nem kérhetünk szabadságot múltbéli hónapra, hétvégére (kivéve ha áthelyezett), munkaszüneti és pihenő napra. Továbbá nem lehet egy napra több szabadságot is igényelni vagy több szabadságot, mint amennyi rendelkezésünkre áll (ezen a felületen az összes üzenetet Toast formájában közöltem).

1. Figyelmeztető üzenetek

A vezetői felületen az Értesítéseim fülön az elfogadás, elutasítás gomb megnyomása után felugrik egy figyelmeztető ablak, hogy valóban végre szeretné e a felhasználó hajtani a műveletet. Itt még van lehetőség a változtatásra, utána már nem, mert végrehajtja a változtatásokat a táblázatunkba (AlertDialog.Builder formájában valósult meg a figyelmeztetés).

A vezetői felületen az Új munkavállaló regisztrációja fülön ellenőrzi a mezők kitöltöttségét, típusát, például adószám csak egy 8 jegyű egész szám lehet, a munkarend csak 0 vagy 1, illetve a szabadságok száma és a heti munkaóra is csak egészek lehetnek. Ez után még egy további ellenőrzést is csinál mégpedig, hogy az adott e-mail címmel rendelkezik e már felhasználó (ezen adatok közlésére szintén Snackbart használtam)

A vezetői felületen A Beosztottaim fülön lehetőség van a beosztottak törlésére. Itt a gomb megnyomása után és a végrehajtás között szintén felugrik egy figyelmeztető ablak megelőzendő a hibákat, mivel ezután visszavonásra már nincs lehetőség.

Minden egyes felületet külön teszteltem az elkészülte folyamán, főleg ügyelve arra, hogy a kötelezően kitöltendő mezőket ne lehessen üresen hagyni. Ellenkező esetben valamilyen hibaüzenet formájában értesítést adjon a felhasználónak az elkövetett hiba létezéséről és típusáról. Fontosnak tartottam, hogy az adatok típusát is mindenképp ellenőrizze, hogy rendben legyenek. Ezeknek főleg a főnöki felület regisztrációs felületén van jelentősége, illetve az adatok változtatása fülön. A jelszó megváltoztatása jelenlegi formájában még továbbfejleszthető. Itt lehetne különböző feltétleket adni arra, hogy milyen karaktereket kell tartalmaznia (pl. kis- és nagybetű, speciális karakterek, számok), illetve mi legyen a minimális hossza. Próbáltam kikerülni minden lehetséges felhasználói hibát, ezért a regisztrációnál Date-Spinnert használtam, biztosítva, hogy biztosan dátum formátumot válasszon. A minimális időpontot is beállítottam az adott napra, így a múltban sem lehet munkavállalót hozzáadni.

1. Tesztelés közben felvetődött hibák kezelése

Második lehetőség a hibázásra a beosztásom fül volt, ahol kiválasztja, hogy mikor szeretne szabadságot kivenni. A hibák elkerülésére itt is kalendáriumot használtam, így a dátum formátum adott volt, a minimális időpontot a kezdés dátumához igazítottam, így az előtte lévő napok nem kattinthatók. Az adott napra való igénylés sem biztosított, mivel a gombra való kattintás ellenőrzi mi volt az előző kattintott nap és arra igényli a szabadságot. Elég sok tesztelés igényelt, míg sikerült tisztázni minden napot a helyére tenni, úgy értendő, hogy melyik napra igényelhető szabadság. Első körben az is problémás volt, hogy a Kotlin Calendar.MONTH\_OF\_YEAR indexelése 0-val kezdődik (míg a nap és év 1-gyel), ez elég zavaró volt. Ennek következtében az összes hónapom el volt csúszva, nem jó helyre illesztette be az igényelt szabadságokhoz tartozó eseményt. Ugyanez igaz volt a dátumozásra, az első változatan sikerült az év-hónap-nap sorrendet felcserélnem, így sajnos mikor beillesztette a táblámba. Ez kiolvasásnál hibákat okozott, melyeket nem volt egyszerű felismerni.

A többi fülön ügyeltem arra, hogy ahol lehet, ott ne legyen felhasználói input felület, ezeknél többnyire csak gombokat lehet kattintani, melyek végrehajtás előtt mindig megerősítést kérnek végrehajtás előtt.

Az utolsó megoldandó problémát, mely tesztelés során feljött (a helyesírási és design jellegű hibák javítása után), az az egyes gombok megnyomása és így a query elküldése után a felület és így az adatok frissítése volt. Ezt úgy oldottam meg, hogy újratöltöttem az Activity-t a gomb megnyomása után, így mindig a friss adatok jelentek meg.

Ezen kívül szerettem volna még a beépített androidos visszafele gombot is módosítani, mert alapbeállításként ez az előző lépéshez visz vissza. Én azt szerettem volna, ha a home felületre vinne, így felül kellett írnom a kotlin egy beépített metódusát is.

# **Továbbfejlesztési lehetőségek**

Az egyik legfontosabb fejlesztési lehetőségnek látom, hogy érdemes lenne az összes platformhoz egy közös API-t létrehozni, amelyet mindegyik platform tudna használni. Ez egy elég nagy és munkaigényes feladat, de mindenképp hatékonyabbá tenné a későbbi fejlesztéseket, ezért mindenképp érdemes lenne rá időt szánni.

Továbbá jelenleg a alkalmazásunk csak rögzített munkaidővel dolgozik, érdekes lenne tovább bővíteni azon irányba, hogy minden nap szabadon felvihető legyen, hogy ki mettől meddig dolgozott, esetleges törzsidők bevezetésével.

Lehetséges továbbfejlesztése lehet a távolmaradási napok típusának bővítése pl. csúsztatás, kiküldetés(ez más fizetési kategóriába eshet). A csúsztatás esetében szükség lenne nem csak a szabadságok jelzésére az applikációban, hanem a hónapban elvárt és teljesített munkaóra jelzésére is

Végül lehetséges lehetne a havi bérkalukláció applikációból való indítására vagy esetleg el is lehetne azt végezni az alkalmazás keretein belül is.

# **Felhasználói dokumentáció**

A kvetkező dokumentáció célja, hogy segítséget adjon a leendő felhasználónak a program telepítésével és használatával kapcsolatos minden probléma megoldásában, segítse a program kezelésének az elsajátítását.

## Hardver követelmények

Az applikáció működéséhez mindenképp szükség lesz egy telefonra Android-os operációs rendszerrel minimális API21: Android 5.0 (Lolipop) rendszerrel. A mérete nem túl nagy 30 MB alatti, így ennyi szabad hely megléte elegendő. A gépem adatai, amelyen készítettem: Processzor-Intel(R) Core(TM) i5-5300U; CPU: 2.30GHz, 2301 Mhz, 2 mag, 4 logikai processzor; telepített fizikai memória (RAM): 8,00 GB. Ennek a minimális megléte elegendő az Android Studiobol való telepítéshez és így az alkalmazás telepítéséhez.

## Szoftver követelmények

Jelenleg az applikáció Android operációs rendszeren fut, ennek minimális követelménye az Android 8.2. Google play áruházból való letöltésre nincs lehetőség, jelenleg csak Android Studio-ból telepíthető. Az sqlite adatbázis az alkalmazáson belül található, nincs szükség annak külön telepítésére.

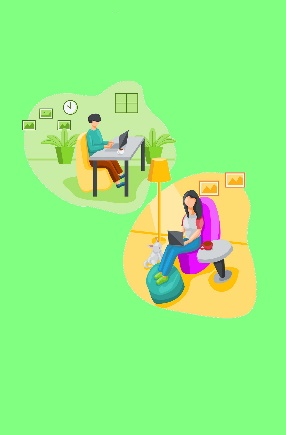
# **A program telepítésének és konfigurálásának a leírása**

A program telepítéséhez jelenleg Android Studio-ra van szükség, mivel nem érhető el a Google play áruházban. A következő Github-os linkről letöltve az alkalmazást

https://github.com/JuditFerenczik/BeosztasApp

Android Studioban való megnyitás után tudjuk telepíteni a megfelelő telefonnal való USB összeköttetés után, ha futtatjuk a programot. Ehhez nem csak az Android Studio telepítésére, hanem a telefon webfejlesztő funkcióinak engedélyezése után van lehetőség. Ennek ismertetését nem részletezem tekintve, hogy ez minden telefonon másképp történik.

## A program használatának a részletes leírása

1. Az alkalmazás indításakor ez a kép várja a felhasználót töltés közben, hogy ne kelljen sokáig tétlenül várakoznia.

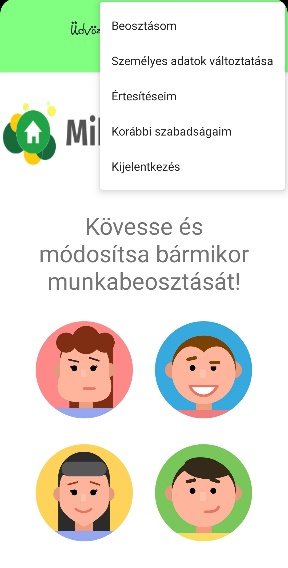
10. ábra Applikáció betöltése



11. ábra Bejelentkező felület

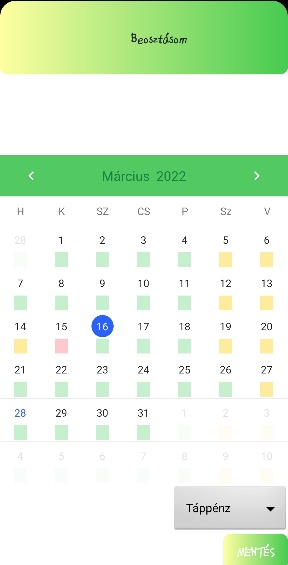
1. A betöltés után az első felület, ami fogadja a felhasználót az a bejelentkező felület. Itt megadhatja a felhasználó az e-mail címet és jelszavát majd a bejelentkezés gombra kattintva lefut egy függvény, amely az adatbázissal egyeztetve eldönti, hogy a felhasználó regisztrált e a rendszerben és ha igen, milyen felhasználói jogokkal rendelkezik. Ezen jogok alapján tovább irányítja az Alkalmazott vagy a Vezető felületre. Itt szeretném ezeket külön választani.
2. Alkalmazott Felület

12. ábra Alkalmazotti Home felület

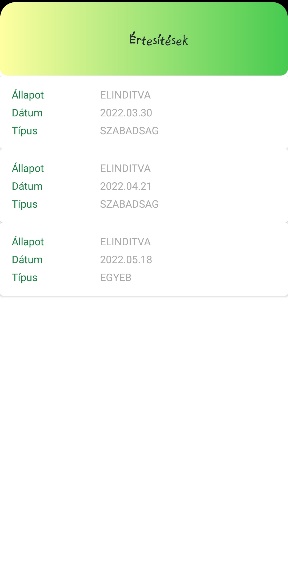


1. Ez az alkalmazottak kezdőfelülete, a 3 pontra kattintva lehet a további menüpontokhoz hozzáférni.
2. Az első almenü a Beosztásom. A napok alatti színes négyzetek jelölik az adott naphoz tartozó munkanap típust. A szabadság mennyisége jelöli, hogy még hány darab szabadságot vehetünk ki a tárgyévben. A napra kattintva kiválasztjuk, melyik napra szeretnénk szabadságot, mely a fenti szövegdobozban meg is jelenik a kattintás után. Ezután ki kell választani a legördülő menüből, milyen típusú szabadságot szeretnénk igényelni (Táppénz /Szabadság/Egyéb) és a Mentés gomb megnyomása után ezt el is küldi a munkáltatónak.

13. ábra Beosztásom felület a szabadságok igénylésére

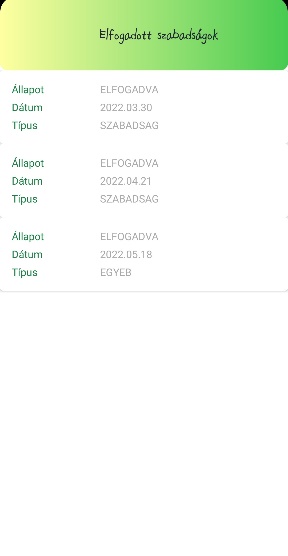


1. A második almenü a Személyes adatok változtatása. Itt lehetőség van a jelszó megváltoztatására, az e-mail cím és a régi jelszó megadása után az új jelszót kétszer megadva (ezeknek meg kell egyeznie) a Mentés gombot megnyomva megváltozik a jelszó.

4. A harmadik almenüpont az Értesítések, ahol a függőben lévő szabadság igénylések listája tekinthető meg.

14. ábra Személyes adatok módosítása

15. ábra értesítések fül

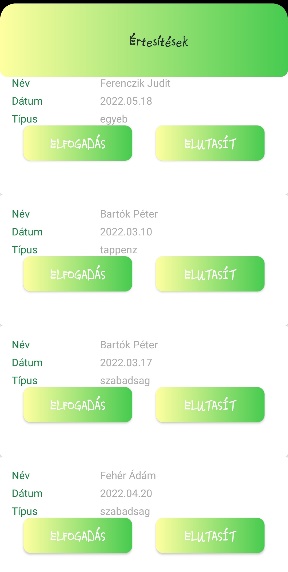
1. A negyedik almenüpont a Korábbi szabadságaim, ahol a már elfogadott szabadságok kerülnek listázásra.
2. Az ötödik menüpont a Kijelentkezés, amellyel a bejelentkező felületre lehet visszajutni.

16. ábra Korábbi szabadságok listázása

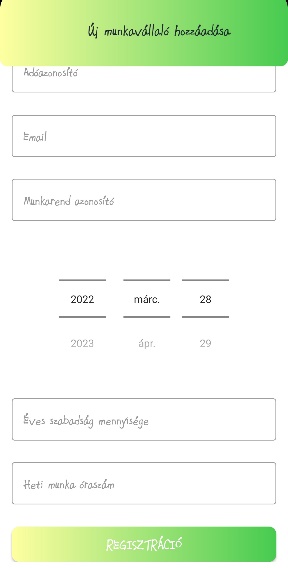
1. Vezetői Felület
2. A vezetői jogokkal rendelkező felhasználókat ez a felület fogadja, ahol a 3 pontra kattintva a következő almenüpontok érhetők el. A Személyes adatok változtatása, Korábbi szabadságaim, Kijelentkezés almenüpontok ugyanúgy működnek, mint az alkalmazott jogkörrel rendelkezőké, így azok ismertetését itt kihagyom. A Beosztásom menüpont is ugyanaz azzal a kitétellel, hogy a vezető szabadság igénylései azonnal elfogadásra kerülnek.

17. ábra Vezetői Home felület

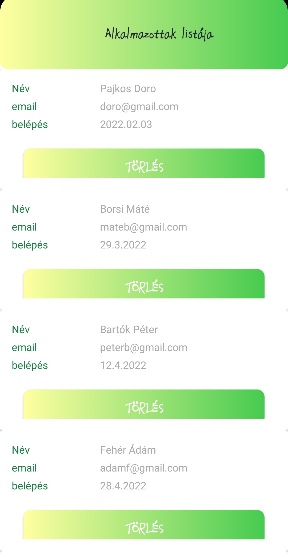


2 A következő almenüpont az Értesítések, ahol az adott vezető a hozzá tartozó munkavállalók által igényelt, különböző típusú szabadságok tekinthetők meg. Ezek elfogadására, illetve elutasítására a megfelelő gombokra való kattintással van lehetőség.

18. ábra Szabadságok elfogadása/elvetése

3. Új munkavállaló regisztrációja almenüpont alatt a vezetőnek lehetősége van új munkavállalók hozzáadására a megfelelő mezők mindegyikének (kötelező) kitöltése után. A mezők kitöltésénél formai követelmények, hogy egyik sem lehet üres, illetve hogy az adószám 8 jegyű egész szám, szabadságok száma és a heti munkaidő pozitív egészek, a munkarend pedig csak 0 vagy 1 lehet. A belépés dátuma minimalizálva van a tárgynapra, így múltban hozzáadni munkavállalót nincs lehetőség. A Regisztráció gombra nyomva hozzáadja az új munkavállalót a személyek táblához saját munkavállalójaként.

19. ábra Új munkavállaló regisztrációja

1. A Beosztottaim menüpont alatt a vezető megtekintheti a hozzá tartozó munkavállalók listáját és a Törlés gombra kattintva lehetséges a munkavállaló törlése is.

20. ábra munkavállalók listája

# **Összegzés**

A cél egy munkaidő kezelő applikáció létrehozása volt, mely kétféle jogosultsági szinttel rendelkezik, mint alkalmazotti és vezetői. Az alkalmazásban szerettem volna szerepeltetni az alapvető CRUD műveletek végrehajtását az adatbázisunkon. Ezt sikerült is hiánytalanul végrehajtani, közel 20 lekérdezést sikerült az alkalmazásban implementálni a könnyebbektől az összetettebbekig. Az ertesitesekF lekérdezés volt a legkomplexebb, itt több táblát kellett összekapcsolni és a visszatérítési tömbre egy külön osztályt is létre kellett hozni. Ezen kívül még négy táblázatokhoz köthető osztályt hoztam létre a különböző lekérdezések visszatérítési tömbjéhez, illetve recyclerView adapterhez köthető osztályt is. Ezek létrehozása is kihívást jelentett, mivel itt a view dinamikus, mindig attól függ hány nézetet fog tartalmazni, ahány elemünk van éppen jelenleg az adatbázisunk adott táblájában az adott feltételre. A naptárhoz sikerült egy olyan függvényt létrehozni, mely ha érzékeli, hogy a Jelenletik tábla az adott személyre nincs feltöltve akkor az adott évre az összes munkanapját beilleszti (ez egy alapbeállítás, ami akár 365 nap is lehet attól függően, hogy mikor kezdett el dolgozni az alkalmazott).

Jelentős kihívást jelentett egy olyan kalendárium létrehozása, amelyben jelölhetők a különböző típusú napok, illetve ezeknek a módosítási lehetősége. Rengeteg kutatómunka után sikerült egy megfelelőt találni, de még, így sem volt egyszerű az alkalmazása. Minden egyes naptípusra létre kellett hoznom egy függvényt, mely eldönti, hogy beletartozik e egy adott típusba. Ezen kívül a tárolt szabadságokat az adatbázisból nyertem ki és mindhárom típusra egy külön adattömb készült, mely ezt letárolja és felhasználja a kalendárium készítésénél. Az év minden napjához hozzárendeltem egy eseményt, melynek a színét az előző függvények határozzák meg.

Összességében úgy gondolom, hogy sikerült az előre kitűzött célokat teljesíteni. Maga a Kotlin nyelv elég nagy kihívásnak bizonyult, főleg a hibakeresés jelentett sokszor akadályt, ehhez volt a legnehezebb hozzászokni. Az sem könnyítette a helyzetet, hogy az online dokumentáció nagyon sokszor JAVA nyelven íródott és ezt át kellett fordítani Kotlinra vagy legalábbis meg alálni a megfelelő változatát, amit tudtam használni. Rengetegszer előfordult, hogy bizonyos kódok, könyvtárak elavulttá váltak, így más megoldást kellett keresni, ami ismételten kutatómunkát igényelt. Ez rengeteg kutatómunkát és hibakezelést igényelt, ami jelentősen hozzájárult a szakmai fejlődésemhez és sokat tudtam belőle tanulni. Szintén sokat tanultam a közös munkából csapattársaimmal, mely a közös ötleteléssel kezdődött, hogy milyen alkalmazást szeretnénk létrehozni, milyen funkciókra legyen képes, mit tudjon az alkalmazásunk. Ehhez létrehoztunk egy github-os repository-t, melyben rengeteg folyamatábrát tároltunk arról, hogyan képzeljük el a folyamatot. Miután ezt sikerült kivitelezni a közös adatbázisunk megtervezése volt a célunk, mely hosszadalmas munka után sikerült kiviteleznünk.

Azt mondhatom, hogy a közös munka az alkalmazásunkon mind szakmailag mind pedig a csoportmunka területén szerzett tapasztalatok révén jelentősen hozzájárult a szakmai fejlődésemhez. Sokszor kihívást jelentett, de a mélypontok áthidalása után azt tudom elmondani, hogy ebből az élményből rengeteget tudtam profitálni.

# **Irodalomjegyzék, forrásjegyzék**

<https://github.com/Applandeo/Material-Calendar-View>

<http://www.androidtutorialshub.com/android-kotlin-login-register-with-sqlite-database-tutorial/>

<https://github.com/jgilfelt/android-sqlite-asset-helper>

<https://guides.codepath.com/android/using-the-recyclerview>

<https://www.javatpoint.com/kotlin-android-options-menu>

<https://www.youtube.com/watch?v=9LYn-OBO5qE&t=1546s>

<https://www.youtube.com/watch?v=hoK9OOP1KZw&t=6961s>

# **Ábrajegyzék**

[1. ábra BeosztasApp szerkezeti felépítés 6](#_Toc100252011)

[2. ábra BeosztasApp funkciói 7](#_Toc100252012)

[3. ábra személyek tábla 9](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252013)

[4. ábra munkarend tábla 10](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252014)

[5. ábra munkaszünetek tábla 10](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252015)

[6. ábra jelenletik tábla 10](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252016)

[7. ábra keresek tábla 11](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252017)

[8. ábra Adatbázis diagram 11](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252018)

[9. ábra Folyamatábra a napok típusának eldöntésére 16](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252019)

[10. ábra Applikáció betöltése 22](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252020)

[11. ábra Bejelentkező felület 22](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252021)

[12. ábra Alkalmazotti Home felület 23](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252022)

[13. ábra Beosztásom felület a szabadságok igénylésére 23](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252023)

[14. ábra Személyes adatok módosítása 24](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252024)

[15. ábra értesítések fül 24](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252025)

[16. ábra Korábbi szabadságok listázása 24](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252026)

[17. ábra Vezetői Home felület 24](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252027)

[18. ábra Szabadságok elfogadása/elvetése 25](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252028)

[19. ábra Új munkavállaló regisztrációja 25](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252029)

[20. ábra munkavállalók listája 25](file:///C:\Judit\WebremekKépek\Vizsgaremek_dokumentacioAndroid.docx#_Toc100252030)