Szakdolgozat

Palla Gergely István

Nyíregyháza, 2023



Palla Gergely István   
Programtervező informatikus BSc   
Vegera József, Mérnöktanár

**2023**

Munkahét menedzsment saját alkalmazás Android eszközre, IntelliJ IDEA környezetben, SQL adatbázissal, Kotlin nyelven

# TARTALOMJEGYZÉK

1. TARTALOMJEGYZÉK 3

2. BEVEZETÉS 6

2.1 Problémafelvetés 6

2.2 Célkitűzés 6

2.3 Megoldandó szoftverkészítési feladat 6

2.4 Eszközválasztás 7

3. IRODALMI ÁTTEKINTÉS 8

3.1 A jelenlegi technológiák: 8

3.1.1 A Lean és a Just in Time: 8

3.1.2 A Toyota-Kata 9

3.1.2.1 A Kata-Coaching: 12

3.1.3 Miért fontos a megfelelő szervezés: 13

3.2 Jelenlegi megoldások 15

3.2.1 A Jira alapjainak bemutatása 15

3.2.2 A Trello alapjainak bemutatása 16

3.2.3 Jira komplexitási problémák: 17

3.2.4 Biztonsági kockázatok népszerű projektmenedzsment programok esetén: 18

4. A FEJLESZTÉS MENETE 19

4.1 Eszközök telepítése 19

4.2 Android operációs rendszer 20

4.3 IntelliJ IDEA 21

4.3.1 Plugin 21

4.3.2 SDK 21

4.4 Gradle 21

4.5 Git 22

4.6 Tesztkörnyezet 22

4.6.1 Virtuális 22

4.6.2 Valós eszköz használata 23

4.7 Program fejlesztése 23

4.7.1 AndroidManifest.xml 23

4.7.1.1 Alkalmazás indítása 24

4.7.2 Activity 24

4.7.2.1 MainActivity class 25

4.7.2.2 ActivityWorkers class 25

4.7.2.3 ActivityTasks class 26

4.7.2.4 Settings class 26

4.7.3 Felhasználói felület 26

4.7.3.1 Layout xml 27

4.7.3.2 Layout alkalmazása 28

4.7.4 További fájlok 28

4.7.4.1 Model mappa 28

4.7.4.2 Data mappa 28

4.7.5 SQL-ről alapjaiban 29

4.7.6 SQLite-ról alapjaiban 30

4.7.6.1 SQLite syntax egyediségek: 31

4.7.7 További tervezett funkciók és működés 31

4.7.7.1 Üdvözlő képernyő 31

4.7.7.2 Jogosultságok 32

4.7.7.3 Szerver SQL 33

5. ELEMZÉS 34

6. KONKLÚZIÓ 35

6.1 Végeredmény 35

6.2 Megoldások 35

6.3 Kotlin értékelése 35

6.4 Lehetséges hasonló szakdolgozatok 35

7. IRODALOM JEGYZÉK 36

# BEVEZETÉS

## Problémafelvetés

A szakdolgozatom témája, olyan Android alkalmazás fejlesztése volt, ami segít egy vezető beosztású dolgozónak a többi alkalmazott felé a feladatok kommunikációját és vissza. Az oka, hogy ezt a témát választottam, hogy nagyon gyakran nem áll rendelkezésre számítógép mindenki számára és a személyes feladat kiosztása is néha korlátok közé szorul. Ha a csapat egy része home office-ba kényszerül vagy utazás miatt válik elérhetetlenné, de az igény a kommunikációra folyamatos, így az ő munkájukat segíthettem, ha Android-ot használó eszközökön megtalálható ez az alkalmazás.

## Célkitűzés

A kitűzött célom Android operációs rendszer, és az erre a rendszerre készülő alkalmazás készítésének menete, és a felhasznált eszközök mélyebb megismerése. Segíteni egy csapat működését egy projekten belül, vagy csak a mindennapokban, megkönnyíteni a munkát és az elvégzését.

## Megoldandó szoftverkészítési feladat

A célkitűzés teljesítését egy az Android operációs rendszerre történő alkalmazással kívánom bemutatni, ami munkahét menedzser néven kívánok fejleszteni. Ezen alkalmazásnak tudnia kell több eszközön több ember számára különböző jogosultságokkal feladatokat kiosztani, a feladatokat böngészni, és jelezni a felmerülő problémákat, valamint a teljesítést. Felhasználói szinten csekély informatikai ismeretekkel is jól használhatónak kell lennie, de magasabb szinten is az egyszerűbb jobb elvvel kell működnie.

## Eszközválasztás

A fejlesztéshez a Kotlin nyelvet találtam a legideálisabbnak, mivel nem közvetlenül az eszközön fut, hanem egy virtuális gépet hoz létre a futtató eszköz, melynek specifikációi adottak, így a különböző eszközökön közel azonosan tud lefutni. A Kotlin egy objektum orientált nyelv, mely a Java programnyelvek családjába tartozik, de a szintaxisa nem kompatibilis a Java-val.

A fejlesztését a JetBrains cég végezte és a célja a saját eszközeinek népszerűsítése volt. A JetBrains által fejlesztett fejlesztőkörnyezet az IntelliJ IDEA, melyen plugin segítségével támogatott a Kotlin nyelv.

Mindezek mellett elengedhetetlen az adatok hosszú távú tárolása és ezek hozzáférhetősége, melyet SQL adatbázis segítségével kívánok teljesíteni.

# IRODALMI ÁTTEKINTÉS

Szakdolgozatom témája egy munkahét-menedzser telefonos alkalmazás elkészítése. Első körben a jelenleg alkalmazott technológiákat és munkahelyi körülményeket vizsgálom meg, majd rátérek arra, hogy miért is fontos egy ilyen alkalmazás elkészítése.

## A jelenlegi technológiák:

### A Lean és a Just in Time:

A megoldás a Toyota termelési rendszerének megértése és átültetése más környezetekre. Ez egyfajta keleti filozófia, amely magában foglalja az emberek, beleértve a saját dolgozók tiszteletét és egy olyan munkahelyi kultúra meglétét, amely a folyamatos újításra és megoldáskeresésre épül. Azt tűzi ki célul, hogy a jelenlegi feladatunkat végezzük el a lehető legjobban és eközben keressük meg a gyenge keresztmetszetet a szervezetünkben, és a gyártási folyamatainkban és apránként módosítva a dolgokat tegyük azt jobbá.

Ennek az egyik legfontosabb, hanem a legfontosabb része a percre kész-gyártás, azaz a Just in Time. A lényege az, hogy felesleges raktárkészletek megléte helyett az alkatrészeknek, alapanyagoknak pont a megfelelő időben kell odaérkezniük, azaz akkor, amikor rájuk szükség van. Ez azonban nem csupán a sima, kézzel fogható alapanyagokat jelenti, hanem magát a gyártáshoz szükséges információkat is.

A Toyota-filozófiájának alkalmazása a múlt század vége felé számos nagyvállalatnál elterjedt. Ezt ma Lean néven emlegetjük. A lean-filozófia lényege, hogy csak az tekinthető értéknek, amit a vevő annak tart, minden más veszteség. A vevő lehet külső és belső is. A belső vevő nem más, mint a gyártási sorrendben a következő munkafolyamaton dolgozó munkatárs. Amennyiben egy termék selejtes, vagy éppen késve érkezik oda, akkor az veszteséget jelent azon a ponton, és ha emiatt csúszik a kiszállítás a külső vevőnek, az az egész szervezet számára veszteséget jelent. Az egyes pontok veszteségei pedig összeadódnak.

A Leant jelenleg már számos nagyvállalat alkalmazza, de a XXI. század elején észrevették, hogy mindezen dolgok átvételének ellenére valami még sem működik megfelelően. Míg a Toyota továbbra is sikert sikerre halmozott, a többi autógyártó éppen, hogy csak vegetált. A helyzet a többi iparágban sem volt sokkal jobb.

Forrás: <https://hu.wikipedia.org/wiki/Just_In_Time>

Forrás: <https://hu.wikipedia.org/wiki/Lean>

### A Toyota-Kata

Mike Rother amerikai kutató tanulmányozni kezdte a Toyota működését annak érdekében, hogy rájöjjön a hiányzó összetevőre. Kutatásait a Toyota-Kata című könyvében írta le: a filozófia átvételén kívül szükség van a szervezeti kultúra teljes átalakítására is. Mi is az-a Toyota-Kata?

A Toyota-Kata az ismert japán autómárka nevéből és a Kata japán szóból áll, ami a harcművészetekből származik és „formagyakorlatot” jelent.

A Kata nem más, mint a harcművészetekben alkalmazott módszer, ami egy előre felépített mozgássorozat, amit nagyon sokszor ismételnek mindaddig, amíg már rutinná nem válik és egy adott harci szituációban gondolkodás nélkül reflex-szerűen és tökéletesen végre tudnak hajtani.

Az üzleti életben az eredeti jelentéséhez hasonló a szó jelentése. A Toyota Kata nem más, mint egy menedzsment módszer, amelynek célja a folyamatok ismétlődő gyakorlásával azok elvégzését rutinszerűvé tenni és az alkalmazottak elméjében úgy elmélyíteni a tevékenységeket, hogy álmukból felkeltve se tudják azt másképp csinálni, mint ahogy azt begyakoroltatták velük.

A TOYOTA-KATA folyamatának elve:

• A hosszútávú célokat lebontjuk közép és rövidtávú célokra.

• Lényegében van egy víziónk, aminek eléréséhez tervet készítünk, melyeknek a legapróbb alkotóelemei olyan elérendő célállapotok, amelyeket addig gyakoroljuk, amíg professzionálisan nem tudjuk azt bármikor elvégezni és ezt követően lépünk tovább egy következő célállapotra, így haladva a víziónk felé.

A Toyota-Kata három fő dologban nyújt segítséget:

• Hogyan érhetjük el, hogy a vállalatunk kezelni tudja a váratlan helyzeteket, a vevőink pedig hosszú távon is elégedettek legyenek

• Hogyan fejleszthetjük és aknázhatjuk ki munkatársaink képességeit, hogy jobb teljesítményre törekedjenek, és sikereket érjenek el

• Hogyan tehetjük cégünknél a folyamatos fejlődést és alkalmazkodást a mindennapi munka részévé

Összefoglalva tehát a Toyota-kata lényegében az agyunk edzése (1. ábra). Egy olyan gondolkodásmód kialakítása, amit folyamatainkban, (életünkben, vállalatunknál stb…) tudunk hasznosítani.

A logo of a brain lifting weights

Description automatically generated with low confidence  
1. ábra: Az agyunk edzése

Két alapvető típusra tudjuk felbontani:

• Fejlesztési-kata

• Kata-coaching

A Fejlesztési és a Coaching kata együttesen alkot egy egészet. A vezetők elsajátítják a coaching eszközöket, hogy a beosztottaknak tovább tudják adni a kata módszert.

A Fejlesztési-kata négy lépésből álló rendszeresen ismétlődő folyamat (2. ábra). Azaz ez a folyamat nem egyszeri alkalom. Újra és újra folyamatosan alkalmazni kell, azért, hogy beépüljön a szokásrendünkbe.

A picture containing text, bicycle

Description automatically generated2. ábra: A fejlesztési Kata lépései

1. Az irány vagy cél meghatározása azaz a VÍZIÓ. A vízió meghatározása a folyamat első lépése ugyan, de annak megvalósulása a végső cél és a folyamat legvégén érjük el.
2. A jelen állapot első kézből történő megismerése. A jelen állapotot azért kell első kézből megszerezni, mert ha az üzenet közvetítőkön keresztül – művezető, üzemvezető, gyártási igazgató – érkezik meg a cégvezetőhöz, akkor az már addigra torzulni fog. Ha már a jelen állapotot sem ismerjük fel rendesen, akkor a végcélig jutás elég bizonytalanná válik.
3. A következő célállapot meghatározása. Ez az az állapot, ami még elég apró változással jár ahhoz, hogy a felmerülő akadályt leküzdjük és ezt a lépést rutinossá tehessük.
4. Lépésről-lépésre haladunk tovább a kijelölt úton, miközben az akadályokat leküzdjük. Itt történik a 3. pontban írt rutinossá tétel. Az itt fellépő akadályokat, megértjük és kidolgozzuk az azok leküzdéséhez szükséges módszereket. Erre a PDCA módszer nagy segítséget jelenthet nekünk.

A PDCA módszer:

A Lean és a TOYOTA-KATA egyaránt használja. A PDCA-ciklus egy ismétlődő, négylépéses menedzsment módszer, amelyet a termékek és folyamatok kontrolljára és folyamatos fejlesztésére használnak. A PDCA-ciklus szorosan kapcsolódik a lean termelés megvalósításához (3. ábra).

A picture containing ball, graphics

Description automatically generatedA betűk jelentése:

P (plan) - tervezés

D (do) – cselekvés

C (check) – ellenőrzés

A (act) - beavatkozás

3. ábra: PDCA-ciklus

#### A Kata-Coaching:

A Kata coaching a fejlesztési kata oktatása. Az oktatás minden módszer elengedhetetlen eleme. A Toyota- kata esetében ez a következőképpen néz ki: mindenkinek van mentora, aki igazából coach, azaz edző szerepet tölt be. Nem válaszolja meg a mentorált kérdéseit, hanem csak támogatja őt abban, hogy az maga jöjjön rá a válaszokra.

A Coaching-Kata esetén az elvárások és a kommunikáció mindig két irányú. A vezető (mentor/coach) felelős a betanításért, a párbeszédekért, felelősséggel tartozik a munkájáért, egyszerre mindig csak egy lépést tesz meg, mint ahogy ugyanez igaz a mentoráltra is.

A célok elérése nem utasításként van kiadva, hanem együtt (a mentor és a mentorált) tartoznak érte felelősséggel, tehát közös munka, felelősség és a célok. Mivel mindenkinek van mentora, és egyúttal ő maga is mentorált, ezért kevesebb a munkahelyi nézeteltérés, hiszen mindenki ugyanazért a célért dolgozik.

A coaching kata célja az, hogy a fejlesztési katát a cégnél mindenki értse és magévá tegye. A mentor vezet, segít, tanít, támogat (4.-ábra).

A picture containing text, screenshot, diagram, line

Description automatically generated4. ábra: A Fejlesztési és a Coaching-Kata kapcsolata

### Miért fontos a megfelelő szervezés:

Egy termeléssel foglalkozó cég esetében nagyon fontos a megfelelő mennyiségű és minőségű információk megfelelő időben történő beérkezése a döntéshozók számára. Egy cégvezető, de akár egy középvezető esetében is nagyon fontos tudni, hogy éppen hol tart az alapanyag szállítása, hol tart egy adott megrendelés gyártása vagy éppen egy termelő berendezés nem hibásodott-e meg. Mindezen információk időben történő beérkezése rugalmasabbá teheti egy vállalat működését. A vezető hamarabb intézkedhet a gyártáshoz szükséges erőforrások átcsoportosításáról, a megrendelők minél gyorsabb tájékoztatásáról, de ugyanígy fontos az is, hogy egy egyszerű fizikai dolgozó is tisztában legyen a gyártási sorrenddel.

pl. Ha egy beszállító csúszik az alapanyag leszállításával, de egy másik megrendelés alapanyaga raktáron van, akkor a vezető gyorsan módosítani tudja a gyártási sorrendet. A fizikai dolgozók esetében pedig az időben beérkező információ hatására a dolgozónak sem kell duplán dolgoznia amiatt, mert át kell állítania a gyártó gépet egy másik termékre, vagy állnia kell feleslegesen a berendezés mellett, mert elfogyott az alapanyag és neki senki sem szólt.

A fejlesztési és a Coaching-Kata kapcsolatából is kitűnik ennek az egész termelési filozófiának a leglényegesebb összetevői a megfelelő szervezés és a döntéshozók (A Toyota-Kata esetén ők határozzák meg a Fejlesztési Katát) megfelelő és időbeli információkkal való ellátottsága ahogy a dolgozóké is. (A Toyota-Kata esetében ők egy személyben az edzők és a Tanulók is.)

A vállalati kultúra átalakítása nem képezi ennek a szakdolgozatnak a kutatási területét, az inkább Humán-Erőforrás menedzsmenti kutatási terület, de a vezetők és beosztottak munkájának megszervezése igen, hiszen egy jó termelés menedzselő szoftver segítségével a fentebb említett problémák könnyen megszüntethetőek. Éppen azért tartottam fontosnak kitérni e vállalati filozófiák ismertetésére, mert ezekből látható, hogy miért is fontos egy olcsó és jól használható mobilalkalmazás a termelő vállalatok vezetőinek menedzselésére. Ráadásul a Lean filozófia a szoftverfejlesztés területén már jelenleg is alkalmazásba került: Gondoljunk csak a Kanban táblára, amely szintén a Toyota termelési rendszeréből került átvételre (5. ábra).

A picture containing text, screenshot, rectangle, display

Description automatically generated  
5. ábra: A Kanban tábla

A mai okoseszközök segítségével (okostelefon, okosóra) minden érintett fél megfelelő időben kaphat pontos tájékoztatást a szükséges adatokról. Ezen kívül persze, ha minden flottul megy, akkor is segíthet egy ilyen termék, hiszen egy- esetleg több hét gyártását előre betervezve könnyebben ütemezhetőek a termelési folyamatok és elkerülhetőek vagy legalábbis csökkenthetőek a késedelmes szállításból, vagy a hullámzó gép- és emberi erőforráskihasználtságból adódó problémák.

Mindezen tényezőket figyelembe véve úgy gondolom, hogy egy ilyen munkahét-menedzselő telefonos applikáció sokat segíthet a fenti ideális, azaz problémamentes gyártásszervezés megvalósításában. A következő fejezetben bemutatom az én megoldásom megvalósítását erre a probléma-felvetésre.

Forrás: Mike Rother-Toyota-Kata - Hogyan ösztönözzük munkatársainkat fejlődésre, alkalmazkodásra és kiváló eredményekre? HVG Kiadó, 2014 ISBN: 9789633041864

## Jelenlegi megoldások

### A Jira alapjainak bemutatása

A Jira egy népszerű projektmenedzsment program, amelyet az Atlassian fejlesztett ki. Elsősorban problémakövetésre és agilis projektmenedzsmentre használják, és széles körben megjelenik a szoftverfejlesztésben, marketingben és számos más iparágakban is.

A Jira lehetővé teszi a felhasználók számára, hogy megoldandó feladatokat, problémákat és hibajegyeket hozzanak létre és nyomon kövessenek, hozzárendeljék azokat a csapattagokhoz, valamint prioritásokat és határidőket állítsanak be. Támogatja az olyan agilis projektmenedzsment módszertanokat is, mint a Scrum és a Kanban, illetve az olyan funkciókat, mint a sprint tervezés vagy burndown diagramok.

Fő funkciói közé tartoznak a testreszabható munkafolyamatok, a speciális keresés, szűrés, és a jelentések készítése, valamint az integráció más eszközökkel és szolgáltatásokkal, mint például a GitHub, a Bitbucket, és a Confluence.

Mivel ilyen sokszínű és rengeteg funkcióval bír, emiatt új felhasználóknak több időt vehet igénybe a használatának betanulása. Ezek a tényezők egy projekt komplexitásától is függhetnek, hiszen minél több plusz integrációs modult használunk annál könnyebb lehet a részletekben is elveszni (6. ábra).

forrás: <https://smartifysol.com/blog/testing/what-is-jira-advantages-and-disadvantages-of-jira-blog>

A screenshot of a computer

Description automatically generated with medium confidence

6. ábra Jira kép

Forrás: <https://community.atlassian.com/t5/Jira-questions/How-to-create-recurring-tasks-in-JIRA/qaq-p/669096>

### A Trello alapjainak bemutatása

A Trello egy projektmenedzsment program, amit a Jira komplexitásával ellentétben inkább a vizuális megoldások jellemeznek, mert az egyes feladatokat kártyák formájában reprezentálja a felhasználóknak, ezeket különböző listákba lehet rendezni, amelyek a projekt állapotait jelzik. A Trellot a Fog Creek Software készítette, majd 2017-ben felvásárolta az Atlassian. A két program közti legfőbb különbségek közé sorolnám, hogy a Trello inkább a kisebb, egyszerűbb projektekhez lehet alkalmasabb az alábbi tényezők miatt:

• Mind a kettő esetben ingyenes megoldás is rendelkezésre áll, de minél több funkcióra van szükségünk, annál drágábbak az elérhető csomagok is, de a Jira esetében az ár nagyobb mértékben emelkedik.

• Jelenleg a Trelloban kevesebb funkció és integritás érhető el, mint a JIRA-ban. Főbb hiányosságai, hogy nem személyre szabhatóak a munkafolyamatokat, nincs időkövetés és haladó jelentéskészítés (7. ábra).

Forrás: <https://trello.com/pricing>

[https://www.simonsezit.com/article/trello-vs-jira/](https://www.simonsezit.com/article/trello-vs-jira/%20)

Screens screenshot of a computer

Description automatically generated with medium confidence7. ábra Trello kép

Forrás: [https://trello.com/hu/tour](https://trello.com/hu/tour%20)

### Jira komplexitási problémák:

Elmondható a jelenlegi piacon használt projekt menedzsment programokról, hogy a projekt típusa, célkitűzése, illetve a cég nagysága alapján dőlhet el, hogy melyik szoftvert választjuk.

Jellemző gyakorlat, hogy sikeres kisebb projektek esetén a cégek már nem választanak új programot, inkább együtt élnek a szoftver gyengeségeivel, esetleg az új frissítések által kivárhatnak a problémáik megoldásaira. Ennek okai közé sorolhatóak a deadlineok, amik betartása főleg egy startup vállalkozás esetén kulcsfontosságú, mert egy nem teljesített projekt akár a vállalat végét is jelentheti.

Egy új munkahely esetén is érheti meglepetésként a munkavállalókat a Jira komplexitása, hiszen vannak vállalaton belüli egyedi speciális tevékenységek, ami viszont máshol kevésbé vagy egyáltalán nem volt jellemző. Különbség lehet például a hibajegyek kezelése. Azoknál a vállalatoknál, ahol minden feladatnál hibajegyet kell felvenniük a felhasználóknak, rengeteg adminisztrációval járhat, például a jelenlegi munkahelyem esetében a teljes ticket cycle folyamaton végig kell menni.

Ennek menete a következő lépésekből áll:

• új hibajegy

• hozzárendelés

• megnyitva

• másodpéldány

• eltérő

• nem hiba

• elutasítva

• újra megnyitva

• kijavítva

• újra letesztelve

• ellenőrzött és lezárva

Forrás: [https://zapier.com/blog/how-to-choose-project-management-software/](https://zapier.com/blog/how-to-choose-project-management-software/%20)

### Biztonsági kockázatok népszerű projektmenedzsment programok esetén:

A Jira és Trello esetében kijelenthető, hogy az új megoldások, biztonsági frissítések implementálása folyamatos, de ez elengedhetetlen egy sikeres projektmenedzsment program esetén, hiszen a népszerűségéből adódóan a csalók is megpróbálják kihasználni.

Biztonsági kockázatokat tekintve is két fontosabb kategóriába sorolnám az incidenseket:

1. Szoftver hiányosságaiból adódó támadás. Például ilyen lehet a szoftver mögött álló kód kihasználása. SQL adatbázisok esetén SQL injection, ahol bemeneti paraméternek tábla módosító értéket írunk be, ezzel hibás működést érhetünk el.

Például: SELECT \* FROM felhasznalok; DROP TABLE fizetes

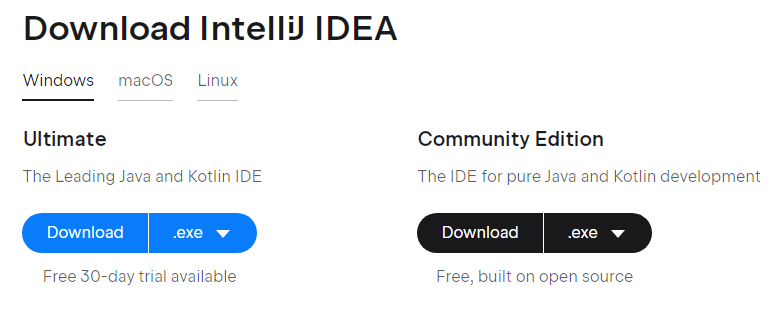
Forrás: [https://www.w3schools.com/sql/sql\_injection.asp](https://www.w3schools.com/sql/sql_injection.asp%20)

1. Egy meglévő felhasználó adataival visszaélés elkövetése: Ezekben az esetekben a felhasználó elveszti az irányitást a fiókja felett és a csaló a megszerzett adatokkal belépve kárt tesz, esetleg zsarolást követel el.

Forrás: <https://crosssec.com/mi-az-a-szemelyes-adatokkal-valo-visszaeles/>

# A FEJLESZTÉS MENETE

## Eszközök telepítése

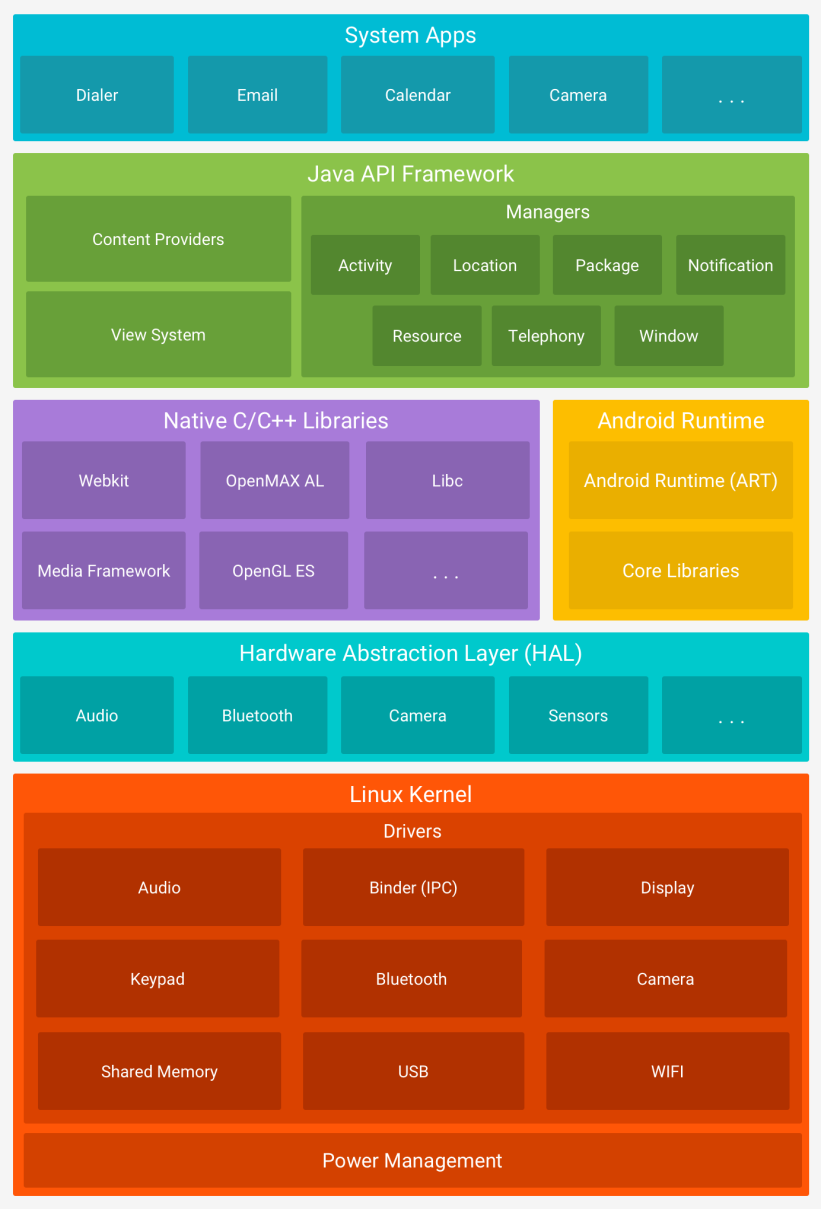
Az IntelliJ IDEA ingyenesen letölthető a <https://www.jetbrains.com/idea/download/> webhelyről a 8. ábra szerint (a Community verzió ingyenes, de létezik fizetős változata is Ultimate néven és létezik egy oktatói kiegészített Community verzió is). 

8. ábra a letöltési képernyőkép

Forrás: <https://www.jetbrains.com/idea/download/>

## Android operációs rendszer

Az Android egy nyílt forráskódú operációs rendszer, mely elsősorban érintőképernyős eszközökre van optimalizálva. Linux kernelt használ, a platform felépítése a 9. ábra szerint történik.



9. ábra platform felépítése

## IntelliJ IDEA

### Plugin

A beépülő modulok kiterjesztik az IntelliJ IDEA alapvető funkcióit.

Például:

* integráció verziókezelő rendszerekkel,
* problémakövetőkkel,
* build felügyeleti szerverekkel és egyéb eszközökkel,
* kódolási segítségnyújtás különböző nyelvekhez és keretrendszerekhez,
* kódolási gyakorlatok, amelyek segíthetnek egy új programozási nyelv elsajátításában.

A Kotlin eléréséhez telepíteni kell a Kotlin plugint az IntelliJ IDEA programon belül.

A File Menü Settings almenüjében a Plugins kiválasztása után meg kell keresnünk a Kotlin plugint, mely telepítése szükséges.

### SDK

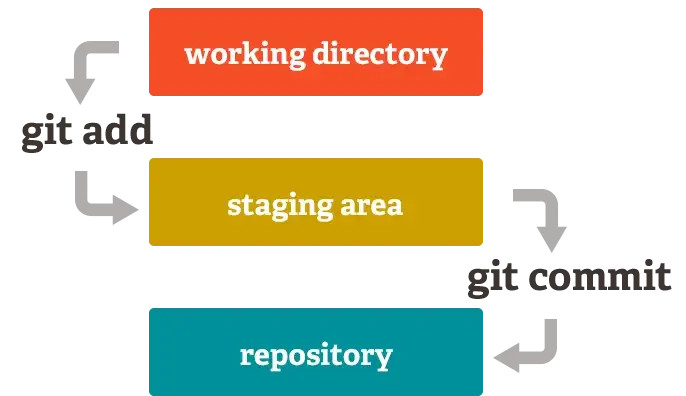
Az SDK letölthető külön a Google szerveréről vagy az IntelliJ IDEA programon belül is.  
A File Menü Settings almenüjében a Appearance & behavior kiválasztása után a System settings almenü Android SDK almenüjében választhatjuk ki a megfelelő SDK-t vagy, ha már telepítésre került akkor itt adhatjuk meg a telepített SDK elérési útját is.

## Gradle

A Gradle egy nyílt forráskódú építési automatizálási eszköz. Build szkriptek határozzák meg a projekt feladatait, függőségeit, beépülő moduljait és egyéb konfigurációit. Egyetlen build tartalmazhat egy vagy több projektet, és minden projekt tartalmazhat saját alprojekteket, melyeknél a munka végrehajtási logikája és tesztek futtatása is meghatározásra kerül. A maventől eltérően nem xml-t, hanem domain specifikus nyelvet használ.

## Git

A Git (10. ábra Git) egy nyílt forráskódú, szoftverforráskód-kezelő rendszer verziókezeléssel. A Git repository teljes verziótörténettel és teljes revíziókövetési lehetőséggel rendelkezik.



10. ábra Git

## Tesztkörnyezet

### Virtuális

IntelliJ IDEA van lehetőség virtuális eszköz futtatására, de ez meglehetősen erőforrás igényes.

Van lehetőség más virtuális eszközön is futtatni az alkalmazást, de ekkor apk-t kell létrehozni és ezt telepíteni. Ilyen eszköz például a BlueStacks, mely tapasztalataim alapján hasonlóan erőforrás igényes, mint az integrált változat.

### Valós eszköz használata

Van lehetőség közvetlenül használni az eszközünket is, mintha virtuális lenne IntelliJ IDEA programon belül adatkábel segítségével fejlesztői módban.

Az elkészült apk fájlt a telefonra másolva elindítani azt és feltelepíteni tesztelés céljából. A fentebb felsorolt módok közül ez a legkevésbé erőforrás igényes lehetőség, de így is létre kell hozni minden alkalommal az új apk fájlt.

## Program fejlesztése

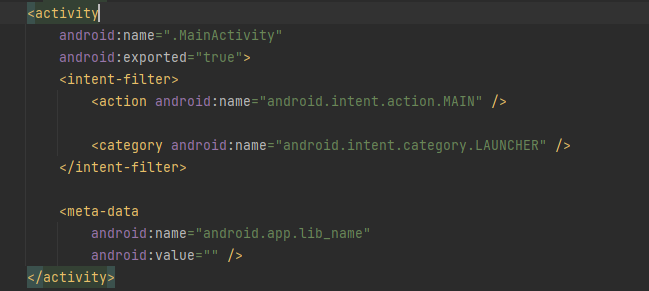
### AndroidManifest.xml

Minden alkalmazásprojektnek rendelkeznie kell egy AndroidManifest.xml fájllal ezen a néven, a projekt forráskészletének gyökerében. Ezen fájl alapvető információkat tartalmaz az alkalmazásról az Android build eszközökről, az Android operációs rendszer számára. Tartalmaznia kell ezen fájlnak az alkalmazás összetevőit, amelyek magukban foglalják az összes tevékenységet, szolgáltatást, adás vevőket és tartalomszolgáltatókat. Minden összetevőnek meg kell határoznia az alapvető tulajdonságokat, például a Kotlin vagy a Java osztály nevét.

Deklarálhat olyan képességeket is, hogy mely eszközkonfigurációkat képes kezelni, valamint szándékszűrőket, amelyek leírják, hogyan indítható el az összetevő, továbbá azokat az engedélyeket, amelyekre az alkalmazásnak szüksége van a rendszer védett részeihez vagy más alkalmazásokhoz való hozzáféréshez emellett deklarál minden olyan engedélyt, amellyel más alkalmazásoknak rendelkezniük kell, ha hozzá akarnak férni az alkalmazás tartalmához. Az alkalmazáshoz szükséges hardver- és szoftverfunkciók, amelyek befolyásolják, hogy mely eszközök telepíthetik az alkalmazást.   
Forrás: <https://developer.android.com/guide/topics/manifest/manifest-intro>

#### Alkalmazás indítása

Itt történik a program indításának meghatározása is, ami ebben az esetben a ”.MainActivity” (11. ábra AndroidManifest.xml Alkalmazás indítása).

****

11. ábra AndroidManifest.xml Alkalmazás indítása

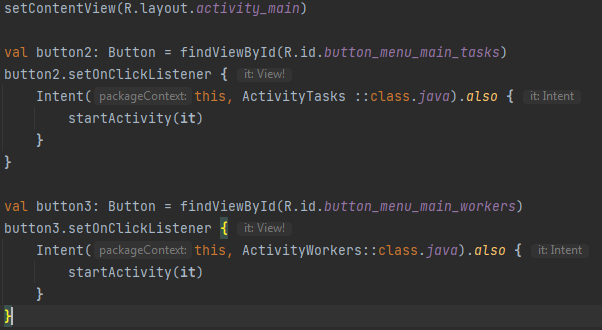
### Activity

Activity tehát a tevékenységek. Az Activity osztály az Android-alkalmazások kulcsfontosságú összetevője, és a tevékenységek elindításának és összeállításának módja a platform alkalmazásmodelljének alapvető része. A mobilalkalmazás-élmény abban különbözik az asztali megfelelőjétől, hogy a felhasználó interakciója az alkalmazással nem mindig ugyanazon a helyen kezdődik, például megnyit egy e-mail alkalmazást a kezdőképernyőről, megjelenhet az e-mailek listája.

Ezzel szemben, ha olyan közösségi médiaalkalmazást használ, amely ezután elindítja az e-mail alkalmazást, akkor közvetlenül az e-mail alkalmazás képernyőjére léphet e-mail írásához. Az Activity célja ennek a paradigmának az elősegítése. Amikor az egyik alkalmazás meghív egy másikat, a hívó alkalmazás egy tevékenységet hív meg a másik alkalmazásban, nem pedig az alkalmazás egészét. Ily módon a tevékenység belépési pontként szolgál egy alkalmazásnak a felhasználóval való interakciójához. Egy tevékenységet az Activity osztály alosztályaként valósít meg. A legtöbb alkalmazás több képernyőt tartalmaz, ami azt jelenti, hogy több tevékenységet tartalmaznak. Az Activity alkalmazásában való használatához regisztrálnia kell a róluk szóló információkat az alkalmazás jegyzékében, és megfelelően kell kezelnie az Activity életciklusait.

#### MainActivity class

A tényleges alkalmazás belépési pontja (ahol már az általam fejlesztett szoftver programkódja fut). Itt a hozzá tartozó layout felület meghatározásra kerül (12. ábra layout. menü vezérlő).

****

12. ábra layout. menü vezérlő

A menün lévő gombra nyomva a setOnClickListener elindul (a nyomógombhoz rendelt clikkelésre vagy ebben az esetben megnyomásra elinduló funkció), és a kód tovább irányítja a felhasználót az ActivityWorkers vagy a másiknál az ActivityTasks Class –ra melyek ugyancsak Activity-k. Létezik egy harmadik gomb is, amely az üdvözlő képernyőre mutat, de mivel már azon vagyunk erre nem került beállításra funkció. Ezen menü és a hozzá tartozó vezérlő megtalálható a többi Activity-n is (ezeken már aktív a MainActivity-re mutató is).

#### ActivityWorkers class

Itt a hozzá tartozó layout felület meghatározásra kerül és a menün ugyancsak megtalálható a setOnClickListener. Továbbá ezen a lapon létezik még egy „Recycler View” nevű felület is. A „Recycler View” megkönnyíti a nagy adathalmazok megjelenítését és kezelését, emellett dinamikus megjelenést biztosít, így könnyen görgethető és végezhető interakció az egyes elemekkel.

val helper = DBHelperWorkers(applicationContext)

val myDataset = DataSource().loadWorkers(helper)

Az applicationContext-et tovább kell küldeni, mert szüksége lesz az SQLite-nak rá, és a DataSource egy listával fog visszatérni, melyben a megjelenítendő adatok vannak.

val rv : RecyclerView = findViewById(R.id.recycler\_view\_workers)

A „Recycler View”-t határozza meg, amelybe az adatokat kívánjuk megjeleníteni (rv néven hivatkozunk rá).

rv.adapter = WorkersItemAdapter(this, myDataset)

Tovább küldjük az adapternek az adatokat (az adapter lesz a felelős a megjelenítésért és az elemek viselkedésért).

#### ActivityTasks class

Itt a hozzá tartozó layout felület meghatározásra kerül és a menün ugyancsak megtalálható a setOnClickListener. Továbbá ezen a lapon létezik még egy másik az ActivityWorkers-nél használthoz nagyon hasonló „Recycler View” felület is, de ez más adatokkal másik SQL táblából dolgozik.

#### Settings class

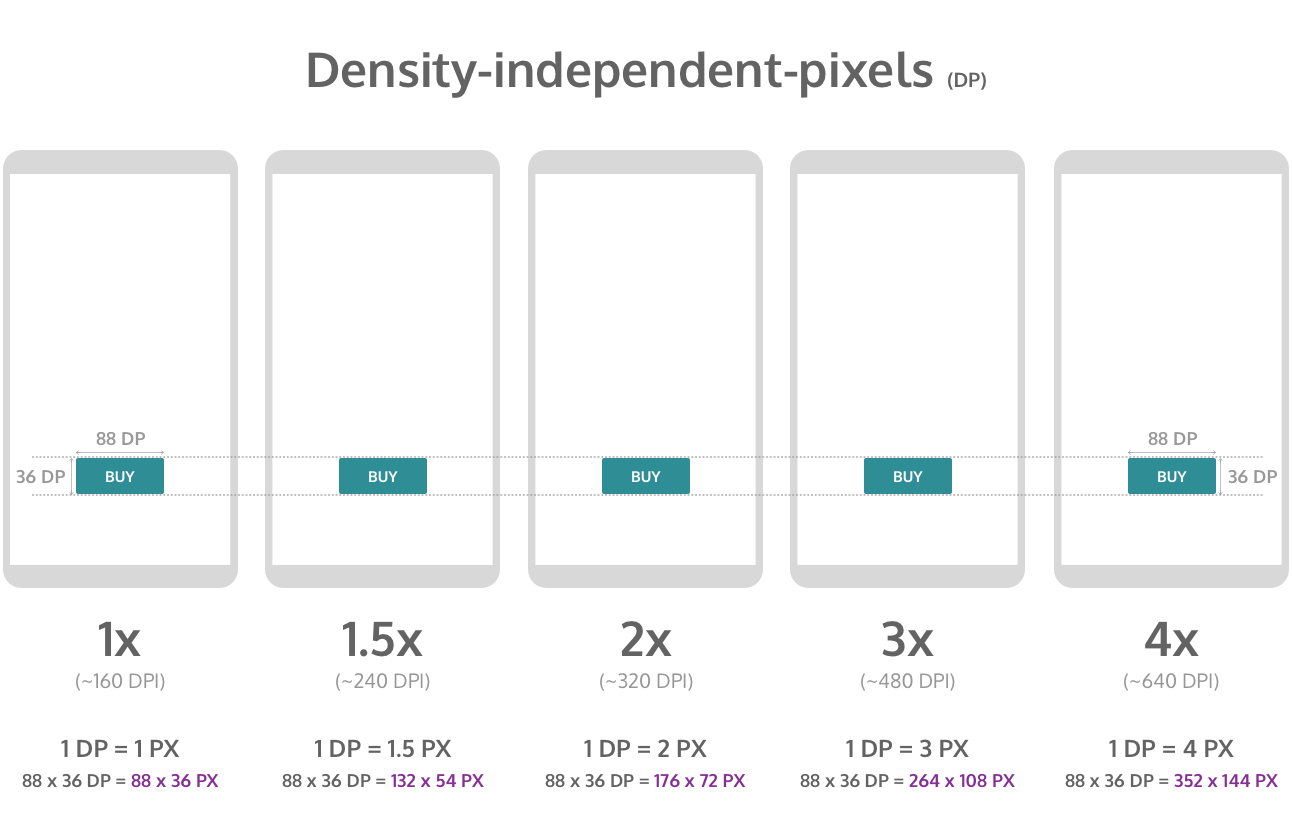
Itt is a hozzá tartozó layout felület meghatározásra kerül és a menün ugyancsak megtalálható a setOnClickListener. Itt kerül meghatározásra a szerver címe és a hozzá tartozó jelszó. Az adminisztrációs belépés csak akkor látható, ha rendelkezik valaki adminisztrációs jogosultsággal, itt ki is lehet léptetni, de minden indításnál automatikusan kiléptet a program az adminisztrációs módból. Továbbá itt lehet még beállítani a saját felhasználó nevet, email címet, de ezeket egy magasabb beosztású felhasználónak jóvá kell hagynia, és a saját jelszavunkat, aminek eszközváltásnál van jelentősége.

### Felhasználói felület

A DPI (dots per inch, pont per hüvelyk) az eszközök kijelzőjének és elrendezésének (álló, fekvő), kezelésére került bevezetésre, mint fogalom.

<https://developer.android.com/training/multiscreen/screendensities>

Ha különböző méretűek a kijelzők és a felbontásuk is különböző a képarányuk mellett akkor is azonos elrendezési elvekre és olvashatóságra törekszünk. A DP (density-independent pixels, sűrűségtől független pixelek) alkalmazásával kell megtervezni a felhasználói felületet, így állandó méretűek maradhatnak az alkalmazás grafikus részei (13. ábrán látható módon).



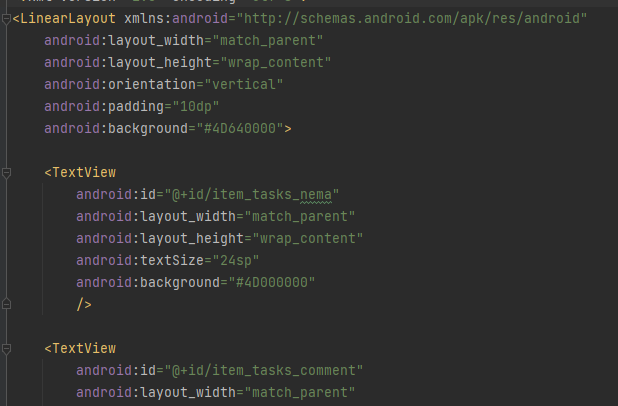
13. ábra dp különböző felbontások mellett

#### Layout xml

A Layout xml lehetővé teszi olyan kifejezések írását, amelyek kezelik a nézetek által elküldött eseményeket. A Data Binding Library automatikusan létrehozza azokat az osztályokat, amelyek szükségesek ahhoz, hogy az elrendezésben lévő nézetek az adatobjektumokkal összekapcsolódjanak.

#### Layout alkalmazása

Az activity\_main.xml az üdvözlő képernyő megjelenítésért felelős lap.

Az item\_list\_task.xml a „recycle view” elemeit határozza meg (14. ábra list\_item\_tesks).   


14. ábra list\_item\_tesks

### További fájlok

#### Model mappa

Itt kerül meghatározásra a DataSource-ban létrehozandó listák elemei, és ezen elemek típusa, létrehozásra kerül külön egy a Tasks és egy a Workers számára.

#### Data mappa

Ide kerül a DataSource és a két SQL táblakezelő fájl.

A DataSource felelős a listák létrehozásáért és feltölti a SQL táblákból származó elemekkel a listákat. Ezen listákat az ActivityWorkers és az ActivityTasks kérte el korábban.

DBHelperTasks és a DBHelperWorkers hozza létre a táblákat (ezeknek kell átadnia a korábban említett applicationContext-et a jogosultságok miatt).

### SQL-ről alapjaiban

Az SQL (Structured Query Language) egy szabványosított programozási nyelv, amely az adatbázisok kezelésére szolgál. Az SQL-t főként arra tervezték, hogy lehetővé tegye az adatbáziskezelő rendszerek használatát, amelyet az alábbi kategóriákba sorolhatjuk:

* Adatdefiníciós nyelv (Data Definition Language – DDL)

adatbázisok sémájának definiálására használják

jellemző utasításai: CREATE, ALTER, DROP

* Adatlekérdező nyelv (Data Query Language – DQL)

adatok lekérdezéséhez, lehívásához használják

jellemző utasításai: SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY

* Adatmanipulációs nyelv (Data Manipulation Language – DML)

adatok létrehozására, módosítására, törlésére használják

jellemző utasításai: INSERT, UPDATE, DELETE

* Adatelérést vezérlő nyelv (Data Control Language – DCL)

jogosultságok adása

jellemző utasítások: GRANT, REVOKE

* Tranzakció vezérlő nyelv (Transaction Control Language -TCL)

Feladatok csoportosítása egyetlen végrehajtási egységbe.

Jellemző utasításai: BEGIN, COMMIT, ROLLBACK, SAVEPOINT

Forrás: [https://webiskola.hu/sql-ismeretek/relacios-adatbazis-sql-fogalmak-peldak/](https://webiskola.hu/sql-ismeretek/relacios-adatbazis-sql-fogalmak-peldak/%20)

### SQLite-ról alapjaiban

Az SQLite nyílt forrás kódú, elsősorban azoknak a fejlesztőknek készült, akik adatbázist szeretnének használni strukturált adatok kezelésére olyan alkalmazásokban, mint az asztali, webes és mobilalkalmazások (15.ábra).

A többi adatbáziskezelő programokhoz képest nagy előnye, hogy könnyen telepíthető és gyorsan beállítható, nem feltétlen szükséges hozzá szerver beállítása sem. Nyelvezete nagyon hasonló a többi adatbáziskezelő nyelvekhez, így más programba való áthelyezése esetén a kódot nem, vagy csak minimálisan szükséges módosítani. Az SQLite könnyen integrálható más programokba és alkalmazásokba, mivel az adatbázis-kezelő rendszer egy egyszerű programkönyvtár formájában használható.

Ami az előnye az a hátránya is az SQLite-nak, mert egyszerűsége miatt korlátozottan elérhetőek funkciók, például az ALTER TABLE és az ALTER COLUMN, ezért az adatbázisok sémájának definiálása nehézkesebb lehet. Nagyobb méretű adatbázisok esetén lassú a működése, mivel kisebb memóriafelhasználással rendelkezik, mint más adatbáziskezelő rendszerek, emiatt nem alkalmas nagy forgalmú webhelyekhez, ahol több szerveren szükséges az adatok tárolása.

forrás: [https://www.sqlitetutorial.net/](https://www.sqlitetutorial.net/%20)

15. ábra

Forrása: [https://www.linkedin.com/pulse/why-sqlite-becomes-popular-embeded-database-sameera-gunathilaka](https://www.linkedin.com/pulse/why-sqlite-becomes-popular-embeded-database-sameera-gunathilaka%20)

#### SQLite syntax egyediségek:

Az SQLite egyszerűvé teszi egy objektum létrehozását anélkül, hogy ellenőrizné annak a létezését az „if exists” és a „if not exists” parancsokkal. Sajnos a TOP parancs nem elérhető, viszont a LIMIT igen, amivel megadhatjuk, hogy maximum hány sort szeretnénk visszakapni az adott lekérdezésnél. LIMIT nevű parancsot csak a teljes lekérdezés végén használhatjuk, míg TOP esetében allekérdezést is használhatunk. Két tábla összekapcsolásához használhatjuk természetesen az alapvető JOIN parancsokat, azonban funkcionális hiányosságai miatt nem támogatott a RIGHT JOIN és a FULL OUTER JOIN parancsok.

Forrás: [https://www.mssqltips.com/sqlservertip/4777/comparing-some-differences-of-sql-server-to-sqlite/](https://www.mssqltips.com/sqlservertip/4777/comparing-some-differences-of-sql-server-to-sqlite/%20)

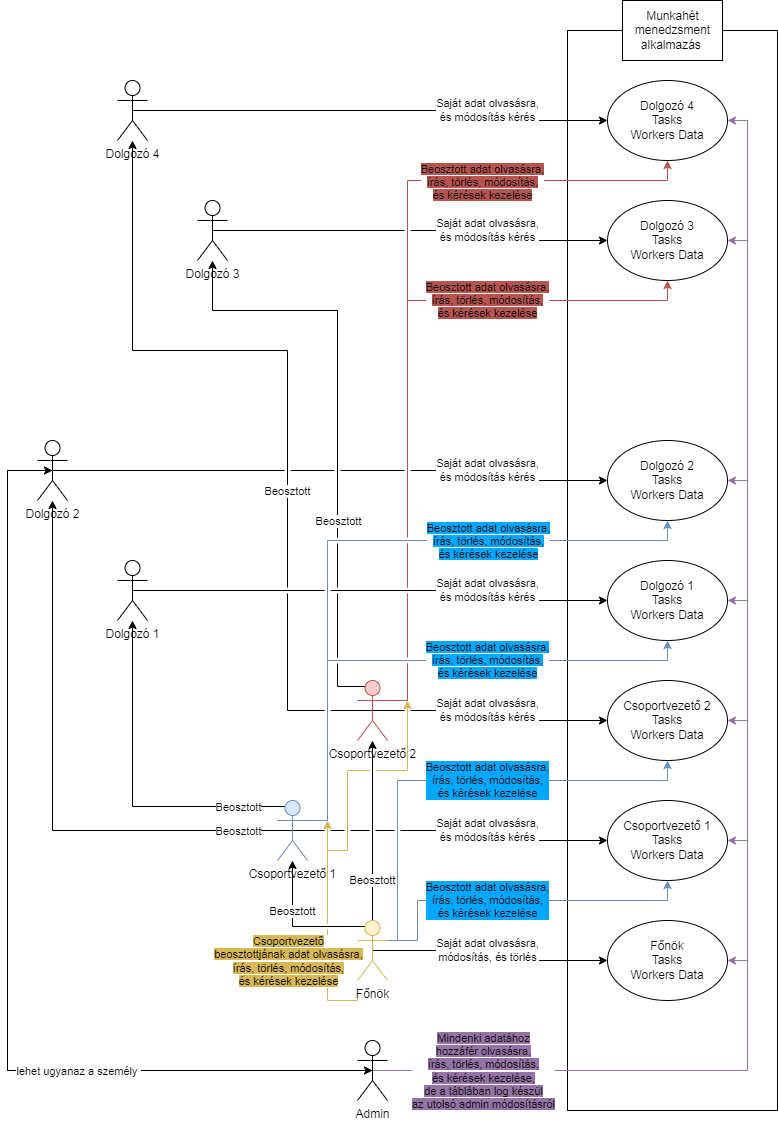
### További tervezett funkciók és működés

#### Üdvözlő képernyő

A szerepe a későbbiekben megváltozik, mert indításkor automatikusan megpróbál a szerverrel szinkronizálni működő internetkapcsolat esetén, és kézzel is lehet majd itt indítani szinkronizálást, továbbá egy súgó is elhelyezésre kerül itt, ami bemutatja a program funkcióit.

#### Jogosultságok

A jogosultságok kiosztása hierarchikus rendszerben történik, fás elrendezésben ahogy az ábrán látható (8. ábra Jogosultságok use-case diagramja).



. ábra Jogosultságok use-case diagramja

Az adott személy a vele egy szinten álló többi személy adatait nem láthatja és a sajátját nem módosíthatja közvetlenül. Az általa szerkesztett módosítást a program megmutatja a felette álló személyeknek és ezt valaki jóváhagyhatja (a befejezést is). A program nem menti a telefonra a szerverről származó nem jogosult adatokat.

Egy olyan felhasználó, aki alá tartoznak dolgozók, az csak a közvetlenül alá tartozó dolgozókat láthatja, és szerkesztheti az adataikat szabadon. A főnök mindenki adatát szabadon szerkesztheti és oszthat ki admin jogosultságokat.

Az admin lehet bármilyen szinten álló felhasználó, akár több is, aki kapott a főnöktől jogosultságot erre (a főnök magának is adhat), és ha az admin felületet aktiválja, akkor átlép az üdvözlő képernyőre a program és lekérdezi az SQL táblát a szerverről, és azt az új jogosultságok szerint kezeli.

Az adminnak (és csak neki) joga van még átírni más felhasználók jelszavát is. Az aktív adminnak joga van visszalépni passzív módba, vagy a programot újra megnyitva automatikusan passzívként indul.

#### Szerver SQL

A szerveren az összes adat egyetlen táblában tárolódik, és az alkalmazás válogatja szét minden indításnál a lekérdezett adatokat és csak azokat menti, amihez rendelkezik jogosultsággal a felhasználó.

# ELEMZÉS

A szakdolgozat megírásának pillanatában munkahelyemen rendszeresen JAVA programozási nyelvet használok, illetve más projekteknél is használtam. Kotlin programozási nyelvre esett a választásom, mert rengeteg hasonlóság szerepel benne általam ismert és alkalmazott nyelvekkel. Nehézséget okozott ,hogy a Kotlin-t kevesebben használják, így kevesebb anyagot és probléma kezelési megoldást találtam róla az interneten. A nyelv gyors változása miatt egy néhány évvel ezelőtt megírt Kotlin kód a jelenlegi frissebb verziókban mára már elavultnak számít, alkalmazása nehézkes, vagy lehetetlen, így rengeteg idő ment el a nyelv felfedezése során arra, hogy megoldásokat találjak apróbb problémákra. Használata rendkívül nagy erőforrás igénnyel rendelkezik, ezért tesztelése a vártnál több időt vehet igénybe, azoknál a személyeknél, akik nem rendelkeznek, az informatikában folyamatosan fejlődő és egyre erősebb hardveres komponensekkel.

# KONKLÚZIÓ

## Végeredmény

A felvetett problémát a program teljesíti. Átlátható a felülete és a kezelése, pont azokat a funkciókat tartalmazza, amelyekre szükség van egy ilyen alkalmazásban.

## Megoldások

Véleményem szerint a Kotlinnak égető szüksége lenne egy átfogó és folyamatosan frissített oktató anyagra az interneten, mert amelyeket a fejlesztés során találtam az vagy nagyon alap dolgokkal foglalkozik csak (https://www.w3schools.com/), vagy elvétve tartalmaz példa programokat (hivatalos dokumentáció), esetleg már elavult és a benne lévő kódok már nem használhatók.

## Kotlin értékelése

Sajnos a program megírása után nem vagyok teljes mértékben megelégedve a Kotlin nyelvvel. Nehézkes benne a fejlesztés, és túlságosan gyorsan változik (ami nem lenne baj, ha ez nem járna a régi megoldások kivezetésével is). Emellett azt is el kell ismernem, hogy alkalmas ettől nagyobb projektek megvalósítására is, és egy fejlesztő csapat létrehozhat összetettebb alkalmazásokat is.

## Lehetséges hasonló szakdolgozatok

Véleményem szerint nagyon sok elem van, ami teljesen megújítaná a projektet. például React vagy Xamarin. Emellett a projekt management rendszer egy más oldalról megközelítve teljesen megújíthatná az egész feladatot hiszen más célhoz más úton is juthatunk el.

# IRODALOM JEGYZÉK

A jelenlegi technológiák:  
<https://hu.wikipedia.org/wiki/Just_In_Time>  
<https://hu.wikipedia.org/wiki/Lean>

Toyota Kata:  
Mike Rother-Toyota-Kata - Hogyan ösztönözzük munkatársainkat fejlődésre, alkalmazkodásra és kiváló eredményekre? HVG Kiadó, 2014 ISBN: 9789633041864

Jira:  
<https://smartifysol.com/blog/testing/what-is-jira-advantages-and-disadvantages-of-jira-blog>  
<https://community.atlassian.com/t5/Jira-questions/How-to-create-recurring-tasks-in-JIRA/qaq-p/669096>

Trello:

<https://trello.com/pricing>

[https://www.simonsezit.com/article/trello-vs-jira/](https://www.simonsezit.com/article/trello-vs-jira/%20)

[https://trello.com/hu/tour](https://trello.com/hu/tour%20)

project-management-software:  
[https://zapier.com/blog/how-to-choose-project-management-software/](https://zapier.com/blog/how-to-choose-project-management-software/%20)

Sql:  
<https://www.w3schools.com/sql/sql_injection.asp>  
<https://crosssec.com/mi-az-a-szemelyes-adatokkal-valo-visszaeles/>  
[https://www.sqlitetutorial.net/](https://www.sqlitetutorial.net/%20)   
<https://www.linkedin.com/pulse/why-sqlite-becomes-popular-embeded-database-sameera-gunathilaka>   
<https://www.mssqltips.com/sqlservertip/4777/comparing-some-differences-of-sql-server-to-sqlite/>

Android fejlesztésének filozófiája és célja (2010. 05. 23.-ai változat)  
[https://web.archive.org/web/20100523002727/http://source.android.com/about/philosophy.html](https://web.archive.org/web/20100523002727/http:/source.android.com/about/philosophy.html)

Android.com hivatalos fejlesztői platformja.  
<https://developer.android.com>

Android.com hivatalos fejlesztői útmutatók.  
<https://developer.android.com/guide>

<https://developer.android.com/training/multiscreen/screendensities>

<https://developer.android.com/guide/topics/manifest/manifest-intro>

A Gradle egy nyílt forráskódú építési automatizálási eszköz.  
<https://docs.gradle.org/current/userguide/what_is_gradle.html>

JetBrins IntelliJ hivatalos útmutató.  
<https://intellij-support.jetbrains.com/hc/en-us>

JetBrins IntelliJ hivatalos dokumentációja.  
<https://www.jetbrains.com/help/idea/getting-started.html>

Google Android hivatalos oktatóanyaga.  
<https://developers.google.com/certification/associate-android-developer>