SZEGEDI TUDOMÁNYEGYETEM
Természettudományi és Informatikai Kar
Szoftverfejlesztési Tanszék

Szakdolgozat

Multiplatform Mobilalkalmazás Fejlesztése Közösségi Árkövetést Támogató Adatbázissal

Development of a Multiplatform Mobile Application with Community Price Tracking Database

Kovács Alex
programtervező informatikus

Témavezető: Kiss-Vetráb Mercedes, beosztása

Szeged

2023

A dolgozat formai követelményei

Ajánlott oldalszám:

BSc szakdolgozat: 20-25 oldal,

MSc diplomamunka: 35-50 oldal

Tanárszakos szakdolgozat: 50000-80000 karakter szóközökkel (lásd. TKK szabályzat)

Másfeles sorköz, sorkizárt, 12-es betű méret (általában Times New Roman), margók: jobb, bal, lent és fent 2,5 cm

Kötelező elemei a dolgozatnak

Címlap

Tartalmi összefoglaló

Tartalomjegyzék

Érdemi rész

(Szakterület specifikus fejezeteket tartalmaz, kérjük, konzultáljon a témavezetővel, hogy a példában közzétett kísérletes tudományterületeken használt fejezetekből az Ön dolgozatában melyikre van szükség!)

Irodalomjegyzék

Nyilatkozat

Tartalmi összefoglaló

(a fejezet új oldalon kezdve)

Munkámmal egy olyan alkalmazást sikerült létrehoznom, ahol a felhasználók könnyen és egyszerűen tudnak termékeket hozzáadni az adatbázishoz. A hozzáadott termékeket be tudják tenni a kedvencek közé, így a kedvencek oldalról könnyebben elérhetik a gyakran látogatott termékeiket. A termék oldalon egy üzlethez tudnak új árat feltölteni és a korábban feltöltötteket is el tudják érni, ahol dátum szerint vannak rendezve, ez azért jó, mert így könnyen nyomon követhetőek a különböző tisztítószerek árainak a változásai. Egy termék oldalára navigálva egyből egy listát látunk, ahol az üzletek nevei ár szerint növekvő sorrendbe vannak rendezve, vagyis a legolcsóbb üzlet van legfelül. Itt nem látszik az összes ár, hanem a különböző áruházakhoz feltöltött áraknak a mediánját láthatjuk. A termékeket hozzáadhatjuk a bevásárlólistához is. Egy felhasználónak csak egy bevásárlólistája lehet, ehhez a listához tud hozzáadni, vagy törölni termékeket. A kedvencek oldalon a felhasználó által hozzáadott kedvenc termékeket láthatjuk. Ezen az oldalon még hozzáadhatunk kedvenc üzleteket is, ami majd a bevásárlólistánál lesz fontos. Az üzletek neveire rá tudunk keresni, majd a kedvencekhez tudjuk hozzáadni. A főoldalról tudunk navigálni a bevásárlólista képernyőre, ahol először is megtekinthetjük a hozzáadott termékeket, majd ez a lista alatt különböző végösszegek láthatóak. Először a kedvencekbe tett üzletek összegeit tekinthetjük meg. Ez azért jó, mert ha van 2-3 vagy akár több üzlet, ahova szeretünk járni, akkor meg tudhatjuk, hogy hol éri meg a következő vásárlásunkat megejteni. A végösszeget csak akkor jeleníti meg, ha az összes bevásárlólistához hozzáadott termék elérhető az adott üzletben. A kedvenc üzletek alatt megtalálható egy olyan üzlet is, ahol az összes üzlet közül a legolcsóbban lehet kijönni a végösszeggel. A főoldalon még megtekinthetünk egy 9 elemből álló horizontálisan görgethető listát, ami a 9 legtöbbet megtekintett terméket listázza ki. Akárhányszor a felhasználó megtekint egy adatbázis elemet, akkor 1-gyel növeljük az értékét. Az applikációban még elérhető egy ajánlatok oldal is, ahol a felhasználók különböző ajánlatokat tekinthetnek meg. Egy ajánlat többféle terméket tartalmazhat, amelyeket könnyen meg lehet tekinteni. Ez az oldal egy leírást tartalmaz, ahol a látogatók többet tudhatnak meg a kedvezményes termékekről. Ugyanakkor az üzletnek a neve is látható, hogy legyen arról információnk, hogy melyik üzletben találhatóak meg az akciós termékek. A beállítások oldalon találunk egy profil fület, ahol meg tudjuk változtatni a keresztnevünket. A beállítások oldalon még ki tudunk jelentkezni a felhasználónkból.

A dolgozat tartalmának rövid (max. 1 oldal) összefoglalása. A következő részekből áll: rövid irodalmi összefoglaló, a dolgozat elkészítéséhez használt módszerek, eredmények, konklúzió

Kulcsszavak: (a dolgozat tartalmára specifikusan jellemző 4-6 szó, egymástól vesszővel elválasztva) flutter, dart, adatbázis, árkövetés, mobilalkalmazás, multiplatform,

Tartalomjegyzék

(új oldalon kezdve)

Bevezetés

(a fejezet új oldalon kezdve)

Jelen dolgozatom témáját az utóbbi időben megnövekedett infláció adta, amelynek folyamán egy olyan alkalmazást készítettem el, ahol a különböző termékek árait a felhasználók fel tudják tölteni és nyomon követni. Mostanában mindenki nap-mint-nap megtapasztalja az üzletekben kaphatok termékek árának változását, ugyanakkor két üzlet között hatalmas lehet a különbség. Innen jött a szakdolgozatom ötlete, hogy praktikus lehetne egy olyan applikációt megtervezni és létrehozni, ahol a különböző termékek árait a felhasználók fel tudják tölteni és nyomon követni. Én leginkább a közösségi árkövetésre koncentráltam, ahol a felhasználók tudnak új termékeket, áruházakat hozzáadni, majd az adott termékekhez a legfrissebb árakat feltölteni. Ezt a logikát szinte mindenféle terméken lehetne alkalmazni, én most viszont csak a takarítószerekre szűkítettem le az applikációmat. Ugyanakkor szerintem nagyon hasznos lehet élelmiszerek, vagy akár szórakoztatási termékek árának követésére is. A felhasználók különböző tisztítószereket és tisztítással kapcsolatos termékeket tudnak feltölteni. A választásom a Flutterre, mint egy nyílt forráskódú szoftverfejlesztő készletre és a Dart programozási nyelvre esett, mert ennek a két eszköznek a segítségével multiplatformos mobilalkalmazásokat lehet készíteni. Ez azért nagyon jó, mert csak egy kódbázist kell fenntartani. Nem kell külön natív applikációkat fejleszteni. Ez egyben idő és költséghatékony is. Mindemellett gyors és esztétikus applikációkat tudunk vele fejleszteni. Azért választottam a mobilalkalmazást a szakdolgozatom témájának, mert mostanában szinte majdnem mindenkinek van okostelefonja, ezért a mobilapplikációk használata nagy népszerűségnek örvend. A Flutter szoftverfejlesztő készletet kimondottan jó választásnak tartom, mert a segítségével gyorsan lehet alkalmazásokat fejleszteni. Egyetemi tanulmányaim során mindig is érdekelt, hogy hogyan lehet egy applikációt felépíteni teljesen a nulláról. A témavezetőm segítségével nagyon sok új eszközt és hasznos dolgokat tanultam meg, amelyeket az applikáció és a dolgozatom elkészítése folyamán kamatoztatni tudtam.

Tartalmazza a problémafelvetést, a témaválasztás indoklását és a munka célját.

A célkitűzések részletes megfogalmazása külön fejezetben az irodalmi áttekintés után legyen.

Irodalmi áttekintés

(a fejezet új oldalon kezdve)

Alfejezetekre osztható.

1. Alkalmazás célja, motivációja

Az elkészített alkalmazásom célja az, hogy megkönnyítse a tisztítószereknek a vásárlását és ennek az eszköznek a segítségével a felhasználók jelentős mennyiségű pénzt tudnak megtakarítani. Ehhez persze a felhasználói interakciók is szükségesek, hogy az adatbázisban minél nagyobb mennyiségű adat álljon rendelkezésükre. Minél több termék és a termékekhez tartozó ár kerül be, annál hatékonyabb tud lenni az applikáció, mert a bevásárlólista funkció segítségével több üzletben is láthatják a végösszegeket.

Személy szerint nekem mindig is fontos volt, hogy a vásárlásaimat a legköltséghatékonyabb módon tudjam teljesíteni. Viszont mindig nagyon sok időbe telt megtalálni a legjobb árakat, vagyis azokat az üzleteket, ahol a legjobb áron tudok hozzájutni különböző termékekhez. Sajnos ez nem mindig jön össze, mert lehet, hogy az egyik áruházban jelentősen tudunk spórolni egy terméken, de lehet, hogy egy másik viszont lényegesen többe kerül. Ahhoz pedig, hogy körbejárjuk az összes környező üzletet, nagyon időigényes tud lenni. A mai rohanó világban sajnos nehezen tehetjük meg azt, hogy akár 8-10 üzletet is meglátogassunk a legjobb árak reményében. Innen jött az ötlet, hogy egy olyan applikációt valósítsak meg, ami kiszámolja és megmutatja nekünk, hogy melyik üzletben járunk a legjobban a végösszeget tekintve.

2. Mobilalkalmazás fejlesztése

2.1 Natív és multiplatform fejlesztői eszközök története

Mivel a fejlesztők a natív alkalmazásfejlesztés során egy adott platformra kódolnak, teljes mértékben hozzáférnek a natív eszközfunkciókhoz. Ez általában nagyobb teljesítményt és sebességet eredményez a keresztplatformos alkalmazásfejlesztéshez képest.

Másrészt, ha egy alkalmazást több platformon szeretne elindítani, a natív alkalmazásfejlesztés több kódot és több fejlesztőt igényel. E kiadások mellett a natív alkalmazásfejlesztés megnehezítheti a különböző platformokon egyidejűleg, konzisztens felhasználói élményt nyújtó indítást. Itt lehetnek hasznosak az olyan keresztplatformos alkalmazásfejlesztési keretrendszerek, mint a Flutter.

Natív alkalmazások is nagy mennyiségben fordulnak elő. Segítségükkel teljesen kihasználhatjuk az adott operációsrendszer adta lehetőségeket. Androidon az Android NDK segítségével fejleszthetünk natív alkalmazásokat. IOS-en pedig a SwiftUI áll rendelkezésünkre. Android alapú alkalmazásokat az összes operációsrendszeren tudunk fejleszteni. Natív iOS alapú alkalmazás fejlesztéséhez szükségünk van egy MacOS operációsrendszerű eszközre. Ugyanakkor egy startup számára kitűnő választás lehet egy multiplatformos keretrendszer választása, mert ezzel időt és pénzt spórólhatnak. Nem beszélve arról, hogy ezzel a fenntartási költségeiket is csökkenthetik, mert csak egy kódbázist kell fenntartaniuk. Sokszor azért fejlesztenek inkább natív applikációkat, mert gyorsabbnak tartják, ugyanakkor napjainkban egyre nagyobb térhez jutnak a multiplatform alapú alkalmazások. Utóbbi időben több keresztplatformos keretrendszer is jelentős népszerűségnek örvend.

A szakdolgozatom tervezési fázisában én is több ilyen keretrendszer közül választhattam. A három legelterjedtebb közül végül a Flutterre esett a választásom, mert ebben a szoftverfejlesztői csomagban nem csak mobilapplikációkat tudok készíteni, hanem ugyanazt a kódbázist fel tudom használni más platformok alkalmazásaihoz is. A Flutter keretrendszer a Dart programozási nyelvet használja.

2.2 Flutter és Dart

A Flutter egy nyílt forráskódú keretrendszer, amelyet a Google fejlesztett ki és támogat. A frontend- és full-stack fejlesztők a Fluttert arra használják, hogy egyetlen kódbázisból több platformra is elkészítsék egy alkalmazás felhasználói felületét (UI).

Amikor a Flutter 2018-ban elindult, elsősorban a mobilalkalmazások fejlesztését támogatta. A Flutter ma már hat platformon támogatja az alkalmazásfejlesztést: iOS, Android, web, Windows, MacOS és Linux. A szakdolgozatomként egy iOS, Android multiplatformos mobilalkalmazást készítettem el. Az alkalmazásomat ezeken az operációsrendszereken teszteltem.

A Flutter leegyszerűsíti a konzisztens, vonzó felhasználói felületek létrehozásának folyamatát egy alkalmazás számára az általa támogatott hat platformon. A Flutter egy platformokon átívelő fejlesztési keretrendszer.

A Flutter előnyei

Közel natív teljesítmény. A Flutter a Dart programozási nyelvet használja, és gépi kódba fordítja. A fogadóeszközök megértik ezt a kódot, ami gyors és hatékony teljesítményt biztosít.

Gyors, konzisztens és testreszabható renderelés. Ahelyett, hogy platformspecifikus renderelőeszközökre támaszkodna, a Flutter a Google nyílt forráskódú Skia grafikus könyvtárát használja az UI rendereléséhez. Ez konzisztens vizuális megjelenítést biztosít a felhasználók számára, függetlenül attól, hogy milyen platformon érik el az alkalmazást.

Fejlesztőbarát eszközök. A Google a Fluttert úgy építette, hogy a hangsúlyt a könnyű használatra helyezte. Az olyan eszközökkel, mint a hot reload, a fejlesztők előzetesen megnézhetik, hogyan fognak kinézni a kódváltozások anélkül, hogy elveszítenék az állapotot. Ez a funkció nagyban megkönnyítette a fejlesztési fázist, mivel egy-egy változtatás után azonnal láttam az eredményt a futtatott emulátoron. Így sokkal egyszerűbb dolgom volt a UI és a funkciók kialakítása közben. Más eszközök, például a widget inspector megkönnyítik a felhasználói felület elrendezésével kapcsolatos problémák vizualizálását és megoldását.

A Flutter és Dart számomra teljesen új volt. Ezért az elején kisebb nehézségekkel kellett megküzdenem. Meg kellett értenem, hogy hogyan is épül fel egy Flutter alkalmazás. A widgeteket jobban megismerve lett könnyebb az alkalmazás fejlesztése. A Flutterben a fejlesztők a felhasználói felület elrendezéseit widgetek segítségével építik. Ez azt jelenti, hogy minden, amit a felhasználó a képernyőn lát, az ablakoktól és a panelektől kezdve a gombokig és a szövegig, widgetekből áll.

A Flutter widgeteket úgy tervezték, hogy a fejlesztők könnyen testre szabhassák őket. A Flutter ezt egy kompozíciós megközelítéssel éri el. Ez azt jelenti, hogy a legtöbb widget kisebb widgetekből áll, és a legalapvetőbb widgeteknek speciális céljaik vannak. Ez lehetővé teszi a fejlesztők számára, hogy új widgeteket kombináljanak vagy szerkesszenek.

A Flutter a widgeteket saját grafikus motorjával rendereli, ahelyett, hogy a platform beépített widgetjeire támaszkodna. Így a felhasználók platformok között hasonló megjelenést és érzetet tapasztalhatnak a Flutter-alkalmazásokban. Ez a megközelítés rugalmasságot is biztosít a fejlesztők számára, mivel egyes Flutter widgetek olyan funkciókat is elvégezhetnek, amelyeket a platformspecifikus widgetek nem.

A Flutter a közösség által fejlesztett widgetek használatát is megkönnyíti. A Flutter architektúrája támogatja, hogy több widgetkönyvtárral rendelkezzen, és a Flutter ösztönzi a közösséget, hogy újakat készítsen és tartson fenn.

A Flutter a nyílt forráskódú Dart programozási nyelvet használja, amelyet szintén a Google fejlesztett ki. A Dartot UI-k építésére optimalizálták, és a Dart számos erősségét a Flutter is felhasználja.

A Dart egy ügyfél-optimalizált nyelv gyors alkalmazások fejlesztésére bármilyen platformon. Célja, hogy a leghatékonyabb programozási nyelvet kínálja a multiplatformos fejlesztéshez, az alkalmazás-keretrendszerek rugalmas végrehajtási futásidejű platformjával párosítva.

A Dart képezi a Flutter alapját is. A Dart biztosítja a nyelvet és a futási időt, amelyek a Flutter-alkalmazásokat működtetik, de a Dart számos alapvető fejlesztői feladatot is támogat, például a kód formázását, elemzését és tesztelését.

Például a Dart egyik, a Flutterben használt jellemzője a null ellenőrzés. A Dart nullellenőrzése megkönnyíti a null hibáknak nevezett gyakori hibák felderítését. Ez a funkció csökkenti a fejlesztők kódkarbantartásra fordított idejét, és több időt hagy nekik arra, hogy az alkalmazásuk fejlesztésére koncentráljanak.

3. UX/UI DESIGN

3.1 Történeti áttekintő

A felhasználói élménytervezés egy konceptuális tervezési tudományág, amelynek gyökerei az emberi tényezőkben és az ergonómiában gyökereznek, egy olyan területen, amely az 1940-es évek vége óta az emberi felhasználók, a gépek és a kontextuális környezet közötti kölcsönhatásra összpontosít, hogy a felhasználói élményt szolgáló rendszereket tervezzen. Donald Norman professzor, a formatervezés, a használhatóság és a kognitív tudományok kutatója alkotta meg a "felhasználói élmény" kifejezést, és tette szélesebb körben ismertté.

A UI azokat a képernyőket, gombokat, kapcsolókat, ikonokat és egyéb vizuális elemeket jelenti, amelyekkel egy weboldal, alkalmazás vagy más elektronikus eszköz használata során interakcióba léphet. A UI feladata a gépek és szoftverek, például számítógépek, háztartási gépek, mobileszközök és egyéb elektronikus eszközök felhasználói felületeinek tervezése, amelynek középpontjában a használhatóság és a felhasználói élmény maximalizálása áll.

A felhasználói élménytervezés (UX) az a folyamat, amelyet a tervezőcsapatok használnak olyan termékek létrehozására, amelyek értelmes és releváns élményt nyújtanak a

felhasználóknak. Az UX-tervezés magában foglalja a termék megszerzésének és integrálásának teljes folyamatát, beleértve a márkaépítés, a design, a használhatóság és a funkció szempontjait. Bár a UI minden bizonnyal hatással lehet az UX-re, a kettő mégis különbözik egymástól.

3.2 Főbb, alkalmazott tervezési szabályok

A UX (felhasználói élmény) tervezési folyamata öt szakaszra osztható: EMPÁTIA, MEGHATÁROZÁS, ÖTLETALKOTÁS, TESZTELÉS és PROTOTÍPUS KÉSZÍTÉS. Ezek a szakaszok gyakran ebben a sorrendben követik egymást, és fontos tudni, hogy a UX egy iteratív folyamat. Ezt a folyamatot szem előtt tartva jobb applikációt tudunk építeni a felhasználó számára.

Empátia: Ez a UX tervezési folyamat első szakasza, ahol meg kell találnod, milyen kihívásokkal szembesülnek az applikáció felhasználói, különböző feladatok elvégzése közben. Itt először is ki kellett találnom, hogy hogyan is szeretném megvalósítani az alkalmazást és miért lehet hasznos más felhasználó számára. Felhasználói szemszögből kellett átlátnom az applikációm részeit. Át kellett gondolnom azt is, hogy a felhasználók milyen funkciókat tudnak beleképzelni az alkalmazásba és milyen logika alapján lehetne hasznos számukra.

Meghatározás: Itt megpróbáltam különböző részekre szétbontani az alkalmazásomat. Először is a legfontosabb részekre koncentráltam, ennek a logikáját próbáltam kitalálni. A nehézségek akkor adódtak, amikor több részt kellett összekapcsolnom. Ezekhez az összetett részekhez több idő kellett mire rájöttem, hogy hogyan is lehetne megfelelően alkalmazni.

Ötletalkotás: Ebben a szakaszban próbáltam minél több ötletet összegyűjteni. Tudtam, hogy nem tudom megvalósítani mindegyiket, de így legalább volt miből válogatnom a későbbiekben. Ugyanakkor, ahogy egyre több és több megvalósítási példa jutott eszembe, úgy néha a korábbiakon is változtatásokat kellett végeznem, hogy összhangban legyenek.

Prototípus: A prototípus készítési folyamat nagyon lényeges volt a tervezési fázisban. Itt az ötleteim alapján próbáltam megvalósítani az alkalmazásom kinézetét. Több olyan funkció is volt, ami ötletként jól hangzott, de a prototípusban már nem mutatott megfelelően. Ezért volt jó, hogy az előző szakaszban több ötletem is volt és ezekből könnyedén tudtam válogatni.

Tesztelés: A tesztelés rész azért fontos, mert ezáltal észrevehetjük a hibákat a tervezési fázisban, és láthatjuk, hogy az eddig létrehozott tervek mennyire felelnek meg az

elvárásainknak. A prototípusomban teszteltem a navigációt a képernyők között, így láthattam, hogy hogyan függenek össze az alkalmazásom részei.

4. MOODBOARD ÉS LOGO

4.1 Moodboard

A moodboard a koncepciótervezés vizuális megjelenítése inspirációk gyűjteménye formájában. Az elnevezés tulajdonképpen elég pontos, mivel lényegében a tervező által összeállított hangulattábláról van szó. A moodboardok lényege, hogy a koncepció megalkotásának szakaszában rendszerezze az inspirációkat, legyen szó akár egy mobilalkalmazásról vagy bármilyen más kreatív munkáról - belsőépítészet, divat stb.

A moodboard lehet digitális és/vagy fizikai formában is. Az első esetben kiválasztott anyagokból készítünk kollázst szoftver segítségével, az "analóg" tábla esetében használhatunk papírt, parafatáblát vagy falat, ahová magazinkivágásokat, anyagokat, fotókat stb. ragasztunk.

A moodboard általában a projekt kezdeti szakaszában készül.

A moodboard nagyon hasznos lehet az ötletek rendszerezésében. Azzal, hogy az összes koncepciót és inspirációt egy helyre helyezzük, gyorsan és hatékonyan ellenőrizhetjük a kompozíciót és annak harmóniáját. Kiderülhet, hogy a javasolt színek, minták vagy motívumok nem feltétlenül harmonizálnak egymással - egy hangulattábla segít abban, hogy már a kezdetektől fogva meghatározzuk és megtartsuk az irányt, és időt takaríthatunk meg a további szakaszokban.

A tervezéskor különböző Moodboard képeket valósítottam meg. Ezek a képek megmutatják az alkalmazásban használt színeket, dizájnokat. Ugyanakkor ezekkel a képekkel megmutathatjuk az alkalmazásunk témáját és hangulatát. Könnyen átláthatjuk vele az applikációnk terveit, stílusait. Különféle dizájn elemeket is megjeleníthetünk ezeken a képeken. Ezeket a képeket a Canva képszerkesztő segítségével valósítottam meg. A kiexportált képernyőképek nagy segítségemre voltak, mert ezeket könnyen be tudtam illeszteni egy-egy Moodboardos képhez. Igyekeztem a jelentősebb funkciókat megjeleníteni, amelyek fontos részét képezik az alkalmazásomnak. Olyan képeket is készítettem, amelyek az alkalmazás témájára voltak kihegyezve, vagyis a takarítószeres motívumok is megjelentek rajta, ezzel is mutatva azt, hogy milyen témában fog majd elkészülni az alkalmazásom.

4.2 Logo

A logó elkészítését nagyban megkönnyítette a tervezési fázisban létrehozott kezdőképernyő kép, mert ebből fel tudtam használni az egyik képet a logóhoz. A tervezés kezdetén próbáltam minél összetettebb képet létrehozni, szöveggel és több kép megjelenítésével, de gyorsan rájöttem, hogy logónak egy egyszerű, de annál feltűnőbb motívumot kell választani. Ezért esett a választásom a kezdőképernyőn szereplő kesztyűkre, mert ezzel a rózsaszín színnel egyedi és feltűnő megjelenést ad az alkalmazásnak. A logó tervezését Canvában hajtottam végre, ez egy képszerkesztő eszköz, ami nagyon megkönnyíti a képekkel való munkát. (további canva ismertető) Ügyeltem arra, hogy a tervezek 1024x1024 felbontásban készítsem el. Ezután az appicon.co weboldalt használtam, ahol le tudtam generálni az alkalmazás ikonjait. Ezen az oldalon több platform közül is választhattam. A megfelelő felületek kiválasztása után egy zip fájlt tudunk letölteni. Ezt a zip fájlt kicsomagolva és az Android mappát megnyitva tekinthetjük meg az elkészült ikonokat többféle felbontásban, ami ahhoz kell, hogy a logó megfelelően jelenjen meg például az alkalmazás áruházban. Ezeket a mappákat a Flutter projekten belül az android > app > src > main > res mappába kellett helyeznem. Figyelnem kellett arra, hogy a mappákban PNG kiterjesztésű fájloknak a neve megegyezzen, mert ez az applikáció buildelésénél gondot okozhat. Ha mégis megváltoztattuk ezeknek a fájloknak a nevét, vagy már úgy generáltuk le, akkor az android > app > src > main > AndroidManifest.xml fájlban az android:icon=”@mipmap/ic_launcher” nevét meg kell változtatnunk majd újrabuildelés után megoldódik a probléma.

Ez első ilyen változtatás után viszont nem az elvárt eredményt kaptam. Sokkal kisebb volt az applikáció ikonja, mint ahogy arra számítottam. Többszöri próbálkozásra sem sikerült megoldani, mikor találtam egy weboldalt(link), ahol le volt írva, hogyha a generált kép négyzet alakban, vagyis nem lekerekítve lett létrehozva, akkor az Android operációs rendszeren kisebb méretűként jelenik meg. Ezt úgy lehet orvosolni, hogy az ikont már Canvából úgy exportálom ki, hogy a sarkai le legyenek kerekítve, de ez még nem elég ahhoz, hogy tökéletes megjelenést biztosítsunk az applikációnknak. Ezt a képet átlátszó háttérrel kellett kiexportálni, hogy fehértől eltérő háttér esetén is megfelelően jelenjen meg. Ha még professzionálisabb logókat szeretnénk tervezni, akkor a segítségünkre lehet az appicons.ai weboldal, ahol mesterséges intelligencia segítségével tudunk szebbnél szebb ikonokat generálni.

5. KÉPERNYŐTERVEK

Az alkalmazásom tervezését először a képernyőtervekkel kezdtem. Ehhez egy kiváló eszközt, a Proto.io-t használtam. A Proto.io egy iparágvezető webes prototípuskészítő platform, amely alkalmas UX-tervezők, vállalkozók, termékmenedzserek, marketingesek és bárki számára, akinek van egy nagyszerű ötlete. A Proto.io segítségével mindenki pillanatok alatt életre keltheti ötletét. Nincs szükség különleges tervezési ismeretekre: a Proto.io komponenskönyvtárak és sablonok segítségével a tervezés olyan egyszerű, mint a drag-and-drop. Könnyen hozzá lehet adni bármit, az egyszerű kattintási pontoktól kezdve a hatékony interakciókig és gyönyörű animációkig, ezzel egy olyan prototípust létrehozva, ami teljesen valós applikációnak tűnik.

Ennek az eszköznek a segítségével interaktív prototípusokat lehet készíteni, megosztani, bemutatni és tesztelni különféle képernyőméretekre: mobil, táblagép, web, TV, okosóra. Könnyen és gyorsan használatba lehet venni ugyanis nem kell telepíteni, és nem kell fájlokat mozgatni: egy teljes körű, webalapú prototípuskészítő platform.

Ennek az eszköz segítségével szinte teljesen fel tudtam építeni az alkalmazásomat, anélkül, hogy egy sor kódot le kellett volna írnom. Számomra ez nagy segítség volt az elején, mert így nem kellett azon izgulnom, hogy minél gyorsabban elsajátítsam az alkalmazásom fejlesztéséhez szükséges eszközöket.

A tervezést úgy kezdtem, hogy az interneten próbáltam minél több mobilapplikáció tervet megtekinteni és ezekből próbáltam ötleteket meríteni. Kezdetben nem ment olyan egyszerűen a képernyők létrehozása, mert egy elég összetett eszköztár fogadott. Első prototípus elkészítése után, ami megadta az applikációnak a vázát, jöhettek a javítások. Azért is jó volt használni ezt az eszközt, mert különböző funkciókat is tudtam tesztelni, mint például az oldalak közötti navigációt.

Miután elkészültem a tervekkel, könnyen ki lehetett exportálni egy HTML fájlba, ami megnyitva egyből megjelenített egy mobileszközt az applikációval. A képernyőterveket külön képként is ki lehet exportálni, ami a Moodboard létrehozásában segített.

6. FEJLESZTÉSI FOLYAMAT

6.1 Trello

A Trello rugalmas eszköz a munkák menedzseléséhez, amelyben a csapatok vizuális, produktív és eredményes módon készíthetnek terveket, dolgozhatnak együtt a projekteken, szervezhetik a munkafolyamatokat és nyomon követhetik az előrehaladást. A Trello segít kézben tartani a közös munka és az eredmények elérésének jelentős mérföldköveit és napi feladatait is az ötleteléstől a tervezésen át a kivitelezésig.

A Trello nagyon sokat segített a munkafolyamat megfelelő követésében. Segítségével rendezetten tudtam tartani a projekttel kapcsolatos feladatokat és a témavezetőmnek is könnyebb volt ellenőrizni az elvégzett és a hátralévő feladataimat. Az alkalmazásom készítésének az elején voltak nehézségeim a használatával, ugyanis ezelőtt még nem használtam ilyen alkalmazást. Elsőre szokatlan volt a felhasználói felülete, mivel elég sok funkciót tartalmazott. Ezeket a funkciókat jobban megismerve lett egyre gördülékenyebb a használata. Egy idő után már teljesen automatikus volt a használata számomra.

A szakdolgozatomhoz a táblát Trelloban még a tervezési folyamatok során készítettem el. Ezen a táblán nyomon követhetem az információkat és a változásokat. Ehhez a táblához listákat hoztam létre. Ezekben a listákban tudtam létrehozni a kártyákat. A listákhoz a munkafolyamat különböző részeit adtam hozzá. Ezeket a következőként neveztem el: Bug, Backlog, Work in Progress Code Review, Done. A tervezési fázisban a Backlog részbe kellett először felvennem a kártyákat, a megvalósítandó funkciók alapján. Ez az elején nagy tervezést igényelt, mert szét kellett bontanom az alkalmazásomat külön álló részekre. Azt is meg kellett néznem, hogy vannak-e olyan képernyők, ahol hasonló widgetek vannak. Ezzel kiszűrtem azokat a részeket, ahol azonosság van, így nem kell őket többször megvalósítani. Minden képernyőhöz két jegyet vettem fel. Az egyik az üzleti logikát jelentette, a másik pedig a UI-t. Ezekhez a kártyákhoz létrehoztam különböző színű címkéket, ezzel jobban átláthatóvá téve az összeillő részeket. Fontossági sorrend szerint is létrehoztam címkéket. Ezekből háromféle volt, alacsony, normál, sürgős. A UI címkével ellátott kártyákat sürgősnek, az üzleti logikával ellátott kártyákat pedig normálnak állítottam be. Amikor elkezdtem dolgozni egy funkción, akkor a hozzátartozó kártyát a Backlogból a Work in Progress listába tettem. Voltak olyan kártyák, ahol egy tennivalók listát is létrehoztam, így könnyebben tudtam figyelni az elvégzendő feladatokat, sikeresen teljesítés után pedig elvégzettként tudtam jelölni. A Trelloban egy olyan funkció is elérhető, amit a fejlesztés során sokat használtam. Minden pull requestet hozzá tudtam kapcsolni egy kártyához, ezáltal könnyen elérhetővé vált az adott pull

request. Miután elkészültem egy funkcióval. a kártyáját áttettem a Code Review részbe. Ez azt jelenti, hogy a témavezetőmnek át kell néznie, majd sikeres ellenőrzés és a pull request elfogadása után át lehetett helyezni a Done listába.

Bug jegyeket akkor vettem fel, amikor egy nem várt működést tapasztaltam egy olyan részen, ami már be lett olvasztva a master ágra. A bug jegynek az alábbi információkat kellett tartalmaznia; hol van a hiba és kép az adott hibáról, hova kell kattintani, vagy mit kell csinálni, hogy előjöjjön, mi lenne az elvárt működés vagy kinézet. Így ezeket a hibákat könnyedén tudtam javítani, mert tudtam, hogy hol jön létre a hiba. Ezekhez a jegyekhez is mindig egy új ágat hoztam létre.

6.2 GitHub

A fejlesztéshez a GitHub verziókövető rendszert használtam. A GitHub egy webes felület, amely valós idejű együttműködést tesz lehetővé. A GitHub segítségével könnyedén nyomon követheti a változásokat és navigálhat a verziók között. A Git a GitHub által használt verziókezelő rendszer. A Git nyílt forráskódú és ingyenesen használható kis és nagy projektek számára. Ez a rendszer követi nyomon a GitHubban végrehajtott minden változtatást.

Ez nagyban megkönnyítette a fejlesztési fázist, mert így tisztán és átláthatóan volt tárolva a kódom. Korábban már használtam GitHubot, viszont ennek az alkalmazásnak a fejlesztése során tanultam meg igazán a használatának a fontosságát. Ebben nagy segítségemre volt a témavezetőm, aki mindig újabb technikákat mutatott meg a verziókövetővel kapcsolatban. Fejlesztés során úgy éreztem magam, mintha egy valós projekten dolgoznék. Nagyon sok olyan álláshirdetést lehet találni mostanában, ahol a git használatát szinte alapnak veszik. Számomra ezért is volt fontos, hogy mélyebben megismerkedjek ennek az eszköznek a helyes használatával.

A fejlesztés git feature-branch munkafolyamat alapján zajlott, ahol minden új funkcióhoz egy új branchet kellett létrehoznom. Ez azért volt fontos, mert így csak a megfelelően megvalósított és működő funkciókat tudtam hozzáadni a master branchez. Így, ha egy új funkció létrehozásánál valami nem működött megfelelően, vagy bugokat észleltünk, akkor ezeket még azelőtt tudtam javítani, hogy egyesítettem volna a master branchez.

Egy új branch létrehozása és megvalósítása úgy zajlott, hogy először is a masterből létrehoztam egy új branchet a git checkout -b new-feature paranccsal. Fontos, hogy mindig a

master legfrissebb verziójával dolgozzak. Abban az esetben, hogy az új branchet nem a masterből hozom létre, akkor az egyesítés előtt egy rebase-t is el kell végezni. Sikeres létrehozás után jöhetett a funkció megvalósítása. A módosított vagy újonnan hozzáadott fájlokat a git status parancs segítségével tudtam megtekinteni. Az elején ennek a parancsnak a használata sokszor hasznos volt, mert így ki tudtam szűrni azokat a generált fájlokat, amelyekre nem volt szükségem és így könnyen tudtam frissíteni a .gitignore fájlt. Ennek a fájlnak a megléte fontos volt, mert így csak a nemgenerált fájlokat töltöttem fel GitHubra, ezzel jelentősen csökkenthetjük a fentlévő könyvtárnak a méretét. A Flutter a projekt létrehozásakor automatikusan létrehoz egy ilyen fájlt, de ez a fájl nem tartalmazza az összes olyan feltételt, ami segít kiszűrni a generált fájlokat. Ezért 2 további sort is bele kellett írnom az alapértelmezetten generált fájlhoz. **/flutter/generated_plugin_registrant. ***/flutter/generated_plugins.cmake Így már csak tényleg azok a fájlok kerülnek fel, amelyek nem generálva vannak. A Flutter egyszerre több .gitignore fájlt generál le az operációs rendszer specifikus mappákban. Ezeket töröltem, mert arra törekedtem, hogy csak egy ilyen fájlom legyen a projekt gyökérkönyvtárában. Ha további fájlokat szeretnénk figyelmen kívül hagyni, akkor csak egy új sort kell hozzáadnunk, ami tartalmazza a feltételt.

A fejlesztési folyamat úgy zajlott, hogy egy új funkció létrehozásakor létrehoztam egy új ágat. Ezt általában a főágról, a masterről származtattam le. Ez akkor volt hasznos, amikor egy teljesen különálló funkciót akartam hozzáadni az alkalmazásomhoz. Azokban az esetekben, amikor viszont szükségem volt egy-egy branch előző állapotára, akkor az adott branchből hoztam létre az új ágat. Ez sokat segített, mert így könnyebben tudtam tesztelni az új funkciót, a már meglévők használatával. Funkciók megvalósítása közben próbáltam minél többet commitolni, hogy könnyebben visszakövethetőek legyenek a változtatások. Miután elkészültem, kellett tartanom egy önellenőrzést a feltöltés előtt. Ehhez egy lista lett létrehozva GitHub Wiki-ként, ahova azokat a pontokat írtam, amelyeket ellenőriznem kellett a feltöltés előtt. Néhány pont, amelyet figyelembe kellett vennem az önellenőrzés folyamán: Padding használata SizedBox helyett, felesleges megjegyzések, vagy extra kódmagyarázó sorok, MediaQuery helyett Expanded vagy FractionallySizedBox használata, a szövegek kiszervezése egy külön fájlba. Ez az ellenőrzés nagyban megkönnyítette a témavezetőm dolgát, amikor átnézte a változtatásaimat. Ezután azt következett, hogy a változtatásokat feltöltöttem a GitHub távoli tárolójába. Mindegyik branchez létrehoztam egy új pull requestet. Ez azért kellett, mert nem egyből a masterre töltöttem fel az új funkciókat. A master branchre beállítottam egy olyan szabályt, hogy legalább egy jóváhagyás szükséges ahhoz, hogy pull requestben össze tudjam

olvasztani az új ágat a masterrel. Így erre a főágra már csak azok a funkciók kerültek be, melyek átnézve és ellenőrizve voltak.

7. ALKALMAZÁS SZERKEZETI FELÉPÍTÉSE

7.1 MVVM

Alkalmazásomat a Model-View-ViewModel (MVVM) architektúra alapján készítettem el. Ennek az architektúrának a segítségével külön tudjuk szedni az alkalmazás UI részét az üzleti logikától. Az MVVM segítségével a View-ből a ViewModelbe tudjuk kiszervezni az üzleti logikát. Így a View csak a megjelenítéssel tud foglalkozni. Több előnye is van az MVVM alkalmazásának:

Karbantarthatóság: Mivel az alkalmazás View és üzleti logika részét külön tudjuk választani, így a kódunk könnyen karbantarthatóvá és újrafelhasználhatóvá válik.

Tesztelhetőség: A View és a ViewModel elválasztása miatt könnyebben tudjuk tesztelni az üzleti logikát. Ezáltal alkalmazásunk tesztelhetősége jelentős mértékben javul.

Bővíthetőség: Ennek az architektúrának az alkalmazásával az applikációnk könnyen bővíthetővé válik. Ami a későbbi változtatásokat nagyban megkönnyíti majd.

A Model segítségével kapcsoljuk össze a View-t és a ViewModel-t. Segítségével valós időben tudjuk tárolni a lekérdezési adatokat, vagy az adatbázissal kapcsolatos lekérdezéseket.

A ViewModel a Model és a View között van. Segítségével tudjuk kezelni a felhasználói eseményeket és az üzleti logikát. A ViewModel segítségével adatokat tudunk lekérni a Modeltől, amit a View-ben tudunk megjeleníteni. Itt tároljuk az adatbázisból lekérdezett adatokat, amelyeket a View-be közvetít tovább.

A View csak a megjelenítéssel foglalkozik. Itt jelennek meg a felhasználó számára látható widgetek. Felhasználói interakció hatására a View jelzi a ViewModel számára a létrejövő eseményt, amit majd a ViewModel fog elvégezni. A ViewModelből kapja meg az adatokat és azokat jeleníti meg.

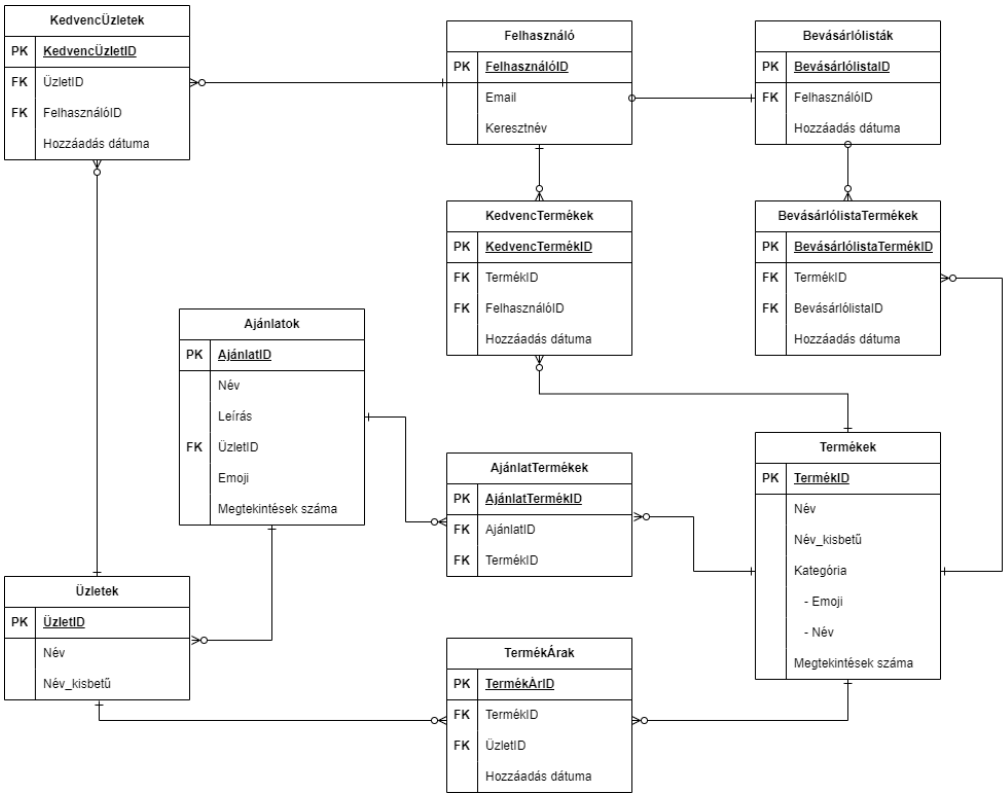
Az MVVM-et manapság széles körben alkalmazzák, mivel támogatja az eseményvezérelt megközelítést, ami Flutter alkalmazásoknál különösen hasznos lehet, mert a komponensek nagy része események alapján történik.

7.2 Adatbázis

Az alkalmazásom egyik fontos része az adatbázis, amelynek a terveit a prototípus elkészítése után valósítottam meg. Fontos volt, hogy meg legyenek tervezve az adatbázis táblák, mert addig nem tudtam foglalkozni az alkalmazásom adatbázis részével. Az applikációmat a Firebase segítségével hostoltam. Ennek a platformnak a segítségével könnyedén megvalósítottam a felhasználók regisztrációját és bejelentkeztetését. Mindemellett a Cloud Firestore adatbázist is tudtam használni az adatok tárolására. A Cloud Firestore egy NoSQL adatbázisszolgáltatás, amelynek segítségével könnyedén tárolhatunk, szinkronizálhatunk és lekérdezhetünk adatokat mobil- és webes alkalmazásaink számára - globális szinten.

Ennek a szolgáltatásnak a tulajdonságait figyelembevéve terveztem meg az adatbázis tábláit. Mindegyik tábla tartalmazott egy ID-t, ezzel megkönnyítve az adatok tárolását. A Firestore minden új dokumentumhoz egy egyedi ID-t hozott létre, így a táblák könnyen összekapcsolhatóvá váltak.

Az elkészült EK-diagram:



1. ábra: EK-diagram

Szükség esetén az ábra eredetére vonatkozó hivatkozás

Mint ahogy a diagramon is látható az adatbázis 10 táblából áll. Ezek a táblák között többféle kapcsolat is látható.

Az MVVM architektúra alapján ezért létre kell hozni egy Service osztályt is, ami az adatbázis műveletekkel foglalkozik. A Service osztályban szereplő metódusokat a ViewModel hívja meg és kezeli a lekérdezések eredményeit. A ViewModel és a Service között egy új réteget kell ezért bevezetnünk, ez lesz a DTO (Data Transfer Object). Ez az alkalmazás legalsó rétege. Ennek az a célja, hogy az adatbázisból jövő dokumentumot a kódunk számára értelmezhető objektummá konvertálja.

Nagyon hasznos ennek a rétegnek használata mivel egy esetleges adatbázis szolgáltatás lecserélése után nem sérül majd a ViewModel és a View közötti kommunikációs objektum. Ez annak köszönhető, hogy a két felső réteg független a legalsó rétegtől. Azért használunk különböző rétegeket, mert így bármikor lecserélhető és egy esetleges cserélnél csak a lecserélt réteg kommunikációs objektumait kell módosítani.

7.3 Üzleti logika

Az üzleti logikát a ViewModel osztályokban valósítottam meg. Ez az a része az alkalmazásomnak, amely összekapcsolja a View és a Model osztályokat. Minden View-hez létrehoztam egy ViewModelt. Ez gondoskodott arról, hogy az adatbázis műveletek megfelelően legyenek meghívva, miközben a hibakezelés is ezen a részen történt. A Model objektum változásait és kezelését a ViewModel végzi. Fontos, hogy az üzleti logika részét a ViewModelben kezeljük, vagyis a View már csak egy Model objektumot kapjon, amiből kiolvasva tud majd adatokat megjeleníteni.

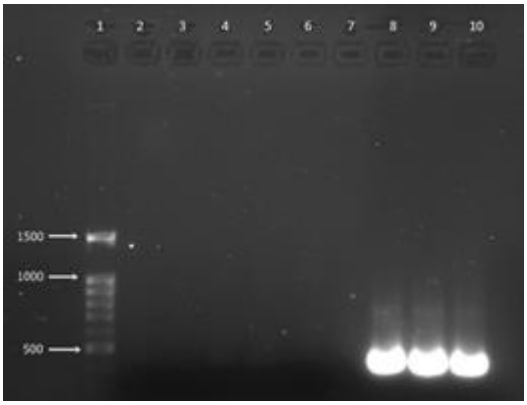
A ViewModel osztályok a ChangeNotifier-t használják, ami az adatok változását figyeli és értesíti a View osztályt a Provider csomag segítségével.

7.4 Kész alkalmazás

A dolgozat témájához kapcsolódó korábbi eredmények ismertetése, a rendelkezésre álló szakirodalmi adatok összefoglalása, elemzése.

A felhasznált irodalmat a szövegben hivatkozni kell (zárójelben) vagy [kapcsos zárójelben]. A hivatkozás történhet számozással (1, 2) vagy az első szerző nevével és a megjelenés évszámával pl. (több szerző esetén: Gipsz és mtsai, 2013; két szerző esetén: Monroe és O'Donell, 2001). A hivatkozott irodalmat a dolgozat végén az irodalomjegyzékben össze kell gyűjteni olyan módon, hogy mások számára is fellelhető legyen!

Tartalmazhat szakirodalmi ábrákat. Ezeket középre kell rendezni és hivatkozni kell rájuk a szövegben (1. ábra). A hivatkozások, akár irodalmi, akár ábrahivatkozás, részei a mondatnak!



2. ábra: Az ábra címe (az ábraaláírást mindig az ábra alá kell elhelyezni!)
Szükség esetén az ábra eredetére vonatkozó hivatkozás

Célkitűzés

Egy olyan alkalmazás létrehozása, amely az MVVM architektúra alapján és a különböző programozási paradigmákat követi. Számomra a Flutter és a Dart teljesen ismeretlen volt. Az ismereteimet először egy Udemy-n elérhető kurzussal bővítettem. Ez azért is volt nagyon hasznos, mert megtanultam az alapokat, hogy hogyan épül fel a Flutter, hogyan kell Widgeteket és új képernyőket létrehozni. Egészen az alapoktól a kurzus vége felé eljutottam haladó technikákig is. Mivel ez volt az első komolyabb alkalmazás, amit egyedül készítettem el, az elején sok akadályba ütköztem, de ilyenkor mindig a legjobb tudásom szerint próbáltam tovább haladni. Ugyanakkor nem csak a programozás részével kellett foglalkoznom, hanem egyéb technológiákat is kellett alkalmaznom. A tervezési fázisban először is meg kellett terveznem az alkalmazásom prototípusát. Ez egy terület volt nekem, mert ezelőtt még nem foglalkoztam UI tervezéssel. Nagy segítségemre volt a proto.io, ahol szinte egy kész alkalmazást tudtam megtervezni, úgy, hogy egy sor kódot sem kellett írnom. Itt utánanéztem a különböző tervezési mintáknak és azokat követve próbáltam megvalósítani az alkalmazásom prototípusát. Ez a későbbiekben nagy segítségemre volt, mert így könnyebben tudtam haladni a fejlesztési fázisban. Az elkészített prototípust könnyen ki tudtam exportálni, így ezzel elkészültem a képernyőképekkel. A képernyőtervek után jöhettek a Moodboardok elkészítése. Ami azért is volt nagyon hasznos, mert ebben a fázisban meg tudtam határozni az applikációm stílusát és a használt színeket, stílusokat. A fejlesztési folyamatot a Trello és a Github segítésével vittem végig. A Trelloban az elején elkészítettem a teljes alkalmazásfejlesztési ütemtervet. Itt először is listákat kellett létrehoznom, majd csak ezután tudtam jegyeket létrehozni. Mindegyik jegy egy funkció megvalósításához tartozott az applikációban. A Github verziókövető rendszer segítségével tudtam rendszerezni az alkalmazásom kódbázisát. Az elején voltak vele nehézségeim, de a végére nagyon sok új és hasznos git parancsot tanultam meg, amelyeket a későbbiek folyamán tudok majd hasznosítani a munkavilágában.

Felhasznált anyagok és eszközök

(a fejezet új oldalon kezdve)

Szakirodalmi feldolgozás esetén nem releváns!

A következő fejezettel (Alkalmazott módszerek) összevonható.

Alkalmazott módszerek

(a fejezet új oldalon kezdve) Szakirodalmi feldolgozás esetén nem releváns!

Alfejezetekre osztható.

Olyan részleteséggel kell megírni, hogy mások számára megismételhetőek legyenek.

Eredmények

(a fejezet új oldalon kezdve)

Az elért eredmények világos és részletes leírását tartalmazza.

Egyértelműen különüljön el a hallgató saját munkájának eredménye az irodalmi áttekintéstől, elméleti alapokat tartalmazó részektől.

Alfejezetekre osztható.

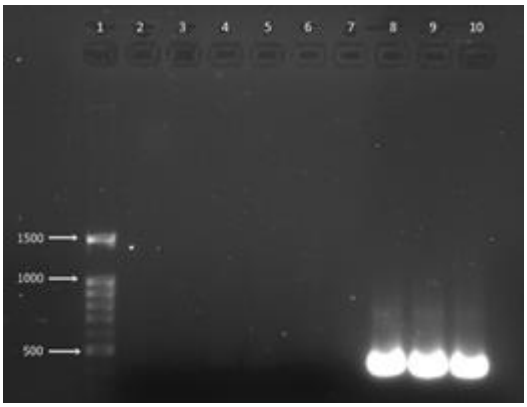
Tartalmazhat táblázatokat, melyekre hivatkozni kell a szövegben (1. táblázat). A táblázatot középre kell rendezni és a szövegben lévő hivatkozás közelében kell elhelyezni.

1. táblázat: A táblázat címe

(a táblázat címe mindig az objektum felett található)

oszlop1	oszlop2	oszlop3
sor1		
sor2		

Tartalmazhat ábrákat. Ezeket középre kell rendezni és hivatkozni kell rájuk a szövegben (x. ábra). Ha az Irodalmi áttekintés fejezet tartalmazott ábrá(ka)t, akkor a számozás ebben a fejezetben nem újra kezdődik, hanem folytatódik,.



3. ábra: Az ábra címe mindig az ábra alatt legyen

Egész oldalas ábrák mellékletben legyenek elhelyezve!!

A képletek, egyenletek az alábbi formában kerüljenek bemutatásra:

$$a = e^{i\pi} + 1 \qquad (1)$$

Középre rendezve és számozva.

A képletben használt mennyiségek legyenek megadva az első használatuknál.

A képlet vagy az MSWord saját Egyenletszerkesztőjével vagy MSWord-ben használható másik egyenletszerkesztővel készítendő.

Összefoglalás

(új oldalon kezdve)

A dolgozat eredményeinek összefoglalása, következtetések levonása.

Az összefoglalásban egyértelműen jelezve legyen a hallgató saját szerepe/eredményei.

Irodalomjegyzék

(új oldalon kezdve)

A hivatkozott irodalom tételes felsorolása olyan módon, hogy mások számára is fellelhető legyen! Nincs előírt formátum, de a dolgozaton belül egységes legyen.

Ha a szövegben a szerző nevével és a publikáció megjelenésének évével történik a hivatkozás, akkor az Irodalomjegyzékben ABC sorrendbe kell tenni a publikációkat.

Példák a hivatkozás formátumára:

Keszthelyi A, Ohkusu M, Takeo K, Pfeiffer I, Litter J, Kucsera J. 2006. Characterisation of the anticryptococcal effect of the FC-1 toxin produced by *Filobasidium capsuligenum*. **Mycoses,** 49:176–183.

vagy:

Oftedal, L., Maple-Grødem, J., Førland, M. G. G., Alves, G., & Lange, J. (2020) Validation and assessment of preanalytical factors of a fluorometric in vitro assay for glucocerebrosidase activity in human cerebrospinal fluid. ***Scientific Reports***, 10(1): 1–8. https://doi.org/10.1038/s41598-020-79104-5

Ha a szövegben számmal történik a hivatkozás a publikációra, akkor a hivatkozásokat sorszámmal kell ellátni a szövegben történt megjelenés sorrendjében.

Példa:

1. Lubelski J, Konings WN, Driessen JM. 2007. Distribution and physiology of ABC type transporters contributing to multridrug resistance in bacteria. *Microbiology and Molecular Biology Reviews***.** 71: 463-476.
2. Higgins CF. 2007. Multiple molecular mechanisms for multidrug resistance transporters. *Nature.* 446: 749-757.
3. https://aws.amazon.com/what-is/flutter/
4. https://trello.com/home
5.

Köszönetnyilvánítás

(nem kötelező elem), (új oldalon kezdve)

Ebben a fejezetben lehet köszönetet mondani mindazoknak, akik segítették a dolgozat elkészülését. Itt lehet megemlíteni továbbá a munkát támogató pályázatokat, ösztöndíjakat, stb.

Nyilatkozat

A szöveg kötött, kérjük ezt használni!

Alulírott, Kovács Alex, programtervező informatikus szakos hallgató, kijelentem, hogy a szakdolgozatban ismertetettek saját munkám eredményei, és minden felhasznált, nem saját munkából származó eredmény esetén hivatkozással jelöltem annak forrását.

Dátum (Szeged, év, hó, nap) _____

aláírás

Mellékletek

(nem kötelező elem, a dolgozat oldalszámaiba nem tartozik bele.), (új oldalon kezdve)

Ebben a fejezetben lehet elhelyezni a nagyobb táblázatokat, ábrákat, adathalmazokat.