Szakdolgozat

Miskolci Egyetem

Androidos alkalmazásfejlesztés bemutatása a horgásznapló rendszer digitalizációjával

Készítette:

Csonka Patrik

Programtervező informatikus

Témavezető:

Dr. Agárdi Anita

Miskolc, 2024

Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Alkalmazott Matematikai Intézeti Tanszék **Szám:**

Szakdolgozat Feladat

Csonka Patrik (CMU4ZN) programtervező informatikus jelölt részére.

A szakdolgozat tárgyköre: Alkalmazásfejlesztés Androidos környezetben

A szakdolgozat címe: Androidos alkalmazásfejlesztés bemutatása a horgásznapló rendszer digitalizációjával

A feladat részletezése:

A Magyarországon használatos papír alapú horgásznapló rendszert fogom megvalósítani Android Studio segítségével egy alkalmazás formájában. A szakdolgozatomban bemutatom a fejlesztés folyamatát ebben a fejlesztői környezetben, illetve részletezem az alkalmazás felépítését és a használatos technikákat, API-kat.

Témavezető: Dr. Agárdi Anita

A feladat kiadásának ideje: 2024 Március 4.

.....
szakfelelős

Eredetiségi Nyilatkozat

Alulírott **Csonka Patrik**; Neptun-kód: `CMU4ZN` a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős Programtervező informatikus szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy *Androidos alkalmazásfejlesztés bemutatása a horgásznapló rendszer digitalizációjával* című szakdolgozatom saját, önálló munkám; az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul veszem, hogy szakdolgozat esetén plágiumnak számít:

- szószerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentem, hogy a plágium fogalmát megismertem, és tudomásul veszem, hogy plágium esetén szakdolgozatom visszautasításra kerül.

Miskolc, ... év ..... hó ..... nap

.....
Hallgató

1. A szakdolgozat feladat módosítása

szükséges (módosítás külön lapon)
nem szükséges

....
dátum témavezető(k)

2. A feladat kidolgozását ellenőriztem:

témavezető (dátum, aláírás):	konzulens (dátum, aláírás):
.....	.....
.....	.....
.....	.....

3. A szakdolgozat beadható:

....
dátum témavezető(k)

4. A szakdolgozat ..... szövegoldalt
..... program protokollt (listát, felhasználói leírást)
..... elektronikus adathordozót (részletezve)
.....
..... egyéb mellékletet (részletezve)
.....
tartalmaz.

....
dátum témavezető(k)

5. A szakdolgozat bírálatra

bocsátható
nem bocsátható

A bíráló neve: .....

....
dátum szakfelelős

6. A szakdolgozat osztályzata

a témavezető javaslata: .....
a bíráló javaslata: .....
a szakdolgozat végleges eredménye: .....

Miskolc, ...
a Záróvizsga Bizottság Elnöke

Tartalomjegyzék

1. fejezet

Bevezetés

A telefonok térnyerése az utóbbi évtizedben tagadthatatlan, életünk számos részét érintette, illetve befolyásolta. Ezek alól nem képeznek kivételt a szabadidős elfoglaltságok, ezáltal a horgászat sem. Ma már nehéz találni olyan horgászati tevékenységet végző embert, akinek a zsebében ne lenne ott egy olyan telefon, amely minimum 50 milliószor több számítást végző kapacitással rendelkezik, mint az 1962-ben létrehozott Apollo 11-es űrrakéta fedélzeti számítógépe [10]. Több horgásztársamnak a fejében is megfordult már, hogy akkor miért kell még mindig a fogásait papírra leírni, majd összegezni az év végén, és leadni azt a helyi horgász szövetségnél.

A rendszerváltás után rendszeresített fogási napló egy fontos problémát hivatott megoldani, ugyanis az egyre népszerűbb sport elfoglaltság veszélyeztette a hazai vizek halállományát, amiről ekkor még oly' keveset tudtunk. Ez a rendszer volt hivatott arra a feladatra, hogy naplózzák, számon tartsák a kifogott halakat és összefüggéseket vonjanak le a hazai vizek élőlény állományáról.

1.1. ábra: 2023. évi állami fogási naplók.

Ezen a ponton találkozott a két kedvenc területem, a horgászat és a programozás. Az egyetemi éveim alatt szerteágazó tudásra tehettem szert, megismerhettem sokféle programozási nyelvet és technikát. Külön kiemelném az adatbázis kezelést, és az objektum orientált programozási elvet, amelyek nagy mértékben hozzásegítettek a szakdolgozati témám kiválasztásához, és megvalósításához. Sokszor szó esett arról, hogy a programozás rengetegszer hétköznapi problémákat hivatott megoldani, ezáltal könnyebbé tenni a felhasználó életét.

Ezen a vonalon haladva határoztam el, hogy a szakdolgozati témám az állami fogási napló rendszer digitalizálása lesz. Ezzel az applikációval szeretném felhívni a figyelmet arra, hogy időleges az előbbiekben említett rendszer átdolgozása, és a mai igényeket kielégítő okos telefonos integráció létrehozása.

A továbbiakban be fogom mutatni, hogy az én elképzelésem alapján hogyan valósítható meg egy Androidos alkalmazás Android Studio fejlesztői környezetben. Célja az alkalmazásnak, hogy felhasználó barát módon segítse horgásztársaimat a hétvégi kikapcsolódás során gyűjtött adatok - például a fogott halak súlya, a fogás pontos helye és időpontja - rögzítésére, és rendszerezésére. Az applikáció ezen túl segít a régebben rögzített fogások visszakeresésére, és áttekintésére. Kiemelt szempont volt az, hogy kezdő telefon felhasználók, ezáltal az idősebb, okostelefont kevésbé használni tudók is könnyen és gyorsan használatba vehessék az alkalmazást. Törekedtem arra, hogy a program kezelői felülete a lehető legegyszerűbb, és átlátható legyen.

1.2. ábra: Horgászati tevékenységet végző ember.

A dolgozat az alábbi felépítést követi: az első fejezet bevezeti az olvasót a jelenlegi piaci helyzetbe, és rámutat a hiányosságokra, majd egy lehetséges megoldást nyújt. A második fejezet a fejlesztéshez szükséges technológiai alapokat, különösen az Android Studio és a hozzá kapcsolódó eszközök használatát mutatja be. A harmadik fejezet az alkalmazás architektúráját és a funkciók részletes leírását tartalmazza, míg a negyedik fejezet az alkalmazás megvalósításának módszereit tárgyalja. Az ötödik fejezet a tesztelési fázist mutatja be, az utolsó fejezet pedig összefoglalja a projekt eredményeit, és javaslatokat fogalmaz meg a jövőbeni fejlesztésekre.

2. fejezet

Az telefonos alkalmazás fejlesztés

A bevezetőben említett probléma nem újkeletű, a MOHOSZ tisztában van a horgász fogási napló digitalizálásával kapcsolatos felmerülő igényekkel, több próbálkozás is volt már, mindent átfogó megoldás azonban eddig még nem valósult meg, csak tervek vannak terítéken[15].

A telefonos alkalmazások fejlesztésére több különböző módszer is létezik, ebben a fejezetben szeretném ezt részletesebben bemutatni.

2.1. Különböző szoftverek bemutatása

A telefonos applikációk fejlesztésére többféle platformon elérhető szoftverek léteznek. Továbbiakban ismertetni szeretném a kifejezetten számítógépes platformra készített népszerűbb programokat, amelyek a telefonos alkalmazás fejlesztést hivatottak megkönnyíteni.

Az egyik legnépszerűbb választás a Flutter, ami a felhasználó számára lehetővé teszi, hogy egyszerre fejlesszenek a két legnagyobb telefonos platformra, azaz Android-ra , és IOS-re (továbbiákban Cross-platform). Ez egy DART keretrendszeralapú ″widget″ fejlszetést kínál, aminek az előnye a gyors User Interface készítés, és széleskörű testreszabhatóság[3]. Készítője a Google.

2.1. ábra: Flutter logó

A másik népszerű választás a React Native[16], amely szintúgy egy Cross-platform alkalmazás készítő szoftver, amelyet a Facebook fejlesztett ki. A program JavaScript, illetve TypeScript programnyelvet használ, nagy hangsúlyt fektet a gyorsaságra.

2.2. ábra: React Native logó

Említésre méltó a Unity[18], amely kifejezetten játék fejlesztésre készült szoftver. Többek között képes számítógépes, konzolos és telefonos alkalmazás fejlesztésre is. Fejlesztési nyelve a C#.

2.3. ábra: Unity logó

A következő alkalmazás az Xcode[1], amely az Apple hivatalos fejlesztői környezete. Ezzel a szoftverrel csak az Apple ökoszisztémán létező operációs rendszerekre tudunk fejleszteni, azaz iOS, iPadOS, macOS, watchOS és tvOS. Fejlesztési nyelve a Swift, és Objective C.

2.4. ábra: Xcode logó

Utoljára pedig az általam választott szoftvert mutatnám be röviden, az Android Studio-t[7]. Ez a Google által fejlesztett hivatalos fejlesztő környezet, amellyel Androidos alkalmazások készítését teszik elérhetővé. Ez egy teljeskörű eszközkészlet, rendelkezik Android SDK-val, és beépített emulátorral, ami megkönnyíti a gyors tesztelést. Fejlesztési nyelve elsősorban a Kotlin, és a Java.

2.5. ábra: Android Studio logó

Az előbbiekben felsorolt fejlesztői környezeteknek nagyon sok előnye, és hátránya van. Az általam megadott szempontok a következők voltak:

- Androidon futtatható legyen az alkalmazás, hozzám közelebb áll a platform szemlélete a nyílt forráskódú applikációk terén.
- A fejlesztő környezet által használt nyelv sem elhanyagolható. Szerettem volna, ha egy magas szintű programozási nyelv lenne a használt az adott környezetben, tekintve hogy azokban szereztem a legtöbb tapasztalatom.
- Fontos volt a beépített emulátor, hogy az alkalmazásomat minél gyorsabban tudjam tesztelni.
- Nem utolsó sorban szerettem volna, ha az általam választott API-kat minél gördülékenyebben tudomm integrálni.

Ezen szempontok alapján haladva úgy döntöttem, hogy az Android Studio lesz számomra a legmegfelelőbb választás.

2.2. Az Android Studio részletezése

Az Android Studio a Google által készített hivatalos fejlesztőkörnyezet, amellyel natív Android alkalmazásokat tudunk készíteni. A környezet a JetBrains IntelliJ IDEA[11] alapjaira épül, ebből lett tovább fejlesztve az Android platformra történő alkalmazások fejlesztésére. Az Androidos platformon ez a legjellemzőbben használt IDE.

2.6. ábra: Android Developers logó

Az Android Studio főbb funkciói, és jellemzői:

- **Emulator:** A beépített emulátorral[6] a fejlesztés során gyorsan ki lehet próbálni az alkalmazáson eszközölt változtatásokat. Ezenkívül az emulatorral beállíthatjuk a tesztelni kívánt Android verziót, illetve készüléket, ezáltal tesztelni tudjuk alkalmazásunkat különböző verziószámú operációs rendszereken.
- **Build System (Gradle):** Az Android Studio Gradle-t[9] használ építési rendszerként, amely megkönnyíti az alkalmazások automatizált építését és kezelheti a különböző verziókat, függőségeket. A Gradle segítségével lehetőségünk van különböző buildek létrehozására, ami kulcsfontosságú volt a fejlesztés egyes fázisaiban.
- **Integrált hibakeresés és teljesítményelemzés:** A fejlesztői környezet lehetővé teszi, hogy könnyen diagnosztizáljuk és kijavítsuk az alkalmazásunk problémáit. Különféle eszközökkel, például CPU és memóriafigyelőkkel segít abban, hogy optimalizáljuk az alkalmazás teljesítményét.

- **Firebase és egyéb felhőszolgáltatások támogatása:** Az Android Studio integrációt biztosít a Google Firebase platformmal[8], ami megkönnyíti az adatbázisok, hitelesítés, felhasználói elemzések, értesítések és más felhőszolgáltatások beépítését az alkalmazásba. Ez volt a legfontosabb szempont számomra, mivel az alkalmazásom több ponton is támaszkodik ezekre az API-kra.
- **Aktív közösség és támogatás:** A Google rendszeresen frissíti az Android Studiot, és nagy fejlesztői közösség[4] is támogatja, ahol gyorsan választ kaphatunk a kérdéseinkre.

2.3. Androidos alkalmazás fejlesztés

Az előbbiekben részleteztem pár szempontot a fejlesztői környezettel kapcsolatban, azonban nem szeretném kihagyni azt sem, hogy milyen előnyei vannak az erre a platformra való fejlesztésnek közvetlenül. A specifikus platform kiválasztás előnyei:

- **Felhasználók aránya:** Magyarországon az Android operációs rendszerrel rendelkező eszközök piaci részesedése eléri a 80% több független forrás szerint is.[17][12]
- **Nyílt forráskód:** Az Android egy nyitottabb ökoszisztéma[5], nagyobb szabadságot kap a fejlesztő az alkalmazás funkcionalitásának, és megjelenítésének létrehozásában.
- **Költségek:** Az alkalmazás fejlesztése, feltéve hogy mi csinálunk mindent, nem kerül pénzbe. Ezalól nem kivétel a fejlesztői környezet használata, alkalmazásunk elérhetővé tétele, és karbantartása sem.
- **Pulikálás:** Az elkészült Androidos alkalmazást sokkal könnyebben tudjuk publikálni különböző áruházakban[2], nem vagyunk rákényszerítve a telefon gyári alkalmazás áruházára.

Az Android platform legfőbb hátránya:

- **Fragmentáció, kompatibilitási problémák:** Legfőbb hátránya a platformnak a megszámolhatatlan mennyiségű hardver, és képernyőméret kombinációk. Ezáltal alkalmazásunk fejlesztésekor nagy valószínűséggel gyártunk olyan hibákat, amelyek láthatatlanok voltak számunkra. Ezentúl különböző Android verziók eltérőek lehetnek mind funkciókban, és használható API-kban. Ezáltal a fejlesztés elhúzódhat, mivel a lehető legtöbb készüléket kell lefednünk az alkalmazásunkal, ami időigényes.

3. fejezet

Tervezés

3.1. Az alkalmazás fejlesztés bemutatása

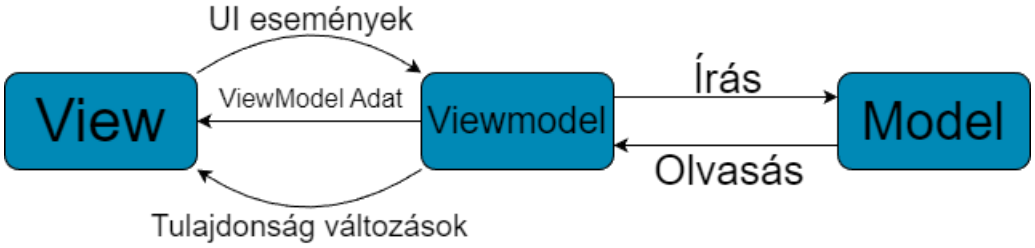
Az Androidos alkalmazás, vagyis az e-Horgásznapló egy Android operációs rendszeren futó, adat rögzítésre használatos alkalmazás. Célom az egyszerű, felhasználóbarát kezelőfelület, ahol a lehető legegyszerűbben rögzíteni tudja a horgász a kifogott hal adatait, és a fogás helyszínét. Az alkalmazást Android Studio segítségével fogom elkészíteni Kotlin segítségével a könnyebb átláthatóság, és kevesebb boilerplate[13] kód miatt. Az alkalmazásom a korábban használt Model-View-controller fejlesztési elv helyett az új Model-View-ViewModel architektúrát használja, ezáltal az alkalmazás jövő állóbb.

3.2. Model-View-ViewModel

Ahogy az előző szekcióban említettem az alkalmazásom a Model-View-ViewModel architektúrát alkalmazza. Ennek a mintának az a célja, hogy elkülönítse az alkalmazás logikai rétegeit, így egyszerűbb lesz a kód karbantarthatósága, és újra használása. Ebben különösen segítségünkre lesz a nemrég bevezetett Jetpack Compose, ami a régi XML elrendezést hivatott leváltani.

A MVVM-nek[14] három fő része van:

- Model: Az adatokat, és a logikát tartalmazó réteg.
- View: A felhasználói felület.
- ViewModel: Köztes réteg, ez köti össze a Model-t, és a View-t. Feladata az adatok feldolgozása, és kezelése.



3.1. ábra: MVVM architektúra diagram

3.2.1. Model

A Model az alkalmazás adatait és üzleti logikáját tartalmazza. Ide tartoznak az olyan elemek, mint az adatbázisok, a hálózati API-k, vagy a fájlkezelés. A Model réteg az adatforrások elérését és az adatok frissítését biztosítja, de nem tartalmaz semmilyen logikát a felhasználói felülethez.

3.2.2. View

A View az a réteg, amely a felhasználói felületet kezeli. Ez felelős az adatok megjelenítéséért és a felhasználói bevitelek kezeléséért. A View réteg nem tartalmaz logikát az adatok feldolgozására; csak megjeleníti az adatokat, amelyeket a ViewModel szolgáltat számára.

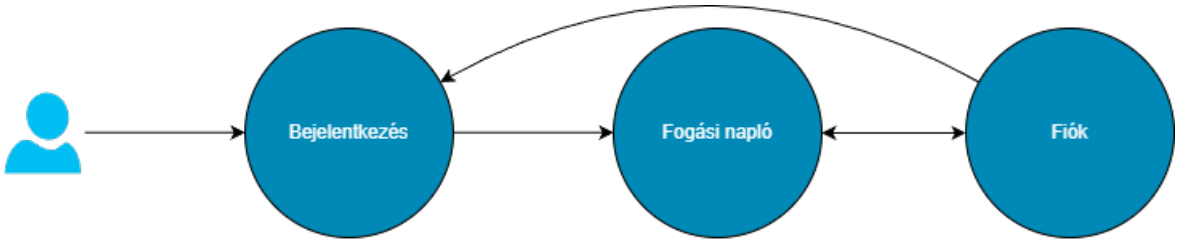
3.2.3. ViewModel

A ViewModel az MVVM minta központi eleme, amely kommunikál a Model és a View között. A ViewModel adatokat kér a Modelltől, feldolgozza azokat, és úgy adja át a View-nak, hogy azok egyből megjeleníthetők legyenek. A ViewModel általában LiveData objektumokat használ, amelyek lehetővé teszik, hogy az adatok automatikusan frissüljenek a View-ban, ha változás történik bennük.

3.3. A program felépítése

A MVVM architektúra kulcsfontosságú szerepet fog betölteni az alkalmazásunkban. Mivel ezen szemlélet alapján kezdem el az alkalmazást, fontos hogy a **mindsk 21** (minimum Operációs rendszer követelmény) legyen beállítva. Ez jelen esetben az Android 5-ös verziót jelenti.

A programban egyszerűségre törekszek, hogy minél szélesebb felhasználó közönség igénybe tudja venni a programot. Fontos, hogy intuitív, és letisztult legyen a kezelői felület, a lehető legkevesebb hibalehetőség merüljön fel a User részéről. Ezért a programom felépítését az alábbiak szerint képzelem el:



3.2. ábra: felhasználói diagram

A felhasználó az applikáció megnyitása után a bejelentkezés kijelzőre lesz irányítva, ahol be tud jelentkezni ideális esetben a MOHOSZ-nál regisztrált adataival. Miután ez megtörtént át lesz irányítva a fogási napló oldalra, ahol látni fogja az összes fogását időrendi sorrendben.

Ezen a ponton tudja feltölteni a legfrissebb fogását a naplóba, ahol a következőket kell megadnia:

- Tó neve
- Tó víztérkódja
- Hal neve
- Hal súlya

Ezen a ponton az alkalmazás automatikusan hozzárendeli a **SystemDate** változót, így elkerülve a különböző csalásokat, későbbi fogás hozzáadásokat a rendszerben. Tehát a felhasználó nem adhatja meg a fogásának az időpontját.

A fogás rögzítése után a rendszer hozzárendeli az adatbázishoz az éppen létrejött adatokat, és sorrend szerint legfelülre rakja. Itt megtekinthetjük a frissen hozzá adott fogást a régebben hozzáadotakkal együtt.

Ezen a ponton felmerülhet egy kérdés, hogy mi a helyzet akkor, hogyha úgy döntött egyik horgásztársunk, hogy a vízpartra mindenféle okos eszköz érkezik meg? Itt jön szóba a fiók oldal, ahol ezt a problémát lenne érdemes orvosolni.

A fogási napló tetején egy **TopBar** változóval hozzárendelünk egy gombot, ami átvezet minket a fiók fülre, ahol megtekinthetjük az éppen bejelentkezett egyént, és felkínáljuk a lehetőséget a kijelentkezésre. Ezen a ponton az alkalmazás vissza vezet minket a bejelentkezés fülre, ahol sport társunk bejelentkezhet saját fiókjával, és nyomon követheti, illetve naplózhatja fogásait.

4. fejezet

Megvalósítás

Ez a fejezet mutatja be a megvalósítás lépéseit. Itt lehet az esetlegesen előforduló technikai nehézségeket említeni. Be lehet már mutatni a program elkészült részeit.

Meg lehet mutatni az elkészített programkód érdekesebb részeit. (Az érdekesebb részek bemutatására kellene szorítkozni. Többségében a szöveges leírásnak kellene benne lennie. Abból lehet kiindulni, hogy a forráskód a dolgozathoz elérhető, azt nem kell magába a dolgozatba bemásolni, elegendő csak behivatkozni.)

A dolgozatban szereplő forráskódrészletekhez külön vannak programnyelvenként stílusok. Python esetében például a 4.1. programkódban látható egy formázott kódrészlet.

Programkód 4.1. Python példa

```
import sys

if __name__ == '__main__':
    pass
```

A stílusfájlok a `styles` jegyzékben találhatók. A stílusok között szerepel még C++, Java és Rust stílusfájl. Ezek használatához a `dolgozat.tex` fájl elején `usepackage` paranccsal hozzá kell adni a stílust, majd a stílusfájl nevével megegyező környezetet lehet használni. További példaként C++ forráskód esetében ez így szerepel.

```
#include <iostream>

class Sample : public Object
{
    // An empty class definition
}
```

Stílusfájlokból elegendő csak annyit meghagyni, amennyire a dolgozatban szükség van. Más, C szintaktikájú nyelvekhez (mint például a JavaScript és C#) a Java vagy C++ stílusfájlok átszerkesztésére van szükség. (Elegendő lehet csak a fájlnevet átírni, és a fájlban a környezet nevét.)

Nyers adatok, parancssori kimenetek megjelenítéséhez a `verbatim` környezetet lehet használni.

```
$ some commands with arguments
1 2 3 4 5
$ _
```

A kutatás jellegű témáknál ez a fejezet gyakorlatilag kimaradhat. Helyette inkább a fő vizsgálati módszerek, kutatási irányok kaphatnak külön-külön fejezeteket.

5. fejezet

Tesztelés

A fejezetben be kell mutatni, hogy az elkészült alkalmazás hogyan használható. (Az, hogy hogyan kell, hogy működjön, és hogy hogy lett elkészítve, az előző fejezetekben már megtörtént.)

Jellemzően az alábbi dolgok kerülhetnek ide.

- Tesztfuttatások. Le lehet írni a futási időket, memória és tárigényt.
- Felhasználói kézikönyv jellegű leírás. Kifejezetten a végfelhasználó szempontjából lehet azt bemutatni, hogy mit hogy lehet majd használni.
- Kutatás kapcsán ide főként táblázatok, görbék és egyéb részletes összesítések kerülhetnek.

6. fejezet

Összefoglalás

Hasonló szerepe van, mint a bevezetésnek. Itt már múltidőben lehet beszélni. A szerző saját meglátása szerint kell összegezni és értékelni a dolgozat fontosabb eredményeit. Meg lehet benne említeni, hogy mi az ami jobban, mi az ami kevésbé jobban sikerült a tervezettnél. El lehet benne mondani, hogy milyen további tervek, fejlesztési lehetőségek vannak még a témával kapcsolatban.

Források

[1] Apple. *Xcode weboldala*. https://developer.apple.com/xcode/.

[2] F-Droid poject. *F-Droid Free and Open Source Android app store*. https://f-droid.org/.

[3] Flutter. *Flutter weboldala*. https://flutter.dev/.

[4] Google. *Android Developer Community*. https://developer.android.com/community.

[5] Google. *Android Open Source Project*. https://source.android.com/.

[6] Google. *Android Studio emulator*. https://developer.android.com/studio.

[7] Google. *Android Studio weboldala*. https://developer.android.com/studio.

[8] Google. *Firebase weboldala*. https://firebase.google.com/.

[9] Gradle Inc. *Gradle weboldala*. https://gradle.org/.

[10] Graham Kendall. *Your Mobile Phone vs. Apollo 11's Guidance Computer*. https://www.realclearscience.com/articles/2019/07/02/your_mobile_phone_vs_apollo_11s_guidance_computer_111026.html. 2021.

[11] JetBrains. *JetBrains intelliJ IDEA*. https://www.jetbrains.com/idea/.

[12] Medve Flóra. *Leading mobile operating systems in Hungary in March 2024, by market share*. https://www.statista.com/statistics/1120778/hungary-mobile-operating-systems-market-share/. 2024.

[13] Meet Zaveri. *What is boilerplate and why do we use it? Necessity of coding style guide*. https://www.freecodecamp.org/news/whats-boilerplate-and-why-do-we-use-it-let-s-check-out-the-coding-style-guide-ac2b6c814ee7/.

[14] Microsoft. *Model-View-ViewModel (MVVM)*. https://learn.microsoft.com/hu-hu/dotnet/architecture/maui/mvvm/.

[15] MOHOSZ. *Elindult a Horgász applikáció: újabb jelentős lépés a horgászbarát ügyintézés és a teljes digitalizáció irányába*. https://nyito.mohosz.hu/index.php/szovetseg/19-kozlemenyek/593-elindult-a-horgasz-applikacio-ujabb-jelentos-lepes-a-horgaszbarat-ugyintezes-es-a-teljes-digitalizacio-iranyaba. 2024.

[16] React. *React Native weboldala*. https://reactnative.dev/.

[17] statcounter. *Mobile Operating System Market Share Hungary*. https://gs.statcounter.com/os-market-share/mobile/hungary. 2024.

[18] Unity. *Unity weboldala*. https://unity.com/.

CD Használati útmutató

Ennek a címe lehet például *A mellékelt CD tartalma* vagy *Adathordozó használati útmutató* is.

Ez jellemzően csak egy fél-egy oldalas leírás. Arra szolgál, hogy ha valaki kézhez kapja a szakdolgozathoz tartozó CD-t, akkor tudja, hogy mi hol van rajta. Jellemzően elég csak felsorolni, hogy milyen jegyzékek vannak, és azokban mi található. Az elkészített programok telepítéséhez, futtatásához tartozó instrukciók kerülhetnek ide.

A CD lemezre mindenképpen rá kell tenni

- a dolgozatot egy `dolgozat.pdf` fájl formájában,
- a LaTeX forráskódját a dolgozatnak,
- az elkészített programot, fontosabb futási eredményeket (például ha kép a kimenet),
- egy útmutatót a CD használatához (ami lehet ez a fejezet külön PDF-be vagy MarkDown fájlként kimentve).