Általános információk

A diplomaterv szerkezete:

1. Diplomaterv feladatkiírás
2. Címoldal
3. Tartalomjegyzék
4. A diplomatervező nyilatkozata az önálló munkáról és az elektronikus adatok kezeléséről
5. Tartalmi összefoglaló magyarul és angolul
6. Bevezetés: a feladat értelmezése, a tervezés célja, a feladat indokoltsága, a diplomaterv felépítésének rövid összefoglalása
7. A feladatkiírás pontosítása és részletes elemzése
8. Előzmények (irodalomkutatás, hasonló alkotások), az ezekből levonható következtetések
9. A tervezés részletes leírása, a döntési lehetőségek értékelése és a választott megoldások indoklása
10. A megtervezett műszaki alkotás értékelése, kritikai elemzése, továbbfejlesztési lehetőségek
11. Esetleges köszönetnyilvánítások
12. Részletesés pontos irodalomjegyzék
13. Függelék(ek)

Felhasználható a következő oldaltól kezdődő Diplomaterv sablon dokumentum tartalma. Ügyeljen a tanszék, a hallgató, a konzulens nevét és a beadás évét jelölő szövegdobozokra, mert azokra külön ki kell adni a frissítést. A mezők tartalma a sablonban a dokumentum adatlapja alapján automatikusan kerül kitöltésre (Fájl/Információ/Tulajdonságok/Speciális tulajdonságok).

A diplomaterv szabványos méretű A4-es lapokra kerüljön. Az oldalak tükörmargóval készüljenek (mindenhol 2.5cm, baloldalon 1cm-es kötéssel). Az alapértelmezett betűkészlet a 12 pontos Times New Roman, másfeles sorközzel.

Minden oldalon - az első négy szerkezeti elem kivételével - szerepelnie kell az oldalszámnak.

A fejezeteket decimális beosztással kell ellátni. Az ábrákat a megfelelő helyre be kell illeszteni, fejezetenként decimális számmal és kifejező címmel kell ellátni. A fejezeteket decimális aláosztással számozzuk, maximálisan 3 aláosztás mélységben (pl. 2.3.4.1.). Az ábrákat, táblázatokat és képleteket célszerű fejezetenként külön számozni (pl. 2.4. ábra, 4.2 táblázat vagy képletnél (3.2)). A fejezetcímeket igazítsuk balra, a normál szövegnél viszont használjunk sorkiegyenlítést. Az ábrákat, táblázatokat és a hozzájuk tartozó címet igazítsuk középre. A cím a jelölt rész alatt helyezkedjen el.

A képeket lehetőleg rajzoló programmal készítsék el, az egyenleteket egyenlet-szerkesztő segítségével írják le.

Az irodalomjegyzék szövegközi hivatkozása történhet a Harvard-rendszerben (a szerző és az évszám megadásával) vagy sorszámozva. A teljes lista névsor szerinti sorrendben a szöveg végén szerepeljen (sorszámozott irodalmi hivatkozások esetén hivatkozási sorrendben). A szakirodalmi források címeit azonban mindig az eredeti nyelven kell megadni, esetleg zárójelben a fordítással. A listában szereplő valamennyi publikációra hivatkozni kell a szövegben. Minden publikáció a szerzők után a következő adatok szerepelnek: folyóirat cikkeknél a pontos cím, a folyóirat címe, évfolyam, szám, oldalszám tól-ig. A folyóirat címeket csak akkor rövidítsük, ha azok nagyon közismertek vagy nagyon hosszúak. Internet hivatkozások megadásakor fontos, hogy az elérési út előtt megadjuk az oldal tulajdonosát és tartalmát (mivel a link egy idő után akár elérhetetlenné is válhat), valamint az elérés időpontját.

Fontos:

* a szakdolgozat készítő/diplomatervező nyilatkozata (a jelen sablonban szereplő szövegtartalommal) kötelező előírás Karunkon, ennek hiányában a szakdolgozat/diplomaterv nem bírálható és nem védhető!
* mind a dolgozat, mind a melléklet maximálisan 15 MB méretű lehet!

Jó munkát, sikeres szakdolgozat készítést ill. diplomatervezést kívánunk!

FELADATKIÍRÁS

A feladatkiírást a **tanszék saját előírása szerint** vagy a tanszéki adminisztrációban lehet átvenni, és a tanszéki pecséttel ellátott, a tanszékvezető által aláírt lapot kell belefűzni a leadott munkába, vagy a tanszékvezető által elektronikusan jóváhagyott feladatkiírást kell a Diplomaterv Portálról letölteni és a leadott munkába belefűzni (ezen oldal HELYETT, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatban már nem kell megismételni a feladatkiírást.



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Konzulens

BUDAPEST, 2023

Tartalomjegyzék

[Összefoglaló 6](#_Toc59896130)

[Abstract 7](#_Toc59896131)

[1 Bevezetés 8](#_Toc59896132)

[1.1 Formázási tudnivalók **Hiba! A könyvjelző nem létezik.**](#_Toc59896133)

[1.1.1 Címsorok 21](#_Toc59896134)

[1.1.2 Képek 21](#_Toc59896135)

[1.1.3 Kódrészletek 21](#_Toc59896136)

[1.1.4 Irodalomjegyzék 21](#_Toc59896137)

[2 Utolsó simítások 22](#_Toc59896138)

[Irodalomjegyzék 23](#_Toc59896139)

[Függelék 24](#_Toc59896140)

Hallgatói nyilatkozat

Alulírott **Balla Péter Keve**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2023. 11. 16.

...…………………………………………….

Balla Pétr Keve

Összefoglaló

Ide jön a ½-1 oldalas magyar nyelvű összefoglaló, melynek szövege a Diplomaterv Portálra külön is feltöltésre kerül.

Abstract

Ide jön a ½-1 oldalas angol nyelvű összefoglaló, amelynek szövege a Diplomaterv Portálra külön is feltöltésre kerül.

# Bevezetés

A következő fejezet pár példán keresztül bemutatja a diplomatervekben és szakdolgozatokban szokásosan előkerülő formázások megvalósítását.

## Motiváció

A dokumentum folyószövegéhez használjuk a **Normál** (angol Word esetén Normal) stílust.

Az én motivációm ezen alkalmazás létrehozásához a pénzügyeim tudatos kezelése iránti elkötelezettségemre vezethető vissza. Az anyagi felelősségvállalás iránti érdeklődésem már fiatal koromban, 8. osztályos, 14 éves koromban kezdődött, amikor megkaptam az első fizetésemet. Ezt az érdeklődést a nagypapámtól örököltem, aki szintén gondosan vezette pénzügyeit. Az első kísérleteim egy egyszerű Excel táblázatban történtek, ahol azonban sajnos elveszett az adataim egy telefonhiba miatt.

Az esetből nem okulva, következő telefonomon ismét Excel táblázatban vezettem pénzügyeimet, de sajnos ez is elveszett, amikor a telefon meghibásodott. Ekkor merült fel bennem az ötlet, hogy a Google táblázatot használjam, így adataim szinkronizálva voltak a Google Drive-fal, és így biztonságban maradtak. Már itt elkezdtem használni funkciókat a könnyebb kezelhetőség érdekében, és terveztem, hogy ha megtanulok programozni, létrehozok egy alkalmazást, ami automatikusan beolvassa és értékeli a pénzügyeimet.

Ezen elképzelésemet az motiválta, hogy nem találtam olyan alkalmazást, amely teljes mértékben testreszabható lenne számomra. Más alkalmazásokban hiányzott a zsebek létrehozásának lehetősége, vagy túl sok olyan felesleges funkció volt, amelyeket nem lehetett kikapcsolni, ami zavaróvá tette a használatukat. Az igényem az volt, hogy egy olyan alkalmazást hozzak létre, amely kizárólag az enyém, minden gomb és funkció pontosan ott, ahol szükséges, és segít a pénzügyi helyzetem átlátásában és kezelésében.

Az egyszerűség elve vezetett, és úgy éreztem, hogy ez egy olyan egyszerű probléma, amit senki sem oldott meg kielégítően. Amikor az egyetem lehetőséget biztosított egy önálló laboratórium tárgy keretében saját projekt végigvitelében, azonnal felhasználtam az alkalmat. Elsőként Android platformon, Kotlin programozási nyelven megalkottam az alkalmazást, majd a szakdolgozatom témájává választottam, átírva azt Flutter keretrendszerre, hogy mind weben, mind mobil eszközökön futtatható legyen. Miután egy félig működő alkalmazással rendelkeztem, az alkotói folyamat számomra nem csak kötelesség, hanem szenvedély lett. Motivációm egyre csak nőtt, és az alkalmazás különböző hasznos funkcióinak tervezése vált az egyik legkedvesebb hobbiommá. Különösen érdekelt a felhasználói felület (UI) tervezése, és elköteleztem magam abban, hogy az alkalmazás rendelkezzen egy ergonomikus és könnyen kezelhető felülettel. A felhasználói élmény tervezése és az ergonómiai szempontok kutatása olyan területté vált, ahol nem csupán technikai, de művészi kreativitásomat is kifejleszthettem, hogy az alkalmazás ne csak hatékony legyen, hanem esztétikailag is kielégítő a felhasználók számára. Habár az esztétika nem állt elől a prioritás listán a fejlesztés során, így azon van még mit javítani.

## A dolgozat felépítése

# Felhasznált technológiák



Az alábbi fejezetben kifejtem, hogy milyen fontosabb technológiákat használtam az alkalmazásom megalkotásakor, valamint leírom, hogy az app milyen területein volt szükség ezekre.

## Firebase Authentication

A legtöbb alkalmazásnak szüksége van a felhasználó személyazonosságának igazolására és az azonosításra. Ez biztosítja az alkalmazás számára a felhasználó adatainak biztonságos tárolását, és lehetővé teszi, hogy a személyre szabott élményt konzisztensen kínálja a felhasználó összes eszközén. A Firebase Authentication lehetőséget biztosít az alkalmazásra való regisztrációra, regisztrációt követő e-mailes megerősítésre, bejelentkezésre, kijelentkezésre, valamint e-mail és jelszó módosításra. A regisztráció és a későbbi bejelentkezés során többféle azonosítási módszer áll rendelkezésre, például e-mail, telefon, Google vagy Facebook alapú azonosítás. Én az alkalmazásomhoz az e-mai címes és jelszavas bejelentkezést választottam. Ennek oka, hogy szükségem volt az alkalmazásban egy egyedi azonosítóra, ami ebben az esetben a bejelentkezett felhasználó e-mail címe.

## Firebase Firestore Database

A Cloud Firestore egy rendkívül alkalmazkodó, könnyen skálázható adatbázis, amelyet a Firebase és a Google Cloud mobil-, web- és szerverfejlesztések során alkalmaznak. Felhőalapú és valós idejű működés jellemzi, azaz adatait szinkronban tartja az adatbázis használói között valós idejű figyelőkön keresztül. Emellett offline támogatást is nyújt, lehetővé téve érzékeny alkalmazások létrehozását, amelyek zökkenőmentesen működnek a hálózati késleltetéstől vagy az internetkapcsolat hiányától függetlenül. Ezen tulajdonságoknak köszönhetően a Cloud Firestore ideális választás az olyan projektek számára, amelyek rugalmasságot, megbízhatóságot és valós idejű adatkezelést igényelnek. A Cloud Firestore NoSQL adatbázis, ami rugalmas dokumentumorientált modellt alkalmaz, nem relációs adatstruktúrával. A NoSQL adatbázisok előnye, hogy alkalmazkodóképesebbek és könnyebben skálázhatók változó adatstruktúrákhoz és nagyobb adatmennyiségekhez. Ez a rugalmasság lehetővé teszi a gyorsabb fejlesztést és a hatékonyabb adatkezelést dinamikus vagy változó adatok esetén.

## SQLite

A Flutter SQLite egy hatékony és könnyen használható adatbáziskezelő megoldás, amelyet a Flutter keretrendszerrel való mobilalkalmazás-fejlesztés során alkalmaznak. A SQLite adatbázis motorra épül, amely egy könnyű, beépített relációs adatbázis, így a Flutter fejlesztőknek lehetőségük van egyszerűen és hatékonyan tárolni, lekérdezni és frissíteni az adatokat az alkalmazásukban. A Flutter SQLite technológia előnyei közé tartozik a platformfüggetlenség, mivel a Flutter maga is egy cross-platform keretrendszer, így az alkalmazások egyszerre futtathatók iOS-en és Androidon. (weben sajnos nem működik) Az SQLite használata lehetővé teszi az adatok helyi tárolását, ami különösen előnyös offline működés vagy adatok gyors elérésének szükségessége esetén. Az én alkalmazásom esetében a Flutter SQLite technológiát kizárólag a mobil Android platformon alkalmazom annak érdekében, hogy az alkalmazás offline módban is hatékonyan működjön. Ennek előnyei közé tartoznak a következők: gyors adatelérés, egyszerű implementáció, offline funkcionalitás.

# Tervezés

Ebben a szakaszban bemutatom az alkalmazás fejlődésének lépéseit az ötleteléstől kezdve egészen az implementációig. Részletezem az eredeti terveimet, amelyeket a szakdolgozatom témájában megvalósítani kívántam, és beszámolok azokról a tervekről, amelyek végül nem kerültek megvalósításra. Emellett részletesen kifejtem, hogy milyen alkalmazásokból merítettem inspirációt, és ezek milyen módon befolyásolták a projektet. Továbbá, részletesen elemzem az alkalmazás architektúráját, kihangsúlyozva azokat a tervezési döntéseket, amelyek a projekt alapját képezik. Ennek során érintem a különböző komponensek közötti kapcsolatokat és az alkalmazás belső struktúráját, hogy átfogó képet adjak az alkalmazás fejlesztési folyamatáról és szervezeti felépítéséről.

## Ötlet

Mindig is lenyűgözött az olyan játékok világa, mint a Clash of Clans, ahol saját virtuális birodalmat építhetsz, gyűjthetsz erőforrásokat és fejleszthetsz. A játékban egyetlen gombnyomással gyűjtheted össze a pénzt (aranyat és elixírt), és aztán saját belátásod szerint fejlesztheted a városodat. Ez a játék nemcsak szórakoztató, hanem tanulságos is volt számomra, hiszen megtanított arra, hogyan kezeljem okosan és tudatosan az erőforrásaimat, különös tekintettel a pénzügyi és energiai erőforrásokra. Azonban a valóságban ez a fajta élvezet sajnos hiányzik, hiszen a valóságban tényleges erőfeszítésre van szükség ahhoz, hogy változás következzen be az életünkben. Nem lehet csupán játékbeli karakterekre bízni azokat a feladatokat, amelyek valódi energiát és elkötelezettséget igényelnek. A virtuális tapasztalataim inspiráltak arra, hogy gondolatban eljátsszam, milyen lenne egy olyan alkalmazás, ami a valóságból merítve lehetőséget adna pénzügyeim okos és átlátható kezelésére. Az ötlet azonban csak akkor kezdett valósággá válni, amikor az egyetemen az önnálló labor keretében lehetőségem adódott a saját projekt tervezésére és végigvitelére. Az alkalmazás tervezésekor még csak Android platformra koncentráltam, de a fejlesztés során felismertem, hogy fontos lenne, ha az alkalmazás minden platformon működne. Így elindultam a cross-platform keretrendszerek kutatása felé, és a szakdolgozatom témáját egy cross-platform alkalmazás kifejlesztésére és optimalizálására tereltem át.

## Előzmény

Miután elkezdtem fizetést kapni 14 éves koromban, azon gondolkoztam, hogyan vezethetném nyomon a pénzügyeimet. Elsőként láttam, aki még tollal és papírral rögzítette a kiadásait és bevételeit és ő volt az első, aki felhívta figyelmemet ennek a fontosságára. Az első kísérleteim során Excel táblázatban próbáltam rendszerezni a pénzügyeimet, azonban az első táblám eltűnt, amikor összetörtem a telefonomat. (mivel csak lokálisan tároltam a készüléken. Később felismertem, hogy más megközelítésre van szükség, és áttértem egy Google Drive táblázatra. Itt függvények segítségével próbáltam egyszerűsíteni a költségvetés vezetését. Ez azért bizonyult jobb megoldásnak, mert ha esetleg megsérült volna a készülékem, a táblázat megmaradt volna a Google Drive felhőszolgáltatásban. Később programozni is megtanultam, és terveim közé tartozott egy olyan program írása, amely képes kiértékelni ezt a táblázatot és lekérdezéseket készíteni belőle. Ezután kínálkozott a lehetőség, hogy egy teljes alkalmazást fejlesszek ki, amelyben nemcsak vezetni lehet a pénzügyeket, de felhő alapú tárolással is rendelkezik, így az adatok mindig elérhetők és lekérdezhetők.

## Legfontosabb funkciók

Az alkalmazás tervezése során kiemelten fontosnak tartottam néhány alapvető funkciónak a megvalósítását. Ezek közé tartozott az egyedi zsebek létrehozása, amelyekben könnyedén nyomon követhető a pénzügyek alakulása, és amelyekből egyszerűen lehet költeni. Emellett kulcsfontosságú volt számomra a lekérdezések lehetősége, amelyek segítségével a felhasználó bármely kiválasztott időintervallumban lekérdezheti az összes kiadást és bevételt. Ezáltal lehetőség nyílik kulcsszavak vagy a tranzakciókhoz tartozó megjegyzések alapján könnyen keresni és adott esetben módosítani.

Az alkalmazás tervezésekor mindenképpen szükségesnek éreztem, hogy ne csak lokálisan tárolja az adatbázist, hanem kihasználja valamilyen felhőszolgáltatás előnyeit. Így a felhasználók számára lehetővé váljon az adatok fel- és letöltése, amely további kényelmet biztosít.

Továbbá, kiemelt figyelmet trveztem fordítani a tartozások egyszerű és okos szétosztásának lehetőségére. Amennyiben valamilyen eseményen, programon sok tartozást kell kezelni, az alkalmazás segítségével egyetlen gombnyomással könnyedén kiszámolható legyen, hogy az egyes személyek mennyit tartoznak, és ki és mennyit fizessen, hogy mindenki megadja és megkapja a neki járó pénzt, mindezt úgy hogy a lehető legkevesebb tranzakció menjen végbe. Ezáltal a pénzmozgás minimálisra csökkenthető, és a folyamat hatékonyabbá tehető.

Végezetül, ami számomra az egyik legfontosabb, az alkalmazás felhasználói élménye érdekében nagy hangsúlyt szerettem volna fektetni egy egyedi, egyszerű, esztétikus és letisztult UI felület kialakítására. Azt vallom, hogy a felhasználóknak az alkalmazás használata során kellemes és zökkenőmentes élményt kell nyújtania, és ennek érdekében igyekeztem az intuitív tervezési elveket követni.

## A projekt megvalósításának folyamata

## Architektúra

Az alkalmazásom fejlesztésénél az MVVM és a Repository mintát/architectúrát követtem.

Az MVVM (Model-View-ViewModel) architektúra főbb céljai:

* Elválasztás: Különválasztja az alkalmazás logikáját, az adatokat és a felhasználói felületet.
* Könnyű karbantarthatóság: Egyszerűsíti a kódbázis karbantarthatóságát és fejlesztését. Könnyebb megtalálni mi hol van.
* Tesztelhetőség: Fokozza a kód tesztelhetőségét.
* Újrafelhasználhatóság: Lehetővé teszi az osztályok és komponensek könnyű újrafelhasználását.

Repository architektúra főbb céljai:

* Adatelérés absztrakciója: Elválasztja az alkalmazás logikáját az adatelérés részleteitől.
* Könnyű cserélhetőség: Lehetővé teszi az adatforrások könnyű cserélhetőségét anélkül, hogy az alkalmazás logikáját érintené.
* Tesztelhetőség: Fokozza az adatelérési réteg tesztelhetőségét.

A képen szöveg, képernyőkép, diagram, sor látható

Automatikusan generált leírás

. ábra MVVM, Repository Architecktúra

**View:**

A View a felhasználói felület megjelenítéséért felel, és fogadja a felhasználói interakciókat. Üzleti logikát nem tartalmaz, csupán a felhasználói felületet reprezentálja és kommunikál a ViewModel réteggel.

**ViewModel:**

A ViewModel az üzleti logikát közvetíti a View és a Model rétegek között. Transformálja az adatokat, amelyeket a View kéri vagy amelyeket a Model réteg szolgáltat. Kapcsolódik a View-hoz és a Modelhez is.

**Model:**

A Model réteg az alkalmazás üzleti logikájáért felel, beleértve az adatmanipulációkat és adatkezelést. Kommunikál a ViewModel réteggel, és tartalmazza a Repository réteget, amely az adatelérés absztrakcióját biztosítja.

**Repository**:

A Repository réteg az adatelérés absztrakcióját jelenti, elválasztva az alkalmazás üzleti logikáját az adatelérési részletektől. Kommunikál a lokális (SQLite) és távoli (Firebase Database) adatforrásokkal, és biztosítja az adatokat a ViewModel számára.

**Local Data Source (SQLite):**

A Lokális Adatforrás (SQLite) a Repository réteggel kommunikál, felelős a lokális adatok tárolásáért és lekérdezéséért, például egy SQLite adatbázis használatával.

**Remote Data Source (Firebase Database):**

A Távoli Adatforrás (Firebase Database) a Repository réteggel kommunikál, és felelős a távoli adatok lekérdezéséért és frissítéséért, jelen esetben egy Firebase adatbázis használatával.

## Mit lehet még fejleszteni

# A feladat megoldása

## A program specifikációja

## A program architektúrája

## Logikai felépítés

## A program működése

# Végeredmény

## Értékelés

## Továbbfejlesztési lehetőségek

# Összegzés

# Köszönetnyilvánítás

# Irodalomjegyzék

### Címsorok

A fejezetcímek esetén a **Címsor 1-4** (Heading 1-4) stílusokat használjuk.

### Képek

A képhez használjuk a **Kép** stílust.

Képaláírást a képen jobb gombbal kattintva a Képaláírás beszúrása… opcióval adhatunk hozzá, így az automatikusan **Képaláírás** (Caption) stílusú lesz.



.. ábra: Példa képaláírásra

### Kódrészletek

Kódrészletek beillesztése esetén használjuk a **Kód** stílust.

using System;

namespace MyApp

{

class Program

{

static void Main( string[] args )

{

Console.WriteLine( "Szia Világ!" );

}

}

}

### Irodalomjegyzék

Az Irodalomjegyzékben szereplő hivatkozásokat **Irodalomjegyzék sor** stílussal formázzuk, a címüket pedig **Irodalomjegyzék forrás** stílussal emeljük ki.

A szövegbe a hivatkozásokat a Kereszthivatkozás beszúrása (Insert cross-reference) funkcióval helyezzük el (példa egy így beszúrt hivatkozásra: [1]), így azok automatikusan frissülnek a hivatkozások átrendezésekor.

# Utolsó simítások

Miután elkészültünk a dokumentációval, ne felejtsük el a következő lépéseket:

* Kereszthivatkozások frissítése: miután kijelöltük a teljes szöveget (Ctrl+A), nyomjuk meg az F9 billentyűt, és a Word frissíti az összes kereszthivatkozást. Ilyenkor ellenőrizzük, hogy nem jelent-e meg valahol a "Hiba! A könyvjelző nem létezik." szöveg.
* Dokumentum tulajdonságok megadása: a dokumentumhoz tartozó meta adatok kitöltése (szerző, cím, kulcsszavak stb.). Erre való a Dokumentum tulajdonságai panel, mely a Fájl / Információ / Tulajdonságok / Dokumentumpanel megjelenítése úton érhető el.
* Kinézet ellenőrzése PDF-ben: a legjobb teszt a végén, ha PDF-et készítünk a dokumentumból, és azt leellenőrizzük.

Irodalomjegyzék

1. Levendovszky, J., Jereb, L., Elek, Zs., Vesztergombi, Gy.: Adaptive statistical algorithms in network reliability analysis, Performance Evaluation - Elsevier, Vol. 48, 2002, pp. 225-236
2. National Istruments: LabVIEW grafikus fejlesztői környezet leírása, <http://www.ni.com/> (2010. nov.)
3. Fowler, M.: UML Distilled, 3rd edition, ISBN 0-321-19368-7, Addison-Wesley, 2004
4. Wikipedia: Evaluation strategy, <http://en.wikipedia.org/wiki/Evaluation_strategy> (revision 18:11, 31 July 2012)
5. Firebase Authentication <https://firebase.google.com/docs/auth>
6. Firebase Firestore Database <https://firebase.google.com/docs/firestore>
7. Flutter SQLite <https://docs.flutter.dev/>

Függelék