



# SZAKDOLGOZAT

*MoneyWatch: Pénzügyi tudatosságot segítő mobilalkalmazás*

Készítette: Kun Máté

Gazdaságinformatikus

Témavezető: Körei Attila egyetemi docens

Konzulens: Piller Imre egyetemi tanársegéd

Miskolci Egyetem

2023

**MISKOLCI EGYETEM**

Gépészmérnöki és Informatikai Kar  
Analízis Intézeti Tanszék

**Sorszám: GEMAN/O4FRG0/2023/BSc**

## **SZAKDOLGOZATI FELADAT**

**Kun Máté**

gazdaságinformatikus jelölt  
részére

**A szakdolgozat tárgyköre:** mobilalkalmazás fejlesztés

**A szakdolgozat címe:**

MoneyWatch: Pénzügyi tudatosságot segítő mobilalkalmazás

**A feladat részletezése:**

A pénzzel való tudatos bánásmód, a takarékoskodás nagyon fontos. A szakdolgozat célja, hogy bemutassa egy ezt segítő mobilalkalmazásnak a megtervezését és kifejlesztését. A rendszer felhasználói az alkalmazás segítségével adott célokat tűzhetnek ki maguk elé, melyek teljesülését nyomon tudják követni, amely motivációt jelent ahhoz, hogy el tudják azokat érni. Az alkalmazás a felhasználók számára utat mutat a pénzügyi tudatosság felé, jobban vizualizálva a felhasználó költségeit.

A dolgozat bemutatja, hogy jelenleg milyen alkalmazások állnak rendelkezésre, amelyek hasonlóképpen segíteni próbálják a felhasználókat a pénzügyekhez kapcsolódó céljaik elérésében. Részletesen specifikálja az elkészítendő alkalmazás felépítését, működését. Bemutatja a tervezés menetét, az előforduló döntési pontokat, az előforduló alternatívákat.

Az alkalmazás egy Android platformon futtatható mobil alkalmazásként kerül megvalósításra Kotlin programozási nyelven.

**Témavezető(k):** Dr. Körei Attila, egyetemi docens

**A feladat kiadásának ideje:** 2023. március 7.

.....  
Dr. Szigeti Jenő  
szakfelelős

1.

szükséges.

A szakdolgozat feladat módosítása

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.

bocsátható.

A szakdolgozat bírálatra

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

## EREDETISÉGI NYILATKOZAT

Alulírott Kun Máté; Neptun-kód: O4FRG0

a Miskolci Egyetem Gépészmérnöki és Informatikai Karának végzős,

Gazdaságinformatikus szakos hallgatója ezennel büntetőjogi és fegyelmi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy

MoneyWatch: Pénzügyi tudatosságot segítő mobilalkalmazás

című szakdolgozatom/diplomatervem 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

# Tartalomjegyzék

1. Bevezetés.....	6
2. A pénzügyi szoftverek .....	8
2.1. Személyes pénzügyi asszisztens szoftverek.....	8
3. A MoneyWatch felhasználói útmutatója.....	13
3.1. Az alkalmazás bemutatása, célja .....	13
3.2. Letöltés utáni teendők.....	13
3.3. A főegyenleg kezelése.....	15
3.4. Egy adott költség rögzítése .....	15
3.5. Megtakarítási célok beállítása .....	16
3.6. Új megtakarítási cél létrehozása.....	16
3.7. Határidős megtakarítási cél hozzáadása.....	17
3.8. Határidő nélküli megtakarítási cél létrehozása.....	18
3.9. A Bevásárlás asszisztens működése.....	18
4. A fejlesztőkörnyezet kiválasztása .....	20
4.1. A kiválasztott programozási nyelv.....	20
4.2. A kiválasztott fejlesztőkörnyezet.....	21
4.3. A Kotlin nyelv szintaktikája .....	21
5. Az alkalmazás implementálása .....	25
5.1. Az alkalmazás lapátmenete .....	25
5.2. Az alkalmazás lapjainak bemutatása .....	26
5.2.2. A főegyenleg képernyő.....	26
5.2.2. A megtakarítási felület.....	31
6. Az elkészült alkalmazás tesztelése és tapasztalatok.....	37
7. További fejlesztési javaslatok .....	38
7.1. Dátumok implementálása minden felületre .....	38
7.2. A határidők és az ismétlődő tranzakciók implementálása.....	38
7.3. Több ösztönzés a megtakarításra.....	39
7.4. Több fül a bevásárlás asszisztensen belül .....	39
Köszönetnyilvánítás .....	39
Összegzés .....	40
Summary .....	42
Forrásmegjelölés.....	44

## 1. Bevezetés

A pénzügyi világban mindenki részt vesz. Van, aki könnyedén átlátja a saját pénzügyeit, de akadnak olyan személyek is, akiknek sokszor az az érzésük, hogy kifolyik a kezükből a pénz. Ekkor gondoltam bele abba, hogy nagyon sokan járnak hasonló helyzetben, mikor hosszútávú elképzeléseik vannak, de nehezükre esik a spórolás, a megtakarítások teljesítése. Sajnos erre a problémára nem mindig található teljes megoldás, de ha kellőképpen figyelünk ezekre a dolgokra, akkor képesek lehetünk nagyban befolyásolni a pénzügyi döntéseinket, amivel az anyagi jólétünk nagymértékben nőhet. Éppen ezért szeretnék a szakdolgozati témámban egy olyan témával foglalkozni, ami egy gyakran előforduló akadály. A megoldásom segítségével képesek lehetünk egyszerűbben elérni a nagy álmainkat. Természetesen nem csak a rossz pénzügyi döntések miatt történhet meg az olyan eset, hogy valakinek nem sikerül a céljait teljesíteni, a tervezett téma egy aspektusból nyújtana megoldást. Leginkább olyan személyeknek nyújtana a dolgozat témája megoldást, akik sokat vásárolnak impulzusszerűen, illetve bevásárlólista nélkül, olyanoknak, akik sokszor érzik azt, hogy nem tudják követni, mivé válik nehezen megkeresett egyenlegük, aki sokszor azt érzi, hogy nem tudja, mit is akar venni pontosan, aminek a végeredménye az, hogy rengeteg olyan dolgot is megvásárol, amire később nem is lesz szüksége. Az üzletek minden trükköt megpróbálnak alkalmazni annak érdekében, hogy minél nagyobb összeget hagyjunk ott a pénztárnál. Azonban, ha pénzügyi tudatossággal állunk minden egyes költéshez, nagyban meggátolhatjuk a felesleges vásárlások mennyiségét. Éppen ezért szeretnék egy olyan megoldást találni, ami sokkal hatékonyabb alternatívát jelent, minthogy papíron vezessük végig minden egyes vásárlásunkat, amiben előbb-utóbb elvesznek a részletek (előfordulhat akár olyan helyzet is, hogy ezeket észre sem vesszük). Megoldást nyújt vásárlás közben is, hogy ne csak az éppen összeírt bevásárlólista legyen a kezünkben, de még a jövőre nézve is, a nagyobb mérföldkövek elérését is támogatni tudja a program. A dolgozatomban megvizsgálom a piacon jelenleg elérhető pénzügyi asszisztensek lehetőségeit, opcióit, funkcionalitásait, mérlegelem az ilyen alkalmazások előnyeit, illetve hátrányait, majd az elérhető telefonos platformra készült programozási nyelvek közül kiválasztom a megfelelőt.

Ezek után az addigi következtetéseket levonva, a saját ötletem alapján implementálom a saját pénzügyi asszisztensem, amit a dolgozat írásakor említett programok problémáit szeretné minimalizálni. Olyan alkalmazást szeretnék létrehozni, ami sokak számára egyszerűen elérhető és használható. A fejlesztés során a fő szempont az egyszerűség, annak érdekében, hogy gyorsan tudjuk az előforduló költségeinket rögzíteni.

## **2. A pénzügyi szoftverek**

Manapság már rengeteg alkalmazást megtalálhatunk, amivel az életszínvonalunkat javítani tudjuk, legyen szó szórakozásról, sportról, utazásról, közösségi szolgáltatásokról stb. Ilyen alkalmazások a pénzügyi szoftverek is. Ezek az alkalmazások a pénzügyi gondjainkat hivatottak csökkenteni, mivel sokan bele szoktak esni abba a csapdába, hogy valamiért nehezükre esik a pénzt beosztani, így egy útmutatót kaphatnak a fogyasztók a pénzügyi világban. Ebben a fejezetben a pénzügyi alkalmazásokat, és annak lehetőségeit vizsgálom meg.

### **2.1. Személyes pénzügyi asszisztens szoftverek**

Manapság már rengeteg problémára képesek lehetünk egy erre a célra készített alkalmazást találni, legyen az játék, sport, étkezés, tanulás, hírek, vagy bármilyen egyéb tevékenység. Ennek következtében a pénzügyi paletta sem üres, sokféle alkalmazás tárul elénk, mikor egy alkalmazást szeretnénk találni a pénzügyi tevékenységünk monitorozásához. Ezen alkalmazásoknak az a célja, hogy a felhasználó számára egy átláthatóbb és megfoghatóbb költési képet tudjon nyújtani, amivel át tudjuk gondolni, hogy egy adott időszakban mi az, amire költünk, és szemünk elé kerülhetnek olyan dolgok is, amire nem feltétlenül lenne szükséges valamekkora összeget kiadni. Néhány program még esetlegesen képes egyéb funkcionalitást is nyújtani, akár egy megtakarítási portfóliót is képesek lehetünk kezelni a meglévő funkcionalitások mellett. Nagyon fontos azonban megkülönböztetni a különböző platformokat, amik a rendelkezésünkre állnak. Vegyük elsőként a számítógépre készült alkalmazásokat. Ilyenek lehetnek például a GnuCash, a HomeBank [1] alkalmazások. „A könnyen használható, nagy tudású és rugalmas GnuCash lehetővé teszi a bankszámlák, részvények, bevételek és kiadások nyomon követését. Ugyanolyan gyors és egyszerű, mintha füzetben vezetnénk a nyilvántartásunkat, mégis professzionális számviteli elveken alapul, biztosítva ezzel a mérleg és a kimutatások pontosságát.” [2] Előnye lehet, hogy mivel ezek a programok számítógépekre készülnek, így nagyobb felület áll a rendelkezésünkre, mint egy telefonnál, ezért a vizualizáció sokkal erősebb eszközzé válhat. Ezekben a programokban tudunk számlákat felállítani, költést és jövedelmet rögzíteni,



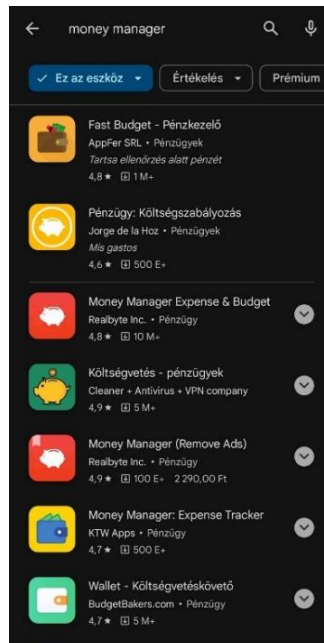
esetlegesen ismétlődő szolgáltatásokat is tudunk benne rögzíteni, így jobban láthatjuk, hogy egy hónapban milyen alapvető dolgaink vannak, amire költünk, és ezáltal jobban tudjuk mérlegelni, hogy mire is van szükségünk. A számítógépes platform hátránya az, hogy minden alkalommal, mikor valamit rögzíteni vagy ellenőrizni szeretnénk, akkor bizony a számítógépet kell elővenni, ami nem feltétlen jelent egy hordozható eszközt (például laptopot), így nagyobbá válik annak a valószínűsége, hogy egy idő után a felhasználó megfedkezik a program használatáról, így elesik a program hasznos részétől. Azonban, ha rendeltetésszerűen kerül használatra a program, akkor egy teljesen részletes pénzügyi portfólió tárul elénk. Ezen alkalmazások más egyébre nem hivatottak működni, nincs bennük megtakarítási cél, sem bevásárlási tervező sem. Manapság már csak akkor van lehetőségünk számítógépen ilyen programot találni, ha kifejezetten keressük az efféle alkalmazásokat. A vállalkozásoknak, illetve akiknek nagyon fontos és komplex pénzügyi rendszert kell vezetniük, annak a számítógépes platform a megfelelő választás. Személy szerint én nem használtam számítógépes költségkezelő alkalmazást, ennek okát a többi mód összehasonlításában kell keresni.



2.1 ábra: A GnuCash sokszínű felülete.

Térjünk át a mobilos eszközökre. A mai világban már sokan elképzelni sem tudják a világot telefon nélkül, hatalmas teret hódított magának az eszköz, mivel nagyon egyszerű a használata, és sok esetben megkönnyítheti az életünket. Olyan fejletté váltak ezek a mobilok, hogy már-már teljesen beépülnek az emberek életébe. A telefonos rendszereken temérdek alkalmazás áll a rendelkezésünkre a rendszer

„áruházában”, ahonnan telepíteni tudjuk a nekünk tetsző és szükséges alkalmazásokat. Így tehát nem meglepő, hogy több opció közül választhatunk, ha a pénzügyi asszisztenset keresünk:

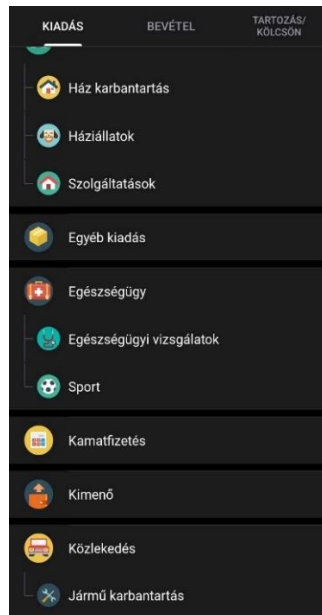


2.1 ábra: Általános pénzügykezelő alkalmazások.

Ezek a programok általában annyit tudnak nyújtani a felhasználó számára, hogy egyenlege betáplálása után rögzítheti a költségeit, majd az alkalmazás elkészít a számára különböző ábrákat hasznos adatokkal, például, hogy mennyit költött el a havi egyenlegéből, kategóriánként mennyi a költségek stb. Ha valaki nem látja át, hogy egy hónapban mire költ, akkor egy ilyen alkalmazás nagyon hasznos lehet a számára. Egy másik előnye a mobilos alkalmazásoknak, az az, hogy mivel a mobil eszközünk a nap nagy részében nálunk van, így sokkal egyszerűbb a költségeket rögzíteni, mint egy asztali rendszert használva, hiszen a felhasználónak csak elő kell vennie a telefonon a pénzügyi kezelőjét, ezután rögzítheti is a tranzakcióit. ez a mai világban egy nyomós érv, mivel a legtöbb ember a gyorsaságot és az egyszerűséget szereti az életben. Emiatt az ok miatt erre a problémára a mobilos megoldást tekinthetjük a legjobb alternatívának.

Példaként vizsgáljuk meg a Money Lover [3] nevű alkalmazást. Ebben az alkalmazásban rögzíthetjük a költségeinket kategória szerint, illetve meg is szabhatjuk, hogy az adott hónapban mennyit szeretnénk az adott kategóriára fordítani.

Hátrányként tapasztalható, hogy mivel sok kategória került implementálásra, így a rögzítési folyamat nehezebbé válik, nehezkessé, hosszassá válik egy-egy költség rögzítése. A rengeteg kialakított kategória miatt még az első beállítás is egy hosszas folyamattá válik, mivel kategóriánként adhatjuk meg azt, hogy miből mennyit szeretnénk keretként alkalmazni.



2.2 ábra: A Money Lover alkalmazásban rengeteg kategória közül választhatunk.

Egy másik alkalmazás, ami pénzügyi nyilvántartást vezet, az az OTP Bank mobilapplikációja. Aki ennél a banknál rendelkezik számlával, az gond nélkül követheti a számláján történő eseményeket. A program végez költségkimutatást csoportosítva, megmutatja, hogy mennyit költöttünk a hónapban, illetve visszanezézhetjük a korábbi számlatörténetet is. 2023. augusztusában került bele a Persely nevezetű újítás is, amivel már meghatározhatunk egy célt is, amire gyűjteni szeretnénk, és az erre fordított összeget bele tudjuk helyezni a megnyitott perselybe. Hátránya viszont, hogy ebben a programban nincs beleépítve a tudatossági rész, hiszen csak összegez, tehát itt nem olyan számottevően látszódik, hogy mire mennyit költünk el egy hónapban, mint egy pénzügyi asszisztenst használva.



2.3 ábra: Az OTP Bank persely funkciója egy ígéretes megtakarítási eszköz.

Elmondható tehát, hogy az ilyen típusú programok általában csak azt hivatottak elérni, hogy egy képet kapjunk a költségeinkről, más opciót azonban nem, vagy csak ritkán találhatunk meg. A saját ötletem egy olyan alkalmazás implementálása, amely a költség megfigyelésen kívül a megtakarítási célokra is fókuszál, illetve, ha vásárlunk, egy kényelmi funkciót is igénybe tudjunk venni, amivel a vásárlás során is végig nyitott szemmel tud majd járni a használója, mivel a kosár értékét meg tudja vele figyelni.

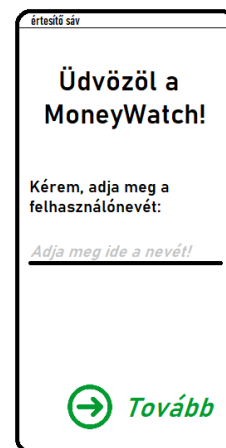
### 3. A MoneyWatch felhasználói útmutatója

Az alábbi fejezetben az alkalmazás számára készült felhasználói útmutató kerül bemutatásra, ami bemutatja a program funkcionalitásait, a megtervezett programot, és végig vezeti a felhasználót a program működésében.

#### 3.1. Az alkalmazás bemutatása, célja

Az alkalmazás célja a felhasználó számára a pénzügyi asszisztens nyújtása. Az alkalmazás segítségével a felhasználó számára jobban átláthatóvá válnak a kiadások, illetve megkönnyíti/segíti a megtakarításokat teljesíteni.

Az alkalmazás nagy segítséget nyújthat továbbá a bevásárlás során a költségek megfigyelésére, hiszen beépített bevásárlólistával is rendelkezik, ami reszponzív adatokat tud nyújtani használat közben.



3.1 ábra: A megnyitás után megjelenő felület.

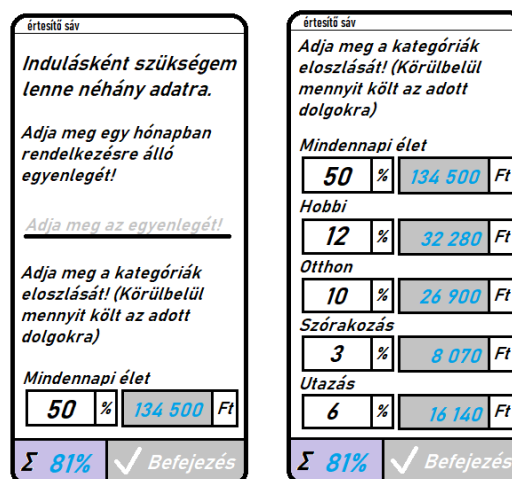
#### 3.2. Letöltés utáni teendők

Az alkalmazás kér egy *felhasználónevet*, hogy a későbbiekben meg tudja majd szólítani a használat. A felhasználónév megadása után megkérdezi az alkalmazás, hogy egy hónapban mekkora összeg áll a rendelkezésére (az alkalmazás minden adatot bizalmasan kezel, az adatokat nem továbbítja), illetve megkérdezi azt is, hogy körülbelül mekkora az az összeg, amit egy hónapban meg szeretne takarítani.

Ezután az alkalmazáson belül kezelt kategóriák arányairól érdeklődik az alkalmazás.

Hogy milyen kategóriák kerüljenek a programba,

az egy nagyon fontos része az egész alkalmazásnak. Ez szerintem egy lényeges része a fejlesztési folyamatnak, mivel sok alkalmazás itt rontja el az egyszerűségi oldalát az egész alkalmazásnak, holott egy ilyen programnál nagyon fontos az egyszerűség főleg



3.2 ábra: A kategóriák arányainak rögzítésére szolgáló oldal.

azért, mert így kevésbé merül feledésbe az adatok rögzítése. Éppen ezért a kategóriákat az OTP Bank alkalmazása és a már elérhető pénzügyi asszisztensek alapján próbáltam meg kialakítani. Az alkalmazásban jelenleg megadott kategóriák a következők:

- *Mindennapi élet* (ide sorolhatóak a következő tárgyak: Étkezés, ruházkodás, havidíjak, szolgáltatások)
- *Hobbi* (A felhasználó számára érdekes tevékenységeinek költségei, ilyen lehet például a kerékpározás)
- *Otthon* (Ritkán előforduló nagyobb tételű vásárlások. Ide sorolhatóak a konyhai eszközök megvásárlása, rezsiköltség, bármilyen felújítás.)
- *Szórakozás* (Házon kívüli étkezés, mozi, szórakozóhely, házibuli és ezekhez hasonló tevékenységek kategóriája)
- *Utazás* (Tömegközlekedés, gépjármű költségei, üzemanyag, szállás finanszírozása)
- *Megtakarítás* (Minden olyan tranzakció, ami a nagyobb céljaink elérését teszi lehetővé, a *megtakarítások* menüpontban van nagy szerepe, mivel az ilyen kategóriájú tranzakciókkal tudunk majd megtakarítást rögzíteni a céljaink gyűjtéséhez.)

Az alkalmazás letöltése és megnyitása után megjelenik a fő képernyő, itt látható az aktuális egyenleg, és néhány egyéb információ, amit az alkalmazás magától kiszámol, például azt, hogy egy napra mennyi kiadás jut, a különböző kategóriákat és az arra költött pénzmennyiséget.

Gyorsgombként rendelkezésre áll a *tétel rögzítés*, illetve a főbb menüpontok között is lehetőségünk van választani (ezek az összes menüponton elérhetőek, váltogathatóak).

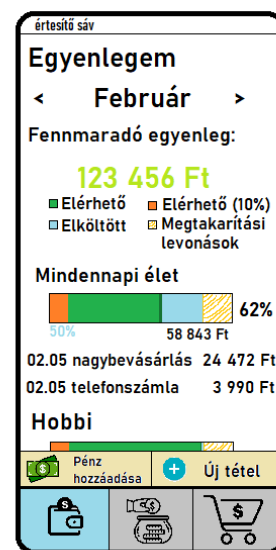
### 3.3. A főegyenleg kezelése

Ebben a menüpontban jelennek meg a naplózott tételek, illetve az egyenleg állapota is itt látható részletesen lebontva.

Ebben a menüpontban is lehetőség nyílik kiadást rögzíteni ugyanolyan módon, mint a főmenünél.

Az alkalmazás a kategóriák szerint naplózza a kiadásokat, így vissza lehet benne keresni, hogy mikor mire költöttünk épp, illetve a havi egyenlegi lebontást is szemlélteti a felhasználónak.

Ha új bevételi forrásunk nyílik, akkor ennél a pontnál szükséges hozzáadni a mennyiséget, ami hasonló módon működik, mint a tétel hozzáadás, de az ilyen tranzakció pozitív irányba mozdítja az egyenleget.



3.3 ábra: Egyenleg felület.

3.4 ábra: A tranzakciók rögzítésére szolgáló lap.

### 3.4. Egy adott költség rögzítése

Nyomjunk rá az *Új tétel* pontra!

Ekkor megjelenik a rögzítés opció az alkalmazásban, itt a következő információkat kell megadnunk:

- A tétel nevét, hogy később emlékeztessen, mire költöttünk ennél a tranzakciónál,
- A tétel összegét,
- A tétel kategóriáját, amit a programban megadott kategóriák közül lehet kiválasztani.
- Esetlegesen egy leírást is hozzáadhatunk, ha a névnél hosszabb szöveget szeretnénk megadni.

A szükséges információk után a rögzítés pontot megnyomva az applikáció automatikusan hozzáadja a kiválasztott kategóriához a megadott tételt, majd levonja az összeget az egyenlegből. Ha mégsem szeretnénk tranzakciót rögzíteni, akkor a mégse gomb megnyomásával visszatérhetünk az előző fülre.

### 3.5. Megtakarítási célok beállítása

A program egyik funkciója, hogy be lehet állítani rajta olyan célokat, amire össze szeretnénk gyűjteni az összeget, és az alkalmazás segíti a felhasználót az úton ahhoz, hogy el tudja érni a megálmodott összeget.

A  *megtakarítások*  gombra nyomva megjelennek az eddig megadott célok, illetve a célokra összegyűlt egyenleg is.

Az alkalmazásban bármennyi célt meg lehet adni, de azért, hogy ne a végtelenségig gyűljön úgy a pénz, hogy semmi nem teljesül, ezért a program egyenleg alapon számítja a megtakarításokat. Ez azt jelenti, hogy a céloknál meg kell határozni egy prioritást, majd az alkalmazás mindig a legnagyobb prioritású dolgokhoz társítja a már megtakarított pénzmennyiséget.

Ha kigyűjtöttük a kívánt célt, akkor az *elköltöm* gombbal teljesíthetjük azt, így az eltűnik, majd a cél összege levonódik a megtakarítási egyenlegből.



3.5 ábra: A megtakarítási lap.

### 3.6. Új megtakarítási cél létrehozása

Az alkalmazás kétféle célt különböztet meg:

- *Határidős cél* (ilyen lehet például a nyaralás, utazás, gyűjtés egy eseményre)
- *határidő nélküli cél* (bármilyen cél, amihez nem lehet konkrét határidőt rendelni, ilyen lehet a lakásmegtakarítás)

Fontos információ: A két típus különböző módon viselkedik, ezért körültekintően válasszunk! Míg a határidős cél során kiszámításra kerül, hogy milyen ütemben kell megtakarítani ahhoz, hogy a cél időben teljesülhessen, addig a határidőt nem tartalmazó célnál a felhasználóra van bízva, hogy milyen ütemben takarítja meg a kívánt összeget a célja eléréséhez.



### 3.7. Határidős megtakarítási cél hozzáadása

Nyomjunk rá az *Új tétel* pontra!

Ekkor megjelenik a *cél rögzítés* menüpont az alkalmazásban.

Ezek után a következő adatokat szükséges megadni:

- Cél neve,
- Cél összege forintban,
- Ebben az esetben a határidőt is meg kell

határozni, amit hónapokban lehet megtenni.

Elfogadás után az alkalmazás hozzáadja a célokhoz a megadott adatokat, illetve az elérhető havi egyenleget az alkalmazás csökkenti a cél ütemével (tegyük fel, hogy 250 ezer forint a havi keret, de ha egy olyan cél kerül rögzítésre, aminek a teljesítéséhez havonta 30 ezer forintot kell megtakarítani, a program csak 220 ezer forintnyi kerettel fog számolni, mivel ilyen módon érhető el időben a megadott cél).

Ha a határidő eléréséig minden rendben történt, akkor a program tájékoztatja a felhasználót arról, hogy a határidő lejárt, és a cél teljesült, ettől az időponttól nem fog többször áthelyezésre kerülni az eddig levont összeg. Ekkor meg kell erősíteni a célkitűzés teljesülését, majd pozitív visszajelzés után az alkalmazás eltávolítja ezt az aktív célok közül.

A cél esetleges törlése esetén a célra eddig megtakarított (levont) egyenleget a megtakarítás számlára helyezi vissza.

Fontos információ: Ahhoz, hogy a gyűjtés ne váljon teljesíthetetlenné, ha a rendelkezésre álló egyenleg 20%-a határidős célra fordul, akkor az alkalmazás figyelmezteti a felhasználót, hogy nagy arányú megtakarításokat szeretne, aminek a gyűjtése nehezen teljesíthetővé válhat.

3.6 ábra: Határidős megtakarítás rögzítése.

### 3.8. Határidő nélküli megtakarítási cél létrehozása

Nyomjunk rá az *Új tétel* pontra!

Ekkor megjelenik a *cél rögzítés* opció az alkalmazásban, itt válasszuk ki a *határidő nélküli* célt!

Ezek után a következő adatokat szükséges megadni:

- Cél neve,
- Cél összege forintban.

Az ilyen típusú tételeket a megtakarított egyenleg segítségével tudjuk összegyűjteni. Ehhez az egyenleghez a *Megtakarítások* fülnél az egyenleg alatti pontnál tudunk összeget adni a rendelkezésre álló főegyenlegből (illetve pénzhez is tudunk jutni a megtakarítási egyenlegből, bár az alkalmazás célja nem ez lenne.)

3.7 ábra: Cél hozzáadása határidő nélkül.

### 3.9. A Bevásárlás asszisztens működése

Az egyik legnagyobb probléma, ami bevásárlás során előfordulhat, az az, hogy még ha listát is készítünk, fizetéskor nem tudjuk mennyit költünk majd. Ennek kiküszöbölésére áll rendelkezésünkre a *bevásárlás asszisztens*, amiben a bevásárlólista mellett az eddig költött összeget is tudjuk figyelni.

Működése:

- Írjuk össze a bevásárolni kívánt dolgokat, készítsünk bevásárlólistát! Új tárgyat az *Új tétel* gomb megnyomásával tudunk létrehozni.
- Adjuk meg a megvásárolni kívánt tárgy nevét, illetve azt is, hogy mennyit szeretnénk az adott dologból vásárolni.
- Ha meggondoltuk magunkat egy bizonyos termékkel kapcsolatban, akkor az „X” gomb megnyomásával eltávolíthatjuk a listából.
- A bevásárlólistát bármikor össze lehet írni, ha újra ide lépünk, akkor is a legutóbbi lista fog megjelenni.

3.8 ábra: A bevásárlólista működése.

A bevásárlás közben megadhatjuk az alkalmazásban, hogy az adott terméknek mennyi az egységára! Ezekkel az adatokkal az alkalmazás automatikusan kiszámolja a termék árát, illetve a képernyő alján folyamatosan számolja az eddigi összes kiadást.

A *bevásárlás véglegesítése* gombbal hozzá tudjuk adni az *egyenlegfigyelő*höz is, így nem kell külön tételt hozzáadni. Ekkor megjelenik előttünk a *tétel rögzítő* felület, ekkor már csak a kategóriát és a tárgy címét kell ellenőrizni (Ennek megtétele után törlődik a bevásárlólista.)

## 4. A fejlesztőkörnyezet kiválasztása

A fejlesztés során nagyon fontos elem, hogy kiválasszuk, milyen platformra és milyen nyelven szeretnénk az alkalmazást fejleszteni. Ebben a fejezetben a választott programozási nyelv és a fejlesztőkörnyezet kerül bemutatásra.

### 4.1. A kiválasztott programozási nyelv

Mivel egy telefonos alkalmazást szeretnék programozni, így egy telefonnal kompatibilis programozási nyelvet szeretnék választani. Itt szóba jöhet a Flutter, amit bármilyen eszköz fejlesztéséhez használhatunk, és a Kotlin programozási nyelv, ami az Android hivatalosan használt nyelve a Google által. Mivel az eddigi összes tapasztalatom az Android rendszerrel kapcsolatban áll fenn, így a Kotlin nyelvre esett a választásom. A Kotlin egy modern programozási nyelv, amelyet a professzionális Android-fejlesztők több mint 60%-a használ, és amely segít a termelékenység, a fejlesztői elégedettség és a kódbiztonság növelésében. [5] A Kotlin programozási nyelvet eredetileg a Java programozási nyelv továbbfejlesztésére tervezték, és gyakran használják a Javával együtt. Annak ellenére, hogy a Kotlin az Android preferált fejlesztői nyelve, a Java-val való átjárhatósága miatt számos alkalmazástípushoz használják. A nyelvnek számos előnye létezik, a fontosabbak ezek közül a következők:

- **Átjárhatóság.** A Kotlin átjárható a Javával, mivel ugyanazt a kódot fordítja le. A Kotlin lefordítható Java kódra, ami lehetővé teszi a programozók számára, hogy just-in-time fordítást végezzenek annak érdekében, hogy a másik programba ágyazott kód zökkenőmentesen fusson. A Java-val közös eszközrendszereket is használ. Ezek a funkciók megkönnyítik a Java-alkalmazások Kotlinra való áttelepítését.

- **Biztonság.** A Kotlin úgy tervezték, hogy segítsen elkerülni a gyakori kódolási hibákat, amelyek tönkreteszhetik a kódot, vagy sebezhetőséget hagyhatnak benne. A nyelv rendelkezik nullbiztonsággal és a null-pointer hibák kiküszöbölésével.

- **Egyértelműség.** A Kotlin kiküszöböli a népszerű nyelvek, például a Java alapszintaktikájának néhány redundanciáját. A Kotlin időt takarít meg a fejlesztők számára, mivel tömörebb kódot biztosít. A fejlesztők kevesebb kóddal írhatnak programokat, ami növeli a termelékenységet.

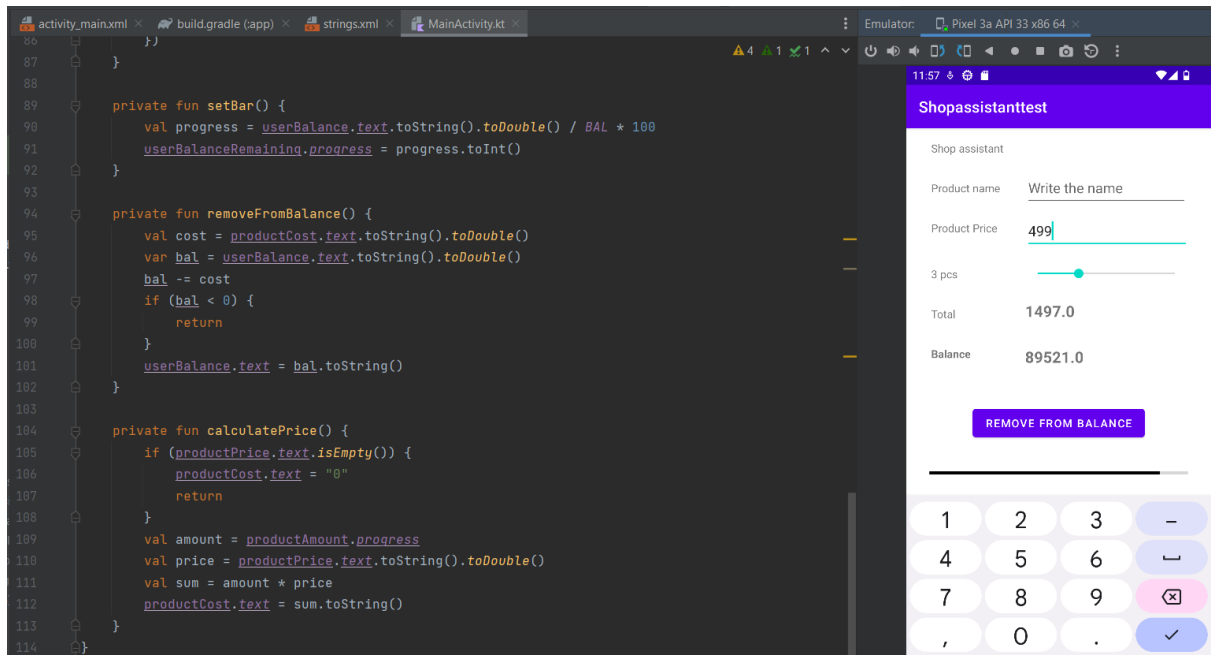
- Eszköztámogatás. A Kotlin rendelkezik az Android eszköztámogatásával az Android fejlesztésre optimalizált eszközökkel, beleértve az Android Studio-t, az Android KTX-et és az Android SDK-t.
- Közösségi támogatás. Bár a Kotlin viszonylag új nyelv a Javához képest, van egy növekvő létszámú fejlesztőkből álló közössége, akik a nyelv fejlesztésén és a dokumentáció biztosításán dolgoznak.

#### **4.2. A kiválasztott fejlesztőkörnyezet**

A kiválasztott programozási nyelv után a fejlesztőkörnyezet kiválasztása következett. A választásom a Kotlin hivatalos fejlesztőkörnyezetére, az Android Studiora esett. Az IntelliJ IDEA nagy teljesítményű kódszerkesztőjére és fejlesztői eszközeire épülő Android Studio még több olyan funkciót kínál, amelyek növelik a hatékonyságot az Android alkalmazások készítése során, például: gyors és funkciógazdag emulátor, egységes környezet, ahol az összes Android eszközre fejleszthetünk, kiterjedt tesztelési eszközök és keretrendszerek [7]. Ez a fejlesztőeszköz szabadon használható, és a saját oldaláról tölthetjük le a fejlesztőkörnyezetet. Az SDK (Software Development Kit) előnye, hogy az oldal design-ját és a benne zajló műveletek implementálását is képesek vagyunk egy programon belül végezni. Magát a programot ilyen formában használom, de egy nagy előnye, hogy rengeteg módon tudjuk szabni a program felületét, ahogyan az adott programozónak tetszik.

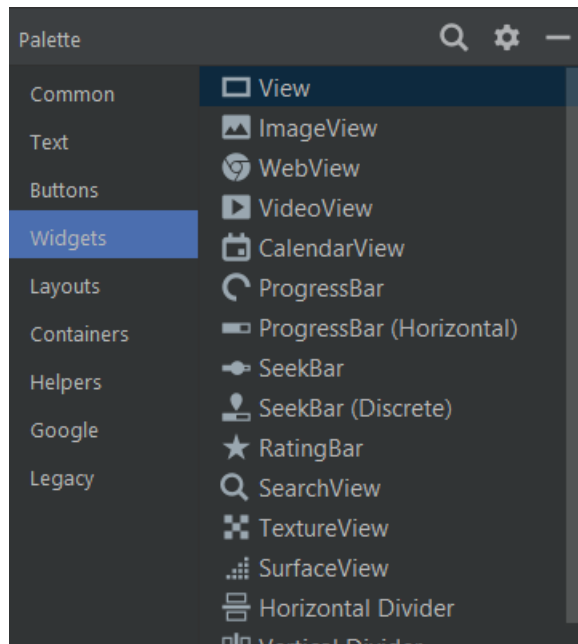
#### **4.3. A Kotlin nyelv szintaktikája**

A Kotlin nyelv működése hasonlít a Java nyelvhez, Ennek fő oka, hogy a Kotlin előtt a Java volt a meghatározó Androidos programozási nyelv, amit használtak a fejlesztők, így az eddigi Java tanulmányaim nagymértékben hozzájárultak ahhoz, hogy a Kotlin nyelvtanja könnyebben tanulható legyen. Ami viszont egyedi dolog ebben a nyelvben, az a különféle elemek használata.



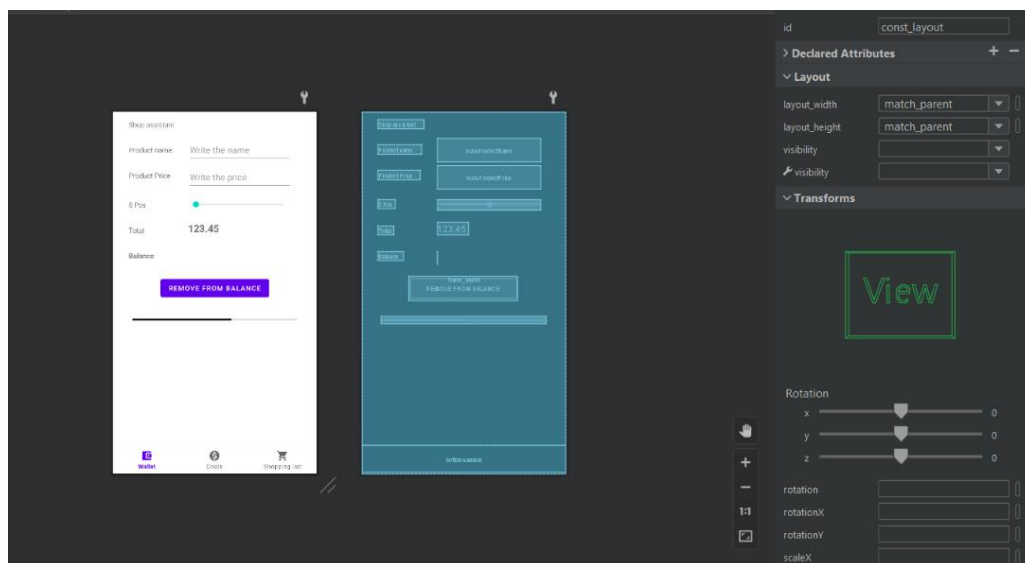
4.1 ábra: Az Android Studio felülete.

A képernyőn megjelenő elemeket View-oknak nevezzük, és minden View-nak különböző neve van. Egy alapvető szöveget TextView segítségével lehet megjeleníteni, ha a felhasználótól szeretnénk adatot kérni, akkor ezt az EditText segítségével tudjuk megtenni. Minden elemnek saját neve van, például a leggyakoribb elemek közül megtalálhatjuk a gombokat, amit Button-ként érhetünk el, ha listázni szeretnénk elemeket, és abból egyet választani kell, azt egy Spinner segítségével tehetjük meg. Elmondható tehát, hogy rengeteg elem közül választhatunk, aminek más-más működési elve van. Példaként, ha SeekBar-t használunk egy haladási csík megjelenítéséhez, akkor a csík előrehaladása mozgathatóvá válik a felhasználó által, míg egy ProgressBar-t használva csak a haladás megjelenítését tudjuk vizualizálni. Ezeknek az elemeknek adhatunk különböző azonosítót is, amivel később hivatkozni is tudunk rá, ha több ugyanolyan elem található meg az oldalon.



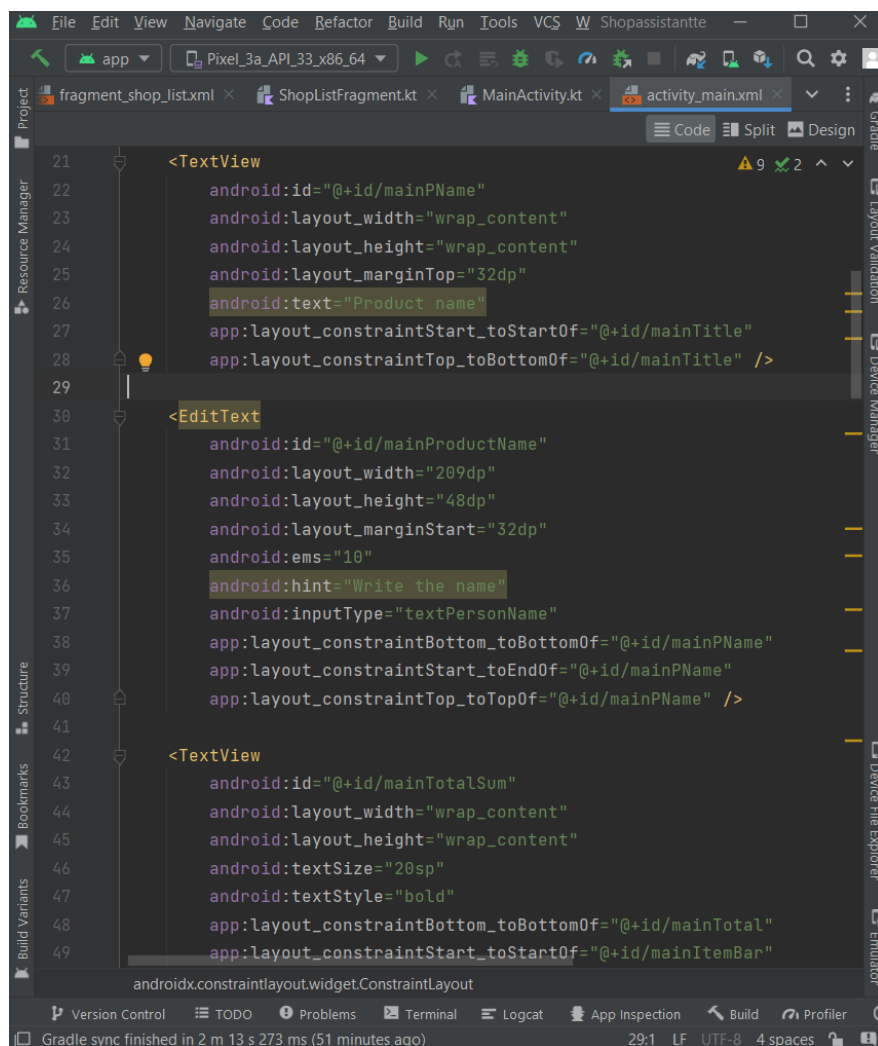
4.2 ábra: Számos View közül választhatunk a fejlesztés során.

A fejlesztés során végezhetjük a tervezést grafikus módon, illetve minden oldal egy .xml formátumú fájlban kerül eltárolásra, amit szöveg formájában tudunk szerkeszteni. A grafikus szerkesztés előnye, hogy úgy jeleníti meg az oldalt előben, mint ha meg lenne nyitva a program előttünk. A szöveges szerkesztési mód előnye, hogy mindig csak azokat az elemeket látjuk, amire nekünk épp szükségünk van. Ez azt jelenti, hogy az SDK-ban nem jelenik meg az adott View minden tulajdonsága, hanem csak azok, amiket a fejlesztő megadott.



4.3 ábra: A grafikus fejlesztői felület.

Minden View-nak meg kell adni azt az értéket, hogy mekkora területet foglalhat el a képernyőből. Ezeket a *layout\_width*, illetve a *layout\_height* változókkal állíthatjuk be. Ezeknek a változóknak kötelezően adni kell értéket, mivel ez alapján fogja tudni majd a rendszer, hogy mit hogyan helyezzen el. Ezeknek a tulajdonságoknak megadhatunk számot, mint értéket, ekkor a megadott számú „density pixel” -en jelenik meg a View. Fontosak azonban a különleges értékek is. A *match\_parent* érték megadásával a terület teljes területén megjelenik a View, a *wrap\_content* használatával pedig mindig csak akkora területet foglal el, amennyi feltétlenül szükséges ahhoz, hogy megfelelően meg tudjon jelenni az információ. Ezen kívül minden View-nak adhatunk meg különböző tulajdonságokat, például képesek vagyunk beállítani szöveget, segítséget a felhasználónak, ami eltűnik, ha adat kerül megadásra, akár a szegélyeket is meg tudjuk határozni stb.



4.4 ábra: XML szerkesztői felület.

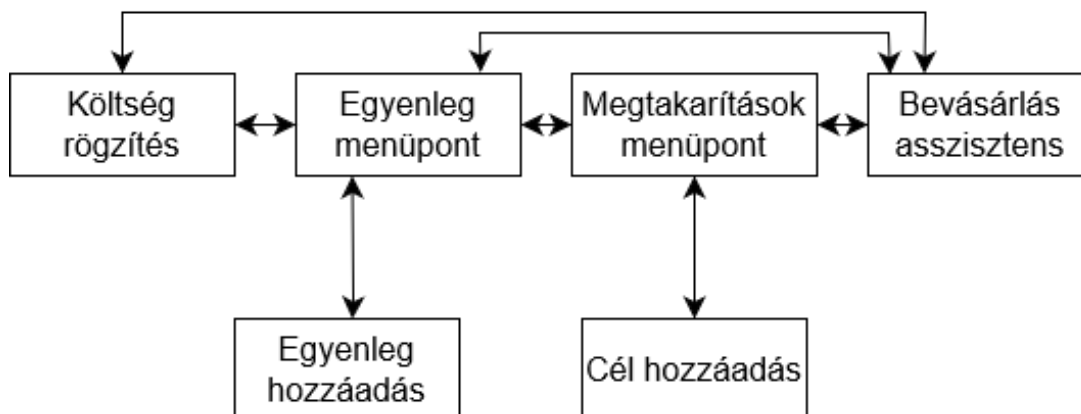


## 5. Az alkalmazás implementálása

Az alkalmazás fő tervezete elkészült a felhasználói leírás alapján. Ezek után az implementálás következik, amit tehát Android Studioban vittem véghez Kotlin nyelvet használva.

### 5.1. Az alkalmazás lapátmenete

Elsősorban meg kell határoznunk, hogy melyik lapból tudunk melyik lapra jutni. Induljunk ki a főegyenlegről! Mivel nagyon gyakori a navigációs csík használata a telefonos alkalmazásokban, így ebben a programban is használhatjuk ezt a funkciót, mivel három fő oldallal fog majd a program rendelkezni. Ez azt jelenti, hogy a *főegyenleg* képernyőről eljuthatunk a *célok* oldalra és a *bevásárlás asszisztens* oldalra is, és ezeken az oldalakon ugyanúgy bármikor visszajuthatunk az egyenleg képernyőre. Ezek tehát oda-vissza kapcsolatban vannak, mindenhol tudunk mindenhol menni a fő oldalak közt. Azonban a fő oldalakon kívül szükségünk lesznek egyéb oldalakra is. Ha az egyenleg teljes funkcionalitását nézzük, akkor ebből az oldalból tudnunk kell majd tranzakciót rögzíteni, és a tranzakció részletes adatait megtekinteni, így ezekhez a funkciókhoz egy-egy oldalt kell majd készíteni. Természetesen ebből ki is kell majd lépni, tehát itt is oda-vissza kell majd tudnunk lépkedni. A célok képernyőjén szükség lesz egy olyan oldalra, ahol új célt tudunk majd létrehozni, de a cél részleteit nem feltétlen szükséges megtekinteni, mivel ezt a kevés adatot a fő oldalon is meg tudjuk jeleníteni. A bevásárlás asszisztens felületéhez is szükség lesz majd egy olyan felületre, ahol a kosár elemeit tudjuk beszúrni és módosítani, de ezt véleményem szerint jobb lenne egy Dialog segítségével megvalósítani, ami segítségével az oldal meghívja ezt a dialógust, és a háttérben futásban marad az oldal. Ezzel elkészült a lapátmenet tervezete, ami a következőképpen néz ki:



5.1 ábra: Az alkalmazás lapátmenetei.

## 5.2. Az alkalmazás lapjainak bemutatása

Az alkalmazást lehetőleg három komponensből szeretném elkészíteni, és mind a három fülnek különböző funkcionalitása lesz. A három fül a főegyenleg képernyő, ahol láthatjuk az egyenlegünket, megtakarítások képernyő, ahol a céljainkat adhatjuk meg, és a bevásárlás asszisztens, amikor vásárolni megyünk, egy külön asszisztenst kapunk a kezünkbe.

### 5.2.2. A főegyenleg képernyő

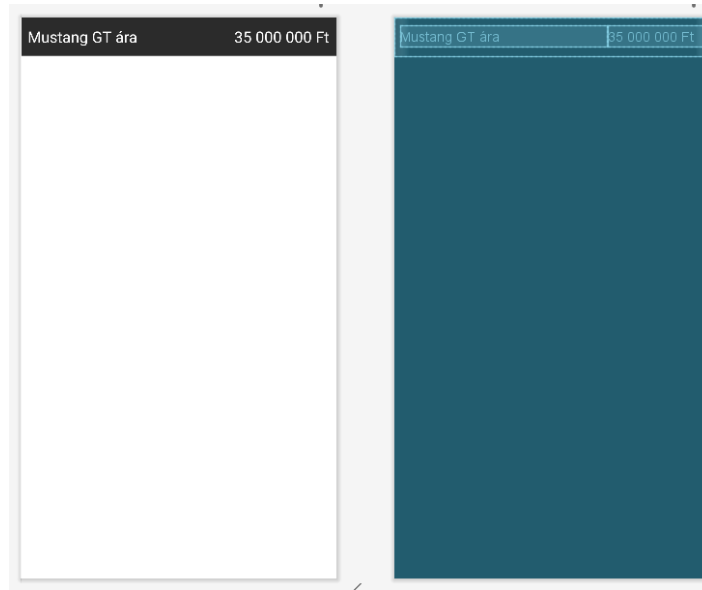
A legfőbb eleme a programnak, hiszen itt fogjuk majd látni a rendelkezésre álló egyenlegünket, és itt tudunk majd költségeket hozzáadni a programhoz, hogy tudjuk követni, hogy milyen dolgokra költöttünk eddig. A program előhossa a legutóbbi költségek nevét, és a tranzakció összegét, időrendben megjelenítve azokat. A költséget piros színnel látjuk, a pénzforrást pedig zölddel. Lehetőségünk van a kategóriák alapján rendszerezni a költségeinket, így csak az adott kategória tranzakcióit láthatjuk. Az osztályok designjait többféle oldalról meg tudjuk közelíteni, ezek lehetnek a Model-View-Presenter (MVP), Model-View-ViewModel (MVVM), Model-View-Intent (MVI) architektúráis felépítések. Az MVP fő előnye, hogy szétválasztja az alkalmazás problémáit, és megkönnyíti az egyes összetevők tesztelését és újrafelhasználását. Az MVVM fő előnye, hogy egyszerűvé teszi a View és a Model közötti kapcsolást, valamint tesztelhetőbbé és karbantarthatóbbá teszi a kódot. Az MVI fő előnye, hogy világos és kiszámítható adatáramlást hoz létre, amely könnyen értelmezhető és javítható. A nézet csak az állapot megjelenítéséért és az Intentek

kibocsátásáért felelős [8]. Az előnyök és hátrányok mérlegelése után az MVVM felépítés tűnik számomra a legjobb módszernek. A tranzakciókat éppen ezért egy adatbázisban kell eltárolni, amit a DAO (Data Access Object) segítségével tárolunk el. Ezzel egy lokális adatbázist tudunk létrehozni, amivel a felhasználó telefonján internethozzáférés nélkül is bármikor a rendelkezésére képes állni. Az adatbázis létrehozásához először egy Interface-t kell létrehoznunk, amiben definiáljuk az adatbázisban szükséges műveleteket, mint például azt, hogy minden elemet le tudjunk kérni:

```
@Query("SELECT * FROM expenses order by id desc")  
fun getAll() : List<BalanceTransaction>
```

Ezzel a Query-vel minden eddigi tranzakciót le tudunk kérni, és ezt majd a felületen meg is fogjuk tudni jeleníteni. Ha bármilyen másféle műveletet végre szeretnénk hajtani az adatbázison belül, akkor az összes SQL parancsot egy ilyen függvény használatával tudjuk létrehozni. Ebben a DAO-ban helyet kapott a beszúrás, a módosítás, egy adott elem törlése, az elemek lehívása, a lehívás kategóriák alapján, és a megtakarítási egyenleget is ezen az adatbázison belül tudjuk lekérdezni. Ezeket a függvényeket később a ViewModel használatával tudjuk ráilleszteni a fő osztályunkra MVVM (Model – View – ViewModel) nézetmodell használatával, ahol a három elemet különválasztjuk egymástól ahhoz, hogy a rétegek szeparáltak legyenek, és könnyebb legyen a különböző rétegeken dolgozni. Ez úgy működik, hogy a View-en keresztül hívjuk meg az adatbázis műveletet a ViewModelt használva, majd a ViewModel a Modelben levő függvénnyel hívódik meg, összekötve ezzel a View-t a Modellel. A tranzakciók adatbázisát tehát egy Model osztályban kell létrehozni, maga a tranzakció majd egy példány lesz, aminek az elemeivel fogunk majd tudni dolgozni. A tranzakció modellének van id-je, neve, összege, kategóriája és egyéb leírása is, amit egy külön fülön tudunk majd megjeleníteni, ha a felhasználó rányom az adott költségre. Ezután következze a felület megtervezése. A cím alatt kap helyet a fennmaradó egyenlegünk, hiszen ez egy fontos adat a számára. Utána a felhasználónak egy összegzést készítünk, hogy mennyi az adott kerete, és eddig ő abból mennyit költött el. Ezután tranzakciók megjelenítésének implementálása következik.

Ezt egy RecyclerView segítségével tudjuk megtenni, amivel ún. kártyákat tudunk megjeleníteni. A kártyát egy külön Layoutban tudjuk létrehozni. Egy ilyen kártya két TextView-ből áll egy LinearLayout-ban, ami a tranzakció nevét és a tranzakció összegét jeleníti meg egy sorban. Tehát a RecyclerView az ilyen Layouttal dolgozik, amit egy osztályban össze kell kapcsolni, majd ebben az osztályban tudunk neki értékeket adni.



5.2. ábra: Így fog kinézni az egyenlegen megjelenő elemünk.

Magát a megjelenítés összekapcsolását a ViewHolder végzi, amit az Adapterben kell létrehozni:

```
class BalanceTransactionViewHolder(view: View) :  
    RecyclerView.ViewHolder(view) {  
    val description : TextView = view.findViewById(R.id.idTransactionLabel)  
    val amount : TextView = view.findViewById(R.id.idTransactionAmount)  
}
```

A `findViewById` a Layout elemeire tud hivatkozni, és ezzel a metódussal tudjuk összekötni a vizuális megjelenést az osztállyal, és az `onBindViewHolder` függvényen belül tudunk neki értéket adni. Fontos megjegyezni, hogy a megjelenő elemeket az adatbázisból tudjuk lekérdezni a `Balance` osztályunkban, így a `RecyclerView` értékadása is itt történik meg. Mivel minden tranzakciónak egyedi értékei vannak, így a `View`-ban minden elemnek van egy sorszáma, amivel tudunk az adott tranzakcióra hivatkozni. Ez fontos is lesz a számunkra, mivel majd egy `onClickListener`-t is kell alkalmaznunk ahhoz, hogy a kívánt tranzakció elemeit egy külön képernyőn tudjuk

megjeleníteni. Ez a beépített függvény arra szolgál, hogy a kattintásokat figyeli, és ha rányomunk egy tranzakcióra, akkor végrehajtódik, ami ebben a metódusban van.

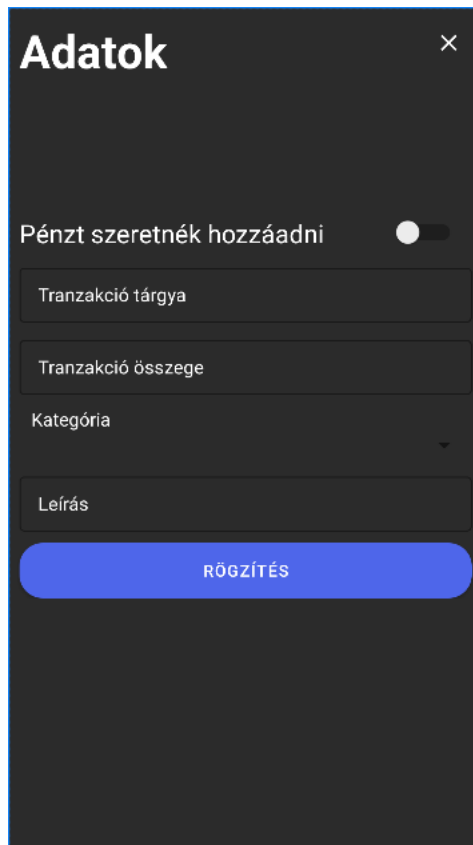
Amire még szükségünk lesz, az egy `FloatingActionButton`, ami olyan gomb, hogy a képernyőn állandóan egy pozícióban van, ezzel meg tudjuk jeleníteni az új tranzakció rögzítésére szánt felületet.

Ezen elemek implementálásával elkezdhetünk dolgozni a fő Activityben, ami maga az egyetlen képernyőt adja, itt fog megjelenni minden, amit eddig létrehoztunk.



5.3. ábra: Így fog majd kinézni az egyenleg oldalunk.

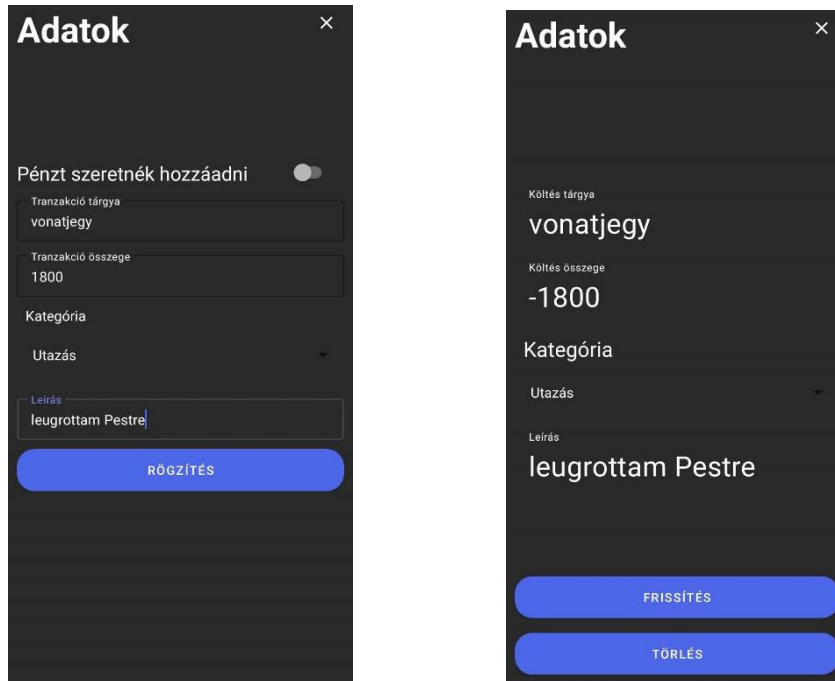
Ezután hozzuk létre a tranzakció létrehozására szolgáló Activityt! Ha a `FloatingActionButton`-hoz hozzákötjük a rögzítésre szolgáló Activityt, amit egy újabb Activity létrehozásával tudunk megtenni, akkor a gomb lenyomásakor meg fog jelenni ez a képernyő, ahol a tranzakció adatait vagyunk képesek betáplálni:



5.4 ábra: A tranzakciókat ezen az oldalon tudjuk rögzíteni.

Itt már fontos ellenőrzéseket kell végrehajtanunk, mert ha a felhasználó rosszul adja meg az adatokat, akkor összeomolhat az alkalmazásunk, ha az adott hibát nem kezeljük le. Elsősorban van egy Switch elem, amit ha igazra állítunk, akkor a program tudni fogja, hogy bevételi forrást szeretnénk hozzáadni a tranzakcióinkhoz, ezért majd pozitívként jelenik meg az összeg. Ezzel együtt a kategóriánk is megváltozik, és a kategória fizetésre változik. A tranzakció tárgyánál bármilyen szöveget meg lehet adni, de fontos kritérium, hogy ez a mező nem lehet üres. A tranzakció összege egy szám, tehát ha ide szeretnénk betáplálni adatot, akkor csak egész számot fogunk tudni beírni. A kategória kiválasztását egy Spinner-rel oldottam meg, ami lényegében egy legördülő listát jelent, és a lista a kategóriák szöveges tömbjéből áll, amit előre definiáltam az alapadatokban. Itt kiválaszthatjuk a számunkra legmegfelelőbb kategóriát. Választhatjuk a mindennapi élethez tartozó kategóriát, ilyen például a bevásárlás, a hobbit, ha van egy hobbink, amire költöttünk, ha a házunkhoz tartozó tevékenységet végzünk, vagy befizetjük otthonunk rezsiköltségét stb., akkor az otthoni kategória a megfelelő. Ha kikapcsolódunk, akkor azok a költségek a szórakozás kategóriájába

kerülhetnek, illetve, ha például levonatozunk a Balatonra, akkor a vonatjegyet az utazás kategóriába tehetjük. Ha pénzt szerzünk, az a fizetés kategóriába kerül. Külön kategóriát képvisel a megtakarítás, ennek a kategóriának a megtakarítások menüpontban lesz fontos szerepe.

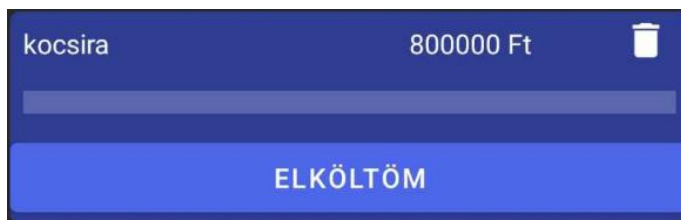


5.5 ábra: A tranzakció rögzítése és a tranzakció részletei.

### 5.2.2. A megtakarítási felület

A program másik fontos eleme, hogy legyen egy olyan felület, ahol félre tudjuk tenni a pénzünket, és ezáltal elindulhatunk a megtakarítási céljaink felé. Legyen az bármilyen cél, azt hozzá tudjuk adni a programhoz. Az Activity felépítése nagyjából hasonló módon épül fel, mint az egyenlegé, ugyanúgy szükségünk lesz majd egy adatbázisra, ahol a céljainkat el tudjuk tárolni, tehát ismét definiálnunk kell egy DAO-t, és mellé a szükséges elemeket is, az adatbázist, az adaptert a RecyclerView-höz ahhoz, hogy rendeltetésszerűen tudjuk tárolni az adatokat. Az egyenleges adatbázishoz képest azonban itt más változókra lesz szükség. A célunknak kell majd egy nevet adnunk, hogy tudjuk mi az, amire gyűjtünk, illetve azt, hogy maga a cél milyen összegben nyilvánul meg, tehát mennyi pénzt kell megspórolnunk ahhoz, hogy a célunkat elérjük. A megtakarítás fül mellett el kell készíteni a cél létrehozására szolgáló lapot is, ahol meg tudjuk adni a cél nevét és az összeget, az integritási

szabályoknak eleget téve. Ezeket az elemeket is egy RecyclerView-en keresztül jelenítem meg, ahol egy cél egy kártyán jelenik meg.



5.6 ábra: Ilyen formában fognak megjelenni a célkitűzések.

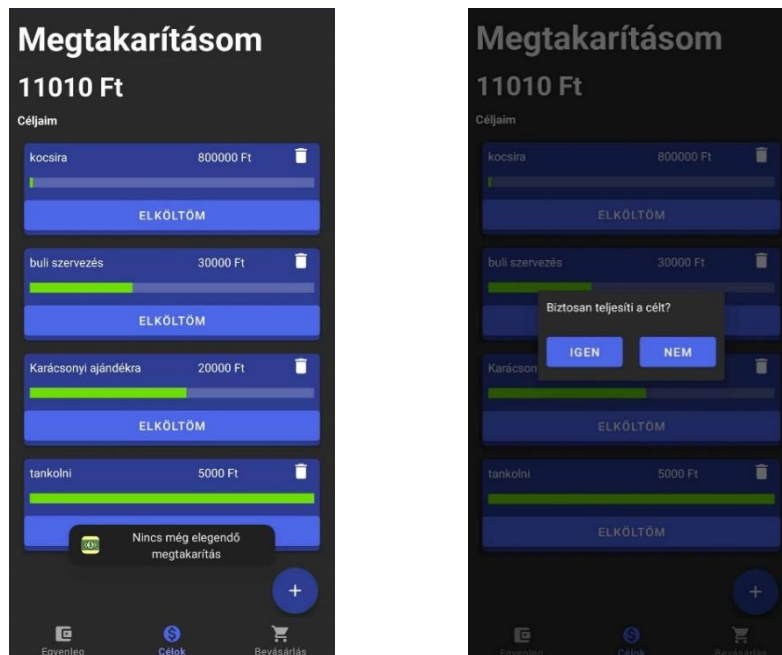
Látható, hogy a kártyán megjelenik a cél neve és az összege. Középen egy Progress Bar mutatja meg nekünk azt, hogy a megtakarított egyenlegünk milyen arányban teljesíti a meghatározott összeget. Ezt úgy számolja ki az alkalmazás, hogy a megtakarítási egyenleget és a cél összegét elosztja egymással, majd a százalékos arányát képezi. Ha a megtakarításunk nagyobb, mint a cél összege, akkor a Progress Bar haladása 100% lesz, mivel az összeg egészével rendelkezünk. Azt, hogy mennyi megtakarításunk gyűlt össze eddig, azt a *megtakarítás* kategória segítségével tudjuk lekérdezni.



5.7 ábra: A megtakarítási felület.



Az összegyűlt megtakarítás összegét úgy számolhatjuk ki, hogy az összes rögzített *megtakarítás* kategóriájú tranzakció összegét összeadjuk, és ebből kivonjuk a már teljesített célokat. A jobb oldali *kukára* nyomva törölhetjük az adott célt, ekkor nem látjuk többé azt, az *elköltem* gombbal pedig véglegesíthetjük a kívánt célkitűzést, amivel le tudjuk vonni a megtakarításunkat, ha sikerült elérnünk a célunkat.

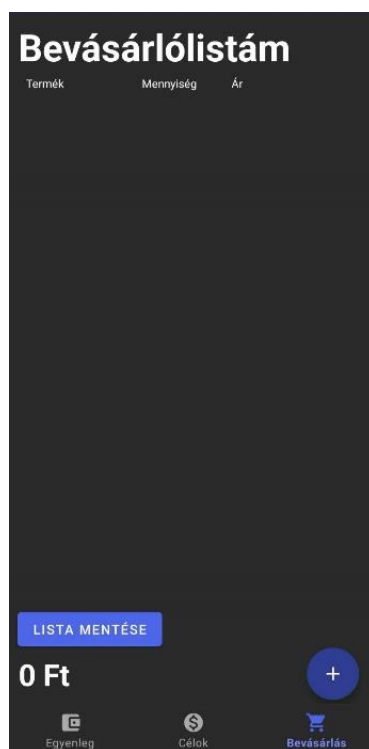


5.8 ábra: Az elköltség gomb különböző interakciói.

Azonban itt még történik egy ellenőrzés azzal kapcsolatban, hogy rendelkezik-e a felhasználó a szükséges összeggel ahhoz, hogy azt ki tudja váltani. Ha nem elég a megtakarított egyenleg, akkor egy üzenet jelenik meg, ami tájékoztat bennünket arról, hogy még nem elég az a mennyiség, amit összegyűjtöttünk. Viszont, ha már rendelkezünk az összeggel, amire szükségünk van, akkor egy dialógus jelenik meg, ahol meg kell erősítenünk a kiváltási szándékunkat. Ha igenre nyomunk, akkor a cél törlődik, és a cél összege levonásra kerül a megtakarítási egyenlegből, a főegyenlegre viszont nincs hatással. A céljaink úgy jelennek meg a felületen, hogy a legmagasabb költségű cél található meg legfelül, majd összeg szerint csökkenő sorrendben jelennek meg a további célok. Természetesen ez a funkció akkor a leghatásosabb, ha a valós környezetben is le tudjuk tenni azt az összeget, amit megtakarításként tartunk számon, és ha elérjük a célunkat, akkor már csak elő kell venni azt, amit eddig megspóroltunk.

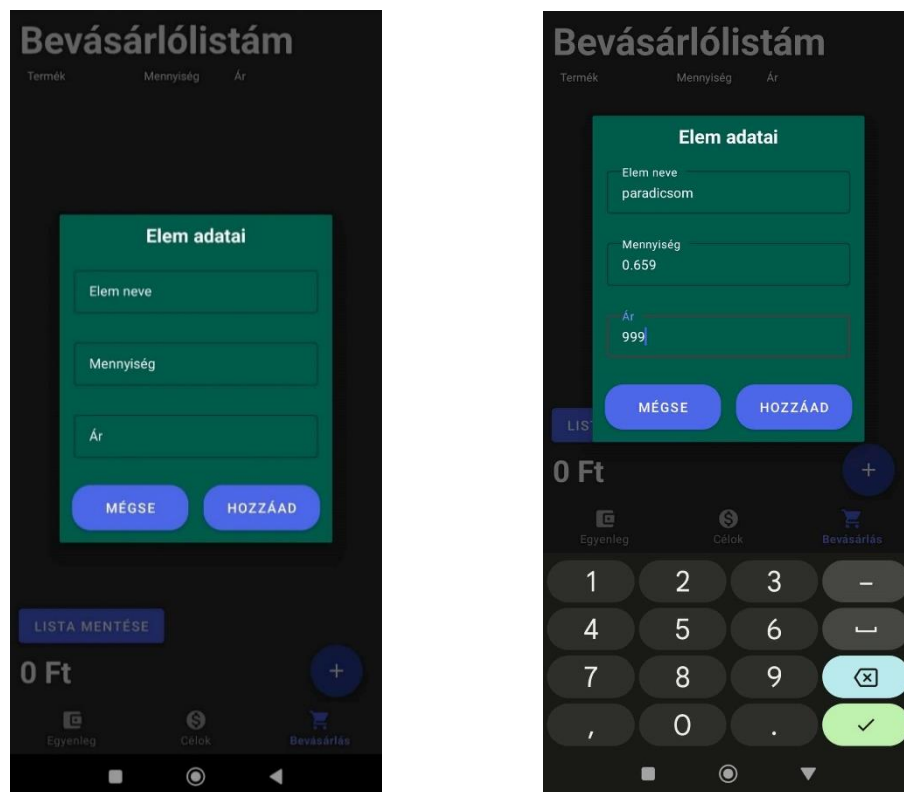
### 5.2.3. A bevásárlás asszisztens

Az egyik legérdekesebb dolog mikor vásárolni megyünk, az szerintem az, mikor csak vásárlunk és vásárlunk, majd egy kis idő után már nem tudjuk követni azt, hogy mekkora összegnél is tart a vásárlásunk. Szerintem sokan járnak így, elég csak abba belegondolni, hogy mennyi ember lepődik meg azon, hogy milyen összeg jelenik meg a pénztár előtt. Természetesen ennek sok oka lehet, de én leginkább azt az oldalt szeretném megfogni, ami az impulzus vásárlások miatt keletkezik. Azt nevezzük impulzus vásárlásnak, amit nem szerettünk volna venni a boltban érkezés előtt, de a pillanat hevében aztán mégis a kosárban végzik ezek a termékek, pedig vásárlási szándék nem állt fenn az ilyen termékekkel kapcsolatban. Sokszor még olyankor is előfordul az ilyen eset, mikor a vásárló egy bevásárlólistával megy el vásárolni, de arról ne is beszéljünk, hogy még rosszabb a helyzet akkor, ha csak spontán – bevásárlólista nélkül – érkezünk vásárolni. Ennek a problémának a megoldására ötleteltem ki a bevásárlás asszisztens felületet, amire a vásárlás közben rögzíteni tudunk minden adatot, így folyamatosan tudjuk követni, hogy milyen összegnél tartunk, illetve, hogy mi mennyibe kerül a kosarunkban. Az elemeket egy adatbázisban tudjuk letárolni, így akár a vásárlás kigondolásakor már használhatjuk is a programot.



5.9 ábra: A bevásárlólista megnyitásakor ez a kép fogad minket.

Egy elem rögzítéséhez szükséges a termék neve, a mennyiség amennyit venni szeretnénk, és az adott termék egységára. Egy új terméket a lent található plusz gomb megnyomásával adhatunk meg, amivel megjelenik a termék rögzítésére szolgáló dialógus. A termék neve egy Stringként kerül eltárolásra, így az bármilyen formátumú lehet, de fontos, hogy nem lehet üres, hiszen később bajban lennénk, hogy vettünk két darab névtelen terméket, aminek 999 Ft az ára. A termék mennyiség eltárolását már érdekesebb kigondolni, hiszen mit írjunk be akkor, ha nem egész számú mennyiségű dolgot veszünk, hanem a termék például Ft/kg egységben van megadva, példaként főleg a zöldségek és gyümölcsök? Éppen ezért a mennyiség egy lebegőpontos szám, tehát megadáskor képesek vagyunk valamiből akár 0,644 egységet is venni. A termék ára viszont egész számként kerül eltárolásra, mivel az alkalmazás forintra lett tervezve, és manapság már nem értelmesebb filléreket.



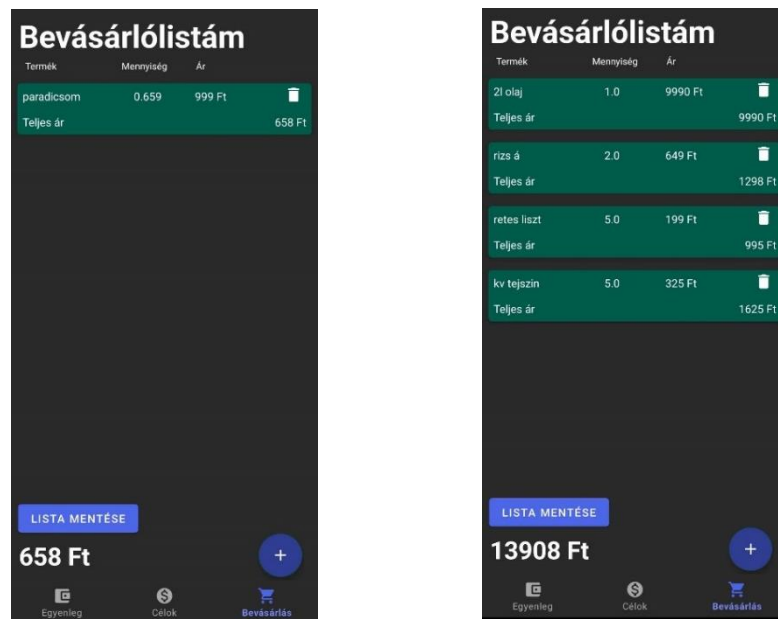
5.10 ábra: Egy termék adatainak rögzítése.

Ha rögzítjük a terméket, ezután a program kiszámolja, hogy ez összesen mennyibe fog kerülni, összeszorozva a mennyiséget a termék árával, illetve, ha nem egész mennyiséget vásárlunk, abban az esetben kerekítést hajt végre. A termék árát a következő kód írja le:

```
val itemTotal : Int = (list[position].itemPrice *  
list[position].itemAmount).roundToInt()
```

Természetesen ez még nem elég ahhoz, hogy teljesen nyomon tudjuk követni az eddigi költésünket, a felületen az összes termék árát is össze kell adnunk ahhoz, hogy megkapjuk, mennyinél jár a kosarunk értéke. Erre egy TextView nyújt megoldást, aminek a szöveges értékét minden adatváltozáskor frissítjük, összeadva benne az eddig megadott elemek árainak összegét.

Természetesen van lehetőségünk a rögzítés után az adatok módosítására is, például, ha valami akciós és nem tudunk róla, akkor az árát tudjuk módosítani, illetve, ha valamiből többet szeretnénk venni, akkor a mennyiséget is módosíthatjuk. Ekkor, ha rányomunk egy termékre, akkor hasonló módon megjelenik a termék rögzítő dialógus, azonban a már megadott adatokat előre betölti számunkra, ezeket lehet módosítani. A módosítás végrehajtása után nem hoz létre egy új kártyát az alkalmazás, hanem a meglévő kártya adatait módosítja.



5.12 ábra: Egy bevásárlólista vizualizációja.

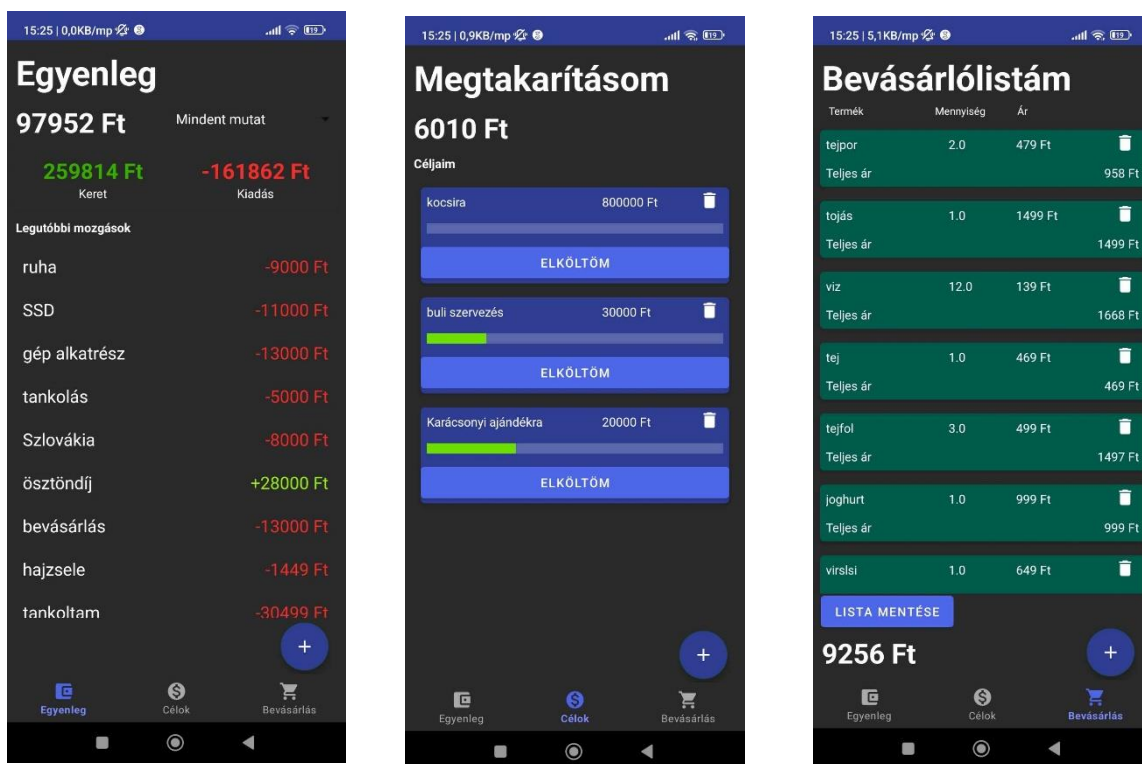
Ha teljesült a vásárlás, akkor a lista mentése gombra kattintva megjelenik a tranzakció rögzítő Activity. A rögzítő felületen a tranzakció összege automatikusan kiegészül a bevásárlólista összegével. Ha teljesítésre kerül a rögzítés, akkor az eddigi bevásárlólista elemei eltűnnek, és egy új, üres lista nyílik meg.

## **6. Az elkészült alkalmazás tesztelése és tapasztalatok**

Miután a program implementálása elkészült, elkezdhető az applikáció tesztelése. A program saját magam, és a szüleim segítségével, használat alapján került tesztelésre. A tesztelés alatt minden funkcionalitását kipróbáltuk a programnak, a jelenlegi keretemet a pénzügyi asszisztens segítségével vezetem. A fejlesztés elkészülte óta napi szinten használatban van a program, és a bevásárlási listákat is ennek segítségével állítjuk össze. Mindazonáltal, hogy a program reszponzív elemeket tud nyújtani, ezért a bevásárlás során elengedhetetlen kelléke alakult ez az eszköz. Néhány célt nekem is sikerült felállítanom, amikre a dolgozat írásakor szeretnék gyűjteni. A tesztelés eredményeként elmondható, hogy a tervezés legtöbb része elkészült, és teljes mértékben egy hatékony alkalmazást sikerült fejleszteni.

Mivel jelenleg nem elérhető a program a Play Store-on belül, ezért .apk fájlkiterjesztéssel lehet feltelepíteni a telefonunkra az alkalmazást, amit az Android Studio generál le. Mivel az alkalmazás teljesen működőképes állapotban van, így lehetőségem lenne majd arra is, hogy a publikus áruházban nyilvános legyen, így ez a program bárki számára elérhetővé válik. Véleményem szerint a célközönség nagy része megfontolná az alkalmazás használatát. A Play Store-ra való feltöltés azonban megköveteli a fejlesztői profil létrehozását, aminek sajnos díja van, tehát emiatt még nem tudom ezt a lépést megtenni. Az áruházba való bekerülés lépései a következők:

- Létre kell hoznunk egy fejlesztői profilt. Ez a lépés azért fontos, mert ha visszajelzés érkezik a programmal kapcsolatban, akkor a Play Store-nak tudnia kell, hogy a fejlesztőt milyen helyeken lehet elérni.
- A fejlesztői profil létrehozása után fel kell töltenünk az elkészített programot a profilunkba.
- Az áruházba kerülés előtt az áruház adminisztrátorai ellenőrzik a programkódot, nehogy kártékony applikáció kerüljön fel az oldalra.
- Ha mindent rendben találnak, akkor nyilvánossá válik az alkalmazás, ami bárki számára elérhető lesz.



6.12 ábra: A jelenleg használatban levő alkalmazás felületei.

## 7. További fejlesztési javaslatok

Az alkalmazás elkészültével teljesen mondható a fejlesztési folyamat. Azonban mindig vannak egyéb lehetőségek, amivel még ki lehet egészíteni, fejleszteni tovább az alkalmazásunkat. Néhány ötletet még összegyűjtöttem, hogy mik azok az opciók, amikkel még a jövőben ki lehetne egészíteni a programot.

### 7.1. Dátumok implementálása minden felületre

Ez az egyik legnagyobb dolog, amivel még ki lehetne egészíteni a program működését. Mivel az időt egy külön elemként kell eltárolni az adatbázisban, ezért az adatbázist újra kellene építeni ahhoz, hogy a dátumot is el tudjuk tárolni a programban.

### 7.2. A határidők és az ismétlődő tranzakciók implementálása

Ennek implementálásával egy teljes költségvetési módszert is lehetne majd implementálni, amivel a felhasználó jobban tudja figyelni, hogy a havi egyenlegéből mennyit költhet még a különböző kategóriákra.

### **7.3. Több ösztönzés a megtakarításra**

Jelenleg csak akkor tudunk megtakarítást létrehozni, ha kifejezetten az egyenlegből készítünk egy tranzakciót, amivel a megtakarításra helyezzük át az egyenlegünk egy részét. Ezt a funkcionalitást lehetne növelni például kerekítéssel, ha mondjuk valaki 13 700 Ft-os tranzakciót ment el, akkor a program feljajánlhatná, hogy kerekítéssel 14 ezer Ft-ra kiegészítve elhelyezze a lecsípett összeget a megtakarítási számlára. Ugyanez a funkcionalitás implementálható lehetne a bevásárlás asszisztens oldalon is.

### **7.4. Több fül a bevásárlás asszisztensen belül**

Önmagában ez nem jelentene nagy változást, de szerintem ez egy hatékonyabb bevásárlást tudna eredményezni, főleg akkor, ha egyszerre több boltba is szeretnénk menni egyszerre. Ekkor a boltokhoz külön-külön listát lehetne hozzáadni, amivel minden boltnál külön-külön tudjuk követni a kosarunk értékét.

## **Köszönetnyilvánítás**

Ezúton is szeretném megköszönni a családomnak a rengeteg segítséget, amit az egyetemi éveim alatt kaptam, az egyetemi tanáraimnak, akitől rengeteg hasznos tudást kaptam, és Piller Imrének, aki rengeteg segítséget nyújtott az egyetemi tárgyaimban, illetve a szakdolgozat megírásában is rengeteg segítséget nyújtott.

## Összegzés

A pénz beosztása mindenki számára egy mindennapos feladat. Van, aki nagyon sokat figyel rá, de akad olyan is, aki a mának élve nem foglalkozik vele annyit, mint amennyit ez a dolog megkövetel. Sokszor elgondolni sem tudjuk, hogy mennyivel másabb módon is be lehetne osztani a pénzünket, mint ahogyan azt sokszor tesszük. Sokan észre sem veszik, hogy milyen dolgokra megy el a pénze, és egyszer csak arra ébred, hogy már csak szűkösen tudja kezelni a bevásárlásokat. Erre a problémára lett hivatott a kigondolt és megtervezett pénzügyi alkalmazásom, amit Kotlinban fejlesztettem le, az Android Studio használatával. A dolgozat megírása alatt egy pénzügyi asszisztent hoztam létre, amivel sokkal átláthatóbbá tudjuk tenni a mindennapos költségeinket, és ezzel egy sokkal átláthatóbb képet kapunk, hogy mire is megy el a havi keretünk. Lehetőségem volt a Kotlin szintaktikáját és működési elvét tanulmányozni, ezáltal sok tapasztalatra tettem szert ezen programnyelv alkalmazásában. A telefonos fejlesztés most éli aranykorát, mivel manapság szinte mindenki használ valamilyen márkájú okostelefont, és rengeteg potenciál lehet ezen rendszerekre való fejlesztésnek. Sokan már mindenre képesek a telefonjuk használatával, képesek irányítani az otthonunkat, az egymással történő kommunikáció is legtöbbször ezen az eszközön történik. A pénzügyeiket is sokan már a telefonjuk segítségével intézik, emiatt az emberek nem érzik annyira az elköltött pénzük súlyát, így emiatt túlköltekeznek. Alkalmazásom azon célt szolgálja, hogy a takarékoságot támogassa, és az ember közelebb kerüljön a pénzéhez. Erre a célra hoztam létre a saját pénzügyi asszisztensem, ami a felhasználó egyenlegét vizualizálja, és egy tisztább képet mutat a pénzügyi helyzetéről. Az *Egyenleg* fülön a felhasználó a mindennapos költségeit tudja vezetni, amelyeket a tisztább átláthatóság érdekében zöld (mint bevétel) és piros (mint kiadás) színekkel jelöltem. Mindemellett a kiadásokat különböző kategóriákba is tudja rendezni, például utazás, szórakozás, hobbi stb. A tranzakciók rögzítésére egy külön oldal nyílik meg, ahol a pénzmozgás adatait rögzíthetjük. Itt tudjuk rögzíteni a kiadásainkat és a bevételeinket is. Ha pénzt szeretnénk hozzáadni, azt a Switch kapcsolásával tehetjük meg. A *Célok* fülön található meg a felhasználó által létrehozott célkitűzések, és az eddig összegyűjtött megtakarítása. A különböző



célokra szánt összeg és a megtakarítás arányát egy Progress Barokkal ábrázoltam. Ha a megtakarított összeg eléri az adott célra szánt díjat, akkor válik elkölthetővé a pénzmennyiség. A *Bevásárlás* oldalon a különböző megvásárolandó termékek jelennek meg. A bevásárlólistához egyszerűen és gyorsan tudunk új tételeket adni a termék neve, mennyisége és egységára alapján. Automatikusan kiszámolja az összeget, valamint a kosár értékét. A kosár mentésével meggyorsíthatjuk a rögzítés folyamatát, valamint a rögzítés után egy új bevásárlólista nyílik meg.

A fejlesztés során megszerzett tudást később is fogom majd tudni kamatoztatni, hiszen ezt a programot is sikerült elkészíteni, ami egy nagyon hasznos eszközzé vált a felhasználóknak. A családban a fejlesztés elkészülte óta napi használatban van a program. Lehetőségeim szerint tovább is fogok tudni ezzel a programmal foglalkozni.

## Summary

Managing money is a daily task for everyone. Some people pay a lot of attention to it, but there are also those who live in the present and don't pay as much attention to it as it requires. Often we cannot even think of how many other ways we could allocate our money than we often do. Many of us don't even realise what our money is going towards, and suddenly we realise that we can only manage our shopping on a tight budget. This is the problem that my financial app, developed and designed in Kotlin using Android Studio, was designed to solve.

While writing this thesis, I created a financial assistant that allows us to be more transparent about our everyday spending, and gives us a more transparent picture of what our monthly budget is being spent on. I have had the opportunity to study the syntax and working principles of Kotlin, which has given me a lot of experience in using this programming language. Phone development is in its golden age, as almost everyone is using some brand of smartphone nowadays, and there is a lot of potential for developing for these systems. Many people can now do everything using their phones, they can control their homes, and most of the communication with each other is done on this device. A lot of people now use their phones to manage their finances, so people don't feel the weight of the money they spend so much, so they overspend. My app is designed to encourage saving and bring people closer to their money. For this purpose, I have created my own financial assistant, which visualises the user's balance and shows a clearer picture of his financial situation. In the Balance tab, the user can keep track of their daily spending, which I have marked in green (as income) and red (as expense) for clarity. You can also sort your expenses into different categories, such as travel, entertainment, hobbies, etc. A separate page opens to record transactions, where you can enter your cash flow details. Here we can record our expenses as well as incomes. If we want to add money, we can do so by flipping the Switch. The Goals tab shows the goals the user has created and the savings he has accumulated so far. I have plotted the ratio between the amount spent on different goals and the savings with a Progress Bar. When the amount saved reaches the amount of money that is to be spent for that goal, the amount of money becomes available for

spending. The Shopping page shows the different products available for purchase. You can quickly and easily add new items to the shopping list by product name, quantity and unit price. It automatically calculates the amount and the value of the basket. By saving the shopping basket, you can speed up the process of storing and a new shopping list will open after recording.

I will be able to use the knowledge I have gained during the development of this program later on, as it has become a very useful tool for users. Since the development was completed, the programme has been in daily use in the family. As far as possible, I will keep on improving this programme.

## Forrásmegjelölés

- [1] AlternativeTo: GnuCash alternatives,  
<https://alternativeto.net/software/gnucash/>, 2023. május 28.
- [2] GnuCash project: GnuCash, Free Accounting software,  
<https://www.gnucash.org/>
- [3] Finsify Technology Co.: Money Lover, Expense manager and budget planner,  
<https://moneylover.me/>
- [4] OTP Bank: Az OTP internet- és mobilbank új Persely funkciója,  
<https://www.otpbank.hu/portal/hu/Uj-IBMB-kisokos/Persely>
- [5] Google for Developers: Kotlin language overview,  
<https://developer.android.com/kotlin>
- [6] TechTarget: What is Kotlin? Usages and benefits of the language,  
<https://www.techtarget.com/whatis/definition/Kotlin>
- [7] Google for Developers: Meet Android Studio,  
<https://developer.android.com/studio/intro>
- [8] Alexandra Grosu: Android app architecture with Kotlin: MVP, MVVM, and more, <https://medium.com/@alexandragrosu03/android-app-architecture-with-kotlin-mvp-mvvm-and-more-6faf7cd1d3cf>, 2023. szeptember 18.

Az internetes hivatkozások utolsó ellenőrzési dátuma: 2023. november 10.