**Szoftver dokumentáció**

Szoftver neve: Személyes pénzügyi menedzser

Szoftver verziószáma: 0.9

Készítette: Bogdán Botond, Orosz Dániel, Faragó Levente

**Tartalomjegyzék**

[**Bevezetés** 3](#_Toc196486787)

[**A szoftver célja** 3](#_Toc196486788)

[**Kiknek ajánljuk?** 3](#_Toc196486789)

[**Program telepítése** 4](#_Toc196486790)

[**A program funkciói felhasználók részére** 5](#_Toc196486791)

[**Bejelentkezés** 5](#_Toc196486792)

[**Regisztráció** 6](#_Toc196486793)

[**Elfelejtettem a jelszavam** 7](#_Toc196486794)

[**Főoldal** 8](#_Toc196486795)

[**Elemzés** 9](#_Toc196486796)

[**Kategóriák** 11](#_Toc196486797)

[**Rendszeres kifizetések** 12](#_Toc196486798)

[**Beállítások** 13](#_Toc196486799)

[**Kijelentkezés** 14](#_Toc196486800)

[**A program funkciói fejlesztők részére** 15](#_Toc196486801)

[**Kezdés** 15](#_Toc196486802)

[**A projekt első látásra** 16](#_Toc196486803)

[**Különböző részek elérése** 17](#_Toc196486804)

[**Különböző részek szerepe** 18](#_Toc196486805)

[**Fooldal.kt** 18](#_Toc196486806)

[**ElemzesActivity.kt** 20](#_Toc196486807)

[**Rendszereskifizetesek.kt** 22](#_Toc196486808)

[**Kategoriak.kt** 24](#_Toc196486809)

[**BeallitasokActivity.kt** 26](#_Toc196486810)

[**Még hozzáadható, vagy változtatható funkciók:** 30](#_Toc196486811)

[**Fejlesztők számára bejelentkezés teszteléshez** 30](#_Toc196486812)

[**Az adatbázis** 31](#_Toc196486813)

[**A tesztelés folyamata** 32](#_Toc196486814)

[**Gyakran előforduló hibák** 32](#_Toc196486815)

[**Más hibák** 32](#_Toc196486816)

[**Segítség a tesztelés során** 33](#_Toc196486817)

[**Összefoglalás** 34](#_Toc196486818)

[**A program a jövőben** 34](#_Toc196486819)

# **Bevezetés**

## **A szoftver célja**

A szoftverünk célja a hétköznapi költekezések egyszerű nyomon követése, illetve a pénzünkkel való egyszerű számítás. Ezek egyszerű, felhasználó barát, és egyértelmű segíti a program különböző funkciói.

## **Kiknek ajánljuk?**

Ez a szoftver akár saját magamnak, az olyan embereknek szól, akik nem feltétlen a legjobb módon tudnak a pénzükkel bánni. Ez a szoftver segít mindenféle költekezéssel előre számolni a bevételünkből, és azzal a reális, számunkra igazán elérhető összeget mutatja, amit eltudunk költeni.

# **Program telepítése**

A GitHubon, az alkalmazás repositoryjában található egy apk, amit ezen a linken érhet el [link]. Ezt letölti a telefonjára, majd telepíti is azt. Ezután el is indíthatja a programot.

[kép a GitHub oldalról, majd a telepítésről]

# **A program funkciói felhasználók részére**

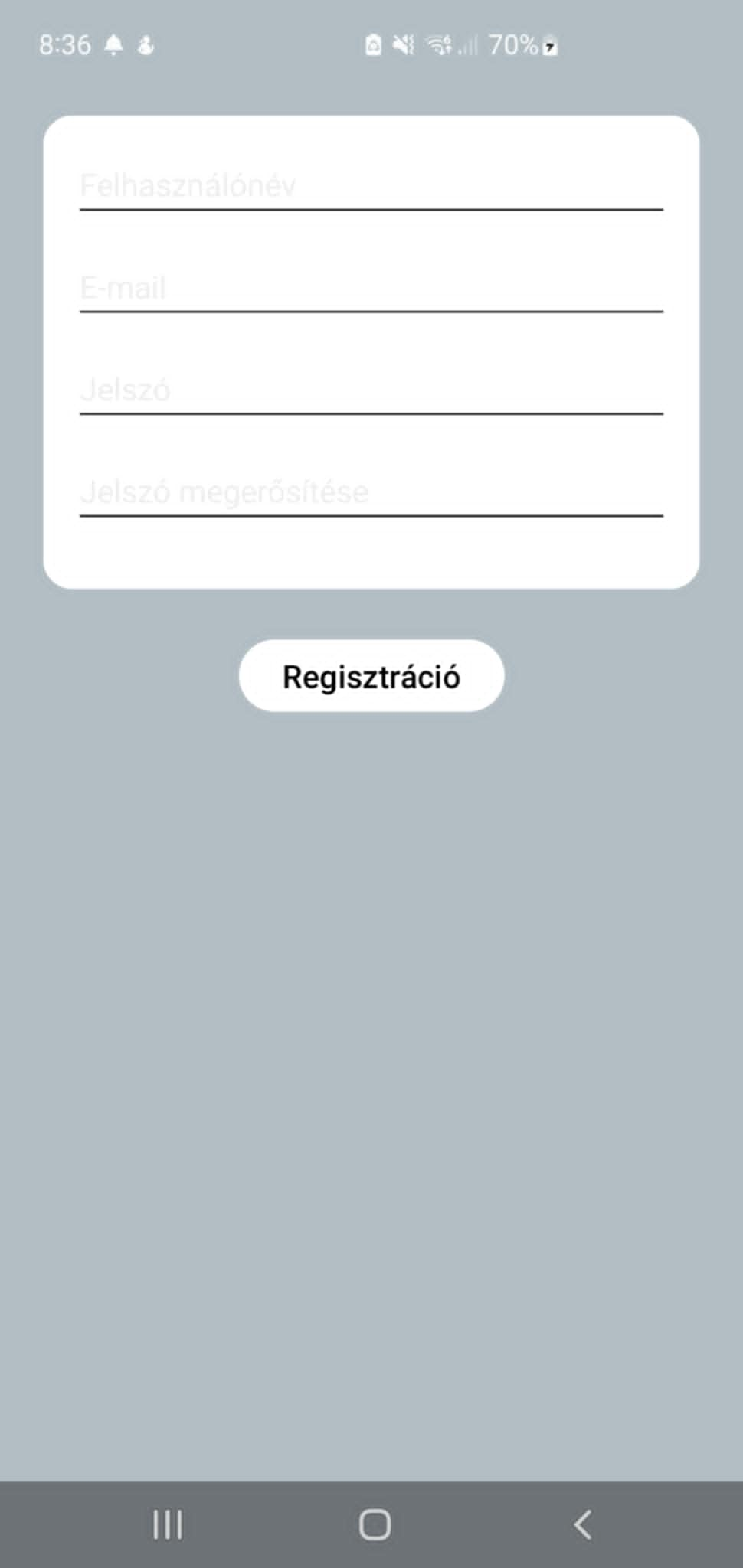
## **Bejelentkezés**

Amikor először elindítja a programot, akkor ez az első oldal, amit látni fog. Itt bejelentkezhet a már létező fiókjába, illetve ha bejelentkezve maradna, akkor a mezőre nyomva ezt a funkciót is alkalmazhatja, így nem kell minden egyes indításkor ezt megtennie. Ha esetleg még nincs fiókja, az ott látható Regisztráció gombbal regisztrálhat.



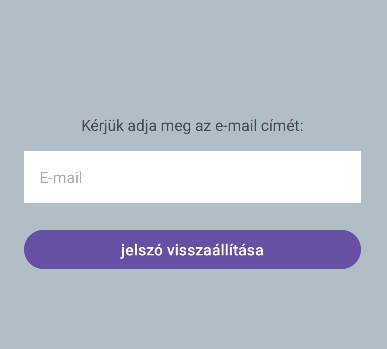
## **Regisztráció**

Ezen a fülön regisztrálhat alkalmazásunk használatához. Itt az e-mailjét, és egy jelszót kell megadnia. Ha ezekkel megvan, utána az alkalmazás fog küldeni magának egy e-mailt, amin igazolnia kell azt, hogy ön hozta létre fiókját. Majd, ha ezt igazolta, bejelentkezhet az alkalmazásba. Egy általa érvényes, és használatban levő e-mailt adjon meg, hogy tudja fogadni az alkalmazástól kapott e-maileket, azonban a jelszónak kell tartalmaznia kis-, és nagybetűt, számot, illetve speciális karaktert (\*).



## **Elfelejtettem a jelszavam**

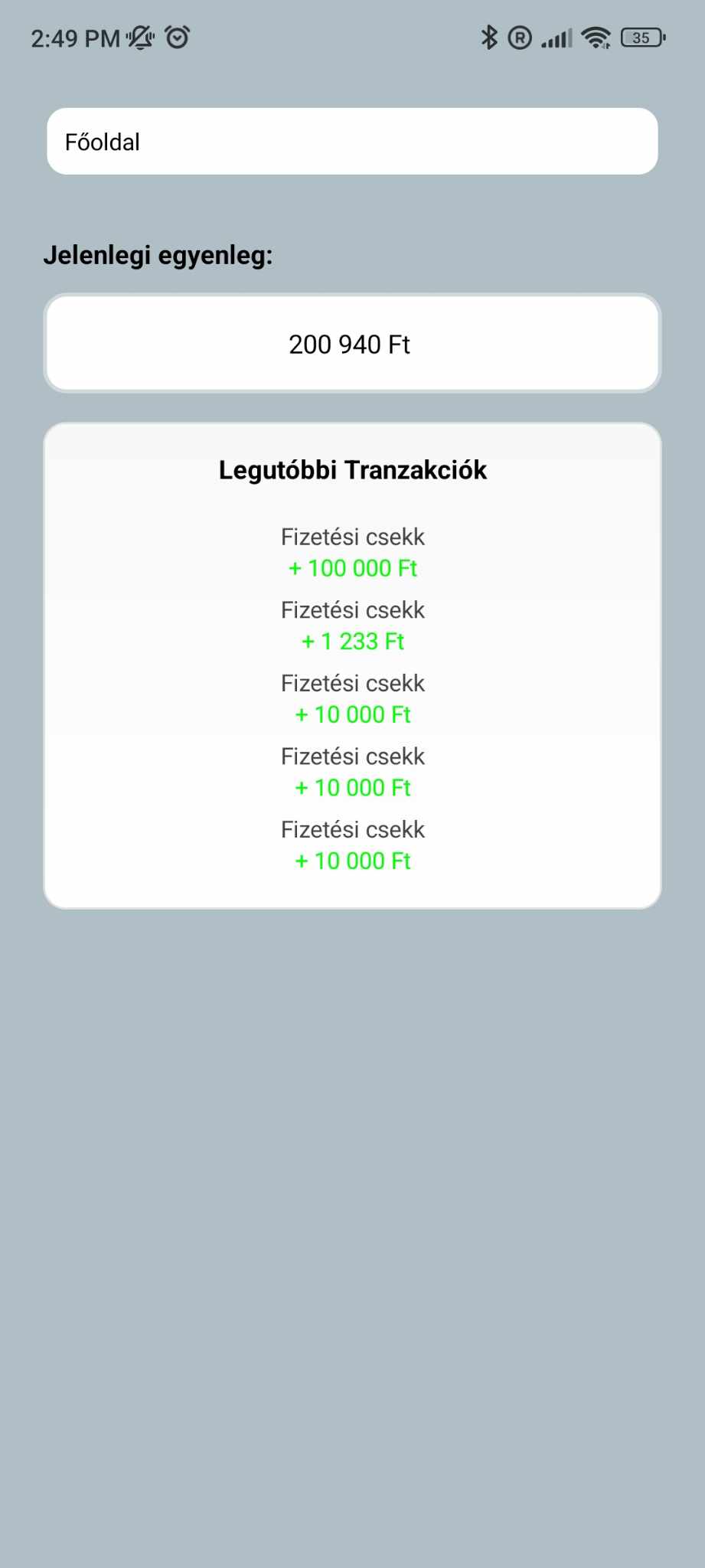
Ha a felhasználó rányom erre a szövegre, akkor átviszi egy másik oldalra, ahol ha beírja az e-mail címét, a program küld egy jelszó visszaállító e-mailt, melyet ha követ beírhat egy új jelszót, amivel utána be is jelentkezhet

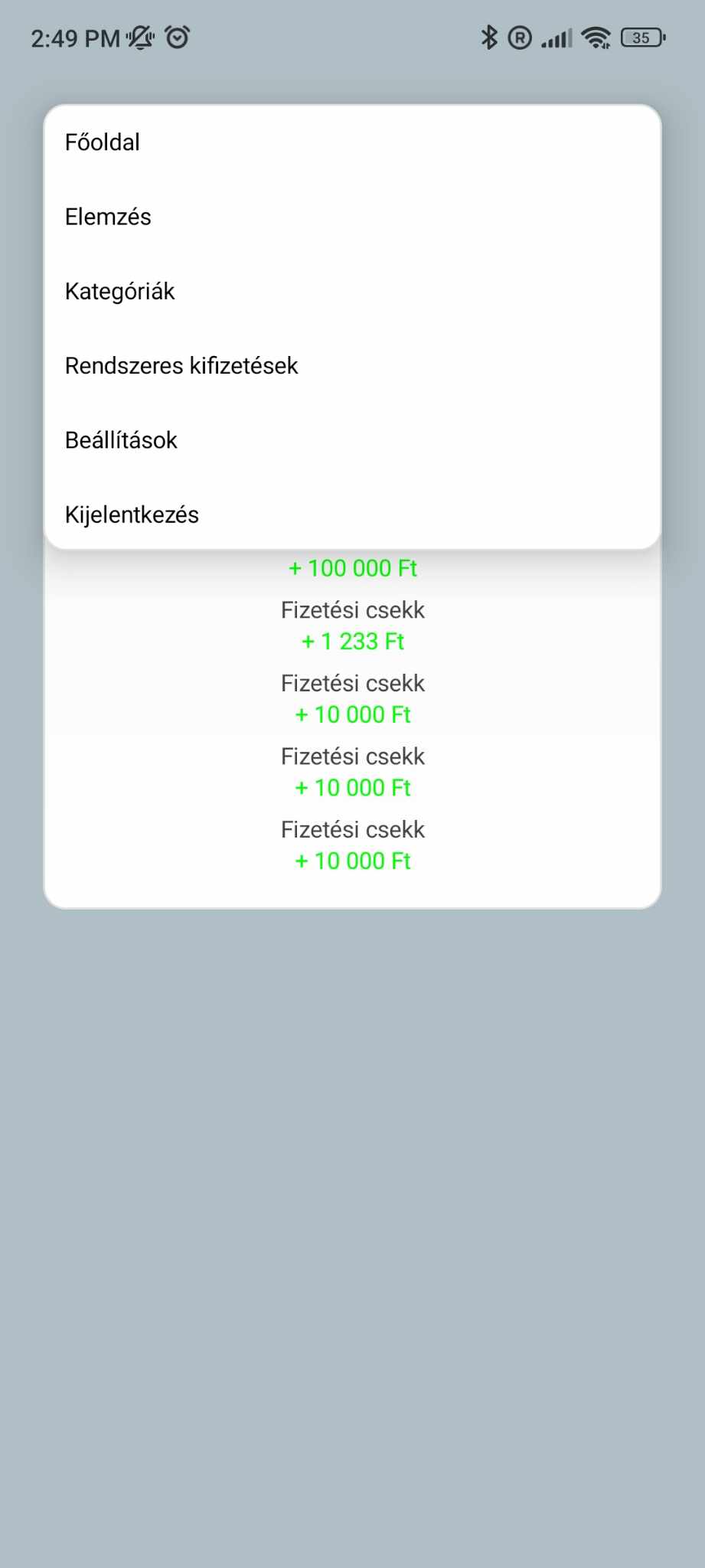


## **Főoldal**

Ha ide belép, 3 dolog fogadja önt. Egy lenyitható menü, ahol a program többi funkciója található, a fő összeg, amivel ön rendelkezik, és a legutóbbi tranzakciók listája. Ezek közül ön csak a lenyitható menüt tudja használni, a többi statikus információ megjelenítőként szolgál. Ha lenyitja a mezőt, hat darab opció közül választhat: Főoldal, Tranzakciók, Kategóriák, Rendszeres kifizetések, Beállítások és Kijelentkezés.

A fő oldal:



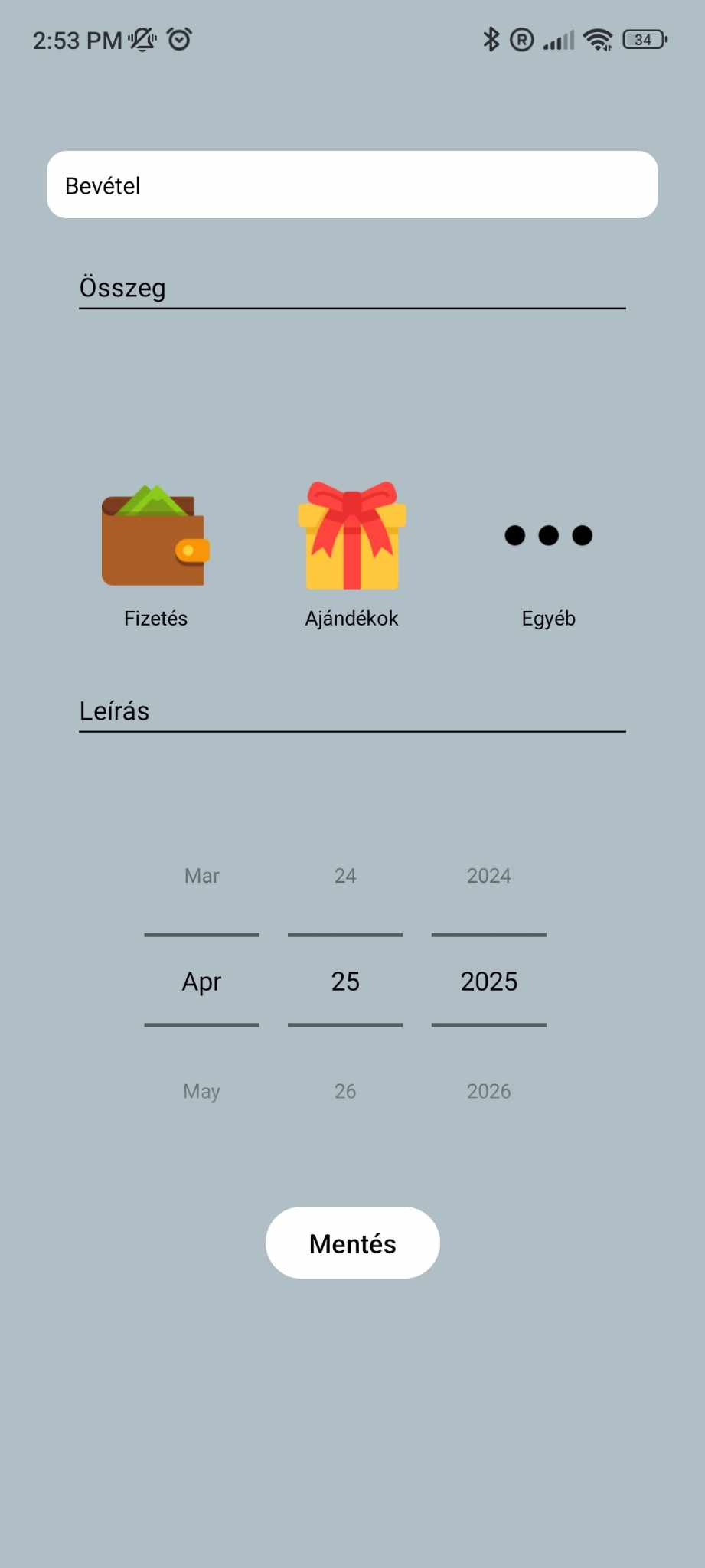
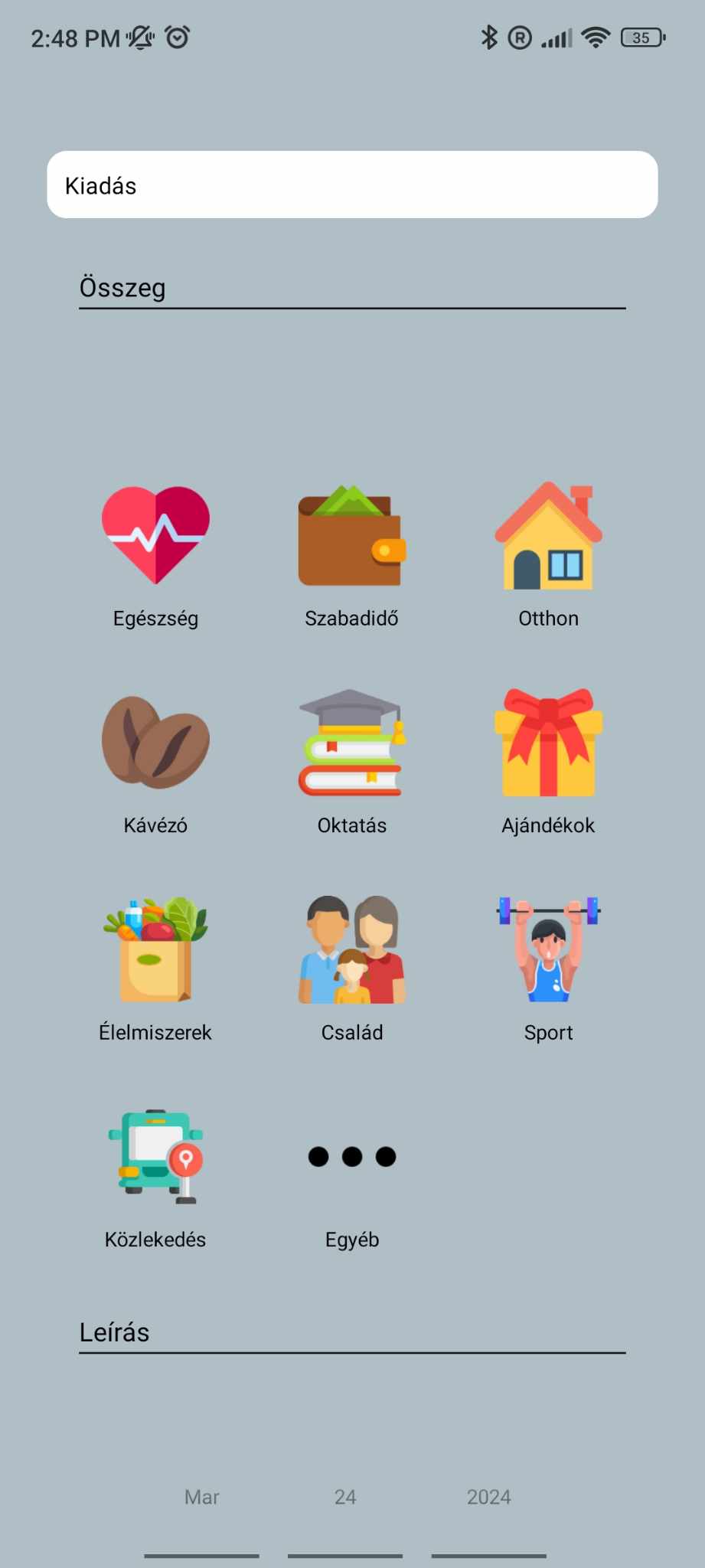


## **Elemzés**

Ez az első olyan oldal, ahol már a felhasználó elé több használható funkció is tárul. Először, amikor erre az oldalra lép, nem fog sok mindent látni, hisz még nem adott hozzá tranzakciókat az alkalmazáshoz, szóval az első gomb, amit ajánlok, hogy megnyomjon, az a Tranzakció hozzáadása gomb lenne. Ez a gomb átviszi magát egy másik fülre, ahol kiválaszthatja, hogy a tranzakció bevétel, vagy kiadás volt, majd ezután, hogy milyen kategóriába tartozik az. Ezután meg kell adnia az összegét ennek, és bár ez opcionális, adhat leírást is a tranzakciónak, hogy biztos tudja később is, hogy mit vett, vagy milyen fajta bevétele érkezett. Ha ezeket kitöltötte, és rányom a mentés gombra, akkor a program fog adni magának egy üzenetet, hogy sikeresen mentette a tranzakciót. Ha nem kíván több tranzakciót hozzáadni, akkor kiléphet ebből a fülből. Ezután láthatja, hogy megjelenik a diagramon a bevétele, illetve a kiadásai összegét, és az alatt kilistázva ezeket. A feliratok („Nap Hét Hónap Év Időszak”) azt a célt szolgálják, hogy milyen időszakban mutassa az elvégzett tranzakciókat. A nap az adott napit, a hét az ahetit, és így tovább. Az időszak fülre, ha rányom, feldob egy ablakot, ahol kiválaszthatja az általa kívánt időszak kezdeti napját, majd, ha rányom az „OK” gombra, akkor az időszak bezáró napját, és az ezen belül elvégzett tranzakciókat fogja a diagram kezelni, és a lista mutatni. A listában az elemre, ha rányom, lenyílik, és megjeleníti a leírását, majd, ha még egyszer rányom, akkor visszacsukja azt. A baloldalán az elemeknek található egy kuka ikon, amire ha rányom, törölheti a tranzakciót. Ilyenkor az összeg is úgy változik, ha bevételt töröl, levon, ha kiadást, hozzáad a fő összeghez.

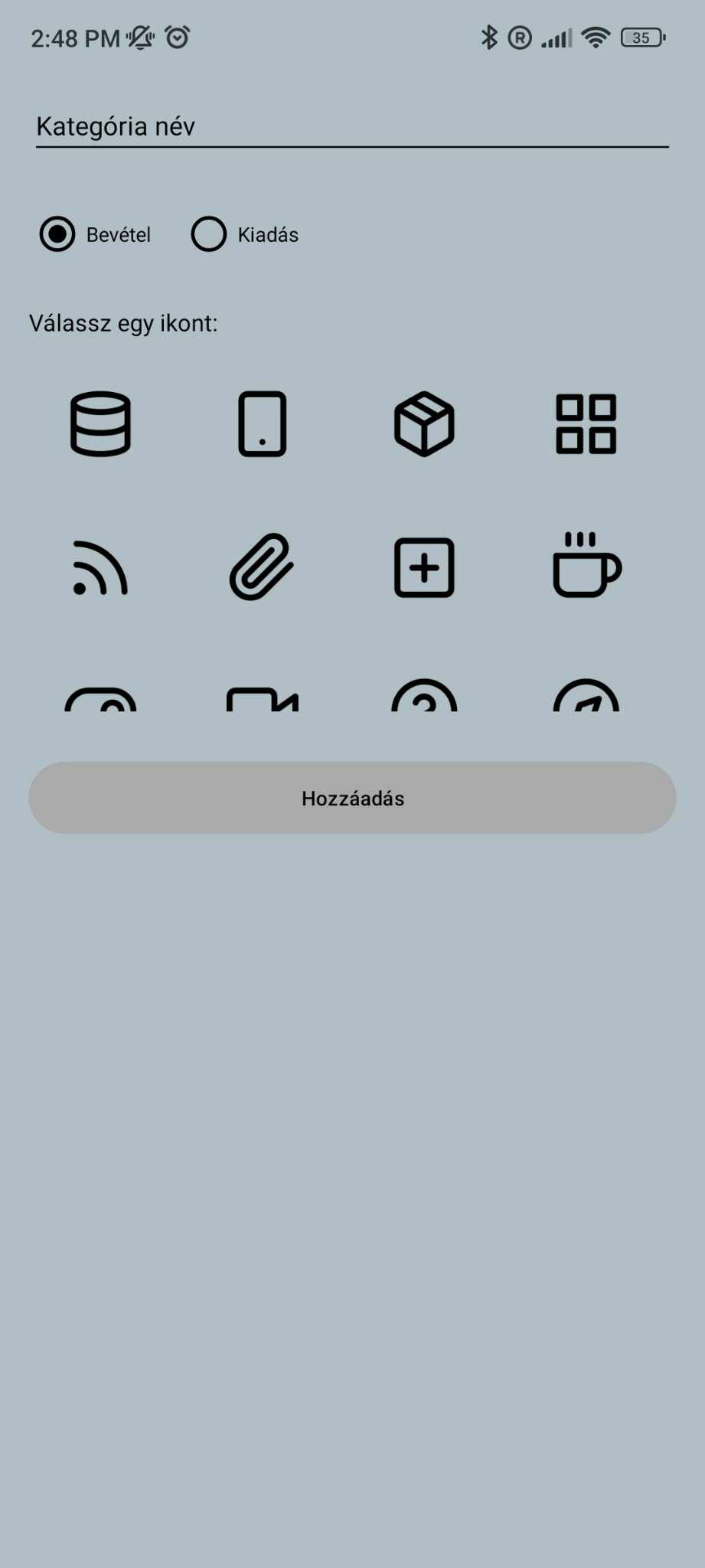
Az Elemzés fül, illetve a tranzakció hozzáadása fül:





## **Kategóriák**

Ez a fül az adott kategóriákhoz tartozó kiadást, vagy bevételt mutatja. Itt megtudja nézni a kategóriára rányomva, az ahhoz tartozó tranzakciókat, ha rányom egyre. Azon kívül felül található két nyíl, mellyel tud lépkedni a napok között, és ehhez mérten változni fog minden tranzakció, viszont, ha nem ilyen aprólékosan lépkedne, akkor a két nyíl között található dátumra rányomva talál egy listát, amiben manuálisan ki tudja választani, hogy melyik nap költségeit, vagy bevételeit szeretné megtekinteni, és itt is található a tranzakciók fülhöz hasonló felirat, („Nap Hét Hónap Év”) ami természetesen az adott időszak tranzakcióit mutatja meg magának. Kategóriát is hozzáadhat, előre megadott ikonokkal, de saját magától megadható leírással. Ezek később a tranzakció fülön is használhatóvá válnak.



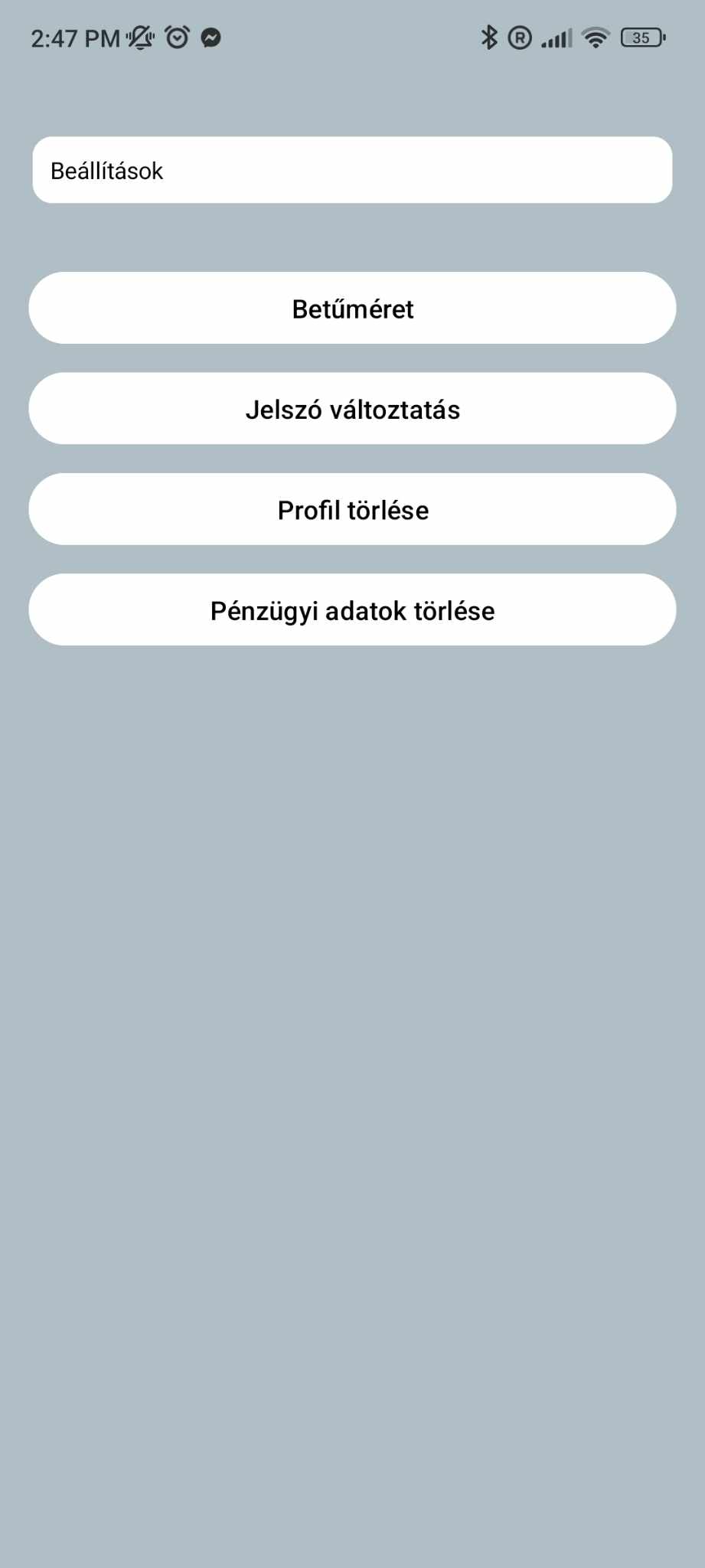
## **Rendszeres kifizetések**

Ezen a fülön a felhasználó a feliratkozásait kezelheti. Hozzáadhatja, törölheti, illetve változtathatja őket. Kiválaszthatja, hogy milyen gyakorisággal vonja le azt az összegből. Szükséges ehhez megadni a nevét a feliratkozásnak, hogy milyen gyakran vonja le, és a nevét, majd meg is jeleníti kilistázva ezeket önnek.



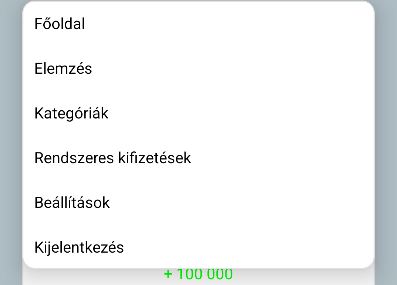
## **Beállítások**

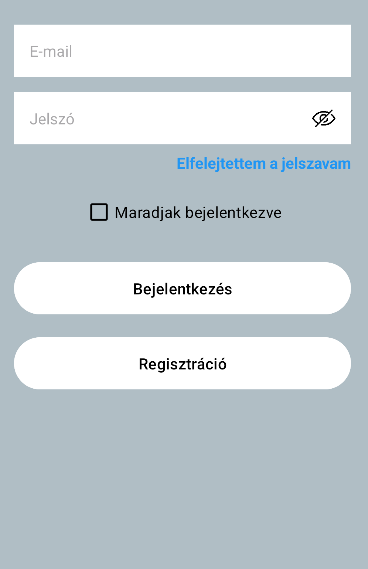
Ezen a fülön különböző gombok fogadják. Az alkalmazás betűméretét tudja változtatni, ha esetleg kicsinek találná az eredetit, vagy nagynak. Ha rányom, feldob egy párbeszéd ablakot, ahol ezt kitudja választani. A jelszó változtatás úgy működik, hogy ha rányom, kap egy e-mailt, amiben ha rányom a linkre tovább viszi a jelszóváltoztatás weboldalra, ahol ezt eltudja végezni. A profil törlése gomb törli a profilt ami kijelentkeztet, és törli profilját az adatbázisból. Ezután ugyanazokkal az adatokkal nem lesz képes belépni. A pénzügyi adatok törlése gomb egyértelműen törli az összes, az ön profiljához rendelt adatot, amit megadott, az adatbázisból, azonban a felhasználója nem törlődik.



## **Kijelentkezés**

Ez egyszerűen kilépteti a programból. Ezután a program visszadobja magát a bejelentkezés oldalra, ahol, ha visszaszeretne jelentkezni, akkor nyugodtan megteheti azt.

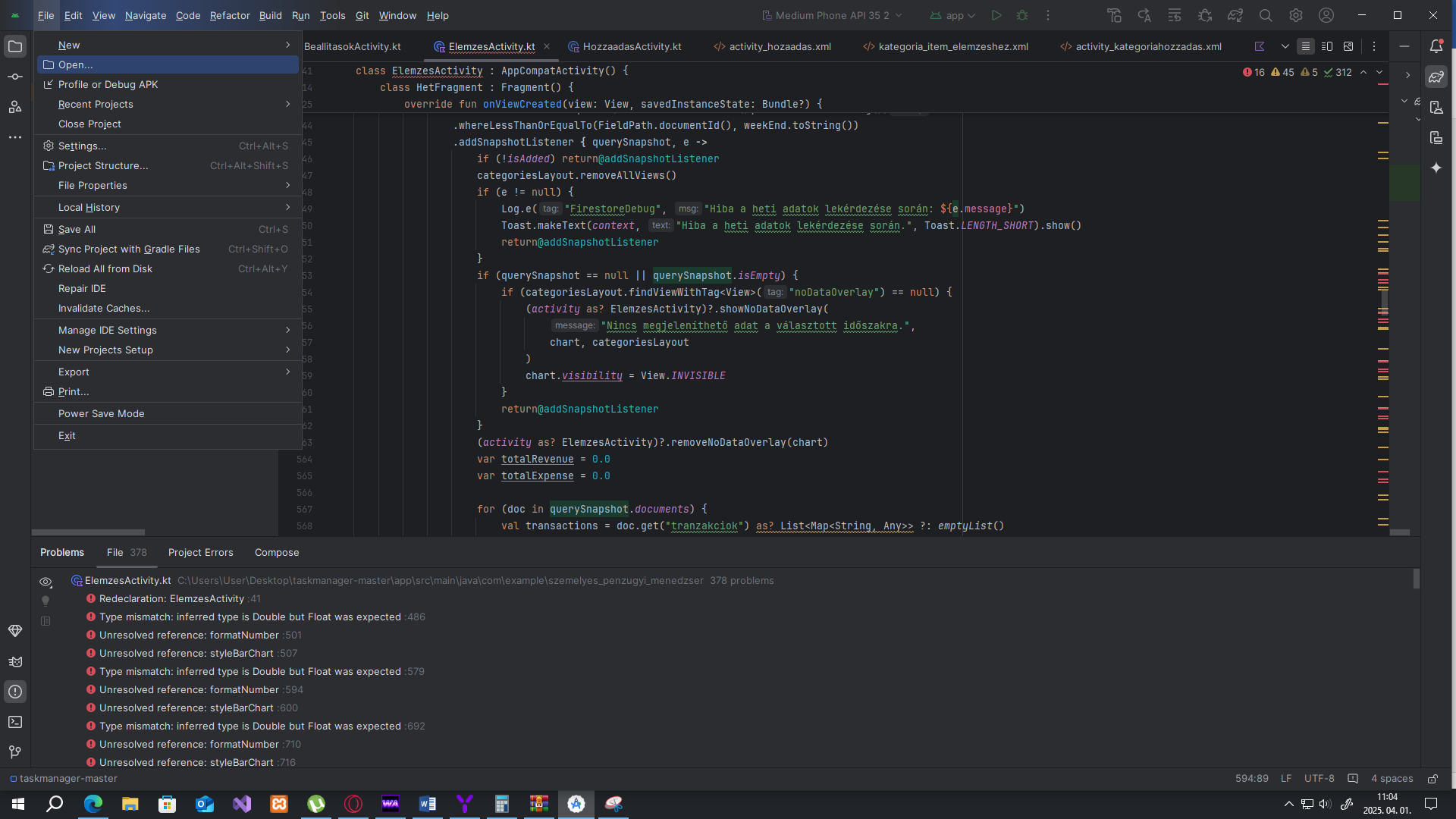


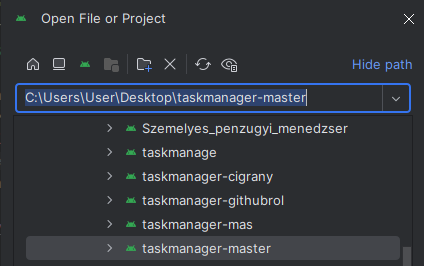


# **A program funkciói fejlesztők részére**

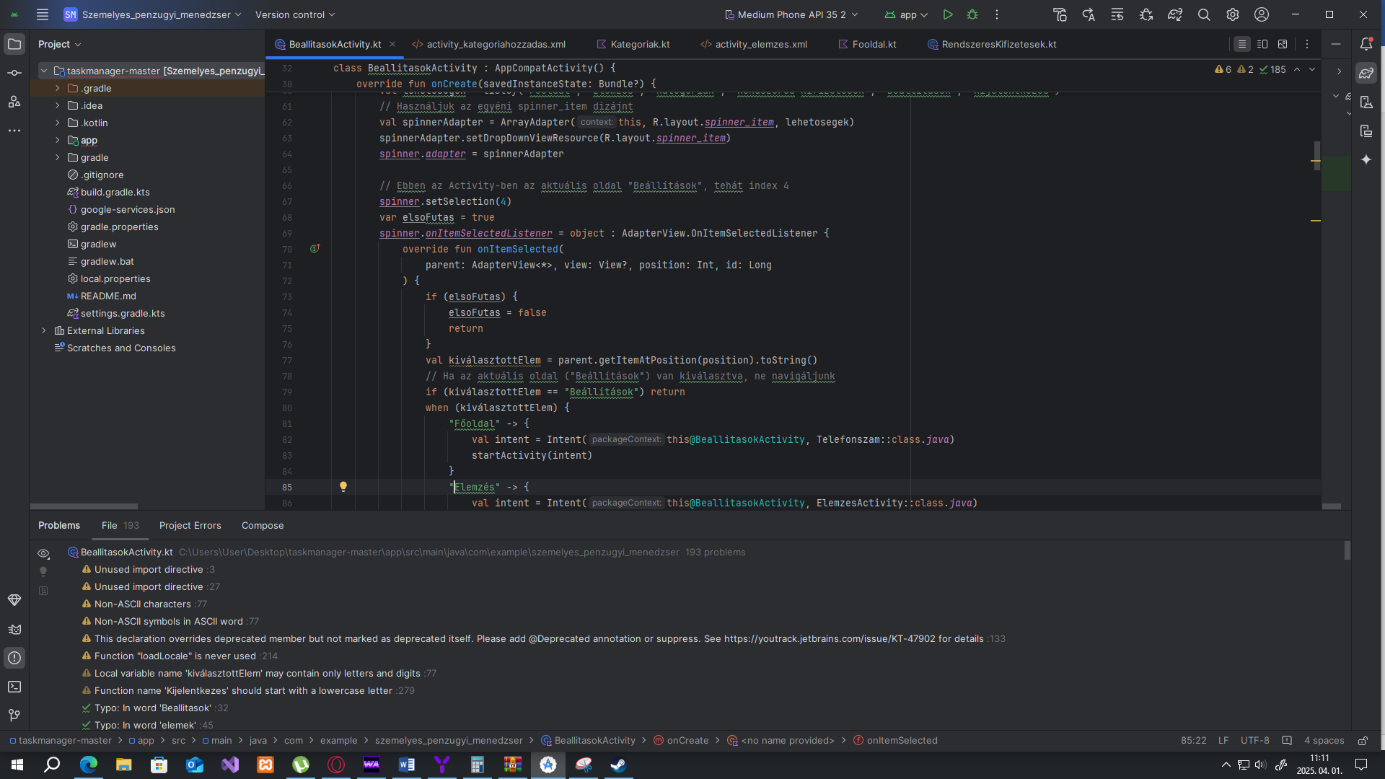
## **Kezdés**

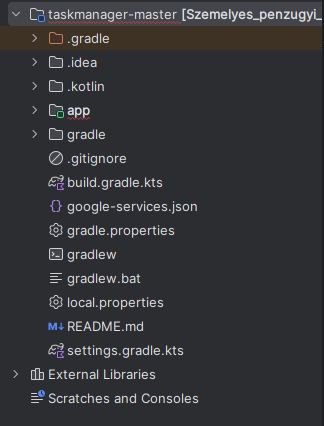
Azért választottuk a Kotlin programozási nyelvet, mivel szerettünk volna megtanulni Androidos alkalmazást fejleszteni, és ez volt az első, ami szembejött velünk, és meg is tetszett számunkra. A program kódszintű futtatásához szükséged lesz az Android Studiora, ahol a menüben a File lenyíló menüben az Open gombra nyomva kiválasztja a projektet. Nálad a projekt neve eltérhet.





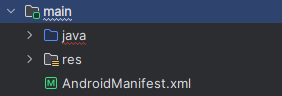
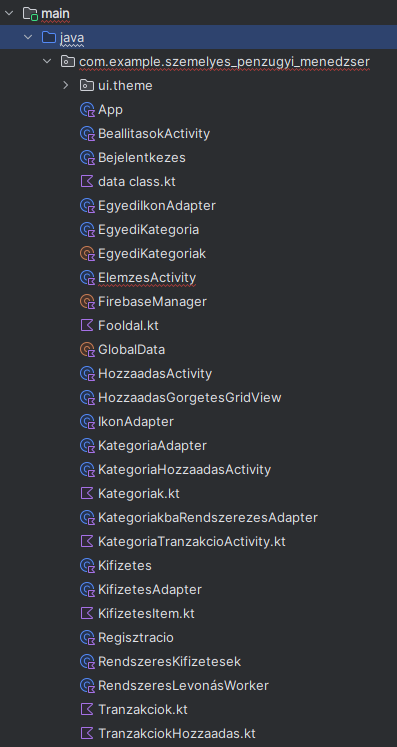
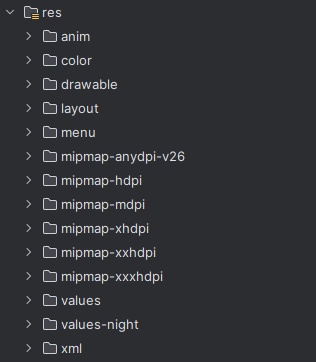
## **A projekt első látásra**

A projekt első megnyíláskor az azon dolgozott oldallal fog megjelenni, minden almappával összecsukva. Eleinte megtalálni mindent nehéz lehet, ezért szeretnék segíteni, és végig vezetni, hogy mit hol találsz.



### **Különböző részek elérése**

Az előző oldalon található utolsó képen való elemeket listájára lesz szükségünk, azon belül is csak az app mappára. Ha azt lenyitjuk megint almappákat kapunk, illetve más fájlokat. Innét az src mappára lesz szükségünk, arra kell nyomni, utána pedig a mainre. Ha ezzel megvagyunk, akkor 2 fontos mappát látunk, ami tartalmazza a programkódot, a „java” és a „res”. Az AndroidManifest.xml minden Activity meghívását, illetve fő engedélyeket tartalmazza. Kérlek ezt ne változtasd. A java mappára nyomva egyből elénk tárulkozik a kódokat tartalmazó különböző funkciókat ellátó fájlok. A res mappa viszont megint almappákat jelenít. Ezek tartalmazzák a „fizikai” elemeket, itt lehet a designt változtatni, vagy képeket a programhoz rendelni, esetleg színeket, és ehhez hasonlókat csinálni.



## **Különböző részek szerepe**

### **Fooldal.kt**

#### **Bevezető**

A Fooldal.kt nevű aktivitásának működését mutatja be. A felület a felhasználók aktuális pénzügyi egyenlegének megjelenítésére, módosítására, valamint az utolsó tranzakciók listájának megtekintésére szolgál.



#### **UI Komponensek**

* aktualisPenzTextView: Az aktuális egyenleget mutató címke.
* AktualisPenzEditText: Szövegmező, amelyben a felhasználó módosíthatja az aktuális egyenleget.
* Spinner: Navigációs menü, amellyel különböző képernyőkre navigálhat a felhasználó.
* recentTransactionsLayout: Egy dinamikusan feltöltött lista az utóbbi 5 tranzakcióról.

#### **Funkcionális áttekintés**

##### Egyenleg Valós Idejű Figyelése (listenToBalance)

* Ez a funkció figyeli a Firebase Firestore-ban tárolt felhasználói dokumentum aktualisPenz mezőjét. Ha ez az érték megváltozik, automatikusan frissíti a felhasználói felületet (PenzosszegFrissites).

##### Egyenleg Frissítése (saveBalanceToFirestore)

* Ha a felhasználó módosítja az aktuális pénzmennyiséget a beviteli mezőben, akkor az adat automatikusan mentésre kerül a Firestore adatbázisba.
* Számformázás

##### Az összeg formázására kétféle metódus is alkalmazásra kerül:

* 3 számjegyenként szóközök beillesztése.
* Magyar formátum alkalmazása (tizedesjel: pont, ezres elválasztó: szóköz).

##### Navigációs Spinner (setupSpinner)

* A spinner segítségével a felhasználó navigálhat az alkalmazás különböző részei között. Ha új oldalra navigál, az aktuális oldalt elhagyja, és a választott aktivitás indul el.

##### Utolsó Tranzakciók Betöltése (loadRecentTransactions)

* Az utóbbi 5 tranzakciót kérdezi le a Firestore adatbázisból, majd rendezi őket dátum szerint csökkenő sorrendben. Minden tranzakció egyedi, kategória, összeg és típus alapján kerül megjelenítésre (createTransactionView).

### **ElemzesActivity.kt**

#### **Bevezető**

Az ElemzesActivity a pénzügyi adatok elemzésére és grafikus megjelenítésére szolgál. A felhasználó különböző időszakokra vonatkozó bevételeit és kiadásait tekintheti meg oszlopdiagramon, valamint részletes tranzakciós listán keresztül.

#### **UI Komponensek**

* Spinner: Navigációs menü a képernyők közötti navigációhoz.
* BarChart: Az MPAndroidChart könyvtár segítségével megjelenített oszlopdiagramok
* LinearLayout (CategoriesLinearLayout): Dinamikusan létrehozott tranzakciós lista megjelenítésére.

#### **Funkcionalitás áttekintése**

##### Navigáció és Fragmentek kezelése

**Az activity lehetőséget biztosít a felhasználónak az alábbi időszakok elemzésére:**

* Nap
* Hét
* Hónap
* Év
* Egyéni időszak

##### Adatok lekérdezése Firestore-ból

* Az activity az aktuális Firebase-felhasználóhoz tartozó tranzakciókat kérdezi le, majd összesíti és megjeleníti őket diagramokon, illetve listákon, időszaki bontás szerint rendezve.

##### Diagramok stílusa

* A diagramok egységes stílust kapnak (zöld és piros színezés, megfelelő címkék).

##### Tranzakciókezelés

* Tranzakció törlése: Frissíti az adatbázist és a fő egyenleget.
* Tranzakció hozzáadása: Frissíti az adatbázist és a fő egyenleget.

##### Adatok hiányának kezelése

* Amennyiben egy adott időszakra nincs adat, az alkalmazás egy egyértelmű üzenettel és ikonnal jelzi azt.

##### Specifikus Fragmentek

**NapFragment, HetFragment, HonapFragment, EvFragment, IdoszakFragment**

* Mindegyik fragment egy adott időszakra szűri a tranzakciókat, majd rendezi, összegzi, és megjeleníti őket.

**Tranzakciók részleteit a createTransactionView metódus készíti el.**

### **Rendszereskifizetesek.kt**

#### **Bevezető**

A RendszeresKifizetesek a rendszeres kifizetések kezelésére szolgál. A felhasználók rendszeres kiadásokat rögzíthetnek, módosíthatnak, és törölhetnek, melyeket az alkalmazás automatikusan kezel, és időszakosan levon a fő egyenlegből.

****

#### **UI Komponensek**

* RecyclerView: A rendszeres kifizetések listájának dinamikus megjelenítése.
* EditText: Kifizetés nevének és összegének megadására.
* Spinner: Navigációs menü és időszakválasztó a kifizetések gyakoriságához.
* Button: Új kifizetés hozzáadására és meglévő kifizetés törlésére.

#### **Funkcionalitás**

##### Rendszeres Kifizetések Kezelése

* Hozzáadás: Lehetővé teszi új kifizetések rögzítését, amelyek automatikusan hozzáadódnak a Firebase Firestore adatbázishoz.
* Törlés: Kifizetések dokumentumait lehet törölni, miközben az egyenleg nem változik.

##### Egyenleg Kezelése

* Az alkalmazás lekéri és formázottan megjeleníti a felhasználó aktuális egyenlegét.

##### Realtime Adatok

* A Firebase Firestore segítségével valós idejű frissítések jelennek meg a kifizetések listájában.

##### WorkManager Integráció

* Az Activity a WorkManager segítségével napi rendszerességgel végrehajtja a kifizetések automatikus levonását az egyenlegből.

##### Navigáció és Kijelentkezés

* Navigáció: Spinner segítségével lehet navigálni az alkalmazás különböző részei között.
* Kijelentkezés: Firebase Auth segítségével történik, amely törli a felhasználó mentett állapotát.

##### Hibakezelés

* Az Activity megfelelően kezeli az adatbázis műveletek során felmerülő hibákat, és logolja az eseményeket, miközben felhasználói visszajelzést nyújt.

### **Kategoriak.kt**

#### **Bevezető**

A Kategoriak Activity kategóriák szerint csoportosítva jeleníti meg a felhasználó pénzügyi tranzakcióit. Lehetővé teszi az időszakok szerinti szűrést és a kategóriák részletes megtekintését.

#### **UI Komponensek**

* ListView: A kategóriák összegzett adatainak megjelenítése.
* Spinner: Navigációs menü, valamint időszakválasztó a tranzakciók szűrésére.
* Button: Új kategória hozzáadására.
* ImageButton: Időszakok közötti gyors navigációra.

#### **Funkcionalitás**

##### Tranzakciók lekérdezése és összegzése

* Az alkalmazás Firebase Firestore adatbázisból kérdezi le a tranzakciókat adott időszakokra, majd összegzi és kategóriák szerint jeleníti meg.

##### Időszakválasztás

* A felhasználók különböző időszakokat választhatnak (nap, hét, hónap, év, egyéni), melyek alapján a tranzakciók szűrése történik.

##### Kategóriák kezelése

* Beépített és egyedi kategóriák támogatása: Lehetővé teszi felhasználói kategóriák hozzáadását.
* Kategória ikonok és nevek normalizálása: Biztosítja a konzisztenciát a kategóriák kezelésében.

##### Navigáció és Kijelentkezés

* Spinner segítségével navigálhatóak az alkalmazás különböző részei, továbbá támogatja a felhasználó kijelentkezését Firebase Auth integrációval.

##### Segédfüggvények

* Számformázás: Számok megjelenítése 3 számjegyenként szóközzel.
* Dátumkezelés: Specifikus időszakok kezdő- és záródátumának meghatározása.

##### Hibakezelés

* A Firestore lekérdezések során megfelelően kezeli a hibákat, üres adatok esetén felhasználóbarát üzeneteket jelenít meg.

### **BeallitasokActivity.kt**

#### **Bevezető**

A BeallitasokActivity az alkalmazás beállításokat kezelő felülete, amely lehetőséget biztosít a felhasználók számára az alapvető fiók- és megjelenítési beállítások konfigurálására. Támogatja a betűméret és a nyelv kiválasztását, a jelszó módosítását, a fiók törlését, valamint a pénzügyi adatok teljes eltávolítását.



#### **UI Komponensek**

**Button:**

* Betűméret beállítása (betumeret\_button)
* Jelszó módosítása (jelszo\_button)
* Profil törlése (profil\_torles\_button)
* Pénzügyi adatok törlése (penzugyi\_adatok\_torles\_button)

**Spinner:**

* Navigációs menü az alkalmazás fő funkciói között.

**AlertDialog:**

* Műveletek visszaigazolásához (pl. jelszóváltás, törlés).

#### **Funkcionalitás**

##### Betűméret beállítása

**Három előre definiált betűméret közül választhat a felhasználó:**

* Kicsi (12sp)
* Közepes (16sp, alapértelmezett)
* Nagy (20sp)

**Az érték elmentésre kerül a SharedPreferences segítségével, majd az Activity újratöltésre kerül az új beállítások alkalmazása érdekében.**

##### Jelszó módosítása

* A művelethez újrahitelesítés szükséges.
* A felhasználónak meg kell adnia a jelenlegi jelszavát.
* A FirebaseAuth újrahitelesítéssel ellenőrzi az adatokat.
* A sikeres hitelesítés után a updatePassword() függvény segítségével kerül sor a jelszó módosítására.

##### Profil törlése

**A teljes Firebase fiók törlésére is lehetőség van:**

* Újrahitelesítés szükséges.
* A sikeres hitelesítés után a delete() metódus véglegesen eltávolítja a fiókot.
* A törlés után kijelentkeztetés és visszairányítás történik a bejelentkezési képernyőre.

##### Pénzügyi adatok törlése

**A felhasználó összes napi pénzügyi tranzakciója (nap gyűjtemény) törlésre kerül:**

* A dokumentumok törlése Firestore batch művelettel történik.
* A művelet nem visszavonható és azonnali hatállyal végrehajtásra kerül.

##### Segédfüggvények

**applyFontSizeToCurrentActivity()**

* Frissíti a teljes Activity betűméretét a felhasználói beállítás szerint.

**updateTextViewsFontSize()**

* Rekurzívan módosítja az összes TextView betűméretét a megadott ViewGroup hierarchiában.

**setLocale(languageCode: String)**

* Frissíti az alkalmazás nyelvét és elmenti a SharedPreferences-be.

**kijelentkezes()**

* Kijelentkezteti a felhasználót, törli a bejelentkezési állapotot és átlépteti a Bejelentkezes képernyőre.

**torolPenzugyiAdatokat()**

* Végigmegy a felhasználó nap Firestore gyűjteményén és törli az összes dokumentumot batch művelettel.

##### Hibakezelés

**Toast üzenetek:**

* Minden fontosabb felhasználói interakció után visszajelzés történik (Toast.makeText()).

**Log üzenetek:**

* A Firestore műveleteknél részletes hibalog (Log.e/Log.d) segíti a hibakeresést.

**Validáció:**

* A jelszó- és adatmezők üres állapotát külön ellenőrzi az alkalmazás.

### **Még hozzáadható, vagy változtatható funkciók:**

A program megjelenését lehetne tökéletesíteni, mivel sem én, sem pedig a projekt társaim sem vagyunk a legszebb dizájnerek sajnos. Emellett még a beállításokhoz lehetne tenni egy olyan gombot, ahol nyelvet lehet választani, hogy szélesebb körben lehessen használni a programot, nem pedig csak magyarul.

## **Fejlesztők számára bejelentkezés teszteléshez**

Az e-mailhez való bejelentkezés:

[szemelyespenzugyimenedzser@gmail.com](mailto:szemelyespenzugyimenedzser@gmail.com)

Jelszó: szemelyespenzugyimenedzser123

A programban ugyanezzel az e-mail címmel tud bejelentkezni, a jelszó pedig: szemelyespenzugyimenedzser123\*

Ezzel az e-mail fiókkal hozzáférése is van az adatbázishoz, ha esetleg szeretné azt megnézni, illetve kezelni, de ha változtatna rajta akkor előtte jelezzen számunkra az e-mailünkön keresztül:

botond.bogdan317@gmail.com

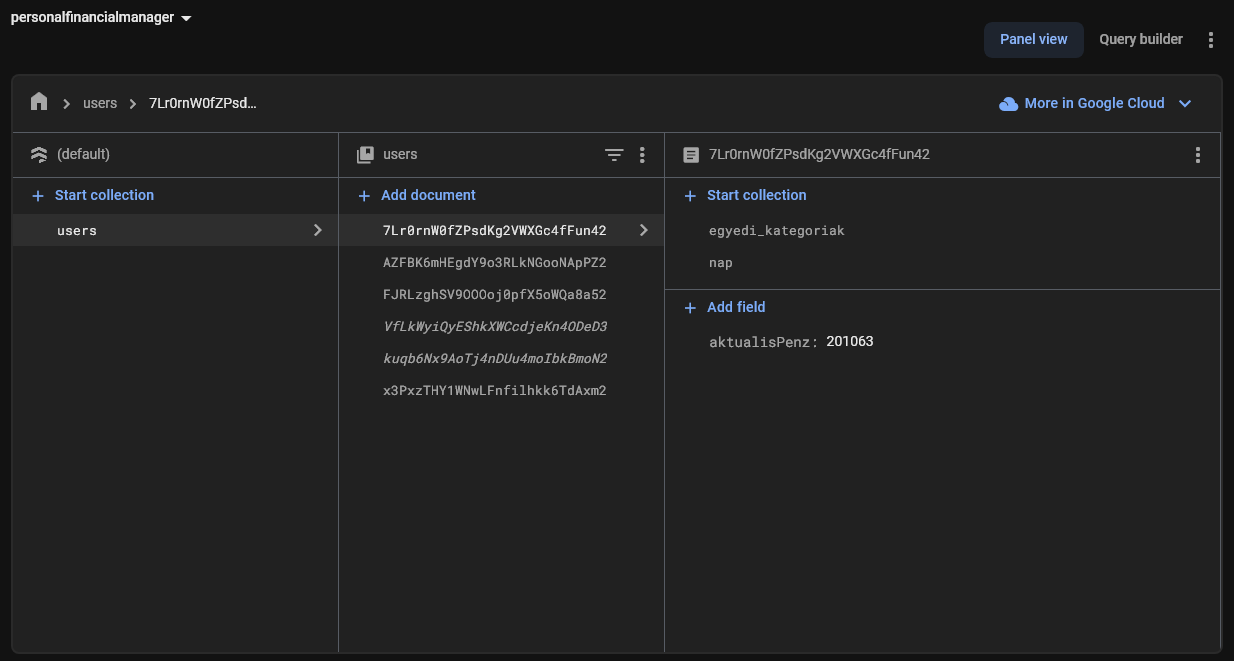
## **Az adatbázis**

Az általunk egyik legkönnyebben, és legpraktikusabban integrálható adatbázis, az egy felhőalapú, Google Firestore Firebase típusú adatbázis. Mivel felhőalapú, ezért könnyedén kezelhettük a programunkban, és könnyedén tudtuk bármelyik részéhez integrálni.

Az adatbázis struktúrája:

* Elsősoron van egy collectionünk, amely a „users” névvel rendelkezik
* Ezen belül minden felhasználó egy felhasználói azonosítót kap, ezt az adatbázis generálja, amikor valaki regisztrál
* Az adott felhasználóhoz van egy mező rendelve, mely tárolja a fő összegét, mellyel a program számol
* És collectionök, melyek közül az egyik minden felhasználó napi tranzakcióit menti el
  + A napi rendben mentett adatok alapján számítja ki a program a többi szükséges műveletet.
  + Ezen mentett adatok dátum formában vannak elmentve
* Ugyanitt található az egyedi\_kategoriak collection is, ami a felhasználó által hozzáadott egyedi kategóriákat menti el

Az adatbázis felépítése ránézésre:



## **A tesztelés folyamata**

Az alkalmazás fejlesztés közben folyamatos tesztelésen esett át az Android Studioban található emulátor segítségével. Akárhányszor új funkciót adtunk hozzá, vagy egy meglévőt frissítettünk, közvetlen utána teszteltük, hogy működik-e, és hogy jól is tegye azt.

### **Gyakran előforduló hibák**

A program fejlesztése során nagyon gyakran futottunk bele verzióhibákba, hogy esetleg újabb verziójú API-t kéne használni, vagy illetve implementálni az APIhoz a build.gradle.kts-ben (alkalmazás szintű gradle fájlban). Ezt természetesen ezek egymáshoz való illesztése, és kompatibilitásának megoldásával állítottuk helyre.

Másik gyakori hiba a Tranzakciók fülön volt található, ahol a kilistázott elemek sosem akartak helyes sorrendben megjelenni, illetve ha mégis, akkor a kódban történő változtatás után megint rosszul történt meg. Ezt végül úgy oldottuk meg, hogy minden változtatást elvégezve az elemek sorrendjét a legvégén változtatva, tehát a kész kód után adtuk hozzá ezt a kódrészletet, és utána ez is működött, azonban ez emiatt instabil, és igényel még foglalkozást.

A program adatbázisa eleinte máshogy volt felépítve, mivel azt nem gondoltuk át teljesen az elején, ezért gyakran kellett a végső formájához igazítani a kódot, hogy rendesen tudja lekérni az adatok, vagy azokat frissíteni az adatbázisban.

### **Más hibák**

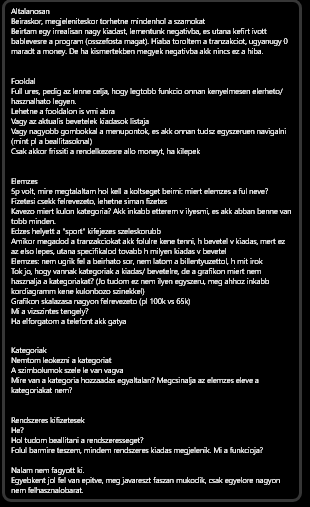
Nagyon nagy számok esetén a program nem tudta kezelni őket. Ez ilyen 100 milliárdos nagyságrendben kezdett el történni, és ez annak az oka volt, hogy az összeget rossz változó kezelte lekérdezés után (int), ezt átállítottuk egy nagyobb számok kezelésére is képes típussá (double), és utána nem állt fent ez a probléma.

A rendszeres kifizetések ablakon az adott időszakot csak kiválasztani lehetett, azonban a levonás nem történt meg, ezért annak a function-jét ki kellett javítanunk, ami pedig úgy történt, hogy konstans tesztelés mellett változtattuk a kódrészletet megfelelésünk szerint, míg nem működött.

### **Segítség a tesztelés során**

Segítséget is kaptunk közeli ismerősöktől, akiket megtudtunk kérni, hogy használják, terheljék az alkalmazást, és jelezzenek vissza számunkra.

Egyik segítőnk:



# **Összefoglalás**

A programunk szerintünk összességében első általunk nulláról épített, saját magunk szinte nulláról való tanításával, és okításával jól sikerült. Nem mondanám tökéletesnek természetesen, hiszen van mit tovább fejleszteni rajta, akár a meglévő dolgok frissítésével, vagy teljesen új dolgok hozzáadásával, azonban én ahhoz képest meg vagyok vele elégedve, ezért remélem másoknak is, ha nem is annyira, mint nekem, de tetszeni fog, és értékelhetőnek találja.

# **A program a jövőben**

Mivel jelenleg időnk hiányában fejleszteni annyira nem vagyunk képesek, mint tanév közben, ezért nem biztos, hogy rendszeres foglalkozást fog kapni, azonban én használni fogom, mivel hasznosnak találom, és szükségem is van rá, és ha fogok tudni vele foglalkozni, akkor fogok is. Nekem aktívan használt alkalmazásommá vált, mivel nagyon is jó, hogy tudom valahol rendszeresebben követni a költekezésem, mint a banki applikációmban (Erste George), és ahol egyértelműbbek is a dolgok.