

UNIVERSITATEA POLITEHNICĂ DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL DE CALCULATOARE



PROIECT DE DIPLOMĂ

Aplicație android pentru gestiunea finanțelor personale

Cezar Gabriel Dimoiu

Coordonator științific:

Conf. dr. ing. Popescu Cornel

BUCUREȘTI

2021

CUPRINS

1	Introducere	1
1.1	Structura lucrării	2
2	Analiza Cerințelor / Motivație	3
3	Fundamentare teoretică	4
3.1	Android	4
3.2	Android Studio	5
3.3	AnyChart	6
3.4	Firebase	6
4	Soluția Propusă	7
5	Detalii de implementare	9
5.1	Autentificare și înregistrare	9
5.2	Structura bazei de date	10
5.3	Pagina principală	12
5.4	Pagină categorii	14
5.5	Pagină Today	16
5.6	Paginile This week și This month	17
5.7	Pagină analytics	19
5.8	Pagină transactions	23
5.9	Pagina My Account	24
5.10	Implementarea interfeței	25
6	Evaluare și îmbunătățiri viitoare	26
7	Concluzii	28

Bibliografie	30
Anexe	31
Anexa A Extrase de cod	31

SINOPSIS

Managementul banilor este un subiect puternic dezbătut de generația tânără, dar nu numai. Cu această aplicație am dorit să ofer un mod eficient de gestiune a finanțelor, prin care putem să ne stabilim lunar bugete pentru diverse categorii și prin care orice cheltuială să fie înregistrată, să știm exact câți bani am cheltuit și câți bani avem rămași pentru fiecare categorie. Mai mult, aplicația oferă și o analiză sumară despre cum ne-am gestionat resursele din ultima zi, săptămână sau lună, însoțită de grafice explicite. În urma unui studiu de caz, aplicația s-a dovedit a fi una intuitivă, ușor și rapid de folosit pentru orice categorie de vârstă, făcând concurență aplicațiilor actuale de pe piață, cu mențiunea că există unele funcționalități unde acest lucru poate fi îmbunătățit.

ABSTRACT

Nowadays, money management is an actual subject for the younger generation, but not only for them. With this application, I wanted to offer an efficient way for managing the money, through we can set monthly budgets for various categories and a way to record any expenditure, to know exactly how much money have we spent and how much we have left for every category. Moreover, the application offers a summary analytic about how we managed our resources for the current day, week and month, along with a visual representation. Following a cast study, the application turned out to be intuitive and easy to use for every age category, being relevant between actual real market applications, mentioning that there are some features where it can be improved.

1 INTRODUCERE

În ultimul timp am început să fiu interesat de domeniul finanțelor, respectiv educația financiară. În fiecare curs urmat a existat ideea de a îți monitoriza cheltuielile, de a-ți împărți bugetul în categorii și de a respecta bugetul respectiv pentru a avea o viață echilibrată din acest punct de vedere. Am început să-mi creez un obicei în a-mi organiza finanțele într-un tabel excel, dar am ajuns la concluzia că acest instrument nu era chiar ușor de folosit. Trebuia să ma loghez pe google drive, să caut documentul respectiv, să fiu atent pe ecranul mic al telefonului ce celula selectez și ce date completez. Astfel, am încercat să gasesc un mod eficient pentru a îmi ține evidența cheltuielilor și după un studiu de caz realizat cu ajutorul câtorva colegi de la facultate, soluția propusă a fost dezvoltarea unei aplicații de mobil pentru gestiunea finanțelor personale. Principalele funcționalități pe care trebuia să le îndeplinească aplicația ar fi fost:

- Împărțirea venitului pe categorii (Transport, Mâncare, Casă, Amuzament, Educație, Donație, Personal, Economii, Investiții, Altele)
- Vizualizarea bugetului rămas și a cheltuielilor totale
- Adăugarea unei cheltuieli pentru o anumită categorie
- Editarea unei cheltuieli din lista de cheltuieli
- Vizualizarea cheltuielilor pentru ziua, săptămâna sau luna în curs
- Accesarea unui istoric al cheltuielilor pentru un interval de timp selectat
- Vizualizarea unor statistici bazate pe buget și pe cheltuielile în curs

Principala nevoie la care raspunde aplicația este aceea de a înregistra cu ușurință o cheltuială, în doar câteva secunde. Din meniul principal se merge în view-ul "Today", se apasă singurul buton de adaugare, se alege categoria, se completează suma și o notă și apoi se salveaza. Pentru un utilizator experimentat, acest proces dureaza aproximativ 10 secunde, iar pentru un utilizator începător, între 17-27 de secunde.

Spre deosebire de alte programe de acest tip, aplicația propusă este gratis de folosit, nu conține reclame, iar codul este open-source publicat pe GitHub¹. Modalitatea de funcționare a acesteia este prezentată în continuare:

Când utilizatorul rulează pentru prima dată aplicația, este redirecționat la pagina de logare. Dacă nu are cont, poate merge la pagina de înregistrare pentru a își crea unul. Utilizatorul rămâne logat chiar dacă aplicația se închide, pentru ca la următoarea pornire a aplicației să nu fie nevoie să-și introducă din nou credențialele. După ce se loghează, utilizatorul vede meniul principal unde vede bugetul alocat dar și suma cheltuielilor pentru ziua, săptămâna și luna în

¹© <https://github.com/cezardimoiu/Money-management-app>

curs, precum și banii ramași, iar restul paginii conține 6 butoane principale, cu următoarele funcționalități:

- "Categories" - unde utilizatorul își poate adauga bugetul pentru categorie
- "Today" - istoric pentru ziua în curs și meniul de unde își poate înregistra o cheltuială
- "Week" - istoric pentru săptămâna în curs
- "Month" - istoric pentru luna în curs
- "Analytics" - aici se pot vedea statistici pentru ziua, săptămâna sau luna în curs, însoțite de grafice
- "Transactions" - aici se poate vedea istoricul pentru un interval de timp selectat

Aplicația a fost dezvoltată în Android Studio folosind ca baza de date serviciul Firebase oferit de Google, datorită ușurinței comunicării dintre aplicație și baza de date, dar și din cauza multitudinii de resurse disponibile legate de integrarea celor doua.

1.1 Structura lucrării

În următoarele secțiuni vor fi prezentate specificarea cerințelor și motivația din spatele aplicației, prezentarea unui studiu de caz pe un eșantion relativ mic (aprox. 7-8 persoane), prezentarea soluției propuse, problemele întâmpinate pe parcurs și detalii de implementare, urmate de o evaluare și câteva concluzii.

2 ANALIZA CERINȚELOR / MOTIVAȚIE

Aplicația se adresează persoanelor care au instalat pe smartphone-ul propriu sistemul de operare Android. Totuși, aplicația este gândită pentru a fi utilizată pe versiuni de android lansate mai recent de 21 august 2017, explicit versiunea 8.0 (Android Oreo). În urma unui studiu de caz, majoritatea potențialilor clienți și-au dorit următoarea listă de cerințe pe care trebuie să le întrunească aplicația:

- Înregistrare bazată pe email și parolă
- Autentificare securizată
- Bază de date în timp real
- Posibilitatea de a rămâne logat în aplicație pentru a ușura procesul utilizării acesteia
- O interfață ușor de folosit pentru orice categorie de vârstă
- Procesul de înregistrare a unei cheltuieli să fie unul intuitiv și să dureze mai puțin de 30 de secunde pentru un utilizator care folosește aplicația pentru prima dată
- Bugetul și cheltuielile din ziua, săptămâna sau luna curentă să fie afișate permanent
- Utilizatorul poate modifica bugetul în orice moment
- Utilizatorul poate edita/șterge o cheltuială în orice moment
- O analiză sumară asupra bugetului și cheltuielilor din ziua, săptămâna sau luna în curs
- Istoric pe un interval de timp selectat de utilizator

Una din principala dorință a potențialilor utilizatori ar fi fost dezvoltarea aplicației folosind tehnologii cross-platform, pentru a fi putea utilizată și pe telefoanele cu sistem de operare iOS, dar am ales să rămân doar la dezvoltarea aplicației pe android, folosind ca mediu de lucru Android Studio și o baza de date non-relațională pusă la dispoziție de Google prin serviciul numit Firebase.

Motivația pentru a dezvolta această aplicație a fost inițial una personală, pentru a îmi gestiona mai bine finanțele personale, dar aceasta s-a dovedit o necesitate în grupul meu de prieteni, ceea ce m-a dus cu gândul la implementarea ei pe baza unui studiu de caz completat un număr mic de persoane apropiate. Mai apoi am descoperit că aplicația poate scala dacă ar veni la pachet cu un curs [2] despre gestionarea finanțelor, dar pentru asta ar trebui să existe un parteneriat cu un consultant financiar.

3 FUNDAMENTARE TEORETICĂ

Acest capitol conține detalii despre arhitectura și uneltele folosite pentru dezvoltarea aplicației.

3.1 Android

Android este un sistem de operare și o platformă de dezvoltare creată de Google. Sistemul de operare poate fi folosit atât pe telefoanele mobile inteligente cât și pe alte dispozitive mobile precum tabletele. Gama de dispozitive suportate este foarte variată, incluzând un număr mare de producători. De asemenea, Android este cea mai populară platformă pentru dispozitive mobile, fiind folosită de milioane de dispozitive în mai mult de 190 de țări. În fiecare zi se adaugă noi utilizatori iar ritmul de creștere este unul foarte rapid.

Motivele datorită cărora aplicațiile sunt dezvoltate pentru Android includ: soluționarea cererilor comerciale, construirea de noi servicii, crearea de noi afaceri, precum și oferirea de jocuri și alte tipuri de conținut pentru utilizatori. De asemenea, dezvoltatorii aleg să folosească aceasta platformă pentru a atinge majoritatea utilizatorilor dispozitivelor mobile.

Experiența oferită de Android este una inedită deoarece aceasta oferă o interfață pentru utilizator prin interacțiunea directă cu ecranul dispozitivului. Interfața se bazează pe gesturi precum glisare, apăsare și ciupire a obiectelor afișate pe ecran. Pentru introducerea de text există o tastatură virtuală, dar Android poate suporta și tastaturi cu dimensiune normală, dacă sunt conectate prin Bluetooth sau USB. Rezultatul interacțiunilor cu sistemul Android este unul imediat, interfața fiind una fluidă, de asemenea feedback-ul poate fi primit și prin intermediul vibrațiilor, dispozitivul fiind capabil de răspuns haptic.

În sprijinul comunicării și al oferirii de informații suplimentare utilizatorului, Android poate folosi și componente hardware precum accelerometrul, giroscopul și senzorul de proximitate. un exemplu în acest sens fiind modificarea modului de vizualizare al conținutului din portret în peisaj la detectarea rotirii dispozitivului.

Arhitectura sistemului de operare Android cuprinde cinci secțiuni grupate pe patru niveluri:

- Kernelul Linux - care conține drivere pentru diferite componente hardware, fiind responsabil cu gestiunea proceselor, memoriei, perifericelor, rețelei și a consumului de energie.
- Bibliotecile - care conțin codul care oferă principalele funcționalități ale sistemului de operare, făcând legătura între kernel și aplicații
- Motorul Android - care rulează serviciile de platformă precum și aplicațiile pe care le utilizează

- Cadranul pentru aplicații - care expune diferitele funcționalități ale sistemului de operare Android către programatori, astfel încât aceștia să le poată utiliza în aplicațiile lor.
- Nivelul de aplicații - unde se regăsesc atât aplicațiile preinstalate, cât și cele instalate ulterior de către utilizator

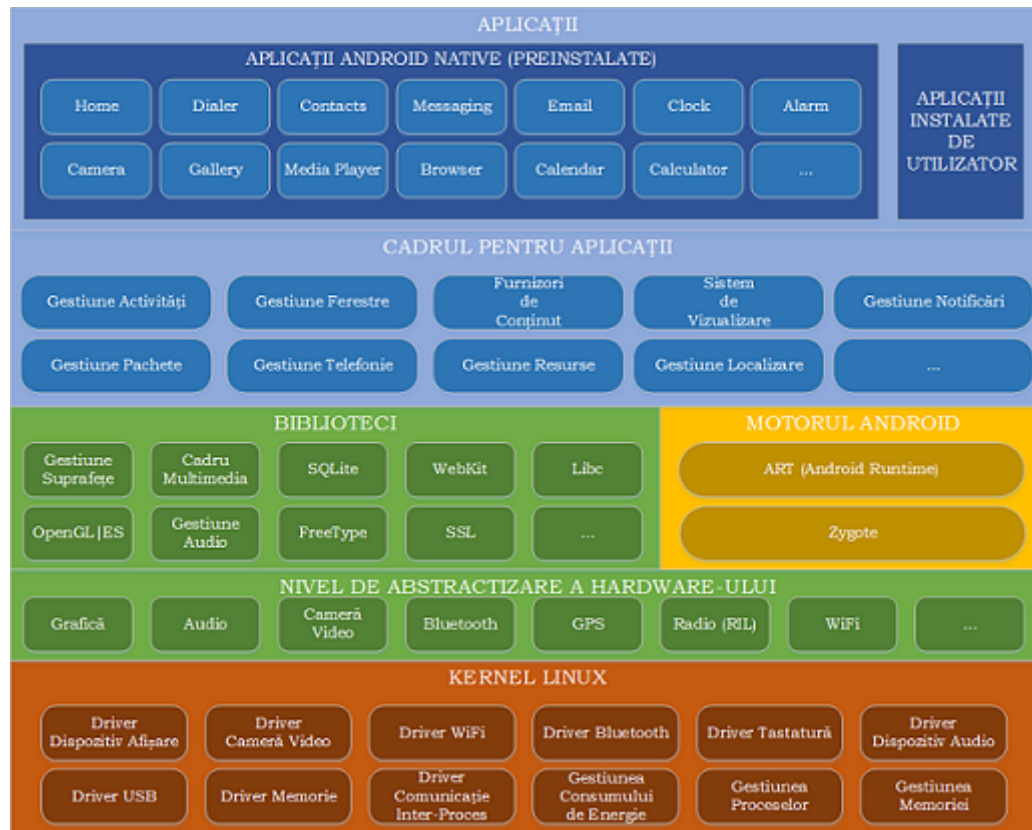


Figura 1: Arhitectura Android

3.2 Android Studio

Android Studio este un IDE oferit de Google, cu scopul de a facilita și ușura dezvoltarea aplicațiilor pentru platforma Android. Până în anul 2013, aplicațiile Android erau scrise în Eclipse IDE, folosit un plugin dezvoltat de asemenea de Google. După lansarea sa, Android Studio a devenit principalul mediu de dezvoltare folosit pentru platforma Android. Instrumentele oferite de acest IDE includ:

- Schimbările asupra codului sunt detectate și semnalate automat, iar aplicația este înnoită în timp real, fără a fi nevoie de o repornire pentru a aplica noile modificări¹
- Emulatorul Android care este o componenta foarte importanta a mediului de dezvoltare, care permite rularea și testarea aplicațiilor pe orice dispozitiv mobil virtual. Acest emulator va avea toate funcționalitățile dispozitivului fizic.

¹cu mici excepții

- Android studio oferă integrare directă cu sistemele de versionare precum Git, facilitând astfel lucrul în echipe mari de dezvoltare.
- Integrare cu Firebase si Google Cloud Platform
- Editor de schema vizuală, prin intermediul fișierelor XML dar și prin interfață grafică
- Analizator de performanță pentru aplicația dezvoltată
- Debugger integrat pentru o mai bună perspectivă asupra rulării aplicației

3.3 AnyChart

AnyChart[1] este o companie care a dezvoltat o bibliotecă scrisă inițial în JavaScript pentru a ajuta dezvoltatorii cu grafice integrate în aplicațiile lor, folosită pe scală largă de către companii cunoscute precum Oracle, 3M, Lockheed Martin și altele. Actual este lider de piață în domeniul "Interactive Data Visualization". Pentru Android există un cod open-source² publicat de companie, care rulează pe versiuni mai recente decât API 19 (Android 4.4), ceea ce presupune acoperire de 99% raportat la telefoanele actuale și conține zeci de tipuri de grafice integrate.

3.4 Firebase

Firebase este o soluție propusă de Google pentru a face viața mai ușoară programatorilor. Este prezentată ca Backend-as-a-Service și oferă utilizări variate precum: autentificare utilizând email, telefon, cont Google, cont Facebook, cont GitHub și altele, serviciu de bază de date în timp real, capacitate de stocare și pune la dispoziție și un API de învățare automată. În proiectul propus s-au folosit doar serviciile de autentificare securizată și bază de date în timp real.

Pe partea de autentificare, utilizatorul aplicației trebuie să folosească un email valid, iar parola introdusă este trecută printr-o funcție hash. Utilizatorului îi este atribuit un ID pe baza căruia se pot accesa datele din baza de date.

Cu Firebase Realtime Database, cum este denumit serviciul de bază de date în timp real, se pot stoca și sincroniza date într-o bază de date nerelațională (NoSQL) stocată în cloud. Datele sunt sincronizate pentru toți clienții în timp real și rămân disponibile chiar și când aplicația nu este utilizată. Datele sunt stocate în format JSON, iar clienții împărtășesc o singură instanță care primește în mod automat actualizări cu cele mai recente date.

²<https://github.com/AnyChart/AnyChart-Android>

4 SOLUȚIA PROPUȘĂ

Capitolul conține o privire de ansamblu a soluției ce rezolvă problema, prin prezentarea structurii / arhitecturii acesteia. Workflow-ul aplicației este următorul: când utilizatorul deschide aplicația va fi direcționat la pagina de login, unde își introduce credențialele pentru a se loga, sau de unde poate merge în pagina de înregistrare. Aceste două pagini comunică printr-un textview care face tranziția între cele două ecrane prin apăsare.

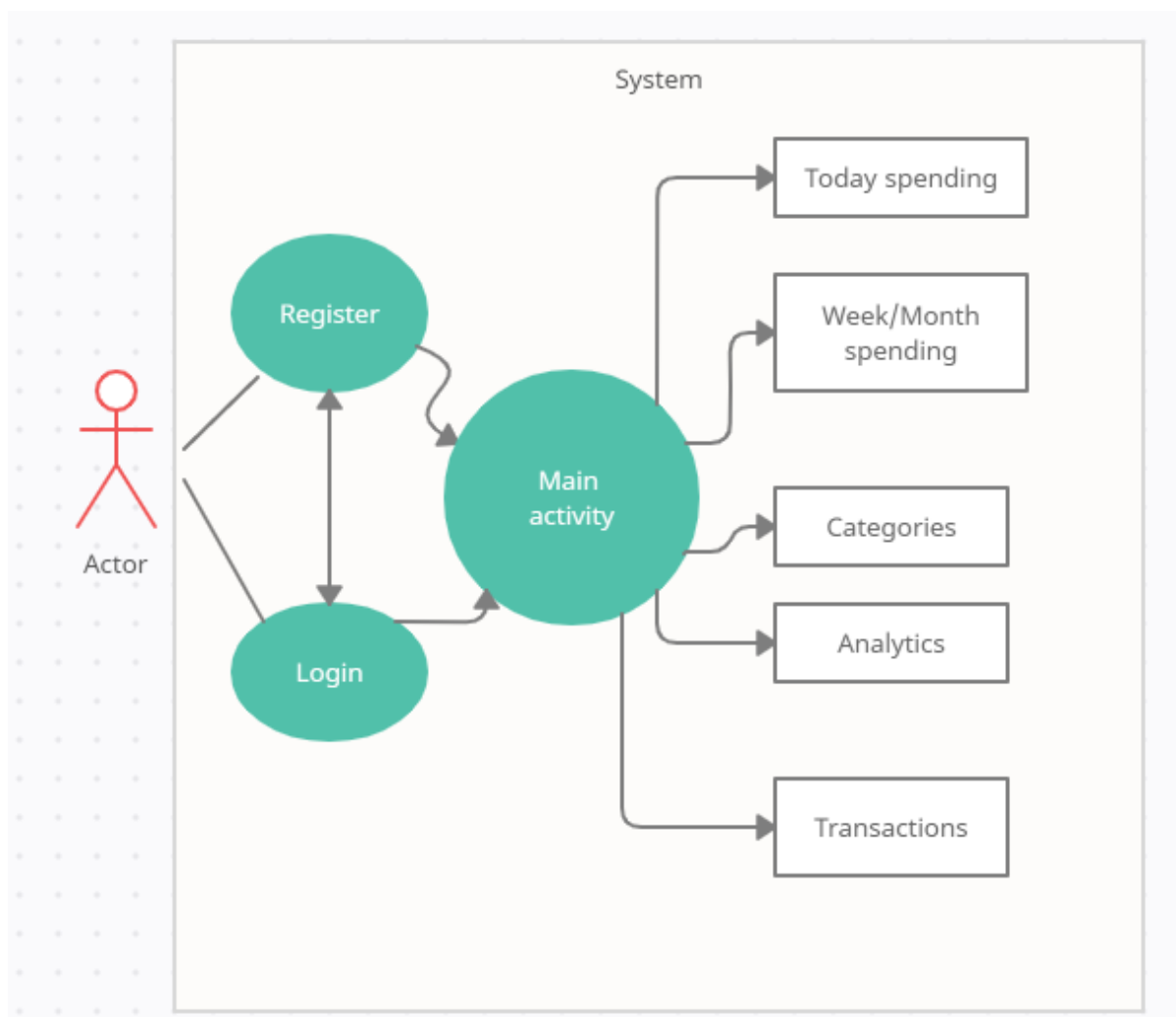


Figura 2: Workflow-ul aplicației

După ce utilizatorul s-a logat/înregistrat, este direcționat la pagina principală a aplicației de unde poate ajunge în mai multe meniuri, în funcție de dorința acestuia. În *categories* el își poate configura bugetul pentru luna în curs, în *today spending* vede istoricul zilei în curs și poate înregistra o cheltuială nouă, în *week spending* și *month spending* își accesează istoricul pentru săptămâna, respectiv luna în curs, în *analytics* are acces la statistici și grafice, iar în

transactions poate selecta o zi pentru a vedea istoricul din ziua selectată.

În ecranul principal, utilizatorul are acces la informații precum bugetul curent, suma rămasă, suma cheltuită pentru ziua, săptămâna și luna în curs. Datele sunt preluate direct din baza de date. Fiecare activitate prezentată în imagine are în spate o cerere către baza de date. Aceasta nu a fost menționată în workflow din motive de redundanță. Cu toate că fiecare în fiecare activitate se face câte o cerere către baza de date, ele împărtășesc aceeași instanță a bazei de date, pentru a utiliza eficient resursele alocate.

5 DETALII DE IMPLEMENTARE

În acest capitol urmează a fi prezentată arhitectura software a sistemului din punct de vedere al funcționalităților existente în aplicație și al locului și rolului acestora în lanțul de procesare. Principala problemă întâmpinată a fost aceea a realizării unui design acceptabil conform cerințelor și aplicațiilor din 2021. Inițial am folosit Adobe XD pentru a realiza interfața aplicației, urmând ca să fac export codului XML generat de Adobe XD și să îl integrez în Android Studio. Pentru că nu am avut un design clar în minte am decis să creez design-ul în paralel cu backend-ul aplicației, astfel ajungând să implementez totul în paralel în Android Studio, renunțând la Adobe XD.

În alegerea paletelor de culori am folosit site-ul *colors.co*¹ pentru a fi în trending cu ultimele tipuri de design folosite pe piață. Paginile de design ale aplicației pot fi găsite în Anexă.

5.1 Autentificare și înregistrare

Pentru partea de autentificare, se primesc emailul și parola ca input de la utilizator, se face o cerere către baza de date în care se verifică dacă datele primite corespund cu cele din baza de date iar în urma răspunsului primit, utilizatorul este sau nu autentificat. Conform cerințelor, utilizatorul trebuie să rămână logat în aplicație chiar și după ce acesta părăsește aplicația. Codul pentru această funcționalitate se poate observa în imaginea următoare:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);

    authStateListener = new FirebaseAuth.AuthStateListener() {
        @Override
        public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
            FirebaseUser user = mAuth.getCurrentUser();
            if (user != null) {
                Intent intent = new Intent( packageContext: LoginActivity.this, MainActivity.class);
                startActivity(intent);
                finish();
            }
        }
    };
};
```

Figura 3: Cod funcție logare automată

¹<https://colors.co/>

Serviciul Firebase dispune de un API prin care se definește un `AuthStateListener` cu ajutorul căruia, atunci când se schimbă o stare a unui utilizator, acest listener este notificat. Când acesta este notificat, se verifică dacă utilizatorul curent mai este logat, pe baza id-ului returnat de funcția `getCurrentUser`, care returnează id-ul utilizatorului, sau `NULL`. În cazul în care se returnează diferit de `null`, utilizatorul respectiv e direcționat direct la pagina principală a aplicației.

Partea de înregistrare este similară cu cea de logare, diferența majoră fiind că inputul primit de la utilizator este salvat în baza de date când facem înregistrarea, și căutat în baza de date când facem logarea. În ambele cazuri, dacă procesul eșuează din diverse motive, utilizatorul este notificat printr-un mesaj că activitatea respectivă nu s-a finalizat cu succes.

5.2 Structura bazei de date

Baza de date este împărțită în 3 mari categorii: `budget`, `expenses` și `personal`. În fiecare categorie se poate observa un șir de caractere general la întâmplare, care reprezintă id-ul unui utilizator. Imaginea următoare exemplifică structura bazei de date în care există 2 utilizatori care au adăugat date, dar doar unul dintre ei și-a setat bugetul.

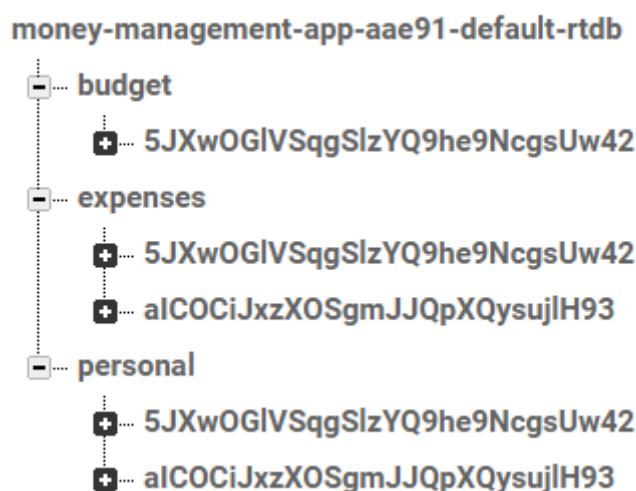


Figura 4: Structura bazei de date

Mergând pe următorul nivel, categoriile `budget` și `expenses` au o structură asemănătoare, categoria `expenses` având în plus o notă folosită pentru a salva informații despre o anumită cheltuială. În continuare găsim prezentată categoria `budget`. Fiecare item din buget pe care îl adăugăm are asociat un id.

După ce expandăm informațiile despre un utilizator, găsim mai multe id-uri, care reprezintă id-urile unui item din buget, iar ajungând pe cel mai jos nivel din ierarhia acestor categorii, găsim mai multe informații despre item-ul respectiv, precum suma totală alocată, data alocării, numele categoriei și câteva alte date care ne ajută la implementarea funcționalității

aplicației. Pentru a oferi și mai multe detalii, spre exemplu week și month reprezintă numărul de săptămâni, respectiv luni care au trecut de la 1 ianuarie 1970, așa cum este timpul definit în termenii unui programator.

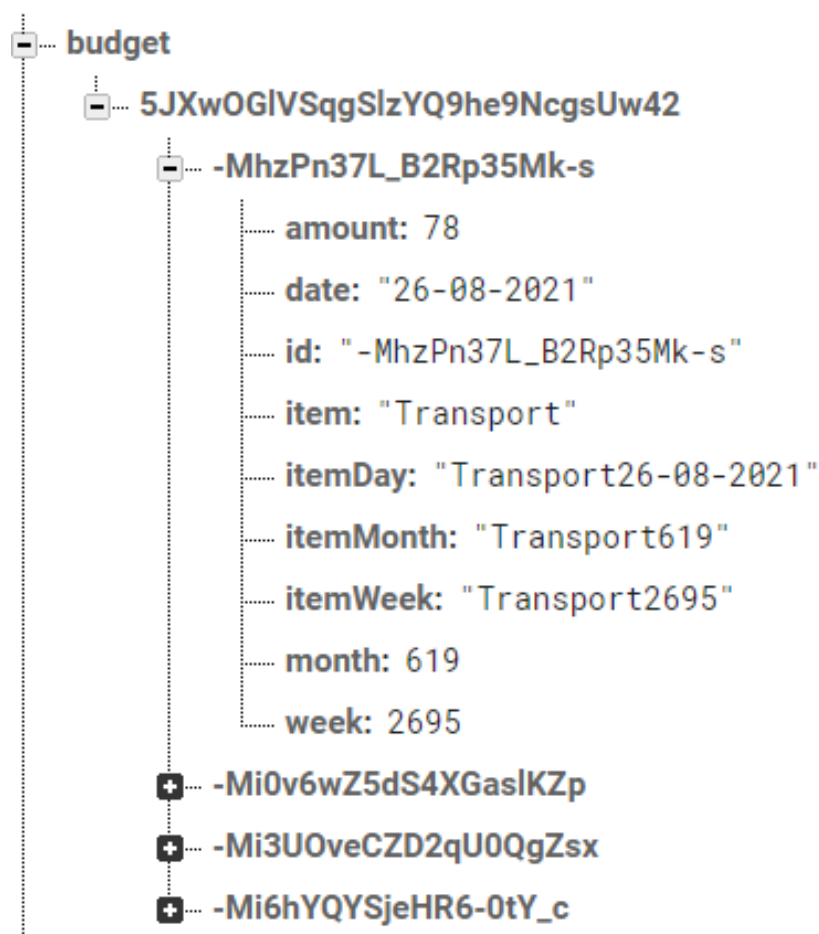


Figura 5: Structura unui buget/Structura unei cheltuieli

Structura categoriei expenses este similară structurii categoriei budget, având în plus un câmp pentru adaugarea unei note suplimentare.

Ca și în cazul celorlalte două categorii, structura de categorie - id utilizator se păstrează și în cazul categoriei personal, fiecare id de utilizator cuprinzând o multitudine de date relevante pentru funcționalitatea de analiză din aplicația propusă. Aici avem stocate bugetul total, suma cheltuielilor din ziua, săptămâna și luna în curs, rația și suma cheltuită pentru ziua și săptămâna în curs pentru fiecare categorie. Pentru alocarea bugetului săptămânal, am împărțit bugetul lunar la 4, iar pentru alocarea bugetului zilnic am împărțit bugetul la 30 (cu toate că unele luni au 28, 29 sau 31 de zile).

Cu ajutorul acestor date putem face o analiză sumară a cheltuielilor, având totul calculat sub forma de numere și procentaje.

5.3 Pagina principală

După ce utilizatorul s-a logat cu succes, el este direcționat către pagina principală. În partea de sus a ecranului va găsi un toolbar și un mic tabel în care are suma totală alocată pentru buget, sumele cheltuite în ziua, săptămâna și luna respective, precum și numărul de bani disponibili pentru cheltuieli, sub titlul de savings. În toolbar va apărea numele aplicației cât și o iconiță de unde se pot accesa datele contului și de unde utilizatorul se poate deloga, dacă dorește acest lucru.

În restul ecranului va găsi butoane sugestive pentru a accesa funcționalitățile aplicației. În meniul *Categories* utilizatorul își setează bugetul pentru o anumită categorie, în *Today* își înregistrează cheltuielile și vede istoricul pentru ziua respectivă, în *This week* și *This month* poate accesa istoricul pentru săptămâna și luna în curs, în *Analytics* poate vedea statistici și grafice pentru fiecare categorie pentru ziua, săptămâna sau luna în curs, iar în *Transactions* poate vedea istoricul pe un interval de timp selectat.

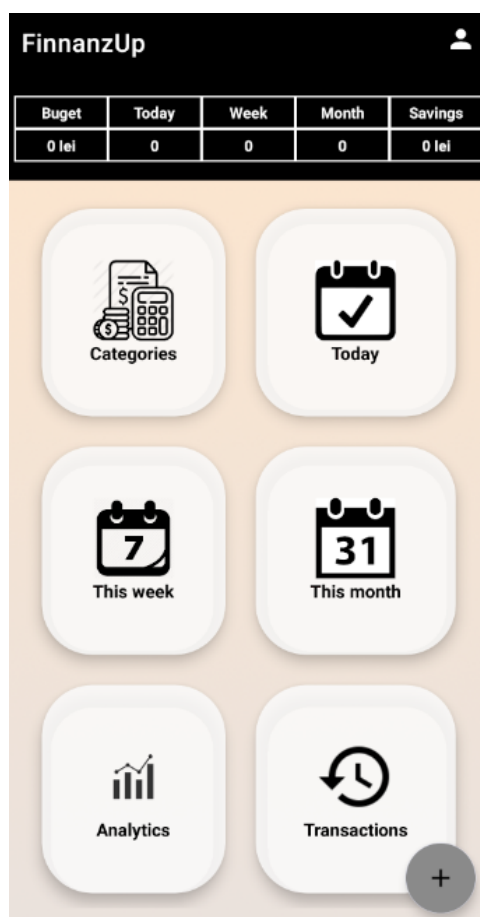


Figura 6: Pagina principală

Pe partea de backend pentru pagina principală găsim în primul rând conexiunea cu baza de date dar și cate un listener pentru fiecare din butoanele prezente în imagine. Un listener este activat atunci când butonul asociat lui este apăsat iar rolul lui este de a direcționa utilizatorul

către pagina asociată butonului. Pentru partea de conectivitate cu baza de date avem *mAuth* care reprezintă instanța utilizatorului curent, *onlineUserId* id-ul userului curent, iar *budgetRef*, *expensesRef* și *personalRef* sunt referințe pentru părțile prezentate în subcapitolul anterior,

5.2 Structura bazei de date

```
mAuth = FirebaseAuth.getInstance();
onlineUserId = FirebaseAuth.getInstance().getCurrentUser().getUid();
budgetRef = FirebaseDatabase.getInstance(url_firebase).getReference(path: "budget").child(onlineUserId);
expensesRef = FirebaseDatabase.getInstance(url_firebase).getReference(path: "expenses").child(onlineUserId);
personalRef = FirebaseDatabase.getInstance(url_firebase).getReference(path: "personal").child(onlineUserId);
```

Figura 7: Inițializare referințe bază de date

Cum Firebase pune la dispoziție o bază de date NoSQL, cea mai bună metodă de a adăuga/edita datele se face cu ajutorul unui map. În poza de mai jos putem observa cum este primit ca parametru un *DataSnapshot* prin care se iterează. Fiecare subsecvență din acest *datasnapshot* este interpretat ca un map, de unde se extrage valoarea din poziția *amount*. După ce toate sumele au fost adunate și stocate într-o variabilă separată, în instanța *personalRef* este adăugată această valoare la poziția *budget*. Practic, s-a iterat prin toate categoriile de buget, s-a extras suma alocată pentru fiecare categorie, s-a calculat suma lor într-o variabilă și apoi a fost salvată în referința *personalRef* sub forma "budget: sum".

```
budgetRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if (snapshot.exists() && snapshot.getChildrenCount() > 0){
            for (DataSnapshot ds : snapshot.getChildren()){
                Map<String, Object> map = (Map<String, Object>)ds.getValue();
                Object total = map.get("amount");
                int pTotal = Integer.parseInt(String.valueOf(total));
                totalAmountBudgetB += pTotal;
            }
            totalAmountBudgetC = totalAmountBudgetB;
            personalRef.child("budget").setValue(totalAmountBudgetC);
        } else {
            personalRef.child("budget").setValue(0);
            Toast.makeText(context MainActivity.this, text "Please set a BUDGET in Categories section", Toast.LENGTH_LONG).show();
        }
    }
});
```

Figura 8: Adăugare în baza de date

Pentru editare în baza de date procedura este similară și va fi prezentată ulterior în secțiunea următoare.

5.4 Pagină categorii

În această secțiune este prezentat funcționalitatea de adăugare buget pentru o anumită categorie. În timp ce această pagină este încărcată vizual, se face o interogare în baza de date pentru a extrage și afișa utilizatorului bugetele deja alocate pentru categorii (dacă acestea există). Adăugarea se face din butonul + (floating action button), care va deschide pe ecran o fereastră pop-up unde utilizatorul trebuie să selecteze o categorie dintr-un dropdown și să introducă suma alocată.

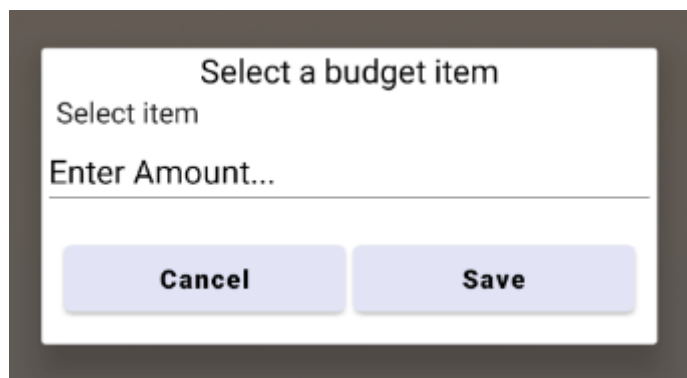
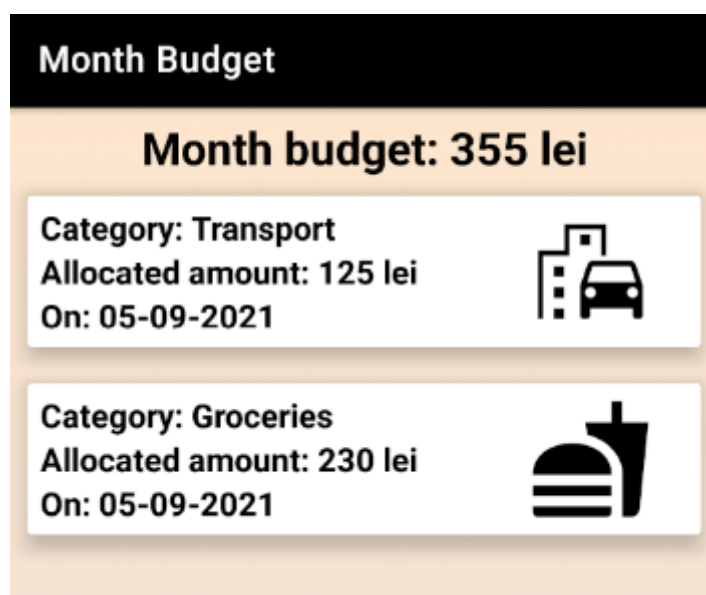


Figura 9: Adăugare în baza de date

Pentru acest lucru este implementată și funcționalitate de validare de input, în sensul în care, dacă utilizatorul nu alege o categorie, îi va apărea pe ecran un mesaj informativ să aleagă o categorie, iar dacă nu introduce nicio sumă, îi va apărea un mic semn roșu cu un mesaj vizibil dacă atinge semnul. Astfel, pentru ca un item să fie adăgat în baza de date, este nevoie ca utilizatorul să completeze toate datele cerute.





Month Budget	
Month budget: 355 lei	
Category: Transport Allocated amount: 125 lei On: 05-09-2021	
Category: Groceries Allocated amount: 230 lei On: 05-09-2021	

Figura 10: Afișare buget categorii

Inserarea bugetului pentru categorii se va face ordonat, cea mai recentă adăugare fiind prima din listă. În partea de sus putem observa totalul bugetului lunar, iar pentru fiecare item adăugat putem vedea categoria aleasă, suma alocată și data alocării, cât și o imagine reprezentativă pentru categorie.

Pentru fiecare item din această pagină putem să modificăm suma alocată sau să îl ștergem complet. Cu un singur click putem pe item putem avea acces la aceste funcționalități dintr-o nouă fereastră pop-up. Pentru ambele funcționalități trebuie identificat în baza de date item-ul ales, singura diferență fiind apelarea metodei *setValue(data)* pentru a face un update în baza de date și apelarea metodei *removeValue()* pentru ștergerea completă din baza de date.

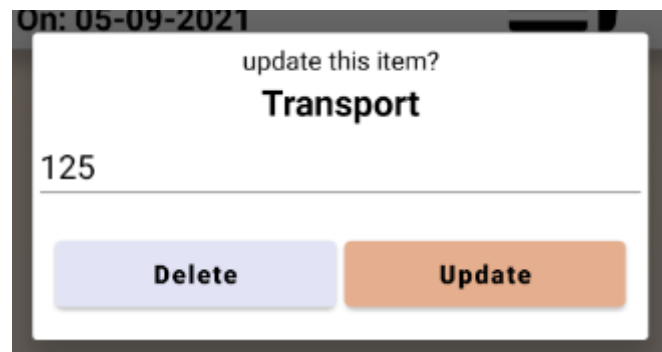


Figura 11: Fereastră pop-up pentru editare item

Pentru partea de update, se construiește un obiect de tip *Data* pe baza informațiilor luate direct din baza de date pentru itemul ales, cu excepția sumei, care este introdusă de utilizator, și apoi se publică obiectul în baza de date, apelând metoda menționată anterior.

Structura obiectului de tip *Data*, împreună cu constructorul clasei, sunt afișate în poza de mai jos:

```
public class Data {  
    String item, date, id, notes, itemDay, itemWeek, itemMonth;  
    int amount, month, week;  
  
    public Data() {  
    }  
  
    public Data(String item, String date, String id, String itemDay, String itemWeek, String itemMonth, int amount, int month, int week, String notes) {  
        this.item = item;  
        this.date = date;  
        this.id = id;  
        this.notes = notes;  
        this.itemDay = itemDay;  
        this.itemWeek = itemWeek;  
        this.itemMonth = itemMonth;  
        this.amount = amount;  
        this.month = month;  
        this.week = week;  
    }  
}
```

Figura 12: Obiect de tip *Data*

5.5 Pagină Today

Pagina *Today* prezintă același design ca pagina *Categories*. Operațiunea de adăugare a unei cheltuieli se face tot de la butonul + (floating action button) care generează o fereastră pop-up în care utilizatorul trebuie să selecteze categoria din care face parte cheltuiala, să introducă suma și o notă informativă. Ca și în cazul înregistrării bugetului, există implementată funcționalitatea de validare a inputului, pentru a oferi utilizatorului indicații în orice moment de timp despre ce date mai are de completat. După ce toate datele sunt completate, se face scrierea în baza de date.

Procesul de editare sau ștergere este identic, printr-o simplă apăsare pe item-ul din listă se generează o fereastră pop-up unde utilizatorul poate modifica datele cheltuielii înregistrare, sau o poate șterge complet din baza de date. În partea de jos este afișată suma totală cheltuită în ziua respectivă.

Pentru a afișa toate cheltuielile dintr-o zi, pe partea de frontend s-a folosit un *scroll menu*, pentru a ne asigura că toate cheltuielile sunt vizibile în aceeași pagină. Pentru asta, în backend s-au realizat două clase, *TodayItemsAdapter* și *TodaySpendingActivity*. Clasa Adapter ne ajută să afișăm detaliile actualizate corespunzătoare unei cheltuieli, precum imaginea potrivită pentru categoria aleasă, modificarea generică a textului dintr-un item gol. Aceste lucruri se petrec atunci când în clasa *TodaySpendingActivity* este instanțiat un obiect de tip Adapter. Tot clasa Adapter se ocupă cu actualizarea sau ștergerea în baza de date.

Clasa *TodaySpendingActivity* este responsabilă cu citirea datelor din baza de date și trimiterea lor către clasa Adapter pentru a putea fi afișate corespunzător. Pentru adăugare unei noi cheltuieli, tot această clasă este responsabilă. După ce se finalizează validarea inputului utilizatorului, datele sunt salvate într-un obiect de tip Data și publicate direct în baza de date, fără ajutorul clasei Adapter.

În imaginea următoare putem observa cum clasa *TodaySpendingActivity* este responsabilă și de comunicarea cu frontend-ul: modificare de text, setare dimensiune scroll menu, activare de iconițe în toolbar:

Fiecare din butoanele și cardurile afișate în frontend, au asociat un id care este inițializat și folosit în partea de backend pentru a face legătura directă între datele din backend și datele din frontend.

```

toolbar = findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
getSupportActionBar().setTitle("Today spending");
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
getSupportActionBar().setDisplayShowHomeEnabled(true);

LinearLayoutManager linearLayoutManager = new LinearLayoutManager(context, this);
linearLayoutManager.setStackFromEnd(true);
linearLayoutManager.setReverseLayout(true);
recyclerView.setHasFixedSize(true);
recyclerView.setLayoutManager(linearLayoutManager);

```

Figura 13: Modificare constrângeri frontend din partea de backend

5.6 Paginile This week și This month

Aceste două pagini sunt similare, diferența dintre ele fiind perioada de afișare a cheltuielilor din baza de date. Cum la pagina *Today* am folosit două clase (clasa principală și o clasa Adapter), pentru aceste două pagini am folosit aceleași două clase numite *WeekSpendingAdapter* și *WeekSpendingActivity*, pentru implementarea funcționalității pentru *This month*, folosind o metodă diferită de cea pentru *This week*, care extrage datele pentru luna curentă din baza de date. Funcționalitatea clasei Adapter este similară cu cea prezentată în secțiunea anterioară.

```

private void readWeekSpendingItems() {
    MutableDateTime epoch = new MutableDateTime();
    epoch.setDate(0);
    DateTime now = new DateTime();
    Weeks weeks = Weeks.weeksBetween(epoch, now);
    Months months = Months.monthsBetween(epoch, now);

    expensesRef = FirebaseDatabase.getInstance(url_firebase).getReference(path: "expenses").child(onlineUserId);
    Query query = expensesRef.orderByChild("week").equalTo(weeks.getWeeks());
    query.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            myDataList.clear();
            for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                Data data = dataSnapshot.getValue(Data.class);
                myDataList.add(data);
            }
        }
    });
}

```

Figura 14: Modificare constrângeri frontend din partea de backend

Pentru că nu a fost explicat în secțiunile anterioare, o să dedic această parte pentru a explica cum se calculează data utilizând API-ul din imaginea de mai sus.

Inițial se declară un obiect epoch de tipul `MutableDateTime`, iar apoi se inițializează cu 0, ceea ce înseamnă data de 1 ianuarie 1970. Apoi se creează un obiect `now` de tipul `DateTime`, pentru a extrage data exactă. Pentru calcularea diferenței se folosesc două obiecte, unul care stochează numărul de săptămâni dintre obiectul epoch și obiectul `now`, iar celălalt, care calculează numărul de luni dintre data stocată în obiectul epoch și data stocată în obiectul `now`.

Următoarele linii de cod din figura de mai sus reprezintă inițializarea instanței `expensesRef` a bazei de date pentru categoria `expenses` prezentată în secțiunea *Structura bazei de date*, apelarea unei interogări în funcție de săptămâna primită ca input, și apoi extragerea datelor din baza de date și adăugarea lor într-un `ArrayList` pentru a putea fi accesate și prelucrate cu ușurință de către programator.

În timp ce se realizează comunicarea cu baza de date și se așteaptă răspunsul, avem un `progressBar` care se învâрте, pentru a oferi utilizatorului un semnal că aplicația e în așteptarea datelor cerute. După ce răspunsul este primit, vizibilitatea bării de progres dispare, se calculează suma totală pentru săptămâna sau luna în curs și se afișează pe ecran, alături de fiecare item din baza de date, stocate în `ArrayList`-ul menționat anterior.

```
if (getIntent().getExtras() != null) {
    type = getIntent().getStringExtra( name: "type");
    if (type.equals("week")) {
        readWeekSpendingItems();
    } else if (type.equals("month")) {
        readMonthSpendingItems();
    }
}
```

Figura 15: Selectarea activității pentru săptămâna sau luna în curs

În secțiunea curentă s-a prezentat cum se folosesc aceleași două clase pentru două prelucrarea și afișarea datelor pentru săptămâna și luna în curs. Un intent reprezintă acțiunea de a schimba interfața curentă pe care o vede utilizatorul cu una nouă. În declarația lor, avem o separație în funcție de nume. Astfel, se verifică șirul de caractere asociat unui intent și se apelează metoda corespunzătoare șirului, `readWeekSpendingItems()` pentru afișarea datelor pentru săptămâna în curs și `readMonthSpendingItems()` pentru afișarea datelor pentru luna în curs.

5.7 Pagină analytics

În această secțiunea va fi prezentată partea de analiză și grafice aferente. Când utilizatorul alege să vizualizeze analiza, este mai întâi direcționat către o pagina care conține trei butoane: unul pentru analiza pentru ziua în curs, următorul pentru analiza pentru săptămâna în curs, iar ultimul pentru analiza pentru luna în curs. Codul pentru un buton care redirecționează către o altă pagină arată în felul următor:

```
todayCardView.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Intent intent = new Intent( packageContext ChooseAnalyticsActivity.this,  
                                   DailyAnalyticsActivity.class);  
        startActivity(intent);  
    }  
});
```

Figura 16: Redirecționare către altă pagină

Butonul conține un listener care, atunci când detectează un click pornește un intent nou cu layout-ul actual și clasa unde se vrea a ajunge. Când se creează un obiect de tipul clasei respective, el apelează o metoda numită *OnCreate* care încarcă interfața pentru noua pagină.

Funcționalitatea pentru cele trei opțiuni este similară, unde variază doar bugetul alocat și durata.

În figura de mai jos este prezentată analiza lunară a bugetului pentru fiecare categorie. În partea de sus avem afișată suma totală cheltuită pentru luna în curs, apoi avem niște buline, și o explicație în dreptul lor: verde - <50%, maro - 50-99% și roșu - >100%. În continuare se face o cerere către baza de date pentru a afla care dintre categorii au alocate bugete pentru luna în curs, iar cele care au bugetul setat pe 0 sunt omise, pentru a nu exista redundanță (nu are sens să facem analiză pentru niște categorii care au buget nealocat).

Fiecare item poate fi reprezentat sub o structură de forma următoare:

- Numele categoriei
- Suma cheltuită din categoria respectivă
- Procentul cheltuielii pentru categoria respectivă raportat la bugetul alocat pentru categorie, în statusul curent²
- Status - care poate fi ziua, săptămâna sau luna în curs
- Bulina colorată care reprezintă una din cele trei stări menționate anterior
- Pictogramă reprezentativă pentru categorie

²de exemplu, pentru o săptămâna am fi avut a 4-a parte din buget

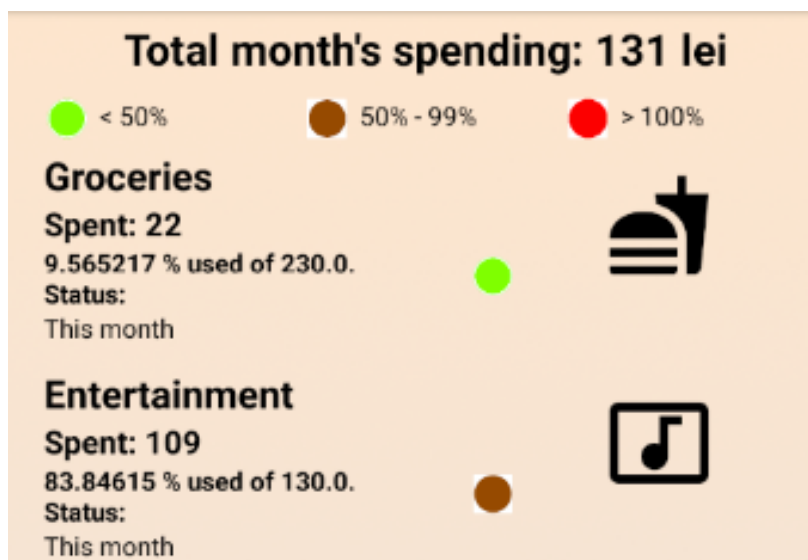


Figura 17: Analiză lunară

Pentru fiecare categorie există câte o metodă care preia datele din baza de date și le parsează corespunzător. În această parte de analiză, pentru fiecare utilizator sunt salvate (în rubrica personal din structura bazei de date, prezentată în secțiunea 5.2) un buget zilnic total, un buget zilnic pentru fiecare categorie, un buget săptămânal total, un buget săptămânal pentru fiecare categorie, un buget săptămânal total iar bugetul lunar pentru fiecare categorie este preluat din rubrica *budget* din structura bazei de date.

Pentru a exemplifica procesul de setare a bulinei corespunzătoare am inserat codul de mai jos (care are în centru categoria *Personal*). Se extrag din baza de date *personalTotal* și *personalRatio*, iar pe baza lor se calculează *personalPercent*. În funcție de procentul rezultat se setează textul și imaginea corespunzătoare.

```
float personalPercent = (personalTotal/personalRatio) * 100;
if (personalPercent < 50){
    progress_ratio_personal.setText(personalPercent + " %" + " used of " +
    personalRatio + ". Status:");
    status_Image_personal.setImageResource(R.drawable.green);
} else if (personalPercent >= 50 && personalPercent < 100){
    progress_ratio_personal.setText(personalPercent + " %" + " used of " +
    personalRatio + ". Status:");
    status_Image_personal.setImageResource(R.drawable.brown);
} else {
    progress_ratio_personal.setText(personalPercent + " %" + " used of " +
    personalRatio + ". Status:");
    status_Image_personal.setImageResource(R.drawable.red);
}
```

Figura 18: Cod pentru reprezentarea corectă a iconiței utilizării bugetului

Cu ajutorul bibliotecii adiționale AnyChart [1] am putut să integrez un grafic în aplicație care permite vizualizarea cheltuielilor sub formă de pie-chart. În partea de sus a graficului se găsește o legendă care asociază o culoare pentru fiecare categorie. În imaginea de mai jos avem cheltuielile înregistrate în luna curentă (pe un cont de test).

Standard, este afișat doar procentul pentru categorii, dar cu click putem afla mai multe informații precum numele categoriei din care face parte cheltuiala, valoarea numerică a acesteia, cât și procentul pe care îl reprezintă din suma totală cheltuită.

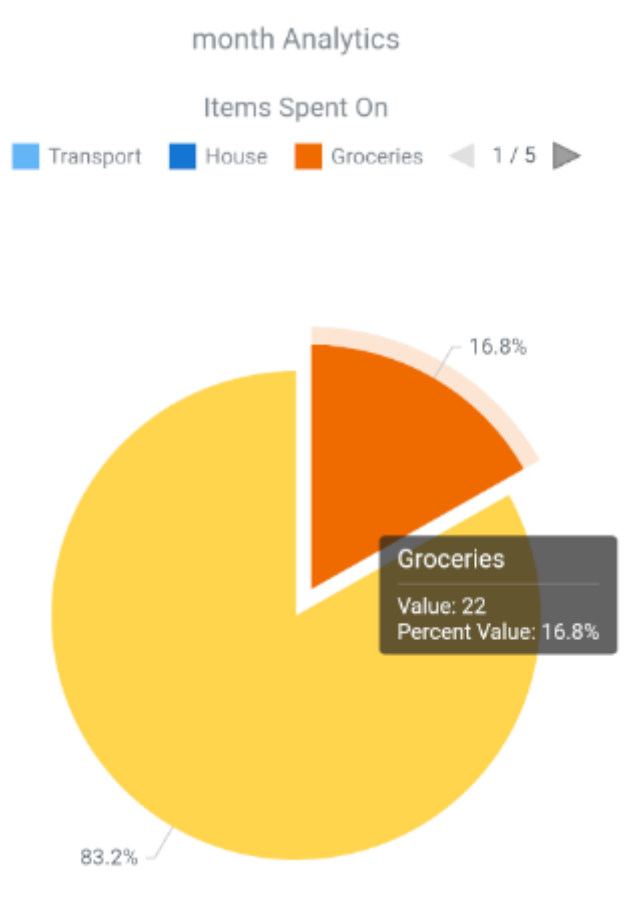


Figura 19: Cheltuielile reprezentate grafic

Ca și documentație pentru folosirea graficului, am folosit repository-ul oficial AnyChart-Android³ unde au oferit și un exemplu clar de prelucrare și inserare a datelor, instanțiere și poziționare în pagină a graficului. Repository-ul de GitHub oferă zeci de grafice pentru numeroase reprezentări, documentate cu o poză pentru vizualizare și cu un fișier aferent pentru exemplificarea codului folosit.

În următoarea imagine putem vedea și codul din spatele diagramei. Pentru crearea graficului se instanțează un obiect de tip *Pie*, iar pentru adăugarea de categorii și sume asociate se folosește un *ArrayList* de *DataRow*, un *DataRow* având o asociere *String* - *Int* (Numele categoriei - Suma alocată). Apoi se apelează câteva metode legate de design-ul graficului, poziția lui în view și legenda. Cu metoda *setChart()* se prelucrează datele și se afișează

graficul.

```
Pie pie = AnyChart.pie();
List<DataEntry> data = new ArrayList<>();
data.add(new ValueDataEntry(x: "Transport", transportTotal));
data.add(new ValueDataEntry(x: "House", houseTotal));
data.add(new ValueDataEntry(x: "Groceries", groceriesTotal));
data.add(new ValueDataEntry(x: "Entertainment", entertainmentTotal));
data.add(new ValueDataEntry(x: "Education", educationTotal));
data.add(new ValueDataEntry(x: "Donation", donationTotal));
data.add(new ValueDataEntry(x: "Personal", personalTotal));
data.add(new ValueDataEntry(x: "Economies", economiesTotal));
data.add(new ValueDataEntry(x: "Investment", investmentTotal));
data.add(new ValueDataEntry(x: "Other", otherTotal));

pie.data(data);
pie.title("month Analytics");
pie.labels().position("outside");
pie.legend().title().enabled(true);
pie.legend().title()
    .text("Items Spent On")
    .padding(0d, 0d, 10d, 0d);
pie.legend()
    .position("center-bottom")
    .itemsLayout(LayoutManager.HORIZONTAL)
    .align(Align.CENTER);
anyChartView.setChart(pie);
```

Figura 20: Porțiune de cod pentru realizarea graficului

În parte de jos a aplicației, imediat sub grafic, utilizatorul poate vedea prezentată statistica sub aceeași structură menționată anterior, dar pentru ziua, săptămâna sau luna în curs, depinzând de meniul pe care îl accesează.

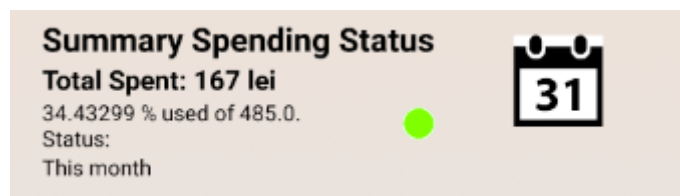


Figura 21: Vizualizare statistică lunară

³ github.com/AnyChart-Android/src/main/java/com/anychart/sample/charts/PieChartActivity.java

5.8 Pagină transactions

În această pagină utilizatorul ar trebui să vadă istoricul cheltuielilor pentru un interval selectat de timp. Pentru moment, această funcționalitate nu este îndeplinită, utilizatorul având posibilitatea să își vadă istoricul doar pentru o zi selectată.

Când accesează pagina transactions, utilizatorul poate să apese doar pe un buton de search, care va afișa o fereastră pop-up care conține un calendar. Pentru aceasta s-a folosit un obiect de tip DatePickerDialog de unde se extrag ziua, luna și anul selectate.

Dacă pentru ziua selectată nu există nicio cheltuială înregistrată, se va afișa un mesaj "You have spent nothing this day". În caz contrar, se vor afișa toate cheltuielile pentru ziua selectată, butonul de search rămânând în pagină pentru a putea fi reutilizat în orice moment.

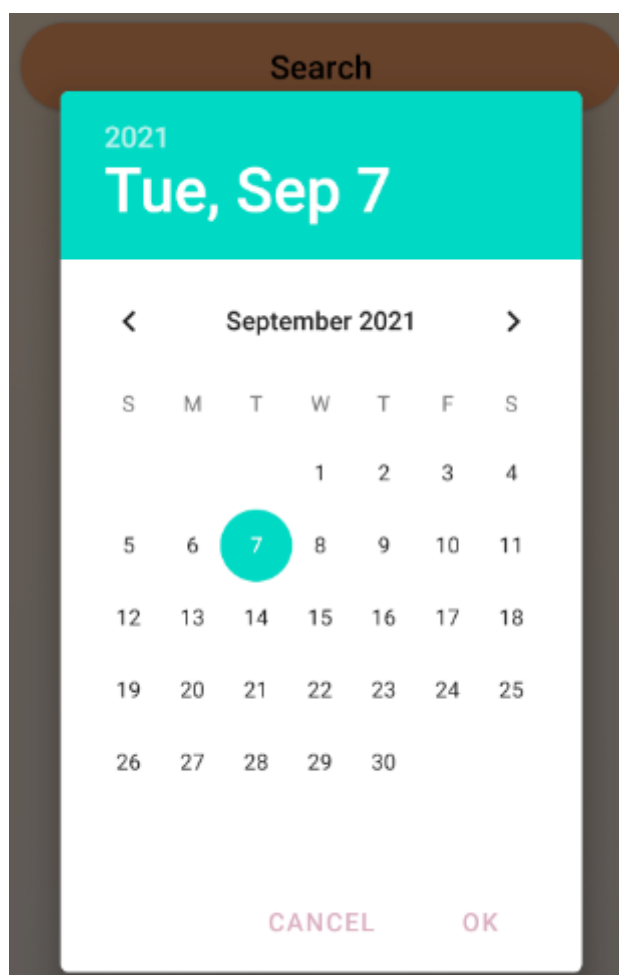


Figura 22: Fereastră pop-up pentru selectarea unei zile

5.9 Pagina My Account

Această pagină poate fi accesată din iconița standard pentru informații cont, situată în colț dreapta-sus în pagina principală. Aici utilizatorul poate găsi informații despre contul lui. Pentru moment, este disponibil doar emailul cu care este logat și funcționalitatea de Log out, dacă utilizatorul vrea să se logheze ulterior pe alt cont.

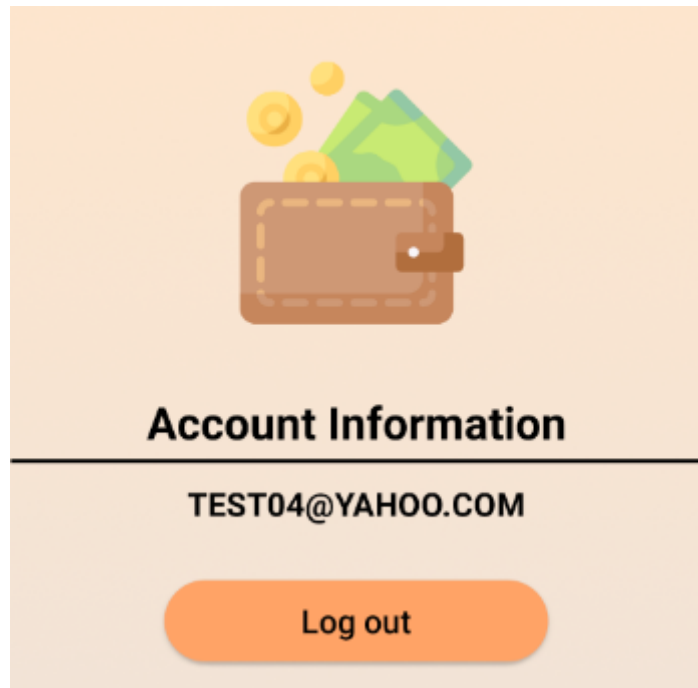


Figura 23: Informații despre cont

5.10 Implementarea interfeței

Implementarea interfeței a fost realizată în Android Studio prin cod XML și prin unealta standard oferită de acest mediu de dezvoltare. Iconițele folosite au fost preluate de pe site-urile flaticon [3] și google fonts [4]. Părțile standard, precum afișarea unui buton sau a unui text au fost realizate cu ajutorul Android Studio.

Părțile complexe din design au fost realizate în cod XML (card view, rotunjirea marginilor butoanelor, crearea background-ului în stil gradient, adăugarea constrângerilor pentru compatibilitatea cu alte dimensiuni ale display-ului). Poze cu codul XML pot fi găsite în *Anexă*.

În următoarea imagine este exemplificat codul pentru crearea imaginii de background care pleacă de la o culoare în partea de sus a paginii și treptat se schimbă în altă culoare, o dată cu coborârea în pagină:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
    <item>
        <shape xmlns:android="http://schemas.android.com/apk/res/android">
            <gradient
                android:layout_width="200dp"
                android:layout_height="100dp"
                android:angle="90"
                android:endColor="#ffe6cc"
                android:startColor="@color/low_background"
                android:type="linear" />
        </shape>
    </item>
</selector>
```

Figura 24: Cod XML pentru imaginea de fundal în stil gradient

6 EVALUARE ȘI ÎMBUNĂTĂȚIRI VIITOARE

Aplicația a reușit să întrunească majoritatea cerințelor prezentate în prima secțiune a acestui document. Aplicația a fost trimisă spre a fi testată de către utilizatori obișnuiți iar feedback-ul primit a fost unul pozitiv și cu sugestii pentru îmbunătățire. Aplicația a reușit să satisfacă dorințele utilizatorilor, este ușor și intuitiv de folosit, timpul pentru logarea unei cheltuieli nu a fost depășit, cu toate că uneori comunicarea cu baza de date este lentă.

Printre principalele mențiuni care vor fi în continuare ținta pentru îmbunătățiri viitoare se numără:

- Posibilitatea utilizatorului de a adăuga și șterge categorii (pentru moment, utilizatorul poate folosi una din cele 10 categorii prestabilite)
- Vizualizarea tranzacțiilor pentru un interval de timp selectat (pentru moment, aplicația oferă doar date pentru o anumită zi, nu pentru un interval de zile)
- Implementarea unui sistem de puncte prin care utilizatorul este recompensat în funcție de câți bani reușește să economisească într-o lună
- Vizualizare analiză și grafice pentru un interval de timp (pentru moment, aplicația oferă analiză și grafice pentru ziua, săptămâna și luna în curs)
- Adăugare amprentă pentru autentificare pentru a facilita un nivel de securitate mai ridicat
- Adăugarea unei teme Dark
- Posibilitatea adăugării unui venit recurent și salvarea bugetului de la o lună la alta (pentru moment, utilizatorul trebuie să își adauge lunar bugetul în aplicație)
- Adăugarea de notificări atunci când bugetul scade sub 10% pentru o categorie
- Adăugarea unei cheltuieli pentru o zi anterioară (pentru moment, se pot adăuga cheltuieli doar pentru ziua curentă)
- Adăugarea mai multor monede (pentru moment, totul este raportat în lei)

Pentru comparație am descărcat de pe Google Play cea mai bine cotate aplicație în acest domeniu, "Money manager, expense, tracker, budget, wallet" cu un rating de 4.9/5, dezvoltată de *Innim Mobile Exp*. Aplicația cuprinde toate funcționalitățile prezentate în specificația cerințelor, dar și pe majoritatea din cele propuse pentru îmbunătățiri viitoare. Comparativ cu soluția propusă în această lucrare, aplicația are o interfață încărcată și este puțin greu de utilizat până când utilizatorul se familiarizează cu interfața.

Cel mai avantaj al soluției propuse este că nu are reclame, spre deosebire de aplicația menționată anterior, care se poate utiliza gratis și cu reclame, sau se poate cumpăra un abonament lunar pentru a nu mai exista reclamele. O altă îmbunătățire care ar schimba major modul de

folosire al aplicației și ar atrage utilizatori ar fi una din cele menționate mai sus și anume: implementarea unui sistem de puncte prin care utilizatorul este recompensat în funcție de câți bani a economisit în luna respectivă și folosirea punctelor pentru a debloca eventuale iconițe sau pictograme.

O problemă semnalată de către utilizatori ține de design-ul aplicației în sensul în care nu este prea intuitivă selectarea unui item. O altă soluție ar fi adăugarea explicită a unui meniu standard de drop-down pentru a sugera utilizatorului să apese pe mesajul "Select item"

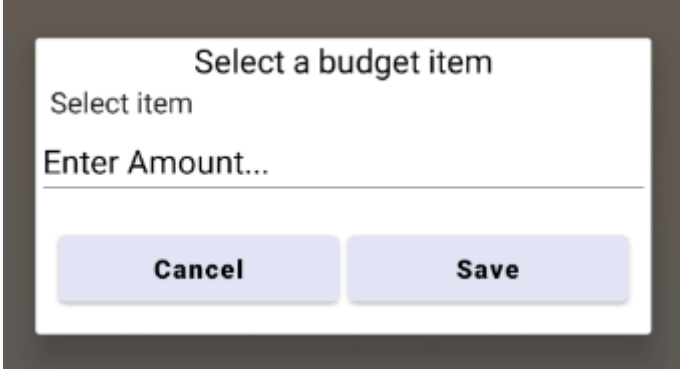
A screenshot of a mobile application dialog box. The dialog has a title "Select a budget item" in bold. Below the title is a label "Select item" and a text input field containing "Enter Amount...". At the bottom of the dialog are two buttons: "Cancel" and "Save". The dialog is set against a dark background.

Figura 25: Drop-down neintuitiv pentru utilizator

7 CONCLUZII

Aplicația a plecat de la ideea de a oferi utilizatorilor un mod eficient și ușor de folosit pentru a își gestiona finanțele personale. Principala nevoie la care răspunde aplicația este aceea de a înregistra o cheltuială cu ușurință, procesul fiind foarte intuitiv. Comparativ cu celelalte aplicații care există în market, aplicația propusă este gratis și nu conține reclame, iar codul sursă este publicat pe GitHub [5].

Principalele funcționalități îndeplinite de aplicație sunt următoarele: împărțirea bugetului pe categorii (Transport, Mâncare, Casă, Amuzament, Educație, Donație, Personal, Economii, Investiții și Altele), cu o viitoare implementare a editării categoriilor de către utilizator, vizualizarea bugetului rămas și a cheltuielilor totale, adăugarea unei cheltuieli pentru o anumită categorie, editarea unei cheltuieli din lista de cheltuieli, vizualizarea cheltuielilor pentru ziua, săptămâna sau luna în curs, accesarea unui istoric al cheltuielilor pentru o zi selectată din calendar și vizualizarea unor statistici bazate pe buget și pe cheltuielile în curs.

Pentru implementarea aplicației au fost folosite tehnologii de ultimă generație iar aplicația este suportată de telefoanele cu sistem android mai recent decât august 2017, ceea ce duce la un procent de acoperire destul de ridicat în rândul utilizatorilor de smartphone.

Motivația pentru a dezvolta această aplicație a fost, în primă fază, una personală, dar pe parcurs s-a dovedit a fi o nevoie, existând posibilitatea să concureze cu aplicațiile de pe market.

Aplicația oferă o interfață plăcută, ușor de folosit, fiind realizată integral în Android Studio în cod XML (auto-generat sau scris manual). Pentru partea de stocare a datelor s-a folosit serviciul pus la dispoziție de Google, numit Firebase, care pune la dispoziție o bază de date nerelațională (NoSQL), conectivitatea cu aceasta fiind integrată în mediul de dezvoltare. Codul aplicației a fost scris în Java îndeplinind cu atenție cerințele specificate. Pentru partea de grafice s-a folosit biblioteca AnyChart pusă la dispoziție pe GitHub, împreună cu exemple pentru folosire. Toate iconițele și pictogramele au fost preluate de pe site-urile menționate în bibliografie.

Pentru partea de evaluare, aplicația a reușit să respecte majoritatea cerințelor propuse iar printre principalele îmbunătățiri care vor fi aduse, se numără: editarea categoriilor de către utilizator, vizualizarea unui istoric pentru un interval de timp selectat, adăugarea unui venit recurent, adăugarea de notificări ca și aducere aminte pentru a înregistra cheltuielile, adăugarea de notificări atunci când bugetul scade sub o valoare setată pentru o anumită categorie, adăugarea mai multor tipuri de monede.

În concluzie, putem considera aplicația una relevantă în domeniu, care poate concura cu

aplicațiile actuale găsite pe market, o aplicație pentru gestiunea finanțelor personale fără a fi deranjat de reclame și una care să ne ajute în a economisi mai mulți bani.

BIBLIOGRAFIE

- [1] Biblioteca anychart pentru folosirea graficelor. <https://github.com/AnyChart/AnyChart-Android#readme>. Ultima accesare: 25 August 2021.
- [2] Financial stability in america. <http://fiftythirtytwenty.com/>. Ultima accesare: 25 August 2021.
- [3] Iconițe folosite în aplicație. <https://www.flaticon.com/>. Ultima accesare: 02 Septembrie 2021.
- [4] Pictograme folosite în aplicație. <https://fonts.google.com/icons>. Ultima accesare: 02 Septembrie 2021.
- [5] Cezar Gabriel Dimoiu. Repository personal github cu implementarea codului. <https://github.com/cezardimoiu/Money-management-app>. Ultima accesare: 09 Septembrie 2021.

A EXTRASE DE COD


```
<TextView
    android:id="@+id/signUpText"
    android:layout_width="60dp"
    android:layout_height="30dp"

    android:layout_marginBottom="40dp"
    android:text="Sign Up"
    android:textAlignment="center"
    android:textColor="#ff751a"
    android:textSize="14sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent" />
```

Figura 26: Cod XML - implementare textview

```
<androidx.appcompat.widget.AppCompatButton
    android:id="@+id/loginButton"
    android:layout_width="80dp"
    android:layout_height="48dp"
    android:layout_marginTop="30dp"
    android:background="@drawable/rounded_corners"
    android:text="Login"
    android:textAllCaps="false"
    android:textColor="#000000"
    app:layout_constraintEnd_toEndOf="@+id/editTextTextPasswordLogin"
    app:layout_constraintStart_toStartOf="@+id/editTextTextPasswordLogin"
    app:layout_constraintTop_toBottomOf="@+id/editTextTextPasswordLogin" />
```

Figura 27: Cod XML - implementare button



Create Account

✉ email


🔒 password

🔒 confirm password

Sign up

Already have an account?
[Sign In](#)

(a) Pagina de înregistrare



Login

Please sign in to continue

✉ email

🔒 password

Login

Don't have an account?
[Sign Up](#)

(b) Pagina de logare

Figura 28: Pagini de on-boarding

```

<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24"
    android:viewportHeight="24"
    android:tint="#ffffff" >
    <path
        android:fillColor="#ffffff"
        android:pathData="M19,13h-6v6h-2v-6H5v-2h6V5h2v6h6v2z"/>
</vector>

```

Figura 29: Cod XML - implementare floating action button

```

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >
    <Spinner
        android:id="@+id/items_spinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:entries="@array/items"
        android:layout_gravity="center"
        android:background="#FFF" >
    </Spinner>

```

Figura 30: Cod XML - implementare spinner în interiorul unui LinearLayout

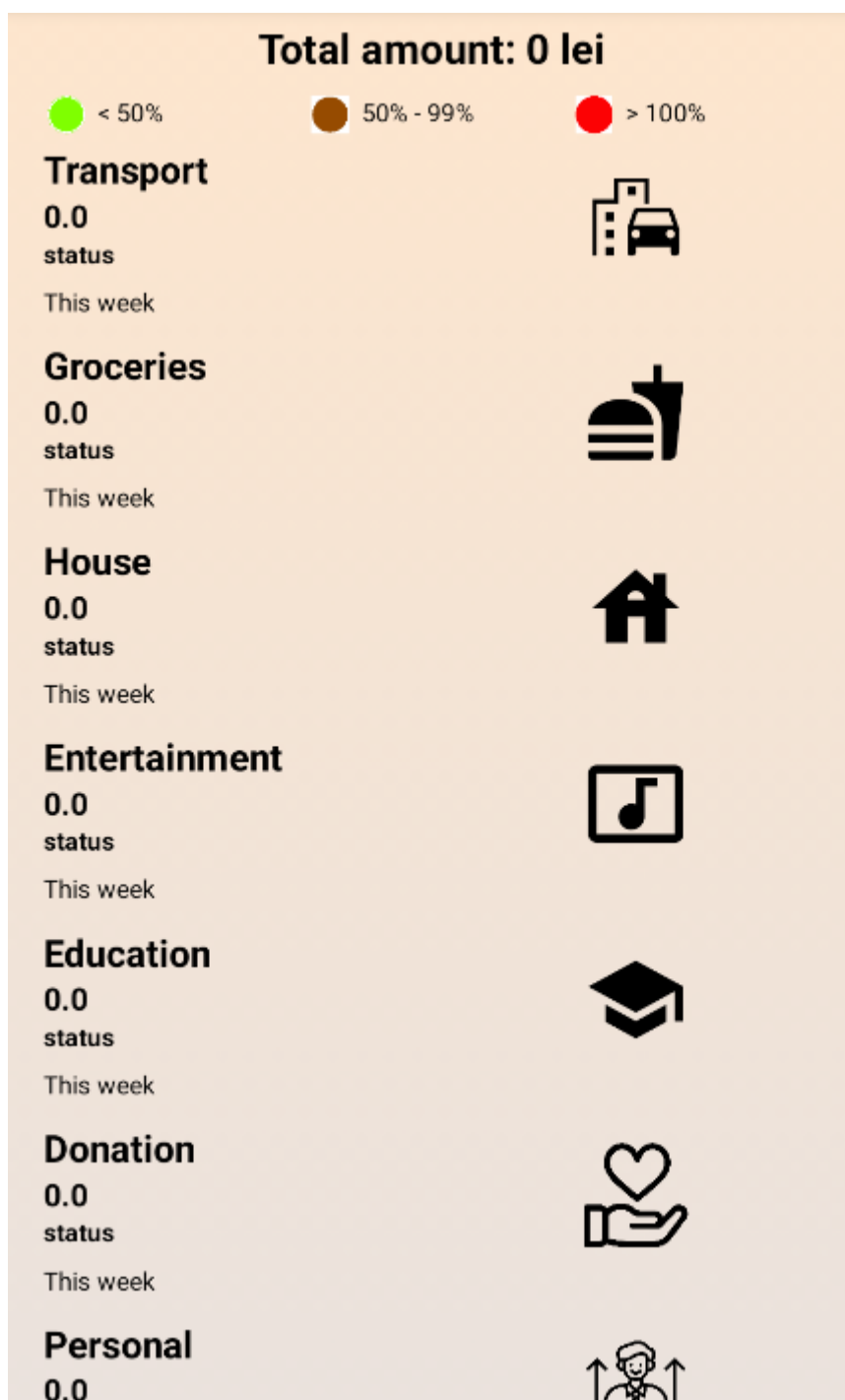


Figura 31: Design analytics