
ACADEMIA DE STUDII ECONOMICE BUCUREȘTI
FACULTATEA DE CIBERNETICĂ, STATISTICĂ ȘI INFORMATICĂ ECONOMICĂ
SPECIALIZAREA INFORMATICĂ ECONOMICĂ

# LUCRARE DE LICENȚĂ

**COORDONATOR ȘTIINȚIFIC**

Conf. univ. dr. Florea Alexandra-Maria-Ioana

**ABSOLVENT**

Țone Iulia-Paula

BUCUREȘTI

2022


ACADEMIA DE STUDII ECONOMICE BUCUREȘTI
FACULTATEA DE CIBERNETICĂ, STATISTICĂ ȘI INFORMATICĂ ECONOMICĂ
SPECIALIZAREA INFORMATICĂ ECONOMICĂ

# LUCRARE DE LICENȚĂ

# APLICAŢIE INFORMATICĂ DE MOBILE BANKING

**COORDONATOR ȘTIINȚIFIC**

Conf. univ. dr. Florea Alexandra-Maria-Ioana

**ABSOLVENT**

Țone Iulia-Paula

BUCUREȘTI

2022

# Cuprins

# INTRODUCERE

De-a lungul timpului fiecare societate și civilizație a crescut, s-a dezvoltat, dar a cunoscut și perioade de declin datorită și din cauza tehnologiei. Tehnologia în sine a existat din timpuri imemoriale, doar că nu chiar în aceeași formă pe care o cunoaștem astăzi. Alfred North Whitehead, un renumit filosof britanic, spunea că *„civilizația avansează prin extinderea numărului de operațiuni importante pe care le putem efectua fără să ne gândim la ele”*[1] - un fapt pe care, cel mai adesea, nu-l conștientizăm, dar care este atât de simplu și atât de înrădăcinat în vieţile noastre de zi cu zi.

Mesopotamia, considerată drept una dintre „leagănele civilizației”, a fost martoră primelor inovații tehnologice care au apărut vreodată și care sunt folosite chiar și în ziua de astăzi. Referindu-ne la sumerienii care au întocmit nenumărate invenţii şi inovaţii, obiecte și concepte de bază ale timpului nostru, dar care în vremurile acelea reprezentau adevărate izbânzi tehnologice menite să schimbe viaţa omenirii. Spre exemplu, roata – de o importanță fundamentală pentru progresul ulterior al tehnologiei şi al civilizației umane, în general, încât simbolizează în conștiința colectivă, însăşi INVENȚIA, cu alte cuvinte, acea idee revoluționară care schimbă din temelii universul tehnologic de până atunci şi oferă progresului un imbold fără precedent, mai apoi navele, sistemul de irigații și chiar cel mai vechi „limbaj de programare” – scrisul[2].

De atunci și până acum, tehnologia a trecut prin mai toate etapele vieții sale, de la „copilărie”, la „pubertate și adolescență”, și este în continuă evoluție, fiind foarte important să ţinem pasul cu ea. Ceea ce noi numim tehnologie modernă „a deschis calea” în ultimele două decade pentru dispozitivele multifuncționale precum calculatorul, smartphone-ul, tableta și smartwatch-ul, care au devenit din ce în ce mai mici, mai portabile, mai rapide și mai puternice, facilitând astfel reducerea considerabilă a timpului și efortului consumat pentru realizarea unor obiective mult mai costisitoare în trecut, așa cum spunea și filozoful Alfred North Whitehead.

[1] (Psihologia persuasiunii - Robert Cialdini, n.d.)
[2] (Wardynski, Technology and Society: How Technology Changed Our Lives, 2019)

Unul dintre lucrurile pe care le putem face acum printr-un simplu gest este reprezentat de transferul bancar, dar și gestionarea tuturor resurselor și activităților noastre bancare cu ajutorul Internet Banking-ului, acesta poate fi accesat fie prin intermediul unui navigator web, ori cu ajutorul unei aplicaţii furnizate de către bancă și, prezentat într-un mod foarte succint, oferă facilități bancare virtuale. Acest concept care a reușit să revoluționeze universul bancar și financiar a apărut pentru prima dată în Statele Unite ale Americii între anii 1994-1996[3] și s-a răspândit rapid în numeroase țări din toate colțurile lumii, ajungând puțin mai târziu, dar inevitabil, și în România. Online banking le-a permis oamenilor să poată accesa detaliile contului lor bancar, să transfere și să-și gestioneze rapid banii, indiferent unde se află, în timp ce statele și autoritățile de supraveghere, ca instituții în sine, se bucură de monitorizarea mult mai eficientă a activității financiare a cetățenilor.

La nivel internațional, Online Banking-ul a declanșat cu adevărat o revoluție în ceea ce privește sistemul bancar și financiar, unele țări dorindu-și chiar eliminarea completă a tranzacţiilor cu bani lichizi şi, aşadar, adoptarea generalizată a cardurilor, cu scopul de a deveni cât mai curând primele societăți fără numerar (”cashless society”). Bătălia se dă între suita de țări scandinave care, conform unui studiu publicat în aprilie 2020[4], înregistrează un procent de utilizare a online banking-ului de 85-95%, Norvegia plasându-se pe primul loc în top.

La celălalt capăt al clasamentului se regăsește și țara noastră, unde ocupă al patrulea loc din coadă, cu o rată de adopţie a soluţiilor şi a aplicaţiilor de banking online de doar 8%[5]. Este cunoscut faptul că România se află printre țările în care încă există o reticență semnificativă în ceea ce privește această tranziție de la fizic la online.

În contextul provocat de pandemia de COVID-19[6], care a durat mai bine de doi ani, a fost vitală exploatarea instrumentelor moderne cu ajutorul cărora să se elimine barierele fizice puse în calea intermedierii dintre clienți și bancă. Pandemia a accelerat maturizarea pe toate fronturile[7]. Ne-a arătat că putem rezista, că ne putem adapta și chiar prospera și în perioade pe care nici nu ni le puteam imagina. Lunile martie şi aprilie 2020, având în vedere noul mod de

[3] (Sarreal, History of Online Banking: How Internet Banking Went Mainstream, 2019)
[4] (Statista Research Department, 2021)
[5] (Statista Research Department, 2021)
[6] (Digitalizarea serviciilor bancare în România, accelerată de epidemia de coronavirus, 2020)
[7] (Voinea, Anul relansării în banking, 2021)

viaţă al românilor, survenit odată cu instituirea stării de urgență, au reprezentat o perioadă care a schimbat comportamente şi obiceiuri de consum şi, concomitent cu acestea, modul de a face banking. Cifrele băncilor românești au arătat faptul că perioada de pandemie, în special începutul acesteia, a generat un salt uriaș în ceea ce privește adopţia online banking-ului, progres care altfel ar fi putut dura ani de zile şi care reușește să aducă România mai aproape de ţările vest-europene privitor la focusul pe digitalizare.

Dacă ne întrebăm ce înseamnă acum, după ce pandemia a accelerat digitalizarea, banking la distanță sau cum va arăta bankingul postpandemie, putem spune ca schimbările structurale accelerate de pandemia de coronavirus (COVID-19) au modificat substanțial modul în care clienții interacționează cu banca și cu banii lor. Aşadar, planul este ca toate operațiunile de bază, fără nicio valoare economică adăugată pentru clienți, să fie realizate printr-o aplicație și într-un mod cât mai eficient și prietenos posibil, iar operațiunile mai complicate, care necesită consultanță sau decizii din partea clienților, să se facă în agenții, într-o manieră care conduce la o experiență cât mai bună. Nu mai avem cum să ne întoarcem la comportamentul financiar pe care îl aveam ante-pandemie, deci, perioada de după COVID-19 va fi o continuare a acestui parcurs, cu accent pe zona de digital şi mai ales pe aplicaţia de mobile banking care încă se află într-un sistem de dezvoltare continuă cu scopul de a aduce clienților bankingul cât mai aproape cu putință.

Conchizând, ceea ce putem observa acum și în domeniul bancar este faptul că[8], în general, clienții preferă o abordare integrată a cât mai multor servicii, toate sub umbrela unui singur instrument financiar. În calitate de clienți, ne dorim din ce în ce mai mult să accesăm orice, oricând, oriunde, iar totul să se întâmple foarte rapid. În consecință, adevărata provocare este chiar aceea de a integra într-un mod cât mai simplu și eficient toate soluțiile la universul de nevoi al clienților într-o singură aplicație. Drept urmare, dezvoltarea unei aplicații de mobile banking care face experiența de utilizare foarte plăcută prin prisma faptului că are o interfață atractivă, prietenoasă și, în același timp, foarte intuitivă și ușor de folosit pentru orice utilizator, indiferent de categoria de vârstă, este o alegere întemeiată în ceea ce privește tema lucrării de licenţă.

[8] (Barliga, 2021)

# CAPITOLUL 1. DESCRIEREA PROBLEMEI ECONOMICE

## 1.1. Descrierea domeniului abordat

Conceptul de Mobile Banking[9] reprezintă un serviciu bancar care oferă clienților accesul non-stop la contul bancar, la orice oră a zilei și în orice zi, inclusiv cele nelucrătoare, prin intermediul unui dispozitiv portabil, de regulă un smartphone, de oriunde există o conexiune la Internet. Din aplicația de mobile banking poți face acum mai toate operațiunile disponibile și la ghișeul unei bănci: poți verifica rapid câți bani ai disponibili în cont sau pe card (soldul contului/cardului), poți vedea toate tranzacțiile făcute anterior (extras de cont), poți face plăți (inclusiv de facturi, taxe, impozite etc.) sau transferuri de bani, poți deschide depozite sau poți obține chiar și un credit.

Este chiar greu de imaginat faptul că, la o dată recentă, a existat un moment în care toate operațiunile bancare obișnuite și tot ceea ce se leagă de conceptul de „bancă" ca instituție financiară ducea cu gândul la un singur lucru, și anume, agenția bancară, clădirea propriu-zisă a băncii. Cu toate că nu au trecut decât în jur de 20 de ani de la digitalizarea proceselor din zona financiar-bancară, apariția conceptului de mobile banking pare și, totodată, se simte un moment mult mai îndepărtat în timp. Doar gândul că părinții noștri erau nevoiți să facă atâtea vizite la unitățile bancare, unde de cele mai multe ori se aștepta îndelungat la coadă alături de alți clienți, toate aceste lucruri doar pentru plata facturilor lunare, lucru care se face atât de simplu și rapid acum, ne face să îi compătimim. Clienții mai tineri, îndeosebi generația Z, au fost familiarizați cu tot ceea ce înseamnă domeniul bancar încă de la bun început prin intermediul online-ului, bucurându-se din plin de comoditatea venită odată cu avansul tehnologic, căci prin doar câteva atingeri de ecran pot verifica soldul contului, pot achita facturi, pot trimite bani celor dragi, oriunde s-ar afla aceștia, ca atare, nici nu vor să se gândească ce presupuneau realmente toate aceste operațiuni banale în trecut.

Își poate imagina cineva cum ar fi fost viața actualmente, dar cu atât mai mult pe durata pandemiei, fără posibilitatea plății online cu cardul bancar prin platformele securizate de

[9] (Ce inseamna mobile banking?, 2021)

procesare plăți sau fără posibilitatea de a verifica soldul și de face transferuri bancare către diverși beneficiari doar de pe telefon? Este cert, ne place digitalizarea bankingului, dar câți dintre noi cunoaștem cu adevărat istoria din spatele acestui concept care a ajuns să fie astăzi sinonim cu normalitatea atunci când ne referim la toate operațiunile de zi cu zi?

Evoluția serviciilor bancare digitale a început prin anii 1980, când definiția și practica produselor și serviciilor bancare bazate pe Internet erau cu totul altele față de cele existente acum. Versiunea prematură și de-a dreptul diferită a ceea ce utilizăm noi astăzi a început în New York în anul 1981. Acesta a fost primul oraș unde s-a testat modul inovator de a face afaceri oferind servicii bancare la distanță doar unui număr strict limitat de clienți de la cele mai renumite patru bănci. Mai apoi, în 1983, a urmat Marea Britanie cu ceea ce se intitula „Homelink", acesta presupunea conectarea folosind un televizor și un telefon pentru a putea efectua transferuri bancare și pentru a plăti facturi[10].

În octombrie 1994, Stanford Federal Credit Union (sau Stanford FCU) a devenit prima instituție financiară din lume care a pus la dispoziție servicii bancare bazate pe Internet tuturor clienților săi. La interval de un an, Presidential Bank a devenit prima bancă din America care a oferit clienților acces la servicii bancare online asociate contului. Avantajele oferite de aceste sisteme bancare online au început să prindă într-un mod facil la public, urmând ca multe alte bănci să preia în scurt timp inițiativa și modelul de afaceri implementat de Presidential Bank[11].

Așa cum era de așteptat, clienții au început să remarce și să aprecieze de îndată comoditatea și beneficiile oferite de utilizarea serviciilor bancare online, ratele dobânzilor mai avantajoase decât în cazul băncilor tradiționale, accesul facil și rapid la conturi, dar și transferurile bancare online fiind doar primele funcționalități disponibile.

Între timp a apărut prima și, totodată, cea mai „modestă" sau „simplistă" formă de mobile banking, și anume SMS banking, sau posibilitatea de a trimite bani printr-un simplu mesaj text. O idee transpusă în aplicațiile de mobile banking din prezent, având în vedere faptul că viteza este numele jocului atunci când vine vorba de transferuri de bani pe mobil, idee susținută și de Mugur Podaru, director Operaţiuni la Distanţă în cadrul CEC Bank, prin afirmația: „Să faci un

[10] (The Evolution of Online Banking, 2012)
[11] (Sarreal, History of Online Banking: How Internet Banking Went Mainstream, 2019)

transfer de bani ar trebui să fie la fel de uşor ca trimiterea unui SMS sau apelarea unui prieten din agenda telefonică."[12] În anul 1999, odată cu apariția conceptului de mobile web, pe scurt, posibilitatea de a accesa site-uri web prin intermediul telefoanelor mobile, a început să se dezvolte cu adevărat intens sistemul de deservire bancară la distanţă Mobile Banking[13]. În perioada următoare tot mai multe bănci au început să ofere diverse servicii prin intermediul site-urilor pentru telefon, după care, în mai 2011, apărând și prima aplicație mobilă dedicată în totalitate operațiunilor bancare, lansată de RBS (Royal Bank of Scotland) mai întâi pentru utilizatorii iOS, și apoi pentru utilizatorii de Android și BlackBerry[14].

În 2010, Fiserv, o multinaţională americană care furnizează servicii de tehnologie financiară (fintech) clienților din piața serviciilor financiar-bancare, a efectuat un sondaj privind evoluția banking-ului digital și metodele de plată preferate de consumatori. În urma studiului s-a constatat faptul că în decurs de 10 ani atât serviciile bancare online, cât și cele mobile, au crescut într-un ritm mai accelerat chiar și decât Internetul. Până în prezent, acestea au continuat să se dezvolte cu o viteză uimitoare. Totodată, atunci când ne referim la preferințele consumatorilor, se poate observa faptul că opțiunea de plată digitală a devenit rapid majoritară, crescând de la 12% în 2000, la nu mai puțin de 45% în 2010[15] [16].

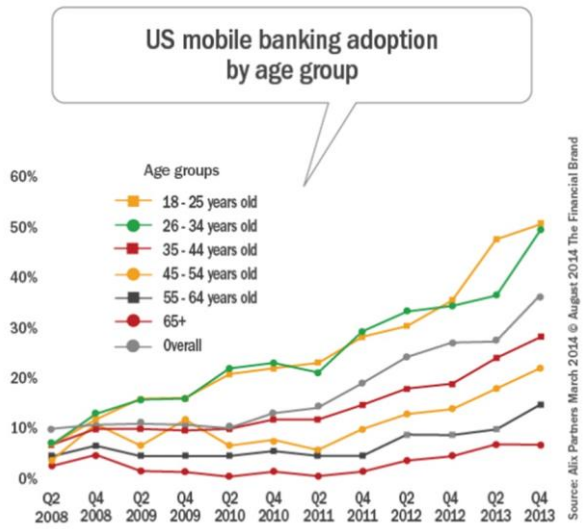



*Figura 1 - Evoluția gradului de adopție a serviciului de mobile banking între anii 2008-2013 (Marous, 2014)*

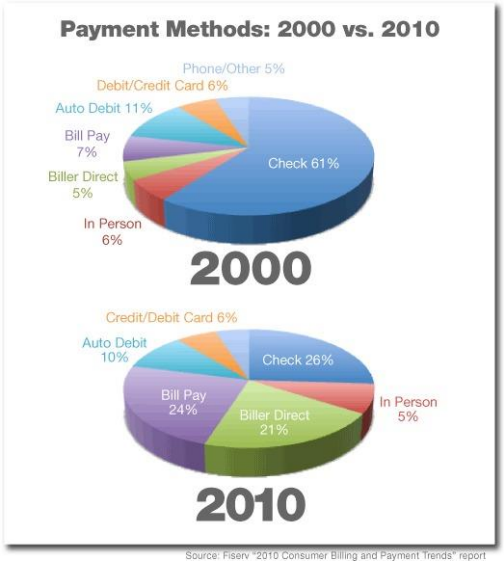



*Figura 2 - Evoluția tendințelor privind modalitățile de plată preferate de clienți (Pilcher, 2010)*

[13] (Mobile banking, 2021)
[14] (Pankhudi Pandey, 2019)
[15] (Pilcher, 2010)
[16] (Phenomenal Growth for Bill Pay, 2010)

Fără îndoială, în zilele noastre, serviciile bancare online și mobile sunt categoric cu mult mai comode și mai rapide decât orice bancă tradițională – de fiecare dată când pornim un telefon, o tabletă sau un laptop, avem propria bancă oricând și de oriunde la dispoziția noastră, ba chiar am ajuns ca de cele mai multe ori, atunci când avem puțin timp liber și nu mai știm ce să „butonăm", să mai efectuăm câteva click-uri sau atingeri pe telefon și să ne verificăm soldul curent, acest lucru întâmplându-se numai așa, din plictiseală, un simplu fapt care, totuși, în trecut putea dura ore întregi. Marele avantaj al banking-ului online și mobil, ba chiar printre cele mai iubite avantaje de către clienți, este posibilitatea de a plăti online în doar câțiva pași toate facturile. Fără a mai fi nevoiți să ne deplasăm până la cea mai apropiată locație oficială sau cele mai apropiate magazine din zona noastră care oferă serviciile de plăți facturi, căci astăzi toate facturile pot fi plătite doar prin câteva click-uri sau atingeri de ecran din confortul casei noastre, ba chiar cu un singur gest pentru că avem și posibilitatea de a automatiza plățile recurente. Totodată, prin prisma plăților digitale, putem acum organiza și gestiona mult mai facil toate facturile, toate acestea pe lângă faptul că ne putem forma o imagine de ansamblu mai aerisită și adaptată vremurilor moderne, având istoricul plăților tot timpul la îndemână.

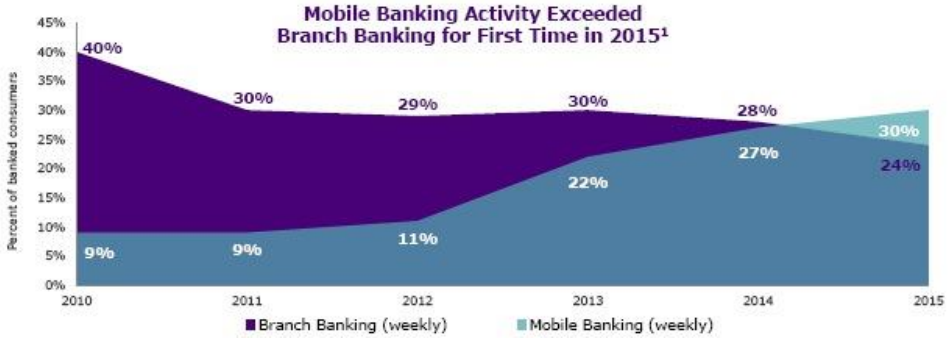



*Figura 3 - 2015, anul în care mobile banking-ul a depăşit banking-ul fizic (ACCESSWIRE, 2017)*

Un alt avantaj foarte important este reprezentat de faptul că nu are orar de funcționare, fiind disponibil 24/7, atât timp cât există o conexiune la Internet. Mai mult decât posibilitatea de a utiliza platforma de online banking 24 de ore pe zi, unele bănci, precum BCR, FirstBank, Banca Transilvania sau Raiffeisen Bank, se întrec pe sine în ceea ce privește serviciul de relații clienți și oferă suport telefonic disponibil 24 de ore din 24, 7 zile din 7.

Serviciile bancare mobile le oferă clienților viteză și eficientă, căci vorbim despre capacitatea de efectua aproape orice tranzacție bancară sau chiar de a solicita un credit nou fără a mai fi nevoie să așteptăm la coadă alături de alți clienți într-o sucursală bancară. Atât timp cât avem o conexiune la Internet și un smartphone sau o tabletă, ne putem accesa toate conturile deținute la bancă în doar câteva secunde, putem solicita un nou card de credit care să ne vina acasă, prin curier, sau putem să deschidem un depozit de economii, să transferăm fonduri între conturile proprii atât in lei, cât și în valută, după caz, și, de asemenea, să putem urmări un istoric complet al tuturor tranzacțiilor noastre, alături de posibilitatea de a deține chiar și un extras de cont generat și printat de noi din aplicație, totul într-un singur loc.

Potrivit unui studiu recent realizat de SAS[17], lider global în soluții de analiză avansată de date - Business Analytics, circa 30% dintre românii care utilizează servicii bancare au declarat faptul că vor renunța la interacțiunea fizică cu banca, continuând să folosească necontenit serviciile digitale sau aplicația mobilă puse la dispoziție de instituția financiară, obiceiurile adoptate în perioada pandemiei urmând să fie păstrate și după trecerea crizei sanitare. Important de menționat, în contextul de față, este și faptul că 20% dintre clienții din România au început să utilizeze un serviciu digital sau o aplicație mobilă pentru prima dată la începutul perioadei pandemice, când, forțați de împrejurări, au fost realmente nevoiți să îmbrățișeze digitalizarea banking-ului. Totodată, se așteaptă ca această tendință de consolidare să continue în viitor și să se dezvolte pe termen lung. Cu toate acestea, dacă circa 18% dintre români au declarat că vor utiliza în paralel serviciile digitale/aplicația de mobile banking și interacțiunile fizice cu reprezentanții băncii, cei mai mulți respondenți (27%) spun că vor înlocui complet deplasarea fizică la agenția bancară cu serviciile digitale. Este îmbucurător faptul că din ce în ce mai mulți români profită de avantajele instrumentelor bancare digitale, însă nu trebuie să pierdem din vedere faptul că România rămâne unul dintre cele mai slab dezvoltate state membre ale UE în ceea ce privește ideea de digitalizare a întregului complex bancar, astfel că încă mai este nevoie să ne prezentăm fizic la sucursală pentru anumite operațiuni care, de pildă, în alte țări se rezolvă astăzi în doar câteva minute, de acasă. Într-adevăr, este bine cunoscut și faptul că în România încă există, din păcate, și o reticență greu de diminuat asupra serviciilor online în general, reticență explicată de motivele menționate în figura de mai jos.

[17] (Aproape un român din trei renunță la interacțiunea cu băncile fizic, trecând la servicii digitale – studiu SAS, 2022)

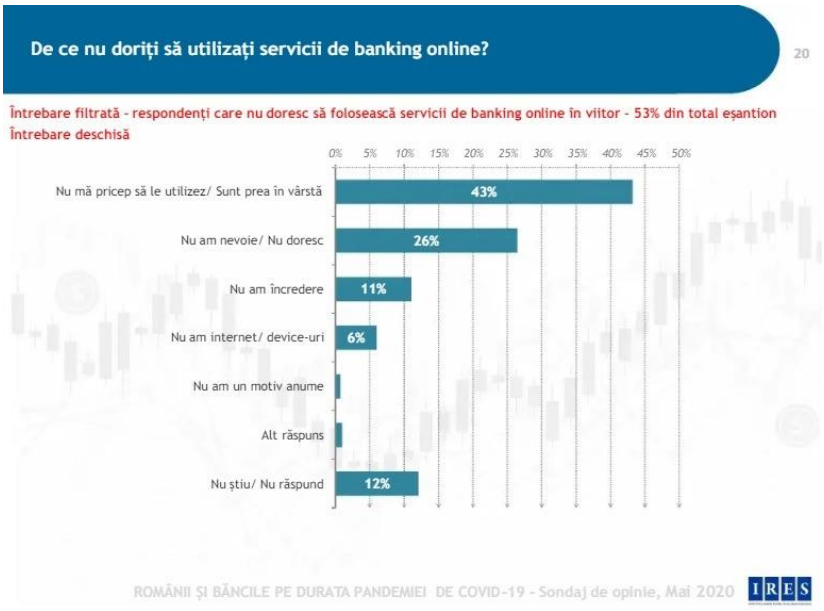



*Figura 4 - Motivele pentru care românii nu doresc să utilizeze Internet Banking sau Mobile Banking (NoCash, 2020)*

## 1.2. Prezentarea activităţii care va fi informatizată

Tehnologia digitală a schimbat obiceiurile consumatorilor cu mult înainte de pandemia de COVID-19, în ultimii ani smartphone-ul devenind un agregator în jurul căruia gravitează pentru majoritatea persoanelor viața modernă, iar aplicațiile bancare mobile se așază în fruntea instrumentelor de interacțiune cu clienții pentru companiile din cele mai variate domenii de activitate.

Proiectul de licență constă într-o aplicație de mobile banking care își propune, în primul rand, să ofere utilizatorilor opțiunea de a accesa de la distanță, din confortul casei lor, produse și servicii bancare esențiale, dar cuprinde și funcționalități unice, pe lângă o interfață atractivă, prietenoasă și, în același timp, foarte intuitivă și ușor de folosit pentru orice utilizator chiar de la prima accesare.

Aplicația este dezvoltată în limbajul Java cu ajutorul mediului de dezvoltare Android Studio, iar ca să devină receptivă și mobilă se implementează o bază de date Firestore în Google Firebase.

La deschiderea aplicației, se lansează fereastra de Login, unde utilizatorul se conectează folosind numărul său de telefon. După parcurgerea pașilor de autentificare, este direcționat către pagina principală, unde sunt afișate toate informațiile profilului său și de unde pot fi accesate serviciile disponibile în cadrul aplicației, și anume:

- soldul curent al contului bancar principal;
- informațiile de pe card (numărul de card, data expirării, codul de securitate etc.), alături de posibilitatea de a le ascunde sau de a le face iar vizibile prin apăsarea unui buton;
- la apăsarea scurtă pe înfățișarea cardului se afișează toate retragerile și depunerile de numerar realizate la bancomate (ATM-uri), iar la apăsarea mai lungă se afișează informațiile necesare pentru realizarea unui transfer bancar (de exemplu, IBAN-ul);
- la apăsarea mai lungă, posibilitatea de a afișa, și mai apoi de a copia în clipboard prin apăsarea unui buton informațiile necesare pentru realizarea de plăți sau alte tipuri de operațiuni de transfer (IBAN-ul, numărul de card, codul CVV etc.);
- vizualizarea ultimelor tranzacţii efectuate cu cardul;
- accesul la depozitele de economii deschise, dar și posibilitatea de a deschide unul nou sau de a închide unul sau mai multe dintre acestea;
- o serie de butoane utilizate în scopul de a fi redirecționat către alte interfețe și funcționalități disponibile.

Soluția de mobile banking permite inițierea de plăți către diverși beneficiari, fie personali, definiți de utilizator, fie publici, definiți de bancă/furnizori de servicii și utilități, iar pentru cei din urmă este implementat un ecran specific, Plăți, unde datele bancare necesare pentru a efectua transferul se completează automat de către aplicație după selectarea unui furnizor din lista de parteneri, căci în aceasta se regăsesc și principalele detalii, precum numele și IBAN-ul.

Fereastra Tranzacții permite vizualizarea istoricului actualizat și complet al tranzacțiilor, dar oferă și posibilitatea de a le filtra și sorta după multiple criterii specifice. Filtrarea se realizează în funcție de mai multe criterii (o dată calendaristică specifică, venituri/cheltuieli, facturi și, respectiv, printr-o casetă de căutare prin care pot fi regăsite toate tranzacțiile a căror

denumire conține șirul de caractere căutat de utilizator), iar sortarea se execută crescător sau descrescător, în funcție de suma de bani. Pentru a servi cât mai bine nevoilor utilizatorilor, cele două module de sortare și filtrare funcționează 100% sincronizat și concomitent, fiind posibilă selectarea mai multor filtre și moduri de sortare în același timp. Tot de aici utilizatorul poate solicita un extras de cont.

În ultima fereastră a meniului aplicației, utilizatorul poate consulta grafice construite pe baza istoricului de cheltuieli. La deschidere este reprezentat graficul general, care oferă o imagine de ansamblu pe baza întregului istoric de cheltuieli, iar mai apoi utilizatorul poate solicita vizualizarea graficelor aferente fiecărei luni în parte. Totodată, elementul inovator al acestei aplicații, care nu este implementat de nicio altă aplicație existentă pe piață, este capacitatea de a oferi utilizatorului o predicție aproximativă a sumei care va fi cheltuită în luna următoare, în această predicție ținându-se cont de obiceiurile financiare înregistrate în istoricul acestuia până în momentul prezent, și este implementată cu ajutorul unui algoritm în limbajul de programare Python integrat în Android SDK.

## 1.3. Comparație cu alte produse/aplicații software existente în domeniul fintech-urilor

În prezent, majoritatea băncilor oferă clienților lor atât posibilitatea de a accesa o platformă web de internet banking, cât și o aplicație de mobile banking. În cele ce urmează, sunt prezentate trei abordări diferite de mobile banking disponibile în acest moment, întrucât fiecare bancă din clasamentul celor mai utilizate și cunoscute din România implementează o altfel de soluționare în ceea ce privește interfața cu utilizatorul și funcționalitățile aplicației mobile.

- **ING HomeBank**

ING Bank[18] are una dintre cele mai complexe aplicații de mobile banking din piața din România, oferind o varietate de funcționalități importante pentru clienții săi și promițând,

[18] (ING Home'Bank)

totodată, faptul că anul acesta vor veni altele și mai importante pentru cei interesați. Fără îndoială, ING Bank nu are în aplicațiile sale chiar tot ceea ce este oferit și de către aplicațiile altor bănci din România, iar acest lucru este perfect normal, dar obiectivele setate pentru perioada următoare ne trimit cu gândul la faptul că vom avea parte de surprize foarte interesante anul acesta de la banca din România. O parte dintre funcționalitățile puse la dispoziție utilizatorilor prin aplicația ING HomeBank fiind următoarele:

- Oferă opțiunea de autentificare biometrică (cu amprentă digitală sau recunoaștere facială) pentru a intra într-un mod cât mai rapid și ușor în aplicație, fără a mai fi nevoie de parole cu combinații anevoioase de cifre, litere mari sau mici, ușor de uitat, și, cel mai important, accesul este sigur, amprentele digitale fiind unice fiecărui individ.

- Se pot plăti facturile online, direct din aplicație, într-un mod cât mai simplu și rapid. Se selectează furnizorii, se vizualizează facturile emise și mai apoi se decide când se va efectua plata. Totodată, utilizatorii au întotdeauna la îndemână evidența cheltuielilor făcute cu cardul, respectiv a sumelor debitate din cont, întrucât au posibilitatea de a genera extrasele de cont în format Excel sau PDF.

- Atunci când este nevoie de IBAN-ul contului personal, se poate afla rapid direct din aplicație, de unde se poate copia în clipboard sau trimite prin altă aplicație, dar și datele cardului.

- Pune la dispoziție opțiunea Reîncărcare cartela de telefon – a ta sau a unei persoane dragi.

- Dacă se dorește modificarea PIN-ului cardului, se poate rezolva direct în aplicație în doar câteva secunde.

- Economisirea cu Round Up. Fiecare plată efectuată cu cardul de debit la POS sau online, ori cu telefonul este rotunjită până la primul multiplu de 5, iar diferența merge din contul curent direct în contul de economii.

- Se pot retrage bani fără card, de la orice bancomat ING, folosind doar aplicația. Se intră în fereastra Plăți, apoi se alege opțiunea Retragere fără card, se va genera un cod care este valid timp de 15 minute, se alege suma dorită și procesul este încheiat!

- În cazul în care utilizatorul și-a pierdut cardul de plastic, se poate emite un card nou direct din aplicație, fără să fie necesară deplasarea la o sucursala bancară, și va fi livrat acasă, prin curier. Sau, de asemenea, se poate emite un card virtual care oferă posibilitatea de a-l folosi imediat.

- Se poate ascunde soldul de privirea celor curioși prin atingerea cu degetul a ochișorului din dreapta-sus a ecranului.

- Dacă sunt necesare transferuri periodice, se pot adăuga la plăți recurente și astfel scapi de grija lor, mai trebuie doar să planifici ziua în care să se efectueze plata și frecvența.

- Pentru a nu mai duce grija cardului, îl poți ține în telefonul tău, activând ING Pay se poate utiliza direct din aplicație sau, mai există posibilitatea de a-l introduce în Google Pay, respectiv Apple Pay, și apoi poți să plătești cu el la orice POS contactless.

- AliasPay - serviciul prin care se pot face transferuri interbancare folosind doar numărul de telefon (astfel nemaifiind nevoie de cunoașterea IBAN-ului beneficiarului, ci doar îl vezi și îl confirmi înainte să semnezi acea plată).

### ➢ George de la BCR

Platforma George de la BCR[19] este probabil cea mai cunoscută în acest moment, iar acest lucru se întâmplă datorită strategiilor puternice de marketing de care a avut parte încă de la lansare, aceasta prezentându-se pretutindeni ca fiind: „George, primul banking inteligent din România”.

Pe lângă funcţiile deja obişnuite ale oricărei aplicaţii bancare – autentificare extrem de simplă, convenabilă şi, în acelaşi timp, sigură, prin amprentă/Face ID sau orice altă opțiune disponibilă pe smartphone-ul utilizatorului (e.g. PIN, Pattern), transferuri și plăți în lei și valută, inclusiv plăți automate, plăți SEPA și plăți de

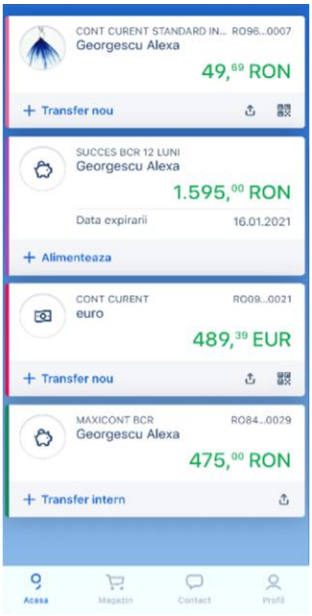


*Figura 5 - Interfață George BCR*

[19] (Aplicația George)

facturi, faptul că reține destinatarii transferurilor și completează automat datele necesare, generare rapidă a unui extras de cont în aplicația mobilă, căutare simplă, de tipul motoarelor web, după nume sau un comerciant, în arhiva tranzacțională, gestionarea completă a cardurilor în aplicație, 100% online, prin posibilitatea blocării și deblocării lor, a recuperării PIN-ului, setarea limitelor de card, iar toate cardurile de debit pentru persoane fizice noi sau cele care trebuie reînnoite sunt livrate prin curier alături de instrucțiunile de activare, la domiciliul clienților, solicitarea unui credit, unui overdraft sau a unei refinanțări credit doar prin câteva atingeri pe ecran, direct din magazinul George, personalizarea șabloanelor de plată și a tranzacțiilor, precum și clasificarea tranzacțiilor, culori și imagini de profil pentru personalizarea conturilor, dar și posibilitatea configurării aspectului ecranului principal etc., aspectele majore care îl diferențiază pe George de alte aplicații de pe piața bancară sunt reprezentate de implementarea funcționalităților precum: George Multibanking, prin intermediul căreia utilizatorii au posibilitatea de a vizualiza și conturile pe care le-au deschise la alte bănci pentru a nu mai fi nevoiți să treacă de la o aplicație la alta (la momentul acesta sunt disponibile doar conturile de la UniCredit, ING, BT și Revolut), George Moneyback, programul de loialitate care oferă bani înapoi, ca reducere, clienţilor BCR care folosesc plata cu cardul, serviciul „Adaugă bani instant” – atunci când clientul are nevoie de bani în contul său George nu mai este nevoie să se deplaseze până la bancomat, fiindcă dacă deține un card la o altă bancă are capacitatea de face transfer de bani în contul său George instant, posibilitatea de a efectua transferuri prin scanare de IBAN/Cod QR/Cod de bare/cod QR din galerie, primirea de informații despre toate acțiunile din conturi prin notificări de tip push, dar și faptul că oferă un proces simplificat de înrolare a cardului bancar în iOS Wallet, direct din App George (Apple Pay) – soluție denumită In-App Provisioning, pe lângă George Pay (soluția de plată a băncii pentru terminale Android), pentru plățile contactless cu telefonul mobil. Aşadar, clienților nu le mai rămân prea multe motive pentru a merge într-o sucursală, în împrejurările în care se pot bucura din confortul casei lor de cele mai importante servicii ale unei bănci.

- **Revolut**

Revolut[20] este prima super-aplicaţie financiară care ajută în momentul actual peste 18 milioane de clienţi din toată lumea, dintre care peste 1,7 milioane de utilizatori activi în România,

[20] (Revolut, 2022)

să își exploateze la capacitate maximă fondurile de care dispun cu ajutorul serviciilor inovatoare oferite de fintech-ul britanic. Dacă ne întrebăm care este diferența dintre serviciile de banking oferite de Revolut și cele oferite de alte bănci obișnuite/„tradiţionale”, Leonardo Da Vinci a spus că „Simplitatea este sofisticarea extremă”, iar acesta este elementul diferențiator de bază, raportându-ne la ceilalți competitori de pe piaţă. Magia Revolut constă tocmai în faptul că s-au dedicat obiectivelor de a demistifica și a debirocratiza industria financiară, care pentru o mare parte dintre clienți a fost cândva imposibil de descifrat, și au revoluţionat-o într-o aplicație ușor de înțeles și de folosit, la care are acces orice persoană, oricând și prin care este capabil să dirijeze în mod direct toate aspectele contului.

֍ Cardul Revolut – beneficiul suprem

Contului romanesc îi poate fi atribuit și un IBAN Revolut în lei, lucru care le permite utilizatorilor conaţionali să poată transfera bani în cont de pe orice card/din orice bancă din lume, mai mult, Revolut pune la dispoziție un card fizic prin care se pot iniția orice fel de plăți și se pot transfera bani oriunde pe glob, mai exact, ai posibilitatea de a efectua plăți online sau la dispozitivele POS ale comercianţilor, de a retrage numerar de la orice bancomat, indiferent de bancă, de monedă și de țară, fără să fie perceput vreun comision. Așadar, atunci când pleci adesea în străinătate, categoric că un asemenea card de debit îți va fi de mare ajutor, deoarece face ca totul să fie mult mai simplu și mai convenabil.

֍ Cardul virtual

Cardul virtual constituie o confirmare a conceptului că banca poate exista pe telefon, oriunde şi oricând, fără sa mai fie nevoie de portofel, căci un astfel de card poate fi folosit exact ca unul fizic, pentru efectuarea de plăți, atât online, cât și cele realizate prin Google sau Apple Pay, dar va apărea doar în aplicația Revolut, nefiind creat fizic vreodată și, implicit, ne putem bucura de privilegiul că nu îl vom putea pierde, pe de altă parte astfel de carduri sunt ideale pentru a face în siguranță plăți online.

֍ Cardul virtual de unică folosință

De fiecare dată când este necesar să faci o plată pe un site care, într-un fel sau altul, pare puțin mai suspect decât celelalte, cea mai potrivită soluție este utilizarea unui așa-numit card

virtual de unică folosință cu ajutorul căruia poți executa acea plată și după aceea acesta se șterge automat, utilizatorul având întodeauna posibilitatea de a-și genera numaidecât oricâte carduri virtuale Revolut de unică folosință are nevoie.

- Funcțiile bancare revoluționare

  - Analiza modului în care cheltui banii - card menit pentru utilizarea de zi cu zi datorită funcțiilor de analiză a cheltuielilor, de estimare a acestora și de stabilire a bugetului personal, arătând când și pe ce sunt cheltuiți banii, împreună cu valoarea sumelor de bani plătite; se poate vedea în funcție de categorie, de comerciant sau de țara în care au fost cheltuiți acei bani.
  - Serviciul Suport Clienți - prin intermediul chat-ului din aplicație un reprezentant oferă în cel mai scurt timp posibil asistență în rezolvarea oricăror probleme întâmpinate
  - Trimiterea și primirea instantanee a banilor către și dinspre persoanele pe care le ai la contacte și care utilizează aplicația, gratuit, indiferent de țară sau monedă sau, de asemenea, top-up-ul cu scopul de a alimenta cardul Revolut
  - Poți solicita plăți prin crearea de link-uri de plată specifice sau coduri QR, care sunt trimise automat, când dorești, celor ce au datorii la tine
  - Sistem de rewards, prin care utilizatorii pot beneficia de numeroase reduceri, grație parteneriatelor pe care Revolut le-a încheiat cu diverse companii din țara noastră
  - Vaults - un fel de cont de economii, unde utilizatorii pot strânge bani pentru diverse obiective personale; una dintre cele mai folosite funcționalități este cea de rotunjire, prin care diferența de bani până la cea mai apropiată valoare rotundă de la fiecare plată cu cardul se poate vira automat într-un cont de economii

- Securitatea Revolut

Compania fintech a făcut eforturi impresionante pentru a se asigura că fondurile clienților sunt cât mai bine protejate. Astfel, pe lângă formele de securitate obișnuite, Revolut oferă și un sistem complet nou în piața bancară, de detectare a potenţialelor tranzacții frauduloase cu cardul. Așadar, dacă se suspectează faptul că cineva dorește să realizeze o plată dintr-o locație

total diferită de a utilizatorului (sau a telefonului său mobil), atunci refuză tranzacția respectivă în mod automat și, totodată, blochează cardul.

Criptomonedele

Totodată, Revolut ne dezvăluie cum este lumea cripto, care de-a lungul timpului a devenit și o competiție pentru bănci și piața valutară Forex. Aceștia au implementat tranzacționarea de criptomonede în ideea de a simplifica procesul de cumpărare, respectiv schimb de criptomonede pentru aproape oricine, aceste lucruri întâmplându-se la unele dintre cele mai bune cursuri de schimb disponibile. Pentru a achiziționa criptomonede trebuie ca utilizatorul să acceseze meniul Criptomonedă din partea superioară a aplicației, aici interfața fiind foarte prietenoasă și intuitivă, și apoi se pot cumpăra instant criptomonede folosind fonduri FIAT (de exemplu, RON). În partea de sus se selectează ce criptomonedă se va cumpăra, iar în partea de jos se trece moneda cu care se va plăti, putându-se cumpăra și fracțiuni ale acesteia. Abordarea face ca spațiul cripto să fie mult mai accesibil și mai atrăgător pentru publicul larg. Pe de altă parte, în urma unui studiu realizat de echipa Revolut, s-a concluzionat faptul că o mare parte dintre utilizatori și-au stabilit ca obiectiv financiar să se informeze mai profund în ceea ce privesc criptomonedele, în 2022.

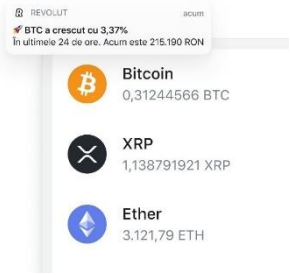



*Figura 6 - Serviciul de criptomonede Revolut (Revolut, 2022)*

În urma studiului efectuat despre aceste trei aplicații, am ajuns la concluzia că un serviciu de mobile banking trebuie să ofere cât mai multe funcționalități, ținta fiind un banking 100% online, să fie cât mai sigur, eficient și rapid, iar pe lângă toate acestea, tot de o importanță deosebită, este faptul că trebuie să aibă o interfață atractivă, prietenoasă, destul de concisă și, în același timp, foarte intuitivă și ușor de folosit pentru orice utilizator, ca totul sa fie mai simplu, mai rapid și mai relaxat.

În ceea ce privește aplicația dezvoltată, mi-am propus să găsesc un echilibru între cele descrise mai sus, astfel încât să pot îmbina eficiența și funcționalitățile de bază ale unei aplicații de banking (cum am văzut în cazul ING) cu extraopțiunile unei aplicații inovative (opțiunile implementate de Revolut) în cel mai util, dar și atractiv mod, toate încapsulate într-o interfață cât mai simplă, modernă și aerisită pentru a oferi o experiență cât mai placută.

# CAPITOLUL 2. TEHNOLOGIILE INFORMATICE UTILIZATE

❖ *Android*

Android[21] este un sistem de operare mobil dezvoltat de Google, bazat pe o versiune modificată a kernel-ului Linux, conceput într-o prima fază pentru a fi utilizat pe dispozitivele mobile cu ecran tactil (denumit și cu anglicismul touchscreen, acesta presupune un tip de ecran sensibil la atingere, interacțiunea realizându-se prin atingerea afișajului dispozitivului cu un creion special, numit și stylus, sau cu degetul), mai exact, smartphone-urile și tabletele, ca mai apoi să se extindă practic peste tot, fiind disponibil și pentru televizoare inteligente (Android TV), ceasuri inteligente (Wear OS) și chiar mașini (Android Auto).

Mediul oficial de dezvoltare integrată (IDE) pentru sistemul de operare în cauză este considerat a fi Android Studio. Se bazează pe software-ul IntelliJ IDEA al JetBrains și reprezintă un instrument complet întrucât oferă un număr mare de posibilități în dezvoltarea de aplicații pentru Android, făcând astfel ca programarea să fie mult mai simplă, mai intuitivă și mai rapidă. Google utilizează multe dintre propriile instrumente și librării cu intenția de a-i ajuta pe dezvoltatori să lucreze, iar toate acestea sunt colectate în Android Software Development Toolkit (SDK). Modulele conțin diferite unelte (Build-Tools, PlatformTools, Debug-Tool, Emulator etc.) care fac posibilă efectuarea editării codului, depanarea, utilizarea instrumentelor de performanță, crearea și implementarea instantanee, dar și multe altele. Cu ajutorul emulatorului, putem testa modul în care funcționează o aplicație și cum este afișată aceasta pe dispozitivele Android (telefoane mobile, tablete etc.). Deși există o listă întreagă de limbaje de programare suportate de Android, Java este considerat limbajul oficial al dezvoltării Android.

❖ *Android vs. iOS*[22]

➢ Actualizări de software: Înfățișează o particularitate prin care utilizatorii OS sunt avantajați, căci actualizările pentru iOS (iPhone Operating System) sunt, în general, disponibile pentru fiecare dintre dispozitivele iOS compatibile. În timp ce pentru dispozitivele Android, nu se oferă actualizări de software la cea mai recentă versiune

[21] (Chen, 2021)
[22] (Android vs. iOS, n.d.)

Android pentru întreg ansamblul de telefoane și tablete din linia de produse, acestea fiind lansate pe rând în funcție de tipul de telefon, respectiv tabletă.

- Confidențialitate: Sistemul IOS are reguli de confidențialitate mai stricte spre deosebire de Android atunci când vine vorba de protejarea informațiilor personale ale utilizatorilor, referindu-ne cu precădere la accesul aplicațiilor instalate asupra acestora.
- Aplicațiile Android au ca și componente cruciale activitățile și fragmentele, întrucât utilizatorul interacționează în mod direct cu ele. O activitate reprezintă ecranul în care este afișată aplicația, fereastra în care aplicația desenează UI-ul, mai precis, interfața aplicației. Fiecare activitate conține fragmente, iar acestea introduc conceptele de modularitate și reutilizabilitate în cadrul activității, permițând divizarea interfeței în bucăți distincte, mai convenabile pentru definirea și gestionarea facilă și a unui singur ecran sau a unei porțiuni a ecranului.
- Aplicațiile iOS se axează pe controlerele de vizualizare. Un astfel de controler gestionează întreaga interfață de utilizare a aplicației sau doar o porțiune din aceasta, dar și interacțiunile dintre acea interfață și datele subiacente. Totodată, facilitează tranzițiile între interfețele cu utilizatorul. Dezvoltarea iOS tinde să fie net mai rapidă, mai facilă și mai eficientă, riscul de erori fiind redus.

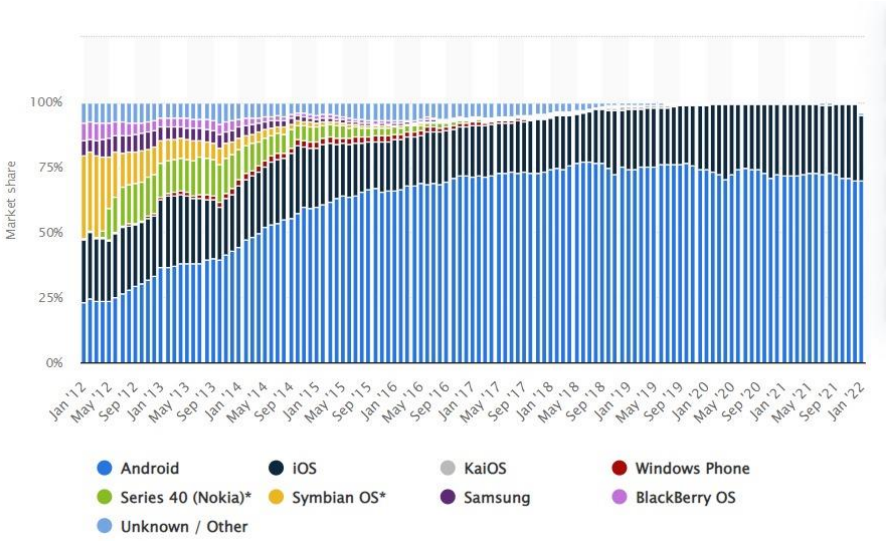



*Figura 7 - Cota de piață a sistemelor de operare mobile la nivel mondial în perioada ianuarie 2012 - ianuarie 2022 (Laricchia, 2022)*

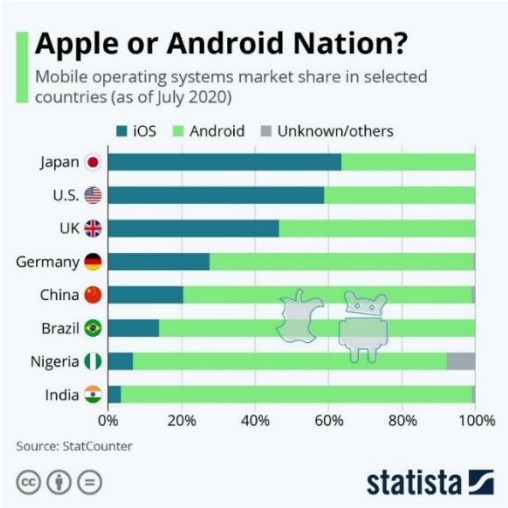



*Figura 8 - Cota de piaţă Android vs. iOS (Buchholz, 2020)*

❖ *Java*

Java[23] reprezintă un limbaj de programare orientat pe obiecte (OOP), puternic tipizat, creat de James Gosling la Sun Microsystems (din 2010 Oracle America, Inc.) la începutul anilor ’90 și lansat în 1995. Totodată, este considerat un limbaj de programare cu scop general, independent de platformă și de sistemul de operare pe care rulează, și putem afirma ferm faptul că a jucat un rol deosebit de important în explozia Internetului, apoi la apariția telefonului inteligent, continuând cu dezvoltarea cloud computing-ului, iar acum își câștigă locul și în industria IoT. Cât mai simplist spus, Java este un program în care se pot produce aplicații. După ce un programator dezvoltă o aplicație Java, aceasta este potrivită pentru a rula pe majoritatea sistemelor de operare, incluzând Windows, Linux, Android și Mac OS, de aici reieșind faptul că este un limbaj versatil, fapt care a adus un aport semnificativ la succesul său.

Limbajul de programare Java se regăseşte în topul limbajelor de programare, la nivel de popularitate, iar locul este bine meritat deoarece are aplicații în multe direcții utile și de uz de masă, acesta fiind folosit într-o arie largă de domenii: de la aplicații Android, la dezvoltare de backend website, gestionare de baze de date, iar de curând, și IoT (Internet of Things).

Pe de altă parte, cum spuneam la început, Java este anul acesta la cea de-a 27-a aniversare, așa că este cert faptul că au fost deja adresate o sumedenie de întrebări, neclarități, dileme, urmând ca mai apoi să fie discutate și disecate majoritatea erorilor. Acest fapt însemnând

[23] (Ghidul începătorilor în Java: Ce este și de ce să înveți acest limbaj de programare, 2019)

că practic, în prezent, cu certitudine putem găsi răspunsuri și soluții demult validate și verificate pe diverse site-uri online, de asemenea, sunt mulți cei care utilizează și dezvoltă programe în Java, așa că acest lucru duce la o comunitate numeroasă de utilizatori și programatori gata să sară în ajutor oricând te confrunți cu o problemă.

- *Python*

Python[24] reprezintă un limbaj de programare interpretat, orientat pe obiecte, cu o sintaxă accesibil de utilizat, care folosește structuri de date complexe și poate fi utilizat împreună cu o multitudine de pachete și librării, ceea ce stimulează posibilitatea reutilizării codului și dezvoltarea diverselor soluții. În Python, programul nu trebuie să fie compilat înainte de a fi executat, ceea ce face ca procesul de programare și testare să fie foarte facil, iar lipsa de artificialitate a codului îl face să devină foarte lizibil, eficient și ușor de întreținut sau dezvoltat pe mai departe.

- *Firestore*

Firebase[25] este o platformă de dezvoltare pentru aplicații mobile și web pusă la dispoziție de Google, care oferă dezvoltatorilor un set de tool-uri ce acoperă o gamă largă de servicii cu scopul de a le permite să se focuseze pe dezvoltarea aplicației propriu-zise, îndeosebi pe eficientizarea experienței utilizatorilor (UX), fiind vitală, și nu pe API-uri de autentificare, de stocare sau de notificare (Firebase Authentication, Firebase Cloud Firestore, respectiv Firebase Cloud Messaging).

Cloud Firestore este o bază de date de tip NoSQL, adică non-relațională, găzduită în Cloud, prin care atât aplicațiile Android și iOS, cât și cele web, pot fi accesate direct prin SDK-urile native. Astfel spus, Firestore funcționează împreună cu kiturile de dezvoltare software (denumite și SDK) și cu un set complet de reguli de securitate, astfel încât nu este nevoie de crearea unui server auxiliar, de sine stătător. Folosind acest serviciu se pot sincroniza automat informațiile vizate între toate dispozitivele grație notificărilor livrate atunci când informația este modificată, facilitând astfel experiența aplicațiilor în timp real.

[24] (What is Python? Executive Summary, n.d.)
[25] (Cloud Firestore, n.d.)

Modelul de date NoSQL al Cloud Firestore presupune stocarea informațiilor în documente ce conțin câmpuri mapate la valori. Aceste documente sunt stocate ulterior în colecții, care acționează precum niște containere pentru documente, și cu ajutorul acestor containere se poate întreprinde organizarea și interogarea datelor. Un document este apt de a accepta tipuri de date diferite, de la tipuri simple la obiecte complexe, imbricate, creându-se subcolecții în interiorul documentelor sau structuri ierarhice de date ce se extind cu aceeași măsură cu care baza de date se mărește.

Avantajul utilizării unei baze de date non-relaționale este dat de faptul că nu necesită o structură specifică, ci structura acesteia poate fi creată în funcție de nevoile dezvoltatorilor. Datele salvate în documente funcționează pe principiul de cheie-valoare. Abordarea aceasta face ca dezvoltarea structurii ierarhice din baza de date să fie mult mai facilă, permițând stocarea informațiilor de volum, dar și complexitate mult mai mare. Pentru a păstra actualizate datele din aplicație, sunt adăugați observatori responsabili de preluarea datelor în timp real, fără a returna întreaga bază de date în momentul efectuării unei actualizări, ci doar informațiile în cauză vor fi modificate, respectiv afișate, în funcție de acțiunea care a pornit procesul.

- *Bazele de date nerelaționale (în speță Firestore) în comparație cu cele relaționale (precum MySQL)*[26] [27]

  - Firestore oferă scalabilitate orizontală, ceea ce înseamnă că partiționarea datelor se face pe mai multe noduri care să le poată procesa, în loc ca toate informațiile să fie într-un singur loc, în timp ce MySQL se scalează pe verticală, adică scalarea se face prin adăugarea de dispozitive fizice cum ar fi adiția de mai multă putere de procesare sau memorie, acest lucru întâmplându-se tocmai pentru că toate informațiile sunt stocate într-un singur loc.
  - Datele în Firestore sunt stocate în documente, similare cu formatul de fișier JSON (JavaScript Object Notation), sub formă de perechi cheie-valoare, pe când în cadrul MySQL datele sunt păstrate în tabele. Firestore utilizează scheme dinamice cu scopul de a facilita datele nestructurate care vin sub forma unui document, pe când MySQL se bazează pe scheme predefinite.

[26] (Dearmer, 2020)
[27] (Choose a Database: Cloud Firestore or Realtime Database, n.d.)

# CAPITOLUL 3. ANALIZA ȘI PROIECTAREA SISTEMULUI INFORMATIC

## 3.1. Specificarea cerinţelor sistemului informatic

### 3.1.1. Cerințe funcționale

Prin acest sistem informatic s-a dorit gândirea și implementarea unei soluții informatice perfect modelate pe specificul activităţilor de banking online, simplu de folosit datorită unui design intuitiv, aerisit și frumos, prin care utilizatorii își pot accesa conturile deținute și pot efectua o serie de operațiuni bancare la distanță mult mai eficient, ca să aibă mai mult timp să se bucure de acele lucruri care contează. Toate datele sistemului informatic sunt păstrate într-o bază de date la distanță și pot fi interogate sau modificate în urma oricărei conectări a utilizatorului în acesta.

Utilizatorul trebuie să poată desfășura operațiunile importante ale unei aplicații mobile de banking, cum ar fi vizualizarea tuturor detaliilor despre contul bancar personal și a informațiilor de securitate ale cardului, deschiderea și închiderea unui depozit de economii, inițierea de plăți de facturi de utilități sau transferuri bancare către diverși beneficiari, accesarea istoricului de tranzacții, filtrarea acestora, dar și opțiunea de a genera unui extras de cont în format electronic.

Pe lângă toate aceste funcționalități de bază, sistemul informatic trebuie să ofere o pagină dedicată în mod special generării și consultării de rapoarte grafice bazate pe istoricul de cheltuieli și, totodată, să-i permită utilizatorului să acceseze și o prognoză cu privire la suma de bani care poate fi cheltuită luna viitoare, aceasta relizându-se prin analiza informațiilor legate de obiceiurile financiare ale utilizatorului.

Datele de intrare sunt reprezentate de informațiile introduse de utilizator pentru conectarea în aplicație, pentru deschiderea depozitului de economii, precum și pentru inițierea

unui transfer bancar, iar datele de ieșire sunt informațiile preluate din baza de date, alături de toate rapoartele generate pe baza lor, inclusiv predicția cheltuielilor.

### 3.1.2. Cerințe non-funcționale

În ceea ce privesc cerințele non-funcționale ale sistemului informațional, acestea se referă îndeosebi la mecanismele de securitate și la confidențialitatea informațiilor, fiind vorba mai ales despre o aplicație din domeniul financiar-bancar. La fel de însemnate sunt și aspectele legate de asigurarea portabilității și a interconectării, dar și eficiența și acuratețea.

Confidențialitatea datelor este cu certitudine o cerință non-funcțională fundamentală a acestui sistem, întrucât orice aplicație bancară trebuie să asigure utilizatorilor săi cel mai ridicat nivel de protecție a datelor personale și a informațiilor de securitate aferente conturilor active la banca respectivă. Pentru a satisface această cerință sunt create conturi individuale unice pentru fiecare dintre utilizatori, accesibile doar în urma parcurgerii unor pași de securitate în procesul de autentificare (conectarea prin cod de siguranţă primit prin mesaje SMS).

Scopul unei astfel de soluţii informatice este de a putea oferi utilizatorilor posibilitatea de a accesa și desfășura operațiuni bancare de la distanță, indiferent de ora și locul în care se află. În cadrul sistemului, cerințele de portabilitate și interconectare sunt satisfăcute atât prin stocarea tuturor informațiilor într-o bază de date externă, în cloud, ci nu într-una locală, cât și prin comunicarea continuă dintre dispozitivul mobil, interfețele aplicației în Android și serverul în care sunt localizate datele.

Eficiența și acuratețea sunt indispensabile atunci când se dorește asigurarea nivelului de satisfacție ridicat al utilizatorului. Operațiile trebuie să se desfășoare într-un mod rapid și corect, mai ales în cazul unui transfer bancar sau când este consultată predicția cheltuielilor, care trebuie să fie cât mai precisă posibil. Pentru a marca cerința legată de eficiență, operațiile de interogare și modificare a informațiilor din baza de date sunt efectuate pe fire de execuție secundare cu scopul de a nu încetini aplicația prin încărcarea firului de execuție principal, iar în ceea ce privește cerința acurateței, unde ne referim în special la predicție, este utilizat un algoritm de tip machine learning scris în Python.

### 3.1.3. Descrierea sistemului și diagrame pentru cazuri de utilizare

În cele ce urmează voi prezenta atât diagrama generală de cazuri de utilizare, cât și diagramele e particulare care oferă o perspectivă mai clară și mai detaliată asupra activităților, alături de o descriere textuală.

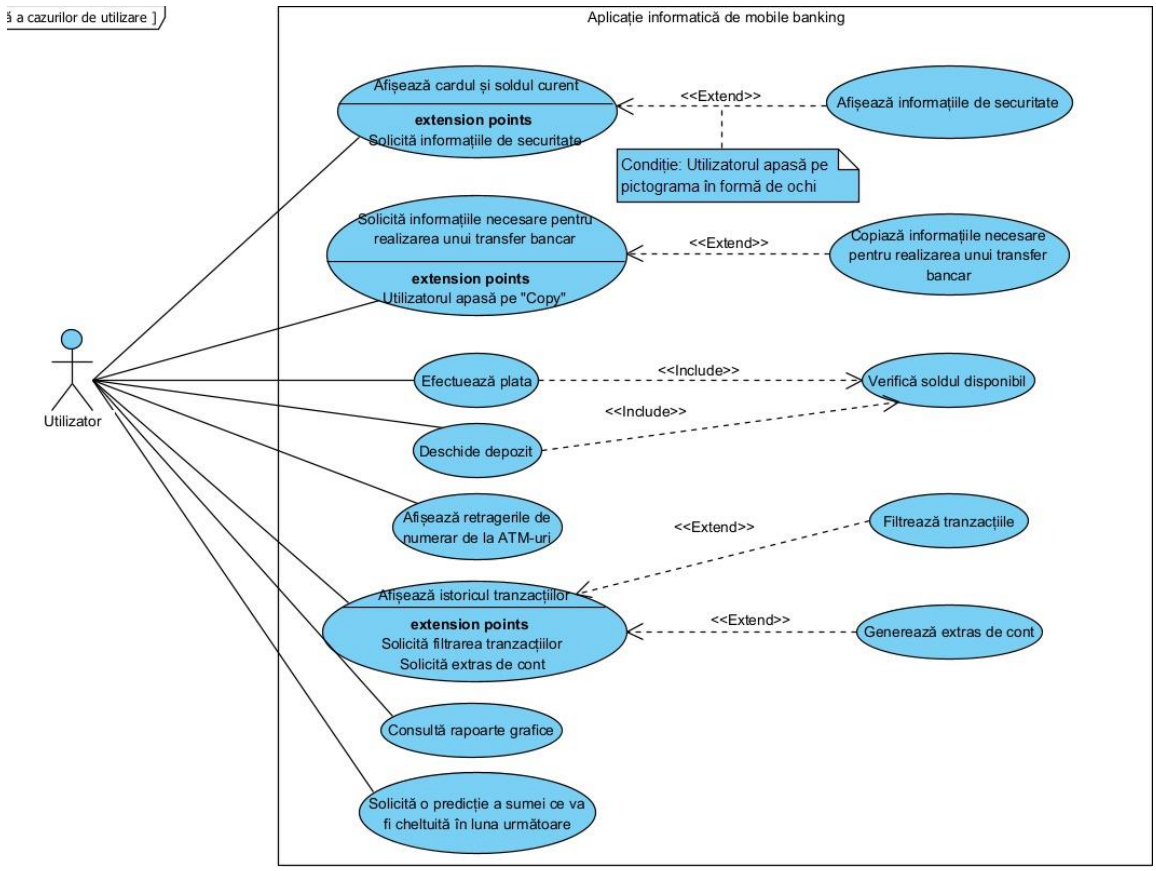



*Figura 9 - Diagrama generală a cazurilor de utilizare*

În figura de mai sus este ilustrată diagrama generală a cazurilor de utilizare prin care se descrie în ansamblu modul de funcționare al aplicației mobile. Este evidențiat actorul principal, utilizatorul, care interacționează cu aplicația și principalele acțiuni pe care acesta le poate întreprinde. În continuare este reliefat cazul de utilizare corespunzător pentru afișarea istoricului de tranzacții.

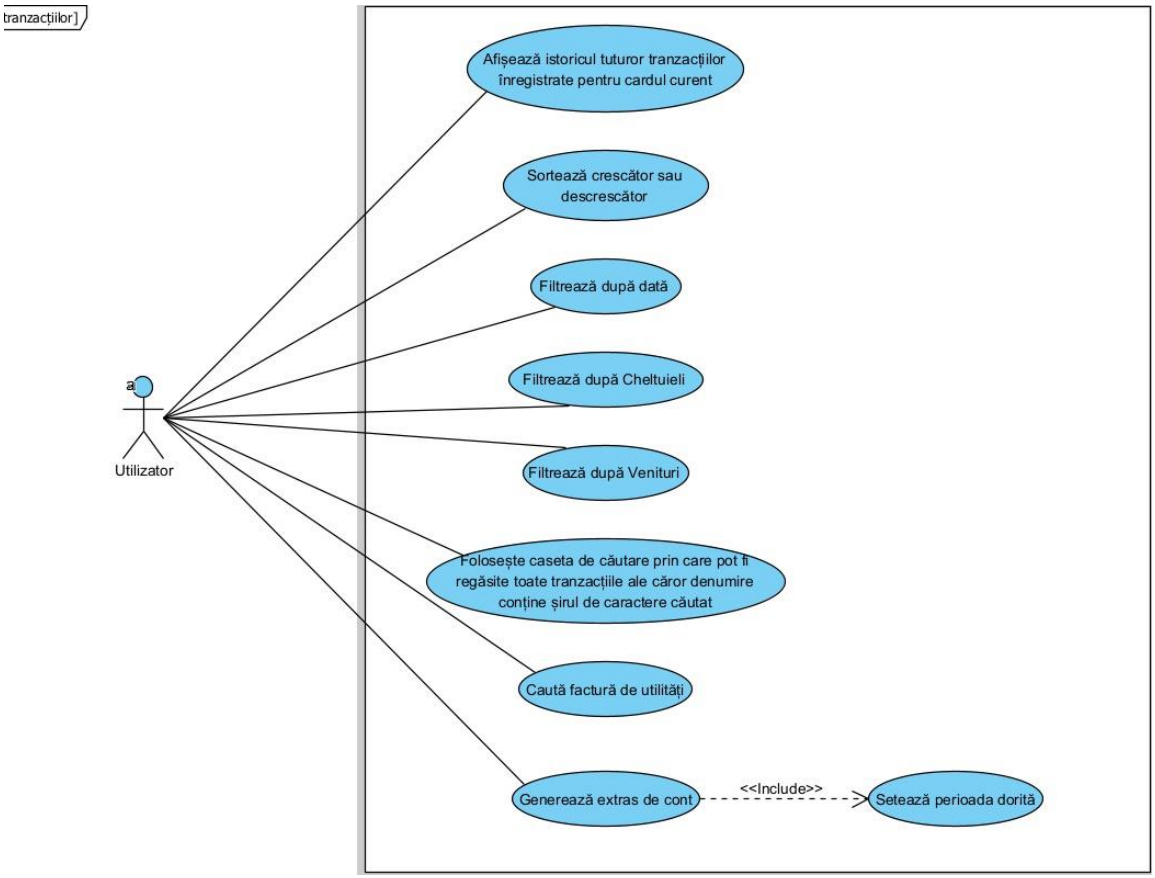



*Figura 10 - Diagrama detaliată a cazului de utilizare „Afișează istoricul tranzacțiilor"*

Istoricul de tranzacții (Figura 10) funcționează în baza interogărilor bazei de date la distanță. Utilizatorul, odată conectat la aplicație, va putea solicita întregul istoric. În momentul cu pricina, aplicația realizează o operație prin care cere toate informațiile din baza de date pe care mai apoi le salvează într-o listă locală cu scopul de a putea fi accesate mai facil într-o împrejurare ulterioară când va fi din nou nevoie de ele. Utilizatorul poate consulta detaliile fiecărei tranzacții, poate sorta și filtra informațiile primite după multiple criterii specifice și, de asemenea, poate solicita un extras de cont pentru o perioadă specifică.

**Descrierea textuală a cazului de utilizare „Afișează istoricul tranzacțiilor”**

| *Element al cazului de utilizare* | *Descriere* |
|---|---|
| Cod | CU01 |
| Stare | Schiță |
| Scop | Accesarea istoricului de tranzacții, filtrarea acestora și generarea unui extras de cont |
| Nume | Afișează istoricul tranzacțiilor |
| Actor principal | Utilizatorul aplicației |
| Descriere | Secțiunea Tranzacții afișează o lista completă a tranzacțiilor din istoricul salvat în baza de date cu posibilitatea de a filtra și sorta informațiile primite după multiple criterii specifice. Tot de aici utilizatorul poate solicita un extras de cont pentru o perioadă arbitrară. |
| Precondiții | Utilizatorul dispune de un smartphone sau o tabletă Android cu conexiune la Internet, are descărcată aplicația dezvoltată, o deschide și se autentifică corespunzător în sistem. Ulterior, acesta accesează secțiunea Tranzacții. |
| Postcondiții | Utilizatorul are o imagine clară asupra evoluției soldului din contul curent și a putut obține într-un mod facil si rapid un extras de cont dacă a solicitat generarea acestuia. |
| Declanșator | Utilizatorul dorește să vizualizeze lista plăților/ tranzacțiilor efectuate. |
| Fluxul de bază | 1. Utilizatorul consultă lista integrală a tranzacțiilor efectuate.<br>2. Utilizatorul filtrează tranzacțiile după mai multe criterii dorite.<br>[Curs alternativ A: Utilizatorul nu folosește opțiunile de |

| | |
|---|---|
| | sortare și/sau filtrare.]<br><br>3. Utilizatorul dorește un extras de cont pe o perioadă aleasă de timp.<br><br>[Curs alternativ B: Utilizatorul nu solicită extras de cont.] |
| Fluxuri alternative | A: Se încheie scenariul.<br><br>B: Se încheie scenariul. |
| Relații | - |
| Frecvența utilizării | Frecvent, cel puțin o dată pe săptămână |
| Reguli ale afacerii | Utilizatorul nu poate obține un extras de cont dacă în istoric nu se găsesc tranzacții efectuate în perioada specificată. |

Tabel 1 - Descrierea textuală a cazului de utilizare „Afișează istoricul tranzacțiilor”

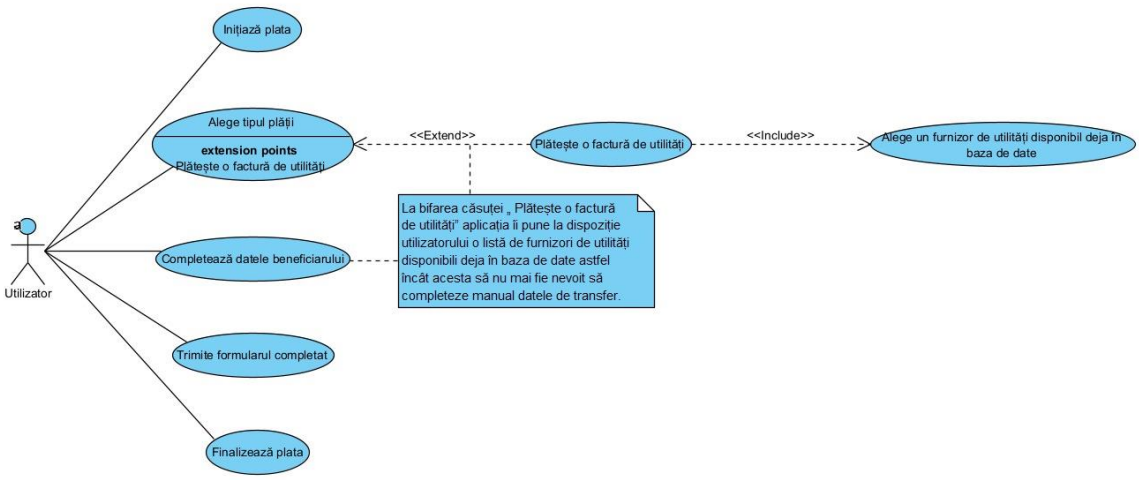



*Figura 11 - Diagrama detaliată a cazului de utilizare „Efectuează plata”*

### ֍ Descrierea textuală a cazului de utilizare „Efectuează plata”

| *Element al cazului de utilizare* | *Descriere* |
|---|---|
| Cod | CU02 |
| Stare | Schiță |
| Scop | Efectuarea de plăti de utilități sau transferuri bancare către diverși beneficiari |
| Nume | Efectuează plata |
| Actor principal | Utilizatorul aplicației |
| Descriere | Presupune inițierea unei plăți rapide către diverși beneficiari, persoane fizice sau juridice, spre exemplu, oameni dragi, instituții, magazine, furnizori de servicii și utilități, etc. |
| Precondiții | Utilizatorul dispune de un smartphone/tabletă Android cu conexiune la Internet, are descărcată aplicația dezvoltată, o deschide și se autentifică corespunzător în sistem. Ulterior, accesează secțiunea Plăți. |
| Postcondiții | Utilizatorul realizează un transfer bancar în doar câțiva paşi. |
| Declanșator | Utilizatorul dorește să trimită bani către un cont IBAN. |
| Fluxul de bază | 1. Utilizatorul completează datele beneficiarului.<br>[Curs alternativ A: Utilizatorul bifează căsuța "Plătește o factură de utilități".]<br>2. Utilizatorul completează detaliile pentru descrierea plății.<br>3. Utilizatorul completează suma de plată.<br>4. Se verifică dacă soldul contului este suficient pentru a efectua plata. |

| | |
|---|---|
| | [Curs alternativ B: Fonduri insuficiente!] |
| Fluxuri alternative | A: 1. I se pune la dispoziţie utilizatorului o listă de furnizori de utilități disponibili deja în baza de date din care alege unul.<br>2. Datele de transfer ale acelui furnizor ales vor fi completate automat.<br>3. Se revine la pasul 3 din fluxul de bază.<br>B: 1. Apare mesajul de eroare: "Tranzacție cu fonduri insuficiente pentru procesare".<br>2. Se șterge suma introdusă inițial și se așteaptă introducerea altei sume de plată.<br>3. Se revine la pasul 4 din fluxul de bază. |
| Relații | - |
| Frecvența utilizării | Frecvent, cel puțin o dată pe săptămână |
| Reguli ale afacerii | - |

Tabel 2 - Descrierea textuală a cazului de utilizare „Efectuează plata”

## 3.2. Analiza sistemului existent

### 3.2.1. Diagrama de activitate

Scopul diagramei de activitate este de a modela aspectele care țin de procesarea procedurală, ajutând la reprezentarea vizuală a secvențelor de acțiuni prin care se doreşte dobândirea unui rezultat. În cadrul acestora, se descrie fluxul de lucru dintr-un punct de plecare până într-un punct de terminare, detaliind căile de decizie care se pot ivi într-o activitate.

În vederea analizării sistemului s-au creat următoarele diagrame de activitate ce descriu principalele acțiuni pe care utilizatorul le poate efectua în cadrul aplicației mobile:

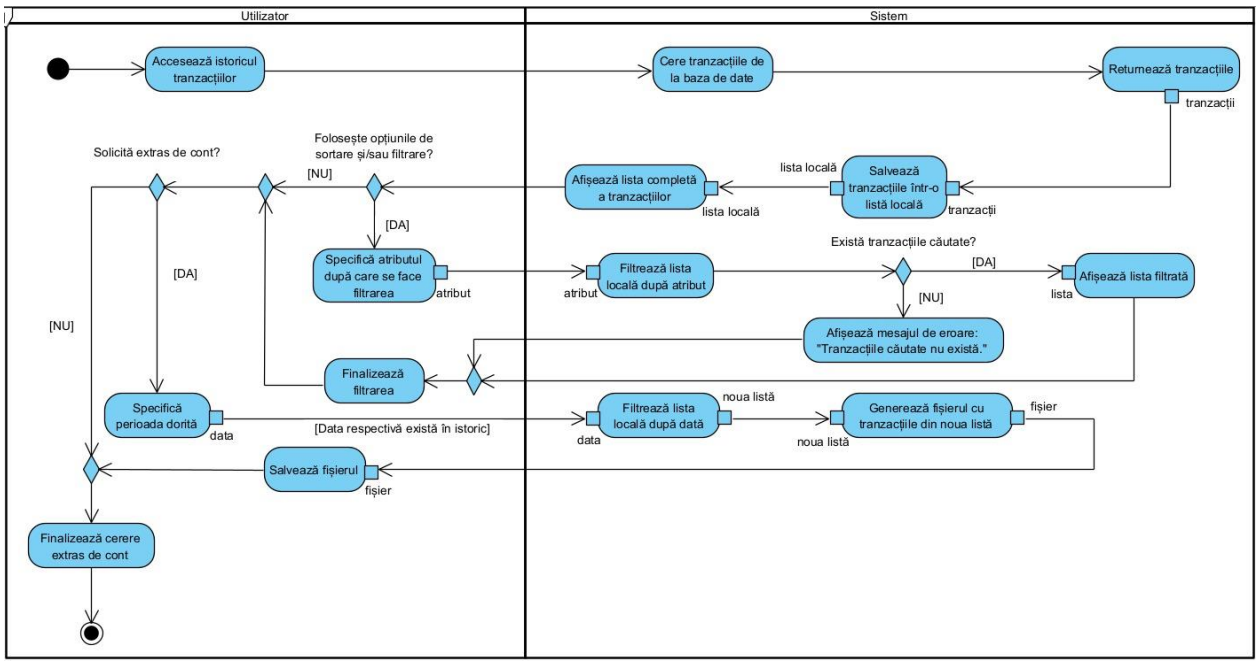



*Figura 12 - Diagrama de activitate pentru operațiile efectuate asupra istoricului de tranzacții*

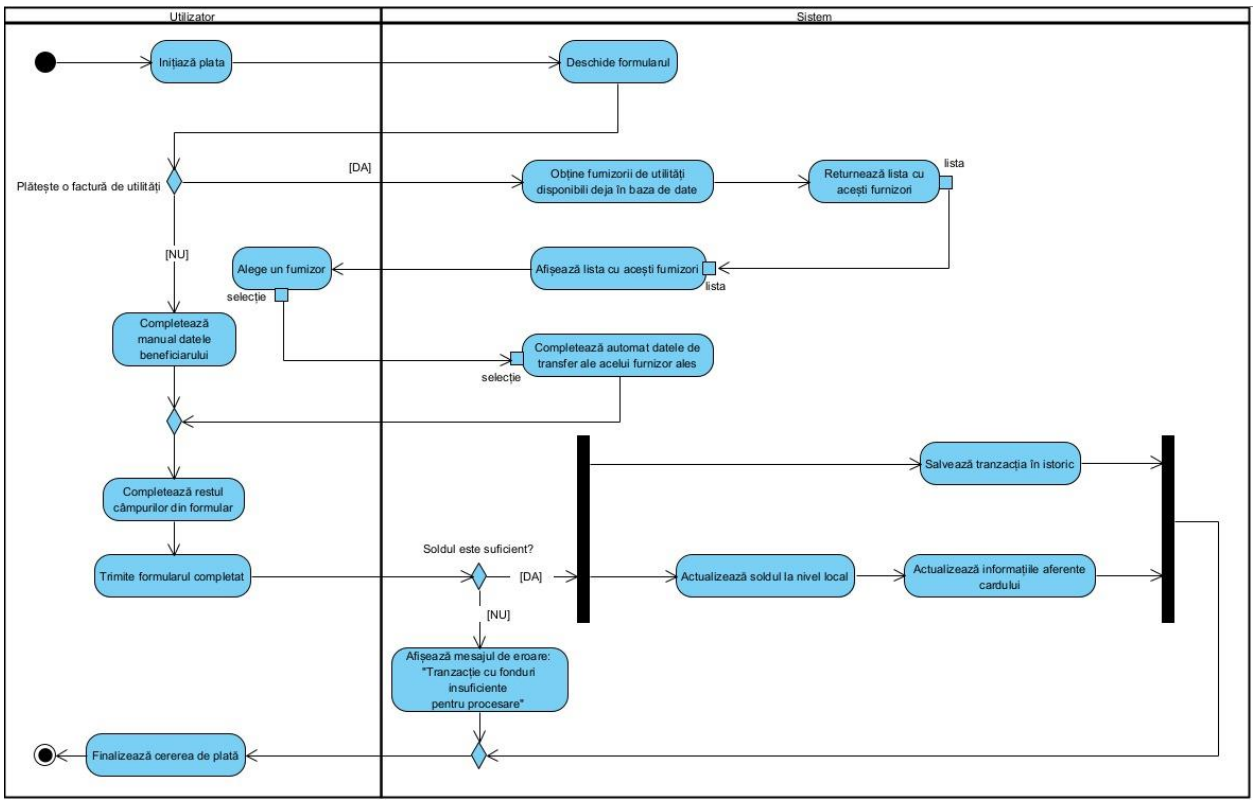



*Figura 5 - Diagrama de activitate pentru operația de efectuare a unei plăți*

O funcție foarte însemnată, de asemenea, într-un sistem informatic destinat operațiunilor bancare prin intermediul internetului este cea prin care se fac plăți și transferuri bancare. Clientul inițiază modulul de plăți, apoi aplicația Android afișează formularul care trebuie completat cu detaliile legate de acea plata. În funcție de tipul de transfer dorit, clientul este pus în situaţia de a alege să completeze singur toate informațiile necesare sau să solicite sistemului o listă de furnizori de utilități din care îl poate alege pe cel de care are nevoie, fiind o cale mai rapidă de completare a formularului. Lista de furnizori disponibili este preluată din baza de date. După trimiterea formularului completat, aplicația verifică dacă datele sunt corecte și, cu precădere, dacă soldul disponibil este suficient pentru acoperirea sumei ca transferul să fie finalizat cu succes. Dupa ce totul este în regulă, plata este finalizată și soldul este actualizat atât în cadrul local, cât și în baza de date de pe server. În caz contrar, utilizatorul primește mesajul de eroare „Tranzacție cu fonduri insuficiente pentru procesare".

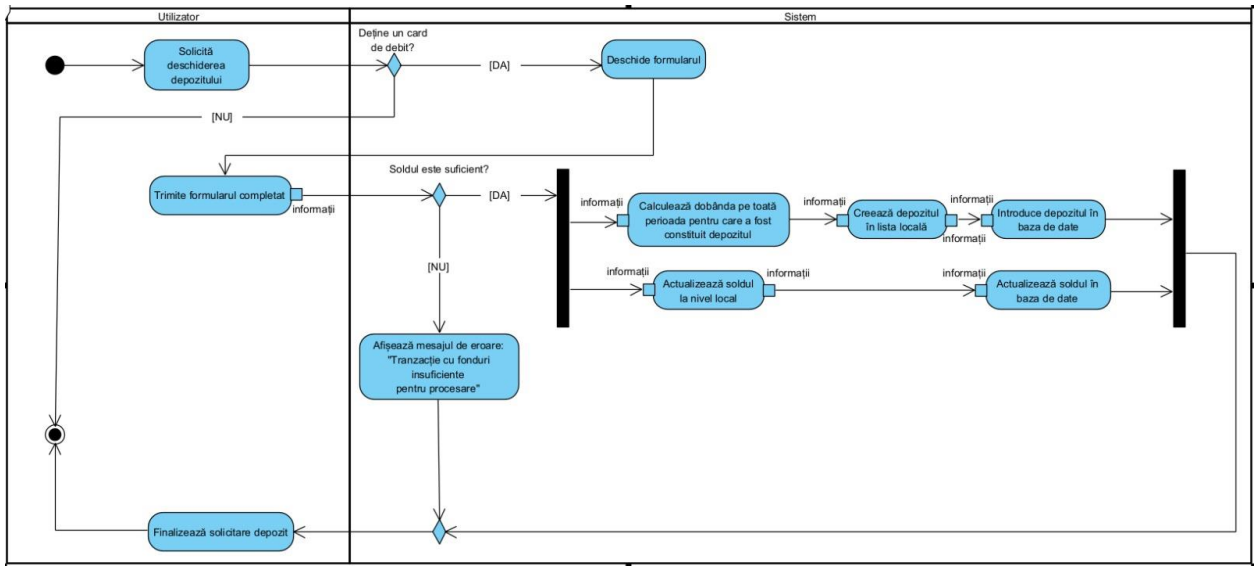



*Figura 14 - Diagrama de activitate pentru operația de introducere a unui depozit de economii*

### 3.2.2. Diagrama de clasă

Diagrama de clasă constituie un tip de diagramă utilizată în scopul descrierii structurii statice, mai exact pentru a implementa cât mai eficient toate funcționalitățile descrise în diagramele de mai sus, adică a claselor existente în cadrul sistemului informatic. Aplicația are, după cum se poate observa și în figura de mai jos, 9 clase, pe care le voi expune în mod amănunțit în Diagrama de clase detaliată.

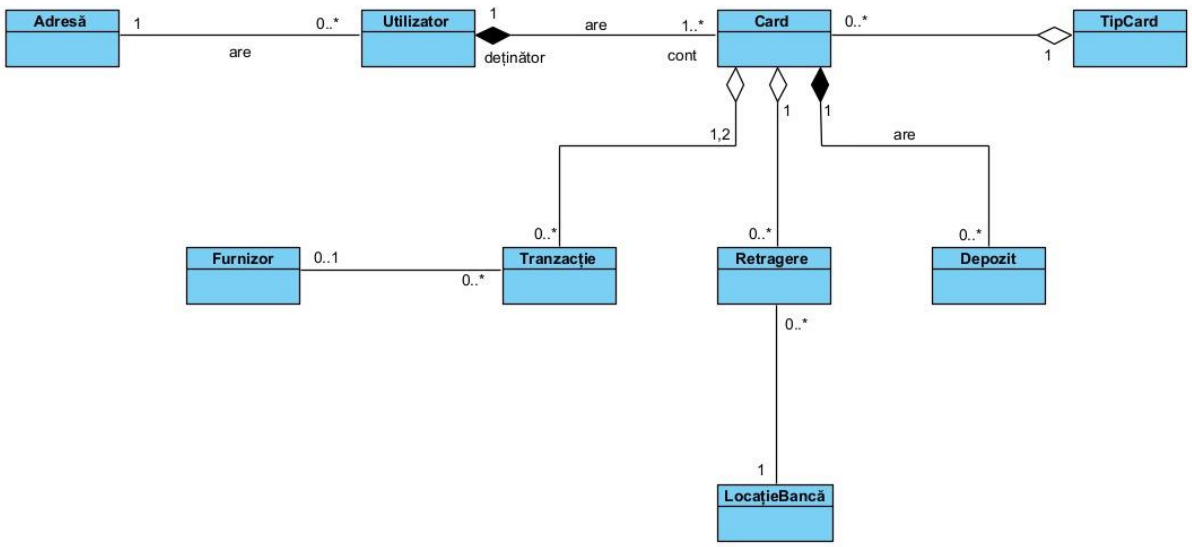



*Figura 15 - Diagrama de clase simplificată*

### 3.2.3. Diagrama de stare

Diagramele de stare sunt folosite pentru a contura mai bine fluxul de lucru, mai precis indică toate stările posibile prin care poate trece un obiect si felul în care ajunge dintr-o stare inițială într-o stare finală.

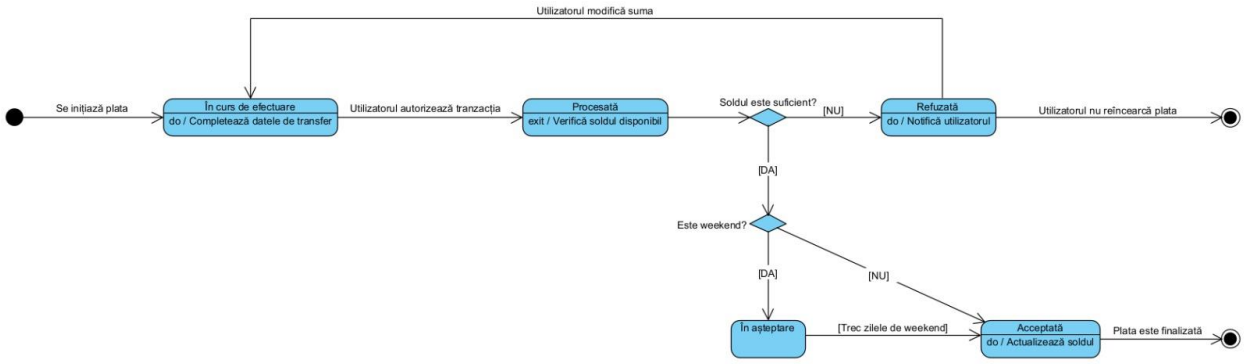



*Figura 16 - Diagrama de stare pentru efectuarea unei plăți*

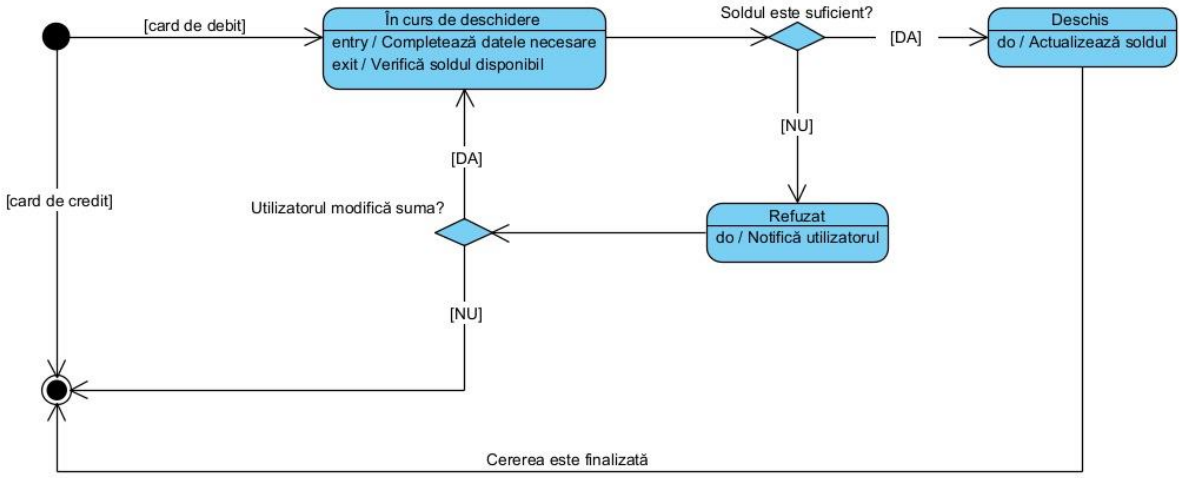



*Figura 6 - Diagrama de stare pentru deschiderea unui depozit de economii*

După cum poate fi observat în diagrama din Figura 17, atunci când un utilizator vrea să deschidă un nou depozit de economii, în prealabil, trebuie să îndeplinească condiția deținerii unui card de debit, având în vedere faptul că nu pot fi deschise depozite cu un card de credit. În cazul favorabil, obiectul de tip Depozit trece în starea „în curs de procesare” în timp ce utilizatorul începe completarea datelor necesare. Prin apăsarea butonului de trimitere a formularului aplicația verifică soldul curent și în funcție de acesta depozitul poate intra în starea de „deschis”, stare în care mai apoi se actualizează automat și soldul cardului, sau în starea „refuzat” în care clientului

îi este adus la cunoștință faptul că soldul este insuficient pentru acest demers și este întrebat dacă dorește modificarea sumei introduse.

### 3.2.4. Diagrama de secvență

Diagramele de secvență fac parte din categoria diagramelor de interacțiune pentru că acestea evidențiază cel mai bine modul în care componentele unei aplicații interacționează și comunică între ele în timpul desfășurării unei operațiuni. În cadrul acestor diagrame sunt conturate atât modul, cât și ordinea în care mesajele sunt trimise dintr-o componentă în altă.

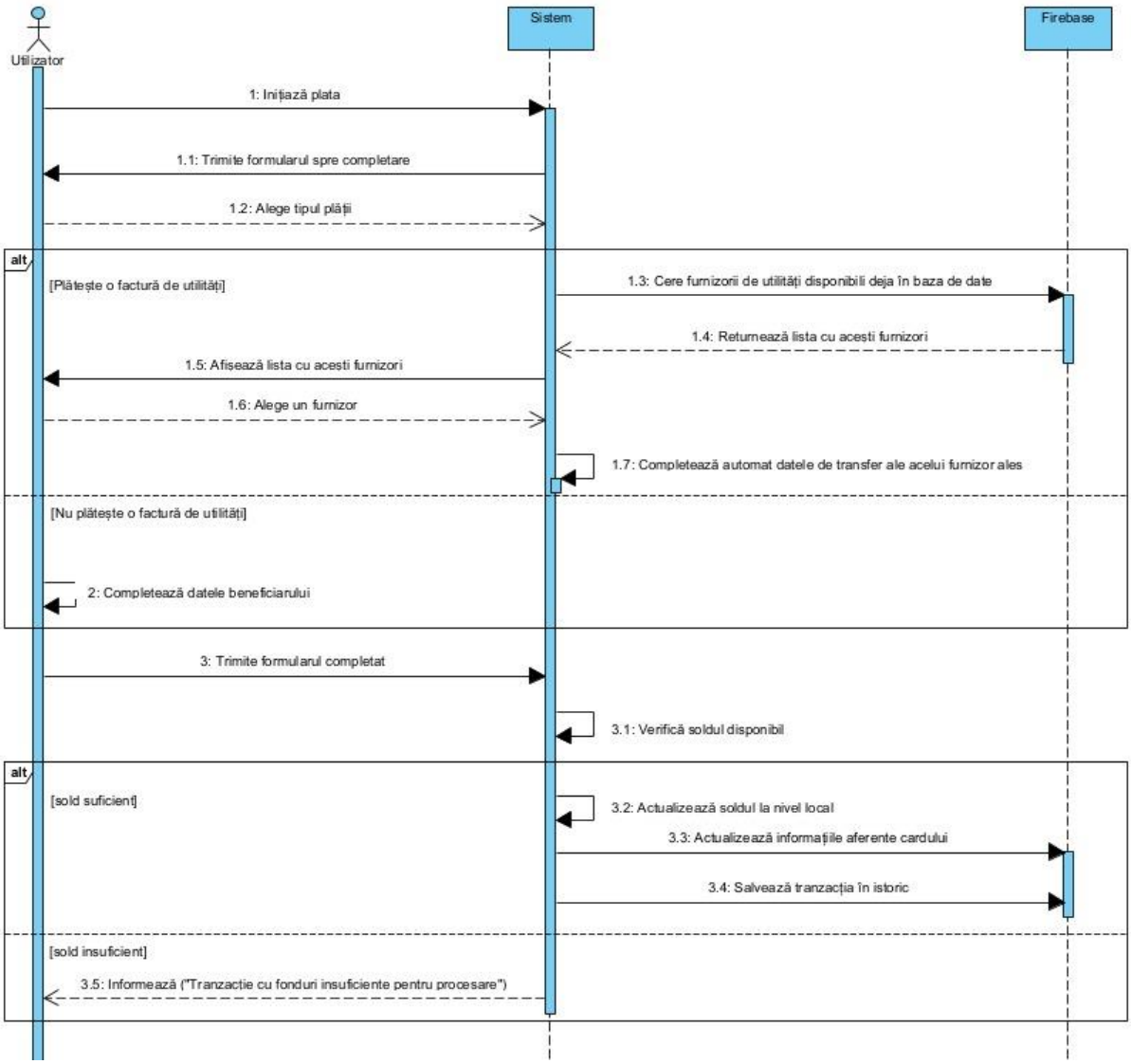



*Figura 18 - Diagrama de secvență pentru descrierea scenariului „Efectuează plata”*

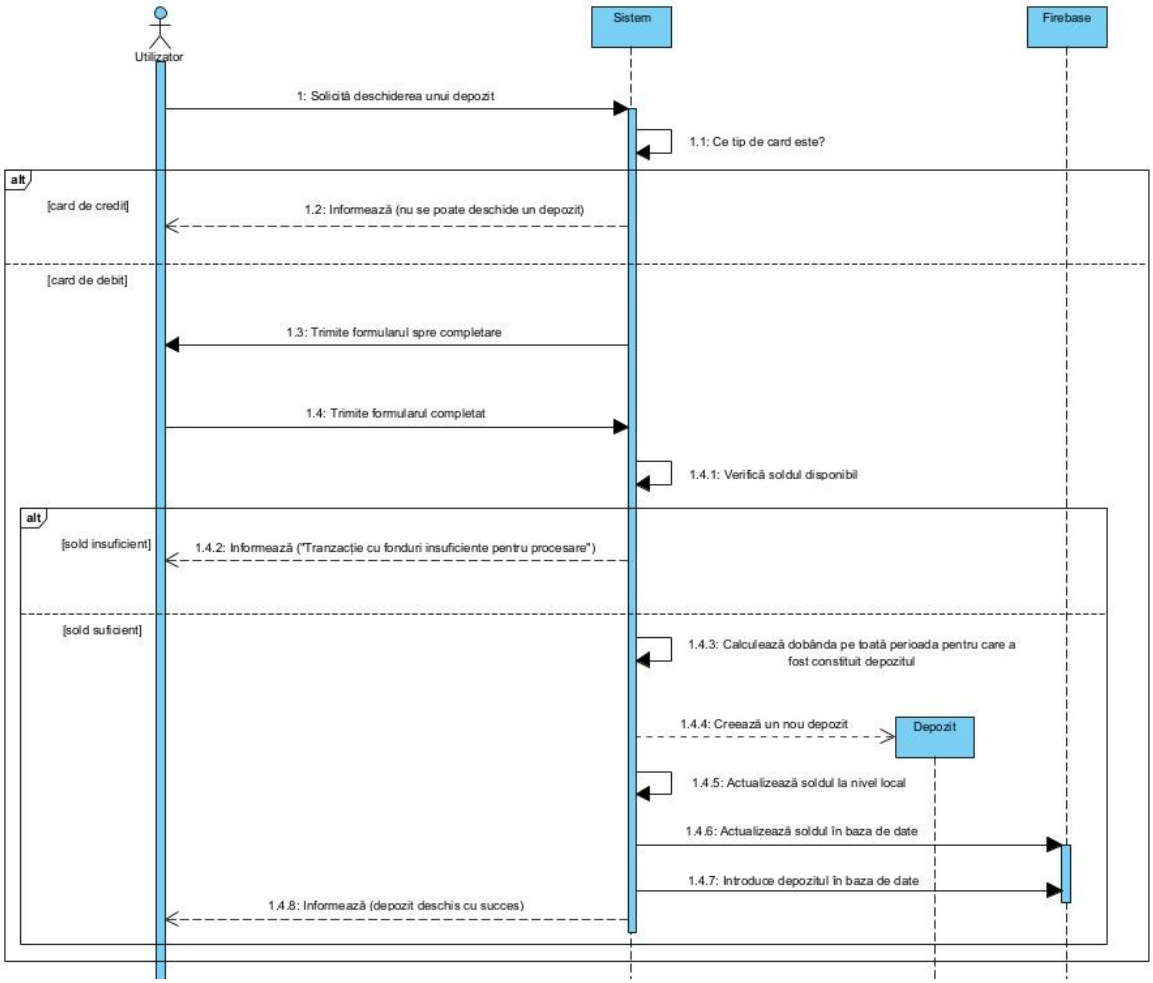



*Figura 19 - Diagrama de secvență pentru descrierea scenariului „Deschide depozit”*

Ținând în continuare cont de procesul de introducere a unui depozit, se poate observa în Figura faptul că utilizatorul este cel care inițiază primul act de solicitare a unui depozit, moment în care aplicația verifică dacă deține un card de debit sau unul de credit. În cazul în care este card de credit informează clientul că nu este posibilă o acțiune în realizarea unui depozit de economii, iar în caz contrar îi trimite utilizatorului formularul care trebuie completat. Acesta completează datele necesare și i-l trimite înapoi aplicației ca să efectueze verificările necesare. Dacă soldul este insuficient, aplicația notifică utilizatorul printr-un mesaj corespunzător. Altfel, sistemul începe procesul de creare și deschidere a depozitului, urmând imediat după aceea să actualizeze soldul curent atât în cadrul local, cât și în baza de date. Noul depozit de economii deschis este adăugat pe server astfel încât toate informațiile să fie actualizate simultan în ambele medii, să nu existe nepotriviri care ar putea conduce ulterior la erori.

## 3.3. Proiectarea sistemului

### 3.3.1. Diagrama de clase detaliată

Diagrama de clase detaliată expune clasele, alături de atributele și metodele aferente acestora. Cu alte cuvinte, această diagramă detaliază descrierea și comportamentul claselor care intră în componența sistemului informatic pentru a facilita înțelegerea funcționalității.

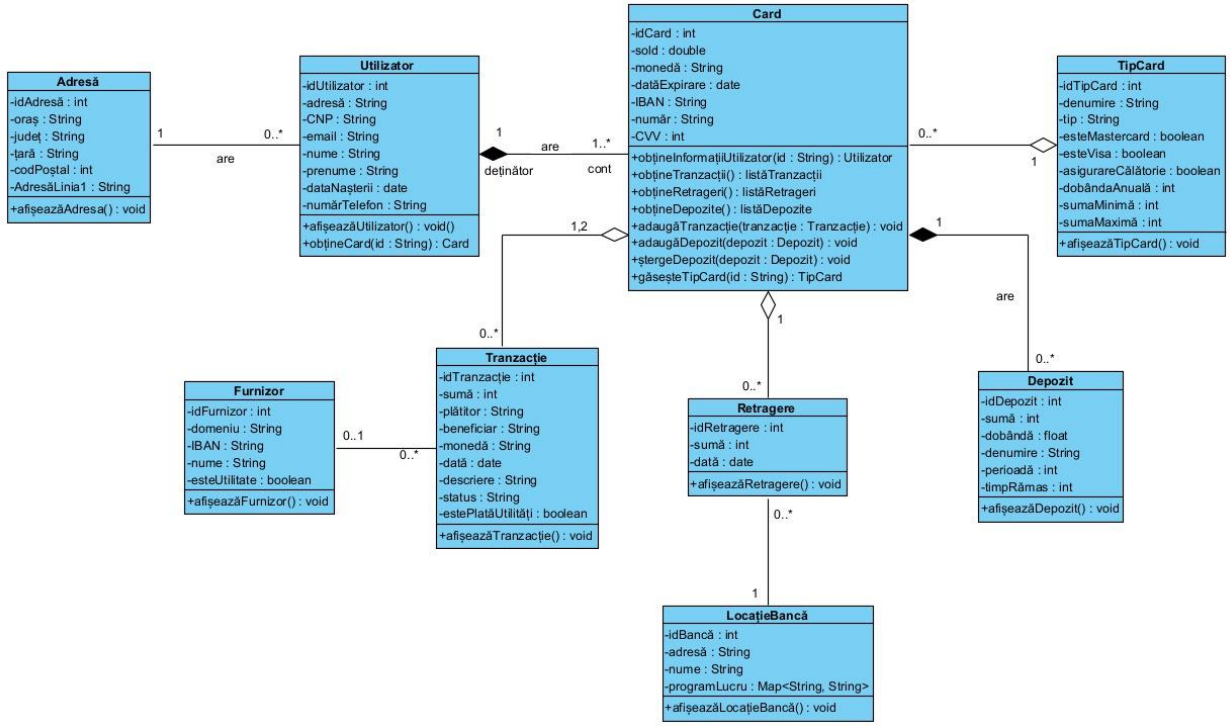



*Figura 20 - Diagrama de clase detaliată*

- ***Utilizator*** - această clasă încapsulează conceptul de utilizator al aplicației, așadar prin obiectele acestei clase se oferă acces la informațiile despre utilizator, acel deținător al unui cont bancar (cont curent) care folosește aplicația de mobile banking

- ***TipCard*** - stochează informații referitoare la tipurile de carduri puse la dispoziție de către bancă utilizatorilor

- ***Card*** - cuprinde toate cardurile utilizatorilor împreună cu informațiile reprezentative pentru acel cont, informațiile de securitate, soldul curent, istoricul tranzacțiilor și retragerile

- ***Depozit*** - informații despre depozitele deschise prin Mobile Banking

- ***Tranzacție*** - informații despre istoricul de tranzacții pe cont

- ***Retragere*** - informații despre toate retragerile de la un bancomat pentru un anumit cont

- ***LocațieBancă*** - informații despre locațiile unităților băncii și ATM-urilor

- ***Furnizor*** - informații despre furnizorii de utilități utilizaţi în procesul de efectuare a unei plăți a facturii

- ***Adresă*** - informații despre toate adresele stocate în baza de date

### 3.3.2. Proiectarea bazei de date

În cadrul acestui subcapitol se săvârșește modelarea cerințelor, pornindu-se de la rezultatele analizei cerințelor sistemului informatic, etapa presupunând:

⇨ **Identificarea entităților***:*

- Utilizator, Card, TipCard, Depozit, Retragere, Tranzacție, Furnizor, LocațieBancă, Adresă

⇨ **Identificarea relațiilor dintre entități:**

- Un utilizator poate avea o singură adresă, dar acea adresă poate să aparțină mai multor utilizatori.

- Un utilizator poate avea mai multe carduri, dar un card poate să aparțină doar unui singur utilizator.

- Un card poate avea un singur tip de card, dar un tip de card poate să aparțină mai multor carduri.

- Un card poate avea 0 sau chiar mai multe depozite de economii, dar un depozit poate să aparțină unui singur card.

- Un card poate avea 0 sau mai multe retrageri de la bancomate, dar o retragere poate să aparțină unui singur card.

- Un card poate avea 0 sau mai multe tranzacții, dar o tranzacții poate să aparțină unui singur card.

- O retragere de la bancomat poate fi făcută dintr-o singură locație, dar o bancă poate să aibă mai multe retrageri în istoric.

- O tranzacție poate avea 0 sau un furnizor de utilități dacă este o factură, dar un furnizor poate apărea în mai multe dintre tranzacții.

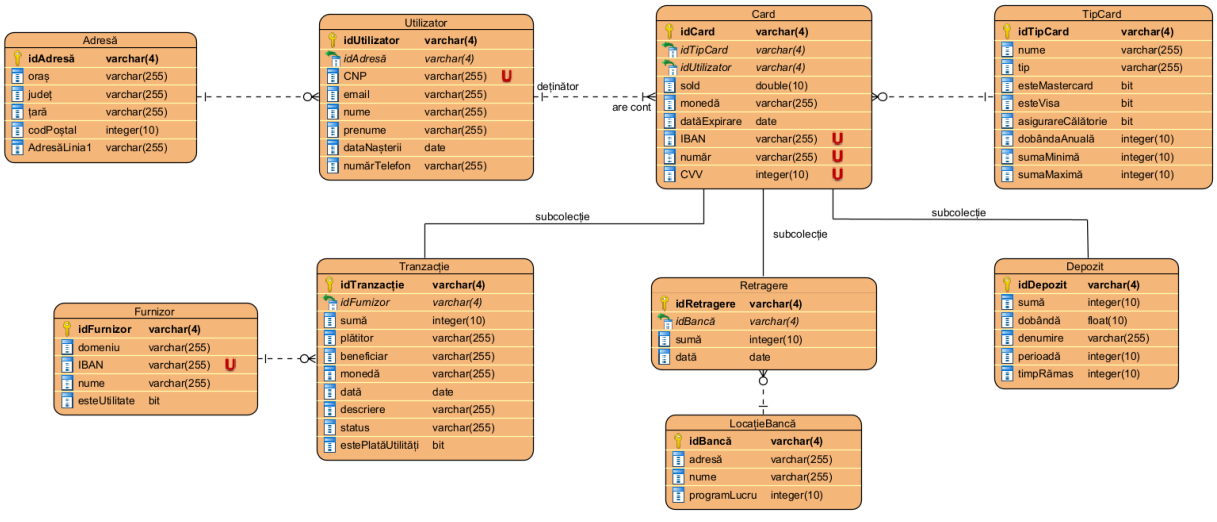



*Figura 21 - Schema logică a bazei de date*

Am decis organizarea tabelelor Tranzacție, Retragere și Depozit ca și subcolecții ale tabelei-părinte Card, așa cum este reprezentat în schema logică din Figura. Astfel, de fiecare dată când clientul dorește să obțină informațiile pentru un card nu o să mai fie nevoie de un JOIN cu alte tabele, ci pur și simplu se accesează lista de subcolecții deținută.

### 3.3.3. Proiectarea interfețelor utilizator

Pentru a reprezenta ferestrele de vizualizare a informațiilor s-au folosit dreptunghiuri cu unghiuri drepte, iar acolo unde aplicația impune utilizatorului introducerea unor date, ferestrele sunt reprezentate prin dreptunghiuri cu colțurile rotunjite. Prin elaborarea acestei scheme grafice, aferente fluxului de activități din sistemul informatic care urmează să fie implementat, au fost conturate, totodată, și principalele cazuri de utilizare. În figură este descrisă o hartă a ecranelor aplicației mobile de dezvoltat.

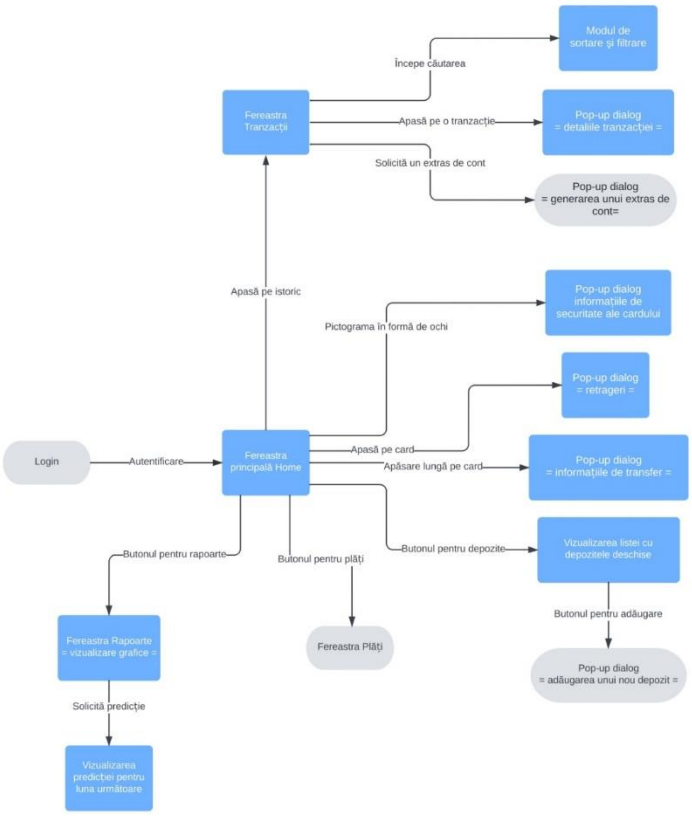



*Figura 22 - Proiectarea interfețelor utilizator*

### 3.3.4. Arhitectura sistemului

Componentele constituie module de cod care, în funcție de conținutul lor, pot fi: componente care conțin cod sursă, componente binare, respectiv executabile. Prezentarea acestora are rolul de a descrie componentele implementate de sistemul informatic, dar și dependențele care există între acestea și resursele alocate.

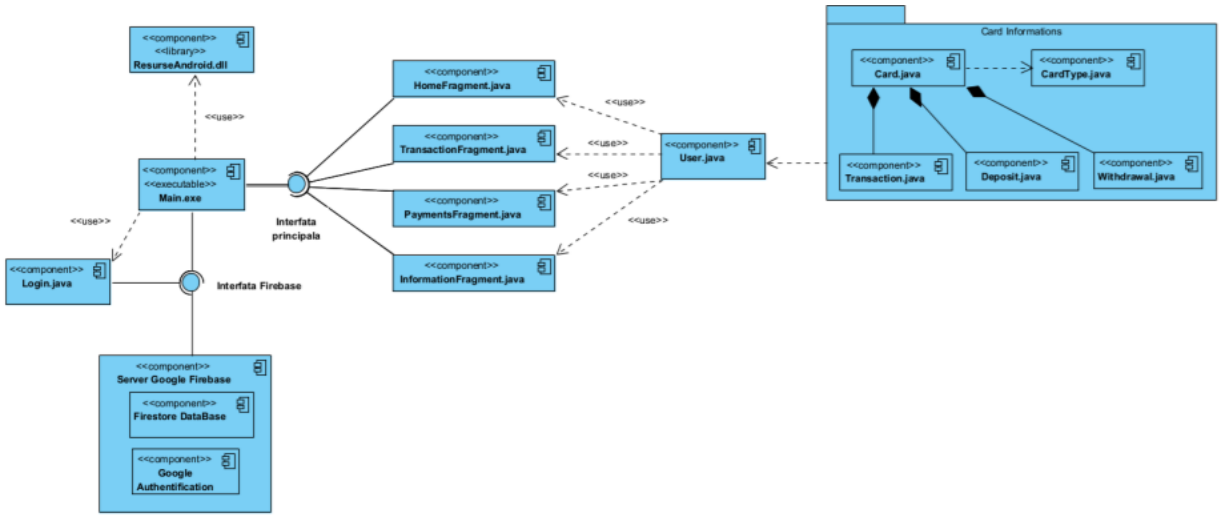



*Figura 23 - Diagrama de componente*

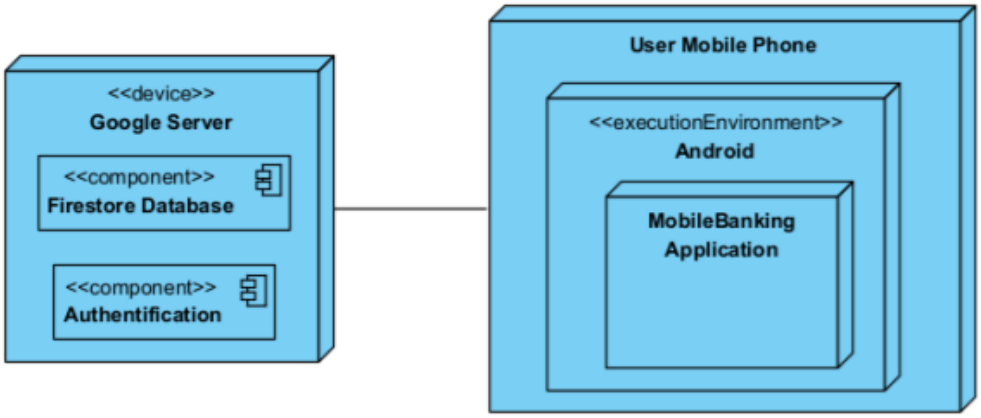



*Figura 24 - Diagrama de desfășurare*

# CAPITOLUL 4. REALIZAREA ȘI PREZENTAREA APLICAŢIEI

## 4.1. Implementarea aplicației

### 4.1.1. Construirea bazei de date

Înainte de a începe implementarea propriu-zisă a aplicației, este necesară construirea bazei de date care să conțină toate informațiile despre clasele de business menţionate în capitolul legat de proiectare a sistemului informatic. Tabelele generate, alături de informațiile stocate vor contribui la simularea unei legături în timp real cu baza de date a unui sistem bancar fictiv.

Firestore, serviciul de cloud database pus la dispoziție de Google Firebase, funcționează ca o bază de date NoSQL, ceea ce le oferă utilizatorilor un grad de libertate mai mare în ceea ce privește construirea tabelelor și a legăturilor dintre acestea. Cu toate că baza de date a aplicației este concepută pentru a ține cont și de regulile SQL, s-a profitat, totodată, de avantajele aduse de Firestore, mai cu seamă acolo unde crearea unor subcolecții ușurează semnificativ realizarea legăturilor dintre două sau mai multe tabele.

Referitor la logica de întocmire a tabelelor „Addresses”, „BankLocations”, „CardTypes”, „Providers” și „Users”, a fost utilizată abordarea SQL, legăturile dintre acestea realizându-se cu ajutorul anumitor câmpuri care au roluri de Id și de „chei secundare”. Tabelele „Cards”, „Transactions”, „Deposits” și „Withdrawals” au, însă, la bază o logică de organizare arborescentă, dat fiind că tabela „Cards” este tabela „părinte” sau „rădăcină”, pe când celelalte trei tabele devin subcolecții, „fii” sau „frunze”, aparținând unui card specific.

Acestă organizare logică a bazei de date ușurează intens procesul de colectare a datelor atunci când aplicația le solicită, întrucât nu mai este necesară căutarea în tabelele cu pricina după cheia secundară, ci literalmente toate datele din subcolecții sunt accesate printr-o singură interogare și se cunoaște, fără îndoială, faptul că acestea aparțin cardului curent, fără a mai fi necesare alte verificări suplimentare.

Firestore este foarte eficient, totodată, și din punctul de vedere al posibilității de a crea o întreagă bază de date utilizând exclusiv interfața cu utilizatorul, nemaifiind necesară scrierea de

cod pentru generare. S-a folosit consola pusă la dispoziție de serviciul Google, iar cu ajutorul ei s-au generat toate tabelele, ținându-se cont de regulile descrise în etapa de proiectare, și s-au populat cu date, având grijă ca toate numele câmpurilor să fie denumite la fel pentru a se putea realiza corespunzător căutările necesare mai târziu în cadrul aplicației.

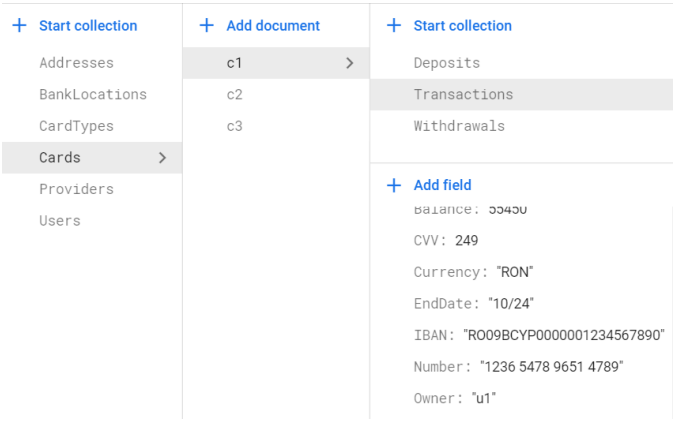



*Figura 25 - Tabela Cards împreună cu subcolecțiile sale*

### 4.1.2. Integrarea Google Firebase cu Android în vederea accesării serviciilor Firestore și Authentication

În cele ce urmează, se dorește folosirea bazei de date generată anterior în cadrul aplicației, astfel încât să fie posibilă actualizarea acesteia în timp real prin operații de interogare, adăugare, ștergere, respectiv editare. Mediul de dezvoltare Android SDK permite conectarea la Google Firebase prin parcurgerea a doar doi pași simpli: autentificarea cu Contul Google în care s-a generat baza de date și adăugarea proiectului din Firestore în aplicația Android prin adăugirea anumitor dependențe în modulul build.gradle.

```
plugins {
    id 'com.android.application'
    id 'com.google.gms.google-services'
}
dependencies {
    classpath "com.android.tools.build:gradle:4.1.1"
    classpath 'com.google.gms:google-services:4.3.5'

dependencies {
    implementation 'com.google.firebase:firebase-firestore:22.1.1'
```

*Figura 26 - Dependențele indispensabile pentru conectarea la Firestore*

Pentru pagina dedicată autentificării este utilizat Firebase Authentification, deoarece pune la dispoziție servicii back-end, SDK ușor de folosit și librării UI gata construite pentru a autentifica utilizatorii aplicației, și se implementează în cadrul aplicației analog cu procesul descris mai sus. Pentru aplicația dezvoltată în această lucrare, s-a optat pentru conectarea pe baza introducerii numărului de telefon, pe care se primește ulterior prin SMS un cod de verificare generat automat și trimis, întrucât această modalitate de logare prezintă un nivel mai înalt de securitate decât cea clasică. Implementarea funcției de logare în cadrul aplicației se poate observa în figura de mai jos:

```
private void verifyPhoneNumberWithCode(String verificationId, String code) {
    pd.setMessage("Verifying Code");
    pd.show();

    PhoneAuthCredential credential = PhoneAuthProvider.getCredential(verificationId, code);
    signInWithPhoneAuthCredential(credential);
}

private void signInWithPhoneAuthCredential(PhoneAuthCredential credential) {
    pd.setMessage("Logging In");

    firebaseAuth.signInWithCredential(credential)
            .addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                @Override
                public void onSuccess(AuthResult authResult) {
                    //successfully signed in
                    pd.dismiss();
                    String phone = firebaseAuth.getCurrentUser().getPhoneNumber();
                    Toast.makeText( context: MainActivity.this,  text: "Logged In as "+phone, Toast.LENGTH_SHORT).show();
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    //failed sigining in
                    pd.dismiss();
                    Toast.makeText( context: MainActivity.this,  text: ""+e.getMessage(), Toast.LENGTH_SHORT).show();
                }
            });

}
```

*Figura 27 - Funcție de logare cu Firebase Authentication*

Odată ce utilizatorul a putut fi identificat din aplicație se salvează în liste separate tranzacțiile din istoricul său, depozitele de economii deschise în cont și retragerile de la bancomate, pentru a putea fi accesate mult mai rapid ulterior. Este recomandat ca toate operațiile care se realizează pe baza de date să fie efectuate pe fire de execuție secundare, în ideea de a nu supraîncărca firul de execuție principal, iar modul de implementare poate fi observat în Figura 1-anexe.

## 4.2. Construirea aplicației în Android

### 4.2.1. Realizarea interfeței cu utilizatorul

Primul pas în procesul de elaborare a unei aplicații mobile este reprezentat întotdeauna de construirea unei interfețe cu utilizatorul care să contureze modul în care va arăta aplicația și care să-l ghideze pe utilizator legat de cum să între în contact cu partea funcțională a acesteia. Android SDK este un mediu de dezvoltare despre care putem afirma faptul că se bazează pe arhitectura Model-View-Controller (MVC), mai exact:

- Partea de Model este reprezentată de clasele corespunzătoare tuturor tabelelor din cadrul bazei de date, iar Google Firestore are capacitatea de a face corespondența perfectă între un obiect din aplicație și o înregistrare din baza de date, atâta timp cât denumirile clasei declarate în aplicație și ale atributelor sale sunt identice cu cele alese în baza de date, dar și dacă au fost implementați atât setteri, cât și getteri pentru toate câmpurile, așa cum se poate observa în Figura 2-anexe.
- Partea de View este despre tot ceea ce reprezintă elemente vizuale, widget-uri, dimensionare, așezare în pagină și animații. Toate acestea pot fi regăsite în fișierele de tip XML ale Android, unde pot fi conturate și personalizate fiecare dintre aspectele interfeței vizuale.

Fișierele XML sunt destul de ușor de folosit, iar cu ajutorul vizualizării în modul Designer, dar și a elementelor implicite pe care le oferă acesta, o interfață decentă poate fi schițată foarte rapid. Cu toate acestea, pentru a putea obține însă o interfață cât mai modernă, mai prietenoasă, capabilă să atragă utilizatorul și care să fie foarte ușor de înțeles și de utilizat, este nevoie de mai multă dedicare, căci impune introducerea de elemente vizuale externe, precum pictogramele sau imaginile în format PNG cu care se pot personaliza ferestre sau butoane, dar și de suprapuneri complexe și de creare a unor animații care vor fi aplicate obiectelor.

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="true"
    android:interpolator="@android:anim/accelerate_interpolator"
    android:startOffset="300">

    <translate
        android:fromYDelta="10%p"
        android:toYDelta="0%"
        android:duration="400"/>
    <alpha
        android:fromAlpha="0.0"
        android:toAlpha="1.0"
        android:duration="400"
        />
</set>
```

*Figura 28 - Exemplificare de animație privind mișcarea în sus*

Fiecare fereastră are asociat un fișier XML, iar cel mai complex ecran al aplicației este cel principal, Home, care include foarte multe elemente suprapuse, fiecare cu un tip diferit de animație.

- Partea de Controller este reprezentată în Android prin activități, fragmente, dar și ecrane de tip pop-up sau dialoguri. Acestea sunt fișiere Java și clase specializate, derivate din modulele AppCompatActivity, Fragment și DialogFragment, care satisfac cuplarea dintre partea vizuală (fișierele XML) și partea de business (clasele Model).

Mai mult decât faptul că în aceste clase se săvârșește „cuplarea" dintre părțile Model și View, aici se întâmplă și implementarea tuturor funcționalităților aplicației (evenimentele declanșate de elemente vizuale, funcțiile de prelucrare a datelor, operațiile în baza de date ș.a.m.d.).

Privind legătura dintre elementele vizuale și partea funcțională, prima dată se setează în interiorul funcției onCreate() starea conținutului de vizualizare aferent activității sau fragmentului respectiv cu layout-ul construit în prealabil în fișierul XML. Mai apoi, se inițializează câte o variabilă locală pentru fiecare element vizual din cadrul fișierului XML, iar prin intermediul acesteia va putea fi controlat comportamentul elementului vizual, ulterior făcându-se diverse operații și modificări în limbajul de programare Java.

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

    BottomNavigationView bottomNavigationView=findViewById(R.id.bottom_navigation);
    bottomNavigationView.setOnNavigationItemSelectedListener(navListener);
```

*Figura 29 - Java - Cuplarea fișierului XML cu MainActivity*

În urma analizei aplicațiilor din spațiul fintech care sunt deja listate pe piață, se poate observa faptul că majoritatea folosesc meniul principal situat în partea de jos a ecranului, acesta fiind cel mai ușor de înțeles și de utilizat. Drept urmare, și aplicația de mobile banking dezvoltată în cadrul acestei lucrări de licență va implementa un astfel de meniu.

Pentru construirea interfeței s-a decis organizarea acesteia în două ferestre de tip activitate (LoginActivity, MainActivity). Cea de-a doua activitate reprezintă fereastra principală a

aplicației, unde a fost generat meniul de control al tuturor sub-ferestrelor, care este de tipul BottonNavigationView, și care poate fi văzut în figura de mai sus.

Cu ajutorul acestui meniu se face trecerea între cele patru sub-ferestre ale aplicației, reprezentate prin fragmente, care se poziționează deasupra activității principale, aceasta având într-un fel sau altul rolul de „gazdă”. Prin acționarea fiecărui buton din cadrul meniului principal, este pus în lumină fragmentul corespunzător, iar întregul proces de schimbare de fragmente este gestionat în MainActivity, acest lucru putând fi observat în Figura 3-anexe.

De îndată ce a fost finisată construirea „scheletului” aplicației, se execută partea de dezvoltare a multitudinii de funcționalități din fiecare fragment în parte.

Fragmentul Home este cel care se deschide în mod implicit, iar aici pot fi vizualizate informațiile de securitate despre cardul utilizatorului, soldul curent, istoricul retragerilor și depunerilor de la bancomate, alături de depozitele de economii deschise și, mai mult, posibilitatea de a deschide sau de a închide unul sau mai multe.

Fragmentul Tranzacții afișează istoricul tuturor tranzacțiilor înregistrate pentru cardul curent, oferind și posibilitatea de a le filtra și sorta pe mai multe criterii concomitent. Tot aici se mai poate genera și un extras de cont care va fi trimis la adresa de e-mail a utilizatorului.

Fragmentul Plăți face posibilă inițierea de plăți, iar Fragmentul Rapoarte este locul unde sunt generate grafice „plăcintă” pe baza istoricului de cheltuieli și unde se pune la dispoziţie o predicție cu privire la suma care va fi cheltuită luna viitoare, folosindu-se de istoricul cheltuielilor.

### 4.2.2. Adaptoare personalizate

Informațiile preluate din baza de date sunt adesea stocate în liste, iar pentru a putea fi prezentate utilizatorului printr-o interfață vizuală este necesar adaptorul. Termenul descrie modul în care o listă de informații este transpusă în interfață, mai exact, fiecărui element din lista i se atribuie același șablon format din mai multe elemente vizuale prin care utilizatorul va putea să vizualizeze și să interacționeze cu lista și, de asemenea, să declanșeze diverse evenimente.

Mediul de dezvoltare Android le asigură programatorilor o gamă de adaptoare deja implementate, care pot fi folosite conform structurii listei cu informații. Însă, aceste adaptoare implicite sunt superficiale, constând, de fapt, în minimul necesar pentru afișarea unei liste, respectiv, doar câmpuri de tipul TextView.

Totuși, pentru a obține o interfață cât mai atractivă și intuitivă cu putință, capabilă să ofere multiple funcționalități utilizatorului, este necesar să existe niște adaptoare cât mai complexe care, pe lângă câmpurile de text, să cuprindă și elemente cu care se poate interacționa: butoane, checkbox-uri sau alte elemente vizuale, cum ar fi imaginile și pictogramele.

Android SDK oferă, de asemenea, posibilitatea de a defini adaptoare personalizate care pot fi utilizate în implementarea aplicației. Un prim pas în definirea unui adaptor este reprezentat de crearea unui fișier XML special dedicat acestuia, în acesta se conturează modul de reprezentare a unui singur item din întreaga listă, rezultând șablonul utilizat pentru reprezentarea ulterioară a fiecărui element din cadrul listei. În acest mod, nu este nevoie de crearea unui design pentru toată lista, ci doar pentru un singur item care este mai apoi replicat de câte ori este nevoie.

Pentru a menține o temă generală care să individualizeze aspectul aplicației, s-a ales același șablon de bază pentru toate adaptoarele create pe parcursul ferestrelor din aplicație, dar fiecare este personalizat în funcție de nevoile listei de informații pe care o reprezintă. Șablonul de la care s-a pornit este prezentat în figura de mai jos, iar codul din fișierul XML din spate se regăsește în Figura 4-anexe.


*Figura 30 - Șablon de adaptor personalizat*

Fișierul XML se ocupă de partea vizuală, dar nu și funcțională, deci, în continuare este nevoie de cuplarea dintre șablon și listă pentru a putea utiliza adaptorul împreună cu partea sa funcțională în cadrul activităților. Această parte funcțională este implementată folosind o clasă Java specială, derivată din clasa ArrayAdapter. În această clasă, două elemente sunt esențiale: constructorul care urmează o „schiță” impusă de adaptoarele pe care le furnizează în mod implicit

Android și o suprascriere a funcției getView(), în cadrul căreia se face propriu-zis corespondența dintre șablonul vizual și partea informațională.

„getView” este o funcție apelată pentru fiecare element în parte din listă și care are rolul de a transpune informația din element în șablonul construit ulterior în fișierul XML. Codul aferent unei clase de tip adaptor poate fi vizualizat în Figura 5-anexe.

### 4.2.3. Ferestre de dialog personalizate

Un context similar cu cea a adaptoarelor este și cel al ferestrelor de dialog. Ferestrele de dialog sunt acele ferestre pop-up care apar pe ecran de fiecare dată când în realizarea unei acțiuni principale este întâlnită una intermediară. Printre cele mai întâlnite și cunoscute situații, se regăsește cea a ferestrei cu o întrebare de siguranță atunci când este declanșată ștergerea unui element.

Pentru o implementare rapidă a acestora, Android le oferă programatorilor așa-numitul AlertDialog.Builder, o modalitate de a defini și de a construi inline o fereastră pop-up, printr-o unică instrucțiune complexă. După cum sugerează și numele, AlertDialog.Builder este o implementare a design pattern-ului Builder, care facilitează crearea de obiecte complexe, cu numeroase atribute sau specificații, care pot fi opționale sau nu. Pentru a inițializa o fereastră simplă cu un mesaj și două butoane care activează două acțiuni distincte, se urmărește schema următoare:

```
new AlertDialog.Builder(view.getContext())
        .setIcon(android.R.drawable.ic_delete)
        .setTitle("Esti sigur?")
        .setMessage("Vrei sa inchizi acest depozit?")
        .setPositiveButton( text: "Da", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Deposit depositToErase=depositList.get(position);

        .setNegativeButton( text: "Nu", listener: null)
        .show();
```

*Figura 31 - Exemplu de utilizare a AlertDialog.Builder*

Pe de altă parte, ferestrele pop-up pot fi conduse la un nivel mai înalt și pot îndeplini mult mai multe funcționalități decât afișarea unui mesaj de siguranță. Ca și la adaptoare, acest lucru necesită implementarea unui dialog personalizat care să aibă un fișier XML, în care se

definește design-ul vizual al ferestrei și o clasă specială, derivată din DialogFragment, care descrie toate acțiunile disponibile în dialog.

În aplicația dezvoltată în cadrul lucrării, sunt implementate mai multe tipuri de dialoguri personalizate, atât pentru afișarea unor cantități mai mari de informații atunci când este selectat un element din interfață, precum și pentru definirea unor formulare de preluare de informații prin care sunt create noi obiecte în aplicație și sunt actualizate concomitent informațiile din baza de date la distanță. Astfel, dialogurile personalizate pot fi chiar considerate ca niște „mini-fragmente" sau „mini-activități" care pot desfășura mai multe funcții în interiorul unei activități „gazdă", fiind astfel oportune pentru îndeplinirea unor acțiuni sau condiții intermediare în completarea unei acțiuni principale. Modul de implementare al ferestrei pop-up în vederea adăugării unui depozit de economii poate fi observat în Figura 7-anexe.

### 4.2.3. Interfața Tranzacții

Pentru implementarea celui de-al doilea fragment, cel aferent tranzacțiilor, primele funcții care se apelează sunt cele care declanșează animații și tranziții de deschidere, într-o manieră similară celor din fragmentul Home, al căror scop este să ofere interfeței un aspect cât mai plăcut și profesionist în fața utilizatorului. Odată cu deschiderea ferestrei sunt preluate informațiile din baza de date și sunt asociate unui ListView care folosește un adaptor personalizat pentru reprezentarea vizuală a acestor informații. Lista este sortată în prealabil și astfel istoricul este afișat în ordine cronologică, începând cu cea mai recentă tranzacție și încheind cu cea mai veche. Fiecare item din listă poate fi selectat, iar atunci când selecția este declanșată, informațiile despre tranzacția vizată sunt trimise către o fereastră dialog personalizată care îi afișează utilizatorului toate detaliile despre transferul bancar.

Chiar și așa, cel mai important aspect din cadrul acestui fragment este modulul de sortare și filtrare a tranzacțiilor, care facilitează utilizatorului organizarea mai convenabilă a propriul istoric. Sortarea se realizează în ordine crescătoare sau descrescătoare, în funcție de suma tranzacționată , iar ambele opțiuni pot fi ascunse sau făcute vizibile pe ecran prin intermediul unui buton menit pentru a activa sau dezactiva partea de sortare.

```
public void sortareCrescator(View view){
    Collections.sort(HomeFragment.transactionList,Transaction.esteCrescator);
    checkForFilter(view);

    deselecteazaToateButoaneleSortare();
    selecteazaButon(crescatorBtn);
}

public void sortareDescrescator(View view){
    Collections.sort(HomeFragment.transactionList,Transaction.esteCrescator);
    Collections.reverse(HomeFragment.transactionList);
    checkForFilter(view);

    deselecteazaToateButoaneleSortare();
    selecteazaButon(descrescatorBtn);
}
```

*Figura 32 - Funcțiile pentru sortarea crescătoare, respectiv descrescătoare*

Componenta legată de filtrare poate fi, de asemenea, afișată sau ascunsă de pe ecran prin apăsarea unui buton declanșator, însă acest modul este semnificativ mai complex. Filtrarea se poate face după patru criterii distincte: facturi, venituri, cheltuieli, respectiv după o dată specifică. Mai mult decât atât, este disponibilă și o casetă de căutare prin care pot fi regăsite toate tranzacțiile a căror denumire conține șirul de caractere căutat de utilizator. Pentru a răspunde mai bine la nevoile utilizatorilor, cele două module de sortare și filtrare funcționează 100% sincronizat și în mod concomitent, iar mai multe filtre și moduri de sortare pot fi selectate în același timp.

A fost scrisă o funcție separată pentru fiecare filtru în parte pentru a gestiona implementarea, iar pentru a se ține cont tot timpul de ce filtre au fost deja aplicate înainte, este declarată la început o listă de string-uri, unde sunt stocate butoanele selectate. Când este aplicat un nou filtru, toate celelalte filtre selectate anterior sunt, de asemenea, verificate. Codul prin care se implementează funcția de filtrare după descrierea specificată poate fi văzut în Figura 8-anexe.

### 4.2.4. Obținerea prin e-mail a extrasului de cont

Utilizatorul poate solicita un extras de cont, care va fi generat automat ca fișier Excel și apoi trimis la adresa de e-mail cu care este conectat în cont imediat după setarea perioadei dorite.

În cadrul ferestrei „Extras de cont” din fragmentul Tranzacții, la apăsarea butonului „Solicită”, întregul istoric este mai întâi filtrat pentru a păstra doar tranzacțiile din perioada căutată. Pe baza acestora, se creează o nouă listă, iar aceasta va fi utilizată apoi ca bază pentru generarea fișierului Excel cu toate informațiile necesare, după generare, acesta fiind salvat în

memoria externă a telefonului. Codul aferent generării fișierului Excel poate fi vizualizat in Figura 9-anexe.

Pentru realizarea operațiunii de trimitere a e-mailurilor a fost utilizat un API (Application Programming Interface) denumit JavaMail, care facilitează automatizarea trimiterii e-mailurilor din cadrul aplicației dezvoltate. Mai întâi de toate, pentru folosirea acestuia au fost necesare importarea a trei arhive jar, respectiv activarea permisiunii de utilizare a Internetului în cadrul fișierului AndroidManifest.

Pentru a trimite automat e-mailul se determină prima dată proprietățile pentru comunicarea securizată pe Internet și pentru stabilirea pachetelor utilizate și, apoi se deschide o nouă sesiune pentru autentificarea contului folosit pentru trimiterea e-mailurilor.

```
Properties props=new Properties();
props.put("mail.smtp.auth","true");
props.put("mail.smtp.starttls.enable","true");
props.put("mail.smtp.host","smtp.gmail.com");
props.put("mail.smtp.port","587");
Session session=Session.getInstance(props,
        new javax.mail.Authenticator(){
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(username, password);
            }
        });
```

*Figura 33 - Proprietățile și o nouă sesiune JavaMailAPI*

Pe mai departe, se creează un obiect nou de tip mesaj pentru care se stabilesc destinatarul, subiectul și conținutul, iar apoi eventual să fie trimis, urmărind însă ca această operație să fie executată pe un fir de execuție secundar.

### 4.2.5. Rapoarte și grafice

În ultima fereastră a meniului aplicației, utilizatorii pot consulta grafice elaborate pe baza istoricului de cheltuieli. Când fragmentul pentru rapoarte este inițializat, se desenează graficul general, care oferă o imagine de ansamblu bazată pe întregului istoric al cheltuielilor, iar în continuare se poate solicita vizualizarea graficelor destinate pentru fiecare lună în parte.

Primul pas în realizarea unui raport sub formă de grafic Pie este importarea unei biblioteci numite MPAndroidChart, care pune la dispoziție toate funcțiile și opțiunile necesare.

După conectarea aplicației la această bibliotecă, noile widget-uri care sunt folosite pentru afișarea unui sumar de informații importante sub forma unor diagrame și grafice devin disponibile în fișierele XML. Pentru acestea se configurează ulterior datele de intrare din codul Java aferent fișierului XML.

În prealabil se inițializează o listă de obiecte de tipul PieEntries pe baza căreia se va construi un obiect PieDataSet atribuit graficului. Odată ce datele numerice care urmează să fie reprezentate în grafic sunt configurate, următorul pas este configurarea aspectului acestuia setând culorile, forma, titlul, legenda și, opțional, o animație de deschidere (Figura 6-anexe).

### 4.2.6. Predicția sumei ce va fi cheltuită luna următoare

Pentru realizarea unei predicții aproximative în ceea ce privește suma care va fi cheltuită luna următoare, se utilizează în principal un algoritm de machine learning scris cu ajutorul limbajului Python, care constă într-un model de predicție construit folosind librăriile sklearn și tensorflow.keras. Odată construit modelul, acesta este mai întâi testat pe setul de date primit pentru verificarea corectitudinii rezultatelor și apoi va fi utilizat pentru a prezice suma pentru luna următoare.

Predicțiile se realizează pe baza sumelor totale cheltuite de-a lungul lunilor precedente, astfel că aceste informații vor fi preluate din baza de date Firestore și pe baza lor va fi generat un fișier CSV în cadrul aplicației care va avea rolul de sursă de date pentru algoritmul din Python.

Pentru integrarea și rularea în cadrul aplicației a algoritmului scris în Pyhon, se folosește Chaquopy, un SDK de Python special pentru Android. După adăugarea dependențelor necesare acestui SDK este adăugat în folderul de assets din proiectul aplicației fișierul cu codul Python respectiv, iar pasul final este reprezentat de inițializarea unei instanțe de tip Python și a unui obiect de tip PyObject prin care scriptul cu algoritmul de predicție va fi rulat atunci când butonul „Predict” este declanșat. Rezultatul returnat de către acest obiect va fi afișat utilizatorului în interfață.

## 4.3. Prezentarea aplicației

La deschiderea aplicației, se lansează în execuție fereastra de Login, unde utilizatorul se autentifică folosind numărul său de telefon, pe care primește prin SMS un cod de verificare generat automat. Autentificarea OTP (One Time Password) este o metodă prin care se asigură faptul că informațiile tale, ca și utilizator, sunt mereu în siguranță și că numai tu ai acces la ele, conferind un nivel mai înalt de securitate decât autentificarea clasică.

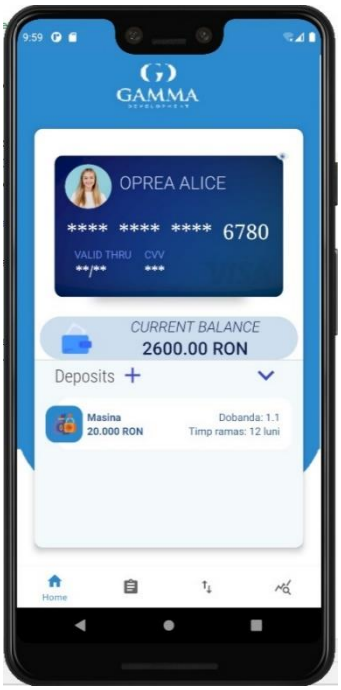
Odată permis accesul în aplicație, se deschide fereastra principală a meniului, Home, afișând utilizatorului informațiile prioritare despre contul său, și anume: soldul curent și principalele informații de securitate despre card. Acestea inițial sunt ascunse, dar pot fi făcute vizibile apăsând butonului cu pictograma în formă de ochi. Totodată, aici pot fi vizualizate retragerile și depunerile de numerar de la bancomate, dar și depozitele de economii deschise, împreună cu posibilitatea de a deschide unul nou sau de a închide unul sau mai multe dintre acestea.

La o apăsare scurtă pe înfățișarea cardului sunt afișate într-o fereastră de tip pop-up toate retragerile și depunerile realizate la bancomate (ATM-uri), iar la o apăsare mai lungă sunt afișate informațiile necesare pentru realizarea unui transfer bancar, și anume: IBAN, numele complet al posesorului, monedă și bancă, oferindu-se și posibilitatea de a le copia automat în clipboard la apăsarea unui buton specific.

La apăsarea succesivă a săgeții din dreptul secțiunii aferente depozitelor, acestea sunt afișate sau ascunse. Pentru a închide un depozit, utilizatorul trebuie să gliseze pe ecran în dreptul depozitului respectiv spre stânga ca să poată face vizibilă această opțiune. Este solicitat consimțământul utilizatorului printr-o întrebare de securitate afișată într-o fereastră pop-up înainte de a fi finalizată acțiunea.

În continuare, utilizatorul poate deschide un nou depozit de economii apăsând butonul „+”, iar când acesta este declanșat, se va deschide o fereastră cu un formular de completat. Se poate seta denumirea depozitului, perioada și suma, iar rata dobânzii este calculată de aplicație în funcție de suma și durata depozitului.

Fereastra Tranzacții afișează o lista completă a tranzacțiilor din istoricul salvat în baza de date, cu posibilitatea de a filtra și sorta informațiile primite după multiple criterii specifice. Inițial, sunt afișate doar tranzacțiile în interfață, dar la apăsarea celor două butoane corespunzătoare pentru sortare, respectiv filtrare, vor fi expuse două module cu mai multe butoane, alături de o casetă de căutare, responsabile de implementarea funcționalităților de căutare. Funcțiile de filtrare și sortare se sincronizează și operează într-un mod concomitent, astfel încât să permită aplicarea mai multor filtre în același timp.



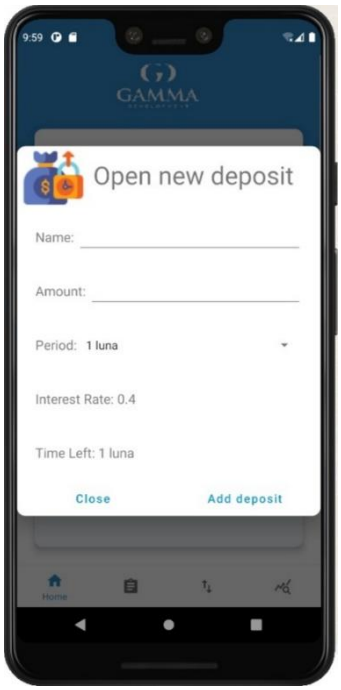





*Figura 34 – Fereastra principală Home* *Figura 35 – Fereastra de adăugare depozit nou* *Figura 36 – Fereastra Tranzacții*

Tot de aici, utilizatorul poate solicita consultarea unui extras de cont. La apăsarea butonului „Extras de cont” se deschide un dialog personalizat în cadrul căruia utilizatorul setează perioada de timp dorită cu ajutorul a două widget-uri de tipul DatePicker. În cazul în care există tranzacții efectuate în acea perioadă, ele vor listate într-un document Excel care se generează în momentul cu pricina și care este salvat mai întâi în memoria externă a telefonului. Ulterior, acest fișier Excel care conține toate informațiile extrasului de cont va fi trimis în mod automat pe e-mailul utilizatorului, iar dacă nu există tranzacții în perioada de timp selectată, utilizatorul va primi un mesaj de notificare.

Fereastra Plăți este locul în care vor putea fi inițiate transferuri bancare către diverși beneficiari și presupune un formular în care trebuie completate datele destinatarului, suma de plată și diverse detalii legate de descrierea plății. La bifarea căsuței „ Plătește o factură de utilități”, aplicația îi pune la dispoziție utilizatorului o listă cu furnizorii de utilități disponibili deja în baza de date, scopul fiind ca acesta să nu mai fie nevoit să completeze manual datele de transfer.

În ultima fereastră a meniului aplicației, Raports, utilizatorul poate consulta mai multe rapoarte reprezentate sub forma unor grafice „plăcintă”, bazate pe istoricul de cheltuieli înregistrat în contul curent. La deschiderea ferestrei este reprezentat graficul general, care oferă o imagine de ansamblu pe baza întregului istoric de cheltuieli, iar mai apoi utilizatorul poate solicita vizualizarea graficelor aferente fiecărei luni în parte. Graficele arată rapoarte pentru diferite categorii de plăți în totalul cheltuielilor. Iar butonul „Predict” declanșează aplicarea unui algoritm de predicție, în urma căruia este aproximată suma ce va fi cheltuită în luna următoare, ținând cont în această predicție de obiceiurile financiare înregistrate în istoricul clientului până în acel moment.

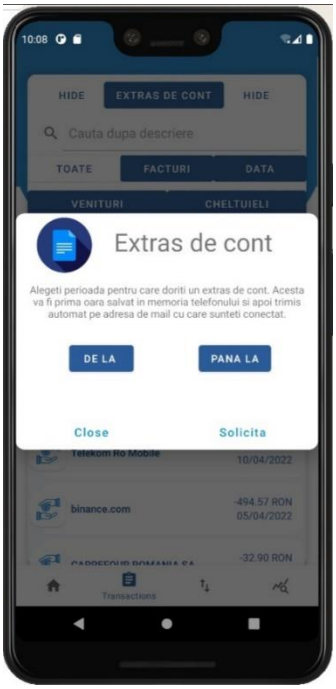



*Figura 37 - Generarea unui extras*

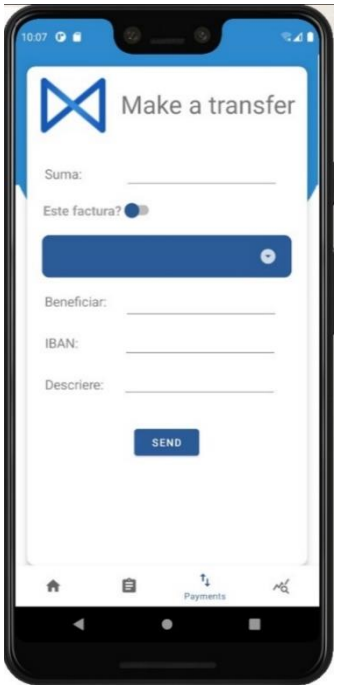



*Figura 38 - Fereastra Plăți*

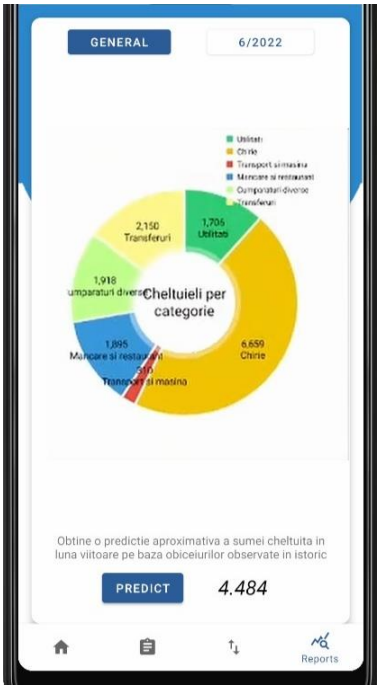



*Figura 39 - Fereastra Rapoarte*

# CONCLUZII

În urma cercetărilor efectuate în vederea realizării lucrării de licență, s-a putut observa cu ușurință faptul că domeniul digitalizării bancare - internet banking sau mobile banking, este unul relativ nou, un concept din sistemul bancar nu de foarte mult timp apărut în viață noastră, dar care s-a dezvoltat cu o viteză uimitoare, devenind chiar indispensabil în anumite circumstanțe. Într-adevăr, există țări în care s-a renunțat în proporții foarte mari la numerar, optându-se pentru varianta plății online sau contactless - cu telefonul mobil, dar și țări în care încă mai există o reticență greu de eradicat în ceea ce privește această tranziție de la fizic la sistemul aproape 100% online. România se află printre aceste țări care încă înregistrează proporții mari de tranzacții cash, oamenii simțind un nivel mai ridicat de siguranță și de apartenență atunci când își țin banii în buzunar, nu pe cardul bancar.

Cu toate acestea, promovarea şi încurajarea adoptării bankingului digital sunt foarte importante, întrucât, mai întâi de toate, acest instrument conferă o imagine de ansamblu mai clară și mai compactă asupra tranzacțiilor care se efectuează la nivel național și internațional, astfel reducându-se drastic mişcările ilegale de bani, iar mai apoi ne putem gândi la contextul pandemic care a fost neașteptat și care a restricționat considerabil contactul între persoane (față în față) și interacțiunea socială.

Aplicația Android, dezvoltată și implementată pentru această lucrare de licenţă, a fost gândită în aşa fel încât să îndeplinească anumite nevoi tocmai cu scopul de a încuraja populația să adopte mai ușor aplicațiile de mobile banking. Printre obiectivele setate se regăseşte utilizarea unei aplicații de mobile banking care face experiența de utilizare foarte plăcută prin prisma faptului că are o interfață simplă, modernă și, în același timp, foarte prietenoasă și ușor de folosit pentru orice utilizator, indiferent de categoria de vârstă, fiind bazată, totodată, pe obiceiurile de navigare ale clienților, căci integrează în funcționalitățile sale gesturile obișnuite, precum derulare, glisare, tap, și astfel crește ușurința de utilizare, rapiditatea si flexibilitatea serviciilor bancare. Pe de altă parte, a fost pus accentul pe implementarea proceselor de bază necesare pentru efectuarea și gestionarea operațiunilor bancare, precum: vizualizarea tuturor informațiilor de securitate ale cardului, dar și a soldului curent într-un singur loc, accesarea depozitelor de economii deschise, dar și posibilitatea de a deschide unul nou sau de a-l închide cu ajutorul

aplicației mobile, fără a mai fi necesară deplasarea până la cea mai apropiată sucursală, consultarea istoricului complet al tranzacțiilor înregistrate și posibilitatea de a le filtra și sorta după mai multe criterii relevante și, de asemenea, inițierea de transferuri bancare rapide către orice beneficiar sau către furnizori de utilități/servicii pentru plata unor facturi recurente.

Este bine cunoscut faptul că populația majoritară din România nu dispune de cunoștințele de bază legate de educație financiară și de întocmirea adecvată a unui buget personal sau de familie. De aceea, în cadrul aplicației a fost întrevăzută o caracteristică specială care să vină în sprijinul utilizatorului în ceea ce privește gestionarea mai eficientă a bugetului. Astfel, aplicația de față se diferențiază de celelalte aplicații similare disponibile pe piață în prezent prin faptul că îi pune la dispoziţie utilizatorului rapoarte grafice lunare, construite per categorii, pe baza istoricului cheltuielilor din lunile trecute, menite să îl ajute să deslușească mai facil maniera în care sunt distribuiți și cheltuiți banii săi de-a lungul timpului.

Totodată, elementul inovator, neimplementat de nicio altă aplicație, este capacitatea de a oferi utilizatorului o predicție aproximativă referitoare la cheltuielile lor totale pentru luna viitoare, ținându-se cont de obiceiurile financiare înregistrate în istoricul tranzacțiilor. Prognoza este realizată pe baza unui algoritm de machine learning scris în limbajul de programare Python și integrat ulterior in interfața Android, algoritm care se actualizează în permanenţă și observă constant obiceiurile utilizatorului.

Pentru o viitoare versiune a aplicației se dorește extinderea modulului de predicție prin implementarea unei algoritm mai specific și mai eficient care să fie capabil să aproximeze cheltuielile pentru fiecare categorie în parte, nu decât per total, și care să poată prezice, de asemenea, raportul dintre venituri și cheltuieli pentru o perioadă mai îndelungată. Mai mult, este luată în considerare și funcționalitatea care permite setarea unor plăti recurente și ca programarea acestora să se realizeze în mod automat după selectarea periodicității, fără vreo intervenție a utilizatorului.

# BIBLIOGRAFIE


*Psihologia persuasiunii - Robert Cialdini.* (n.d.). Preluat de pe academia.edu: https://www.academia.edu/32101024/Psihologia_Persuasiunii_Robert_Cialdini_pdf

Wardynski. (2019, 11 24). *Technology and Society: How Technology Changed Our Lives.* Preluat de pe brainspire.com: https://www.brainspire.com/blog/technology-and-societyhow-technology-changed-our-lives

Sarreal, R. (2019, 05 21). *History of Online Banking: How Internet Banking Went Mainstream.* Preluat de pe GOBankingRates: https://www.gobankingrates.com/banking/banks/history-online-banking/

Statista Research Department. (2021, 05 28). *Online banking penetration in European markets 2020.* Preluat de pe statista.com: https://www.statista.com/statistics/222286/onlinebanking-penetration-in-leading-european-countries/

*Digitalizarea serviciilor bancare în România, accelerată de epidemia de coronavirus.* (2020, iulie 16). Preluat de pe iSense Solutions: https://www.isensesolutions.ro/digitalizarea-serviciilor-bancare-in-romania-accelerata-de-epidemia-de-coronavirus/

Voinea, O. (2021, martie 9). *Anul relansării în banking.* Preluat de pe revistabiz.ro: https://www.revistabiz.ro/omer-tetik-bt-anul-relansarii-in-banking/

Barliga, G. (2021, 11 25). *Mobile banking pentru era digitală.* Preluat de pe revistabiz.ro: https://www.revistabiz.ro/andrei-stamatian-unicredit-bank-mobile-banking-pentru-era-digitala/

*Ce inseamna mobile banking?* (2021, 05 14). Preluat de pe Bancherul.ro: https://www.bancherul.ro/stire.php?id_stire=20627&titlu=ce-inseamna-mobile-banking

*The Evolution of Online Banking.* (2012, 08 03). Preluat de pe instantShift: http://www.instantshift.com/2012/08/03/the-evolution-of-online-banking-infographic/

Cașotă, F. (2021, 04 20). Preluat de pe start-up.ro: https://start-up.ro/clientii-cec-bank-vor-putea-face-transferuri-cu-numarul-de-telefon-catre-orice-banca/

*Mobile banking.* (2021, 05 06). Preluat de pe Wikipedia: https://en.wikipedia.org/wiki/Mobile_banking#History

Pankhudi Pandey. (2019, 06 23). *Mobile Banking: History and what it does.* Preluat de pe ShapeMyApp.com: https://www.shapemyapp.com/blogs/mobile-banking-history-and-what-it-does

Pilcher, J. (2010, 06 04). *Survey Proves Value of Online Banking, Bill Pay.* Preluat de pe The Financial Brand: https://thefinancialbrand.com/12000/fiserv-2010-online-banking-bill-paytrends-report/

Marous, J. (2014, 08 11). *300 Mobile Payment and Digital Banking Trends.* Preluat de pe The Financial Brand: https://thefinancialbrand.com/41465/300-mobile-banking-digital-paymentstrategic-planning-statistics/

*Phenomenal Growth for Bill Pay.* (2010, 12 15). Preluat de pe MitekMobile: https://miteksystems.wordpress.com/2010/11/15/phenomenal-growth-for-bill-pay/

The Financial Brand. (2010, Iunie 4). *The financial brand*. Preluat de pe Survey Proves Value of Online Banking: https://thefinancialbrand.com/41465/300-mobile-banking-digital-payment-strategic-planning-statistics/

*Aproape un român din trei renunță la interacțiunea cu băncile fizic, trecând la servicii digitale – studiu SAS.* (2022, februarie 3). Preluat de pe https://edupedu.ro/: https://www.edupedu.ro/aproape-un-roman-din-trei-renunta-la-interactiunea-cu-bancile-fizic-trecand-la-servicii-digitale/

Laricchia, F. (2022, februarie 7). *Market share of mobile operating systems worldwide 2012-2022.* Preluat de pe Statista: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

Buchholz, K. (2020, august 26). *Apple or Android Nation? Operating System Popularity Across Countries.* Preluat de pe Statista: https://www.statista.com/chart/22702/andoid-ios-market-share-selected-countries/

Chen, J. (2021, 02 03). *Android Operating System.* Preluat de pe Investopedia: https://www.investopedia.com/terms/a/android-operating-system.asp

*Android vs. iOS.* (n.d.). Preluat de pe Diffen: https://www.diffen.com/difference/Android_vs_iOS

*Ghidul începătorilor în Java: Ce este și de ce să înveți acest limbaj de programare.* (2019, 09 12). Preluat de pe CODECOOL: https://codecool.com/ro/blog/ghid-java-incepatori/

*What is Python? Executive Summary.* (n.d.). Preluat de pe Python.org: https://www.python.org/doc/essays/blurb/

*Cloud Firestore.* (n.d.). Preluat de pe Firebase Documentation: https://firebase.google.com/docs/firestore

Dearmer, A. (2020, 12 01). *Firebase vs. MySQL: Battle of the Databases.* Preluat de pe Xplenty: https://www.xplenty.com/blog/firebase-vs-mysql/

*What is Python? Executive Summary.* (n.d.). Preluat de pe Python.org: https://www.python.org/doc/essays/blurb/

# ANEXE

## Listă de figuri

## Listă de tabele



## Listă de figuri - anexe

```
private void getTransactions() {

    Thread thread = new Thread() {
        @Override
        public void run() {
            database.collection( collectionPath: "Cards/"+cardId+"/Transactions")
                    .get()
                    .addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
                        @Override
                        public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
                            transactionList.clear();
                            for(QueryDocumentSnapshot document:queryDocumentSnapshots){
                                Transaction transaction=document.toObject(Transaction.class);
                                transactionList.add(transaction);
                                //Toast.makeText(getContext(), transaction.toString(), Toast.LENGTH_SHORT).show();
                            }
                            Collections.sort(transactionList,Transaction.esteCronologic);
                        }
                    });
            new Handler(getMainLooper()).post(new Runnable() {
                @Override
                public void run() {

                }
            });
        }
    };
    thread.start();
}
```

*Figura 7- Preluarea tranzacţiilor din Firestore pe un thread secundar*

```
public class Withdrawal {
    private String Location;
    private int Amount;
    private String Date;

    public Withdrawal() {
    }
    public Withdrawal(String location, int amount, String date) {
        Location = location;
        Amount = amount;
        Date = date;
    }

    public String getLocation() { return Location; }

    public void setLocation(String location) { Location = location; }

    public int getAmount() { return Amount; }

    public void setAmount(int amount) { Amount = amount; }

    public String getDate() { return Date; }

    public void setDate(String date) { Date = date; }

    @Override
    public String toString() {
        final StringBuffer sb = new StringBuffer("Withdrawal{");
        sb.append("Location='").append(Location).append('\'');
        sb.append(", Amount=").append(Amount);
        sb.append(", Date='").append(Date).append('\'');
        sb.append('}');
        return sb.toString();
    }
```

*Figura 8 - Exemplificare de clasă corespondentă tabelelor din baza de date*

```
    BottomNavigationView bottomNavigationView=findViewById(R.id.bottom_navigation);
    bottomNavigationView.setOnNavigationItemSelectedListener(navListener);

    getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,new HomeFragment()).commit();
}

private BottomNavigationView.OnNavigationItemSelectedListener navListener=
        new BottomNavigationView.OnNavigationItemSelectedListener(){
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                Fragment selectedFragment=null;

                switch (item.getItemId()){
                    case R.id.bottom_home:
                        selectedFragment = new HomeFragment();
                        break;
                    case R.id.bottom_transactions:
                        selectedFragment = new TransactionsFragment();
                        break;
                    case R.id.bottom_pay:
                        selectedFragment = new PaymentsFragment();
                        break;
                    case R.id.bottom_more:
                        selectedFragment = new MoreFragment();
                        break;
                }

                getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,selectedFragment).commit();

                return true;
            }
        };
```

*Figura 3 - Schimbarea fragmentelor în urma declanşării butoanelor din meniu*

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="60dp">

    <View
        android:id="@+id/view"
        android:layout_width="357dp"
        android:layout_height="58dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:background="@drawable/item_background"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/imageView3"
        android:layout_width="52dp"
        android:layout_height="48dp"
        android:layout_marginStart="24dp"
        android:src="@drawable/withdrawal2"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.508" />

    <TextView
        android:id="@+id/withdrawal_location"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sucursala"
        android:textSize="18sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@+id/imageView3"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/withdrawal_sum"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="500"
        android:textSize="16sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toStartOf="@+id/textView7"
        app:layout_constraintHorizontal_bias="0.836"
        app:layout_constraintStart_toEndOf="@+id/withdrawal_location"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView...>

    <TextView...>


</androidx.constraintlayout.widget.ConstraintLayout>
```

*Figura 4 - Fisier XML pentru adaptor personalozat*

```java
public class DepositAdapter extends ArrayAdapter<Deposit> {
    private Context context;
    private List<Deposit> depositList;
    private LayoutInflater inflater;
    private int resource;

    public DepositAdapter(@NonNull Context context, int resource,
                          @NonNull List<Deposit> objects, LayoutInflater inflater) {
        super(context, resource, objects);
        this.context = context;
        this.depositList = objects;
        this.inflater = inflater;
        this.resource = resource;
    }

    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
        View view= inflater.inflate(resource,parent, attachToRoot: false);

        Deposit deposit=depositList.get(position);
        if(deposit!=null){

            TextView depositname=view.findViewById(R.id.tv_depositname);
            TextView depositsum=view.findViewById(R.id.tv_depositsum);
            TextView depositrate=view.findViewById(R.id.tv_depositrate);
            TextView deposittimeleft=view.findViewById(R.id.tv_deposittimeleft);

            depositname.setText(deposit.getName());
            depositsum.setText(String.valueOf(deposit.getAmount()));
            depositrate.setText(String.valueOf(deposit.getInterestRate()));
            deposittimeleft.setText(deposit.getTimeLeft() +" luni");
        }
        return view;
    }
```

*Figura 5 - Clasă Java cu scopul implementării unui adaptor personalizat*

*Figura 6 - Formular pentru adăugarea unui depozit cu actualizarea în baza de date*

```
private void setUpPieChartMyPh(MyPh myph) {
    List<PieEntry> pieEntries=new ArrayList<>();
    for(int i=0;i<10;i++)
    {
        if(myph.grupValori[i]!=0) {
            pieEntries.add(new PieEntry(myph.grupValori[i], myph.etichete[i]));
        }
    }
    PieDataSet dataSet=new PieDataSet(pieEntries, label: "");

    dataSet.setColors(ColorTemplate.COLORFUL_COLORS);
    PieData data=new PieData(dataSet);

    PieChart chart=(PieChart)findViewById(R.id.grafic1);
    chart.setData(data);
    chart.invalidate();

}
```

*Figura 7 – Grafic Pie*

```
private void cautaDupaDescriere(View view){

    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String query) { return false; }

        @Override
        public boolean onQueryTextChange(String newText) {

            cuvantCautat=newText;
            List<Transaction> transactiiDupaDescriere= new ArrayList<>();

            for(Transaction transaction:HomeFragment.transactionList){
                if(transaction.getDescription().toLowerCase().contains(newText.toLowerCase())){

                    if(filtreSelectate.contains("toate")){
                        transactiiDupaDescriere.add(transaction);
                    }
                    else{
                        for(String filter:filtreSelectate){
                            if(filter.equals("facturi")){
                                if(transaction.getIsUtilityBill())
                                {
                                    transactiiDupaDescriere.add(transaction);
                                }
                            }
                            if(filter.equals("cheltuieli")){
                                if(transaction.getAmount()<0)
                                {
                                    transactiiDupaDescriere.add(transaction);
                                }
                            }
                            if(filter.equals("venituri")){
                                if(transaction.getAmount()>0)
                                {
                                    transactiiDupaDescriere.add(transaction);
                                }
                            }
                            if(filter.equals(transaction.getDate()))
                            {
                                transactiiDupaDescriere.add(transaction);
                            }

                        }
                    }
                }
            }

            setAdapter(view, transactiiDupaDescriere);

            return false;
        }
    });

}
```

*Figura 8 - Funcţie aferentă modulului de filtrare*

```
private void genereazaFisierExcel(List<Transaction> transactionsWithinDates) {
    HSSFWorkbook hssfWorkbook = new HSSFWorkbook();
    HSSFSheet hssfSheet=hssfWorkbook.createSheet();

    HSSFRow row = hssfSheet.createRow( rownum: 0);
    HSSFCell cell1= row.createCell( column: 0);
    cell1.setCellValue("Date");
    HSSFCell cell2= row.createCell( column: 1);
    cell2.setCellValue("Description");
    HSSFCell cell3= row.createCell( column: 2);
    cell3.setCellValue("From/To");
    HSSFCell cell4= row.createCell( column: 3);
    cell4.setCellValue("Amount");
    for (int i=0;i<transactionsWithinDates.size()-1;i++){
        HSSFRow row_i = hssfSheet.createRow( rownum: i+1);
        HSSFCell cell1_i= row_i.createCell( column: 0);
        cell1_i.setCellValue(transactionsWithinDates.get(i+1).getDate());
        HSSFCell cell2_i= row_i.createCell( column: 1);
        cell2_i.setCellValue(transactionsWithinDates.get(i+1).getDescription());
        HSSFCell cell3_i= row_i.createCell( column: 2);
        String beneficiarOrExpeditor;
        if(transactionsWithinDates.get(i+1).getAmount()<0){
            beneficiarOrExpeditor=transactionsWithinDates.get(i+1).getCardTo();
        }
        else{
            beneficiarOrExpeditor=transactionsWithinDates.get(i+1).getCardFrom();
        }
        if(beneficiarOrExpeditor.contains("p"))
        {
            cell3_i.setCellValue(HomeFragment.providersList.get(beneficiarOrExpeditor).getName());
        }
        else if(beneficiarOrExpeditor.contains("c")) {
            String user = HomeFragment.cardList.get(beneficiarOrExpeditor).getOwner();
            cell3_i.setCellValue(HomeFragment.userList.get(user).getFirstName());
        }
        HSSFCell cell4_i= row_i.createCell( column: 3);
        cell4_i.setCellValue(transactionsWithinDates.get(i+1).getAmount());
    }
    try{
        if(!filePath.exists()){
            filePath.createNewFile();
        }
        FileOutputStream fileOutputStream=new FileOutputStream(filePath);
        hssfWorkbook.write(fileOutputStream);

        if(fileOutputStream!=null){
            fileOutputStream.flush();
            fileOutputStream.close();
        }
    }
    catch(Exception e){
        e.printStackTrace();
    }
```

*Figura 9 - Funcția de generare a fișierului EXCEL*