Android aplikacija za praćenje i planiranje kućnog proračuna

Maričević, Josip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

ANDROID APLIKACIJA ZA PRAĆENJE I PLANIRANJE KUĆNOG PRORAČUNA

Diplomski rad

Josip Maričević

Osijek, 2022

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 05.12.2022.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Josip Maričević
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina	D-1141R, 22.10.2020.
OIB studenta:	55078689260
Mentor:	Prof. dr. sc. Krešimir Nenadić
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Alfonzo Baumgartner
Član Povjerenstva 1:	Prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 2:	Dr. sc. Hrvoje Leventić
Naslov diplomskog rada:	Android aplikacija za praćenje i planiranje kućnog proračuna
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Kratko opisati načine planiranja i praćenja kućnog proračuna. Primijeniti navedene postupke u funkcionalnostima aplikacije za Android platformu. Opisati postupak izrade aplikacije kao i izrađenu aplikaciju. Modelirati i izraditi bazu podataka za potrebe pohrane svih podataka koji su potrebni za rad aplikacije. Tema rezervirana: Josip Maričević
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina

Datum prijedloga ocjene od strane mentora:	05.12.2022.
Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 05.01.2023.

Ime i prezime studenta:	Josip Maričević
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1141R, 22.10.2020.
Turnitin podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom**: Android aplikacija za praćenje i planiranje kućnog proračuna**

izrađen pod vodstvom mentora Prof. dr. sc. Krešimir Nenadić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.

Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja __Josip Maričević_____, OIB: __55078689260_____, student/ica Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek na studiju _Diplomski sveučilišni studij Računarstvo_, kao autor/ica ocjenskog rada pod naslovom: _Android aplikacija za praćenje i planiranje kućnog proračuna**,**

dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).
Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor/ica ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti
b) studentima/icama i djelatnicima/ama ustanove
c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

__Osijek, 05.01.2023._____ _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

SADRŽAJ

1. UVOD

Poznato je da je ponekada jako teško pratiti i držati se svoga novčanog proračuna. No to je ključno za postizanje financijskog uspjeha. Naročito ako je to prvi susret s problemima vođenja proračuna, i želi se znati koliko novca se unese ili iznese iz kućanstva u nekom određenom periodu, ili koji dio proračuna se izdvaja za određene kategorije proizvoda.

Kako se tehnologije s vremenom razvijaju sve brže, tako je došlo i do pojave raznih alata koji omogućuju jednostavno praćenje proračuna. Najjednostavnija i najbrža opcija će uvijek biti pametni telefoni koje korisnici imaju uz sebe, i svakoj situaciji, čak i nakon izlaska iz trgovine, može se na brz i efikasan način zabilježiti sve promjene u proračunu.

Budget Manager Android aplikacija omogućuje jednostavno i pouzdano praćenje financijskog proračuna, brzo i efikasno praćenje i uređivanje kućanstva gdje god se nalazili. Kako bi aplikacija bila realizirana, potrebno je pronaći rješenje za neke zahtjevnije probleme, kao što su; omogućiti korisniku registraciju i prijavu u aplikaciju, na jednostavan i pregledan način prikazati stanje i prethodne promjene u proračunu, i omogućiti jednostavan unos novih ili izmjenu postojećih transakcija ili promjena u proračunu. Te pored svega toga, sve podatke koje korisnik unese, spremiti u nekom udaljenom servisu za pohranu podataka, kako bi se korisniku omogućio pristup podacima s većine Android uređaja u bilo kojem trenutku.

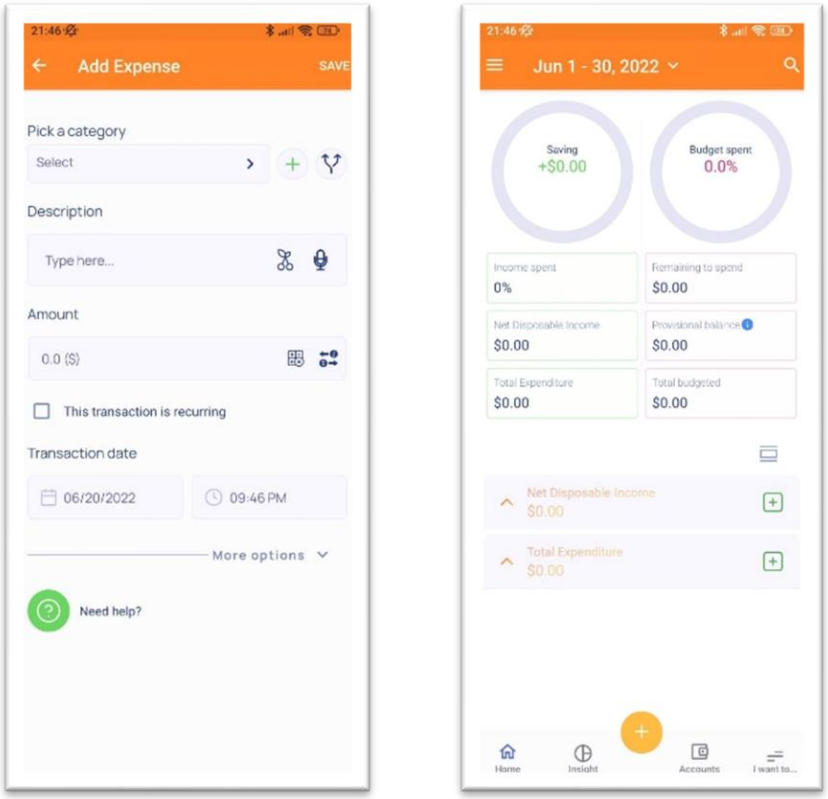
U drugom poglavlju dan je pregled nekih postojećih rješenja za ovaj problem, te njihovih prednosti i nedostataka. Zatim je rečeno nešto o Android operacijskom sustavu, njegovim prednostima i o tome zašto se u ovom diplomskom radu koristi Android. Nakon toga je rečeno nešto o izradi aplikacije, Android Studio razvojnom alatu te ostalim alatima i tehnologijama koje su korištene pri izradi Android aplikacije. Zatim je rečeno nešto o problemu pri izradi aplikacije, dani su primjeri rješenja koji postoje, te koje rješenje je korišteno u ovome radu i zašto.

2. PREGLED POSTOJEĆIH RJEŠENJA

U ovome poglavlju dani su neki primjeri već postojećih rješenja, odnosno Android aplikacija s Google Play-a koje nude slične mogućnosti, dan je kratak opis i navedene su neke od mogućnosti i funkcionalnosti koje aplikacija nudi. [1]

2.1. iSaveMoney

Ova Android aplikacija svojim korisnicima omogućava jednostavno planiranje i praćenje proračuna za neko prethodno određeno vrijeme. Također nudi i dodavanje cilja, odnosno nekog iznosa koji se želi steći ili uštedjeti kroz zadano vrijeme, te praćenje napretka, što se vidi na slici 2.1.

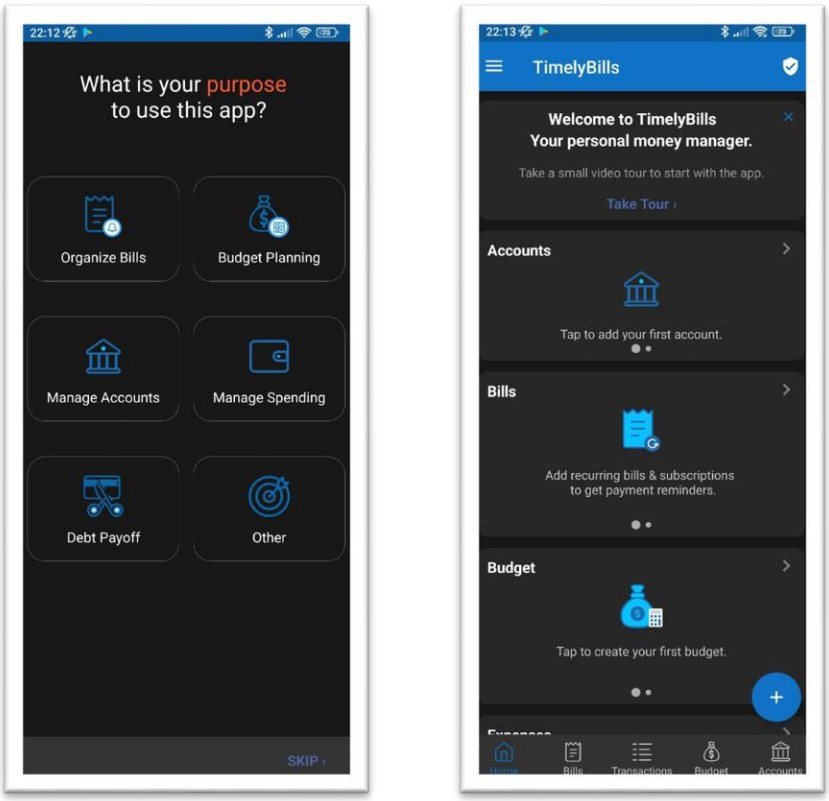


Slika 2.1 *Prikaz iSaveMoney Android aplikacije*

2.2. TimelyBills

TimelyBills je dobra proračunska aplikacija za praćenje potrošnje i troškova. TimelyBills je dobar financijski kalkulator, a jednostavan za korištenje i omogućuje jednostavno upravljanje

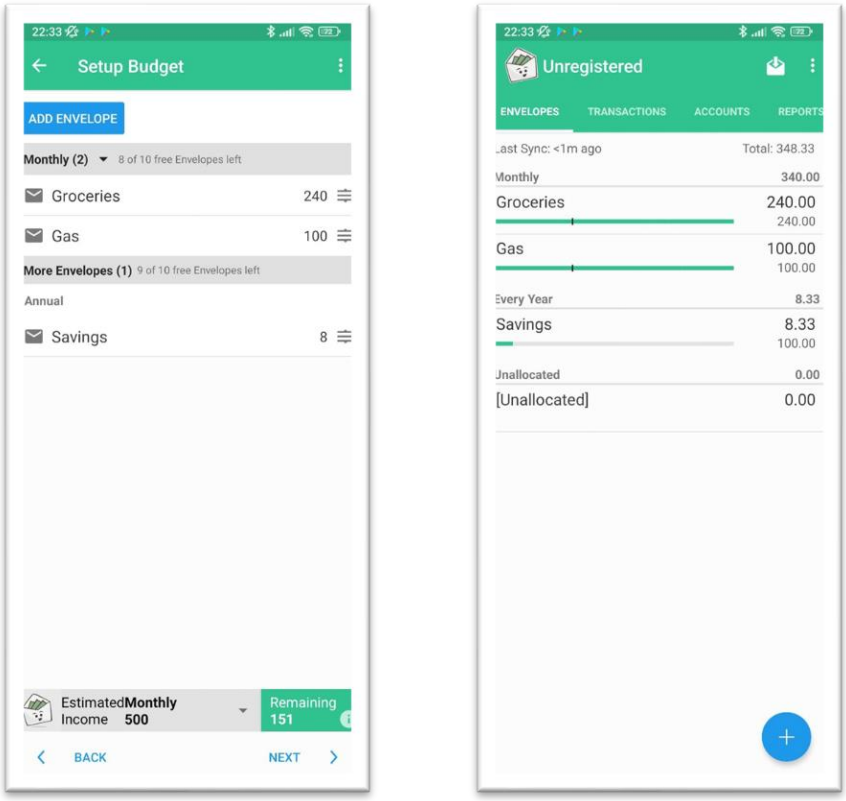
novcem. TimelyBills je financijski kalkulator koji će pomoći u planiranju proračuna, bilježenju transakcija, predviđanju proračuna i uštedi novca. Dobar je alat za organizatore plaćanja računa i proračuna. Slika 2.2. prikazuje izbornik koji prikazuje alate za kao planiranje otplate duga, praćenje duga, ali i kao aplikaciju za plaćanje i aplikaciju za podsjetnike računa.



Slika 2.2 *Prikaz zaslona TimelyBills aplikacije*

2.3. Goodbudget

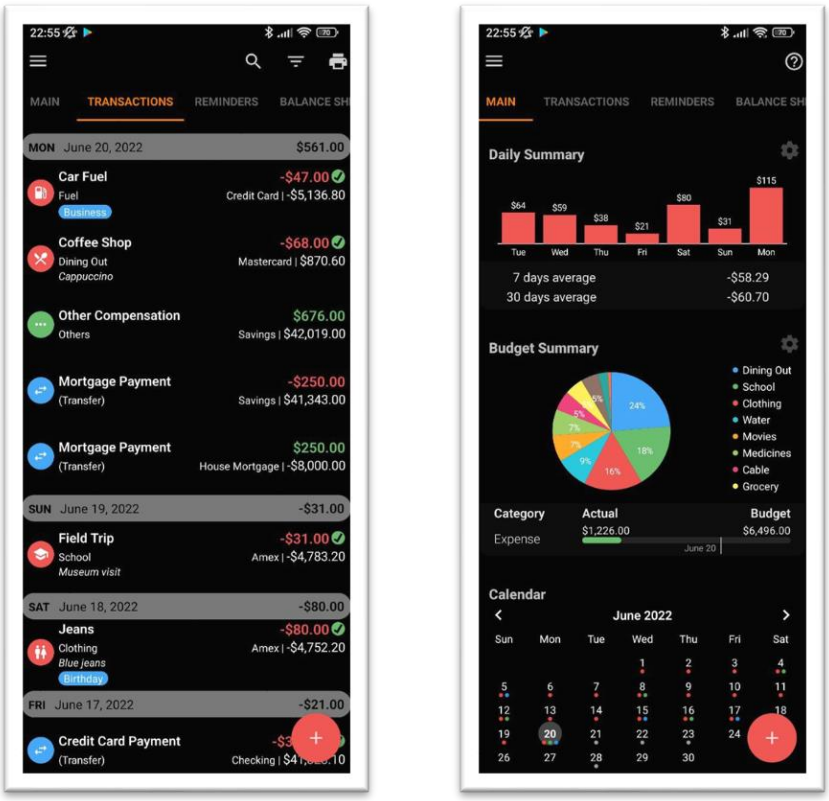
Goodbudget (bivši EEBA, Easy Envelope Budget Aid) je Android aplikacija koja se koristi kao alat za upravljanje novcem i praćenje troškova koja je izvrsna za planiranje kućnog proračuna. Ovaj upravitelj osobnih financija virtualno je ažuriranje starih pismenih načina praćenja. Napravljen za jednostavno praćenje u stvarnom vremenu. Nudi sinkronizaciju kroz više platformi, Android, iPhone i web kako bi se proračun mogao dijeliti s partnerima za proračun. Slika 2.3. prikazuje potrošnju po kategorijama (engl. *Envelope).*



Slika 2.3 *Android aplikacija Goodbudget*

2.4. Bluecoins

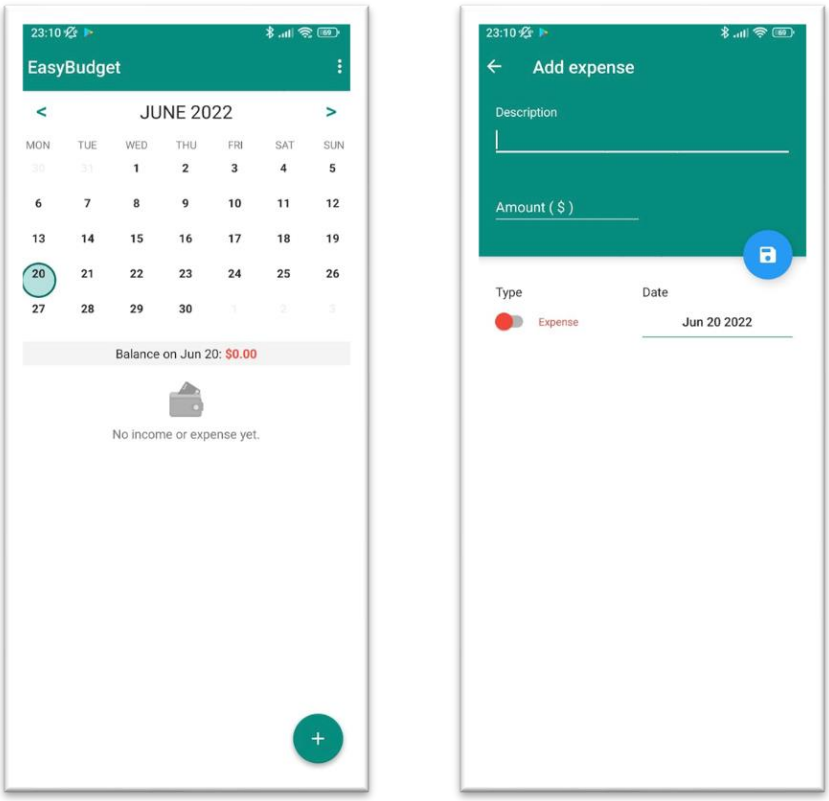
Bluecoins je jednostavna aplikacija za financije koja je izvrstan alat za praćenje troškova, proračuna i novca. To je aplikacija za izvješćivanje, analizu i upravljanje novcem, troškovima, prihodima i proračunom. Na slici 2.4. se vidi prikaz ugodnog korisničkog sučelja s kratkim pregledom u obliku grafikona. Može se koristiti za osobne financije, kao planer obiteljskog proračuna ili za male tvrtke. Omogućava generiranje izvješća o mjesečnim troškovima, upravljanje obiteljskim proračunom sa svojim partnerom ili izvoz financijskih zapisa u proračunske tablice/pdf.



Slika 2.4 *Demo zaslon Android aplikacije Bluecoins*

2.5. EasyBudget

EasyBudget je besplatna aplikacija za upravljanje novcem i financijsko praćenje proračuna koja objedinjuje sve financije. Ima mogućnost prikaza stanja računa, planiranja proračuna, praćenja troškova i plaćanja dugova, sve upravljanje novcem je na jednom mjestu, vidi na slici 2.5.



Slika 2.5 *EasyBudget Android aplikacija*

3. TEHNOLOGIJE I ALATI KORIŠTENI PRI IZRADI

Kako bi se ovo rješenje moglo realizirati, potrebno je izraditi dva odvojena dijela. Prvi dio je Android aplikacija, a drugi dio je udaljena pohrana podataka. Za tu svrhu koristi se *Firebase Firestore* platforma koja se ponaša kao *backendu*[1] aplikacije, te omogućuje jednostavno rukovanje podacima. Također se koristi i *Firebase Authentication* usluga koja je olakšala rukovanje i rad s korisnicama, te njihovo uređivanje, dodavanje i brisanje. Koristi se i *Firebase Console* usluga koja je omogućila jednostavnije rukovanje podacima. Za izradu Android aplikacije koristi se Android Studio IDE[2], te su korištene neke od tehnologija koje se koriste pri razvoju Android aplikacija, kao što su programski jezik Java i XML[3] jezik za označavanje podataka. Prije početka izrade Android aplikacije, korištenjem alata Figma, je kreiran dizajn aplikacije.

3.1. Android

Android je naziv za operacijski sustav koji se najčešće koristi za mobilne uređaje, te uređaje s dodirnim zaslonima kao što su pametni telefoni i tablet uređaji. Temelji se na Linux jezgri i drugih sličnih softvera. Android je razvijen od skupine programera pod nazivom *Open Handset Aliance* i komercijalno se sponzorira od strane Googlea. Prvi puta je predstavljen 2007. godine, prvi Android uređaj je HTC Dream i lansiran je 2008. godine.

Android je jako operacijski sustav koji podržava jako veliki broj aplikacija na pametnim telefonima. Hardver koji zahtjeva Android operacijski sustav je ARM[4] arhitekture, koja je otvorenog koda, što znači da je besplatan i svima dostupan.

---

[1] Backend je dio internet stranice koji korisnik ne vidi, on rukuje s podacima i odgovara na zahtjeve korisnika slanjem informacija

[2] IDE (engl. *Integrated Development Environment*) je softverska aplikacija koja nudi sveobuhvatne mogućnosti računalnim programerima za razvoj softvera

[3] XML (engl. *eXtensible Markup Language*) je jezik za označavanje koda, format datoteke za pohranu, prijenos i rekonstrukciju proizvoljnih podataka

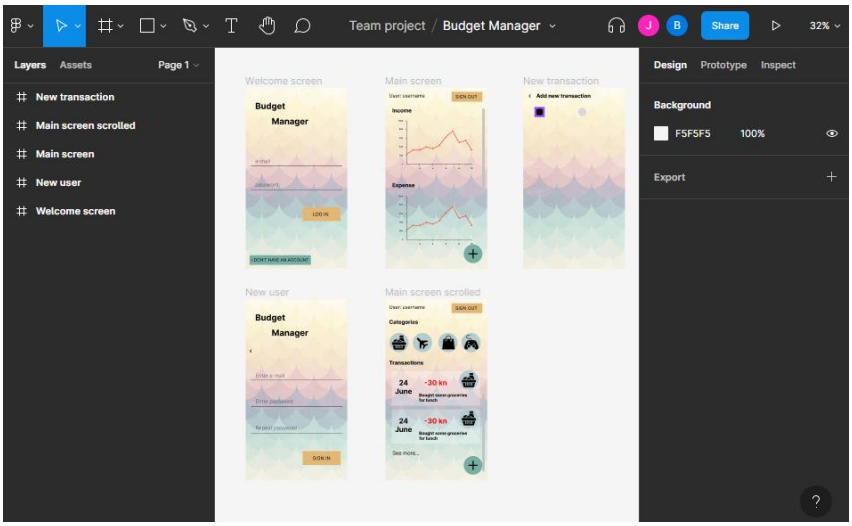
[4] ARM (engl. Advanced RISC Machines) je obitelj računalnih arhitektura smanjenog skupa instrukcija za računalne procesore, konfiguriranih za različita okruženja

3.2. Figma

Figma je alat koji se koristi za uređivanje i kreiranje vektorske grafike, te za izradu prototipa web i mobilnih aplikacija, kojim se prvenstveno može koristiti iz Internet preglednika, s mogućnošću korištenja i u izvan-mrežnom načinu kao Windows i macOS program, ili kao iOS i Android aplikacija.

Figmina *always online* (engl. uvijek online) priroda pruža neke od najmoćnijih alata. Jedan od njih je mogućnost uređivanja uživo te suradnja između korisnika u stvarnom vremenu. Svaki korisnik se u svakom trenutku može prijaviti i raditi promjenu na bilo kojemu dizajnu. Činjenica da su svi ovi dizajni spremljeni online, znači da se nikada ne mora brinuti o tome dali je neki korisnik ispao iz sinkronizacije. Tome pridonosi i značajka koja konstanto pohranjuje sve promjene u dizajnu, te u slučaju gubljenja veze s Figmom omogućuje da nastaviti raditi izvan mrežno, jer se tada sve promjene pohranjuju u internetskom pregledniku, te se ponovnom prijavom na Figmu, sve promjene ažuriraju i sinkroniziraju. [2]

Za izradu prototipa mogu se stvoriti veze i žarišne točke na dizajnu kako bi se moglo simulirati na koji način bi korisnik prolazio kroz to sučelje. Za fazu kodiranja Figma može generirati SVG[5] kod, CSS[6] i iOS i Android kod.



Slika 3.1 *Primjer projekta u Figma alatu*

[5] SVG (engl. Scalable Vector Graphics) je format vektorske slike temeljen na XML-u za definiranje dvodimenzionalne grafike, s podrškom za interaktivnost i animaciju

[6] CSS (engl. Cascading Style Sheets) je stilski jezik koji se koristi za opisivanje prezentacije dokumenta napisanog u označnom jeziku

3.3. Firebase

Firebase platformu je razvio Google za kreiranje mobilnih i web aplikacija. Osnovana je 2011. godite kao API[7] aplikacije za dopisivanje, te je početno bila samostalna kompanija, sve dok ju Google 2014. godine nije otkupio, te je sada vodeća Googleova platforma koja olakšava funkcionalnosti aplikacije kroz osigurane API-je.

Koncept *Firebasea* je jednostavan, pri izradi aplikacije pisane bilo kojim programskim jezikom ili korištenjem bilo kojega *frameworka*[8], jednostavnim implementiranjem *Firebasea*, aplikacija se pretvara u samostalnu aplikaciju kojoj nije potreban server, također uklanja se potreba upravljanja bazama podataka, jer to radi sama platforma te omogućuje rukovanje s podacima. Zaključno s time, rečeno je da implementacija *Firebasea* predstavlja uključivanje gotovog *backenda*[9] u aplikaciju koji pridodaje dinamičnosti te eliminira potrebu za pisanjem pozadinskog koda od nule, već daje potpuno funkcionalan i upotrebljiv kod.

Kada se govori o sigurnosti ima i izričito ugrađena sigurnosna pravila koja ga čine pouzdanim rukovateljem podataka i poslužitelja. Uz to, dobiven je zaštićeni *backend* korištenjem ovih pravila.

Slika 3.2 *Firebase platforma*

[7] API (Application Programming Interface) je softverski posrednik koji omogućuje dvjema aplikacijama međusobnu komunikaciju

[8] Alat koji pruža gotove komponente i rješenja koja su prilagođena kako bi se ubrzao razvoj

[9] Dio aplikacije koji nije vidljiv i zadužen je za pohranu i rukovanje podacima

3.3.1. Firestore

Prvi *Firebaseov* proizvod je *Firebase Realtime Database*, API koji sinkronizira sve aplikacijske podatke preko iOS, Android i web uređaja i sprema ih u Firebaseovom oblaku, te olakšava programerima suradnju u razvoju aplikacija u stvarnom vremenu. Njegov nasljednik je *Cloud Firestore,* baza podataka orijentirana na dokumente u stvarnom vremenu.

Firestore je visoko skalabilna usluga pohrane podataka koja koristi *NoSQL*[10] bazu podataka. Omogućava jednostavno strukturiranje podataka pomoću kolekcija i dokumenata izgradnjom hijerarhije kojem se olakšava pohranu, sinkronizaciju i dohvaćanje određenih podataka na mobilnim i web aplikacijama.

Firestore omogućava izgradnju stvarni bez-serverskih aplikacija te isporučuje SDK[11] koji dolazi na svim mobilnim i web platformama. Također omogućuje sinkronizaciju podataka preko svih uređaja, uz mogućnost obavještavanja o promjenama u podatcima što omogućava suradničko iskustvo i rad u stvarnom vremenu. [3]

3.3.2. Autorizacija

U 2014. Firebase je lansirao dva nova produkta, jedan od njih je bio i *Firebase Authentication. Firebase Authentication* cilja na jednostavnu izgradnju sigurnog sustava autorizacije, koji je jednostavan za korištenje svima, pa i krajnjim korisnicima. Obuhvaća sve načine autentifikacije korisnika, to jest podržava prijavu pomoću e-pošte i zaporke, telefonske autentifikacije i prijavu preko servisa kao što su Google, Twitter, GitHub i slično. Nudi fleksibilno korisničko sučelje koje olakšava upravljanje i praćenje korisnika. [4]

Uz *Firebase Authentication* postavljanje sustava autentifikacije može se svesti s nekoliko mjeseci rada, na samo desetak linija koda.

[10] NoSQL (engl. Non Structured Query Language) - pruža mehanizam za pohranjivanje i dohvaćanje podataka koji je modeliran na sredstva različita od tabličnih odnosa koji se koriste u relacijskim bazama podataka

[11] SDK (engl. Software Development Kit) - skup alata za razvoj softvera u jednom paketu koji se može instalirati

3.4. Android Studio

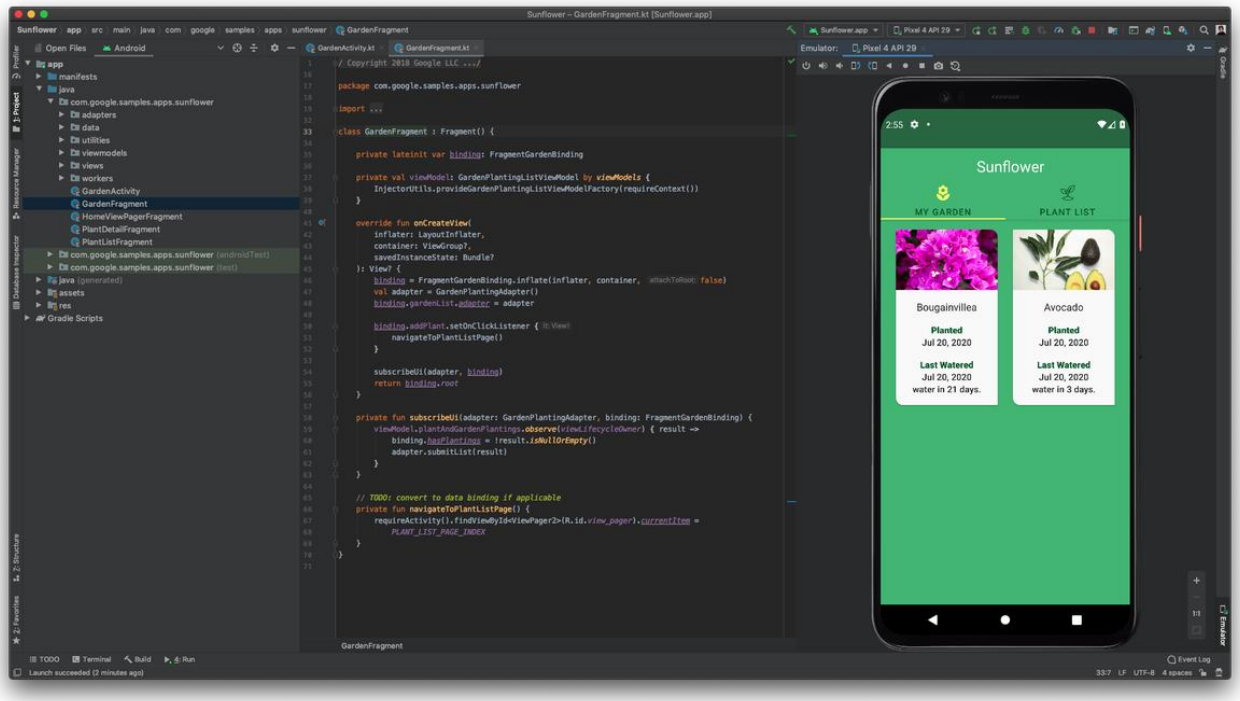
Android studio je službeno integrirano razvojno okruženje koje je razvio Google, za uređaje pogonjene Android operacijskim sustavom, izgrađen je na JetBrainsovom IntelliJ IDEA softveru i dizajniran prvenstveno za razvoj Android aplikacija. Na slici 3.3. prikazan izgled otvorenog projekta u Android Studio IDE-u. Besplatan je za korištenje i dostupan je na većini operacijskih sustava, uključujući, Windows, macOS i Linux. Prva stabilna verzija Android Studio razvojnog alata objavljena je u prosincu 2014. godine.

Prvenstveno je koristio Javu i C++ programske jezike, sve dok ih u svibnju 2019. godine nije zamijenio Kotlin, kao Googleov preferirani programski jezik.

Android Studio razvojnoi alat je temeljen na fleksibilnom *gradle*[12] sustavu izgradnje i ima ugrađen brz emulator bogat značajkama koji omogućuje pregled i testiranje aplikacije prilikom razvoja. Ima konsolidirano programsko okruženje u kojemu se mogu razvijati aplikacije za sve Android uređaje. Nudi opsežne alate i *frameworke* za razvoj i testiranje, te podržava C++ programski jezik i NDK[13] set alata koji omogućavaju implementaciju i korištenje biblioteka drugih programskih jezika kao što su C i C++. Pruža ugrađenu podršku za *Google Cloud Platform*, tj. *Firebase*.

[12] Gradle - alat za automatizaciju izrade za višejezični razvoj softvera

[13] NDK (engl. Native Development Kit) - omogućuje da se kod napisan u C/C++ može kompajlirati u ARM ili x86 izvorni kod



Slika 3.3 *Izgled Android Studio IDE-a s otvorenim emulatorom*

3.5. XML i layouti

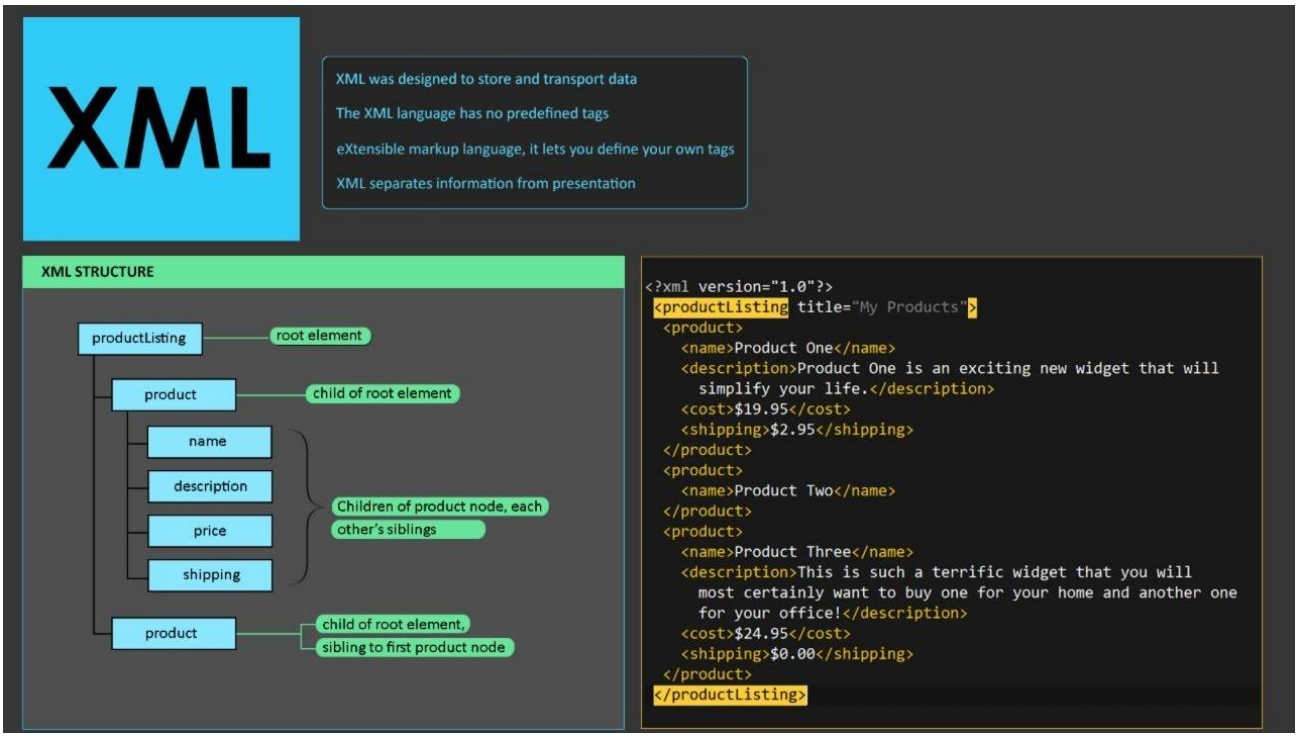
XML je jezik koji se koristi za pisanje oznaka, jako sličan HTML[14]-u. Glavna razlika je da se kod HTML-a moraju koristiti njegove predefinirane oznake, a XML nudi mogućnost samostalnog definiranja i dodavanja novih oznaka koje su dizajnirane prema potrebama.

Moćan je način pohranjivanja podataka koji se može lako pohraniti, pretražiti ili dijeliti. Najvažnija prednost mu je to da je temeljni XML format standardiziran, što znači da se XML datoteke lako dijele i prenose preko različitih sustava ili platformi, bilo to lokalno ili putem interneta, primatelj i svi drugi korisnici mogu jednostavno analizirati i pristupiti njegovim podacima zbog standardizirane XML sintakse.

Uglavnom se u Androidu XML koristi za implementaciju podataka vezanih za korisničko sučelje. Korisničko sučelje u Android aplikacijama izgrađeno je korištenjem hijerarhije različitih glavnih layouta, te objekata, koji se nazivaju *widgeti*. *Layouti* su *ViewGroup* objekti koji predstavljaju

[14] HTML (engl. HyperText Markup Language) - standardni označni jezik za dokumente dizajnirane za prikaz u web pregledniku

spremnike za widgete. *Widgeti* predstavljaju objekte za prikaz, kao što su gumbi, tekstualni okviri za prikaz i unos podataka i slično. Na slici 3.4. prikazana je struktura i primjer XML koda.



Slika 3.4 *Struktura i primjer XML zapisa*

3.6. Java

Java je objektno orijentirani programski jezik visoke razine, baziran na klasama, koji je dizajniran tako da ima što manje ovisnosti o implementacijama. Ovaj programski jezik omogućuje svim programerima da samo jednom pišu kod, a koriste ga svugdje - *WORA* (*engl. Write Once, Run Anywhere*), što znači da se kod pisan u Javi može koristiti i pokrenuti na bilo kojoj platformi koja podržava Javu, bez potrebe za ponovnim prevođenjem. Java aplikacije se obično prevode u bajt-kod koji se može pokrenuti na svim Java virtualnim strojevima, bez obzira na temeljnu arhitekturu računala. Sintaksa jave je jako slična onoj koju imaju C i C++, ali za razliku od njih, ima puno manje nisko razinskih funkcionalnosti i karakteristika, pa se zato smatra programskim jezikom više razine. [5]

Od 2019. godine, Java je jedan od najpopularnijih programskih jezika po broju korištenja prema statistici stranice GitHub, naviše za svrhu izrade klijent-poslužitelj web aplikacija, s prijavljenih 9 milijuna razvojnih programera.

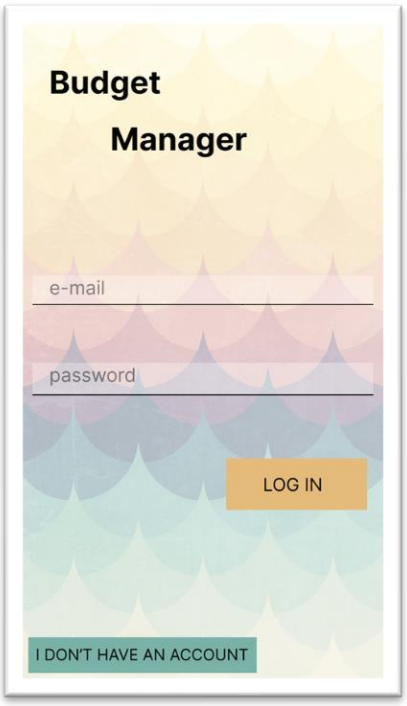
4. TEHNIČKI OPIS I IDEJA RJEŠENJA

U ovom poglavlju je ukratko objašnjeno na koji način je zamišljena izrada rješenja aplikacije, uključujući njen dizajn izrađen pomoću alata Figma, te na koji način je strukturiran kod aplikacije, odnosno kako su podijeljene određene funkcionalnosti.

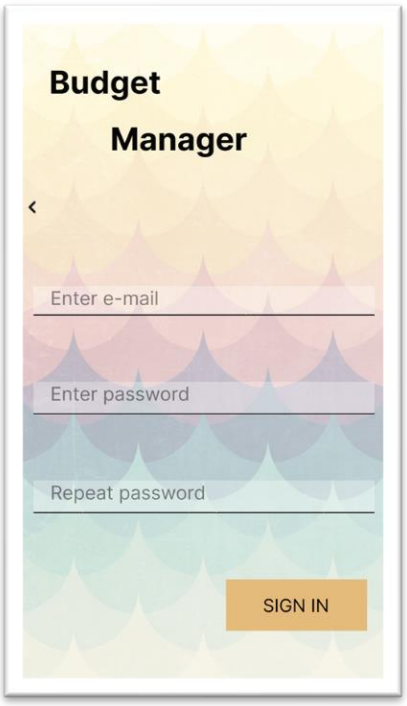
4.1. Dizajn prototipa

4.1.1. Početni zaslon i prijava korisnika

Prije korištenja aplikacije, korisnik mora imati mogućnost registracije ili prijave u sustav. Na slici 4.1. prikazan je izgled početnog zaslona gdje korisnik može unijeti svoje podatke, ili pritisnuti tipku na dnu u slučaju da nema račun, što će ga odvesti na zaslon na slici 4.2. gdje može unijeti podatke za registraciju.



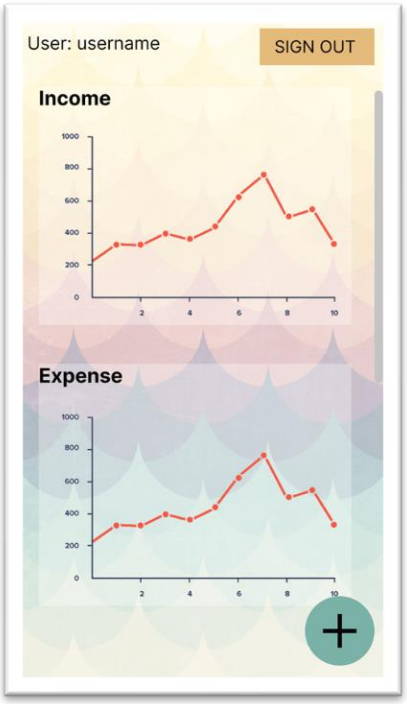
Slika 4.1 *Dizajn početne stranice za prijavu korisnika*



Slika 4.2 *Dizajn zaslona za registraciju novih korisnika*

4.1.2. Glavni zaslon

Nakon prijave, korisnik je odveden na glavni zaslon aplikacije, na kojemu su prikazani svi njegovi podaci, uključujući korisničko ime. Pokraj imena je tipka kojom se korisnik može odjaviti, a ispod su prikazani podaci o provedenim transakcijama, uključujući dva grafa koji prikazuju povijest prihoda i rashoda u određenom periodu, popis kategorija po kojima su sortirane transakcije, te zadnjih nekoliko transakcija.



Slika 4.3 *Prikaz grafova na glavnom zaslonu*

Slika 4.4 *Prikaz kategorija i transakcija na glavnom zaslonu*

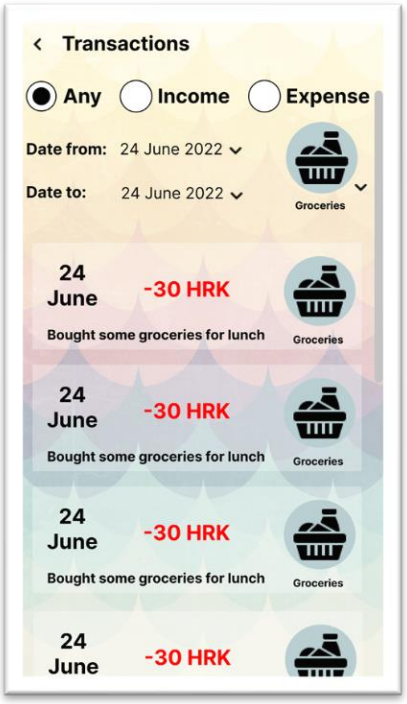
Na slikama 4.3 i 4.4 se vidi dizajn prototipa početnog zaslona te na koji način su prikazani podaci o korisniku i transakcijama. Za svaku transakciju navedeni su njeni podaci, što uključuje datum, vrijednost, opis i kategoriju. Također se vidi i tipka u donjem desnom uglu zaslona koja vodi na slijedeći zaslon i omogućuje korisniku unos novih transakcija.

4.1.3. Pregled i dodavanje transakcija

Pritiskom na tipku za dodavanje novih transakcija, otvara se novi zaslon, u kojemu je moguće dodavanje nove transakcije. Na slici 4.5 se vidi kako izgleda taj zaslon i što sve korisnik treba unijeti kako bi dodao novu transakciju. S obzirom na to da će tipka za dodavanje novih transakcija već imati ponuđen odabir dali se dodaje prihod ili rashod, na ovom zaslonu neće biti potrebe ponovno odabrati vrstu transakcije, ali je svakako moguće promijeniti. Nakon toga će korisnik moći odabrati kategoriju transakcije, unijeti količinu te po potrebi dodati opis.

Na slici 4.6 se je dan prikaz svih transakcija po određenom filteru. Korisnik ima mogućnost odabrati koju vrstu transakcije želi pregledati, kategoriju kojoj pripada ta transakcija te datum kada je odrađena. Ispod će biti prikazane sve transakcije koje pribadaju odabranom filteru, kako je to prikazano na slici.

Slika 4.5 *Dizajn zaslona za dodavanje novih transakcija*



Slika 4.6 *Dizajn zaslona za prikaz i pretragu transakcija*

4.2. Opis programskog rješenja

Programski kod aplikacije je zamišljen tako da bude podijeljen u tri skupine, fragmenti, aktivnosti i servisi, odnosno kontroleri koji će obavljati akcije u pozadini i neće biti vidljivi korisniku.

Prvo su izrađeni XML *layouti* po prethodno objašnjenoj figmi, što uključuje kreiranje i konfiguriranje *widgeta* koji su korišteni za interakciju s korisnicima. Točnije *widgeti* se koriste za prikaz podataka, unos novih podataka te će omogućuju kretanje kroz aplikaciju. Izrada *layouta* također uključuje i određivanje rasporeda i međusobnih položaja widgeta u odnosu na njihove roditelje i ostale elemente.

Nakon izrade XML-a, kreirani su fragmenti, odnosno njihove klase, te su prvo povezane s prethodno kreiranim layoutima. Zatim, kako bi se mogli koristiti elementi na layoutu koji koristi taj fragment, dohvaćene su reference za svaki od elemenata, kako bi se s njima moglo nešto konkretno raditi, na primjer postaviti vrijednost widgeta na nešto, pročitati upisanu vrijednost ili pratiti akciju kao što je klik. Unutar fragment klasa je također odrađena logika za postavljanje vrijednosti u te widgete, jer nakon dobivanja podatka s *backedna*, mora ih se negdje i prikazati.

Aktivnosti su zadužene za nešto kompleksnije radnje, kao što je prikazivanje i izmjena prikazanih fragmenata. Svaka cjelina ima svoju aktivnost, te fragment za svaku nižu i manju odgovornost. Jedan primjer je aktivnost zadužena za sustav autorizacije. Ta aktivnost ima dva fragmenta, jedan za prijavu, drugi za registraciju novih korisnika. Dok su na fragmentima samo metode za rukovanje i prikazom podataka, aktivnost ima ipak malo veću odgovornost, te je zadužena za komunikaciju između fragmenata i servisa.

Servisi imaju ulogu komunikacije za backendom te za razmjenu podataka s njime. Jedan od primjera je servis za autorizaciju. Servis za autorizaciju u sebi sadržava metode za prijavu i registraciju korisnika, koje za parametre primaju podatke koje su preko aktivnosti dostavljene od korisnikovog unosa.

5. IZRADA APLIKACIJE

U ovom poglavlju je objašnjena izrada aplikacije i svaki korak pri izradi aplikacije. Prvo je rečeno nešto kratko o *backendu* aplikacije i kako je strukturirana baza podataka. Zatim je u potpunosti objašnjen frontend, odnosno izrada aplikacije u Android studiju.

5.1. Backend – Firebase

Kao *backend*, u ovoj aplikaciji se koristi gotova Firebase platforma. Pomoću nje su jednostavno realizirane sve procedure potrebne za rukovanje podacima. Također *backend* je zadužen i za rukovanje s korisnicima, njihovu prijavu i kreiranje novih korisnika.

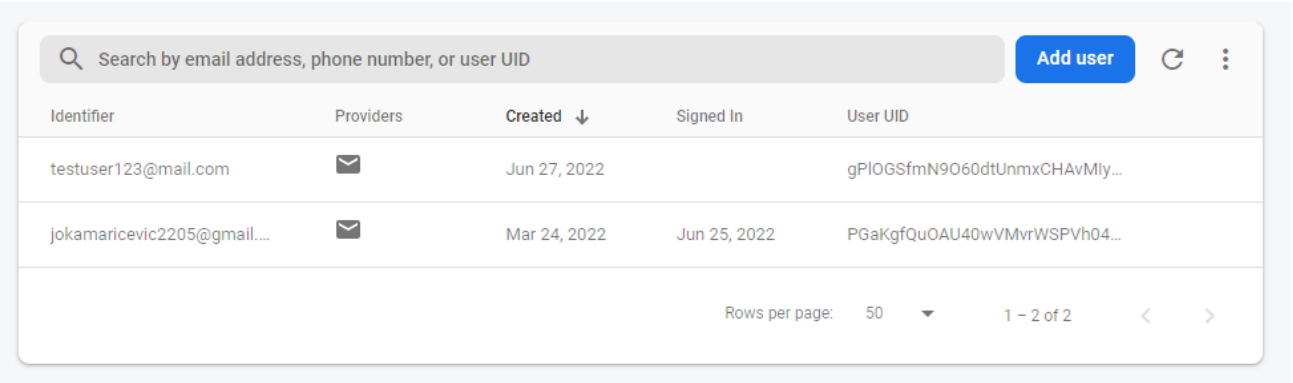
5.1.1. Autorizacija

Kako bi u aplikaciji mogli koristiti *Firebaseovu* autorizaciju, prvo je kreiran novi projekt na *Googleovoj Firebase* konzoli, te je zatim omogućena autorizacija. Zatim su nakon prijave u Android Studio s tim računom dodane sve potrebne ovisnosti, kako bi bilo moguće koristiti njegove funkcionalnosti.

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.4.2'
    implementation 'com.google.android.material:material:1.6.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'com.google.firebase:firebase-auth:21.0.6'
    implementation 'com.google.firebase:firebase-firestore:24.2.0'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    implementation 'com.github.clans:fab:1.6.4'
}
```

Slika 5.1 *Prikaz dodavanja Firebase-a u projekt*

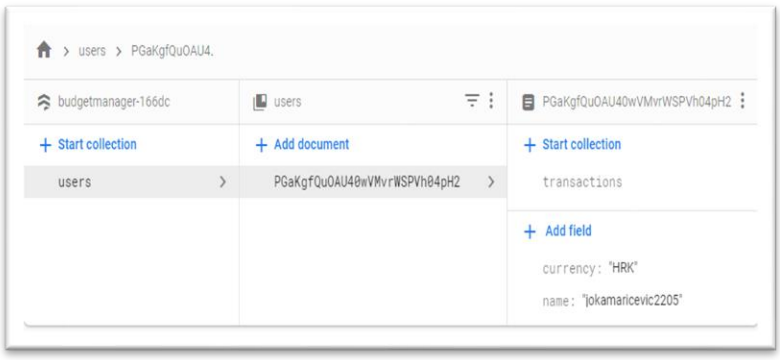
Nakon toga se u Firebase konzoli mogu pratiti i kontrolirati svi korisnici. Točna implementacija s Android strane je objašnjena kasnije pod poglavljem frontend.



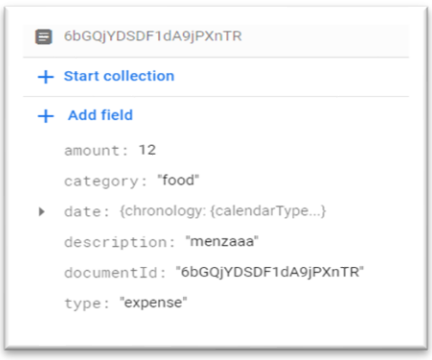
Slika 5.2 *Kontrola korisnika u Firebase konzoli*

5.1.2. Pohrana podataka – Firestore

Za korištenje *Firestorea*, prvo je dodan njegov *dependency*, što se može vidjeti gore na slici 5.1. Podaci su strukturirani tako da svaki korisnik ima svoj dokument. U tom dokumentu se nalazi kolekcija koja predstavlja popis svih transakcija i polja u kojima su spremljeni korisničko ime i oznaka valute u kojoj korisnik želi prikazati sve podatke. Svaka od transakcija ima svoj dokument koji sadrži podatke o količini, kategoriji transakcije kojoj ona pripada, datumu kada je provedena transakcija, tip transakcije, kratak opis i njenu identifikacijsku oznaku.



Slika 5.3 *Primjer dokumenta korisnika*



Slika 5.4 *Primjer polja transakcije*

5.2. Frontend – Android Studio

5.2.1. Layout-i

Prvo su izrađene XML datoteke koje predstavljaju izgled aplikacije. Layouti su izrađeni po Figmi koja je već prije pojašnjena. Kao glavni layout koristi se *ConstraintLayout*, jer pruža najviše mogućnosti i može se lako koristiti bez puno ugnježđivanja.

Za svaki element koji je korišten, postavljeni su određeni atributi kojima je opisano ponašanje i smještaj elementa. Prvi važan atribut je identifikacijska oznaka elementa, oznaka je postavljena kako bi se kasnije moglo pristupiti tom elementu i postaviti ili pročitati mu određene vrijednosti programski. Slijedeći važni atributi koji su postavljeni pri korištenju ConstraintLayouta su ograničenja (*engl. Constraints)* kojima je opisano na kojem položaju se nalaziti element u odnosu na roditelja ili neki drugi element unutar tog roditelja. Slijedeći su postavljeni visina i širina elementa, koji mogu po potrebi biti postavljeni na neku određenu dimenziju iskazanu u jedinici dp[15], ili se može postaviti tako da se prilagodi svom sadržaju ili roditelju. Na slikama 5.5 i 5.6 su prikazana dva *widgeta* s postavljenim atributima.

```
<Button
    android:id="@+id/btnRegister"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="32dp"
    android:layout_marginEnd="4dp"
    android:text="Register"
    app:layout_constraintEnd_toEndOf="@+id/etPass2"
    app:layout_constraintTop_toBottomOf="@+id/etPass3" />
```

Slika 5.5 *Dodavanje Button elementa u layout*

```
<ImageView
    android:id="@+id/back"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="32dp"
    android:src="@drawable/ic_baseline_arrow_back_24"
    app:layout_constraintBottom_toTopOf="@+id/etEmail2"
    app:layout_constraintStart_toStartOf="@+id/etEmail2"
    app:tint="#DA0909" />
```

Slika 5.6 *Dodavanje ImageView elementa u layout*

Najčešće korišten element je *TextView*. On je korišten za prikaz teksta na zaslonu. Ako je potrebno prikazati statičan tekst, potrebno mu je odmah predati tekst koji će biti prikazan, u suprotnom, ako je potrebno dinamičko ponašanje, dodana mu je identifikacijska oznaku, kako je navedeno.

[15] Veličina koja u Android sustavu označava mjeru za virtualni piksel

Za unos teksta, korišten je element *EditText*. Njemu su postavljeni svi gore navedeni atributi, uključujući i identifikacijsku oznaku kako bi mu mogli pristupiti kasnije te pročitati vrijednost koju je korisnik unio.

Button je element koji je korišten kao gumb, na kojega je također dodana identifikacijska oznaka kako bi mu se moglo pristupiti programski i postaviti *EventListener*-e, više o tome napisano je u idućem poglavlju.

Za prikaz slika se koristi ImageView element, u ovoj aplikaciji se ne prikazuju nikakve slike, ali se ovaj element koristi za prikaz ikone „natrag“ za povratak na prethodni zaslon.

Za prelazak na zaslon za dodavanje novih prihoda ili rashoda koriste se FloatingAction gumbovi. Ovdje je korištena vanjska implementacija tih gumbova i za to su dodani *dependency* u *gradle,* kako bi se mogli koristiti. Zatim je u layout dodan jedan *FloatingActionMenu* koji u sebi ima dva *FloatingActionButtona,* koji se prikazuju na dodir *FloatingActionMenuju.* Također dodani su svi navedeni atributi, uključujući i identifikacijsku oznaku.

Od elemenata je još koristen i *RecyclerView* element, on je korišten za prikaz dinamičkih lista s prilagodljivim unutarnjim pod elementima. Više o njemu je rečeno kod programske implementacije ovog elementa.

5.2.2. Fragmenti

Fragment predstavlja dio korisničkog sučelja aplikacije koji se može ponovno upotrijebiti. Fregment upravlja vlastitim izgledom, ima vlasti životni ciklus i može rukovati vlastitim ulaznim događajima. Fragment ne može postojati sam za sebe i mora biti prikazan pomoću drugog fragmenta ili aktivnosti. [6]

Kako bi fragment bio funkcionalan, prvo mu je dodjeljen njegov layout, za to je korišten *LayoutInflater*[16]. Nakon dodavanja *layouta,* dohvaćeni su svi elementi toga *layouta* pomoću metode *findViewById,* kojoj se kao parametar predaje ID, odnosno identifikacijska oznaka koja je prethodno spomenuta.

[16] LayoutInflater - klasa koja se koristi za instanciranje XML datoteke rasporeda u odgovarajuće objekte prikaza koji se mogu koristiti u Java programima

```
private void initViews(@NonNull View view) {
    TextView alertText = view.findViewById(R.id.alertText);
    EditText etEmail = view.findViewById(R.id.etEmail);
    EditText etPass = view.findViewById(R.id.etPass2);
    Button btnSignIn = view.findViewById(R.id.btnSignIn);
    btnSignIn.setOnClickListener(v -> {
        this.authActivity.hideKeyboard();
        String email = etEmail.getText().toString();
        String pass = etPass.getText().toString();
        if (email.isEmpty() || pass.isEmpty())
            alertText.setText("Please fill in all the fields!!!");
        else {
            alertText.setText("");
            this.authActivity.authService.signIn(this.authActivity, email, pass);

        }
    });
```

Slika 5.7 *Dohvaćanje elemenata i postavljanje EventListenera*

Nakon što su dohvaćeni svi elementi, na one elemente za koje je to potrebno je dodan osluškivač događaja. Na slici 5.7 je prikazano dodavanje *OnClickListenera* na gumb za prijavu korisnika. *OnClickListener* je metoda koja se poziva na dodir elementa kojemu je dodijeljena, to jest u prethodno spomenutom primjeru, ova metoda pokreće proces prijave korisnika.

Fragment je zadužen za prikaz podataka i razmjenu podataka između korisnika i servisa koji rukuju s tim podacima. Jedina funkcionalnost za koju je odgovoran fragment je prelazak s jednog fragmenta na drugi, primjer toga se vidi na slici 5.8, gdje se na gumb dodaje *OnClickListener*, koji će nakon dodira na gumb otvoriti novi fragment.

```
Button btnGoToRegister = view.findViewById(R.id.btnGoToRegister);
btnGoToRegister.setOnClickListener(v -> {
    Fragment registerFragment = new RegisterFragment();
    FragmentTransaction transaction = requireActivity().getSupportFragmentManager().beginTransaction();
    transaction.replace(R.id.authActivity, registerFragment);
    transaction.addToBackStack(null);
    transaction.commit();
});
```

Slika 5.8 *Prijelaz na drugi fragment nakon klika na gumb*

5.2.3. Aktivnosti

Aktivnost je zadužena za prikaz fragmenata, i u sebi sadrži veze na bazu podataka i sustav autorizacije. Kreirane su dvije aktivnosti, *AuthActvity* koja je zadužena za prikazivanje fragmenata vezanih za autorizaciju, i *MainActivity* aktivnost koja je zadužena za prikaz i uređivanje transakcija i ostalih podataka. Pokretanjem aplikacije, prvo se pokreće *AuthActivity*, koja odmah prikazuje fragment za prijavu korisnika. Na slici 5.9. je prikazano na koji način se otvara fragment za prijavu korisnika nakon pokretanja aplikacije.

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

}

private void openSignFragment() {
    Fragment signFragment = new SignFragment();
    FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
    transaction.replace(R.id.authActivity, signFragment);
    transaction.commit();
}
```

Slika 5.9 *Otvaranje fragmenta za prijavu nakon pokretanja aplikacije*

Prije otvaranja fragmenta za prijavu, inicijalizirane su varijable kojima je omogućen pristup bazi i servisu autorizacije, te se s inicijaliziranim servisom za autorizaciju prije prikazivanja fragmenta provjerava dali je korisnik od prije prijavljen u aplikaciju. U slučaju da je, prekida se funkcija i otvara se glavna aktivnost. Na slici 5.10 je prikazano na koji način se vrši provjera korisnika i prelaz na drugu aktivnost.

```
if(this.authService.getCurrentUser() != null ) {
    goToMainActivity(this.authService.getCurrentUser());
    return;
}
openSignFragment();
```

Slika 5.10 *Provjera korisnika i prelazan da glavnu aktivnost*

Nakon otvaranje glavne aktivnosti, inicijalizirane su varijable, odnosno reference na bazu i autorizacijski servis, na isti način kao i na prethodnoj aktivnosti. Kako su aktivnosti odgovorne za komunikaciju s *Firebaseom*, svaki fragment unutar neke aktivnosti, ima referencu te aktivnosti u sebi, kako bi mogao koristiti te definirane reference.

5.2.4. Autorizacija

Za potrebe autorizacije korisnika kreiran je servis *AuthService*, koji sadrži referencu na *Authentication* platformu, te metode za rad s korisnicima, kao što su *signInUser*, *createUser*, *signOutUser* i getCurrentUser. Za kreiranje korisnika, *createUser* metodi su proslijeđeni e-mail i zaporka koje je korisnik unio. Nakon kreiranja korisnika, pomoću reference na bazu kreiran je novi dokument koji predstavlja jednog korisnika, te su mu također kreirani polje koje sadrži njegovo korisničko ime, koje se dobila iz e-maila i kreirana mu je nova kolekcija koja predstavlja sve transakcije koje taj korisnik ima. Na slici 5.11. je prikazano na koji način se postavlja korisničko ime i prelazi na glavnu aktivnost nakon uspješne registracije korisnika.

```
this.mAuth.createUserWithEmailAndPassword(email, password)
        .addOnSuccessListener(r -> {
            FirebaseUser firebaseUser = r.getUser();
            this.db.collection( collectionPath: "users").document(firebaseUser.getUid()).get()
                    .addOnSuccessListener(documentSnapshot -> {
                        if(!documentSnapshot.exists()) {
                            Map<String, Object> name = new HashMap<>();
                            name.put( k: "name", firebaseUser.getEmail().split(String.valueOf('@'))[0]);
                            this.db.collection( collectionPath: "users").document(firebaseUser.getUid()).set(name);
                            authActivity.goToMainActivity(Objects.requireNonNull(this.mAuth.getCurrentUser()));
                        }
                    });
        })
```

Slika 5.11 *Postavljanje korisničkog imena i prelazak na glavnu aktivnost*

5.2.5. Pohrana podataka

Za pohranu podataka korištena je *Firebase Firestore* platforma. Za korištenje ove platforme, u aktivitijima je kreirana referenca na *Firestore* pod nazivom *db*. *Firestore* instanca se dohvaća pomoću naredbe *FirebaseFirestore.getInstance*. Za korištenje baze, prvo se dohvaća kolekcija, odnosno dokument kojega želimo pročitati. Na slici 5.12. prikazano je na koji način se dohvaća korisničko ime korisnika. Prvo je dohvaćena kolekcija *users*, zatim je dohvaćen dokument korisnika pod njegovim Uid-jem. Nakon čeka je iz toga dokumenta pročitano polje *name* pomoću naredbe *get*.

```
this.mainActivity.db.collection( collectionPath: "users")
        .document(this.mainActivity.authService.getCurrentUser().getUid()).get()
        .addOnSuccessListener(documentSnapshot -> {
            tvUserName.setText(documentSnapshot.get("name").toString());
        });
```

Slika 5.12 *Dohvaćanje korisničkog imena iz baze podataka*

Ovaj način je realizirano samo rukovanje s korisničkim imenom, a za CRUD[17] operacije s transakcijama, za te potrebe kreirana je *TransactionFirestoreManager* klasa, koja olakšava rad s transakcija. Ova klasa u sebi koristi instancu *CollectionReference* klase, koja je nakon instaciranja omogućila osnovne operacije nad transakcijama u samo par linija. Za primjer, na slici 5.13. je prikazano kako se dohvaća referenca na kolekciju i kako se na jednostavan način uz predavanje objekta transakcije on kreira i pohranjuje.

```
private  TransactionFirestoreManager(String userId) {
    this.firebaseFirestore = FirebaseFirestore.getInstance();
    this.transactionCollectionReference = this.firebaseFirestore
            .collection(COLLECTION_NAME)
            .document(userId)
            .collection(USER_COLLECTION_NAME);
    this.userId = userId;
}

public void createTransaction(Transaction transaction) {
    this.transactionCollectionReference.add(transaction);

}
```

Slika 5.13 *Dohvaćane reference i kreiranje nove transakcije*

[17] CRUD (engl. Create, Read, Update, and Delete) - stvaranje, čitanje, ažuriranje i brisanje, četiri su osnovne operacije trajne pohrane

6. ZAKLJUČAK

Upravljanje novcem nije najlakša radnja. Sada kada mnogi od nas više nemaju stanje na čekovnoj knjižici, praćenje troškova i praćenje bankovnog stanja može postati malo otežano. Poznavanje kamo ide naš novac prvi je korak prema razumijevanju i praćenju financija. Dali trošimo više nego zaradimo ili nam prerano ponestaje novca. Ovo su problemi s kojima se veliki broj ljudi kad-tad susreće.

U ovom diplomskom radu dano je rješenje za praćenje budžeta i svih financija u obliku Android aplikacije. Android aplikacija *BudgetManager* omogućuje praćenje i pohranu svake napravljene transakcije, uz mogućnost odabira je li transakcija prihod ili rashod, te kategorizaciju uz dodatne opise koja omogućava praćenje budžeta i pruža informacije o tome koliko novca i na što trošimo.

Uz lako praćenje budžeta, loša strana ove aplikacije je to što nakon svake obavljene transakcije, moramo ju provesti i pohraniti pomoću aplikacije, što neće uvijek biti lako održavati. Ono što bi riješilo i ovaj problem je povezivanje, odnosno integracija bankovnih računa u aplikaciju. Što bi omogućilo još jednostavnije praćenje budžeta uz još manje obaveze.

LITERATURA

[1] Primjeri rješenja aplikacija, dostupni na: https://play.google.com/store/search?q=budget+manager&c=apps/ [23. rujna 2022.]

[2] Figma - The modern interface design tool, dostupno na: https://www.figma.com/ui-design-tool/ [19. travnja 2022.]

[3] Cloud Firestore, dostupno na: https://firebase.google.com/docs/firestore [4. lipnja 2022.]

[4] Firebase Authentication, dostupno na: https://firebase.google.com/docs/auth [4. lipnja 2022.]

[5] Learn Java For Android App Development – A Complete Guide, dostupno na: https://www.geeksforgeeks.org/learn-java-for-android-app-development-a-complete-guide/ [13. svibnja 2022.]

[6] Application Fundamentals, dostupno na: https://developer.android.com/guide/components/fundamentals [13. svibnja 2022.]

SAŽETAK

Današnjim brzim rastom tehnologija, posebice pametnih uređaja kao što su tableti i telefoni, omogućeno je olakšano odrađivanje svakodnevnih poslova i zadataka. Najviše zato što jer na koji god problem naiđemo, već postoji nekakvo rješenje, jer je netko već imao takvih problema i za tu potrebu kreirana je mobilna aplikacija. Svaka osoba danas većinu vremena uz sebe ima pametni mobilni uređaj koji koristi z rješavanje svakodnevnih problema.

Odabran je problem praćenja financijskog budžeta. Za izradu rješenja ovog problema odabrana je platforma Android zbog njene rasprostranjenosti i jednostavnosti korištenja. Uvidom u već gotove primjere rješenja, odlučeno je napraviti aplikaciju koja omogućuje brzo i jednostavno rukovanje s transakcijama. Omogućena je registracija korisnika kako bi se budžet mogao neometano pratiti s više različitih uređaje u isto vrijeme.

Ključne riječi: aplikacija**,** Android, budžet, financije, java

ABSTRACT

Android aplication for monitoring and planning the home budget

Today's rapid growth of technologies, especially smart devices such as tablets and phones, has made it possible to perform everyday tasks and tasks more easily. Mostly for the reason that whatever problem we come across, there is already some kind of solution, because someone has already had such problems and a mobile application was created for that need. Today, every person has a smart mobile device with them most of the time, which they use to solve everyday problems.

The problem of monitoring the financial budget was chosen. To create a solution to this problem, the Android platform was chosen because of its prevalence and ease of use. By looking at already finished examples of solutions, it was decided to create an application that enables quick and easy handling of transactions. User registration is enabled so that the budget can be monitored without interruption from several different devices at the same time.

Keywords: application, Android, budget, finance, java