Aplikacija za praćenje i planiranje osobnih financija

Tometić, Vedran

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Vedran Tometić

APLIKACIJA ZA PRAĆENJE I PLANIRANJE OSOBNIH FINANCIJA

Diplomski rad

Pula, veljača 2021

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Vedran Tometić

APLIKACIJA ZA PRAĆENJE I PLANIRANJE OSOBNIH FINANCIJA

Diplomski rad

JMBAG: 0016112293, redovni student

Studijski smjer: Informatika

Predmet: Mobilne aplikacije

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: izv. prof. dr. sc. Siniša Sovilj

Pula, veljača 2021.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani **Vedran Tometić**, kandidat za **magistra informatike** ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da niti jedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, 2021. godine

IZJAVA

o korištenju autorskog djela

Ja, **Vedran Tometić** dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom **Aplikacija za praćenje i planiranje osobnih financija** koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____

Potpis

_____

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Pula, 11. ožujka 2020.

DIPLOMSKI ZADATAK

Pristupnik:	**Tometić Vedran (0016112293)**
Studij:	Sveučilišni diplomski studij Informatike
Naslov (hrv.):	**Aplikacija za praćenje i planiranje osobnih financija**
Naslov (eng.):	Application for personal budget tracking and planning
Opis zadatka:	Zadatak je izraditi Android mobilnu aplikaciju koja korisniku omogućava dodavanje osobnih prihoda i troškova na dnevnoj ili tjednoj bazi. Osim toga aplikacija omogućava izradu tjednih, mjesečnih i godišnjih planova za raspodjelu osobnih prihoda i troškova te generiranje i vizualizaciju tjednih, mjesečnih i godišnjih izvještaja o prihodima, troškovima i stanjima računa. Uz navedeno, aplikacija uz privolu korisnika može slati podsjetnike o nadolazećim troškovima. Aplikacija se treba temeljiti na tehnologijama: Java, HTML, CSS, Firebase. Izraditi potrebne UML dijagrame - use case, use sequence, opisati implementaciju te na kraju izraditi kratke korisničke upute. Istražiti mogućnosti i usporediti tehnologije vizualizacije podataka unutar mobilnih aplikacija.

Zadatak uručen pristupniku: 11. ožujka 2020.

Rok za predaju rada: 11. veljače 2021.

Mentor:

_____

doc.dr.sc. Siniša Sovilj

Komentor:

_____

dr.sc. Nikola Tanković

Sadržaj

1. Uvod

Vizualizacija podataka u današnjem svijetu, kada broj podataka konstantno raste predstavlja jedan od najvažnijih alata za izdvajanje informacija iz tih podataka. Značaj vizualizacije će u budućnosti vjerojatno samo rasti zato što će se i sam broj podataka konstantno povećavati. Mogućnost iščitavanja informacija iz vizualiziranih podataka će također biti sve važnija vještina jer će u nekim slučajevima upravo to predstavljati granicu između dobre i loše odluke u poslovanju.

Aplikacija *BudgettApp* nudi korisnicima mogućnost kreiranja vlastitih planova za raspolaganje kućnim budžetom te kreiranje izvještaja na temelju kojih korisnik može vidjeti razliku između planiranih i ostvarenih troškova i prihoda za odabrano razdoblje. Osim toga korisnik može generirati izvještaj kako bi usporedio ostvarene prihode i troškove. Cilj ove aplikacije je potaknuti korisnike na razmišljanje o vlastitim prihodima i troškovima te im prikazati važnost planiranja osobnog budžeta.

Ovaj diplomski rad sastoji se od sedam poglavlja, u drugom poglavlju teorijski se obrađuje pojam vizualizacije i neke od tehnika vizualizacije podataka. Osim toga upućuje se na ključnu razliku između podatka i informacije, nakon toga slijedi kratak pregled povijesnog korištenja vizualizacije i nekih područja na kojima se i danas koristi. Na kraju drugog poglavlja nalazi se pregled modernog pristupa vizualizaciji podataka. Treće poglavlje sužava rad na vizualizaciju podataka u mobilnim aplikacijama, gdje se prolazi kroz neke probleme s kojima se suočavaju sudionici koji sudjeluju u izradi mobilnih aplikacija kojima je jedna od funkcionalnosti i sama vizualizacija podataka. Nakon toga u okviru trećeg poglavlja obraduju se preporuke najbolje prakse za vizualizaciju podataka prilikom izrade mobilnih aplikacija i na kraju su prikazane neke od tehnologija za vizualizaciju podataka na mobilnim uređajima. Četvrto, peto i šesto poglavlje posvećeno je projektnoj dokumentaciji unutar koje se detaljno opisuje proces izrade aplikacije i gdje se nalaze korisničke upute za procese unutar aplikacije. Na samom kraju rada u poglavlju osam ponuđen je zaključak na temelju obrađene literature i izrađene aplikacije.

2. Vizualizacija podataka općenito

U današnjem vremenu Informacije su postale najvažniji resurs, kako u poslovanju tako i u politici i svim ostalim područjima ljudskog djelovanja. Još jedan resurs kojeg nemamo u neograničenim količinama je vrijeme. Izgubljeno vrijeme ne možemo vratiti, stoga je važno da donošenjem loših odluka ne gubimo taj dragocjeni resurs, što nas opet vraća na informacije. Prava informacija na pravom mjestu i u pravo vrijeme je ključna za donošenje dobre odluke. Na suprotnoj strani spektra nalazimo podatke kojih u današnjem vremenu ima mnogo i njihov broj svakodnevno raste, međutim kvalitetnom obradom podataka pretvaramo ih u Informacije čime njihova vrijednost drastično raste. Uočimo kao smo ovdje napravili razliku između podatka i informacije. Iako se često koriste kao sinonimi postoji razlika među njima.

Primjerice ako imamo podatak da će padati kiša to nam ne govori puno i nije od nekog značaja za nas, međutim ako imamo informaciju da će padati kiša danas od 14:00 do 16:00 to nam uvelike pomaže kod donošenja naše odluke.

„Poput podataka, informacije također predstavljaju svojstva predmeta i događaja, ali to čine kompaktnije i korisnije od podataka. Razlika između podataka i informacija je funkcionalna, a ne strukturna. Informacije se nalaze u opisima, i odgovaraju na pitanja koja započinju takvim riječima kao tko, što, kada, gdje i koliko.“ Ackoff i Russel (1989)

Prethodna razmišljanja dovode do pitanja gdje je vizualizacija podataka u svemu tome? Vizualizacija podataka je na neki način pretvaranje podataka u oblik iz kojega možemo izvući mnoge korisne informacije, kao što su trendovi rasta ili pada, grupe koje drastično odstupaju od ostatka uzorka i slično. Primjerice ako uzmemo podatke o ukupnom broju ljudi na Zemlji od 2000. godine do danas i vizualiziramo ga na načina da na os x postavimo godine, a na os y ukupan broj stanovnika puno je lakše uočiti trend rasta nego što bi to bilo gledanjem u neobrađene podatke. Unwin (2020) navodi kako vizualizacija podataka znači crtanje grafičkih prikaza za prikazivanje podataka. Ponekad se izvuče svaka podatkovna točka, kao u dijagramu raspršenja, ponekad se mogu prikazati statistički sažetci, kao u histogramu. Prikazi su uglavnom opisni, koncentrirani su na neobrađene podatke i jednostavne sažetke. Glavni je cilj vizualizirati podatke i statistiku, interpretirajući prikaze kako bi se dobile informacije.

Često čujemo kako slika vrijedi tisuću riječi, možemo reći kako to vrijedi i za podatke. Vizualizirani podatci su nam od puno većeg značaja u odnosu na neobrađene podatke ili podatke prezentirane u nekom drugom obliku. Sinar (2015:118) navodi kako su prednosti vizualizacije podataka sljedeće:

- Vizualizacija čini podatke dostupnima širokoj publici.

- Demokratizira pristup podatcima, interpretaciju i analizu oslanjajući se na naše subnacionalne vizualne vještine i koristeći uobičajene vizualne referente.

- Vizualizacija bez obzira na veličinu podataka na koju se primjenjuje je korisnija u odnosu na tekstualne ili tablične prikaze podataka.

- Omogućava otkrivanje odnosa koji bi inače ostali skriveni i to kroz svoj utjecaj na ljudsku spoznaju

- Vizualizacije podataka donose koristi za donošenje odluka, učenje i analitičko zaključivanje

2.1. Definicija i metode vizualizacije podataka

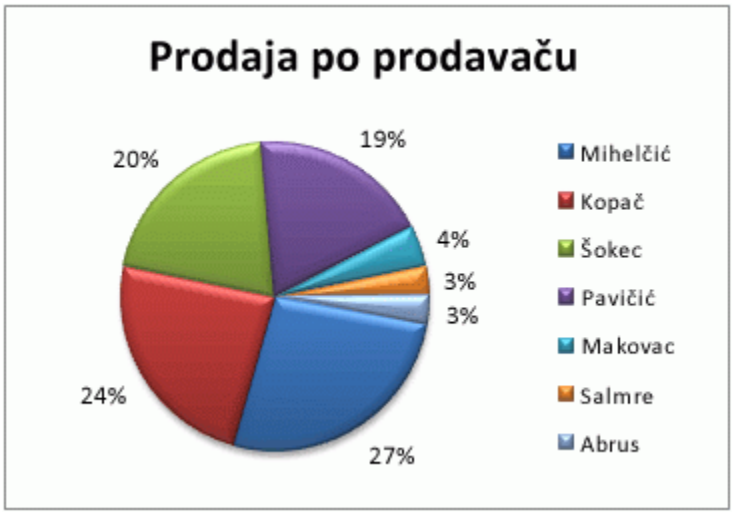
U Literaturi nalazimo mnoge definicije Vizualizacije podataka. Neke od njih su prikazane u nastavku:

„U svom najjednostavnijem obliku, vizualizacija podataka skup je metoda za grafički prikaz informacija na razumljiv i jednostavan način, u idealnom slučaju uz istovremeno uključivanje estetskih razmatranja za poticanje angažmana i interesa kako bi se zauzvrat privukla pažnja ciljane publike.“ Sinar (2015:116)

„Vizualizacija podataka široko je područje na raskrižju matematike, informatike, kognitivnih i percepcijskih znanosti te inženjerstva. Pokriva svaku disciplinu koja dijeli principe s vizualizacijom, od teorije signala do slike i od računalne grafike do statistike“ Friendly (2008:1)

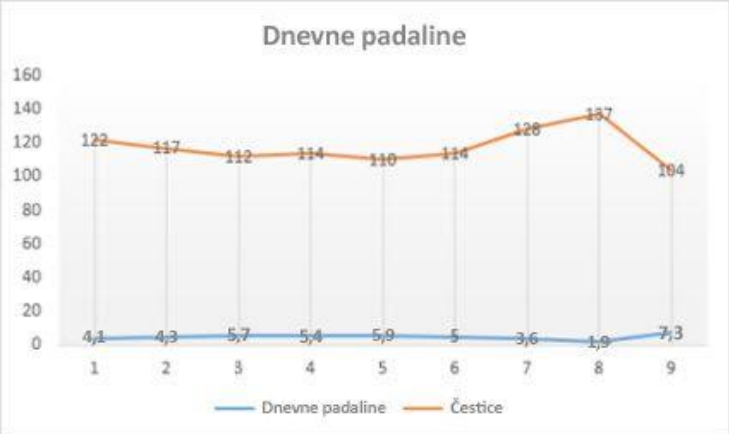
„Vizualizacija podataka novi je pojam. Izražava ideju koja uključuje više od pukog predstavljanja podataka u grafičkom obliku (umjesto korištenja tablice). Informacije koje stoje iza podataka također bi trebale biti otkrivene u dobrom prikazu; grafika bi trebala pomoći čitateljima ili gledateljima da vide strukturu u podatcima.“ Chen, Härdle i Unwin (2007:6)

Na temelju navedenih definicija možemo zaključiti kako je vizualizacija podataka široko područje koje se proteže kroz mnoga druga znanstvena područja kao što su matematika, informatika, inženjerstvo i dr., a služi nam kako bi otkrili informacije iza podataka, odnosno kako bi otkrili skrivenu strukturu u podatcima. To radi na jednostavan i razumljiv način koristeći različite tehnike vizualizacije. Neke od najčešćih tehnika vizualizacije podataka su:



Slika 1. Primjer tortnog grafa, Izvor (https://support.microsoft.com [01.02.2021])

Tortni graf (*eng. pie chart*) služi za prikaz udjela pojedine vrijednosti unutar ukupnog skupa vrijednosti. Na slici 1 vidimo primjer tortnog grafa koji prikazuje postotak prodaje po pojedinom prodavaču. U pravilu se koriste za usporedbu dijelova cjeline, međutim imaju i druge namjene.



Slika 2. Primjer linijskog grafa, Izvor (https://support.microsoft.com [01.02.2021])

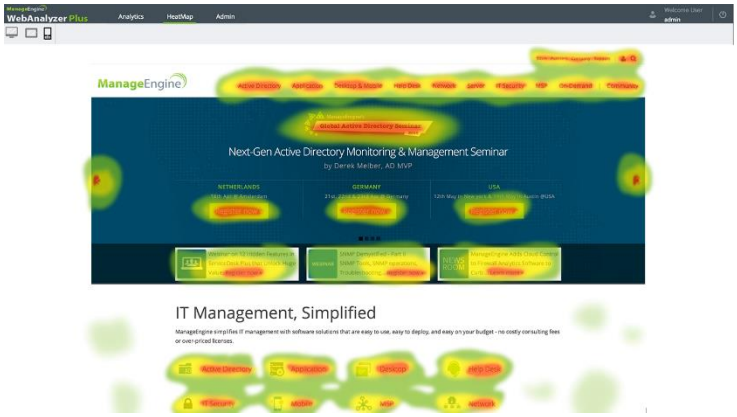
Linijski graf (*eng. line chart*), koji se prikazuje s oznakama za prikaz pojedinačnih vrijednosti podataka ili bez njih, može se koristiti za prikaz trendova tijekom vremena ili po ravnomjerno raspoređenim kategorijama, osobito kada postoje brojne točke podataka, a važan je i redoslijed kojim su predstavljene. Linijski grafovi najbolje funkcioniraju kad je na grafu prisutno više nizova podataka u slučaju samo jednog niza podataka, preporučuje se korištenje raspršenog grafa. (https://support.microsoft.com [01.02.2021]). U primjeru prikazanom na slici 2 možemo vidjeti linijski graf koji prikazuje dnevnu razinu padalina i razinu čestica u padalinama.

Slika 3. Primjer stupčastog grafa, Izvor (https://support.microsoft.com [01.02.2021])

Stupčasti grafovi (*eng. bar chart*) koriste se za usporedbu različitih količina po određenim kategorijama. Postoje različite varijacije stupčastih grafova neki primjeri su grupirani stupčasti graf (*eng. Grouped bar chart*) koji za razliku od običnog prikazuje grupe količina po kategorijama naslagani stupčasti graf (*eng.stacked bar chart*), koji za razliku od grupiranog stupčastog grafa koji sve grupe prikazuje kao zasebne stupce, grupe prikazuje kao udjele unutar jednog stupca. Stupčasti graf treba razlikovati od histograma. Glavna je razlika u tome što se histogrami koriste samo za prikaz učestalosti pojavljivanja vrijednosti u kontinuiranom skupu podataka koji je podijeljen prema određenim kategorijama, stupčasti grafovi se mogu koristiti za puno drugih vrsta varijabli. Na slici 3 prikazan je stupčasti graf koji prikazuje prodaju nekog proizvoda na tržištu Istočne Azije za 2009. i 2010. godinu po tromjesečjima.

Slika 4. Prikaz raspršenog grafa, Izvor (https://support.microsoft.com [01.02.2021])

Raspršeni graf (*eng. scatter plot*) ima dvije osi vrijednosti: vodoravnu (x) i okomitu (y) os vrijednosti. Na njemu se vrijednosti x i y objedinjuju u jedinstvene točke podataka te se prikazuju u nepravilnim intervalima ili klasterima. Raspršeni grafovi najčešće se koriste za prikaz i usporedbu numeričkih vrijednosti, ako što su znanstveni, statistički i tehnički podatci. Na slici 4 vidimo primjer raspršenog grafa koji prikazuje razinu čestica u kiši. (https://support.microsoft.com [01.02.2021])



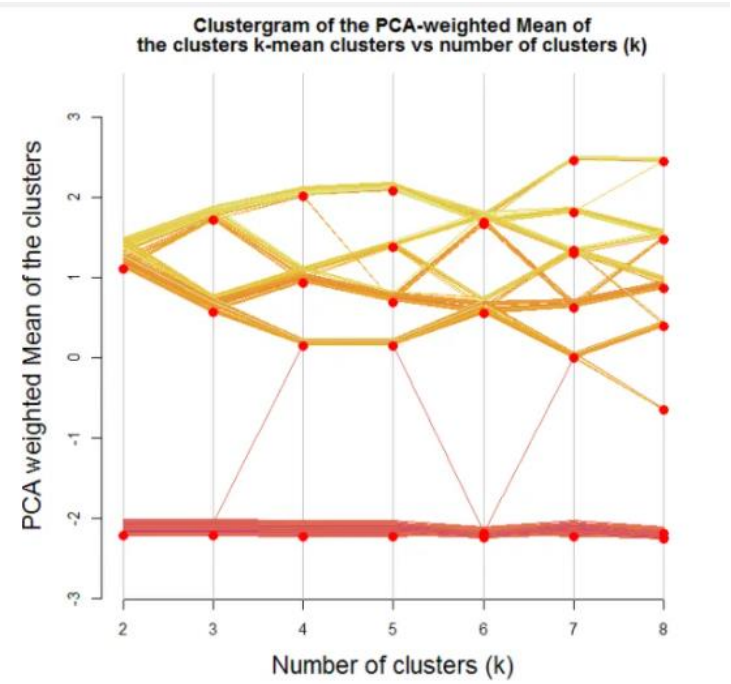
Slika 5. Primjer toplinske mape, Izvor (https://www.manageengine.com/web-analytics/free-website-heat-map-analysis-tool.html [01.02.2021])

Karte (*eng. maps*) su popularni načini vizualizacije podataka koji se koriste u različitim industrijama. Omogućuju lociranje elemenata na relevantnim objektima i područjima - zemljopisne karte, planovi zgrada, rasporedi web stranica itd. Među najpopularnijim vizualizacijama karata su toplinske karte, karte raspodjele točaka, kartogrami. Na slici 5 vidimo toplinsku mapu koja prikazuje broj klikova na određenoj stranici, po određenom području te stranice. Primjerice u konkretnom primjeru crvenom bojom su označena područja s visokim vrijednostima brojeva, žutom srednje vrijednosti brojeva, a zelenom najmanje vrijednosti. (https://www.digiteum.com/data-visualization-techniques-tools [01.02.2021])

Ranije navedene metode možemo svrstati u tradicionalnije metode vizualizacije podataka, a metode koje slijede u nastavku svrstavamo u moderne metode vizualizacije.

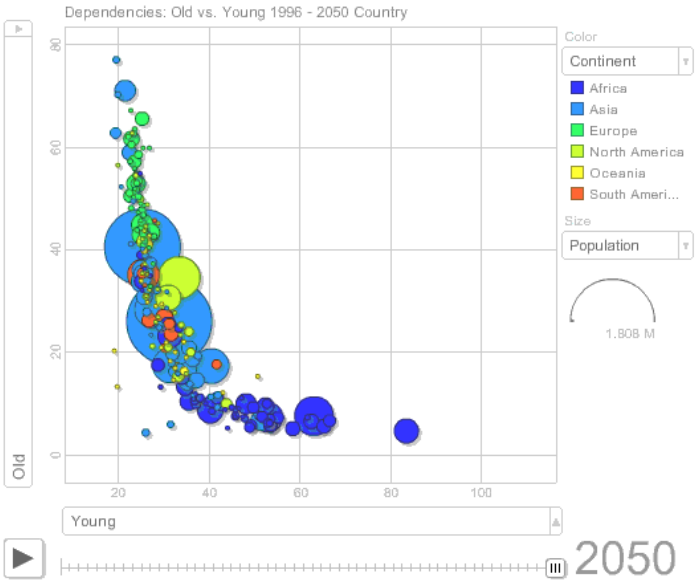
Slika 6. Primjer oblaka oznaka, Izvor:(https://www.nngroup.com/articles/tag-cloud-examples [03.02.2021])

Oblak oznaka *(eng.Tag Cloud)* - pojam koji opisuje sadržaj dokumenta ili skupa ili vizualni opis popularnog Web 2.0, koji se sastoji od kratkih fraza. Ključne riječi ili referencirani objekti izvedeni iz sadržaja dokumenta obično se opisuju tehnikama obrade na prirodnom jeziku. Veličina, boja i položaj riječi ovise o njihovoj važnosti, estetskim kriterijima i kriterijima udobnosti. Ova metoda usvaja koeficijent težine za svaki element. Što je koeficijent veći, to će veličina fonta biti veća. Koeficijent težine ovisi o važnosti elementa koji je odredio stručnjak, učestalosti njegovog stanja i drugim čimbenicima. Hajirahimova i Ismayilova (2018)



Slika 7. Primjer klastergrama, Izvor:(https://www.r-statistics.com [03.02.2021])

Klastergram *(eng. Clustergram)* - metoda vizualizacije koja se koristi za demonstraciju načina na koji su elementi podataka integrirani u klastere kako se broj klastera povećava tijekom analize. Izbor broja klastera važan je parametar za analizu klastera. Hajirahimova i Ismayilova (2018)



Slika 8. Primjer grafa kretanja, Izvor: (https://excelcharts.com/google-motion-chart-api-visualization-population-trends [03.02.2021])

Graf kretanja *(eng. Motion charts)* - omogućuje učinkovito istraživanje i interakciju s velikim i višedimenzionalnim podatcima pomoću dvodimenzionalnih mjehurića. Mjehurići (glavni objekti ove metode) mogu se kontrolirati u skladu s razmatranom promjenom slike. Hajirahimova i Ismayilova (2018)

Slika 9. Primjer nadzorne ploče, Izvor(https://www.geckoboard.com/dashboard-examples [03.02.2021])

Nadzorna ploča *(eng. Dashboard)* - prikazuje datoteke dnevnika različitih formata i filtrira ih na temelju odabranih raspona podataka. Nadzorna ploča sastoji se od tri sloja, i to podataka (sirovi podatci), analize (podatci koji se prenose iz formula i podatkovnih slojeva u tablice) i prezentacije (grafičke prezentacije temeljene na sloju analize). Hajirahimova i Ismayilova (2018)

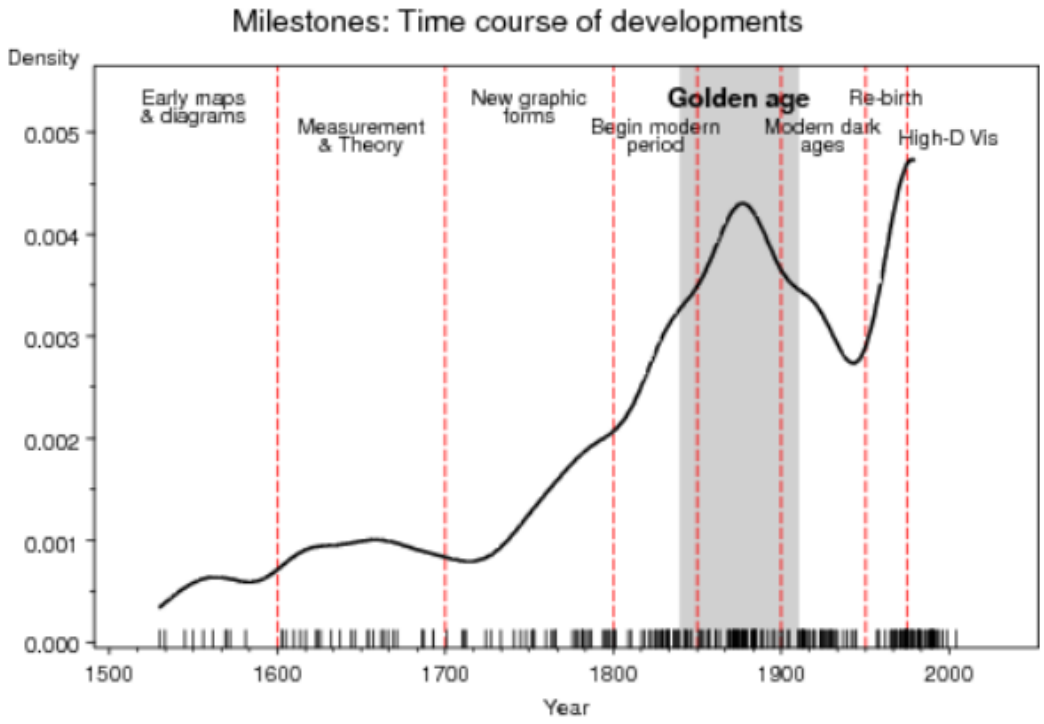
2.2. Područja upotrebe vizualizacije podataka

Iako je vizualizacija podataka relativno novi pojam, prve tragove vizualizacije nalazimo već u 17. stoljeću. Friendly, Sigal i Harnanansingh (2012) navode kako su neki od orijentira u ovoj priči bile prve tematske karte u 1600-ima, izum stupčastog grafa i linijskog grafa u ranim 1800-ima te današnja dinamična i interaktivna grafika.

Glavne grafičke metode koje se danas koriste - tortni, linijski i stupčasti grafovi – se uglavnom pripisuju Williamu Playfairu u radovima s početka 19. stoljeća. Sve su to u osnovi univarijatni prikazi nekog aspekta jedne varijable. Sljedeći bi logičan korak bio izumiti metodu kojom će se otkriti odnos između dvije varijable - onoga što danas znamo kao raspršeni dijagram. Do 1886. Francis Galton iskoristio je ovaj doista bivarijatni prikaz, što je dovelo do otkrića korelacije i regresije, i na kraju, do većine sadašnjih multivarijatnih statistika. Međutim, nije bio prvi koji je koristio ovu grafičku tehniku i iznenađuje da nitko nije široko zaslužan za njezin izum. Friendly, Sigal i Harnanansingh (2012:3)

Nakon navedenih događaja, Friendly, Sigal i Harnanansingh (2012) navode kako vizualizacija ulazi u svoje zlatno doba koje otprilike traje od 1850. do 1900. (± 10), nakon čega ulazi u moderno mračno doba koje tako nazivaju zbog relativno malog broja doprinosa ovom području. Nakon mračnog doba koje završava polovicom 20. stoljeća dolazi ponovni uspon vizualizacije , a pojavom računala kakva danas poznajemo granice vizualizacije pomiču se na

nove razine. Na slici 10 možemo vidjeti vremensku raspodjelu događaja koji su se smatrali prekretnicama u povijesti vizualizacije podataka.



Slika 10. Vremenska crta razvoja vizualizacije Izvor: (Friendly, Sigal i Harnanansingh (2012:4))

Kao što smo ranije spomenuli vizualizacija podataka je područje koje se proteže na mnoga druga znanstvena područja kao što su : ekonomija, informatika, inženjerstvo ali i mnoga druga.

Informatika - u današnje vrijeme tvrtke u IT sektoru imaju pristup više podataka nego ikad prije samo neki od njih su: podatci o svojim kupcima, izvedbi njihovih oglasa, dosezanju njihovih proizvoda do ljudi i gotovo svim ostalim poslovnim mjernim podatcima. Međutim samo posjedovanje podatka nije dovoljno kao što smo ranije objasnili, podatci su nam gotovo beskorisni ako iz njih ne možemo izvući podatke. Korištenje grafova a pomaže nam u prepoznavanju trendova, slabosti i neočekivanih prilika. To se posebno odnosi na IT sektor koji je jedinstven po samoj količini podataka kojima može pristupiti. Kao primjer korištenja vizualizacije podataka u IT sektoru možemo navesti *Google Analytics* pomoću kojega imamo mogućnost identifikacije trendova tijekom vremena.

Inženjerstvo - za inženjere, kao i za ljude s manje tehničkog iskustva, čitanje informacija s grafova se može previdjeti zbog razine važnosti koju oni nose. Na primjer, značajna

istraživanja utvrdila su da inženjeri imaju tendenciju da prakticiraju vizualizaciju dok rade s podatcima. Vizualizacija igra ključnu ulogu u analizi podataka, jer pomaže u iznošenju precizne količine podataka za razbijanje logičkih i relacijskih obrazaca u kojima bi sirovi podatci bili nerazumljivi. Shah (2016:1)

Ekonomija - je područje u kojem se možda i najviše koristi vizualizacija podataka. Ekonomija, vizualizaciju podataka koristi kao moćan alat za dolaženje do novih informacija o promjenama u trendovima ponude i potražnje, kako bi se pravovremeno donijele odluke koje će osigurati ostanak u tržišnoj utakmici s konkurencijom.

2.3. Moderni pristup vizualizaciji podataka

2.3.1. Veliki podatci

Ušli smo u eru na temelju podataka u kojoj se podatci kontinuirano prikupljaju u razne svrhe. Sposobnost donošenja pravodobnih odluka na temelju dostupnih podataka presudna je za poslovni uspjeh, kliničko liječenje pa čak i nacionalnu sigurnost te upravljanje katastrofama. Uz to, podatci generirani u simulacijama velikih razmjera, astronomskim zvjezdarnicama, eksperimentima velike propusnosti ili senzorima visoke rezolucije pomoći će do novih otkrića ako znanstvenici imaju odgovarajuće alate za izvlačenje znanja iz njih. Međutim, većina podataka postala je jednostavno prevelika i često ima prekratak životni vijek. Gotovo sva područja proučavanja i prakse na kraju će se suočiti s ovim problemom velikih podataka *(eng. big data).* Vizualizacija se pokazala učinkovitom ne samo za predstavljanje bitnih informacija u ogromnim količinama podataka, već i za vođenje složenih analiza. Analitika i otkrivanje velikih podataka predstavljaju nove mogućnosti istraživanja zajednici računalne grafike i vizualizacije. Keim, Qu, Ma (2013)

Mnoga područja nastojanja imaju problema s velikim podatcima. Neke se klasične poslovne aplikacije već neko vrijeme suočavaju s velikim podatcima (npr. Sustavi rezervacija zrakoplovnih kompanija), a noviji poslovni programi za iskorištavanje velikih podataka su u izradi (npr. Skladišta podataka, federacije baza podataka). Interaktivnost važna, međutim ona nije uvijek moguća na velikim skupovima podataka. Problem velikih podataka možemo prepoznati kao dva različita problema: velike zbirke podataka i objekti velikih podataka. Zbirke velikih podataka agregati su mnogih skupova podataka. Tipični skupovi podataka su višestruki, često su multidisciplinarni i uglavnom se distribuiraju među više fizičkih mjesta i često su više baza podataka. Cox, Ellsworth (1997)

Uz odgovarajuću obradu podataka i analitiku, doslovno svaka vrsta ili veličina poduzeća može izvući korist iz velikih podataka. Prednosti vizualizacije podataka toliko su raznolike da svaki donositelj odluke, bilo da je riječ o vlasniku male maloprodajne trgovine ili menadžmentu velike proizvodne tvrtke, može pronaći način za izvlačenje i korištenje uvida u podatke u korist vlastitog poslovanja. U nastavku navodimo nekoliko osnovnih prednosti vizualizacije velikih podataka su: olakšavanje percepcije i razumijevanja; prepoznavanje odnosa i ovisnosti; pronalazak uzoraka u mjerilu i davanje predviđanja; omogućava nadzor, kontrolu i trenutni odgovor u stvarnom vremenu; utvrđivanje pogrešaka, kvarova i rizika. (https://www.digiteum.com/data-visualization-your-business [01.02.2021])

2.3.2. Pametni podatci

Iafrate (2015) navodi da ako usporedimo odnos između velikih podataka i njihove vrijednosti za poslovanje u povezanom svijetu u kojem se informacijske tehnologije neprestano razvijaju, nekoliko milijardi ljudi spaja se na internet i svakodnevno razmjenjuje informacije u stalnom protoku uz to sve će više objekata biti povezano s internetom i također svakodnevno razmjenjivati informacije u stalnom protoku što sa sobom povlači mnogo prilika za osobe i poduzeća koja će imati pristup tim podatcima i iz njih moći izvući razumne zaključke.

Iz navedenoga možemo zaključiti kako su pametni podatci (*eng. smart data*) upravo to, korisni podatci izvučeni iz velikih podataka. No pametne podatke možemo promatrati i kao podatke koje dobivamo od senzora u pametnom gradu, no neovisno o pogledu na pametne podatke, vizualizacijom će se moći iz njih izvući još dodane vrijednosti.

García-Gil, Luengo, García i Herrera (2019) još navode kako se pametni podatci (fokusirani na istinitost i vrijednost), čiji je cilj filtriranje buke i isticanje dragocjenih podataka, koje tvrtke i vlade mogu učinkovito koristiti za planiranje, rad, nadzor, kontrolu i inteligentno donošenje odluka . Tri su ključna atributa potrebna da bi podatci bili pametni, oni moraju biti točni, djelotvorni i pokretljivi:

- točno: podatci moraju biti ono što kažu s dovoljnom preciznošću da bi se postigla vrijednost. Kvaliteta podataka je bitna.

- djelotvorno: podatci moraju pokretati trenutno skalabilnu akciju na način koji maksimizira poslovni cilj poput dosega medija na različitim platformama. Važna je skalabilna akcija.

- okretno: podatci moraju biti dostupni u stvarnom vremenu i spremni za prilagodbu promjenjivom poslovnom okruženju. Važna je fleksibilnost.

3. Vizualizacija podataka u mobilnim aplikacijama

3.1. Izazovi vizualizacije podataka prilikom izrade mobilne aplikacije

Khan, Shah i Ahmad (2014) navode kako se vizualizacija informacija na mobilnom uređaju suočava s raznim izazovima zbog zaslona ograničene veličine i odsutnosti standardnih uređaja za unos, što otežava uvođenje novih tehnika vizualizacije. Te prema tome izdvajaju sljedeće izazove prilikom vizualizacije podataka na mobilnim uređajima.

- Veličina zaslona: mobilni uređaji imaju mali zaslon pa je važno pametno ga koristiti i odabrati one podatke koji su korisnicima najvažniji.
- Prezentacija informacija: Prikazivanje korisnih informacija na malim ekranima nudi više izazova s mobilnim uređajima i postaje najvažniji problem s kojim se treba nositi.
- Interaktivnost sučelja: Za upravljanje informacijama na dostupnom malom zaslonu potrebni su učinkoviti i djelotvorni interaktivni mehanizmi za korisnike.
- Značajke vizualizacije: Da bi se stvorila učinkovita vizualizacija, tehnike moraju imati značajke vizualizacije poput funkcionalnosti, upotrebljivosti, učinkovitosti i korisnosti.

Uz navedene izazove treba imati na umu da mobilne uređaje koristi daleko više korisnika u odnosu na računala, te da se koristeći vizualizaciju na mobilnim uređajima predstavljamo široj javnosti. Iz tog razloga moramo razmišljati o vizualizacijama koje omogućuju istraživanje podataka na intuitivan i razumljiv način. Imajući to u vidu Blumenstein i sur. (2015) navode i slijedeće:

- Vizualizacija i kompatibilnost više uređaja: U odnosu na različite uređaje koji bi se mogli koristiti za istraživanje podataka, bit će potrebno koristiti okvir koji podržava višestruku platformu kompilacije. Uz to će biti potrebno automatizirano (semantičko) skaliranje za prikazanu vizualizaciju u odnosu na veliku raznolikost veličina zaslona uređaja.
- Vizualizacija i interakcija dodirom: Trebao bi postojati generalizirani skup gesta koje rade za širok raspon tehnika vizualizacije koje se često koriste. Prema tim tehnikama postoji drugo zanimljivo pitanje, ovisno o različitim veličinama zaslona i rezolucijama uređaja.

Osim izazova vezanih za same mobilne uređaje, njihovu različitost i mogućnost povezivanja s drugim podatcima, postoje i izazovi vezani uz same podatke. Sadiku i sur. (2016) navode ovdje dolazi do novog skupa pitanja, vezanih uz performanse, operativnost i stupanj diskriminacije, koji izaziva vizualizaciju i analizu velikih podataka. Stvaranje velikog simuliranog skupa podataka teško je i oduzima puno vremena. Također je teško odlučiti koji bi vizualni prikaz bio najbolje koristiti.

3.2. Najbolja praksa za vizualizaciju podataka prilikom izrade mobilne aplikacije

Imajući na umu ranije navedene izazove vizualizacije podataka na mobilnim uređajima možemo sastaviti okvirne smjernice za vizualizaciju podataka na mobilnim uređajima. Od samih početaka vizualizacije ljudi su je nastojali standardizirati te na neki način sastaviti okvire i smjernice za vizualizaciju podataka. Brightpointinc navodi kako su najbolje prakse korištene u papirnatim vizualizacijama statičkih podataka koje su započele krajem 1700-ih, otvorile su put tehnikama koje se danas koriste u stolnim računalima. No kako tehnologija napreduje, tako se moraju razvijati i najbolje prakse i tehnike za vizualizaciju podataka. Moramo uzeti najbolje od onoga što je u povijesti dobro funkcioniralo i iskoristiti mogućnosti novije tehnologije kako bismo dodali komunikacijsku vrijednost vizualizacije. Osim toga Brightpointinc navodi i sljedeće smjernice:

- Prikazivanje dovoljno podataka: graf sa samo 20 podatkovnih točaka neće biti ni približno sveobuhvatan kao onaj s preko 1000. Zapravo, ako možemo prikazati samo 20 podatkovnih točaka u isto vrijeme, korisniku će biti teško uočiti bilo što značajno trendovi ili obrasci u podatcima. Zaslonske značajke na računalu imaju puno više prostora u odnosu na mobilni zaslon. Za razliku od prosječne radne površine na kojoj imamo luksuz od preko milijun piksela, na prosječnom mobilnom telefonu imamo manje od 20% značajki na tom zaslonu. Stoga je važno da korisnicima pružimo uvid u razumnu količinu podataka.

- Povećavanje i smanjivanje: pružanje načina za prijelaz s pregleda na visokoj razini na detaljniji prikaz i povratak važno je kada pokušavate analizirati ukupne trendovske informacije i detalje ispod tih trendova. Na primjer, gledanje podataka o burzi kroz godine, mjesece i dane daje korisniku informacije o trendovima na visokoj razini i detaljne zrnaste podatke.

- Odabir i detalji podatkovne točke: korisnikov način treba odabrati određene podatkovne točke unutar niza podataka i dobiti povezane informacije. Kao primjer, kada korisnik gleda trend cijene dionica tijekom jednogodišnjeg razdoblja, želio bi također moći odabrati točku na liniji trenda i vidjeti cijenu za određeni dan. Osim toga, korisnik također može željeti vidjeti visoku, nisku, otvorenu, zatvorenu i količinu zaliha za isti dan. Ova sposobnost odabira zadane podatkovne točke i uvida u detalje podataka posebno je važna kada ste ograničeni na to koliko piksela morate predstavljati podatke grafa.

Osim tih smjernica Rockcontent (2020) navodi i sljedeće:

- Misliti na mobitel od početka: jedan od najvažnijih savjeta je odoljeti iskušenju kopiranja sadržaja radne površine na mobitel. Iako se ova praksa može činiti načinom uštede vremena i ponovne upotrebe materijala, rezultati nisu prihvatljivi. Stoga biste mobilni trebali razmišljati od nule. Korisničko iskustvo na mobilnom uređaju razlikuje se od pristupa na stolnim računalima. Stoga je vrlo vjerojatno da vaši grafovi i tablice jednostavno neće raditi na oba kanala. Morate uzeti u obzir, na primjer, veličinu zaslona uređaja kako biste izbjegli neuredne i nečitke podatke

- Razmotriti mobilne značajke: umjesto da mobilni uređaj i stolno računalo mislite kao jedno, iskoristite njihove razlike. Mobilni uređaji imaju jedinstvene značajke koje se mogu koristiti za poboljšanje potrošačkog iskustva. Jedan od vrhunaca je *GPS*. Ako kreirate interaktivne karte, koje su izvrstan sadržaj za privlačenje potrošača, geolokacija se može primijeniti za optimizaciju navigacije. Na primjer, možete istaknuti različite točke na karti kako biste označili područja koja sadrže određene informacije.

- Odrediti važnost sadržaja: da bi vizualizacija podataka imala željeni utjecaj, morate znati s kime komunicirate. Stoga je u marketinške svrhe važno raditi razmišljajući o osobnim interesima i preferencijama. Kakve informacije javnost želi? U kojem formatu publika radije troši podatke? Odgovori na ova pitanja trebali bi voditi čitav proces - uostalom, strategija sadržaja mora se usredotočiti na privlačenje i zadržavanje pozornosti publike.

- Odabir pravih alata: bitno je odabrati najprikladnije alate. Softver za vizualizaciju podataka omogućuje integraciju različitih izvora podataka, što olakšava njihov prikaz na jednoj platformi. Izvrstan alat koristi razne formate za pretvaranje brojeva u vizualni

sadržaj. Prije nego što odlučite koji ćete softver koristiti, provjerite sadrži li program funkcije prilagođene mobilnim uređajima.

3.3. Tehnologije vizualizacije podataka u mobilnim aplikacijama

Material Design je besplatni sustav dizajna koji je Google stvorio kako bi pomogao timovima u stvaranju visokokvalitetnih digitalnih iskustava za *Android, iOS, Flutter* i web. *Material Design* je prilagodljiv sustav smjernica, komponenata i alata koji podržavaju najbolje prakse dizajna korisničkog sučelja. *Material Design* pojednostavljuje suradnju između dizajnera i programera i pomaže timovima da brzo izgrade lijepe proizvode. *Material Design* sadrži niz komponenata koje pokrivaju niz potreba za izgradnju sučelja, uključujući:

- Zaslon: postavljanje i organiziranje sadržaja pomoću komponenata poput kartica, popisa i listova.

- Navigacija: Omogućavanje korisnicima kretanja kroz proizvod pomoću komponenata poput ladica za navigaciju i kartica.

- Akcije: Omogućavanje korisnicima da izvršavaju zadatke pomoću komponenata poput plutajućeg gumba za akciju.

- Ulaz: Omogućavanje korisnicima unosa podataka ili odabira pomoću komponenata poput tekstualnih polja, čipova i kontrola odabira.

- Komunikacija: Obavještavanje korisnika o ključnim informacijama i porukama pomoću komponenata poput brzih banera, natpisa i dijaloških okvira.

Material Design (2021)

Material Design sadrži mnoštvo alata za vizualizaciju podataka temeljenih na principima točnosti, korisnosti i skalabilnosti neki od tih alata su: tortni grafovi, razne varijacije stupčastih grafova, linijski grafovi, raspršeni grafovi i mnogi drugi. Osim toga za *Material Design* postoji i opsežna dokumentacija, dostupna na njihovim mrežnim stranicama i mnoštvo primjera dostupnih na internetu.

Drugi u nizu alata koje ćemo predstaviti u ovom radu je *MPA android charts* (u nastavku *MPA*). *MPA* je besplatna biblioteka fokusirana isključivo na vizualizaciju podataka. Slično kao i *Material Design*, iako u manjem opsegu, *MPA* sadrži razne alate za vizualizaciju podataka. Također *MPA* ima i dokumentaciju i mnoštvo primjera dostupnih na internetu. Za

razliku od *Material Design-a MPA* je namijenjen isključivo za izradu android aplikacija, te ne postoji dostupna inačica za izradu *iOS* aplikacija. *MPA* je izrazito jednostavan i intuitivan za korištenje U nastavku se nalazi popis nekih od projekata gdje je korišten *MPA*:

- *George Go*[1]

- *Drivvo*[2]

- *Smoke Free*[3]

AnyChart je lagana i robusna *JavaScript* biblioteka grafova s izvrsnim *API*-jem , dokumentacijom i podrškom na razini poduzeća. Razvijen je od 2003. s jednom glavnom idejom - bilo kojem programeru trebalo bi biti lako integrirati lijepe grafove u bilo koji mobilni, stolni ili web proizvod. Kao rezultat toga, *AnyChart* je danas sloj vizualizacije podataka za tisuće proizvoda. *AnyChart* radi s bilo kojom bazom podataka i radi na bilo kojoj platformi. Zanimljiva značajka je da sadrži i uređivač grafova, koji je *AnyChart* proširenje za korisničko sučelje koje omogućuje stvaranje i postavljanje različitih vrsta grafova. Rezultat se može spremiti s podatcima u *XML* ili *JSON* formatu ili u formatu za višekratnu upotrebu s *JavaScript* kodnim nizom. AnyChart (2021)

Okvir *SwiftPlot* je biblioteka na više platformi koji primarno omogućava crtanje grafova u *swiftu*. Postojeći *swift* okviri za crtanje (kao što je *CorePlot*) rade samo na *iOS*-u ili *mac*-u. Ideja *SwiftPlota* je stvoriti biblioteku na više platformi koja radi na *iOS*-u, *mac*-u, *linux*-u i *windows*-u. Sav kod za kreiranje i korisne funkcije uključeni su u modul *SwiftPlot*. Svaki prikazivač (*eng. Renderer*) implementiran je kao zasebni modul. Svaki prikazivač mora imati *SwiftPlot* kao ovisnost (*eng. dependency*) i mora odgovarati protokolu prikazivača definiranom u *Renderer.swift* modulu SwiftPlot-a. Svaka vrsta grafa je generička koja prihvaća podatke koji odgovaraju protokolu *FloatConvertible*. Trenutno *FloatConvertible* podržava i *Float* i *Double*. Prikazivač protokola definira sve potrebne funkcije koje prikazivač treba implementirati. Svaki graf mora biti u skladu s protokolom grafa. Trenutno ovaj protokol definira potrebne varijable i funkcije koje svaki graf mora implementirati kako bi podržao podgrafove. Moguće je i dodati serije na grafove koristeći njihove odgovarajuće funkcije, što se pohranjuje u niz kao serija objekata. Možete postaviti druga svojstva kao što su *plotTitle*, *plotLabel*, *plotDimensions* itd. Kako bi se generirao graf, poziva se funkcija *drawGraph* ili *drawGraphAndOutput*. Ovim se

[1] Dostupno na: https://www.erstegroup.com/en/news-media/media-library/photos/george
[2] Dostupno na: https://www.drivvo.com/en
[3] Dostupno na: https://smokefreeapp.com

izračunavaju svi parametri potrebni za generiranje grafa, poput koordinata obruba, skaliranih točaka za crtanje itd. Zatim ti podatci šalju prikazivaču koji ima funkcije za crtanje primitiva poput crta, pravokutnika i teksta. *SwiftPlot* biblioteka za sada sadrži stupčasti i linijski graf, histogram i raspršeni graf. SwiftPlot Dokumentacija (2021)

4. Motivacija za izradu aplikacije

4.1. Swot Analiza

Planiranje i kontroliranje vlastitog budžeta oduvijek je bila jedna od ključnih vještina na putu do financijskog uspjeh. U današnje vrijeme važnost te vještine je važnija no ikad prije, na sreću postoje mnogi alati koji nam pomažu kako bi što lakše vodili računa o tome. Neke od ključnih značajki koje takav alat treba imati su: Visoka razina prilagodljivosti korisnikovim potrebama, kvalitetni sustav planiranja i provjere ispunjavanja kreiranog plana, te dobar sustav izvještavanja. Većini dostupnih alata fali nešto od navedenog, primjerice aplikacija ima dobar sustav izvještavanja ali nedostaje joj dobar sustav planiranja i slično. Čak i kada aplikacije dostignu zadovoljavajuću razinu navedenih značajki tada gube na svojoj intuitivnosti i postaju prekomplicirane za upotrebu. Cilj ove aplikacije je implementirati navedene značajke zadržavajući pritom jednostavno korisničko sučelje.

Slika 11. SWOT analiza

Nakon provedene SWOT analize koja je prikazana na slici 11. uočeni si neki nedostatci na postojećoj aplikaciji te je stoga kreiran plan za daljnje poboljšanje aplikacije. Neke od ideja za daljnji razvoj su:

- Implementiranje mogućnosti zaključavanja aplikacije kako bi se onemogućio slučajan i nekontroliran unos
- Poboljšanje korisničkog sučelja za sustav izvještavanja
- Redizajniranje procesa kreiranja planova u svrhu kreiranja kvalitetnijih izvještaja
- Implementiranje mogućnosti u aplikaciju koja bi omogućila spajanje više korisnika u kućanstvo, čime bi aplikacija dobila dodatnu dimenziju i mogućnost upravljanja ne samo osobnim već i budžetom kućanstva
- Implementacija *SplashScreen-a* i dodavanje animacija na grafove što bi rezultiralo povećanjem atraktivnosti aplikacije
- Poboljšanje sustava obavještavanja
- Dodavanje mogućnosti kreiranja ciljeva, što bi dovelo po povećanja motivacije korisnika za korištenjem aplikacije
- Dodavanje različitih kalkulatora koji bi pomogli korisniku u izračunavanju isplativosti određenih ulaganja i daljnjem planiranju troškova. Primjerice dodavanje kalkulatora za izračun potrošnje goriva.
- Implementiranje modula za raspodjelu troškova, što bi bilo iznimno korisno za studente
- Implementiranje mogućnosti preračunavanja iznosa ovisno o odabranoj valuti na temelju dnevnih tečajnih lista.

4.2. Slične aplikacije na tržištu

U nastavku se nalazi pregled sličnih aplikacija koje su trenutno među najpopularnijima na tržištu.

Slika 12. Wally Logo, Izvor: (https://www.finder.com/uk/wally-app-review) [04.02.2021]

Wally je jedna od danas najkorištenijih aplikacija za praćenje budžeta. Wally se može okrenuti praćenju svih računa i plaćanja, sinkronizirat će se sa svima njima, a povrh toga Wally je dostupan u 70 zemalja. Wally je kompatibilan s 15.000 banaka, a može raditi i sa 60 različitih valuta. Finder (2021) navodi kako Wally nudi sljedeće mogućnosti:

- Sinkronizaciju računa: mogućnost povezivanja sa svojim tekućim računima, štednim računima, kreditnim karticama i zajmovima, te provjera najnovijih stanja.
- Dodavanje prihoda i troškova: mogućnost skeniranja i prijenosa svih fizičkih računa
- Stvaranje grupa za upravljanje zajedničkim računima.
- Postavljanje dnevnih proračuna: Wally automatizirano obavještava korisnika o preostalom iznosu za potrošnju ovisno o kreiranom proračunu
- Uvid u potrošnju: mogućnost organizacije troškova organizirati u kategorije, mjesta i oznake, te automatizirani izračun potrošnje ovisno o kategoriji te vizualizacija navedenih podataka
- Dijeljenje plaćanja s prijateljima.

Slika 13. Money Lover Logo, Izvor: (https://moneylover.me) [04.02.2021]

Comaprehero (2021) navodi kako je Money Lover u osnovi aplikacija za praćenje troškova. Aplikacija nudi mogućnost postavljanja i nekoliko 'Novčanika' i dodavanja transakcija (kao trošak ili prihod) u svaki novčanik u bilo kojem trenutku, što također može biti automatizirano, idealno za mjesečne račune. Uz to, postoje i mnoge druge ugrađene značajke, poput kalkulatora, deviznih tečajeva, mogućnosti skeniranja računa itd. Radi preciznijeg praćenja troškova (i kao usluga pretplate), Money Lover se također može povezati s određenim bankama i uslugama u Maleziji, poput *CIMB*-a, *Maybank*-a, Javne banke, *RHB*-a, *AmBank*-a, *Bank Islam* i *Hong Leong* banke , kao i *Paypal*. Nedavno su dodali i kriptovalutu na svoj popis povezanih usluga.

Slika 14. Goodbudget Logo, Izvor: (https://www.pcworld.com/article/3287145/goodbudget-review.html) [04.02.2021]

Prije poznat kao *Easy Envelope Budget Aid*, ili *EEBA*, Goodbudget je proračunska usluga koja pomaže pri raspoređivanju osobnih financije. Pristupa mu se putem putem web preglednika ili besplatne aplikacije na *iOS* i Android uređajima iz *App Storea* i *Google Play* trgovine. Račun se sinkronizira između aplikacije i web mjesta, što olakšava upravljanje proračunom kod kuće ili u pokretu. *Goodbudget* daje redovite preporuke za česte troškove kao što su najam, namirnice i zabava. Također omogućava raspodjelu na godišnje troškove i uštede poput odmora, božićnih poklona ili hitnog fonda. Nerdwallet (2021)

5. Implementacija i funkcionalnosti BudgetApp aplikacije

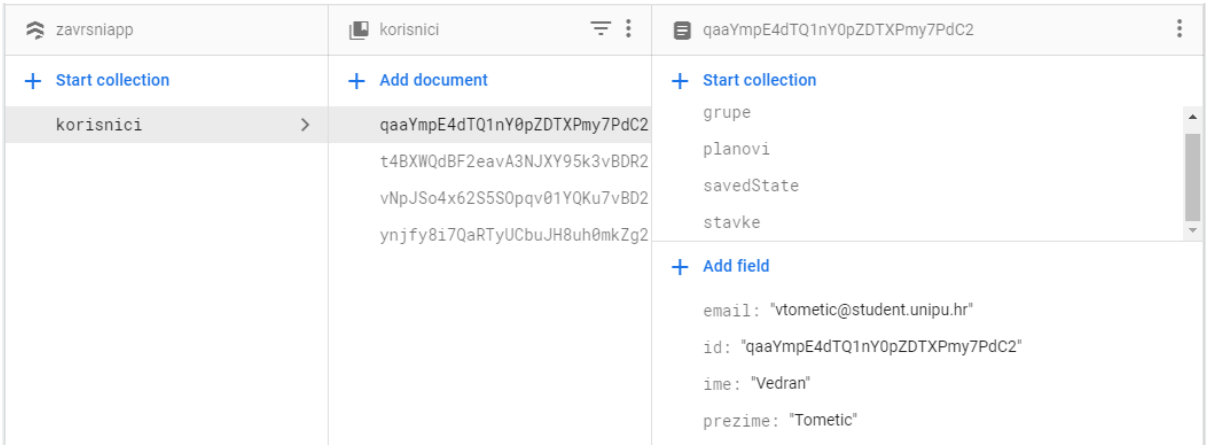
Aplikacija je izrađena tako da je mogu koristiti samo registrirani korisnici, stoga je prvi korak koji se očekuje od novog korisnika da se registrira ili ako se radi o postojećem korisniku da se prijavi u aplikaciju. Oba obrasca i prijava i registracija provjeravaju ispravnost podataka te ako je neki od unesenih podataka neispravan javljaju grešku korisniku, no sami proces prijave i registracije je detaljnije razrađen na sekvencijalnim dijagramima *(eng. Use Case Sequence)*. Nakon registracije, odnosno prijave u aplikaciju korisnik dolazi na početnu stranicu koja služi kao svojevrsni izbornik i s koje korisnik ima pristup ostalim dijelovima aplikacije, no logičan slijed događaja je da korisnik kreira grupe prihoda i troškova bez kojih ne može kreirati planove, niti unositi novonastale stavke prihoda ili troškova. Nakon što kreira grupe korisniku postaju dostupne ostale značajke aplikacije, kao što su kreiranje planova, dodavanje novonastalih prihoda i troškova te generiranje izvještaja. Osim toga korisnik ima mogućnost uređivanja vlastitih podatka na profilu te mogućnost prilagođavanja aplikacije unutar postavki. U nastavku slijedi detaljna razrada funkcionalnosti i razrada implementacije baze podataka i same aplikacije.

5.1. Izrada baze podatatka

Za izradu baze podataka korišena je *Cloud Firestore* baza podataka. *Cloud Firestore* fleksibilna je, skalabilna baza podataka za razvoj mobilnih uređaja, weba i poslužitelja iz *Firebasea* i *Google Cloud-a*. Poput *Firebase* baze podataka u stvarnom vremenu, on sinkronizira vaše podatke u klijentskim aplikacijama putem slušatelja u stvarnom vremenu i nudi izvanmrežnu podršku za mobilne uređaje i web, tako da nudi mogućnost izrade responzivnih aplikacija koje rade bez obzira na kašnjenje mreže ili internetsku povezanost. *Cloud Firestore* također nudi besprijekornu integraciju s ostalim *Firebase* i *Google Cloud* proizvodima, uključujući funkcije u oblaku. *Cloud Firestore* je *NoSQL* baza podataka u poslužitelju u oblaku kojoj iOS, Android i web aplikacije mogu pristupiti izravno putem izvornih *SDK*[4]-ova. Cloud Firestore je također dostupan u izvornim *Node.js*, *Java*, *Python*, *Unity*, *C ++* i *Go SDK*-ima, uz *REST* i *RPC API*-je. Cloud Firestore (2021)

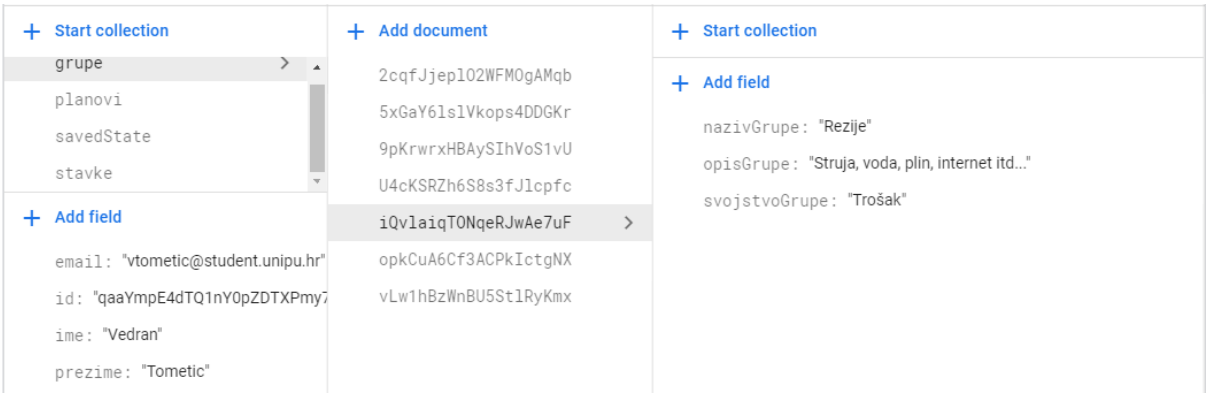
[4] software development kit

Na slici 15 vidimo prikaz korijenske kolekcije unutar baze podataka. Kolekcija korisnici sadrži popis dokumenta od kojih svaki sadrži potkolekcije i podatke koji se pohranjuju za svakog pojedinog korisnika. Dokument za svakog pojedinog korisnika sadrži potkolekciju grupe, planovi, savedState i stavke. Osim toga svaki dokument sadrži i podatke o korisniku kao što su e-mail, id korisnika, ime i prezime korisnika.



Slika 15. Struktura baze (korisnici)

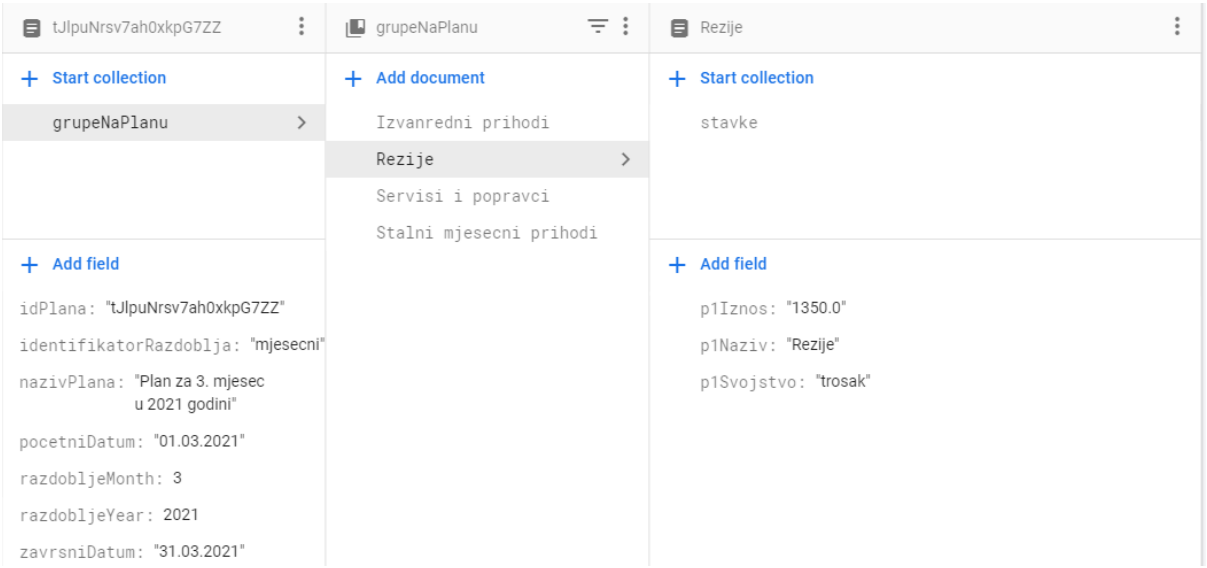
Slika 16 prikazuje strukturu kolekcije grupe, kolekcija grupe je potkolekcija kolekcije korisnici, a sadrži popis dokumenata koji sadrže podatke o grupama. prihoda i troškova. Svaki dokument sastoji se od naziva grupe, opisa grupe i svojstva grupe.



Slika 16. Struktura baze (grupe)

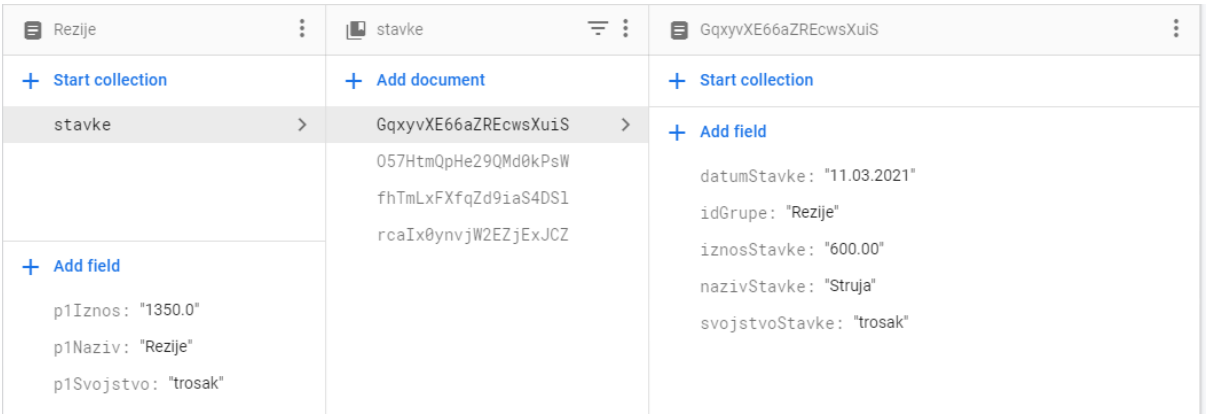
Kolekcija planovi je isto kao i grupe potkolekcija korijenske kolekcije korisnici. Ona se sastoji od popisa dokumenata od kojih svaki sadrži dodatnu potkolekciju grupe na planu i podatke koji opisuju dokument. Podatci koji opisuju dokument su: id plana, identifikator

razdoblja, naziv plana, početni datum, završni datum i ovisno o identifikatoru razdoblja može sadržavati *„razdoblježYear“* u slučaju da se radi o godišnjem planu, *„razdoblježYear“* i *„razdoblježMonth“* i *„razdoblježYear“*, *„razdoblježMonth“* i *„razdoblježWeek“* ako se radi o tjednom planu. Struktura kolekcije planovi prikazana je na slici 17.



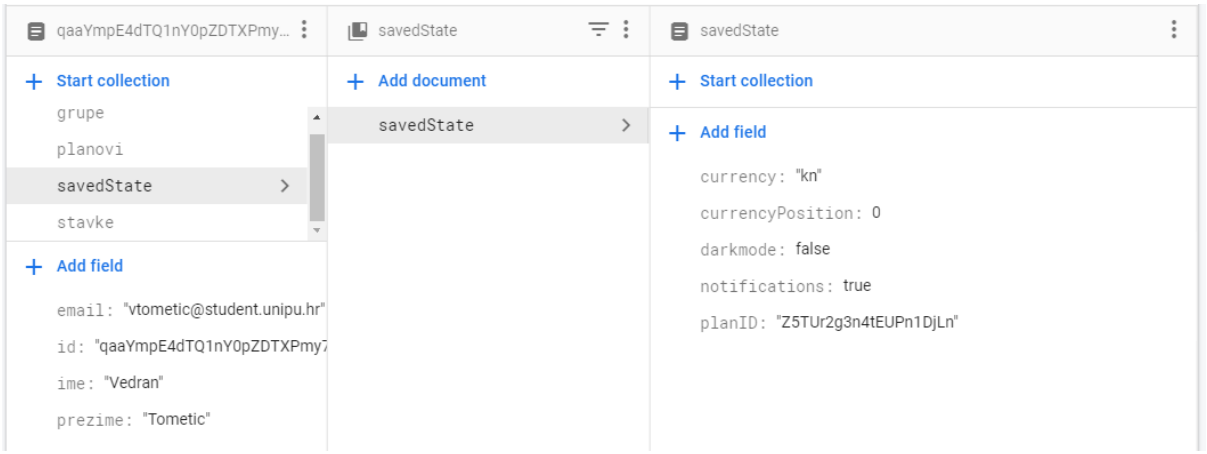
Slika 17. Struktura baze (planovi)

Slika 18 prikazuje strukturu kolekcije grupe na planu i njene potkolekcije stavke. Kolekcija grupe na planu sadrži popis dokumenta od kojih svaki sadrži opisne podatke kao što su *„p1Iznos“* što predstavlja iznos grupe na planu (izračunava se zbrajanjem iznosa svih stavki unutar grupe), *„p1Naziv“* što predstavlja naziv grupe na planu i *„p1Svojstvo“* što predstavlja svojstvo grupe na planu. Nadalje svaki dokument kolekcije grupe na planu sadrži i potkolekciju stavke koja sadrži sljedeće podatke: datum stavke, id grupe, iznos stavke naziv stavke i svojstvo stavke.



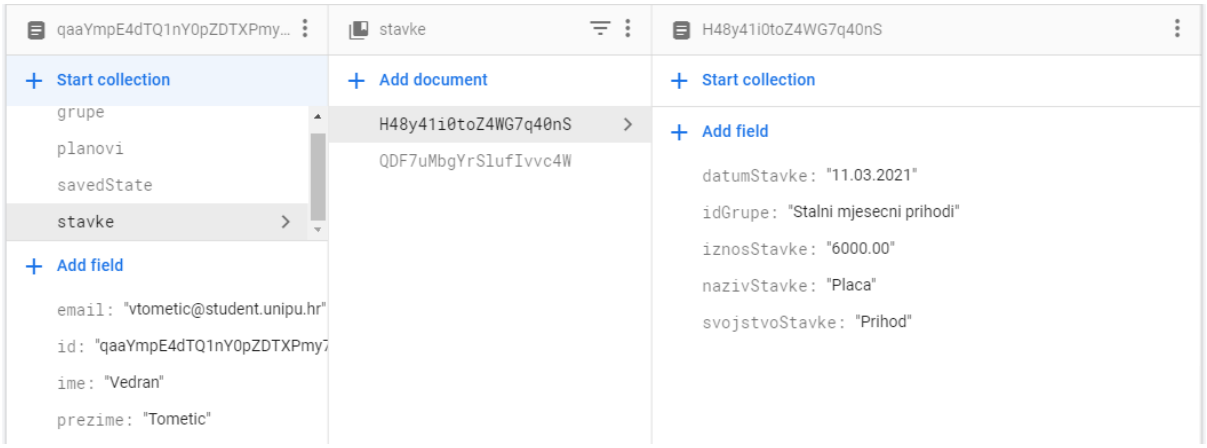
Slika 18. Struktura baze (stavke plana)

Na slici 19 nalazi se struktura kolekcije *„savedState“*, koja je potkolekcija korijenske kolekcije korisnici. Kolekcija *„savedState“* kreira se prilikom registracije, a popunjava prilikom odjave korisnika s razlogom spremanja korisničkih postavki aplikacije kada se korisnik sljedeći puta prijavi u aplikaciju. U kolekciju *„savedState“* pohranjuje se dio podataka iz klase *„Shared Preferences“*. Podatci koji se pohranjuju su: oznaka valute, pozicija valute, stanje noćnog načina, stanje obavijesti te id plana, ako postoji plan koji nije zaključen.



Slika 19. Struktura baze (savedState)

Zadnja kolekcija u okviru baze podataka je stavke. Unutar kolekcije stavke pohranjuju se novonastali prihodi i troškovi, a služe za generiranje izvještaja o prihodima i izvještaja o troškovima koji uspoređuju planirane i ostvarene prihode i troškove. Kolekcija stavke je potkolekcija korijenske kolekcije korisnici. Svaki dokument unutar kolekcije sadrži sljedeće podatke: datum nastanka stavke, id grupe, iznos stavke, naziv stavke i svojstvo stavke. Struktura te kolekcije prikazana je na slici 20.



Slika 20. Struktura baze (stavke)

Na kraju još možemo napomenuti kako se implementacija baze odvijala paralelno s implementacijom aplikacije. Svim dokumentima unutar baze pristupa se putem upita koji će biti prikazani u nastavku.

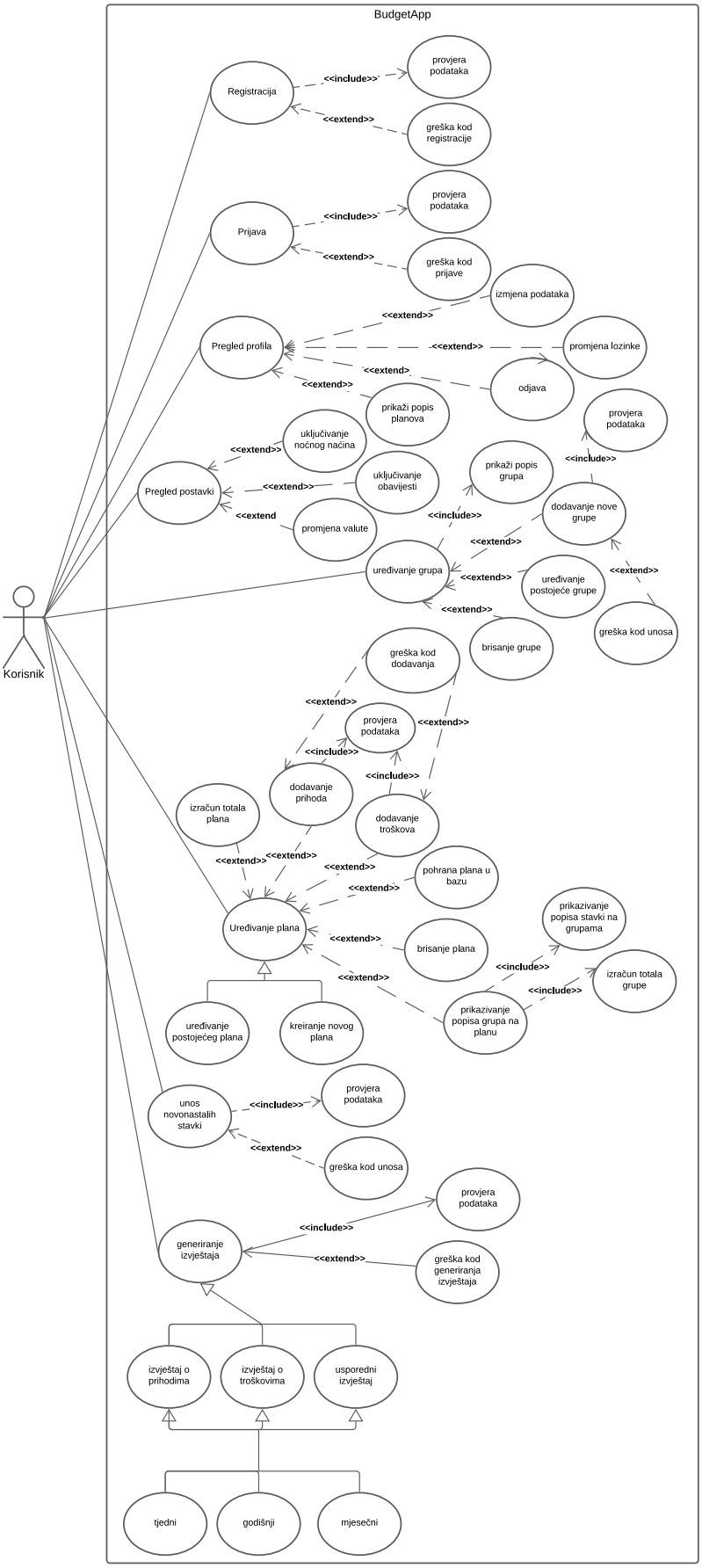
5.2. Izrada Mobilne aplikacije

Slika 21 prikazuje dijagram slučajeva korištenja *(eng. Use Case diagram)* aplikacije gdje je vidljivo da aplikaciju koriste isključivo korisnici. Oni se nalaze van granice sustava, koje su definirane pravokutnikom izvan kojega se nalaze sudionici.

Korisnik ima sljedeće slučajeve korištenja:

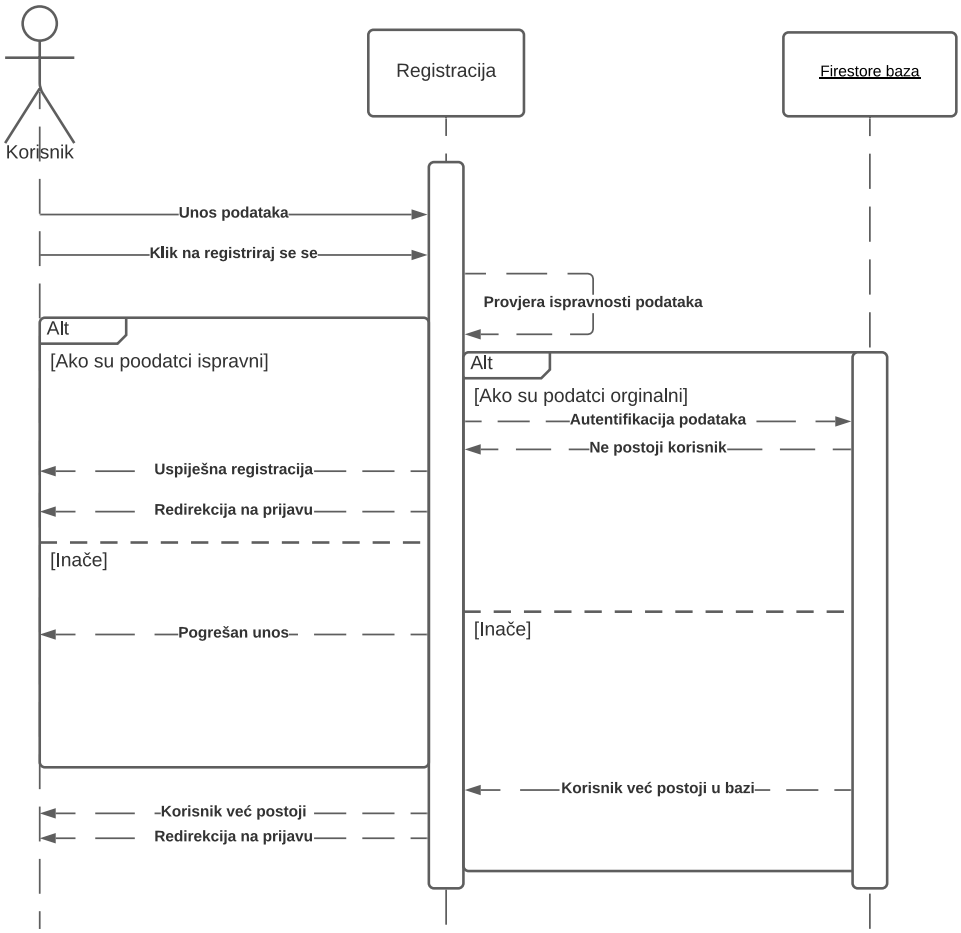
- Registracija
    - Kreiranje korisničkog računa
- Prijava
    - Prijava na postojeći račun
- Profil
    - Uređivanje podataka (slika, e-mail, ime, prezime, lozinka)
    - Odabir sa liste kreiranih planova (pristup uređivanju odabranog plana)
    - Odjava s prijavljenog korisničkog računa
- Postavke
    - Uključivanje ili isključivanje noćnog načina rada aplikacije
    - Uključivanje ili isključivanje obavijesti
    - Odabir određene valute

- Uređivanje grupe
    - Dodavanje novih grupa prihoda ili troškova
    - Uređivanje postojećih grupa prihoda ili troškova
- Uređivanje planova
    - Kreiranje novih planova
    - Uređivanje postojećih planova
- Unos novonastalih stavki
    - Dodavanje novonastalih prihoda
    - Dodavanje novonastalih troškova
- Generiranje izvještaja
    - Generiranje tjednih, mjesečnih, godišnjih izvještaja o prihodima (usporedba planirani i ostvareni)
    - Generiranje tjednih, mjesečnih, godišnjih izvještaja o troškovima (usporedba planirani i ostvareni)
    - Generiranje tjednih, mjesečnih, godišnjih izvještaja o ostvarenim prihodima i troškovima (usporedba troškovi i prihodi)



Slika 21. Use Case dijagram aplikacije

5.2.1. Prijava i Registracija



Slika 22. Sekvencijalni dijagram registracije

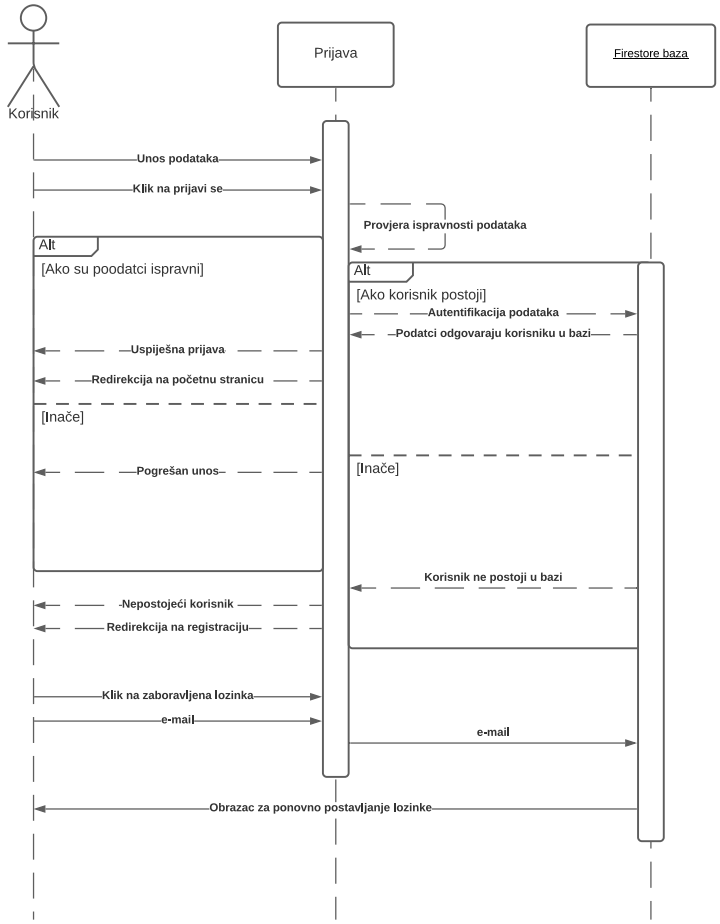
Na slici 22 nalazi se Sekvencijalni dijagram za proces registracije novih korisnika. Proces započinje tako da korisnik unosi tražene podatke, a to su: ime, prezime, e-mail, lozinka, ponovljena lozinka, nakon toga aplikacija provjerava jesu li uneseni podatci u traženom formatu i ako nisu korisniku se vraća greška o neispravnosti. Ako su podatci u traženom formatu šalje se zahtjev bazi za kreiranje novog korisnika, baza provjerava postoji li već korisnik s navedenim e-mailom i ako postoji korisniku se vraća greška da korisnik s unesenim podatcima već postoji, a ako ne postoji korisnik dobiva obavijest o uspješnoj registraciji i preusmjerava ga se na prijavu.

Samu funkciju za registraciju možemo vidjeti na primjeru koda 1. Funkcija prvo poziva ostale, manje *boolean*[5] funkcije, koje provjeravaju ispravnost podataka i tek kada svaka od tih pozvanih funkcija vrati vrijednost *true* što znači da su svi podatci ispravni. Nakon toga funkcija kreira korisnika s unesenim e-mailom i lozinkom. Nakon što se to uspješno završi funkcija, Funkcija putem *Snackbara*[6] obavještava korisnika o uspješnoj registraciji, dohvaća automatski kreirani id korisnika te kreira dokument s tim id-em u bazi podataka. Tom dokumentu dodaje podatke koje uzima iz kreirane *Hash Mape* pod nazivom korisnik. Na kraju funkcija još kreira kolekciju *savedState* za korisnika te u njoj kreira dokument u koji dodaje početne vrijednosti za vizualnu konfiguraciju sučelja korisnika. Nakon što i taj zadatak uspješno izvrši Aplikacija preusmjerava korisnika na početnu stranicu. U slučaju da nešto nije u redu s registracijom te se ona ne izvrši iz nekog razloga aplikacija obavještava korisnika o tome putem *Snackbar* poruke.

[5] Boolean vrijednost je vrijednost s dva izbora: istinitim ili lažnim.
[6] Snackbar je vrsta kratke poruke o procesima u aplikaciji koja se pojavljuje na dnu zaslona

```
public void logIn (View v){
    if(!validateIme()
            | !validatePrezime()
            | !validateEmail()
            | !validateLozinka()
            | !validatePonLozinka()){
        return;
    }
    final String ime = mIme.getEditText().getText().toString().trim();
    final String prezime = mPrezime.getEditText().getText().toString().trim();
    final String email = mEmail.getEditText().getText().toString().trim();
    final String lozinka = mLozinka.getEditText().getText().toString().trim();
    fAuth.createUserWithEmailAndPassword(email,lozinka).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if(task.isSuccessful()){

                Snackbar.make((View) mEmail.getParent(), "Korisnik "+ime+" "+prezime+"
kreiran", Snackbar.LENGTH_LONG).show();
                userID = fAuth.getCurrentUser().getUid();
                DocumentReference documentReference =
fStore.collection("korisnici").document(userID);
                Map<String,Object> korisnik = new HashMap<>();
                korisnik.put("id",userID);
                korisnik.put("ime",ime);
                korisnik.put("prezime",prezime);
                korisnik.put("email",email);
                documentReference.set(korisnik).addOnSuccessListener(aVoid -> {
                    sharedpref = new SharedPref(getActivity());
                    Map<String,Object> savedState = new HashMap<>();
                    savedState.put("darkmode",false);
                    savedState.put("notifications",false);
                    savedState.put("planID",  String.valueOf("prazno"));
                    savedState.put("currency","kn");
                    savedState.put("currencyPosition",1);
                    DocumentReference dr = fStore.collection("korisnici").document(userID)
                            .collection("savedState").document("savedState");
                    dr.set(savedState);
                });

                FragmentTransaction fragmentTransaction =
getActivity().getSupportFragmentManager().beginTransaction();
                fragmentTransaction.replace(R.id.container,
PrijavaFragment.newInstance());
                fragmentTransaction.addToBackStack(null);
                fragmentTransaction.commit();

            } else {
                Snackbar.make((View) mEmail.getParent(),
"Greška!"+task.getException().getMessage(), Snackbar.LENGTH_LONG).show();

            }
        }
    });
}
```

Kod 1. Funkcija za registraciju



Slika 23. Sekvencijalni dijagram prijave

Na slici 23 prikazan je proces prijave u aplikaciju, koji je gotovo identičan procesu registracije uz neke iznimke. Podatci koje korisnik unosi su samo e-mail i lozinka, nakon čega aplikacija kao i kod registracije provjerava zadovoljavaju li uneseni podatci traženi format, i ako ne zadovoljavaju korisniku se vraća greška s informacijama o pogrešnom unosu. Ako je traženi format zadovoljen podatci se šalju na provjeru u bazu i ako korisnik s unesenim e-mailom i lozinkom postoji korisnika se obavještava o uspješnoj prijavi te se on preusmjerava na početnu stranicu. Osim navedenih mogućnosti korisnik može i zatražiti ponovno postavljanje lozinke. Ako korisnik zatraži ponovno postavljanje, aplikacija to prosljeđuje bazi, a baza zatim korisniku šalje e-mail s uputstvima za ponovno postavljanje lozinke.

Primjer koda 2 prikazuje funkciju za prijavu, koja kao i funkcija za registraciju prvo poziva *boolean* funkcije za provjeru unesenih podataka te nakon toga prijavljuje korisnika u aplikaciju s e-mailom i lozinkom te u *Shared preferences*, koji se sprema na korisnikov uređaj, postavlja podatke za vizualnu konfiguraciju sučelja u aplikaciji. Te u *Snackbar* obavijesti

ispisuje korisniku poruku dobrodošlice. U slučaju neuspjele prijave korisniku se ispisuje poruka s detaljima greške koja je nastala kod prijave.

```
private void logIn(View v) {
    if (!validateEmail() | !validatePassword()) {
        return;
    }
    final String email = mEmail.getEditText().getText().toString().trim();
    String password = mLozinka.getEditText().getText().toString().trim();
    fAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(task ->
{
        if (task.isSuccessful()) {
            Snackbar.make((View) mEmail.getParent(), "Dobrodošli " + email + "!",
Snackbar.LENGTH_LONG).show();
            sharedpref.setUserID(fAuth.getUid());
            DocumentReference dr =
fStore.collection("korisnici").document(sharedpref.loadUserID())
                    .collection("savedState").document("savedState");
            dr.get().addOnCompleteListener(new
OnCompleteListener<DocumentSnapshot>() {
                @Override
                public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                    sharedpref = new SharedPref(getActivity());

sharedpref.setNotifications(Boolean.valueOf(task.getResult().get("notifications").t
oString()));

sharedpref.setNightModeState(Boolean.valueOf(task.getResult().get("darkmode").toStr
ing()));

sharedpref.setPlanID(String.valueOf(task.getResult().get("planID")));

sharedpref.setCurrency(String.valueOf(task.getResult().get("currency")));

sharedpref.setCurrencyPosition(Integer.valueOf(task.getResult().get("currencyPositi
on").toString()));

                }
            });
            FragmentTransaction fragmentTransaction = getActivity()
            .getSupportFragmentManager().beginTransaction();
            fragmentTransaction.replace(R.id.container,
            PocetnaFragment.newInstance());
            fragmentTransaction.addToBackStack(null);
            fragmentTransaction.commit();

        } else {
            Snackbar.make((View) mEmail.getParent(), "Greška! " +
task.getException().getMessage(), Snackbar.LENGTH_SHORT).show();
        }
    });
}
```

Kod 2. Funkcija za prijavu

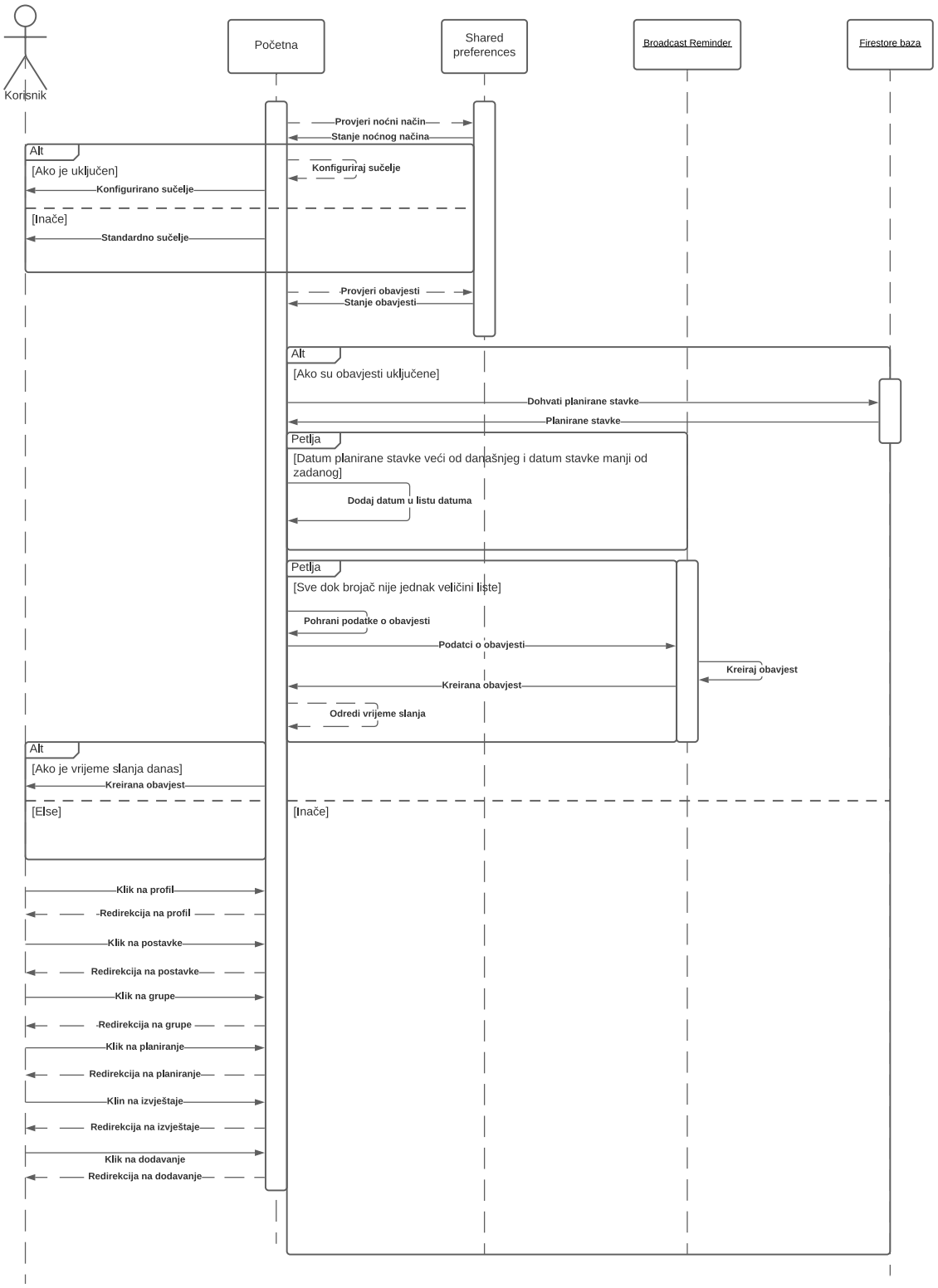
Kod prijave korisnik još ima mogućnost postavljanja nove lozinke u slučaju da je staru zaboravio. To vrši klikom na tekst „zaboravljena lozinka“ nakon toga se provjerava postoji li uneseni e-mail i ako postoji korisniku se šalje e-mail s uputama o daljnjem postupku promjene

lozinke. U slučaju da e-mail nije unesen korisnik se obavještava o tome. Kod za ponovno postavljanje lozinke prikazan je na primjeru koda 3.

```
resetPassword.setOnClickListener(v13 -> {
    if (validateEmail()) {

fAuth.sendPasswordResetEmail(Objects.requireNonNull(Objects.requireNonNull(mEmail.getEditT
ext().getText().toString())))
                .addOnSuccessListener(aVoid -> {
                    Snackbar.make((View) mEmail.getParent(), "Poruka sa linkom za promjenu
lozinke poslana je na: "
                            + mEmail.getEditText().getText().toString(),
Snackbar.LENGTH_SHORT).show();
                });
    }
});
```

Kod 3. Ponovno postavljanje lozinke

5.2.2. Početna i profil



Slika 24. Sekvencijalni dijagram početne stranice

Prilikom dolaska na početnu stranicu aplikacija prvo provjerava stanje noćnog načina koje je pohranjeno na uređaju i ovisno o stanju konfigurira prikaz sučelja. Nakon toga aplikacija provjerava jesu li obavijesti uključene i u slučaju da jesu, dohvaća sve datume stavki planova. Kada su datumi stavki dohvaćeni aplikacija dodaje stavke koje su veće od današnjeg datuma i manje od zadanog datuma. I pohranjuje ih u zasebnu listu. Nakon toga prolazi se kroz navedenu listu te se datumi šalju Klasi *BoadcastReminder* koja kreira obavijesti i zatim vraća obavijest. Kada je obavijest vraćena izračunava se vrijeme slanja te obavijesti. I kada to vrijeme dođe putem *Alarm Manager-a* obavijest se šalje korisnikov uređaj. Osim toga, kao što smo ranije naveli, početna stranica služi i kao izbornik, koji sadrži šest kartica koje predstavljaju najvažnije funkcionalnosti aplikacije. Taj proces prikazan sekvencijalnim dijagramom je na slici 24.

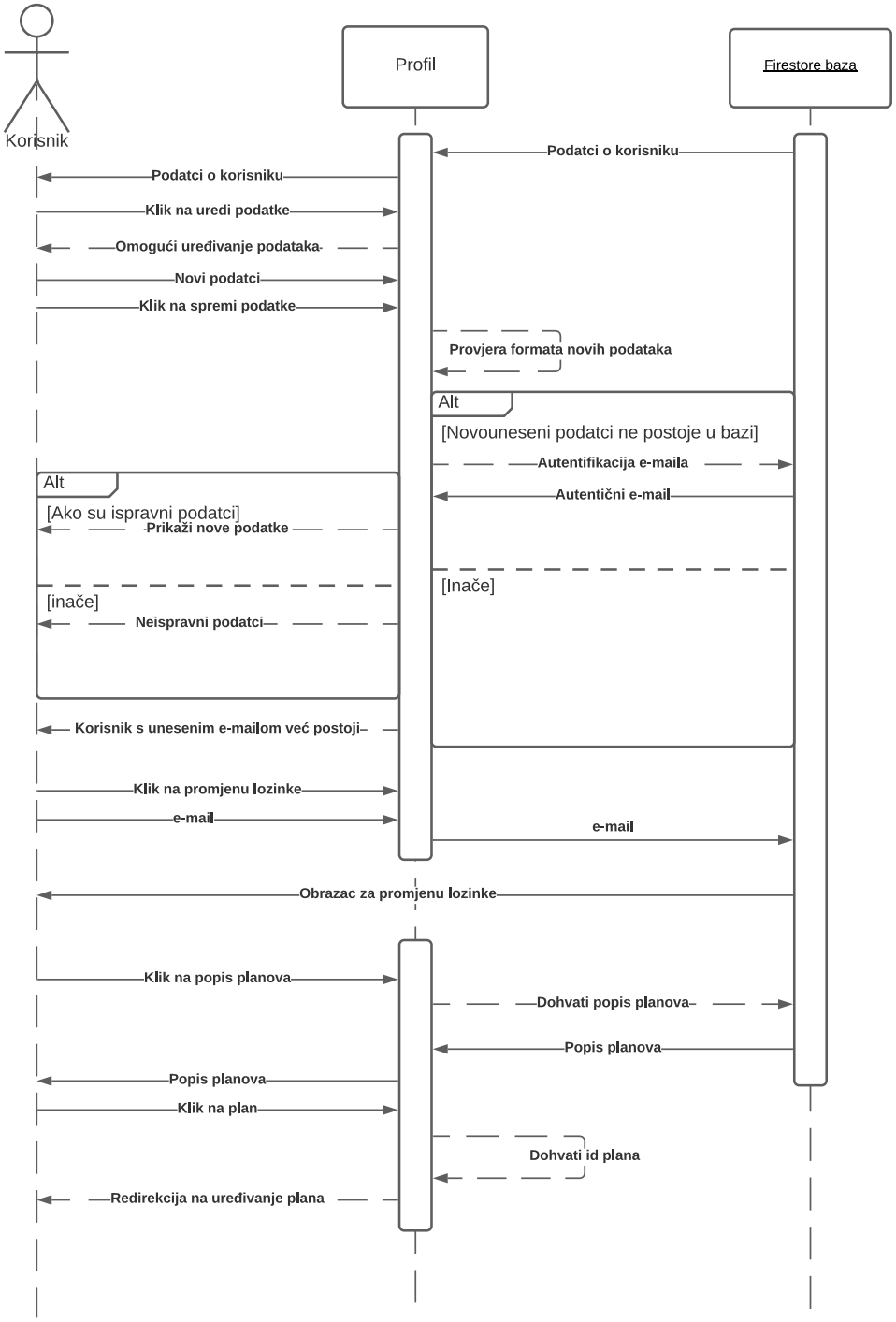
```
public void onReceive(Context context, Intent intent) {
    NotificationCompat.Builder bulider = new NotificationCompat.Builder(context,
"notifyUser")
            .setSmallIcon(R.drawable.ic_logo_final)
            .setContentTitle("Nadolazeći trošak")
            .setContentText("Prema vašim kreiranim planovima postoje troškovi kojima je
rok dospijeća danas")
            .setPriority(NotificationCompat.PRIORITY_DEFAULT);

    NotificationManagerCompat notificationManager =
NotificationManagerCompat.from(context);
    notificationManager.notify(200, bulider.build());
}
```

Kod 4. Kreiranje obavijesti

Primjer koda 4 prikazuje ranije opisano kreiranje obavijesti, a primjer koda 5 slanje obavijesti

```
if (sharedpref.loadNotifications()) {
    sendNotifications();
    Intent intent = new Intent(getActivity(), BroadcastReminder.class);
    PendingIntent pendingIntent = PendingIntent.getBroadcast(getActivity(), 0, intent, 0);
    AlarmManager alarmManager = (AlarmManager)
getActivity().getSystemService(Context.ALARM_SERVICE);
    getPlans();
    long timesAtClick = System.currentTimeMillis();
    long TenSecs = (1000 * 10);
    alarmManager.set(AlarmManager.RTC_WAKEUP, timesAtClick + TenSecs, pendingIntent);
}
```

Kod 5. Slanje obavijesti



Slika 25. Sekvencijalni dijagram profila)

Na slici 25 vidimo sekvencijalni dijagram za profil. Na početku aplikacija dohvaća iz baze podatke o korisniku te ih postavlja na sučelje profila. Ako korisnik želi uređivati podatke to može uključivanjem prekidača dostupnog na sučelju. U slučaju da korisnik promjeni podatke i želi da oni ostanu spremljeni to čini klikom na gumb spremi. Nakon toga pokreće se proces provjere podataka i ako je sve u redu izmijenjeni podatci se pohranjuju. A ako nešto nije u redu

korisniku se vraća obavijest o tome. Primjerice neke od grešaka koje mogu nastati prilikom promjene podataka su :

- uneseni e-mail već postoji u bazi

- uneseni e-mail nije u ispravnom formatu

- neko od polja je prazno

U okviru profila korisnik ima i mogućnost pregleda do sada kreiranih planova, promjene lozinke i odjave s trenutno prijavljenog korisničkog profila. U slučaju promjene lozinke pokreće se procedura koja je već opisana kod prijave. U pregleda, do sada kreiranih planova, klikom na „Moji planovi“ korisniku se otvara padajući izbornik s do sada kreiranim planovima. Ako korisnik klikne na određeni plan aplikacija ga preusmjerava na sučelje za uređivanje plana kojemu prosljeđuje id odabranog plana. I na kraju ako se korisnik želi odjaviti to postiže klikom na odjava, čime se pokreće procedura kojom se u bazu, s uređaja, spremaju podatci za konfiguraciju sučelja kao što su noćni način, obavijesti, odabrana valuta, pozicija odabrane valute i id plana. U nastavku na primjeru 6. nalazi se kod za dohvaćanje svih planova koje je korisnik kreirao. Ovaj kod je ujedno i primjer jednostavnog *Firestore* upita koji obuhvaća sve dokumente i pohranjuje odabrane podatke s tih dokumenata u liste.

```
private void initPlans() {

    Query colRef = fStore.collection("korisnici").document(sharedpref.loadUserID())
            .collection("planovi");
    colRef.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            for (QueryDocumentSnapshot document : task.getResult()) {
                mNazivi.add(document.get("nazivPlana").toString());
                mRaspon.add(document.get("idPlana").toString());

            }
        }
    }).addOnFailureListener(e ->
            Toast.makeText(getActivity(), "Greška", Toast.LENGTH_SHORT).show());
}
```

Kod 6. Upit za dohvaćanje planova

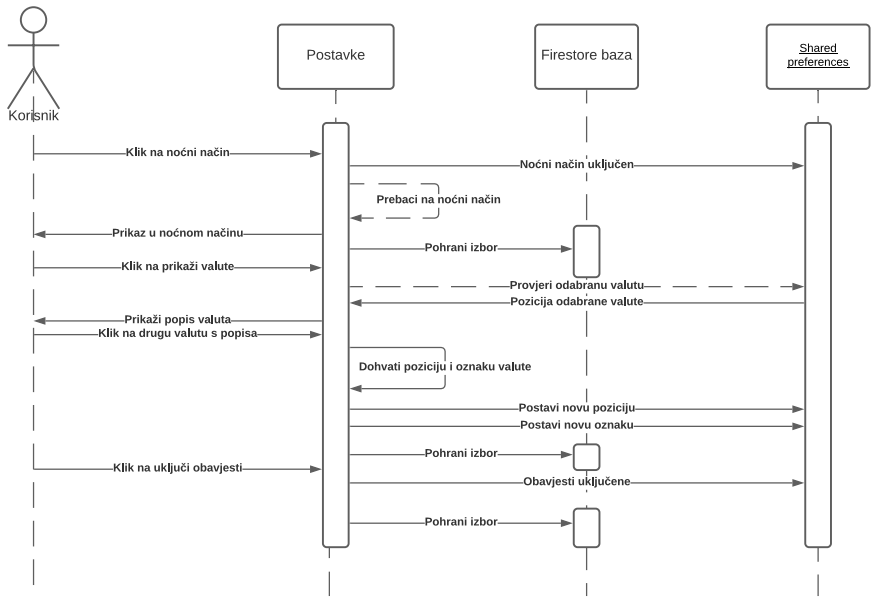
Na primjeru koda 7 prikazana je funkcija za odjavu koja je ranije opisana. Funkcija pohranjuje podatke za konfiguraciju zaslona u bazu i ako se to uspješno izvede aplikacija postavlja podatke za konfiguraciju zaslona na početne vrijednosti i prebacuje korisnika na obrazac za prijavu.

```
private void signOut() {
    Map<String, Object> savedState = new HashMap<>();
    savedState.put("darkmode", sharedpref.loadNightModeState());
    savedState.put("notifications", sharedpref.loadNotifications());
    savedState.put("planID", String.valueOf(sharedpref.loadPlanID()));
    savedState.put("currency", sharedpref.loadCurrency());
    savedState.put("currencyPosition", sharedpref.loadCurrencyPosition());

    DocumentReference dr =
fStore.collection("korisnici").document(sharedpref.loadUserID())
            .collection("savedState").document("savedState");
    dr.set(savedState).addOnSuccessListener(aVoid -> {
        sharedpref = new SharedPref(Objects.requireNonNull(getActivity()));
        sharedpref.setNotifications(false);
        sharedpref.setNightModeState(false);
        sharedpref.setPlanID("prazno");
        sharedpref.setCurrency("kn");
        sharedpref.setCurrencyPosition(1);
        sharedpref.setUserID("guest");
        fAuth.signOut();
        FragmentTransaction fragmentTransaction =
getActivity().getSupportFragmentManager().beginTransaction();
        fragmentTransaction.replace(R.id.container, PrijavaFragment.newInstance());
        fragmentTransaction.addToBackStack(null);
        fragmentTransaction.commit();
    })
            .addOnFailureListener(e -> {
                String error = e.getMessage();
                Snackbar.make((View) btnSave.getParent(), "Greška: " + error,
Snackbar.LENGTH_SHORT).show();
            });
}
```

Kod 7.Funkcija za odjavu

5.2.3. Postavke

Slika 26 prikazuje sekvencijalni dijagram korisnikovog pregleda sučelja postavki. U slučaju uključivanja i isključivanja prekidača za obavijesti i noćni način informacija o stanju se pohranjuje na korisnikov uređaj putem *Shared Preference*-a, te se njegov izbor pohranjuje u bazu podataka. U slučaju klika na odabranu valutu korisniku se prikazuje popis dostupnih valuta i kada korisnik odabere neku drugu valutu. Za prikaz popisa valuta koristi se *Recyclerview adapter*.



Slika 26. Sekvencijalni dijagram postavki

Recyclerview, kao što naziv govori, reciklira te pojedinačne elemente. Kad se stavka pomakne s ekrana, *Recyclerview* ne uništava njezin prikaz. Umjesto toga, ponovno je koristi za prikaz novih stavki koje su pomaknute na zaslonu. Ključne funkcije koje *Recyclerview* koristi su:

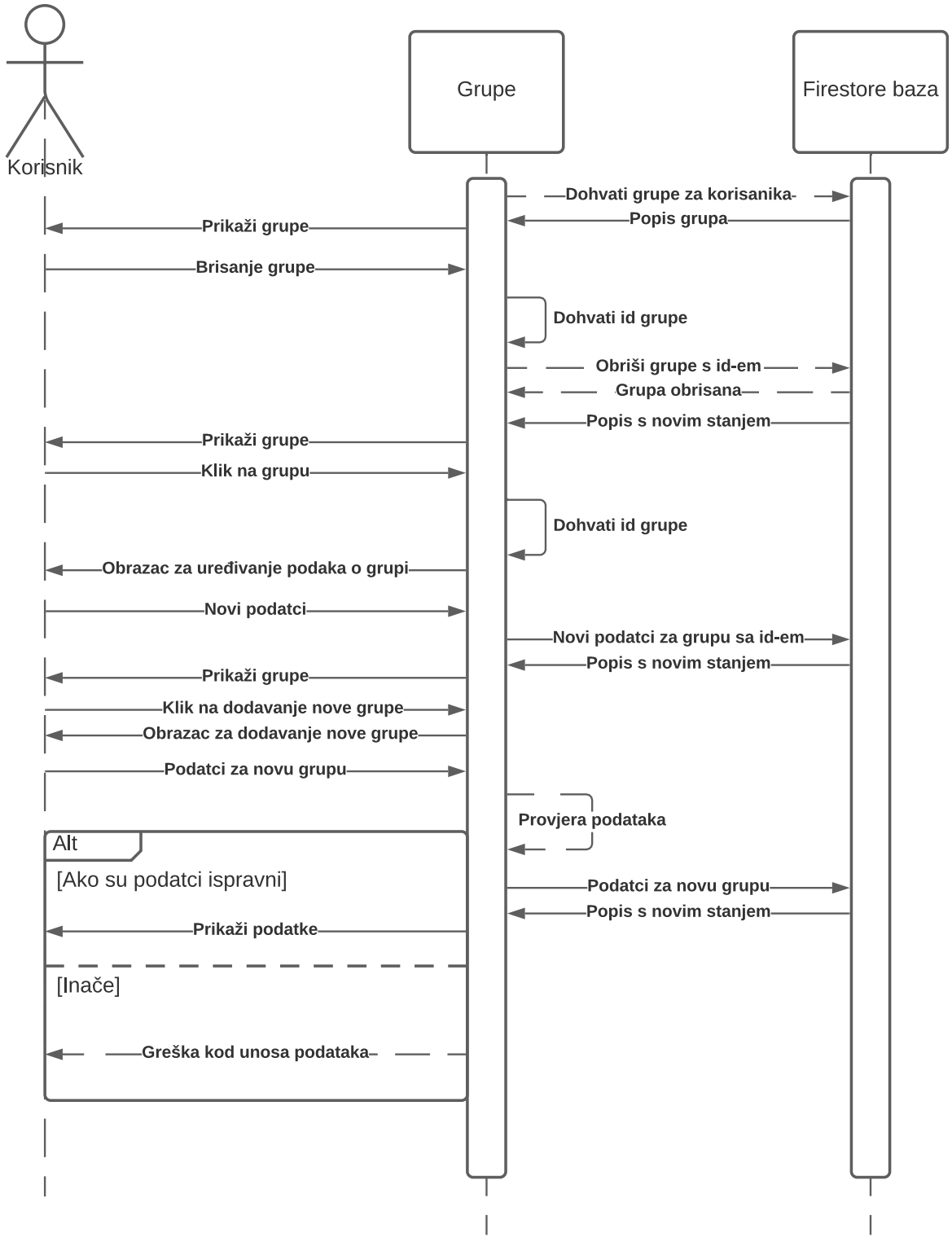
- *onCreateViewHolder()* koja se poziva kod kreiranja nove stavke unutar *Recyclerview*-a međutim ova metoda ne vezuje pripadajuće podatke za kreiranu stavku.
- *onBindViewHolder()* metoda koja veže pripadajuće podatke za kreirani *ViewHolder*
- *getItemCount()* koja vraća ukupan broj stavki unutar *Recyclerview*-a

Osim navedenih funkcija *Recyclerview* sadrži i klasu *ViewHolder* koja pridružuje vizualni okvir, kreiran kao *xml* datoteka, svakoj pojedinoj stavki adaptera. U nastavku, na primjeru 8 nalazi se programski kod za Recyclerview koji prikazuje valute u okviru postavki.

```
public class ValutaViewAdapter extends RecyclerView.Adapter<ValutaViewAdapter.Viewholder>
{

    SharedPref sharedpref;
    Button btnPotvrdi;
    private ArrayList<String> nazivi;
    private ArrayList<String> oznake;
    private Context mContext;
    int selectedPosition;

    public ValutaViewAdapter(Context mContext, ArrayList<String> nazivi, ArrayList<String>
oznake) {
        this.nazivi = nazivi;
        this.oznake = oznake;
        this.mContext = mContext;
    }

    @NonNull
    @Override
    public Viewholder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
                    .inflate(R.layout.valuta_item, parent, false);
        Viewholder holder = new Viewholder(view);
        return holder;
    }

    @Override
    public void onBindViewHolder(@NonNull Viewholder holder, int position) {
        int color = holder.valItem.getResources().getColor(R.color.cardcolor);
        int colorA = holder.valItem.getResources().getColor(R.color.colorPrimaryDark);
        sharedpref = new SharedPref(mContext);
        holder.valItem.setBackgroundColor(color);
        holder.valNaziv.setText(nazivi.get(position));
        holder.valOznaka.setText(oznake.get(position));
        selectedPosition = sharedpref.loadCurrencyPosition();
        if (selectedPosition == position) {
            holder.valItem.setBackgroundColor(colorA);
        } else
            holder.valItem.setBackgroundColor(color);

        holder.valItem.setOnClickListener(v -> {
            selectedPosition = position;
            notifyDataSetChanged();
            sharedpref.setCurrencyPosition(selectedPosition);
            sharedpref.setCurrency(holder.valOznaka.getText().toString());
        });
    }

    @Override
    public int getItemCount() {
        return nazivi.size();
    }

    public class Viewholder extends RecyclerView.ViewHolder {
        RelativeLayout valItem;
        TextView valNaziv, valOznaka;
        public Viewholder(@NonNull View itemView) {
            super(itemView);
            valNaziv = itemView.findViewById(R.id.valNaziv);
            valOznaka = itemView.findViewById(R.id.valOznaka);
            valItem = itemView.findViewById(R.id.valItem);
        }
    }
}
```

Kod 8. Recyclerview adapter za valute

5.2.4. Kreiranje grupa za troškove i prihode



Slika 27. Sekvencijalni dijagram grupe

Na slici 27 prikazan je sekvencijalni dijagram uređivanja i kreiranja grupa troškova i prihoda od strane korisnika. Na početku aplikacija dohvaća popis grupa s baze te ih prikazuje

korisniku unutar *Firestore Recyclerview*-a . U slučaju da korisnik želi obrisati neku od grupa dovoljno je da povuče odabranu stavku u lijevu ili u desnu stranu, što pokreće proceduru za brisanje grupe. Ako želi urediti postojeću grupu dovoljno je da klikne na grupu koju želi urediti, nakon čega mu se otvara dijalog za uređivanje grupe. Nakon izmjene podataka klikom na „ok“ korisnik potvrđuje izmjenu podataka, te se oni mijenjaju u bazi. Ako je izmjena uspješno izvršena baza vraća popis s izmijenjenim podatcima i aplikacija ju prikazuje korisniku. Ako korisnik želi dodati novu grupu, klikom na gumb s oznakom „+“, otvara mu se obrazac za dodavanje nove grupe, nakon što korisnik ispuni obrazac klikom na „ok“ pokreće se provjera podataka i procedura unosa nove grupe u bazu. Ako je unos uspješno izvršen Baza vraća aplikaciji novo stanje liste i zatim aplikacija dodaje tu novu stavku unutar *Recyclerview*-a.

Primjer koda 9 prikazuje funkciju za pohranu podataka u bazu koja se poziva u ranije opisanom procesu. Funkcija prvo poziva *boolean* funkcije za provjeru ispravnosti formata podataka. Ako su podatci ispravnog formata aplikacija ih dodaje u bazu, ako nisu postavlja grešku i objašnjenje na polje za unos koje ne zadovoljava traženi format. Ako je unos uspješan korisnik dobiva poruku da je grupa unesena. Korisnik također dobiva poruku u slučaju da je došlo do greške prilikom unosa. I na kraju zatvara dijalog.

```
private void saveToBase(String naziv, String opis, String svojstvo, AlertDialog dialog) {
    if (validateNaziv() | validateOpis()) {
        return;
    }
    Map<String, Object> grupa = new HashMap<>();
    grupa.put("nazivGrupe", naziv);
    grupa.put("opisGrupe", opis);
    grupa.put("svojstvoGrupe", svojstvo);
    DocumentReference dr =
fStore.collection("korisnici").document(sharedpref.loadUserID())
            .collection("grupe").document();
    dr.set(grupa).addOnSuccessListener(aVoid ->
            Snackbar.make((View) relativeLayout.getParent(), "Grupa unesena",
Snackbar.LENGTH_LONG).show())
            .addOnFailureListener(e -> {
                String error = e.getMessage();
                Snackbar.make((View) relativeLayout.getParent(), "Greška: " + error,
Snackbar.LENGTH_LONG).show();
            });
    dialog.dismiss();
}
```

Kod 9. Funkcija za spremanje grupe u bazu

Primjer koda 10 prikazuje funkciju koja se poziva prilikom izmjene podataka o grupi, funkcija je slična ranije opisanoj funkciji spremanja grupe u bazu, razlika je u tome što ova

funkcija ne dodaje novu stavku u bazu već uređuje postojeću stavku do koje dolazi temeljem id-a.

```
private void updateBase(String naziv, String opis, String svojstvo, String docid,
                        AlertDialog dialog) {
    if (validateNaziv() | validateOpis()) {
        return;
    }
    DocumentReference dr = fStore
            .collection("korisnici").document(sharedpref.loadUserID())
            .collection("grupe").document(docid);
    dr.update("nazivGrupe", naziv);
    dr.update("opisGrupe", opis);
    dr.update("svojstvoGrupe", svojstvo);
    dialog.dismiss();
}
```

Kod 10. Upit za promjenu podataka o grupi

U prethodnom poglavlju prikazan je *Recyclerview* adapter. Ovdje se koristi varijacija takvoga adaptera, a to je *Firestore Recyclerview* adapter, za razliku od običnog adaptera *Firestore adapter* komunicira direktno s bazom što omogućava jednostavan prikaz te brzo i lako dohvaćanje i izmjenu podataka baze. Iz tog razloga potrebno je kreirati model na temelju kojega će adapter dohvaćati podatke te ih vezati za stavke prikazane unutar *Recyclerview-a*. Primjer tog vezanja podataka možemo vidjeti na primjeru koda 11.

```
@Override
protected void onBindViewHolder(@NonNull GrupaHolder holder, int position, @NonNull
GrupaModel model) {
    holder.gNaziv.setText(model.getNazivGrupe());
    holder.gOpis.setText(model.getOpisGrupe());
    holder.gSvojstvo.setText(model.getSvojstvoGrupe());

}
```

Kod 11. BindViewHolder Firestore Recyclerview adaptera

Takav način rada omogućava i iznimno jednostavno brisanje, nakon inicijalizacije adaptera definiran je novi *itemTouchHelper*, koji i ovisno o smjeru povlačenja stavke možemo definirati funkciju koja će se izvršiti. U konkretnom slučaju, kao što je prikazano na primjeru koda 12, neovisno o smjeru povlačenja poziva se funkcija za brisanje definirana u *Friestore Recyclerview* adapteru. Funkcija za brisanje prikazana je na primjeru koda 13.

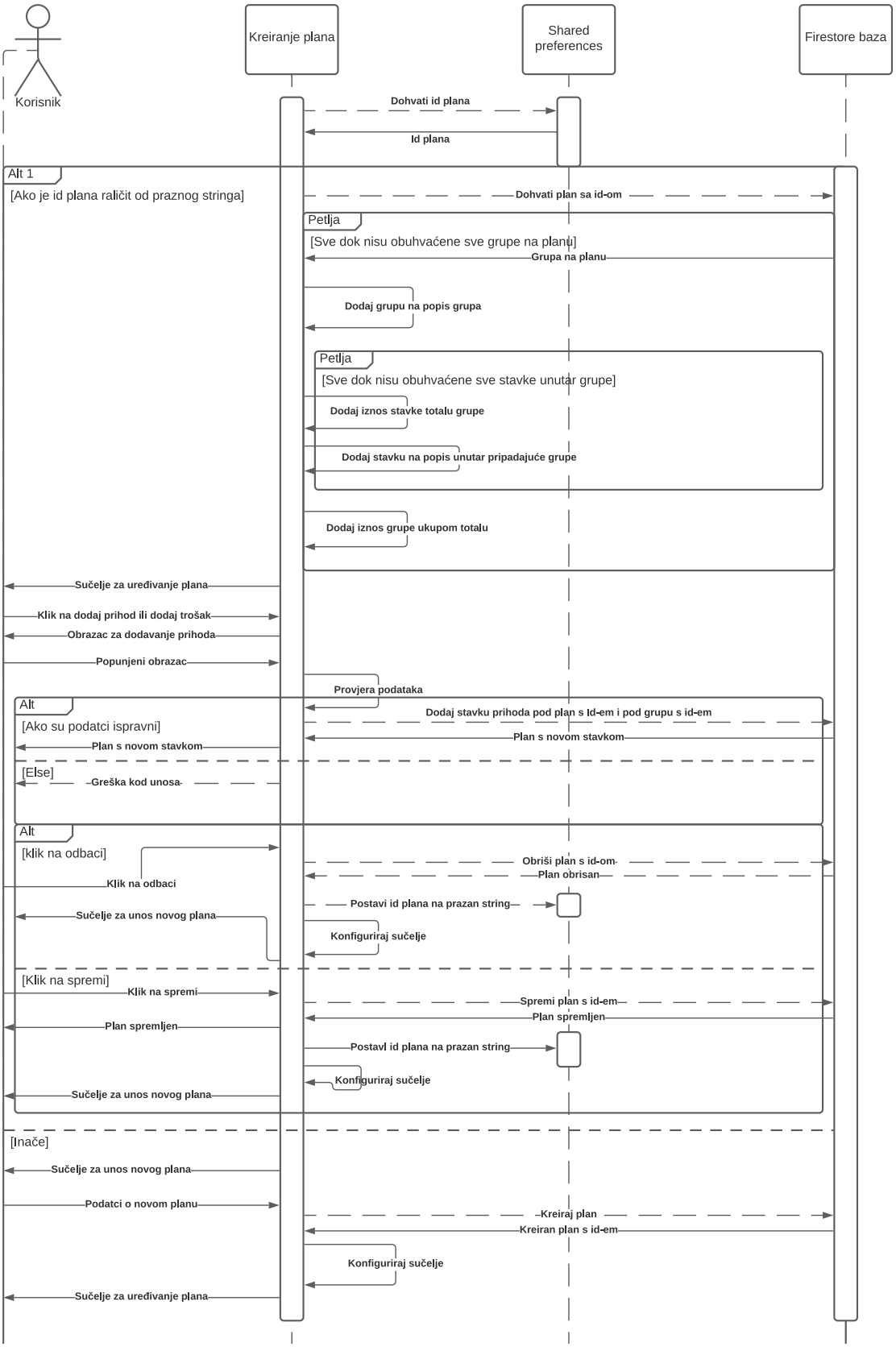
```
new ItemTouchHelper(new ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT |
ItemTouchHelper.RIGHT) {
    @Override
    public boolean onMove(@NonNull RecyclerView recyclerView,
    @NonNull RecyclerView.ViewHolder viewHolder,
    @NonNull RecyclerView.ViewHolder target) {
        return false;
    }

    @Override
    public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder, int direction) {
        adapter.deleteItem(viewHolder.getAdapterPosition());
    }
}).attachToRecyclerView(groupRecycler);
```

Kod 12.Brisanje grupe sa popisa

```
public void deleteItem(int position) {
getSnapshots().getSnapshot(position).getReference().delete();
}
```

Kod 13. Funkcija za brisanje grupe

5.2.5. Kreiranje planova



Slika 28. Sekvencijalni dijagram planova

Slika 28 prikazuje sekvencijalni dijagram za izradu i generiranje planova. Sučelje za planiranje generira se ovisno o tome radi li se o izradi novog plana ili o uređivanju postojećeg plana. Ako se radi o uređivanju postojećeg plana, dohvaća se id plana koji je putem *Shared Preference-a* pohranjen na korisnikovom uređaju. Nakon toga s baze se dohvaća popis grupa na generiranom planu te se taj popis grupa prikazuje unutar *Firestore Recyclerview-a*, čiji je rad objašnjen u prethodnom poglavlju. Za razliku od prethodnog Svaka stavka *Firestore Recyclerview-a* sadrži ugniježđeni *Recyclerview* koji prikazuje popis stavki svake grupe.

Osim toga korisnik ima mogućnost dodavanja novog prihoda ili troška. Sučelje također prikazuje ukupan iznos plana koji se računa tako da se, od zbroja iznosa grupa u svojstvu prihoda, oduzme zbroj iznosa grupa u svojstvu troška. Iznos grupe se računa tako da se zbroji svaki iznos stavke unutar grupe. Ukupan iznos i iznos svake grupe se ažuriraju svakim dodavanjem ili brisanjem stavke. Programsko rješenje za navedeni problem prikazano je na primjeru koda 14.

```
private void calculateTotal(Query query, View v) {
    txtIznosTotal = v.findViewById(R.id.iznosPlan);
    query.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            for (QueryDocumentSnapshot document : task.getResult()) {
                double total = Double.parseDouble(txtIznosTotal.getText()
                   .toString().trim());
                double iznosStavke;
                String svojstvo = String.valueOf(document.get("p1Svojstvo"));
                if (svojstvo.equals("prihod")) {
                    iznosStavke = Double.parseDouble(
                            String.valueOf(document.get("p1Iznos")));
                    total = total + iznosStavke;
                    txtIznosTotal.setText(formatIznos(total));
                } else {
                    iznosStavke = Float.valueOf(String.valueOf(document.get("p1Iznos")));
                    total = total - iznosStavke;
                    txtIznosTotal.setText(formatIznos(total));
                }
            }
        }
    });
}
```

Kod 14. Funkcija za izračun ukupnog iznosa

Brisanje grupa i stavki odvija se isto kao i brisanje grupa opisano u prethodnom poglavlju. Korisnik nove prihode dodaje klikom na zeleni gumb s oznakom „+“ , a nove troškove klikom na crveni gumb s oznakom „-“. Ovisno o tome dodaje li se prihod ili trošak korisniku se otvara pripadajući obrazac za unos. Nakon što korisnik unese podatke pozivaju se funkcije za provjeru podatka i ako je sve u redu stavka se unosi u bazu te se prikaz ažurira

komunikacijom baze i *Firestore adaptera*. Ako nešto nije u redu s formatom unosa korisnik dobiva upozorenje na polje za unos podatka, a ako nešto nije u redu s unošenjem u bazu korisnik dobiva *Snackbar* poruku o tome.

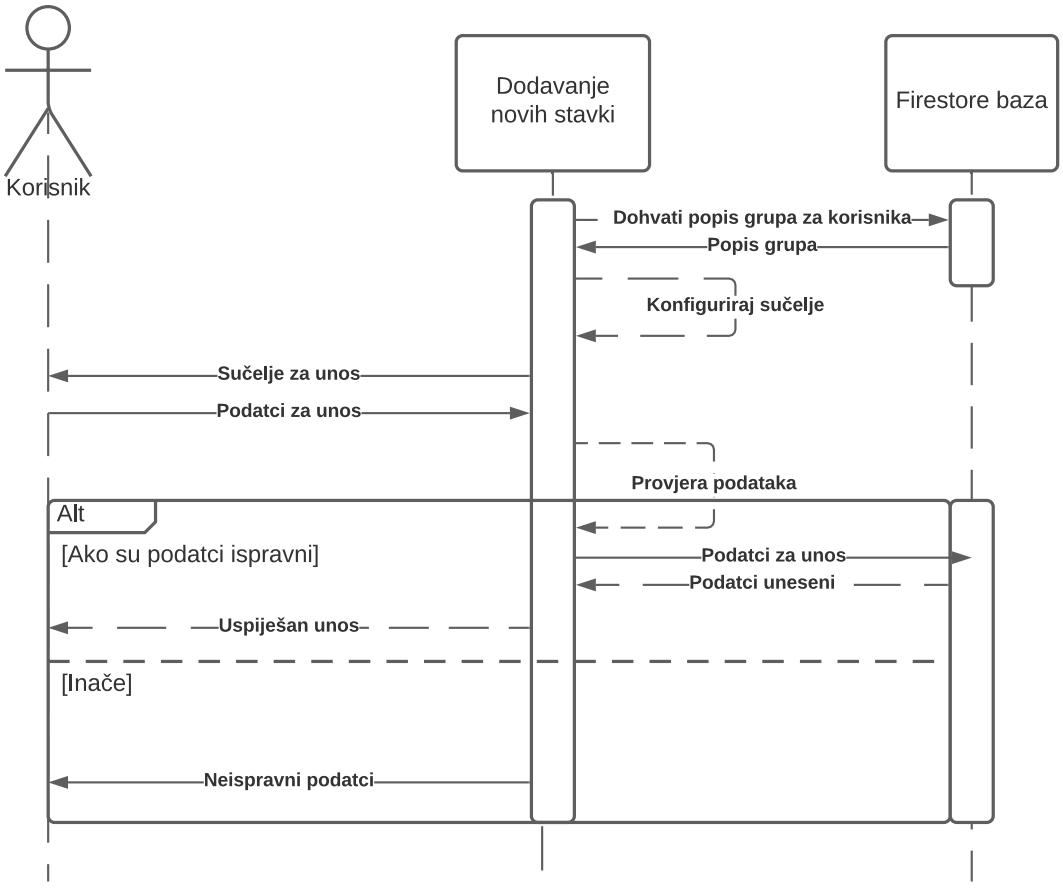
Kada je korisnik gotov s popunjavanjem plana klikom na gumb spremi id plana na uređaju se postavlja na „prazno“ što rezultira konfiguracijom sučelja na prikaz za unošenje novih planova i nema potrebe za dodatnim spremanjem zbog Stalne komunikacije *Firestore adaptera* i baze. Međutim ako korisnik nije zadovoljan planom ima mogućnost brisanja istog klikom na gumb odbaci. Klikom na taj gumb pokreće se složeni proces brisanja plana. Dio složenosti je u tome što za brisanje plana moramo obrisati i sve dokumente njegove potkolekcije. Kod funkcije za brisanje plana prikazan je na primjeru 15 .

```
private void deletePlan() {
    SharedPref sharedpref = new SharedPref(getActivity());

    CollectionReference deleteGroup = fStore.collection("korisnici")
            .document(sharedpref.loadUserID())
            .collection("planovi").document(sharedpref.loadPlanID())
            .collection("grupeNaPlanu");
    deleteGroup.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            for (QueryDocumentSnapshot document : task.getResult()) {
                deleteStavke(document.getId());
                DocumentReference dr1 = fStore.collection("korisnici")
                        .document(sharedpref.loadUserID())
                        .collection("planovi").document(sharedpref.loadPlanID())
                        .collection("grupeNaPlanu").document(document.getId());
                dr1.delete();
            }
            DocumentReference deletePlan = fStore.collection("korisnici")
                    .document(sharedpref.loadUserID())
                    .collection("planovi").document(sharedpref.loadPlanID());
            deletePlan.delete().addOnSuccessListener(new OnSuccessListener<Void>() {

                @Override
                public void onSuccess(Void aVoid) {
                    FragmentTransaction fragmentTransaction = getActivity()
                   .getSupportFragmentManager().beginTransaction();
                    fragmentTransaction.replace(R.id.container, KreiranjeFragment
                   .newInstance());
                    fragmentTransaction.addToBackStack(null);
                    fragmentTransaction.commit();
                    sharedpref.setPlanID("prazno");
                    timeSelectors.setVisibility(View.VISIBLE);
                    planComands.setVisibility(View.GONE);
                    sharedpref.setZgVisibility(true);
                }
            });
        }
    });
}
```

Kod 15. Funkcija za brisanje plana

U slučaju da se radi o kreiranju novog plana sučelje se konfigurira tako da korisnik mora odabrati razdoblje za koje će kreirati datum. Razdoblje odabire tako da u izborniku odabere identifikator plana i zatim mu se klikom na polje za unos razdoblja, otvara prozor za odabir razdoblja. Prozor za odabir razdoblja kreira se ovisno o odabranom identifikatoru, a identifikator može biti tjedni, mjesečni ili godišnji. Nakon odabira razdoblja u bazi se kreira plan te korisniku dolazi prikaz za uređivanje plana s prethodno navedenim mogućnostima.

5.2.6. Dodavanje nastalih troškova i prihoda

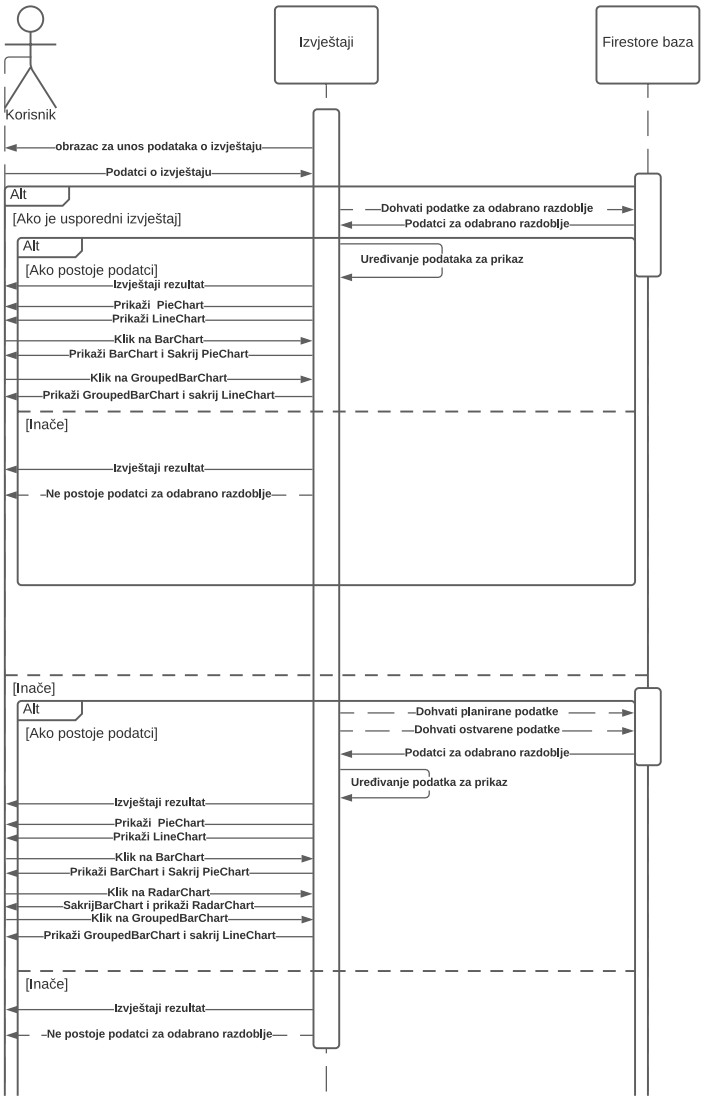


Slika 29. Sekvencijalni dijagram dodavanja novonastalih prihoda/troškova

Sekvencijalni dijagram na slici 29 prikazuje proces dodavanja novonastalih prihoda ili troškova. Proces započinje dohvaćanjem grupa prihoda i troškova iz baze podataka nakon čega se sučelje konfigurira u obrazac za unos novonastalog prihoda ili troška. Taj obraza je varijacija ranije opisanog dijaloga za unošenje stavke prihoda ili troška, jedina razlika je u tome gdje se podatci pohranjuju. U ranijem slučaju podatci su se pohranjivali u kolekciju stavke koja je bila potkolekcija dokumenata iz kolekcije grupe na planu, a u ovom slučaju podatci se pohranjuju u kolekciju stavke koja je potkolekcija dokumenata u kolekciji korisnici. Podatci se pohranjuju

u različite kolekcije zbog toga što se nalaze u različitim svojstvima. Ranije spomenuti podatci su predstavljali planirane stavke dok podatci koji se unose ovim obrascem spadaju u ostvarene stavke. Ta dva tipa stavaka se međusobno uspoređuju kod kreiranja izvještaja koji će biti prikazani u nastavku.

5.2.7. Kreiranje Izvještaja



Slika 30. Sekvencijalni dijagram izvještaja

Slika 30 prikazuje sekvencijalni dijagram generiranja izvještaja. Za generiranje izvještaja korištena je besplatna biblioteka za *Android studio* pod nazivom *MPA Android Charts*. Na početku procesa generiranja Korisnik odabire tip izvještaja koje želi generirati. I zatim odabire

početni datum i završni datum koji predstavljaju donju i gornju granicu vremena za koje se podatci promatraju. Gornja i donja granica važne su ukoliko korisnik odabere tip izvještaja „o prihodima“ ili tip izvještaja „o troškovima“. Ovi tipovi izvještaja uspoređuju planirane i ostvarene stavke kao što smo spomenuli u prethodnom poglavlju. Ako ne postoje planirane ili ostvarene stavke za odabrani raspon aplikacija javlja korisniku da ne postoje podatci za uneseno razdoblje. Ako postoje podatci generira se set grafova koji su prikazani unutar kartica. Prva kartica prikazuje općenite odnosno ukupne podatke. Na prvoj kartici za dva ranije navedena tipa izvještaja generiraju se:

- Tortni graf koji prikazuje odnos između ukupnog iznosa planiranih i ostvarenih stavki.
- Stupčasti graf koji prikazuje odnos između ukupnog iznosa planiranih i ostvarenih stavki.
- Radar graf prikazuje odnosi između ukupnog iznosa grupa planiranih i grupa ostvarenih stavki

Druga kartica prikazuje drugi set grafova koji prikazuju ukupan iznos planiranih i ostvarenih stavki s obzirom na datum nastanka. Grafovi unutar drugog seta mijenjaju vrijednosti i način prikaza ovisno o tome radi li se o tjednom i mjesečnom ili godišnjem izvještaju. Na drugoj kartici generiraju se:

- Linijski graf koji prikazuje razliku planirane – ostvarene stavke s obzirom na datum
- Grupirani Stupčasti graf koji prikazuje ukupan iznos planiranih i ostvarenih stavki s obzirom na datum nastanka

U slučaju da je korisnik odabrao tip grafa „usporedni“ tada se u obzir uzimaju samo ostvarene stavke ali tada se usporedba vrši prema svojstvu stavke, odnosno nalazi li se stavka u svojstvu prihoda ili troška. U slučaju odabira ovog tipa izvještaja generiraju se svi grafovi kao i u prethodnom slučaju osim Radar grafa koji tada nema smisla jer ne možemo uspoređivati grupe koje se nikada neće preklapati. Grafovi u ovom slučaju prikazuju razliku prihodi-troškovi. Kao i u prethodnom slučaju i ovdje korisnik prima poruku ako ne postoje podatci za odabrano razdoblje.

Neovisno o ranije navedenim tipovima izvještaja kod vizualizacije najvažnije je bilo pripremiti podatke za vizualizaciju. Na primjeru 16 prikazan je dio programskog koda koji kreira listu ostvarenih stavki ovisno o svojstvu. Ako se kod pozove u svojstvu prihoda tada će

upit dohvaćati samo stavke koje imaju poje su pohranjene kao prihod, u suprotnom upit će vraćati samo troškove. Nakon toga prolazi se kroz listu dokumenata i provjerava se odgovara li datum stavke uvjetu da bude veći ili jednak početnom datumu koji je korisnik na početku unio i manji ili jednak završnom datumu koji je korisnik unio. Ako stavka prođe navedene uvjete njezin naziv i iznos pohranjuju se u globalno deklariranu dvodimenzionalnu listu.

```
Query query = fStore.collection("korisnici").document(sharedpref.loadUserID())
        .collection("stavke").whereEqualTo("svojstvoStavke", "Prihod");
query.get().addOnCompleteListener(task -> {
    if (task.isSuccessful()) {
        List<List<String>> ostvareneStavke = new ArrayList<List<String>>();
        int y = 0;
        for (QueryDocumentSnapshot document : task.getResult()) {
            String iznosStavke, nazivStavke;
            String targetDate = String.valueOf(document.getString("datumStavke"));
            try {
                if (validateDatOd(startDate, endDate, targetDate)) {
                    nazivStavke = String.valueOf(document.get("idGrupe"));
                    iznosStavke = String.valueOf(document.get("iznosStavke"));
                    ostvareneStavke.add(new ArrayList<String>());
                    ostvareneStavke.get(y).add(nazivStavke);
                    ostvareneStavke.get(y).add(iznosStavke);
                    ostvareneStavke.get(y).add(targetDate);
                    y++;
                }
            } catch (ParseException e) {
                e.printStackTrace();
            }
        }
        getDatesPlanData(ostvareneStavke, startDate, endDate, generatorID, isYearly);
    }
});
```

Kod 16.Dio koda za kreiranje liste ostvarenih stavki

Na kraju prethodno opisanog procesa poziva se funkcija koja traži kreirani plan čiji početni i završni datum odgovaraju unesenom početnom i završnom datumu. Kod te funkcije prikazan je na primjeru 17. Ako nađe takav plan, tada se izvršava upit koji dohvaća sve grupe na tom planu. Nakon toga za svaku grupu se poziva funkcija koja će dodati stavke te funkcije.

```
public void getDatesPlanData(List<List<String>> ostvareneStavke, String startDate, String
endDate, String generatorID, Boolean isYearly) {
    if (generatorID.equals("prihod")) {
        Query queryPlan = fStore.collection("korisnici").document(sharedpref.loadUserID())
                .collection("planovi");
        queryPlan.get().addOnCompleteListener(task12 -> {
            if (task12.isSuccessful()) {
                for (QueryDocumentSnapshot document1 : task12.getResult()) {
                    if (startDate.equals(document1.get("pocetniDatum")) &&
                            endDate.equals(document1.get("zavrsniDatum"))) {
                        Query stavkePlana =
                        fStore.collection("korisnici").document(sharedpref.loadUserID())
                        .collection("planovi").document(document1.get("idPlana")
                        .toString().trim())
                        .collection("grupeNaPlanu").whereEqualTo("p1Svojstvo", "prihod");

                            stavkePlana.get().addOnCompleteListener(task13 -> {
                            if (task13.isSuccessful()) {
                                for (QueryDocumentSnapshot document2 :
                                      task13.getResult()){
                                    getDatesPlanStavke(document1, document2,
                                    ostvareneStavke, startDate, endDate, generatorID);

                                }
                            }
                        });

                    }
                }
            }

        });
    }
```

Kod 17. Upit za dohvaćanje grupa na planu

Nakon toga za svaku grupu se poziva funkcija koja će dodati datum stavke, iznos stavke i id grupe te stavke u globalno deklariranu dvodimenzionalnu listu planirane stavke. Kod za tu funkciju prikazan je na slici 18.

```
public void getDatesPlanStavke(QueryDocumentSnapshot document1, QueryDocumentSnapshot
document2, List<List<String>> ostvareneStavke, String startDate, String endDate, String
generatorID) {

    Query stavkeGrupe = fStore.collection("korisnici").document(sharedpref.loadUserID())
            .collection("planovi").document(document1.get("idPlana").toString().trim())
            .collection("grupeNaPlanu").document(document2.getId()).collection("stavke");
    stavkeGrupe.get().addOnCompleteListener(task -> {
        List<List<String>> planiraneStavkeTemp = new ArrayList<List<String>>();
        int y = 0;
        for (QueryDocumentSnapshot document3 : task.getResult()) {
            String iznosStavke = String.valueOf(document3.get("iznosStavke"));
            String nazivStavke = String.valueOf(document3.get("idGrupe"));
            String targetDate = String.valueOf(document3.getString("datumStavke"));
            planiraneStavkeTemp.add(new ArrayList<String>());
            planiraneStavkeTemp.get(y).add(nazivStavke);
            planiraneStavkeTemp.get(y).add(iznosStavke);
            planiraneStavkeTemp.get(y).add(targetDate);
            y++;
        }
        planiraneStavke.addAll(planiraneStavkeTemp);
    });
}
```

Kod 18.Dio kod za generiranje liste planiranih prihoda

Nakon što su generirane liste s planiranim i ostvarenim stavkama, ovisno o grafu koji se generira pozivaju se funkcije koje dodatno prilagođavaju podatke u listama za dodatni prikaz i nakon toga funkcije za generiranje grafova. Na primjeru 19 nalazi se dio koda koji generira tortni graf prilagođen za prikaz u noćnom načinu, kada je identifikator za generiranje grafa prihod, odnosno kada se generira tortni graf koji prikazuje planirane i ostvarene troškove.

```
public void generateBarTotal(String generatorID, ArrayList<BarEntry> ukupno) {
    BarDataSet barDataSet = new BarDataSet(ukupno, "");
    barDataSet.setValueTextSize(15f);
    if (generatorID.equals("prihod")) {
        if (sharedpref.loadNightModeState() == true) {
            barDataSet.setColors(dColors);
            barDataSet.setValueTextColor(Color.WHITE);
            BarData barData = new BarData(barDataSet);
            oBarChart.setData(barData);
            oBarChart.getDescription().setEnabled(false);
            Legend legend = oBarChart.getLegend();
            legend.setTextColor(Color.WHITE);
            LegendEntry l1 = new LegendEntry("Ostvareni prihod",
                    Legend.LegendForm.CIRCLE, 10f, 2f, null, Color.rgb(0, 255, 255));
            LegendEntry l2 = new LegendEntry("Planirani prihod", Legend.LegendForm.CIRCLE,
                    10f, 2f, null, Color.rgb(0, 0, 0));
            legend.setCustom(new LegendEntry[]{l1, l2});
            legend.setWordWrapEnabled(true);
            XAxis xAxis = oBarChart.getXAxis();
            xAxis.setTextColor(Color.TRANSPARENT);
            YAxis leftAxis = oBarChart.getAxisLeft();
            YAxis rightAxis = oBarChart.getAxisRight();
            rightAxis.setEnabled(false);
            leftAxis.setTextColor(Color.WHITE);
            oBarChart.invalidate();
            oBarChart.animate();
        }
```

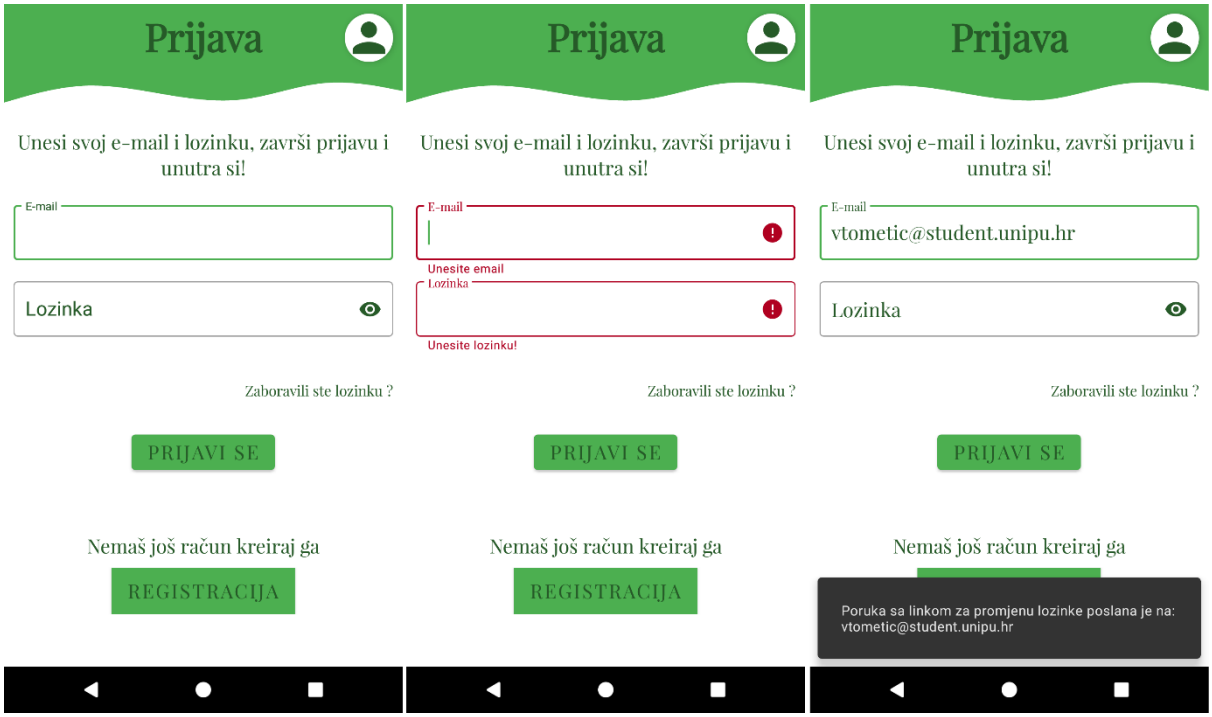
Kod 19. Dio koda za generiranje tortnog grafa

Ranije prikazane funkcije su gotovo identične kada se generira usporedni tip grafa razlika je što se onda ne poziva funkcija za dohvaćane planiranih stavaka već se poziva funkcija za dohvaćanje ostvarenih prihoda i nakon toga funkcija za dohvaćanje ostvarenih troškova. Još jedna od značajnijih razlika javlja se kada je riječ o generiranju drugog seta grafova, tada se još pozivaju funkcije koje dodatno uređuju os x i u slučaju generiranja linijskog grafa funkcija koja oduzima planirane od ostvarenih stavaka, odnosno ostvarene prihode od ostvarenih troškova.

6. Korisničke upute

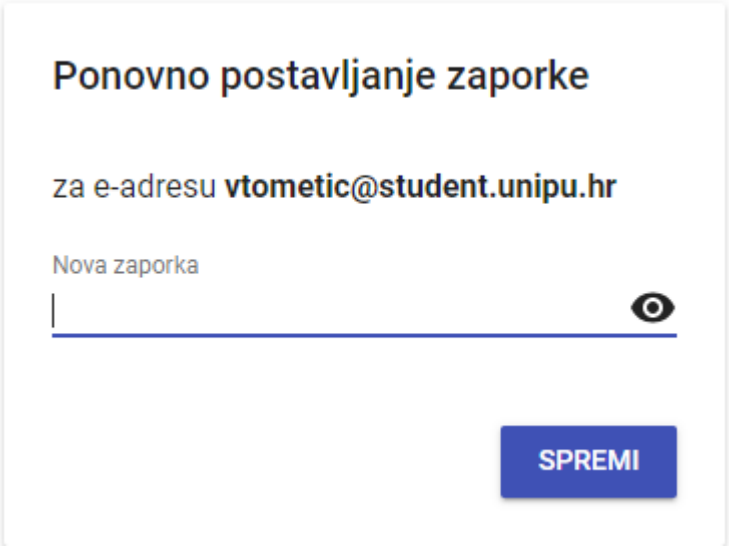
6.1. Registracija i prijava

Nakon pokretanja aplikacije, ako se radi o korisniku koji nije prijavljen, otvara se stranica za prijavu, kao što je prikazano na slici 31. U slučaju da korisnik ima postojeći račun, prijavljuje se, te ga aplikacija prebacuje na početnu stranicu. Ako se radi o novom korisniku, korisnik se može lako prebaciti na obrazac za registraciju klikom na gumb registracija.



Slika 31. Prijava

U slučaju da je korisnik zaboravio lozinku na stranici za prijavu ima mogućnost ponovno ju postaviti klikom na tekst „Zaboravili ste lozinku?“. Nakon čega se korisniku na upisani e-mail šalje poruka s linkom za ponovno postavljanje lozinke, te ga se obavještava o tome. Klikom na taj link korisniku se otvara obrazac za ponovno postavljanje lozinke kao što je prikazano na slici 32.

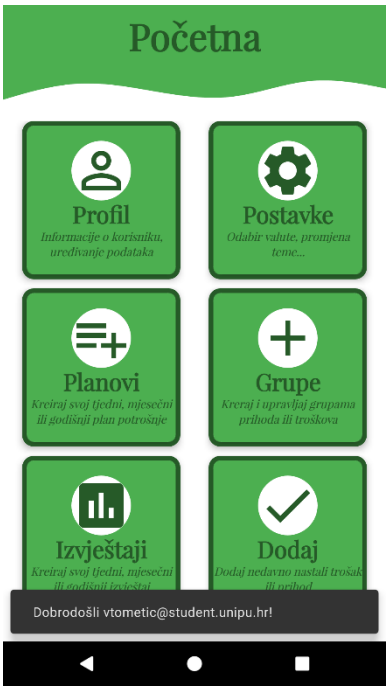


Slika 32. Ponovno postavljanje lozinke

Prilikom registracije korisnik unosi podatke kako bi se registrirao kao novi korisnik. Ako se korisnik bez ispunjenih polja pokuša registrirati na svako polje zasebno mu dolazi upozorenje. Ako vrijednosti unesene u polja ne zadovoljavaju traženi format, primjerice lozinka mora biti duža od 8 znakova ili ponovljena lozinka mora biti identična lozinki tada mu se ispisuju upozorenja samo na poljima s greškom, kao što je prikazano na slici 33. Ako je sve u redu korisniku se kreira račun te ga se preusmjerava na prijavu.

Slika 33. Registracija

Nakon uspješne prijave korisniku se otvara početna stranica i ispisuje poruka dobrodošlice kao što je prikazano na slici. Početna stranica će se otvarati umjesto prijave i ako se korisnik ne odjavi. Slika 34 prikazuje početnu stranicu. Na početnoj stranici nalazi se 6 kartica koje vode na najvažnije funkcionalnosti aplikacije. Početna stranica ujedno služi i kao glavni izbornik.

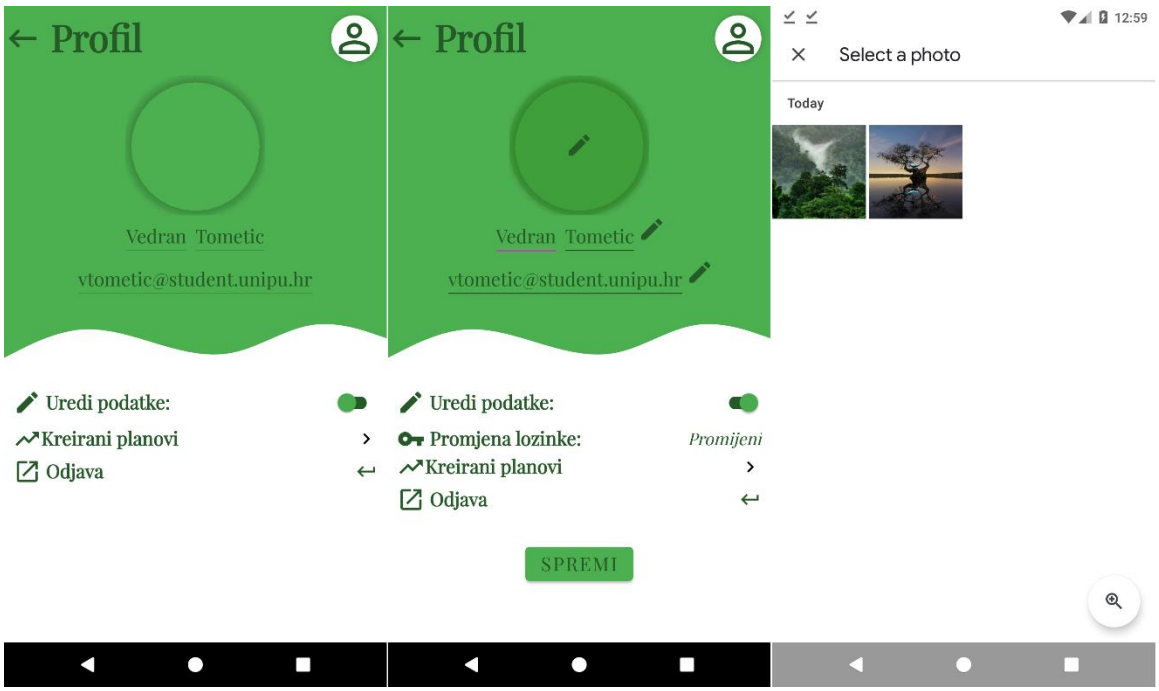


Slika 34. Početna stranica

6.2. Profil i upravljanje korisničkim podatcima

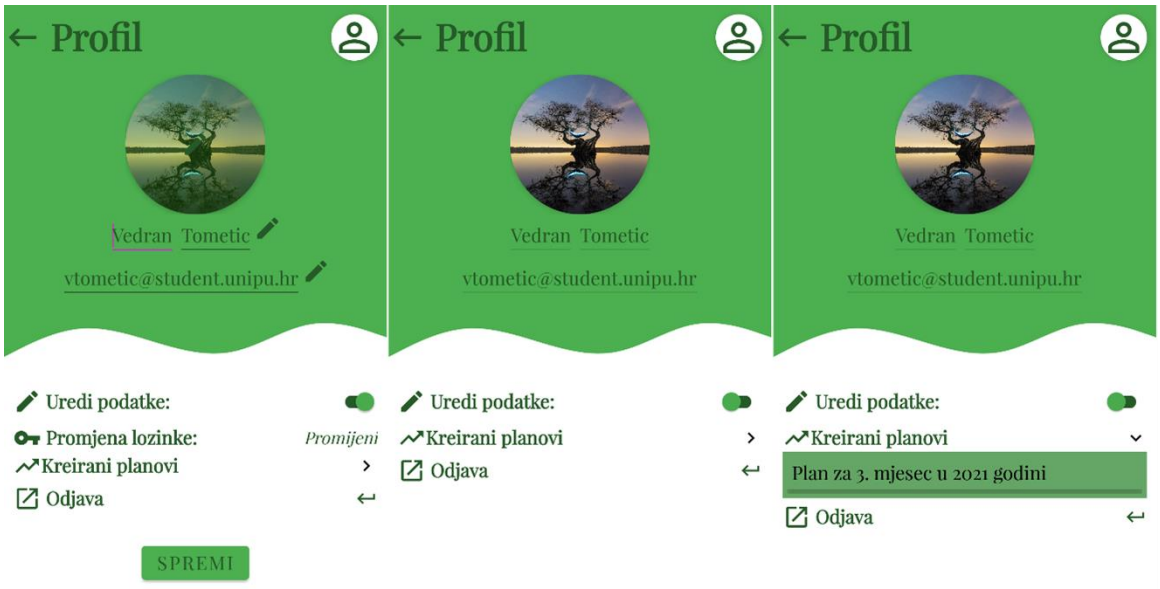
Na sučelje profila korisnik dolazi tako da na početnim zaslonu odabere karticu profil, nakon čega se otvara sučelje profila. Ovdje korisnik ima mogućnost urediti svoje ime, prezime, promijeniti e-mail, dodati sliku ili promijeniti lozinku. Kako bi to napravio korisnik treba uključiti prekidač *(eng. switch)* pod nazivom „uredi podatke“ nakon čega se pojavljuje gumb za spremanje promjena, opcija za promjenu lozinke, te se kod imena, prezimena , e-maila i okvira za fotografiju pojavljuje ikona olovke. Klikom na tu ikonu ili na podatak koji želi promijeniti korisnik pristupa modu za uređivanje. Ako korisnik želi spremiti podatke to čini klikom na gumb spremi, a ako ne želi tada samo isključuje prekidač „uredi podatke“. Korisnik

se u bilo kojem trenutku može vratiti na početnu stranicu klikom na strelicu u lijevom dijelu zaglavlja. Sučelje profila prikazano je na slici 35 i slici 36.



Slika 35. Profil (uređivanje podatka

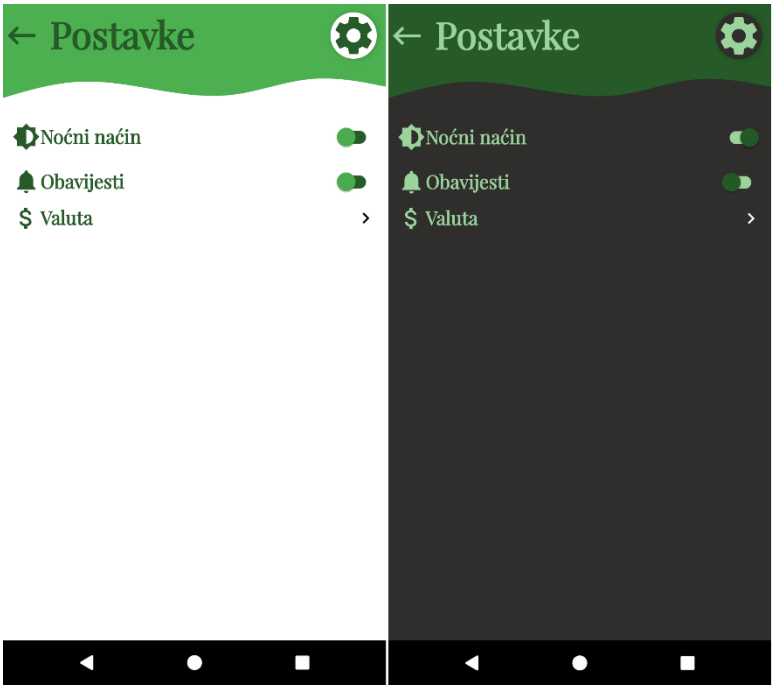
Osim navedenih mogućnosti, na sučelju profila korisnik može i pristupiti popisu do sada kreiranih planova, klikom na određeni plan korisnika se preusmjerava na stranicu planiranje, na koju se učitava odabrani plan. Sučelje za kreiranje plana detaljnije je opisano u poglavlju 6.5.



Slika 36. Profil (mogućnosti

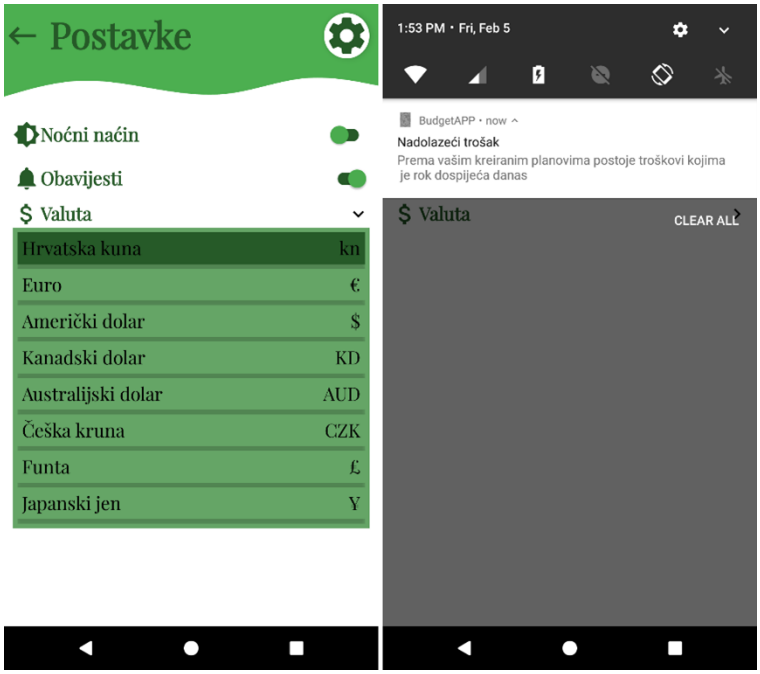
6.3. Postavke

Na sučelje postavke korisnik dolazi tako da na početnim zaslonu odabere karticu postavke, nakon čega se otvara sučelje postavke. U okviru postavki korisnik ima mogućnost uključivanja i isključivanja noćnog načina rada, obavijesti te odabira valute koju želi da mu se prikazuje prilikom kreiranja planova. Na slici 37 prikazana je razlika između noćnog načina i običnog načina.



Slika 37. Postavke (noćni način)

Klikom na prekidač „Obavijesti“ korisnik pokreće proceduru obavještavanja. Korisnik prima obavijesti tako dugo dok su ispunjeni sljedeći uvjeti : prekidač za obavijesti mora biti uključen i korisnik mora redovito koristiti aplikaciju. Obavijesti će se prestati pojavljivati ako korisnik duže vrijeme ne koristi aplikaciju. Klikom na izbornik valute korisniku se prikazuje popis dostupnih valuta. Klikom na određenu valutu korisnikov odabir se sprema i ostaje isti tako dugo dok korisnik opet ne promjeni valutu. Na slici 38 vidimo primjer obavijesti koju korisnik prima i izbornika za odabir valute.



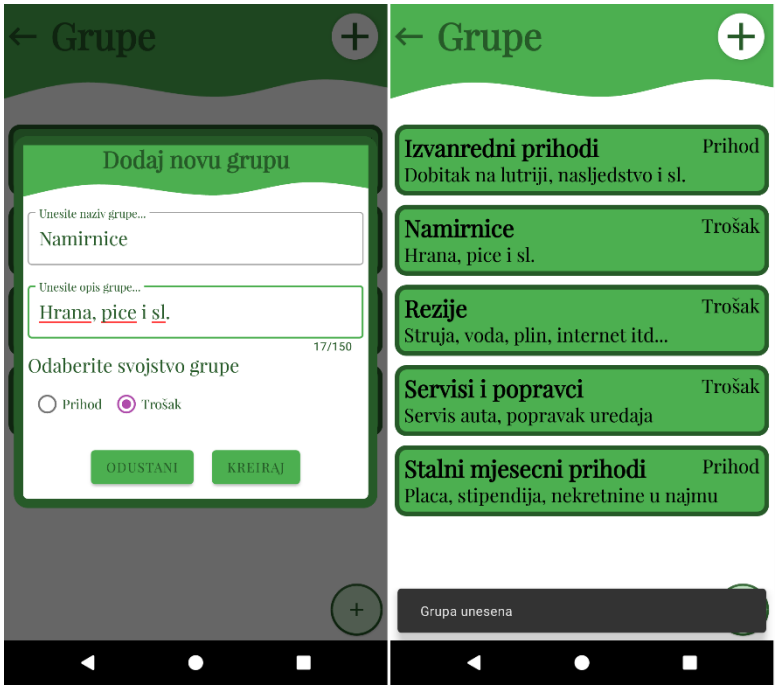
Slika 38. Postavke (mogućnosti)

6.4. Kreiranje grupa

Sljedeće u nizu sučelja kojima korisnik može pristupiti je sučelje za kreiranje grupa prihoda i troškova, koje prikazano na slici 39. Do tog sučelja korisnik dolazi odabirom kartice grupe na početnom zaslonu. Sučelje sadrži popis kreiranih grupa. Korisnik ovdje ima mogućnost uređivanja postojećih grupa klikom na stavku s popisa ili brisanja grupa povlačenjem stavke u desnu ili lijevu stranu.

Slika 39.Grupe (uređivanje postojećih)

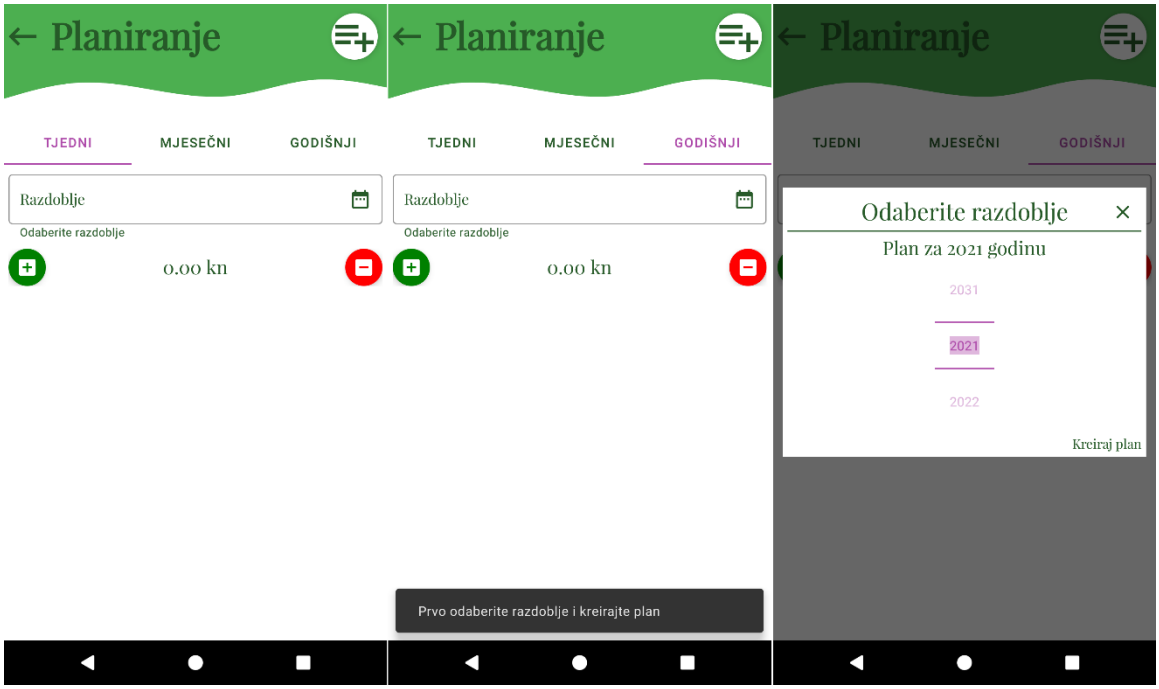
U slučaju da ne postoji niti jedna kreirana grupa sučelje prikazuje samo gumb na dnu stranice koji služi za kreiranje nove grupe. Klikom na taj gumb korisniku se otvara prozor unos podataka na temelju kojih se kreira grupa. Nakon što unese odgovarajuće podatke klikom na gumb kreiraj podatci se unose u bazu podataka, prozor za unos se zatvara i u slučaju da je kreiranje uspješno izvršeno korisniku dolazi obavijest o kreiranju grupe i ta grupa se pojavljuje na popisu. Opisani proces prikazan je na slici 40.



Slika 40. Grupe (dodavanje novih)

6.5. Kreiranje planova

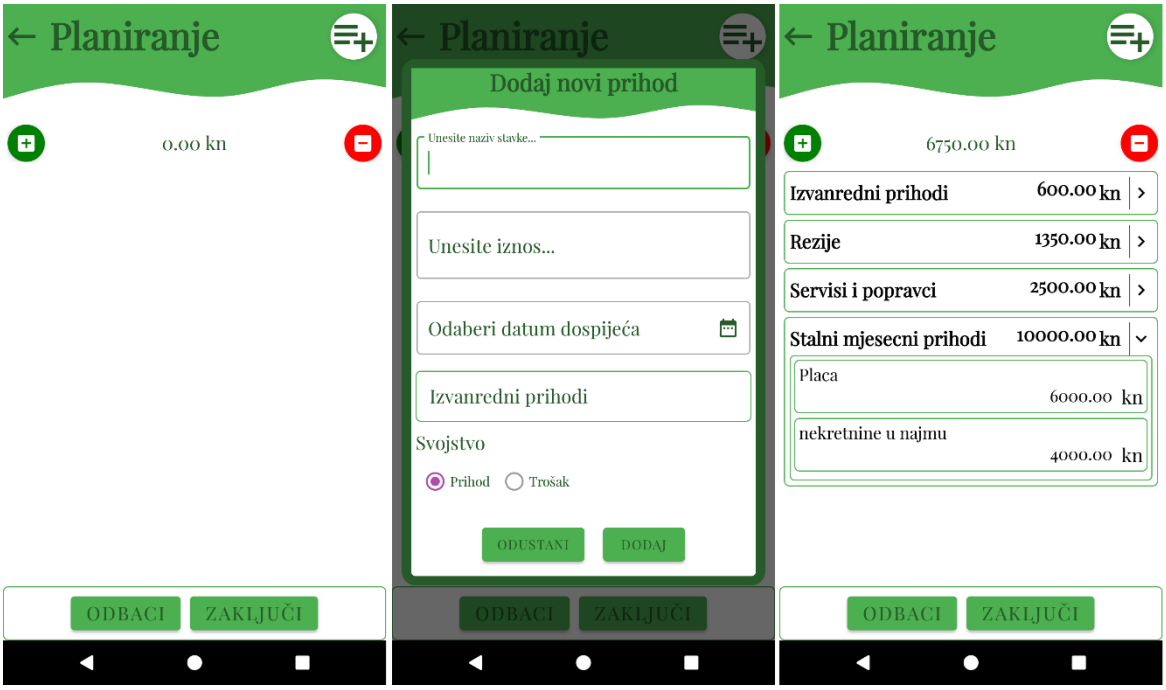
Sučelje za kreiranje planova, uz sučelje za kreiranje izvještaja, predstavlja jedno od najvažnijih sučelja ove aplikacije. Korisnik mu pristupa odabirom kartice planovi na početnom zaslonu ili odabirom plana s popisa već kreiranih planova u okviru sučelja profil. Ako ne postoji kreirani plan korisniku se prikazuje prazno sučelje s izbornikom na kojem može odabrati želi li kreirati tjedni, mjesečni ili godišnji plan, polje za unos razdoblja, i gumbi za dodavanje prihoda ili troškova. Ako korisnik klikne na jedan od tih gumba prije nego što je odabrao razdoblje dolazi mu obavijest kako prvo mora odabrati razdoblje. Klikom na polje za unos razdoblja, ovisno o odabranoj stavci u izborniku, korisniku se otvara novi prozor u okviru kojeg odabire razdoblje kao što je prikazano na slici 41. Nakon što je odabrao razdoblje korisnik klikom na tekst „Kreiraj plan“ poziva proceduru koja kreira plan i pohranjuje ga u bazu.



Slika 41. Kreiranje plana

Nakon što je plan kreiran izbornik za odabir kategorije plana i polje za unos razdoblja nestaju i ostaju samo gumbi za dodavanje prihoda i tekst između njih koji predstavlja razliku između prihoda i troškova. Iznos u tekstu se mijenja dodavanjem ili brisanjem stavki prihoda i troškova. Klikom na gumb „dodaj prihod“ ili „dodaj trošak“ korisniku se otvara prozor za unos podataka važnih za stavke plana.

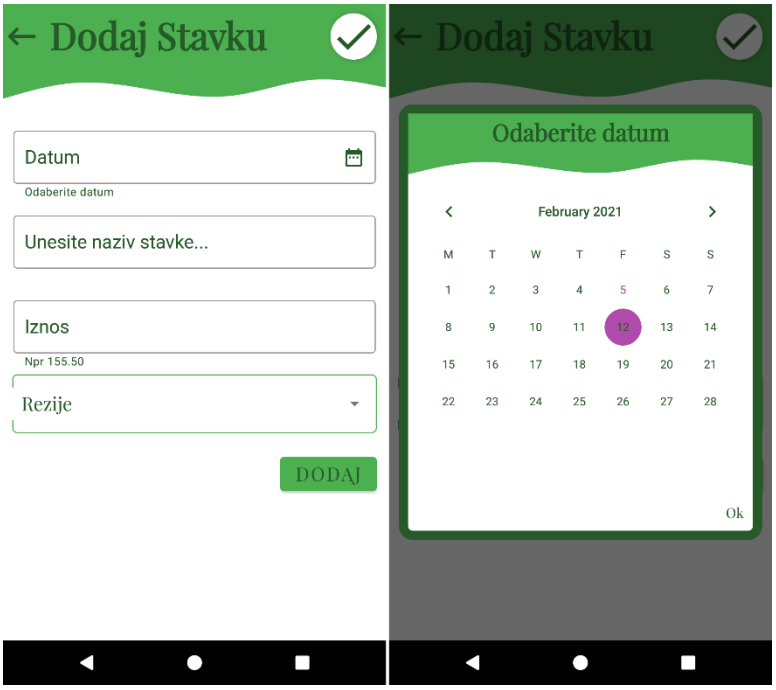
Nakon što popuni polja i klikne na gumb dodaj stavka se pakira u odabranu grupu, prikazuje na popisu, te se izračunava ukupni iznos. Popis u okviru ovog sučelja prikazuje naziv grupe prihoda ili troškova i ukupan iznos svake pojedine grupe. Klikom na grupu otvara se dodatan popis koji sadrži sve nazive stavki pripadajuće grupe i njihove iznose, kao što je prikazano na slici 42.



Slika 42. Kreiranje plana (dodavanje stavki i prikaz)

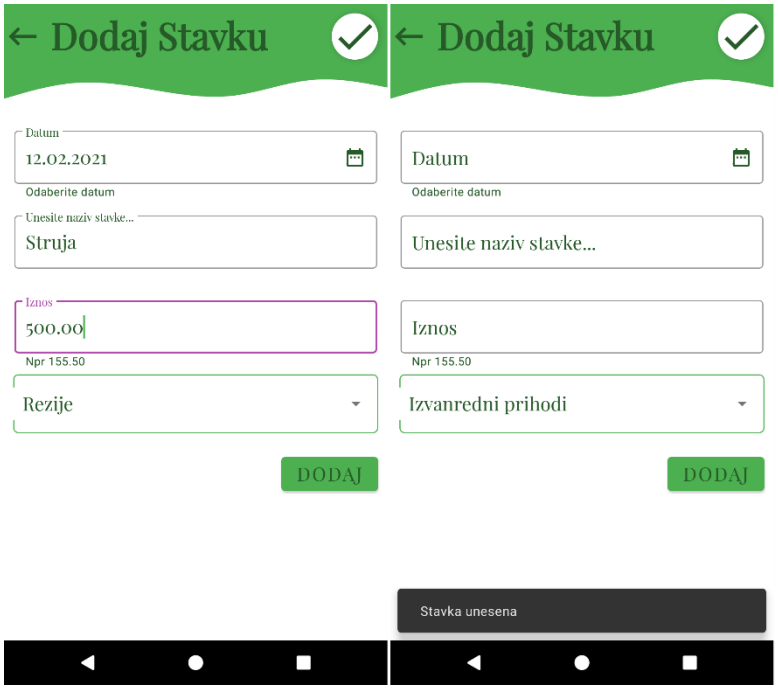
6.6. Dodavanje nastalih prihoda i troškova

Sučelju za dodavanje nastalih prihoda ili troškova korisnik pristupa odabirom kartice dodaj na početnom zaslonu. Sučelje za dodavanje novonastalih prihoda ili troškova je u obliku obrasca za unos i sadrži polja za odabir datuma, odabir grupe te polja za unos naziva i iznosa stavke. Klikom na polje za odabir datuma korisniku se otvara prozor za odabir datuma, klikom na polje za odabir grupe korisniku se prikazuje popis kreiranih grupa. Na temelju odabrane grupe aplikacija određuje svojstvo stavke koja se unosi. Opisano sučelje prikazano je na slici 43.



Slika 43. Obrazac za dodavanje novih prihoda i troškova

Nakon popunjavanja polja klikom na gumb dodaj, korisnik unosi stavku u bazu i ako je unos uspješan korisnik se obavještava o tome, kao što je prikazano na slici 44. U slučaju da je neko od polja prazno korisniku se postavlja upozorenje na to polje.



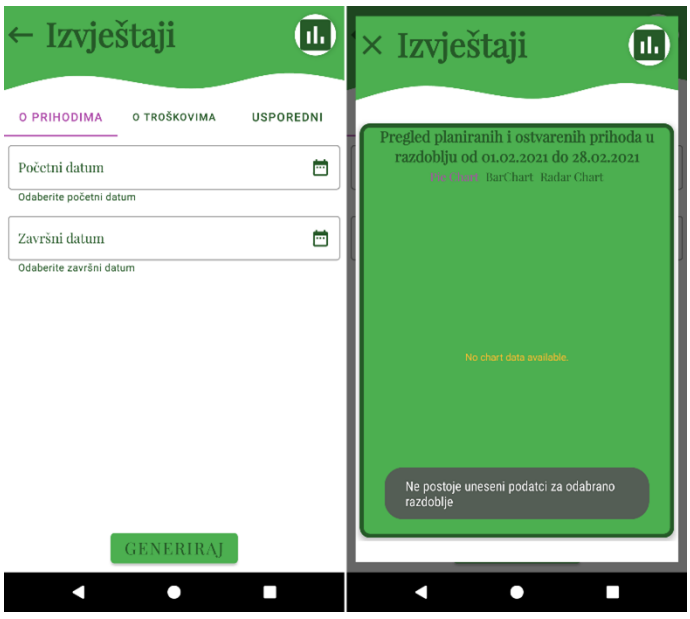
Slika 44. Unos novonastalog prihoda/troška

6.7. Kreiranje izvještaja

Kreiranje izvještaja je sučelje na kojem korisnik ima mogućnost vizualizirati svoje podatke koje je na prethodno opisanim sučeljima unosio. Kreiranju izvještaja korisnik pristupa odabirom kartice izvještaji na početnom zaslonu. Nakon toga korisniku se otvara sučelje na kojem u izborniku može odabrati kakvu vrstu izvještaja želi kreirati. U trenutnoj verziji aplikacije korisnik može odabrati između tri mogućnosti :

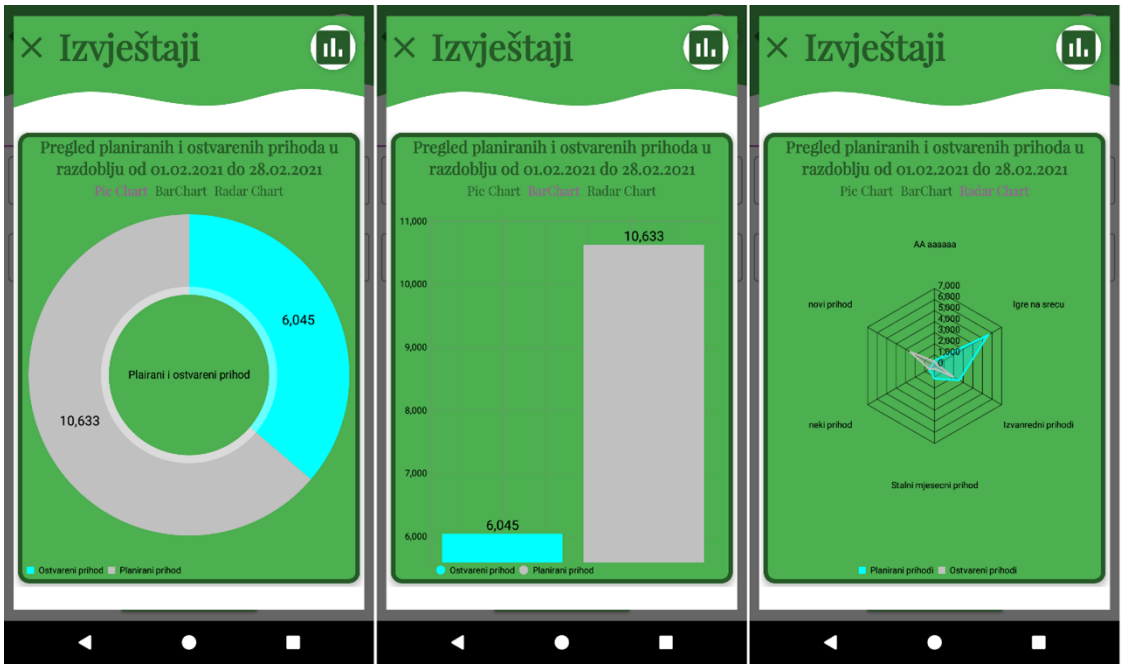
- Izvještaji na kojima se uspoređuju planirani i ostvareni prihodi
- Izvještaji na kojima se uspoređuju planirani i ostvareni troškovi
- Izvještaji na kojima se uspoređuju ostvareni prihodi i troškovi

U slučaju da korisnik ne odabere datume a pritisne gumb za generiranje izbacit će se greška na poljima za unos i prozor s rezultatima se neće otvoriti. Ako se radi o izvještajima koji uspoređuju planirane i ostvarene prihode ili troškove, moraju prethodno postojati generirani planovi za odabrano razdoblje u protivnom se korisniku otvara prozor s rezultatima, ali mu se ispisuje poruka da ne postoje uneseni podatci za odabrano razdoblje. Taj slučaj prikazan je na slici 45.



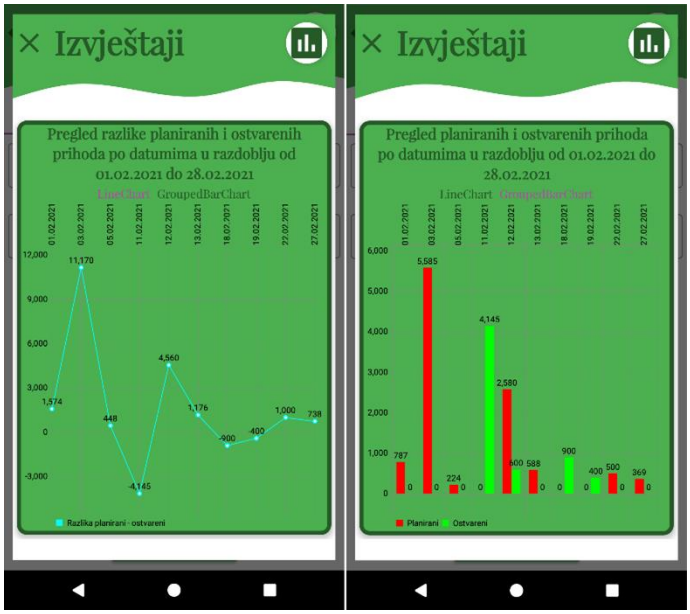
Slika 45. Izvještaji (pogrešni unosi)

Kada korisnik ispravno unese sve podatke kreiraju se grafikoni kao na slikama 46 i 47. Na slici 46 prvi set grafova koji se kreira, a on uključuje varijaciju tortnog i stupčastog grafa koji prikazuju odnos ukupnog iznosa planiranih i ostvarenih troškova neovisno o odabranoj opciji u izborniku. Treći graf na slici 46 je radar graf koji se kreira samo u slučaju kada su u izborniku odabrane opcija „O prihodima“ ili opcija „O troškovima“. Ovaj graf prikazuje sume iznosa stavki planiranih i ostvarenih prihoda ili troškova, kategorizirane po kreiranim grupama. Ovaj graf ne generira se u slučaju kada je odabrana opcija „Usporedni“ jer ta opcija generira grafove koji uspoređuju ostvarene prihode i troškove, a oni nikad ne spadaju u iste grupe tako da ih nema smisla uspoređivati po grupama. Primjerice neki trošak i neki prihod nikada neće spadati u istu grupu jer grupa može biti isključivo u svojstvu prihoda ili isključivo u svojstvu troška.



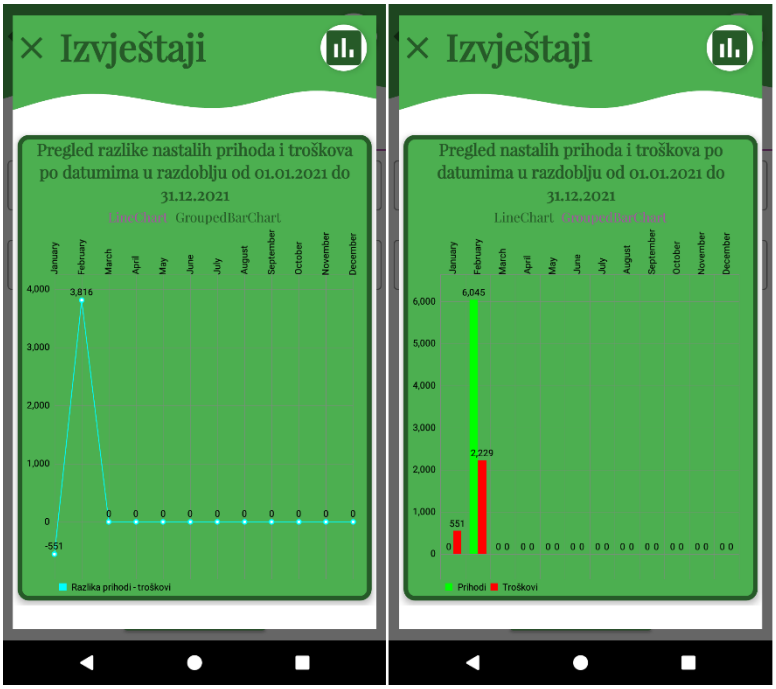
Slika 46. Izvještaji (prvi set grafova)

Drugi set grafova koji se kreiraju uključuje linijski i grupirani stupčasti graf, a prikazani su na slici 47. Ovaj set grafova kreira se neovisno o opciji odabranoj u izborniku. Linijski graf koji se kreira prikazuje razliku između prihoda i troškova po datumima nastanka u slučaju da je odabrana opcija „Usporedni“ u izborniku ili razliku između planiranih ako je odabrana opcija „O prihodima“ ili opcija „O troškovima“. Drugi graf u setu grupirani stupčasti graf koji prikazuje iznos prihoda ili troškova u slučaju opcije „Usporedni“ ili iznos planiranih i ostvarenih prihoda ili troškova ovisno o odabranoj opciji.



Slika 47. Izvještaji (drugi set grafova, mjesečni)

Međutim kod ovog seta grafova postoji razlika ovisno o rasponu odabranom na početku. U slučaju da se radi o tjednoj ili mjesečnoj usporedbi na os x postavljaju se datumi kao što možemo vidjeti na slici 47 gdje je generiran mjesečni izvještaj. Drugi slučaj je da se radi o godišnjem izvještaju tada se pozivaju dodatne funkcije koje grupiraju podatke po mjesecima te ih tako prikazuju, što je prikazano na slici 48.



Slika 48. Izvještaji (drugi set grafova, godišnji)

7. Zaključak

Iz ovog diplomskog rada možemo zaključiti kako vizualizaciju podataka nije jednostavno definirati, ali nije nemoguće. Vizualizacija podataka je područje na presjeku mnogih drugih kao što su matematike, fizike, informacijskih tehnologija i mnogih drugih. Iako, u usporedbi s nekim drugim područjima, vizualizacija podataka nema dugu povijest primjene, vizualizacija podataka se čini kao jedno od područja koje će igrati ključnu ulogu u budućnosti. Primjerice, nedvojbeno možemo zaključiti kako će vizualizacija podataka i u budućnosti igrati veliku ulogu zbog sve većeg broja podataka iz kojih je nekako potrebno izvući informacije, a vizualizacija je možda i najatraktivniji način za to. Osim toga vizualizacija podataka će igrati veliku ulogu i u konceptu pametnih gradova koji se, već sad, sve više počinje primjenjivati u praksi.

Za razvoj aplikacije je odabran android studio koji koristi java programski jezik i *xml* za implementaciju aplikacije. Za pohranu podataka koje aplikacija generira odabrana je Cloud *Firestore* baza podataka zbog svoje *NoSQL* strukture koja uvelike olakšava pristupanje podatcima. Razvijenu aplikaciju mogu koristiti isključivo registrirani korisnici, Aplikacija korisnicima nudi mogućnost generiranja vlastitih planova za raspodjelu prihoda i troškova, dodavanje novonastalih prihoda i troškova, koje kasnije korisnik može vizualizirati u modulu generiranja izvještaja. Za razliku od ostalih aplikacija na tržištu, ova aplikacija, daje korisnicima slobodu generiranja vlastitih grupa za pohranu prihoda i troškova. Ta mogućnosti bi trebala privući korisnike, jer im pruža visoku razinu personalizacije planova i izvještaja, te iam olakšava njihovo shvaćanje. Aplikacija ima svojih nedostataka, no svakim danom testiranja dolazi se do novih ideja za daljnji razvoj aplikacije, koje će se u budućnosti implementirati.

Cilj izrade aplikacije bio je izraditi aplikaciju koja korisnicima nudi mogućnost kreiranja vlastitih planova za raspolaganje kućnim budžetom te kreiranje izvještaja na temelju kojih korisnik može vidjeti razliku između planiranih i ostvarenih troškova i prihoda za odabrano razdoblje. Još jedan od ciljeva ove aplikacije je potaknuti korisnike na razmišljanje o vlastitim prihodima i troškovima te im prikazati važnost planiranja osobnog budžeta.

Literatura

Knjige

1. Sinar, E. F. (2015)., Data visualization. *Big data at work: The data science revolution and organizational psychology*, Izdavač: Routledge.
2. Friendly, M. (2008). A brief history of data visualization. In *Handbook of data visualization* (pp. 15-56). Springer.
3. Chen, C. H., Härdle, W. K., & Unwin, A. (Eds.). (2007). *Handbook of data visualization*. Springer Science & Business Media.
4. Iafrate, F. (2015). From big data to smart data (Vol. 1). John Wiley & Sons.

Članci

1. Unwin, A. (2020). Why is Data Visualization Important? What is Important in Data Visualization?. *Harvard Data Science Review*, *2*(1)., Dostupno na: https://hdsr.duqduq.org/pub/zok97i7p/release/1, [Pristupljeno 30.01.2021]
2. Friendly, M., Sigal, M., & Harnanansingh, D. (2012). The Milestones Project: a database for the history of data visualization. *Visible Numbers: The History of data Visualization. Ashgate Press, London.*, Dostupno na : https://www.datavis.ca/papers/MilestonesProject.pdf [Pristupljeno 01.02.2021]
3. Shah, D. K. (2016). Impact of Visualization on Engineers–A Survey. Dostupno na: (https://scholarcommons.usf.edu/cgi/viewcontent.cgi?article=7581&context=etd) [Pristupljeno 01.02.2021]
4. Keim, D., Qu, H., & Ma, K. L. (2013). Big-data visualization. IEEE Computer Graphics and Applications, 33(4), 20-21. Dostupno na: (https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6562707) [Pristupljeno 01.02.2021]
5. Cox, M., & Ellsworth, D. (1997). Managing big data for scientific visualization. In ACM siggraph (Vol. 97, pp. 21-38). Dostupno na: (https://www.researchgate.net/profile/David_Ellsworth2/publication/238704525_Managing_big_data_for_scientific_visualization/links/54ad79d20cf2213c5fe4081a/Managing-big-data-for-scientific-visualization.pdf) [Pristupljeno 01.02.2021]

Internet izvori

1. Ackoff , Russell L. From data to wisdom. *Journal of applied systems analysis*, 1989, 16.1: 3-9., Dostupno na: (https://softwarezen.me/wp-content/uploads/2018/01/datawisdom.pdf), [Pristupljeno 30.01.2021]
2. Microsoft, https://support.microsoft.com [Pristupljeno 01.02.2021]
3. Digiteum, https://www.digiteum.com/data-visualization-techniques-tools [Pristupljeno 01.02.2021]
4. Manageengine, https://www.manageengine.com/web-analytics/free-website-heat-map-analysis-tool.html [Pristupljeno 01.02.2021]
5. Digiteum, https://www.digiteum.com/data-visualization-your-business [Pristupljeno 01.02.2021]

6. García-Gil, D., Luengo, J., García, S., & Herrera, F. (2019). Enabling smart data: noise filtering in big data classification. Information Sciences, 479, 135-152. Dostupno na: (https://www.sciencedirect.com/science/article/abs/pii/S0020025518309460) [Pristupljeno 01.02.2021]
7. Khan, M., Shah, A., & Ahmad, I. (2014). Framework for interactive data mining results visualization on mobile devices. *International Journal of Database Theory and Application*, *7*(4), 23-36., Dostupno na: (https://www.researchgate.net/profile/Muzammil_Khan5/publication/264623460_Framework_for_Interactive_Data_Mining_Results_Visualization_on_Mobile_Devices/links/54128d630cf2bb7347daeabf/Framework-for-Interactive-Data-Mining-Results-Visualization-on-Mobile-Devices.pdf)[Pristupljeno 02.02.2021]
8. Blumenstein, K., Wagner, M., Aigner, W., von Suess, R., Prochaska, H., Püringer, J., ... & Sedlmair, M. (2015). Interactive data visualization for second screen applications: State of the art and technical challenges. *Proc. Int. Summer School on Visual Computing*, 35-48., Dostupno na : (https://www.researchgate.net/profile/Kerstin_Blumenstein/publication/293301207_Interactive_Data_Visualization_for_Second_Screen_Applications_State_of_the_Art_and_Technical_Challenges/links/56b713ff08ae3c1b79addbe9.pdf)[Pristupljeno 02.02.2021]
9. Sadiku, M., Shadare, A. E., Musa, S. M., Akujuobi, C. M., & Perry, R. (2016). Data visualization. International Journal of Engineering Research And Advanced Technology (IJERAT), 2(12), 11-16., Dostupno na: (https://www.researchgate.net/profile/Adebowale_Shadare/publication/311597028_DATA_VISUALIZATION/links/5851945608aef7d0309f20a7/DATA-VISUALIZATION.pdf)[Pristupljeno 02.02.2021]
10. Brightpointinc, Dostupno na: (http://www.brightpointinc.com/download/mobile-data-visualization-design)[Pristupljeno 02.02.2021]
11. Rockcontent (2020) , Dostupno na: (https://en.rockcontent.com/blog/data-visualization-for-mobile)
12. Material Design (2021), Dostupno na: (https://material.io) [Pristupljeno: 03.02.2021]
13. AnyChart (2021), Dostupno na: (https://www.anychart.com/technical-integrations/samples/android-charts/#license)[Pristupljeno: 03.02.2021]
14. SwiftPlot Dokumentacija (2021), Dostupno na: (https://github.com/KarthikRIyer/swiftplot#documentation) [Pristupljeno: 03.02.2021]
15. Hajirahimova, M., & Ismayilova, M. (2018). Big data visualization: Existing approaches and problems. Problems of Information Technology, 9, 72-83., Dostupno na: (https://jpit.az/uploads/article/en/2018_1/BIG_DATA_VISUALIZATION_EXISTING_APPROACHES_AND_PROBLEMS.pdf) [Pristupljeno: 03.02.2021]
16. Nielsen Norman Group (2021), Dostupno na: (https://www.nngroup.com/articles/tag-cloud-examples) [Pristupljeno: 03.02.2021]
17. R statistics (2010) Dostupno na: (https://www.r-statistics.com/2010/06/clustergram-visualization-and-diagnostics-for-cluster-analysis-r-code/) [Pristupljeno: 03.02.2021]

18. Excelcharts (2021), Dostupno na: (https://excelcharts.com/google-motion-chart-api-visualization-population-trends) [Pristupljeno: 03.02.2021]
19. Geckoboard (2021), Dostupno na: (https://www.geckoboard.com/dashboard-examples) [Pristupljeno: 03.02.2021]
20. Finder (2021), Dostupno na: (https://www.finder.com/uk/wally-app-review) [Pristupljeno: 04.02.2021]
21. Comparehero (2021), Dostupno na: (https://www.comparehero.my/technology/articles/money-lover-app-review-track-your-expenses), [Pristupljeno: 04.02.2021]
22. Money Lover (2021), Dostupno na: (https://moneylover.me), [Pristupljeno: 04.02.2021]
23. Nerdwallet (2021), Dostupno na: (https://www.nerdwallet.com/blog/finance/goodbudget-app-review) [Pristupljeno: 04.02.2021]
24. PC World (2021), Dostupno na: (https://www.pcworld.com/article/3287145/goodbudget-review.html) [Pristupljeno: 04.02.2021]
25. Cloud Firestore (2021), Dostupno na: (https://firebase.google.com/docs/firestore) [Pristupljeno: 05.02.2021]

Popis slika

Popis primjera

Sažetak

Ovaj diplomski rad se temelji na teorijskom i praktičnom dijelu razvoja aplikacije za vođenje osobnih financija. U teorijskom dijelu rada prolazi se kroz teoriju u okviru vizualizacije podataka. Započinje se sa definicijom vizualizacije zatim se razrađuju neke od najkorištenijih metoda vizualizacije podatka. Nakon toga prolazi se kroz neka područja vizualizacije te se govori o pojmovima vezanim uz moderni pristup vizualizaciji podataka. Na kraju teorijskog dijela razrađuju se neki od problema kod vizualizacije podataka na mobilnim uređajima te se izvode smjernice za izradu vizualizacije podataka. U praktičnom dijelu rada se nalaze opisani *UML use case* i sekvencijalni dijagrami aplikacije, te potom slijedi opis implementacije i funkcionalnosti aplikacije. Aplikacija *BudgetApp* nudi praktično rješenje za kreiranje financijskih planova za raspolaganje osobnim budžetom, te kreiranje izvještaja na temelju planiranih i ostvarenih prihoda i troškova.

Ključne riječi: vizualizacija podataka na mobilnim uređajima, mobilna aplikacija za vođenje osobnih financija, planiranje budžeta, firebase, android

Summery

This thesis is based on the theoretical and practical part of the development of applications for personal finance management. The theoretical part of the paper goes through the theory within the data visualization. It starts with the definition of visualization and then develops some of the most used methods of data visualization. After that, we go through some areas of visualization and talk about the concepts related to the modern approach to data visualization. At the end of the theoretical part, some of the problems with data visualization on mobile devices are elaborated and guidelines for creating data visualization are derived. The practical part of the paper contains the described UML use case and sequence diagrams of the application, followed by a description of the implementation and functionality of the application. The BudgetApp application offers a practical solution for creating financial plans for disposing of personal budgets and creating reports based on planned and realized revenues and expenses.

Keywords: data visualization on mobile devices, a mobile application for personal finance management, budget planning, firebase, android