Mobilna Android aplikacija za planiranje i praćenje obveznih i slobodnih aktivnosti korisnika

Tojčić, Selma

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

Sveučilišni studij računarstva

MOBILNA ANDROID APLIKACIJA ZA PLANIRANJE I PRAĆENJE OBVEZNIH I SLOBODNIH AKTIVNOSTI KORISNIKA

Završni rad

Selma Tojčić

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 12.09.2020.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Selma Tojčić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4147, 26.09.2019.
OIB studenta:	34358036153
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Mobilna Android aplikacija za planiranje i praćenje obveznih i slobodnih aktivnosti korisnika
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	12.09.2020.
Datum potvrde ocjene Odbora:	23.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 25.09.2020.

Ime i prezime studenta:	Selma Tojčić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4147, 26.09.2019.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Mobilna Android aplikacija za planiranje i praćenje obveznih i slobodnih aktivnosti korisnika**

izrađen pod vodstvom mentora Prof.dr.sc. Goran Martinović

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

Planiranje je postupak u kojem se definiraju sve aktivnosti koje su potrebne za dolazak do željenog cilja. Planiranje aktivnosti vrlo je važno, jer je izravno povezano s produktivnošću. Bitno je planirati sve obvezne aktivnosti kako bi se izvršilo sve što osoba mora obaviti u danu. Međutim, ljudi se često fokusiraju na obvezne aktivnosti pa zanemare one slobodne za koje bi se uvijek trebalo pronaći vremena. Korištenje ove aplikacije pojednostavilo bi korisnicima upravljanje svojim vremenom, kako obvezama tako i hobijima.

Tema ovog završnog rada je izrada mobilne Android aplikacije koja će korisniku olakšati planiranje i praćenje obveznih i slobodnih aktivnosti te dodavanje željenih aktivnosti u svoj svakodnevni raspored. U praktičnom dijelu rada stoga je ostvareno programsko rješenje mobilne aplikacije razvijene za Android platformu. Aplikacija, pomoću ankete o sklonostima korisnika, pomaže kreirati raspored aktivnosti. Na temelju parametara iz ankete, aktivnosti se unose u kalendar. Mobilna aplikacija također omogućuje registriranje i kasniju ponovnu prijavu korisnika.

U drugom poglavlju opisuje se problem planiranja i praćenja aktivnosti te su navedena i kratko opisana postojeća rješenja za zadani problem i opisan način na koji se ova aplikacija razlikuje od prethodno spomenutih postojećih rješenja. Treće poglavlje prikazuje model aplikacije za planiranje i praćenje aktivnosti. U tom su poglavlju opisani parametri i postupak planiranja. Također je prikazana svaka komponenta aplikacije i korisničko sučelje. Četvrto poglavlje opisuje korištene alate i tehnologije koji su potrebni za izradu mobilne aplikacije te prikazuje pojedine komponente aplikacije s pripadajućim programskim kodom. U petom poglavlju je opisan način korištenja aplikacije i ispitane su njene mogućnosti na temelju odabranih parametara.

1.1. Zadatak završnog rada

U završnom radu treba proučiti i opisati načine planiranja i praćenja obveznih i slobodnih aktivnosti, te mogućnosti korištenja mobilne aplikacije u tu svrhu. Koristeći profil sklonosti korisnika, te liste navedenih aktivnosti određenih opisom aktivnosti, kalendarskim vremenom i geolokacijom, potrebno je definirati model provedbe aktivnosti, postupak planiranja, predviđanja i stvaranja preporuka aktivnosti korisniku. Nadalje, treba definirati arhitekturu aplikacije i opisati potrebne programske tehnologije i razvojnu okolinu. U praktičnom dijelu rada, treba razviti mobilnu aplikaciju s bazom

podataka i ugraditi navedeni model provedbe i planiranja aktivnosti, te stvaranja preporuka. Mobilna aplikacija treba omogućiti unos i pohranu navedenih podataka, prikaz plana i preporuka aktivnosti, te njihovo praćenje. Mobilnu aplikaciju potrebno je ispitati i analizirati za odgovarajuće profile korisnika, aktivnosti i ulazne parametre.

2. PLANIRANJE SLOBODNIH AKTIVNOSTI

U ovom poglavlju opisano je planiranje aktivnosti te način na koji aplikacija predlaže određene slobodne aktivnosti. Na kraju poglavlja objašnjene su sličnosti i razlike aplikacija poznatih tehnoloških tvrtki i ove aplikacije.

2.1. Planiranje i predlaganje aktivnosti

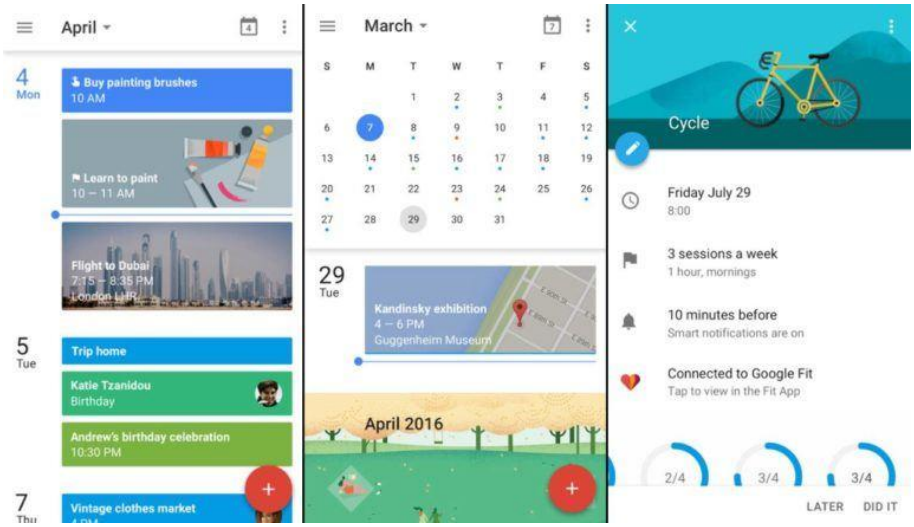
Planiranje je proces u kojem definiramo sve aktivnosti koje su potrebne da dođemo do željenog cilja. Prema [1], planiranje je misaono prethodno rješavanje budućeg djelovanja odmjeravanjem različitih alternativa djelovanja i odlukama o najpovoljnijem putu. Taj proces je vrlo potreban za postizanje učinkovitosti i produktivnosti. Prema [2], upravljanje vremenom je proces mudrog organiziranja i planiranja kako podijeliti vrijeme između naših aktivnosti i zadataka. Na temelju unesenih parametara u početnoj anketi koju aplikacija sadrži, u raspored korisnika se predlaže dodavanje slobodnih aktivnosti u raspored. Dopuštanjem očitanja lokacije korisnika omogućena je i opcija dodavanja događaja u blizini trenutne lokacije korisnika.

2.2. Prikaz postojećih rješenja za planiranje slobodnih aktivnosti

Na tržištu postoji mnogo aplikacija za planiranje aktivnosti. Većina takvih aplikacija je orijentirana na određenu vrstu aktivnosti, na primjer tjelesne aktivnosti. U nastavku će biti opisana tri različita rješenja mobilne aplikacije za neku vrstu praćenja aktivnosti.

2.2.1. Google Calendar

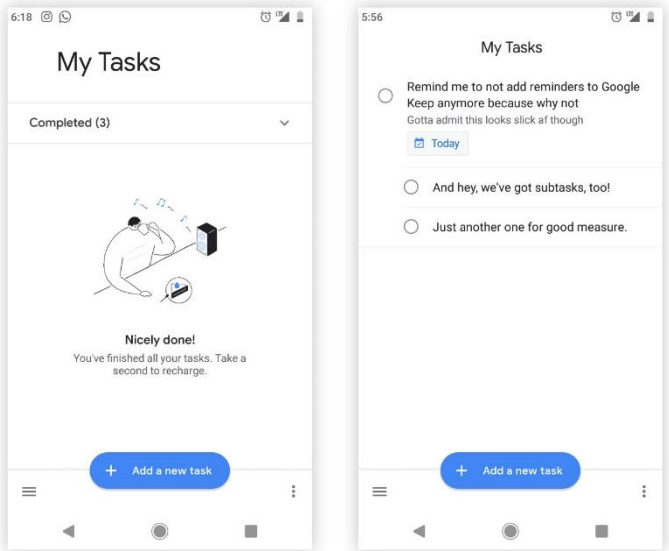
Google, jedna od najpoznatijih i najutjecajnijih svjetskih tehnoloških tvrtki, razvio je aplikaciju [3], za planiranje u kojoj postoje unos događaja i aktivnosti u kalendar, podsjetnik za obveze, unos različitih dnevnih i dugoročnih ciljeva i slično. Na mobilni uređaj šalje podsjetnike na različite događaje kako korisnik ne bi propustio željenu aktivnost. Ova aplikacija se povezuje s ostalim Google-ovim uređajima i aplikacijama što može biti vrlo korisno jer stavlja sve potrebne informacije na jedno mjesto. Na slici 2.1 je vidljivo korisničko sučelje ove aplikacije.



Slika 2.1. Sučelje aplikacije Google Calendar

2.2.2. Google Tasks

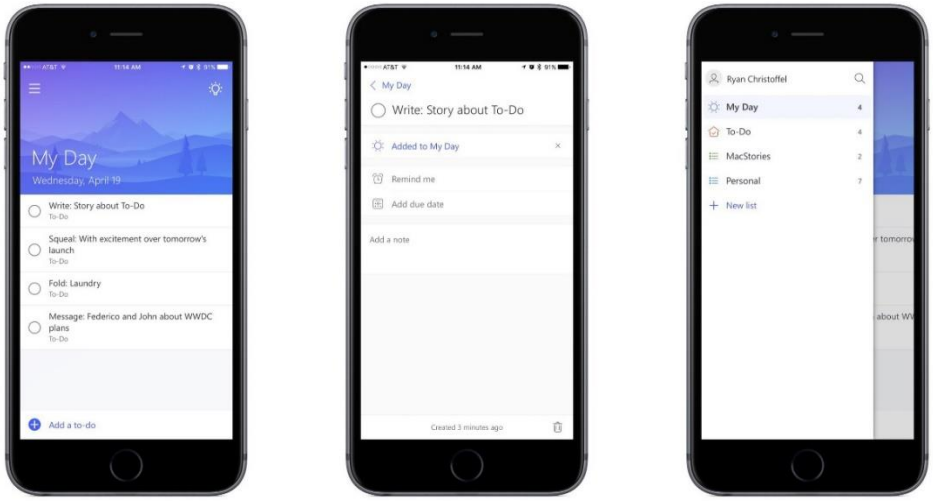
Google Tasks [4], je još jedna aplikacija za planiranje razvijena od strane tvrtke Google. Koristeći ovu aplikaciju korisnik ima mogućnost izrade popisa zadataka s obvezama, određivanje zadataka koji su prioritet, postavljanje rokova za svaki zadatak, organiziranje zadataka prema datumu te pregled napretka u zadanim aktivnostima. Kao i prethodno spomenuta aplikacija Google Calendar, Google Tasks je moguće povezati s ostalim uređajima i aplikacijama tvrtke Google. Na slici 2.2 je prikazano sučelje ove aplikacije.



Slika 2.2. Sučelje aplikacije Google Tasks

2.2.3. Microsoft To Do

Microsoft To Do je, prema [5], mobilna aplikacija tvrtke Microsoft za upravljanje zadacima i temelji se na cloud tehnologiji. Ova aplikacija omogućuje unos dnevnog plana, predlaganje zadataka, postavljanje podsjetnika, prilaganje datoteka u zadatke i slično. Izgled aplikacije je prikazan na slici 2.3.



Slika 2.3. Sučelje aplikacije Microsoft To Do

2.3. Idejno rješenje vlastite mobilne aplikacije za planiranje aktivnosti

Razvoj ove aplikacije su motivirale prethodno spomenute aplikacije te mnoge druge aplikacije koje postoje s ovom temom. Korisnik ima mogućnost unosa željenih zadataka, ciljeva, aktivnosti koje je obavezan napraviti i aktivnosti koje želi uvrstiti u svoj raspored. Mogućnost aplikacije koja ju razlikuje od prethodno spomenutih primjera je unos događaja s obzirom na lokaciju korisnika. Na temelju lokacije korisnika bit će predloženi događaji koji se nalaze u njegovoj blizini te će korisnik imati mogućnost prihvatiti ili odbiti unos tog događaja u svoj raspored. U poglavlju 3 je detaljnije opisan model aplikacije i sve mogućnosti koje ima.

3. MODEL APLIKACIJE ZA PLANIRANJE AKTIVNOSTI

3.1. Korišteni parametri planiranja slobodnih aktivnosti

Parametri koji se koriste za dodavanje preporučenih slobodnih aktivnosti i događaja su interes, raspoloživost i blizina. Putem ankete koju korisnik ima mogućnost ispuniti pri prvoj prijavi u aplikaciju, saznaje se koje vrste aktivnosti i kakvi događaji zanimaju korisnika, odnosno njegov interes za njih. Pri svakoj prijavi korisnika, pregledava se njegov raspored i predlažu aktivnosti u vrijeme kada korisnik nema već zauzet raspored te ima dovoljno vremena za preporučenu aktivnost. Ukoliko korisnik omogući aplikaciji da očitava njegovu lokaciju, biti će mu predložen događaj u njegovoj blizini pod uvjetom da za taj događaj nisu rasprodane ulaznice i da postoji još dovoljno mjesta i vremena za prijaviti se na taj događaj.

3.2. Korišteni postupak planiranja slobodnih aktivnosti

Problemi planiranja se najčešće rješavaju multikriterijskim pristupom. Prema [6], takvi problemi se rješavaju pomoću ASAP i ALAP metode. ASAP metoda, odnosno „As soon as possible“ metoda i ALAP, „As Late As Possible“ metoda koriste se kod planiranja u slučaju da se aktivnosti dijele na kritične i nekritične. Prema [7], za planiranje je potrebno pratiti proces na slijedeći način: identifikacija ciljeva, identifikacija opcija za postizanje ciljeva, identifikacija kriterija koji su potrebni za uspoređivanje opcija, analiza opcija, odlučivanje i povratna informacija. Ciljevi se u ovoj aplikaciji identificiraju pomoću ankete koju korisnik rješava. Opcije za postizanje ciljeva predstavljaju raspored korisnika u koji se unose određeni zadaci jer će moguće opcije biti samo u vrijeme kada je korisnik slobodan. Kriteriji za predlaganje aktivnosti dobivaju se iz ankete, točnije iz odgovora na pitanja postavljena korisniku. Sljedeći korak je analiza opcija odnosno traženje slobodnog vremena u rasporedu korisnika. Odlučivanje je korak u kojem se s obzirom na odgovore i količinu slobodnog vremena unosi određena aktivnost u raspored.

3.3. Funkcionalnosti mobilne aplikacije

U ovoj aplikaciji za planiranje od korisnika se prvo zahtjeva registriranje. Za registriranje su potrebni adresa e-pošte i lozinka. Kada se korisnik registrira, dobiva opciju da mu aplikacija predloži događaje i aktivnosti koje bi ga mogle interesirati s obzirom na njihove odgovore na postavljena pitanja u anketi. Ako korisnik ne želi preporuke, otvara se posljednji zaslon ankete i zatim se otvara glavni zaslon aplikacije. U slučaju da korisnik želi preporuke, prikazuju se zasloni ankete. Kada korisnik odgovori na sva postavljena pitanja, otvara se posljednji zaslon ankete i izabrani odgovori se spremaju u bazu podataka Firebase. Baza podataka obrađuje te odgovore i na temelju toga prikazuje preporuke korisniku u daljnjem korištenju aplikacije.

3.3.1. Registriranje korisnika

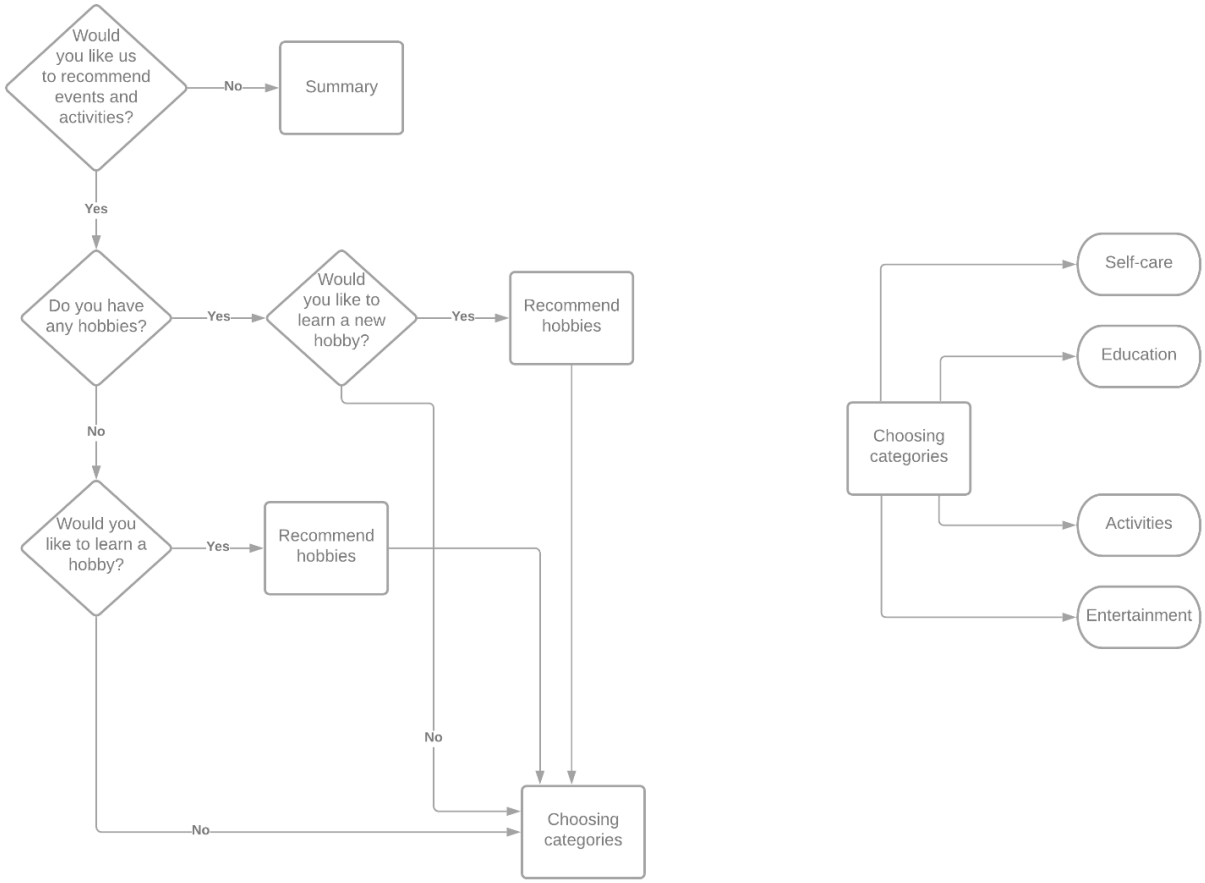
Pri ulasku u aplikaciju, korisniku je vidljiv zaslon za prijavu na kojem se nalazi gumb koji vodi na zaslon za registriranje. Zaslon za registriranje sastoji se od polja za unos imena korisnika, adrese e-pošte te dva polja za unos lozinke kako bi se izbjegla slučajna pogreška korisnika pri unosu željene lozinke. Ispod prethodno spomenutih polja nalazi se gumb za registriranje. Registriranje se obavlja putem Firebase Auth-a koji omogućava kreiranje jednog računa na jednoj adresi e-pošte i obavlja sve potrebne radnje za čuvanje korisničkih podataka uključujući davanje jedinstvenog ID-a koji služi za razlikovanje korisnika. Kada korisnik upiše potrebne podatke i oni su ispravni, otvara se zaslon za prijavu na kojemu su automatski upisani podaci koje je korisnik unio na zaslonu za registriranje.

3.3.2. Prijava korisnika

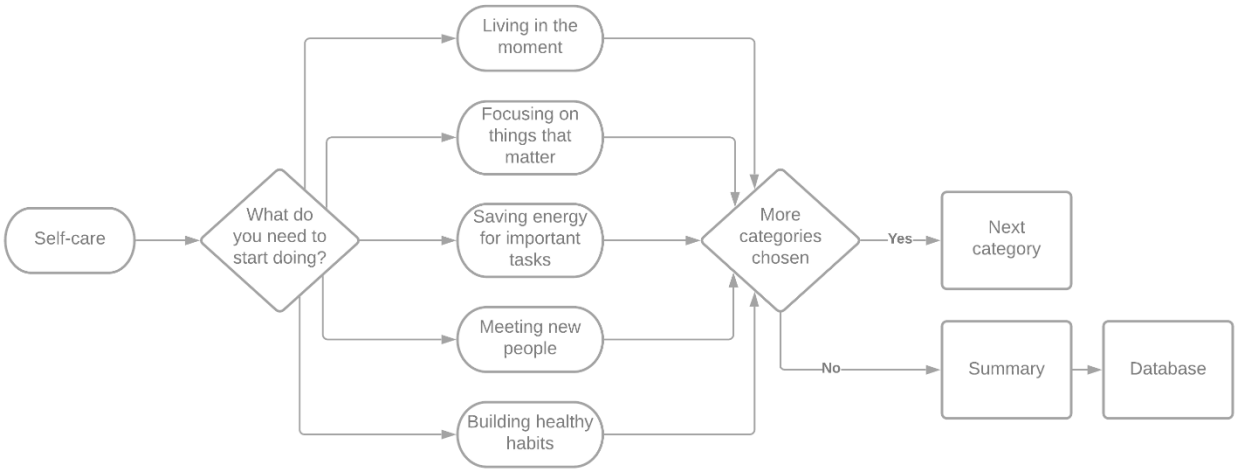
Na zaslonu za prijavu nalaze se dva polja u koja korisnik upisuje svoje podatke u slučaju da već ima račun. Prvo polje služi za unos adrese e-pošte, a drugo polje za unos lozinke. Ispod polja za unos lozinke nalazi se gumb koji se koristi ako je korisnik zaboravio lozinku. Tada Firebase Auth šalje poruku na upisanu adresu e-pošte u kojoj se nalazi hiperveza za obnovu korisničke lozinke. Ispod prethodno spomenutog gumba nalazi se gumb za prijavu koji otvara anketu ili početni zaslon aplikacije. Ukoliko se korisnik prijavio prvi put, otvara se početni zaslon ankete na kojem se korisniku nudi opcija predlaganja preporuka. Ako nije korisnikova prva prijava, otvara se početni zaslon aplikacije sa zadacima unesenim od strane korisnika i predloženim od strane aplikacije.

3.3.3. Anketa o navikama korisnika

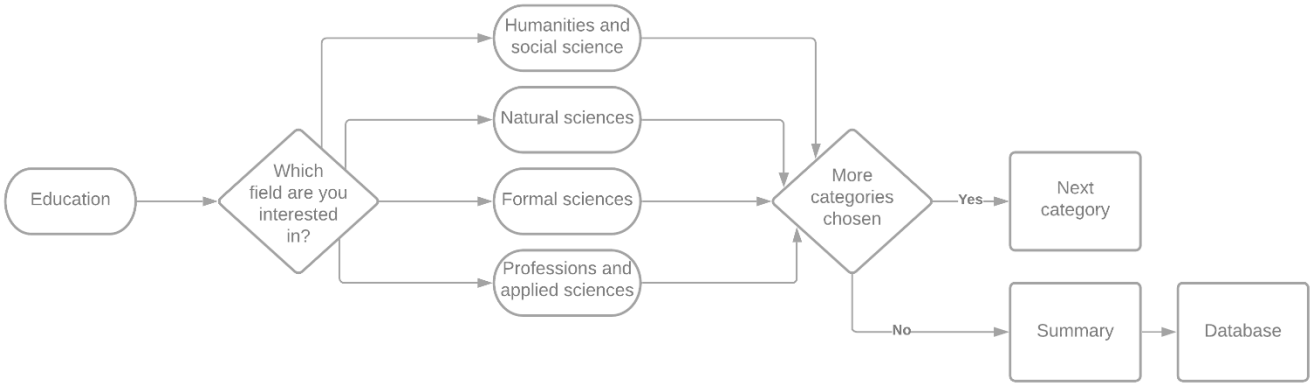
Nakon prve prijave korisnika otvara se prvi zaslon ankete na kojemu korisnik ima izbor o dodavanju preporuka na njegov račun. Ako korisnik ne želi preporuke, otvara se posljednji zaslon ankete i nakon toga glavni zaslon aplikacije, to jest zaslon sa zadacima korisnika i događajima u blizini. Pod uvjetom da korisnik želi preporuke, prikazuje se zaslon na kojemu su postavljena pitanja o hobijima korisnika. Nakon toga se otvara zaslon s kategorijama gdje korisnik bira kategorije koje ga zanimaju i s obzirom na izabrane dobiva pitanja. Kada korisnik odgovori na sva postavljena pitanja, otvara se posljednji zaslon ankete i izabrani odgovori se spremaju u bazu podataka Firebase. Baza podataka obrađuje te odgovore i na temelju njih prikazuje preporuke korisniku u daljnjem korištenju aplikacije. Na slikama 3.1, 3.2, 3.3, 3.4 i 3.5 vidljivi su modeli tijeka rješavanja ankete. Modeli su napisani na engleskom jeziku jer je i sama aplikacija tako napisana.



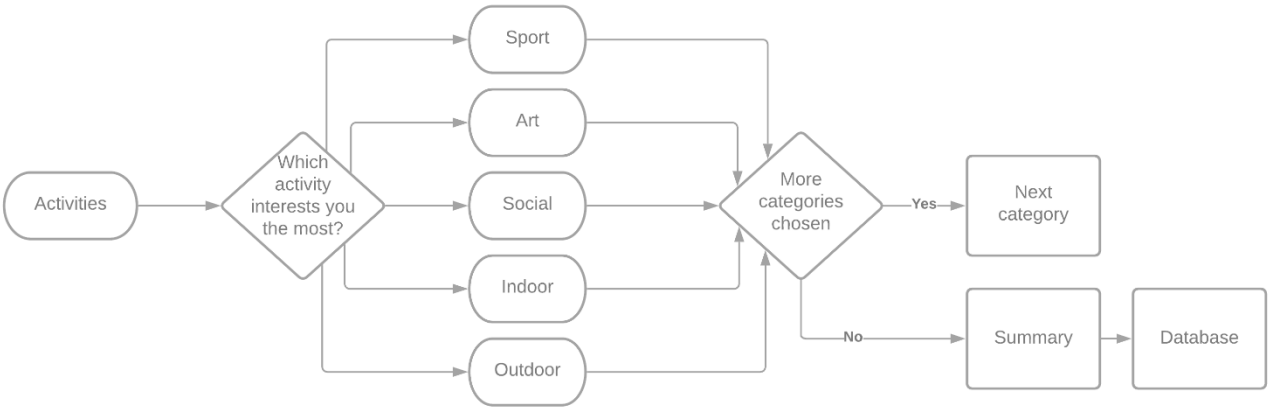
Slika 3.1. Tijek aktivnosti pitanja o hobijima i biranje kategorija pitanja



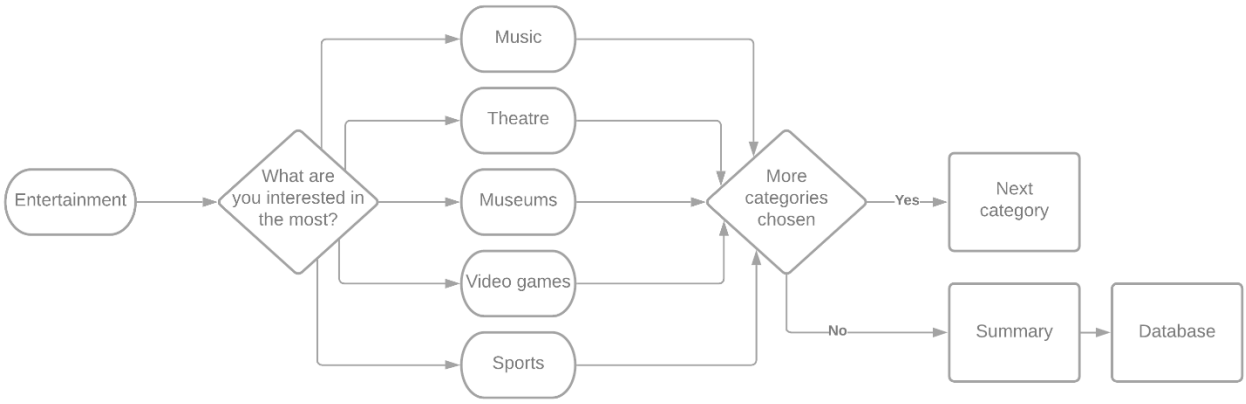
Slika 3.2. Tijek aktivnosti pitanja iz kategorije „Self-care“



Slika 3.3. Tijek aktivnosti pitanja iz kategorije „Education“



Slika 3.4. Tijek aktivnosti pitanja iz kategorije „Activities“



Slika 3.5. Tijek aktivnosti pitanja iz kategorije „Entertainment“

3.3.4. Dodjeljivanje preporuka s obzirom na odgovore u anketi

S obzirom na odgovore korisnika u prethodno opisanoj anketi, aplikacija jednom dnevno korisniku predlaže određene aktivnosti i događaje. Za svaki odgovor postoji više preporuka koje se nasumično predlažu korisniku. U tablicama 3.1, 3.2, 3.3 i 3.4 vidljivo je s kojim odgovorima na pitanja u anketi su povezane koje preporuke. U prvom stupcu svake tablice napisani su nazivi odgovora korisnika iz ankete na engleskom jeziku, a u drugom stupcu su napisani na hrvatskom jeziku. Treći stupac je popunjen preporukama za korisnika koje mu se predlažu unutar aplikacije, a te preporuke su napisane na engleskom jeziku. U posljednjem stupcu tablice preporuke su prevedene na hrvatski jezik.

Tablica 3.1. Odgovori korisnika i preporuke iz kategorije „Self-care“.

Nazivi odgovora korisnika na engleskom	**Nazivi odgovora korisnika na hrvatskom**	**Nazivi preporuka na engleskom**	**Nazivi preporuka na hrvatskom**
Living in the moment	Življenje u trenutku	Clean clutter from your room	Očisti nered u svojoj sobi
		Visit a friend	Posjeti prijatelja
Focusing on things that matter	Fokusiranje na bitne stvari	Take time for yourself	Pronađi vrijeme za sebe
		Take a bath	Okupaj se
Saving energy for important tasks	Čuvanje energije za bitne zadatke	Go outside	Provedi vrijeme na svježem zraku
		Get a massage	Idi na masažu

Meeting new people	Upoznavanje novih ljudi	Find people with similar interests	Upoznaj ljude sa sličnim interesima
		Join a class or a club	Pridruži se tečaju ili klubu
Building healthy habits	Stvaranje zdravih navika	Drink water	Pij vodu
		Cook a meal	Napravi obrok

Tablica 3.2. Odgovori korisnika i preporuke iz kategorije „Education“.

Nazivi odgovora korisnika na engleskom	**Nazivi odgovora korisnika na hrvatskom**	**Nazivi preporuka na engleskom**	**Nazivi preporuka na hrvatskom**
Humanities and social science	Humanističke I društvene znanosti	Watch a documentary about history	Pogledaj dokumentarni film o povijesti
		Watch a documentary about psychology	Pogledaj dokumentarni film o psihologiji
		Read an article about economics	Pročitaj članak o ekonomiji
Natural sciences	Prirodne znanosti	Watch a documentary about physics	Pogledaj dokumentarni film o fizici
		Watch a documentary about astronomy	Pogledaj dokumentarni film o astronomiji
		Read an article about biology	Pročitaj članak o biologiji
Formal sciences	Formalne znanosti	Watch a documentary about statistics	Pogledaj dokumentarni film o statistici
		Read an article about computer science	Pročitaj članak o informatici
		Read an article about logic	Pročitaj članak o logici
Professions and applied sciences	Zanimanja I primijenjene znanosti	Watch a documentary about medicine	Pogledaj dokumentarni film o medicini
		Read an article about architecture	Pročitaj članak o arhitekturi
		Read an article about business	Pročitaj članak o poduzetništvu

Tablica 3.3. Odgovori korisnika i preporuke iz kategorije „Activities“.

Nazivi odgovora korisnika na engleskom	**Nazivi odgovora korisnika na hrvatskom**	**Nazivi preporuka na engleskom**	**Nazivi preporuka na hrvatskom**
Sport	Sport	Go jogging	Idi trčati
		Take a bike ride	Provozaj se biciklom
		Go to the gym	Idi u teretanu
Art	Umjetnost	Draw something	Nacrtaj nešto
		Make a DIY project	Napravi sam svoj projekt
		Decor your room	Uredi svoju sobu
Social	Društvene aktivnosti	Meet with a friend	Sastani se s prijateljem
		Volunteer	Volontiraj
		Organize a game night	Organiziraj večer igranja igara
Indoor	Aktivnosti u zatvorenom	Watch a movie	Pogledaj film
		Play a game	Igraj igru
		Read a book	Pročitaj knjigu
Outdoor	Aktivnosti na otvorenom	Have a picnic	Idi na piknik
		Plant some flowers	Posadi cvijeće
		Take photographs	Fotografiraj

Tablica 3.4. Odgovori korisnika i preporuke iz kategorije „Entertainment“.

Nazivi odgovora korisnika na engleskom	**Nazivi odgovora korisnika na hrvatskom**	**Nazivi preporuka na engleskom**	**Nazivi preporuka na hrvatskom**
Music	Glazba	Go to a concert nearby	Idi na concert u blizini
		Learn to play a new instrument	Nauči svirati novi instrument
Theatre	Kazalište	Go to a play	Idi na predstavu
		Join a drama club	Pridruži se dramskoj grupi
Museums	Muzeji	Visit an art exhibition	Posjeti umjetničku izložbu
		Go to a virtual museum tour	Virtualno posjeti muzej
Video games	Video igre	Play your favorite game	Igraj svoju najdražu igru
		Go to a gaming convention nearby	Idi na konvenciju igara u blizini

Sports	Sport	Watch an old match you enjoyed	Pogledaj staru utakmicu u kojoj si uživao
		Watch a game of a sport you never watched	Pogledaj utakmicu sporta koji nisi prije gledao

3.3.5. Popis zadataka korisnika i događaji u blizini

Na prvom zaslonu nakon prijave nalaze se zadaci korisnika koji su prikazani s obzirom na opciju koju korisnik izabere. Korisnik može vidjeti zadatke koji se odnose na taj dan, taj tjedan ili taj mjesec. Ispod tih zadataka nalaze se predloženi događaji u blizini korisnika ako je korisnik omogućio očitavanje njegove lokacije. U donjem desnom kutu nalazi se gumb koji otvara zaslon za dodavanje zadatka koji će biti pobliže opisan u poglavlju 3.4.9.

3.3.6. Kalendar

Sa zaslona na kojem se nalazi popis zadataka korisnika i događaji u blizini pritiskom na srednji gumb u navigaciji otvara se zaslon s kalendarom. Zaslon s kalendarom sastoji se od prikaza trenutnog mjeseca s ikonicama na danima u koje je upisan neki događaj ili aktivnost. Pritiskom na takav dan, ispod prikaza kalendara otvara se popis svih događaja i aktivnosti u tom danu koje si je zadao korisnik i koje mu je preporučila aplikacija. U donjem desnom kutu nalazi se gumb koji otvara zaslon za dodavanje zadatka koji će biti pobliže opisan u poglavlju 3.4.9.

3.3.7. Korisnički profil

Pritiskom na posljednji gumb u navigaciji otvara se zaslon korisničkog profila. Na zaslonu korisničkog profila vidljiva je korisnička slika koju može sam izabrati iz galerije i njegovo ime. Ispod glavnih korisničkih podataka nalazi se lista zadataka koje korisnik želi odraditi svaki dan. Ispod te liste nalazi se lista svih zadataka korisnika poredana od vremenski najbližih do najdaljih. Na zaslonu se također nalazi gumb koji vodi na zaslon za korisničke postavke. U donjem desnom kutu nalazi se gumb koji otvara zaslon za dodavanje zadatka koji će biti pobliže opisan u poglavlju 3.4.9.

3.3.8. Dodavanje zadatka

Na zaslonu za stvaranje zadatka nalaze se dva polja, prvo za unos naslova i drugo za unos opisa zadatka. Ispod polja nalaze se opcije za namještanje vremena zadatka i koliko će se on često pojavljivati u rasporedu. Na zaslonu postoje i četiri gumba. Prvi gumb služi za izlaz iz izrade zadatka.

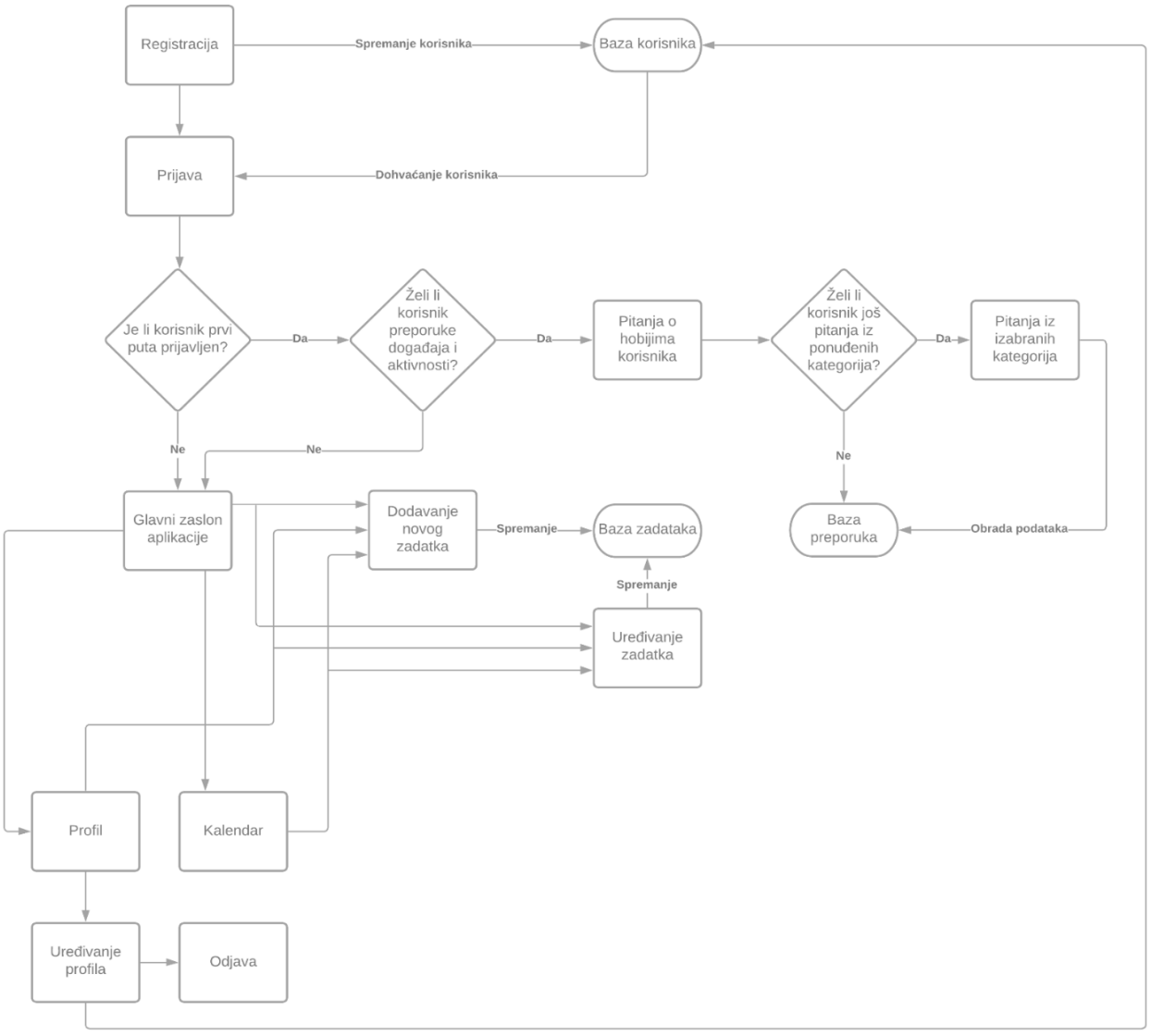
Pritiskom na drugi gumb korisnik ima mogućnost biranja ikone zadatka od ponuđenih. Ispod tog gumba nalazi se gumb za izbor boje kojom će zadatak biti prikazan na kalendaru. Na samom dnu zaslona je posljednji gumb koji služi za spremanje podataka o zadatku u bazu podataka i njegov prikaz u samoj aplikaciji.

3.3.9. Uređivanje korisničkog profila i odjava korisnika

Pri ulasku na zaslon za uređivanje korisničkog profila vidljiva su četiri gumba. Prvi gumb služi za promjenu imena korisnika. Gumb ispod njega koristi se za promjenu lozinke. Ispod gumba za promjenu lozinke nalazi se gumb za odjavu korisnika i posljednji gumb koji se koristi za brisanje korisničkog računa. Svaki od tih gumbova mijenja korisničke podatke u bazi podataka Firebase i promjene su vidljive unutar same aplikacije.

Odjavom korisnika svi njegovi podaci ostaju sačuvani i zaštićeni u bazi podataka Firebase. Ponovnim pokretanjem aplikacije, svi korisnički podaci jednako su prikazani u aplikaciji. Nakon odjave aplikacija otvara zaslon prijave.

Na slici 3.5 vidljiv je tijek aktivnosti u aplikaciji za planiranje i praćenje obveznih i slobodnih aktivnosti korisnika opisan u ovom poglavlju.



Slika 3.5. Tijek aktivnosti aplikacije za planiranje i praćenje obveznih i slobodnih aktivnosti korisnika

3.4. Dizajn mobilne aplikacije

Bitan korak izrade aplikacije je izrada samog dizajna aplikacije. Pomoću alata Figma [8] za kreiranje dizajna i prototipa web i mobilnih aplikacija napravljen je dizajn ove aplikacije. Iz Figme se izvezu slike i oblici koji su potrebni za dizajn aplikacije i u samom razvojnom okruženju Android Studio pomoću jezika za označavanje podataka XML te se slike i oblici namještaju na željena mjesta na zaslonu aplikacije.

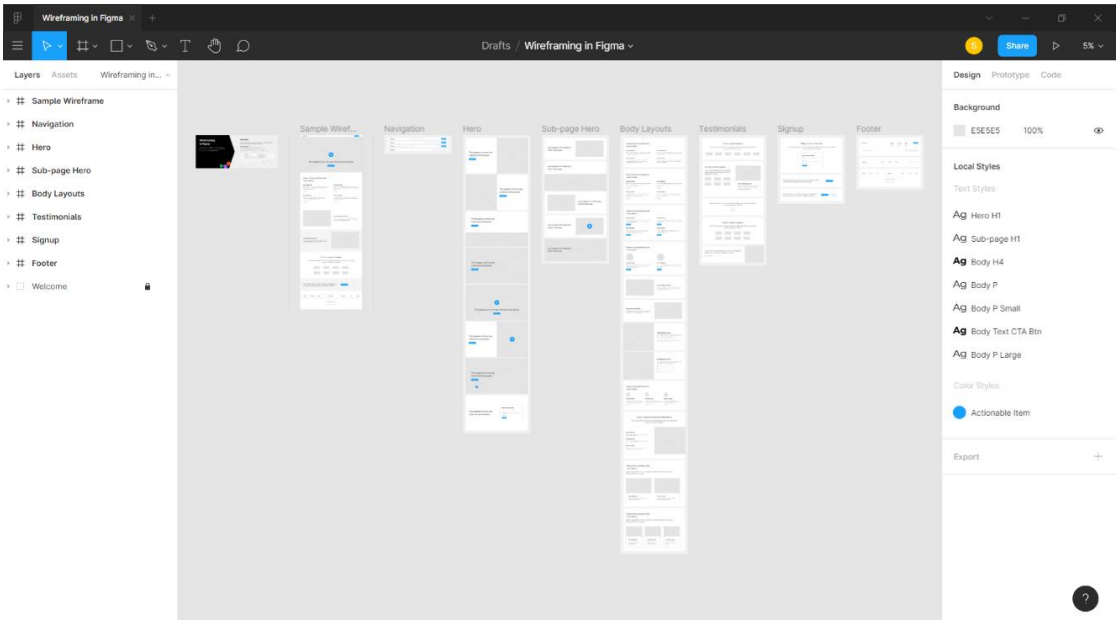
4. PROGRAMSKO RJEŠENJE APLIKACIJE ZA PLANIRANJE AKTIVNOSTI

4.1. Korišteni alati i tehnologije

Aplikacija za planiranje i praćenje obveznih i slobodnih aktivnosti korisnika najprije je dizajnirana u alatu za dizajn sučelja Figma. Nakon dizajniranja aplikacija je izrađena u integriranom razvojnom okruženju Android Studio. Programski jezik u kojem je napisana aplikacija je Kotlin, a za korisničko sučelje se koristi jezik za označavanje podataka XML. Za sustav autentifikacije i bazu podataka korišten je Firebase. Aplikacija je napravljena po uzoru na MVVM arhitekturu koja olakšava odvajanje korisničkog sučelja od pozadinske logike aplikacije.

4.1.1. Figma

Figma [8], je alat za kreiranje dizajna i prototipa web i mobilnih aplikacija. Korisnicima Figme je, pomoću jednostavnih alata za dizajniranje, omogućen velik spektar mogućnosti. Prvi korak svake aplikacije je izrada dizajna, a ovaj alat olakšava taj proces. Figma ima mogućnost izvoza svakog dijela dizajna, od oblika do boja i fontova te tako pojednostavljuje dodavanje potrebnih elemenata u Android Studio. Na slici 4.1 je vidljivo korisničko sučelje ovog alata.



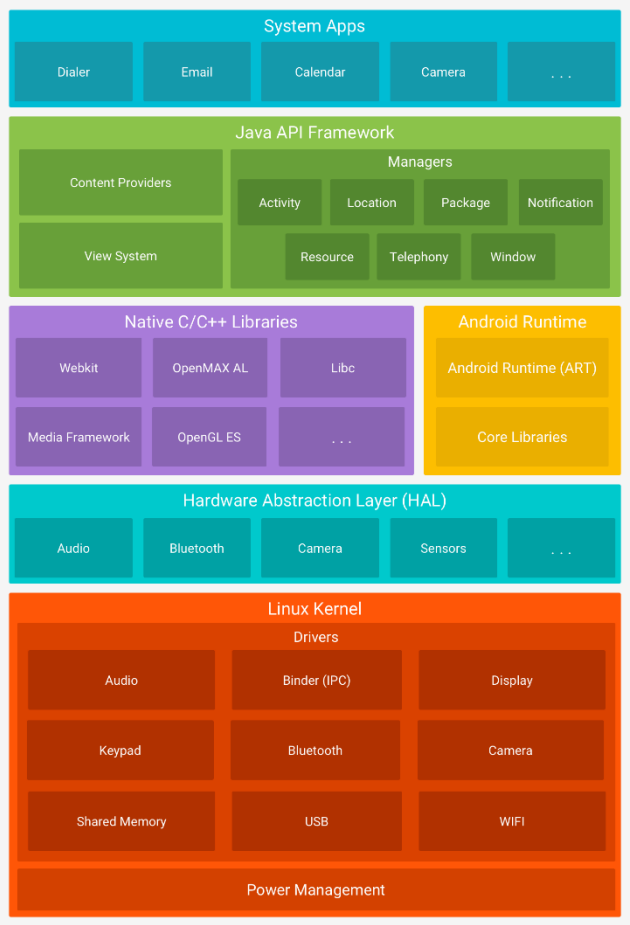
Slika 4.1. Sučelje alata Figma

4.1.2. Operacijski sustav Android

Prema [9], Android je operacijski sustav otvorenog koda stvoren za širok spektar uređaja i mogućnosti baziran na Linux-u. Na slici 4.2 preuzetoj s [10] prikazane su glavne komponente Android platforme i njihov opis slijedi u nastavku. Linux Kernel je osnova Android platforme te služi kao osnova za rad s procesima, memorijom, pogonskim programima i upravlja potrošnjom energije. Također omogućuje proizvođačima uređaja da razvijaju upravljačke programe sklopovlja. Sljedeći sloj je Hardware Abstraction Layer (HAL) koji pruža standardna sučelja koja prikazuju mogućnosti sklopovlja uređaja višem nivou Java API okvira. Ovaj sloj se sastoji od više modula biblioteka od kojih svaki implementira sučelje za određenu vrstu sklopovske komponente, poput kamere.

Sloj iznad HAL-a je Android Runtime koji je napisan tako da pokreće više virtualnih računala na uređajima s malo memorije izvršavanjem DEX datoteka, formata koji je posebno dizajniran za Android. Android Runtime omogućuje kompilaciju, optimiziranu kolekciju smeća, pretvorbu DEX datoteka u strojni kod i mnoge druge mogućnosti. Sljedeći sloj su Native C/C++ Libraries. Komponente Android sustava kao što su prethodno spomenute Android Runtime i HAL, napravljene su od izvornog koda za koji su potrebne izvorne datoteke napisane u programskom jeziku C i C++. Android platforma omogućuje otkrivanje funkcionalnosti nekih od tih izvornih biblioteka aplikacijama.

Sloj iznad Native C/C++ Libraries i Android Runtime je Java API Framework koji obuhvaća cijeli set značajki Android operacijskog sustava. Ovaj sloj se sastoji od dijelova kao što su View System, Resource Manager, Notification Manager, Activity Manager i Content Provider. Ti dijelovi omogućuju pojednostavljenu izradu aplikacije i dodavanje svih potrebnih dijelova. Posljednji sloj je System Apps koji obuhvaća skup osnovnih aplikacija za e-poštu, SMS poruke, kalendare, kontakte i slično. Ove aplikacije rade na isti način kao i aplikacije koje korisnik odluči sam instalirati. Android operacijski sustav je najrašireniji operacijski sustav za mobilne uređaje, no često je korišten i u drugim tehnologijama kao što su pametni televizori, pametni satovi, automobili i mnogi drugi.

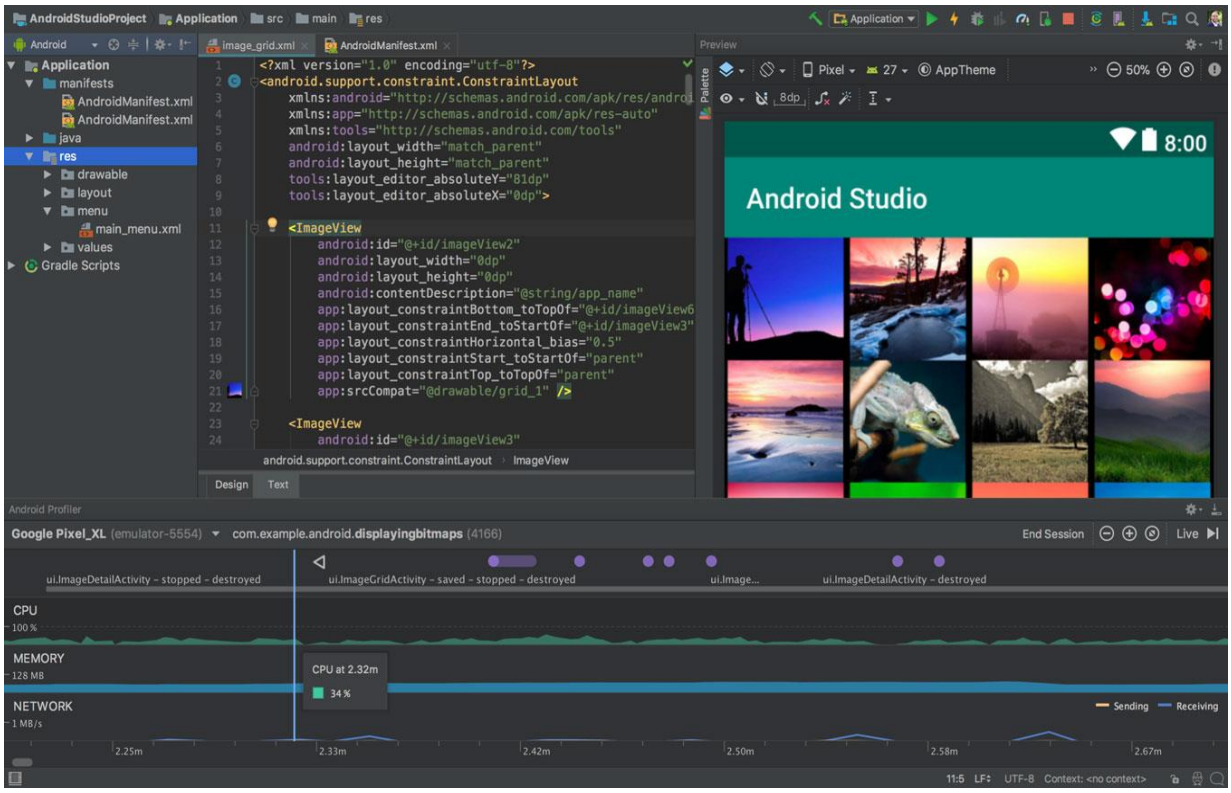


Slika 4.2. Građa Android platforme [10]

4.1.3. Android Studio

Android Studio po definiciji na službenoj stranici [11], je službeno integrirano razvojno okružje (eng. *Integrated Development Environment*) za razvoj aplikacija u Android-u, temeljeno na IntelliJ IDEA integriranom razvojnom okruženju. Osim IntelliJ-evog uređivača koda i alata za programere, Android Studio nudi još značajki za poboljšanje izrade Android aplikacija. Neke od tih značajki su brz emulator, unificirano okruženje za izradu aplikacija za sve Android uređaje, primjena promjena na aplikaciji bez ponovnog pokretanja aplikacije, predlošci kodova i integrirani GitHub sustav, opsežni alati i okviri za testiranje, podrška za C++, Android SDK (eng. *Android software development).* Programski jezici za pisanje koda koje Android Studio podržava su Java i noviji Kotlin koji je baziran na Javi. Kotlin je riješio mnoge probleme koje programski jezik Java ima te više o ovom programskom

jeziku u slijedećem poglavlju. Slika 4.3 prikazuje integrirano razvojno okruženje Android Studio, točnije njegovu najnoviju inačicu 4.0 i neke od prethodno spomenutih značajki koje ima.



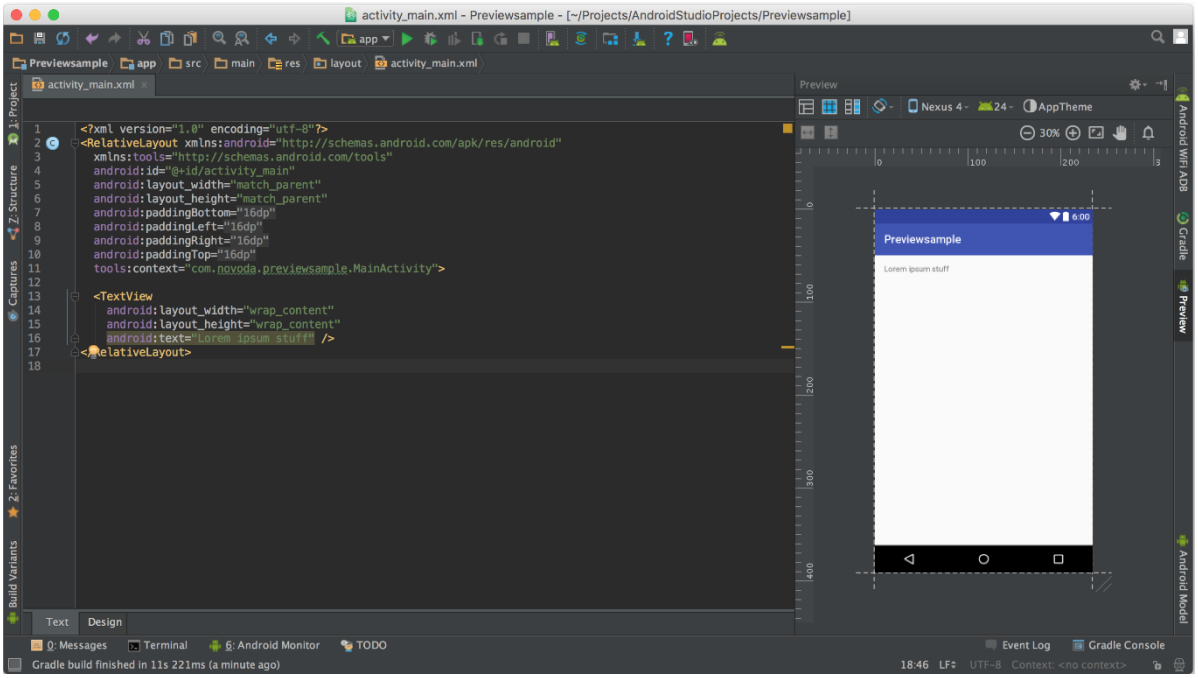
Slika 4.3. Android Studio

4.1.4. Programski jezik Kotlin

Kotlin [12], je statički programski jezik opće namjene za razvoj multiplatformskih aplikacija. Relativno je nov programski jezik, prihvaćen kao vodeći programski jezik za razvijanje Android aplikacija od strane Google-a 2019. godine. Do 2019. godine vodeći je jezik bio Java, s kojom je Kotlin interoperabilan. Kotlin je postao prihvaćen jer je koncizan, siguran, interoperabilan i moguć za korištenjem u raznim razvojnim okruženjima, kao što su IntelliJ IDEA, Android Studio, Eclipse i drugi.

4.1.5. XML

XML (eng. *Extensible Markup Language)* je prema definiciji [13], jezik za označavanje podataka koji je vrlo jednostavno čitljiv i ljudima i računalnim programima. Način pisanja XML-a je poprilično jednostavan, odgovarajući opisni sadržaj se treba uokviriti oznakama koje ga opisuju i imaju lako shvatljivo značenje. Format ovog jezika za označavanje podataka vrlo je sličan formatu oznaka u HTML-u. XML se u razvojnom okruženju Android Studio koristi za izradu izgleda korisničkog sučelja. Na slici 4.4 je najjednostavniji primjer XML koda u okruženju Android Studio.

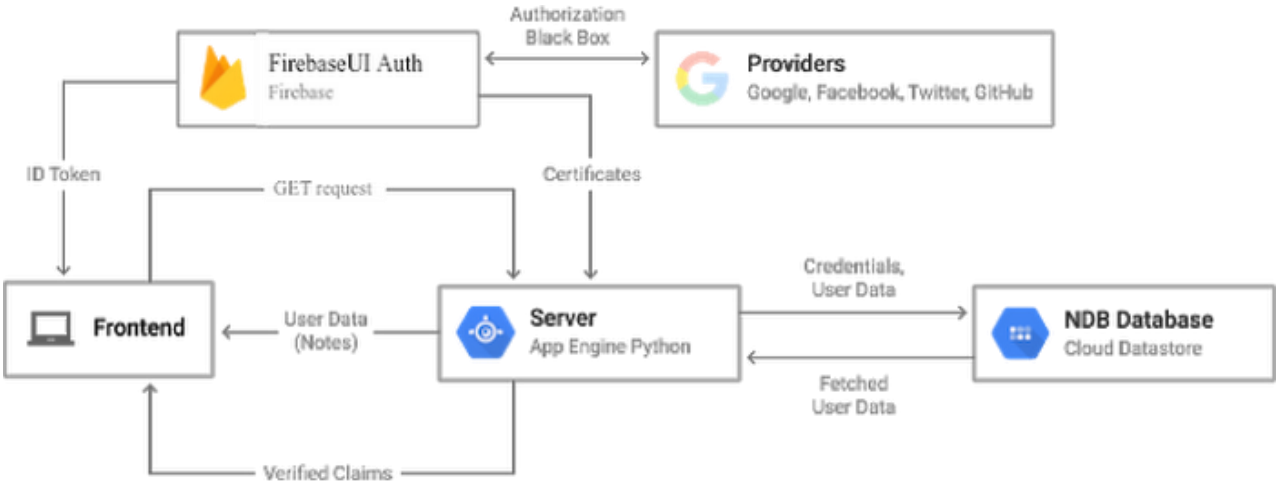


Slika 4.4. Primjer XML koda

4.1.6. Firebase

Firebase [14] je platforma za razvoj mobilnih i web aplikacija koju je razvio Firebase, Inc. 2014. godine, a 2014. godine kupio Google. Ova platforma se koristi za Google analitiku, razvoj aplikacija, testiranje stabilnosti, poboljšanje kvalitete aplikacija i slično. Neke od mogućnosti Firebase-a za razvoj aplikacija su Cloud Firestore, Firebase ML, Cloud Functions, Authentication, Hosting, Cloud Storage i Realtime Database. Jedna od najčešće korištenih značajki Firebase-a je Firebase Authentication koji služi za registriranje korisničkih računa, najčešće pomoću e-pošte i lozinke. Ta značajka je korištena u ovoj aplikaciji za registriranje i prijavu korisnika. Na slici 4.5 preuzetoj s [15]

detaljno je prikazan proces autentifikacije korisnika na Google App Engine-u pomoću Firebase-a. Još jedna vrlo bitna značajka korištena u ovoj aplikaciji je baza podataka Realtime Database koja se koristi za spremanje dokumenata različitih formata i također se koristi u ovoj aplikaciji. Spremanje se vrši pomoću JSON (eng. *JavaScript Object Notation*) formata. Taj proces biti će detaljnije pojašnjen u poglavlju 4.



Slika 4.5. Dijagram autentifikacije pomoću Firebase-a [15]

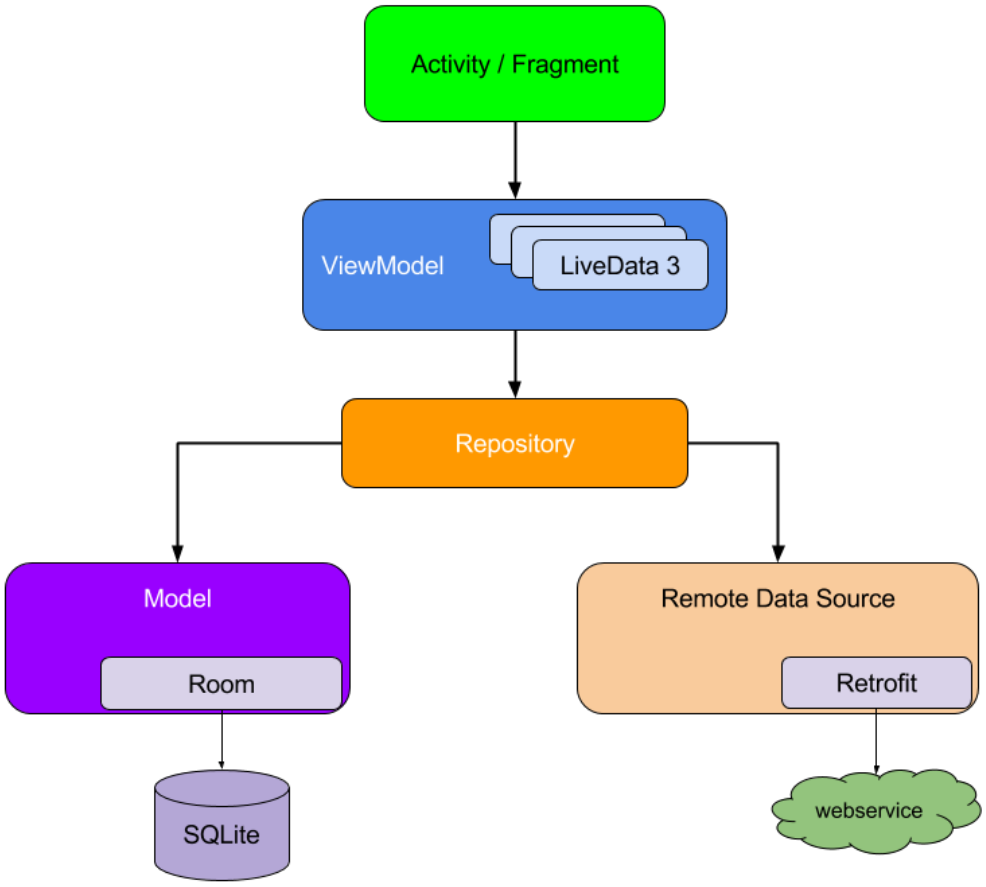
4.2. Predložak mobilne arhitekture MVVM

Mobilna arhitektura je prema [16], skup tehnika i obrazaca koje je potrebno slijediti kako bi se izgradila potpuno strukturirana mobilna aplikacija. Te tehnike i obrasci su formulirani imajući u vidu zahtjeve proizvođača i industrijske standarde. Jedna od tih mobilnih arhitektura je MVVM, odnosno Model-View-ViewModel arhitektura. Najvažniji dijelovi te arhitekture su prema [17], View koji informira ViewModel o korisnikovim aktivnostima, ViewModel koji prikazuje tokove podataka važne za View te radi s DataModel-om kako bi se dobili i spremili podaci i DataModel koji apstrahira izvor podataka. ViewModel prikazuje tokove događaja na koje se View-ovi mogu vezati i tako ViewModel ne mora držati referencu na View. View također obavještava ViewModel o različitim radnjama i tako je podržano dvosmjerno povezivanja podataka između njih i postoji veza više-na-više među njima. ViewModel nema podatke o View-u, a View ima referencu na njega što znači da onaj

koji koristi podatke treba znati o proizvođaču, no on ne treba znati tko koristi podatke. Na slici 4.6 prikazana je struktura klasa ove arhitekture.

Slika 4.6. Struktura klasa MVVM arhitekture

RxJava je prema [18], biblioteka za sastavljanjem asinkronih programa temeljenih na događajima koristeći se takozvanim *Observables*, odnosno promatračima. DataModel koristi promatrače za sastavljanje podataka iz više izvora na sličan način kao baza podataka. DataModel se kreira za svaku značajku aplikacije kako bi se slijedio Single Responsibility Principle. Single Responsibility Principle je smjernica za programiranje koja govori da [19], klasa ili modul trebaju imati jedan i samo jedan razlog za promjenu, što znači i jednu odgovornost u kodu. ViewModel je apstrakcija View-a koja dohvaća potrebne podatke iz DataModel-a, primjenjuje potrebnu logiku i prikazuje bitne podatke koje će View koristiti. ViewModel također prikazuje podatke preko promatrača. View je stvarno korisničko sučelje u aplikaciji koji može biti Activity, Fragment ili neki prilagođeni View. Ovakav način programiranja olakšava zamjenu različitih elemenata korisničkog sučelja s minimalnim promjenama u drugim klasama. Na slici 4.7 preuzetoj sa [20], prikazana je arhitektura MVVM s pojedinačnim dijelovima i kako su oni povezani.



Slika 4.7. Arhitektura MVVM [20]

4.3. Prikaz programskog rješenja po komponentama

U ovom poglavlju biti će prikazani svi dijelovi aplikacije i kod za svaki pojedinačni dio.

4.3.1. Registriranje i prijava korisnika

Zbog korištenja predloška mobilne arhitekture MVVM potrebno je rasporediti funkcionalnosti u fragmente i „View Model“-e. U slučaju registriranja, to su „RegisterFragment“ i „RegisterViewModel“. Kada se na zaslonu pritisne na gumb „SignUp“ „RegisterFragment“ šalje potrebne podatke „RegisterViewModel“-u, nakon toga „RegisterViewModel“ uključuje korutinu „viewModelScope“. Nakon uključenja korutine se poziva funkcija iz „Repository“ koja šalje podatke Firebase-u. Sve to biti će pobliže opisano u nastavku poglavlja. Na slici 4.8 vidljiva je metoda

„initViewModel“ u kojoj je na instanci „registerViewModel“ pozvana metoda „observe“ koja promatra „LiveData“-u i svaki puta kad se ona promijeni, napravi što je potrebno. S obzirom na različite statuse u kojem je trenutno proces, napraviti će određene aktivnosti. Na primjer ako je proces u statusu učitavanja, prikazati će se ikonica učitavanja.

```
private fun initViewModel() {

    if (registerViewModel.currentUser() != null)
        findNavController().popBackStack()

    registerViewModel.createUserLiveData.observe(viewLifecycleOwner, { it: Resource<User>!

        when(it.status) {

            Resource.Status.LOADING -> requireActivity().activity_login_spinner.visibility = View.VISIBLE

            Resource.Status.SUCCESS -> {

                requireActivity().activity_login_spinner.visibility = View.GONE

                findNavController().popBackStack()
            }

            Resource.Status.ERROR -> {

                requireActivity().activity_login_spinner.visibility = View.GONE
                Toast.makeText(requireContext(), it.message, Toast.LENGTH_LONG).show()
            }
```

Slika 4.8. Metoda za inicijalizaciju
„ViewModel“-a u „RegisterFragment“-u

Metoda „registerUser“ iz „RegisterFragment“-a sa slike 4.9 postavlja atribute podatkovne klase „registrationCredentials“ koja se sastoji od imena, adrese e-pošte, lozinke i potvrde lozinke na podatke koje je korisnik unio u polja za unos tih podataka na zaslonu za registriranje. Na instanci „RegisterViewModel“-a pozvana je metoda koja se nalazi u „RegisterViewModel“-u i njoj predan objekt „registrationCredentials“.

```
private fun registerUser() {

    val name = register_edit_text_name.text.toString().trim()
    val email = register_edit_text_email.text.toString().trim()
    val password = register_edit_text_password.text.toString().trim()
    val confirmPassword = register_edit_text_check_password.text.toString().trim()

    val registrationCredentials = RegistrationCredentials(name, email, password, confirmPassword)

    registerViewModel.signUp(registrationCredentials)
}
```

Slika 4.9. Metoda za registriranje korisnika u „RegisterFragment“

Prethodno spomenuta metoda „signUp“ iz „RegisterViewModel“-a vidljiva je na slici 4.10 „ViewModelScope“ je definiran za svaki „ViewModel“, on služi za određivanje kada će se korutine pokrenuti i kada „ViewModel“ nije više aktivan, posao se automatski otkazuje. U ovom slučaju posao će se otkazati kada završi registriranje. Prvo se provjerava jesu li lozinke koje su unesene u polja „password“ i „confirmPassword“ jednake i ako nisu korisniku stiže obavijest da lozinke nisu jednake te se traži ponovan upis lozinke i proces registriranja opet započinje. U slučaju da lozinke odgovaraju provjerava se jesu li popunjena sva potrebna polja za registriranje, ako nisu, korisniku stiže obavijest da ne smije ostaviti prazna polja. Ukoliko su sva potrebna polja popunjena, provjerava se sljedeći uvjet. Ako lozinka ima manje od osam znakova, korisniku se šalje obavijest da lozinka mora biti duga barem osam znakova. Ukoliko su svi prethodno spomenuti uvjeti ispunjeni poziva se metoda iz repozitorija naziva „firebaseSignUp“.

```
fun signUp(registrationCredentials: RegistrationCredentials) {

    viewModelScope.launch { this: CoroutineScope

        if (registrationCredentials.password != registrationCredentials.confirmPassword) {

            _createUserLiveData.value =
                Resource(Resource.Status.ERROR,  data: null,  message: "Passwords don't match!")

        } else if (registrationCredentials.name.isEmpty() || registrationCredentials.email.isEmpty()
            || registrationCredentials.password.isEmpty() || registrationCredentials.confirmPassword.isEmpty()) {

            _createUserLiveData.value =
                Resource(Resource.Status.ERROR,  data: null,  message: "Fields can't be empty!")

        } else if (registrationCredentials.password.length < 8) {

            _createUserLiveData.value =
                Resource(Resource.Status.ERROR,  data: null,  message: "Password has to be at least 8 characters long!")

        } else {

            repository.firebaseSignUp(registrationCredentials)
                .onStart { _createUserLiveData.value = Resource(Resource.Status.LOADING,  data: null,  message: null) }
                .catch { _createUserLiveData.value = Resource(Resource.Status.ERROR,  data: null, it.localizedMessage) }
                .collect { _createUserLiveData.value = it }
        }
    }
}
```

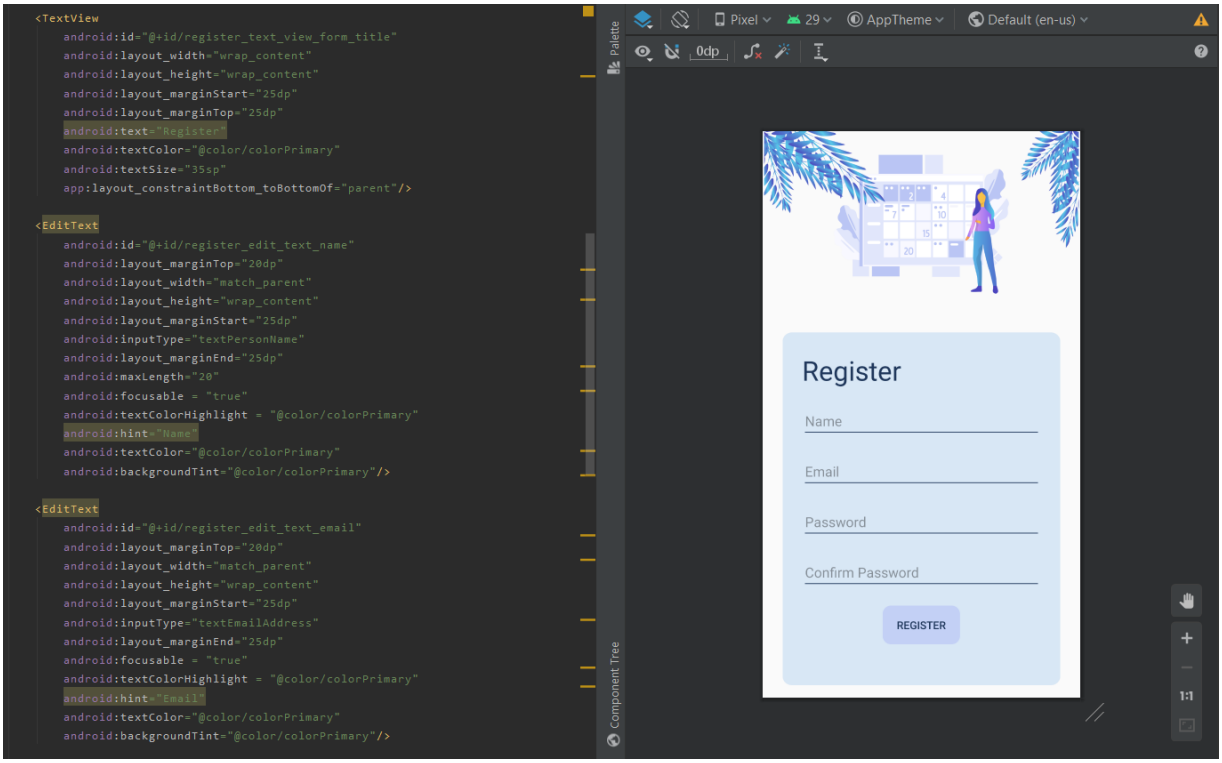
Slika 4.10. Metoda za registriranje korisnika
u „RegisterViewModel“

Metoda „firebaseSignUp“ nalazi se u „Repository“ i prikazana je na slici 4.11. Ta metoda, isto kao „signUp“ metoda, prima objekt podatkovne klase „RegistrationCredentials“ i vraća „Flow“. „Flow“ [21], je asinkroni tok podataka koji služi za davanje obavijesti što se događa dok Firebase dohvaća podatke. Funkcija Firebase Auth-a „createUserWithEmailAndPassword“ prima adresu e-pošte i lozinku koje su dobivene iz „RegisterFragment“-a. Ako je registriranje uspješno u bazu se postavlja vrijednost atributa „first_time“ na „true“ zbog prikazivanja zaslona ankete i ostali atributi koji će biti potrebni za ispravan rad aplikacije. Rad s bazom podataka Firebase biti će pobliže opisan u poglavlju 4.3.3. Iz polja za unos imena sa zaslona aplikacije, metodom iz Firebase „setDisplayName“ postavlja se ime korisnika na zaslonu profila. Ako je status procesa registriranja da je uspješan, korisniku se šalje obavijest o uspješnosti, a ako dođe do pogreške, također se šalje obavijest o tome što se dogodilo.

```
fun firebaseSignUp(registrationCredentials: RegistrationCredentials): Flow<Resource<User>> = callbackFlow { this: ProducerScope<Resource<Noth

    val listener = firebaseAuth.createUserWithEmailAndPassword(
        registrationCredentials.email,
        registrationCredentials.password
    ).addOnCompleteListener { it: Task<AuthResult!>

            if (it.isSuccessful) {

                val firebaseUser = firebaseCurrentUser()
                if (firebaseUser != null) {

                    getUsers().child(firebaseUser.uid).child( pathString: "first_time").setValue(true)
                    getUsers().child(firebaseUser.uid).child( pathString: "tasks").child( pathString: "0").child( pathString: "id").setValue(0)
                    getUsers().child(firebaseUser.uid).child( pathString: "refresh").setValue(0L)

                    val updateUserInfo =
                        UserProfileChangeRequest.Builder()
                            .setDisplayName(registrationCredentials.name)
                            .build()

                    firebaseUser.updateProfile(updateUserInfo)
                    firebaseUser.sendEmailVerification()
                }

                offer(Resource(Resource.Status.SUCCESS,  data: null,  message: "Sign up successful!"))

            } else {

                offer(Resource.error(it.exception?.localizedMessage!!,  data: null))
            }
    }
    awaitClose { listener.addOnCompleteListener{} }
}
```

Slika 4.11. Metoda za registriranje korisnika
u „Repository“

Dio XML koda za izgled zaslona za registriranje je vidljiv na slici 4.12 zajedno sa samim izgledom zaslona za registriranje.



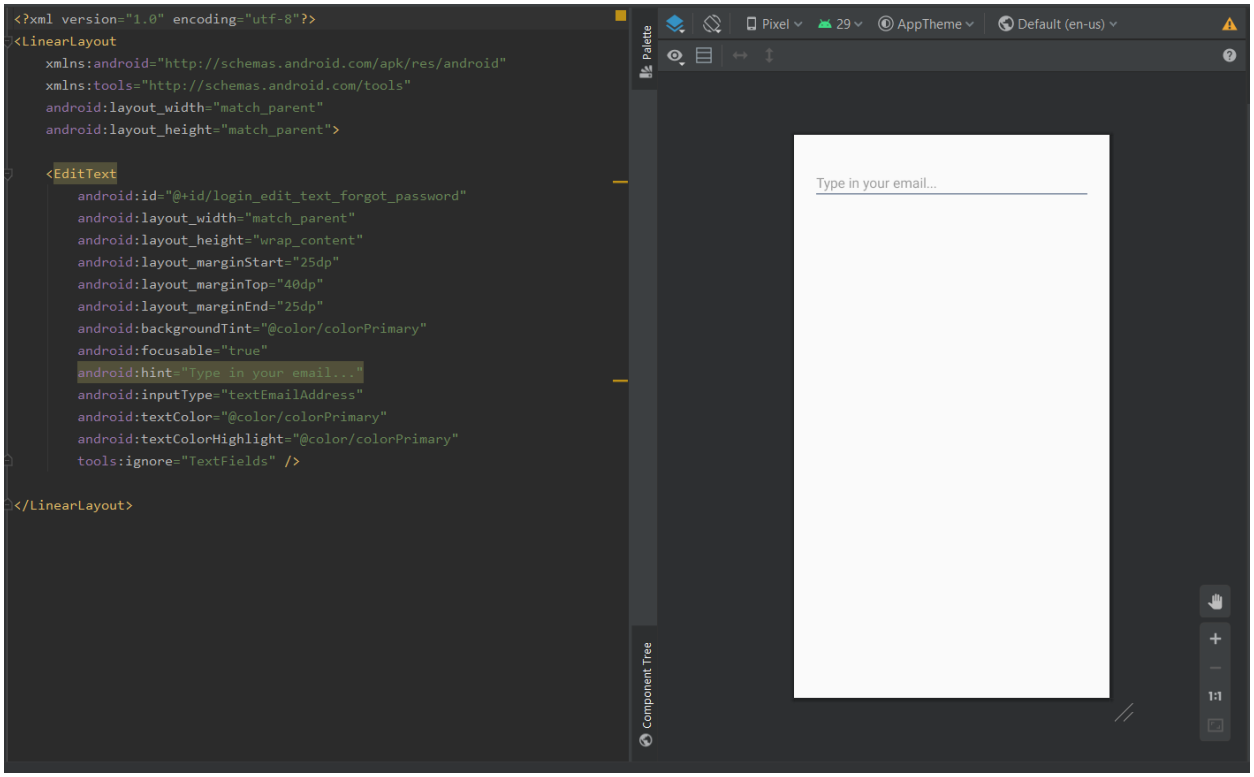
Slika 4.12. Dio XML koda
„fragment_register“-a i izgled zaslona

Fragment, ViewModel i sveukupan kod za prijavu korisnika vrlo je sličan kodu za registriranje korisnika. Metoda „initViewModel“ u „LoginFragment“-u jednaka je metodi istog imena u „RegisterFragment“-u. Tijek slanja podataka funkcionira na isti način kao i pri registriranju korisnika. Metoda „initListeners“ sa slike 4.13 postavlja „listener“-e na gumbe sa zaslona za prijavu koji služe za odrađivanje određene funkcionalnosti kada se pritisne na gumb. Kada je pritisnut gumb „Sign in“, pozvana je metoda sa slike 4.14 koja će biti objašnjena u nastavku poglavlja. Kada je pritisnut gumb „Forgot password“ otvara se dijalog za upis adrese e-pošte čiji je kod prikazan na slici 4.15. U metodi „initListeners“ podešen je dijalog za slanje hiperveze na upisanu adresu e-pošte kako bi se omogućila promjena lozinke. Promjena lozinke je omogućena metodom „forgotPassword“ koja se nalazi u „Repository“ i iz njega se poziva funkcija Firebase-a istog imena. Treći gumb imena „Sign up“ vodi na zaslon za registriranje korisnika.

Slika 4.13. Metoda „initListeners“ iz „LoginFragment“

Prethodno spomenuti gumb „Sign in“ poziva metodu „signIn“ iz „LoginViewModel“-a vidljivu na slici 4.14 u kojoj se postavljaju potrebni podaci za prijavu, adresa e-pošte i lozinka. Ti podaci se predaju instanci „LoginViewModel“-a koji poziva metodu „signIn“ iz repozitorija koja radi na vrlo sličan način kao metoda „signUp“ iz repozitorija.

Slika 4.14. Metoda „signIn“ iz „LoginViewModel“



Slika 4.15. XML kod dijaloga za obnovu lozinke

4.3.2. Anketa o navikama korisnika

Anketa o navikama korisnika sastoji se od osam fragmenata. Prvi fragment je „IntroductionFragment“ na kojemu korisnik ima izbor želi li odgovoriti na pitanja iz ankete ili ne. Na slici 4.16 prikazana je klasa „IntroductionFragment“ i njena metoda „initListeners“ koja postavlja „listener“-e na gumbe koji se nalaze na tom zaslonu. O tome koji je gumb pritisnut ovisi hoće li idući fragment biti „HobbiesFragment“, što bi značilo da je korisnik izabrao opciju da želi riješiti anketu, ili „SummaryFragment“ koji je posljednji fragment ankete i s njega se ulazi u samu aplikaciju i time preskače rješavanje ankete što je vidljivo u samom kodu.

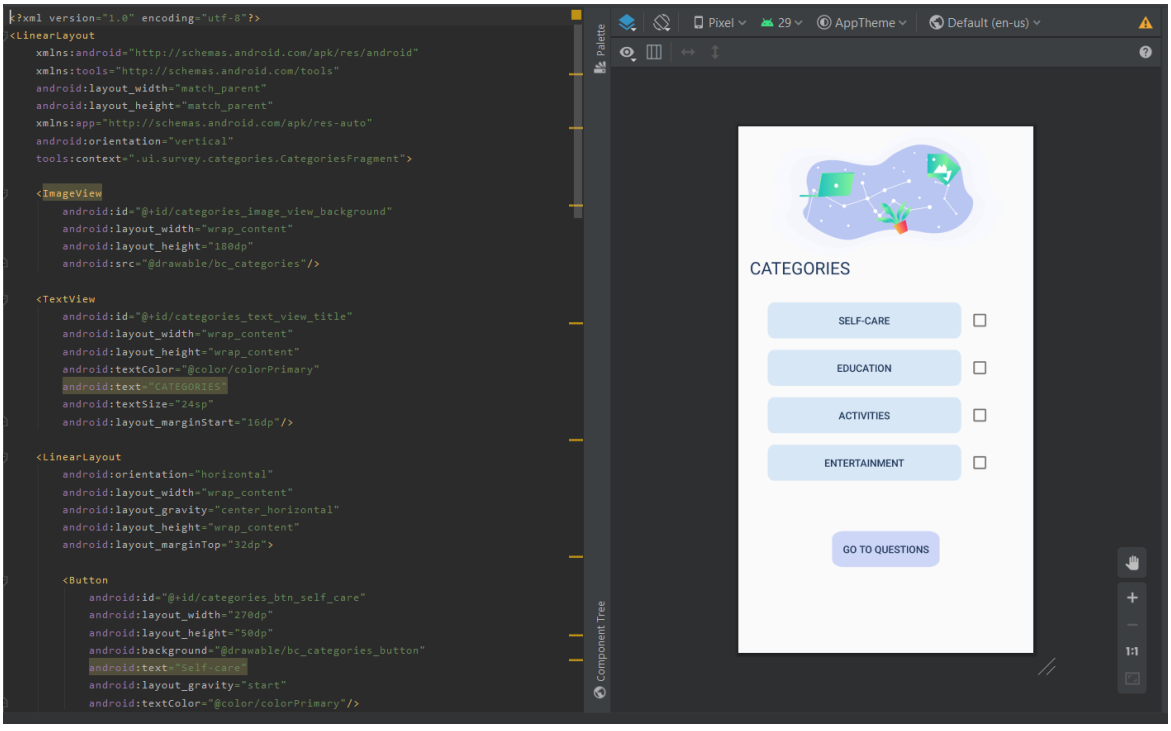
```
class IntroductionFragment : Fragment(R.layout.fragment_introduction) {

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        initListeners()
    }

    private fun initListeners() {

        introduction_btn_skip.setOnClickListener { it: View!

            findNavController().navigate(IntroductionFragmentDirections.actionHobbiesFragmentToSummaryFragment( state: false))
        }

        introduction_btn_continue.setOnClickListener { it: View!
            findNavController().navigate(IntroductionFragmentDirections.actionIntroductionFragmentToHobbiesFragment())
        }
    }
}
```

Slika 4.16. Klasa „IntroductionFragment“

Ukoliko je korisnik izabrao riješiti anketu, sljedeći fragment koji se otvara je „HobbiesFragment“ na kojemu korisnik označuje koji ga hobiji zanimaju. Na tom fragmentu nalazi se gumb koji vodi na sljedeći fragment, „CategoriesFragment“. Na zaslonu „Categories“ korisnik ima mogućnost izabrati iz koje kategorije želi imati postavljena pitanja. Na slici 4.17 prikazana je metoda iz „CategoriesFragment“. Ta metoda pomoću „onClickListener“ provjerava koje su kategorije označene. Izgled XML koda za „CategoriesFragment“ prikazan je na slici 4.18. Ako korisnik nije izabrao nijednu kategoriju, dobije obavijest da mora izabrati barem jednu. U slučaju da korisnik ima barem jednu kategoriju izabranu, s obzirom koje su kategorije izabrane, ti će se fragmenti otvoriti.

```
private fun initListeners() {

    categories_btn_continue.setOnClickListener { it: View!

        val checkedCategories = listOf(
            categories_checkbox_self_care.isChecked,
            categories_checkbox_education.isChecked,
            categories_checkbox_activities.isChecked,
            categories_checkbox_entertainment.isChecked
        )
        val categories = mutableListOf<Int>()

        checkedCategories.forEachIndexed { index, b ->

            if (b)
                categories.add(index)
        }

        categoriesViewModel.setCategories(categories)

        if (categories.isEmpty())
            Toast.makeText(requireContext(),  text: "You have to select at least one category.", Toast.LENGTH_SHORT).show()
        else
            when(categories.first()) {

                0 -> findNavController().navigate(CategoriesFragmentDirections.actionCategoriesFragmentToSelfCareFragment(categories.toIntArray()))
                1 -> findNavController().navigate(CategoriesFragmentDirections.actionCategoriesFragmentToEducationFragment(categories.toIntArray()))
                2 -> findNavController().navigate(CategoriesFragmentDirections.actionCategoriesFragmentToActivitiesFragment(categories.toIntArray()))
                3 -> findNavController().navigate(CategoriesFragmentDirections.actionCategoriesFragmentToEntertainmentFragment())
            }
    }
}
```

Slika 4.17. Metoda „initListeners“ iz „CategoriesFragment“



Slika 4.18. Dio XML koda „fragment_categories“-a i izgled zaslona

Fragmenti „SelfCareFragment“, „EducationFragment“, „ActivitiesFragment“ i „EntertainmentFragment“ su vrlo slični. Sastoje se od različitih pitanja, odgovora i preporuka koje se nude s obzirom na odgovore. Na primjeru „SelfCareFragment“-a u nastavku će biti objašnjen način na koji taj dio ankete funkcionira.

„SelfCareFragment“ također ima metodu „initListeners“ koja je prikazana na slici 4.19. Atribut „category“ se postavlja u 0 jer je ova kategorija nulti element liste kategorija. Zatim se pomoću metode „setOnClickListener“ provjerava koji je od ponuđenih odgovora izabran i nakon toga se otvara sljedeći fragment, fragment iz neke druge kategorije ili posljednji fragment ankete, ovisno o tome je li ovo jedina kategorija koju je korisnik izabrao. Metoda „setAnswer“ se poziva u „QuestionsViewModel“-u koji koriste svi fragmenti s kategorijama. Ta je metoda prikazana na slici 4.20.

```
private fun initListeners(){

    val category = 0

    self_care_btn_answer1.setOnClickListener { it: View!

        questionsViewModel.setAnswer( answer: 0, category)
        navigateNextQuestion()
    }

    self_care_btn_answer2.setOnClickListener { it: View!

        questionsViewModel.setAnswer( answer: 1, category)
        navigateNextQuestion()
    }

    self_care_btn_answer3.setOnClickListener { it: View!

        questionsViewModel.setAnswer( answer: 2, category)
        navigateNextQuestion()
    }

    self_care_btn_answer4.setOnClickListener { it: View!

        questionsViewModel.setAnswer( answer: 3, category)
        navigateNextQuestion()
    }

    self_care_btn_answer5.setOnClickListener { it: View!

        questionsViewModel.setAnswer( answer: 4, category)
        navigateNextQuestion()
    }

}
```

Slika 4.19. Metoda „initListeners“ iz „SelfCareFragment“

```
fun setAnswer(answer: Int, category: Int) {

    repository.setAnswer(answer, category)
}
```

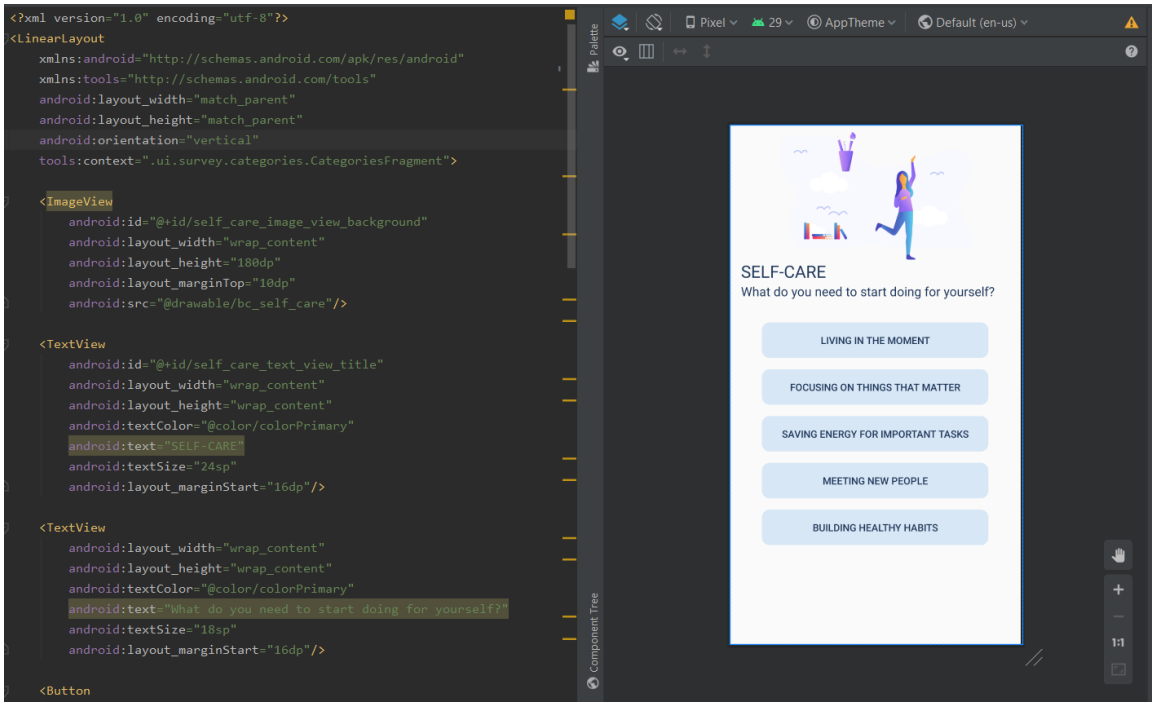
Slika 4.20. Metoda „setAnswer“ iz „QuestionsViewModel“

Metoda „setAnswer“ iz „QuestionsViewModel“ na instanci „Repository“-a poziva metodu istog imena koja se nalazi u repozitoriju i predaje joj broj odgovora i broj kategorije. Metoda istog imena iz „Repository“ prikazana je na slici 4.21. Ta metoda koristi Firebase-ovu metodu „firebaseCurrentUser“ kako bi u bazu unijela odgovore korisnika i na temelju toga prikazala preporuke korisniku. Rad s bazom biti će pobliže objašnjen u poglavlju 4.3.3.

```
fun setAnswer(answer: Int, category: Int) {

    getUsers().child(firebaseCurrentUser()?.uid!!).child( pathString: "profile").child( pathString: "categories").child(
        category.toString()
    ).setValue(answer)
}
```

Slika 4.21. Metoda „setAnswer“ iz „Repository“

Izgled „SelfCareFragment“-a i XML kod kojim je postignut taj izgled prikazan je na slici 4.22. XML kod ostalih fragmenata kategorija je jednak, a sveukupan izgled vrlo sličan.



Slika 4.22. Dio XML koda
„fragment_self_care“-a i izgled zaslona

Posljednji fragment ankete je „SummaryFragment“ koji ima metodu „initViewModel“ čiji je kod vidljiv na slici 4.23. Ta metoda poziva dvije metode koje se nalaze u „SummaryViewModel“-u u kojem se pozivaju istoimene metode iz „Repository“ prikazane na slici 4.24.

```
private fun initViewModel() {

    summaryViewModel.removeFirstTimeUser()
    summaryViewModel.removeSurveyFromUser(args.state)

}
```

Slika 4.23. Metoda „initViewModel“ iz
„SummaryFragment“

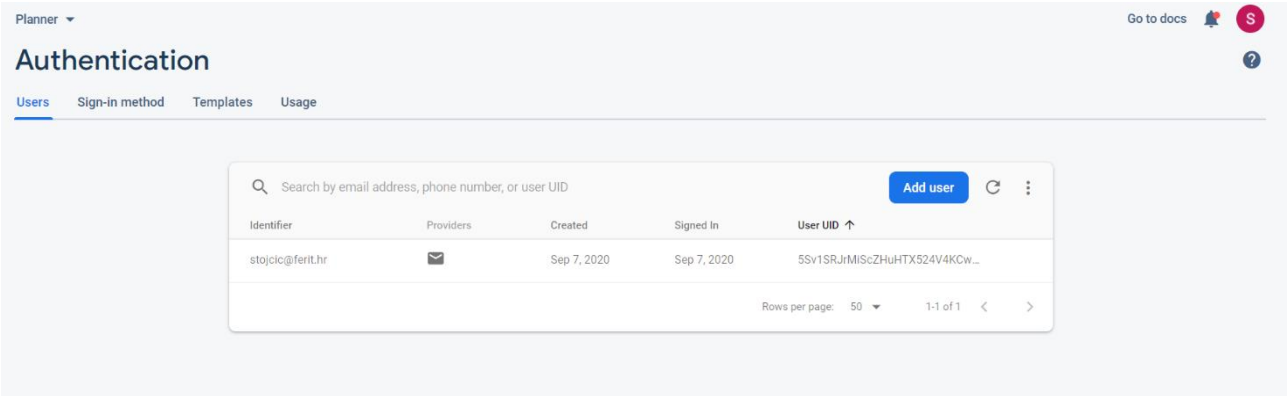
```
fun removeFirstTimeUser() {

    getUsers().child(firebaseCurrentUser()?.uid!!).child( pathString: "first_time").setValue(false)
}

fun removeSurveyFromUser(survey: Boolean) {

    getUsers().child(firebaseCurrentUser()?.uid!!).child( pathString: "survey").setValue(survey)
}
```

Slika 4.24. Metode iz „Repository“ vezane uz „SummaryViewModel“

Prethodno spomenute metode iz „SummaryViewModel“ se pozivaju na repozitoriju te se iz „Repository“ pozivaju na Firebase-u i postavljaju potrebne atribute za kasniji rad s bazom i preporukama što će biti opisano u poglavlju 4.3.3.

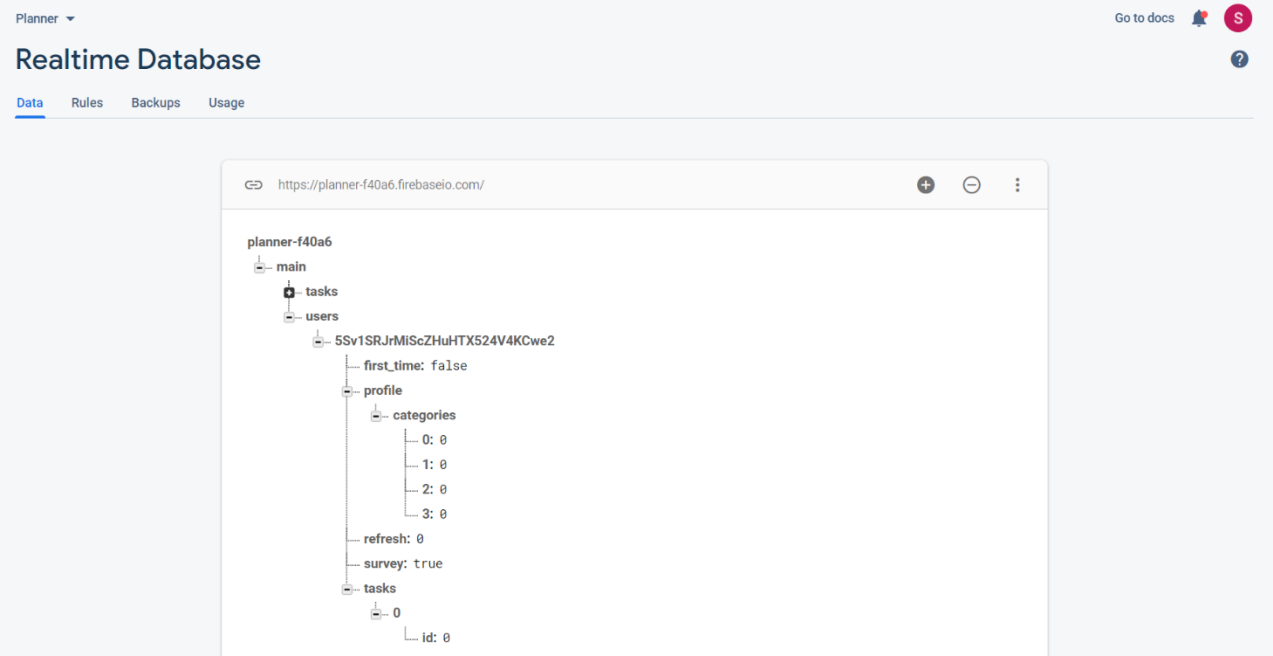
4.3.3. Rad s bazom podataka Firebase

U ovoj aplikaciji korištena su dvije usluge baze podataka Firebase, Firebase Auth i Realtime Database. Firebase Auth korišten je za autentifikaciju korisnika, odnosno njihovo registriranje i prijavljivanje u aplikaciju, a Realtime Database za čuvanje podataka o korisniku te svih podataka ankete. Sučelje Firebase Auth-a s korisnicima aplikacije i njihovim jedinstvenim UID oznakama prikazano je na slici 4.25. Firebase je povezan s aplikacijom pomoću „Repository“ i samo s njim komunicira te on šalje podatke ostalim dijelovima aplikacije.



Slika 4.25. Sučelje Firebase Auth-a

Firebase Auth i Realtime Database povezani su putem UID oznake korisnika. Svaki korisnik ima u Realtime Database spremljene podatke o njemu koji uključuju njegov UID, atribut „first_time“ koji u aplikaciji služi da bi se znalo je li potrebno pokretati anketu ili to nije korisnikova prva prijava pa će otvoriti početni zaslon same aplikacije i preskočiti anketu. Atribut „categories“ prikazuje brojeve kategorija i brojeve odgovora koje je korisnik izabrao, nula predstavlja prvu kategoriju, odnosno „Self care“, a nula u njemu predstavlja nulti odgovor na postavljeno pitanje. Pomoću tih brojeva predložene su preporuke korisniku, čiji je dio u bazi prikazan na slici 4.27. Atribut „refresh“ označava je su li već taj dan predložene aktivnosti i događaji korisniku, ako je prošlo dvadeset četiri sata, aplikacija će predložiti nove nasumične aktivnosti s obzirom na odgovore korisnika. Sljedeći atribut je „survey“ koji označava je li korisnik na početnom zaslonu ankete pristao riješiti ju ili nije. Posljednji atribut je „tasks“ u koji se upisuju sve aktivnosti i događaji koje je korisniku preporučila aplikacija i svi koje je on sam dodao. Na slici 4.26 prikazan je Realtime Database dio koji prikazuje podatke o korisniku.



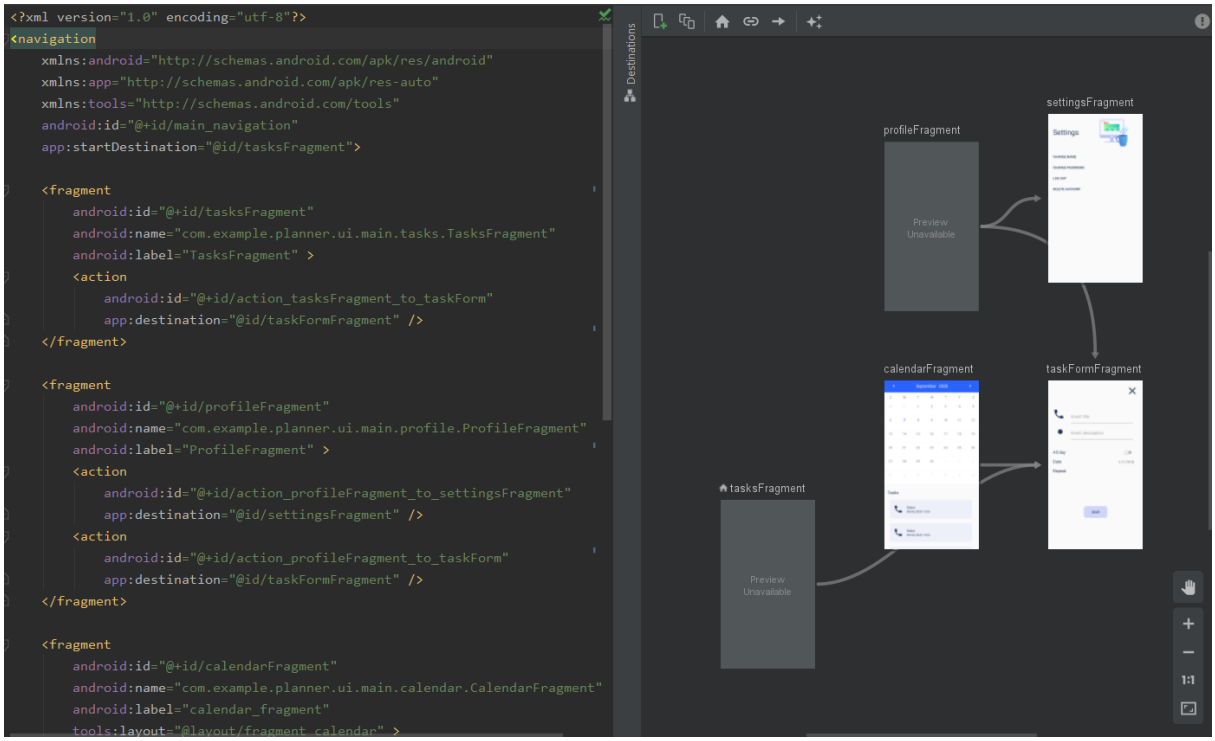
Slika 4.26. Sučelje Realtime Database-a s podacima o korisniku

Slika 4.27. Sučelje Realtime Database-a s podacima o preporukama

Na slici 4.27 prikazan je dio preporuka po kategorijama. U Realtime Database, točnije u dijelu baze u kojem je popis svih preporuka su brojevi prema kojem se biraju koje će se preporuke prikazati korisniku. Detaljan popis svih preporuka napisan je u obliku tablica u poglavlju 3.4.4.

4.3.4. Popis zadataka korisnika i događaji u blizini

Popis zadataka korisnika i događaji u blizini prikazani su na prvom zaslonu aplikacije koji se otvori nakon prijave kada je anketa riješena, odnosno na „TasksFragment“-u. „TasksFragment“ je pomoću navigacije povezan sa kalendarom, to jest „CalendarFragment“, koji će biti opisan u poglavlju 4.3.5. i sa profilom korisnika, odnosno „ProfileFragment“, koji će biti opisan u poglavlju 4.3.6. Na slici 4.28 prikazan je dio XML koda i način na koji su tri prethodno spomenuta fragmenta povezana međusobno i kako su povezani s fragmentima „SettingsFragment“ i „TaskFormFragment“ koji će biti opisani u nadolazećim poglavljima.




```
<?xml version="1.0" encoding="utf-8"?>
<navigation
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main_navigation"
    app:startDestination="@id/tasksFragment">

    <fragment
        android:id="@+id/tasksFragment"
        android:name="com.example.planner.ui.main.tasks.TasksFragment"
        android:label="TasksFragment" >
        <action
            android:id="@+id/action_tasksFragment_to_taskForm"
            app:destination="@id/taskFormFragment" />
    </fragment>

    <fragment
        android:id="@+id/profileFragment"
        android:name="com.example.planner.ui.main.profile.ProfileFragment"
        android:label="ProfileFragment" >
        <action
            android:id="@+id/action_profileFragment_to_settingsFragment"
            app:destination="@id/settingsFragment" />
        <action
            android:id="@+id/action_profileFragment_to_taskForm"
            app:destination="@id/taskFormFragment" />
    </fragment>

    <fragment
        android:id="@+id/calendarFragment"
        android:name="com.example.planner.ui.main.calendar.CalendarFragment"
        android:label="calendar_fragment"
        tools:layout="@layout/fragment_calendar" >
```

Slika 4.28. Dio XML koda „main_navigation“-a

Fragment „TasksFragment“ sadržava metodu „initViewModel“ koja je prethodno spomenuta u potpoglavlju 4.3.1 pa neće biti opet opisana u ovom potpoglavlju. Također sadržava metodu „initViewPager“ koju se može vidjeti na slici 4.26. Ta metoda, kao što njeno ime govori, inicijalizira „View Pager“ [22], koji se koristi za micanje među fragmentima unutar „TasksFragment“-a. „Tab Layout“ [23], se koristi zajedno s „View Pager“-om za prebacivanje fragmenata kada se pritisne na određeni „tab“, odnosno prozor. Metodi „TabLayoutMediator“ sa slike 4.29 predaju se komponente „View Pager“ i „Tab Layout“ koje su napravljene u XML datoteci. Unutar te metode se određuju imena svakog prozora.

Slika 4.29. Metoda „initViewPager“ iz „TasksFragment“

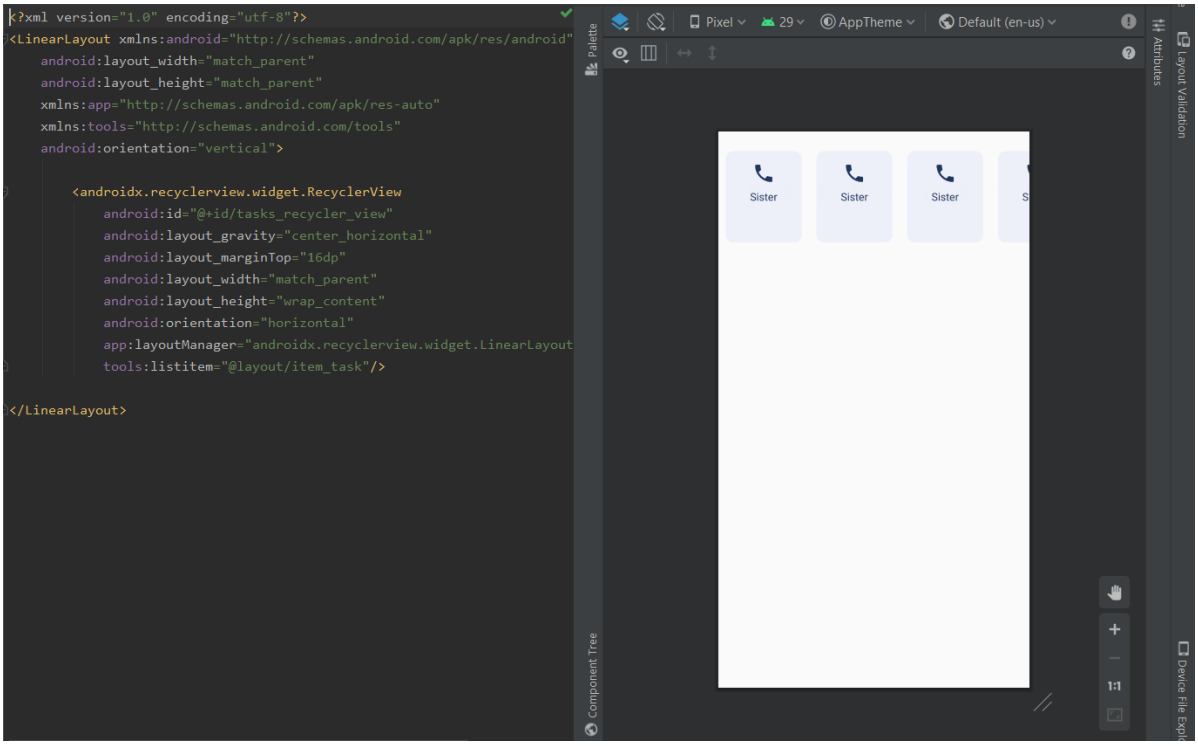
Kako bi se mogli prikazati zadaci, potrebno je napraviti adapter za njih, a to je adapter imena „TasksViewPagerAdapter“. Unutar metode sa slike 4.29, „initViewPager“, postavljen je prethodno spomenuti adapter. Kod adaptera prikazan je na slici 4.30. Taj adapter kreira, odnosno prikazuje, željeni fragment ovisno o tome koji je prozor pritisnut. Jedan od ta tri fragmenta otvara se unutar „TasksFragment“-a i prikazuje popis zadataka korisnika.

Kako bi se „Recycler View“ mogao prikazati unutar „View Pager“-a potrebno je, kao i za „View Pager“, napraviti adapter. Adapterom se povezuje XML kod zadataka i njihovog „Recycler View“-a sa podatkovnom klasom zadatka.

```
class TasksViewPagerAdapter (fragment: Fragment, private val count: Int) : FragmentStateAdapter(fragment) {

    override fun getItemCount(): Int {
        return count
    }

    override fun createFragment(position: Int): Fragment {

        return when (position) {

            0 -> TasksTodayFragment()

            1 -> TasksThisWeekFragment()

            2 -> TasksThisMonthFragment()

            else -> TasksTodayFragment()
        }
    }
}
```

Slika 4.30. Klasa „TasksViewPagerAdapter“

Zadatak je napravljen pomoću XML koda te da bi se popis zadataka mogao *scrollati* bilo je potrebno napraviti „Recycler View“ za njih. Taj kod vidljiv je na slici 4.31.



Slika 4.31. XML kod „tasks_recycler_view“-a

Na zaslonu sa zadacima i događajima u blizini, ispod zadataka koje korisnik ima, nalazi se popis događaja u blizini. Ideja je bila da se pomoću vanjske usluge, odnosno korištenjem API-ja neke druge aplikacije koja se koristi za pretraživanje događaja, za ovu aplikaciju pronađu događaji ovisno o lokaciji. No, s obzirom na trenutnu situaciju u svijetu i koronavirus, ne odvijaju se skoro nikakvi događaji i bilo je nemoguće na taj način koristiti aplikaciju. U aplikaciju je dodana podatkovna klasa „Event“ i ručno uneseni neki izmišljeni događaji. Podatkovna klasa „Event“ prikazana je na slici 4.32.

```
data class Event (
    var title: String,
    var description: String,
    var location: String,
    var distance: Float
)
```

Slika 4.32. Podatkovna klasa „Event“

Fragment „TasksFragment“ ima metodu „initRecyclerView“ koja je prikazana na slici 4.33. U toj metodi, kao što samo ime govori, inicijalizira se „Recycler View“ za događaje, odnosno objekte „Events“ podatkovne klase. Tom adapteru predaje se lista događaja. Ta lista događaja je napisana u repozitoriju i iz repozitorija je poziva instanca „TaskViewModel“-a koja sadrži metodu „getEvents“, istog imena kao i metoda u repozitoriju. Na sam „Recycler View“ koji se nalazi u XML datoteci, postavlja se prethodno podešeni adapter.

```
private fun initRecyclerView() {

    tasksEventRecyclerAdapter = TasksEventRecyclerAdapter()
    tasksEventRecyclerAdapter.submitList(tasksViewModel.getEvents())
    tasksEventRecyclerAdapter.notifyDataSetChanged()

    events_recycler_view.adapter = tasksEventRecyclerAdapter
}
```

Slika 4.33. Metoda „initRecyclerView“ iz „TasksFragment“

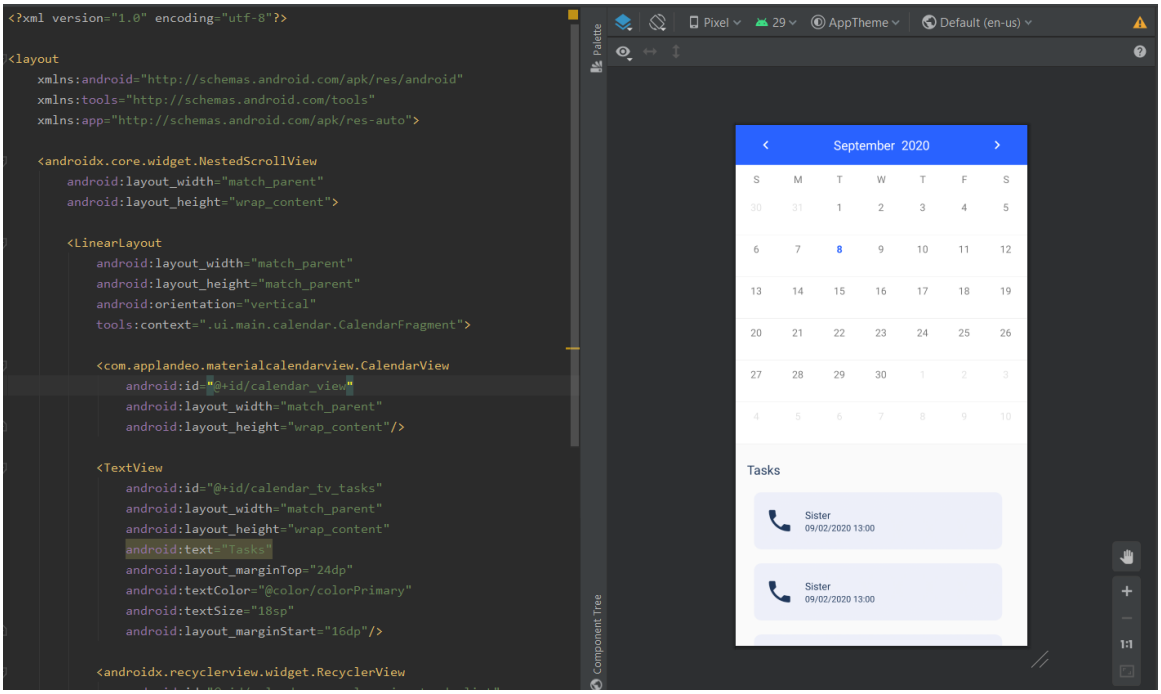
4.3.5. Kalendar i zadaci

Pritiskom na srednji gumb u navigaciji koja je vidljiva na dnu same aplikacije, otvara se zaslon s kalendarom. „CalendarFragment“ također ima prethodno opisanu metodu „initViewModel“. Osim „ViewModel“-a, potrebno je inicijalizirati sam kalendar. Metoda za inicijalizaciju kalendara vidljiva je na slici 4.34. Na „calendar_view“-u koji je postavljen u XML datoteci, pozvana je metoda „setOnDayClickListener“ i unutar nje prepisana metoda „onDayClick“ kojoj je predana podatkovna klasa „EventDay“. Unutar te metode pozvana je metoda „getCurrentDayTasks“ koja se nalazi u „CalendarViewModel“ i prikazuje zadatke tog dana. „CalendarViewModel“, kao „ViewModel“-i inače, radi s repozitorijem koji radi direktno s bazom i na taj način „CalendarFragment“ dobiva i prikazuje potrebne podatke. Kalendar koji je korišten u ovoj aplikaciji je „MaterialCalendarView“, [24].

```
private fun initCalendar() {

    calendar_view.setOnDayClickListener( object : OnDayClickListener {

        override fun onDayClick(eventDay: EventDay) {

            calendarViewModel.getCurrentDayTasks(eventDay.calendar.time)
        }
    })
}
```

Slika 4.34. Metoda „initCalendar“ u „CalendarFragment“

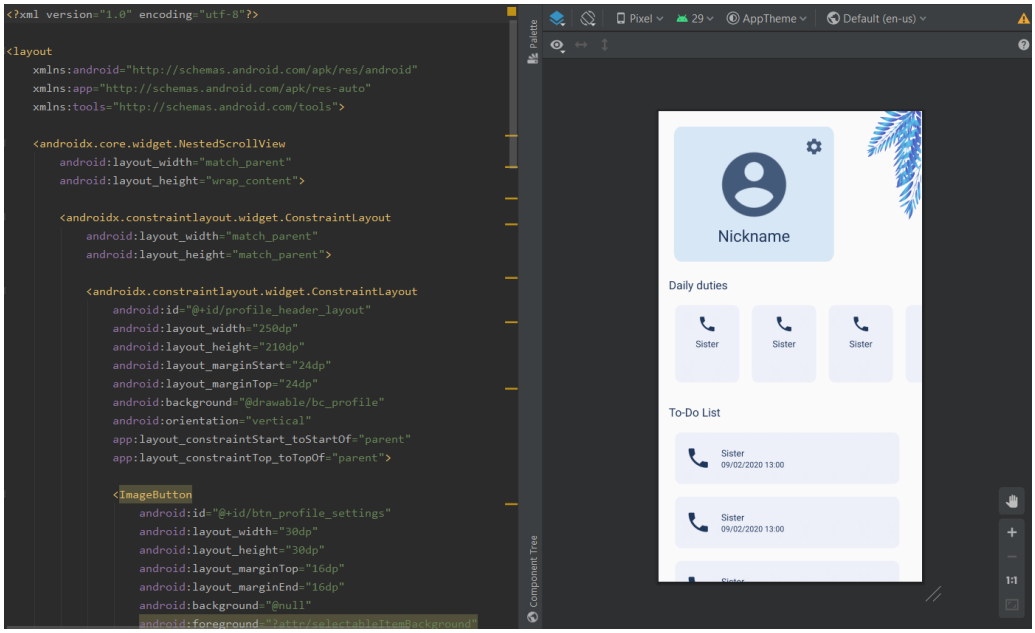
Fragment „CalendarFragment“ sastoji se od kalendara i „Recycler View“-a za prikaz aktivnosti dana na koji korisnik pritisne. Dio XML koda i izgled zaslona s kalendarom prikazan je na slici 4.35. Kako bi se zadaci mogli prikazati i *scrollati* unutar „Recycler View“-a, napravljen je adapter kao i na „TasksFragment“-u. Jedan prikazani zadatak na kalendaru sastoji se od naslova aktivnosti ili događaja, datuma kojeg se održavaju i ikonice koja prikazuje koja je vrsta aktivnosti ili događaja u pitanju.



Slika 4.35. Dio XML koda
„fragment_calendar“-a i izgled zaslona

4.3.6. Profil korisnika

Pritiskom na posljednji gumb u navigaciji koja je vidljiva na dnu same aplikacije, otvara se zaslon s profilom korisnika. Kao i dosad fragmenti, „ProfileFragment“ ima metode za inicijalizaciju „listener“-a, „ViewModel“-a i „Recycler View“-a. Na profilu se također koristi prethodno opisani „View Pager“ i „Tab Layout“ koji se koriste zajedno s „Recycler View“-om za prikaz zadataka koje aplikacija predlaže korisniku, odnosno dnevnih zadataka. Osim tog „Recycler View“-a, na profilu se koristi još jedan i to za prikaz popisa svih zadataka i događaja koje je korisnik zadao sebi i koje mu je aplikacija preporučila. Dio XML koda i prikaz zaslona profila prikazan je na slici 4.36.



Slika 4.36. Dio XML koda
„fragment_profile“-a i izgled zaslona

Osim dnevnih zadataka i popisa zadataka korisnika, na profilu se može postaviti profilna fotografija. Za rad sa slikom i dodavanje slike iz galerije koristi se „Glide“ [25]. Kod za dodavanje profilne slike nalazi se u „ProfileFragment“-u te je prikazan na slici 4.37.

```
private fun setProfilePicture(imageUri: Uri?) {

    Glide.with(profile_iv_profile_photo.context)
        .asDrawable()
        .load(imageUri)
        .apply(
            RequestOptions()
                .circleCrop()
                .placeholder(R.drawable.ic_profile)
                .error(R.drawable.ic_profile)
        )
        .into(profile_iv_profile_photo)
}
```

Slika 4.37. Metoda „setProfilePicture“ iz
„ProfileFragment“

Pritiskom na gumb za postavke na profilu, otvara se novi fragment. „SettingsFragment“ ima mogućnosti promjene imena i lozinke korisnika, odjavu korisnika te brisanje cijelog korisničkog profila. Za sve spomenuto, osim za odjavu korisnika, pritiskom na gumb otvara se dijalog u kojem se upisuje što je potrebno i pritisne na gumb za potvrdu ili gumb za odbacivanje promjene. Metoda za promjenu lozinke nalazi se u „SettingsViewModel“ i povezana je s repozitorijem i bazom podataka Firebase. Kod za promjenu lozinke vidljiv je na slici 4.38. Kao i pri kreiranju lozinke, osigurano je da lozinke unesene u polja „password“ i „confirmPassword“ budu jednake, polja popunjena i da se lozinka sastoji od barem 8 znakova. Na instanci „Repository“-a pozvana je metoda „changePassword“ koja provodi promjenu lozinke.

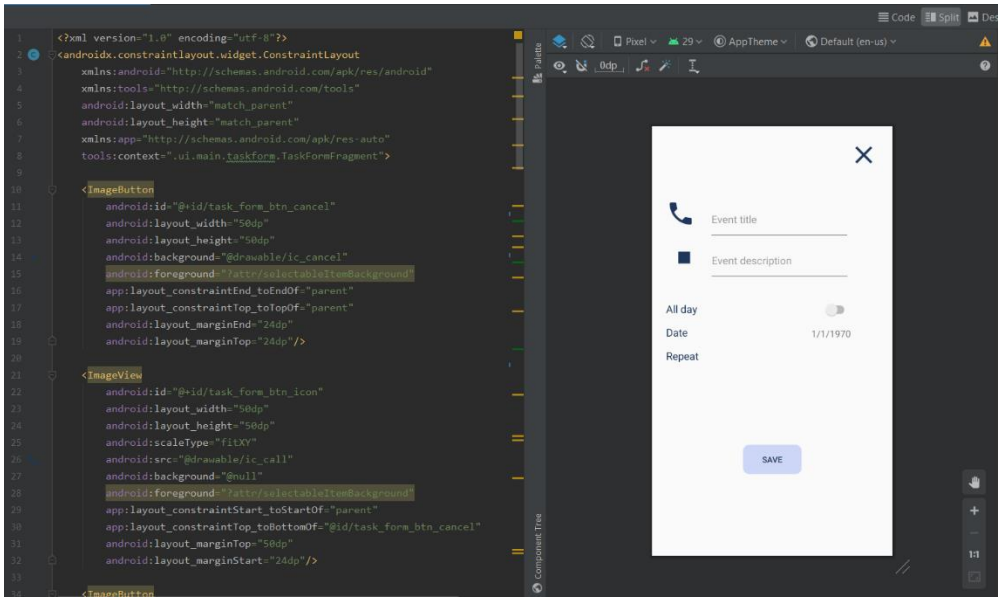
```
fun changePassword(password: String, confirmPassword: String) {

    viewModelScope.launch { this: CoroutineScope

        if (password != confirmPassword) {

            _changePasswordLiveData.value =
                Resource.error( message: "Passwords don't match!")

        } else if (password.isEmpty() || confirmPassword.isEmpty()) {

            _changePasswordLiveData.value =
                Resource.error( message: "Fields can't be empty!")

        } else if (password.length < 8) {

            _changePasswordLiveData.value =
                Resource.error( message: "Password has to be at least 8 characters long!")

        } else {

            repository.changePassword(password)
                .onStart { _changePasswordLiveData.value = Resource.loading() }
                .catch { _changePasswordLiveData.value = Resource.error( it.localizedMessage) }
                .collect { _changePasswordLiveData.value = it }
        }
    }
}
```

Slika 4.38. Metoda „setProfilePicture“ iz „ProfileFragment“

4.3.7. Dodavanje zadatka u raspored

Posljednji fragment u aplikaciji je „TaskFormFragment“ koji se otvara pomoću „Floating Action Button“-a [26], koji se nalazi u donjem desnom kutu zaslona sa zadacima i događajima, zaslona s kalendarom i zaslona korisničkog profila. Dio XML koda i zaslon „TaskFormFragment“-a vidljiv je

na slici 4.39. Na zaslonu postoje mogućnosti biranja ikone zadatka, njegovog naslova i opisa, „Switch“ [27], koji se koristi za biranje jesu li aktivnost ili događaj cjelodnevni ili se odvijaju u određeno vrijeme. Ispod „Switch“-a bira se datum događaja ili aktivnosti i posljednja opcija koja se može izabrati je unutar „Spinner“-a [28], i korisnik može izabrati je li to događaj ili aktivnost koji se odvijaju jednom, svaki dan, jednom mjesečno ili jednom godišnje. Inicijalizacija „Spinner“-a odvija se u „TaskFormFragment“-u i prikazana je na slici 4.40.



Slika 4.39. Dio XML koda „task_form_fragment“-a i izgled zaslona

```
private fun initSpinners() {

    val repeatList = arrayOf("One time", "Daily", "Monthly", "Yearly")
    val spinnerAdapter = ArrayAdapter(requireContext(), android.R.layout.simple_spinner_item, repeatList)
    spinnerAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)

    task_form_spinner_repeat_options.adapter = spinnerAdapter
    task_form_spinner_repeat_options.dropDownVerticalOffset = 10

    task_form_spinner_repeat_options.onItemSelectedListener = object : AdapterView.OnItemSelectedListener {

        override fun onItemSelected(
            parent: AdapterView<*>?,
            view: View?,
            position: Int,
            id: Long
        ) {
            repeat = position
        }

        override fun onNothingSelected(parent: AdapterView<*>?) {
            repeat = 0
        }
    }

}
```

Slika 4.40. Metoda „initSpinners“ u „TaskFormFragment“-u

Fragment „TaskFormFragment“ također se koristi za uređivanje zadatka. Pritiskom na bilo koji zadatak na zaslonu s zadacima i događajima u blizini, kalendaru ili profilu, otvara se isti zaslon na kojem je moguće promijeniti sve postavke zadatka.

5. NAČIN KORIŠTENJA I ISPITIVANJE RADA APLIKACIJE

U ovom poglavlju biti će opisan način korištenja aplikacije, koraci koje korisnik prolazi u anketi i u korištenju same aplikacije i prikazane funkcionalnosti koje aplikacija ima.

5.1. Način korištenja aplikacije

Prvi korak korištenja aplikacije je sama instalacija. Nakon instalacije, korisniku se prikazuje zaslon za prijavu na kojem postoji opcija da se korisnik registrira. Pritiskom na gumb, otvara se zaslon za registriranje. Ako su svi uvjeti registriranja ispunjeni, kreira se novi korisnički račun i korisnik se upisuje u bazu podataka. Tim istim podacima korisnik je prijavljen u aplikaciju i nudi mu se opcija rješavanja ankete za preporuke. Ukoliko korisnik ne želi preporuke, pritisne na gumb za odbacivanje koji ga vodi direktno u aplikaciju. Ako korisnik želi riješiti anketu prvo mu se nude pitanja o njegovim hobijima. Nakon pitanja o hobijima korisnik dobiva popis kategorija iz kojih će mu biti postavljena pitanja te bira one koje ga zanimaju. S obzirom na izabrane kategorije, korisnik dobiva pitanja. Svi odgovori korisnika spremaju se u bazu podataka te će na osnovu tih odgovora biti izabrane preporuke aktivnosti i događaja za korisnika. Završetkom ankete otvara se glavni dio aplikacije.

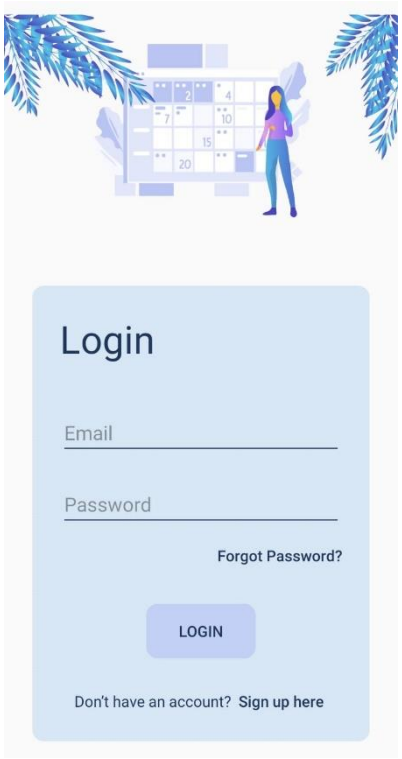
Glavni dio aplikacije sastoji se od tri dijela, zaslona sa zadacima korisnika i događajima u blizini, zaslona s kalendarom i zaslona profila korisnika. Na dnu svakog od tih zaslona nalazi se navigacija koja vodi s jednog na drugi zaslon. Prvi zaslon koji je otvoren je zaslon sa zadacima korisnika i događajima u blizini. Korisnik ima mogućnost vidjeti popis zadataka s obzirom na taj dan, taj tjedan ili taj mjesec. Pritiskom na događaj ili aktivnost, korisniku se otvara zaslon za njegovo uređivanje na kojemu može promijeniti ikonicu, boju, naslov i opis zadatka, želi li da to bude cjelodnevna aktivnosti, datum i koliko često želi da se odvija. Drugi zaslon, do kojeg se dolazi pritiskom na srednji gumb navigacije, je kalendar. Na zaslonu s kalendarom se nalazi, naravno, kalendar na kojemu su vidljive ikonice događaja tog dana. Ispod kalendara nalazi se popis zadataka tog dana koji se otvori kada se pritisne na određeni dan. Treći zaslon je zaslon profila na kojemu korisnik ima mogućnost promijeniti svoju profilnu fotografiju. Na tom zaslonu su također svi predloženi zadaci korisnika i popis svih zadataka koje su korisniku predložene i koje je sam sebi zadao. S profila pritiskom na gumb za postavke, otvara se zaslon s postavkama. Na zaslonu s postavkama korisnik ima mogućnost promijeniti svoje ime ili lozinku, odjaviti se i obrisati svoj korisnički račun.

S tri glavna zaslona moguće je doći na zaslon za stvaranje zadatka pritiskom na gumb u donjem desnom kutu. Zaslon za stvaranje zadatka sastoji se od istih mogućnosti kao i prethodno spomenuti zaslon za uređivanje zadatka.

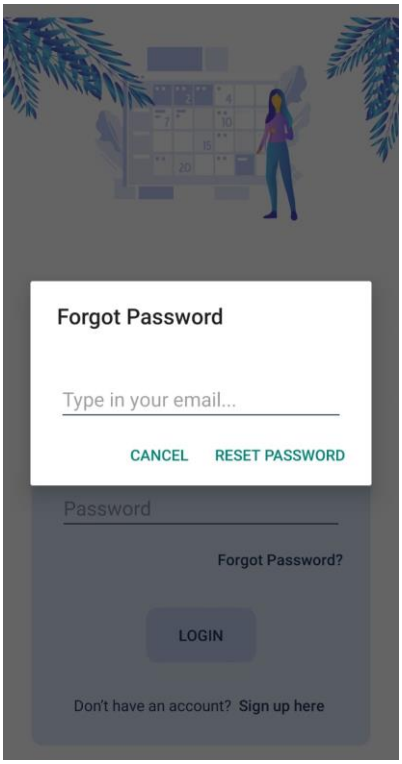
5.2. Prikaz ispitivanja rada aplikacije i postavke

5.2.1. Prikaz prijave korisnika

Pri prvom pokretanju aplikacije, otvara se zaslon prijave korisnika. Ako korisnik ima račun, u polja za unos podataka unosi svoju adresu e-pošte i lozinku i pritišće gumb „LOGIN“ čime je postupak prijave završen ukoliko su uneseni podaci ispravni. Sučelje za registriranje prikazano je na slici 5.1. U slučaju da je korisnik zaboravio svoju lozinku, pritiskom na gumb „Forgot password“, otvara se dijalog na kojem je polje za upis adrese e-pošte na koju korisnik želi da mu se pošalje hiperveza za promjenu lozinke. Dijalog je vidljiv na slici 5.2. Na zaslonu za prijavu se također nalazi gumb „Sign up here“ koji korisnik treba pritisnuti u slučaju da nema napravljen račun i želi se registrirati.



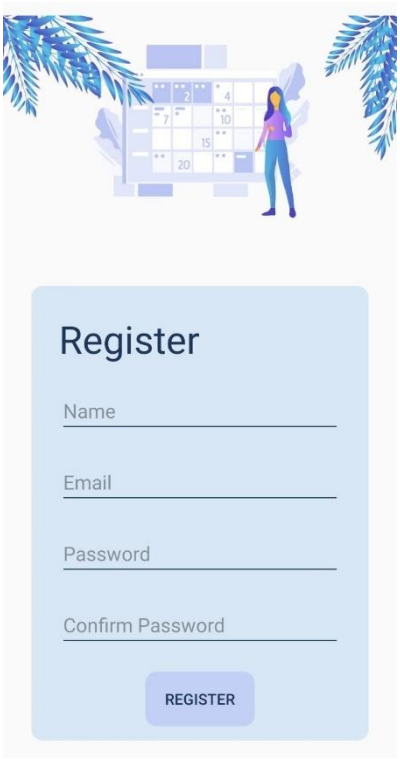
Slika 5.1. Sučelje prijave



Slika 5.2. Dijalog za postavljanje nove lozinke

5.2.2. Prikaz registriranja korisnika

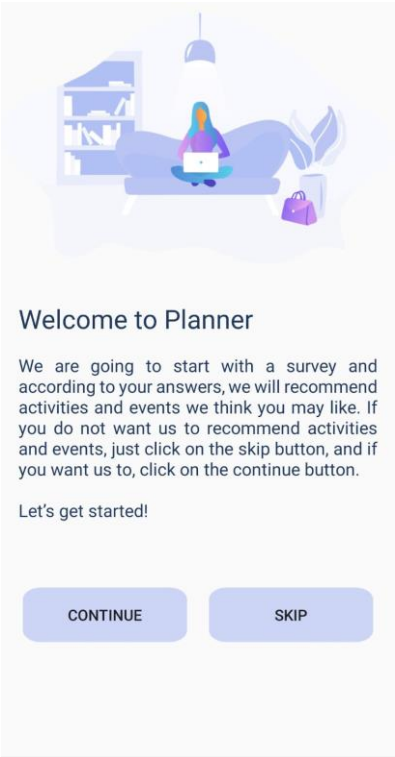
Nakon pritiska na gumb „Sign up here“ na zaslonu prijave, otvara se zaslon za registriranje. Kako bi se korisnik mogao registrirati, potrebno je unijeti podatke u polja za unos. Od korisnika se zahtjeva unos imena, adrese e-pošte, lozinke te ponovan unos lozinke kako bi se izbjegle slučajne pogreške pri unosu. Nakon što je korisnik ispravno upisao sve potrebne podatke, pritišće gumb „REGISTER“ i time je registriranje korisnika završeno te se otvara zaslon prijave i automatski unose podaci upisani na zaslonu za registriranje. Ako je korisnik prijavljen prvi put, što će biti uvijek u slučaju registriranja, otvara se sučelje ankete. Na slici 5.3 prikazano je sučelje za registriranje.



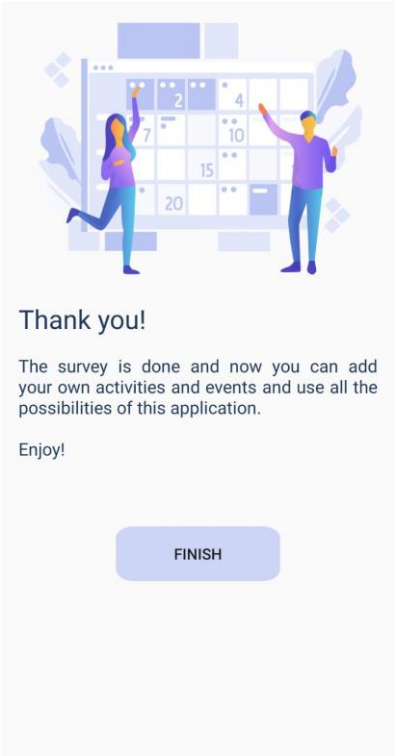
Slika 5.3. Sučelje za registriranje

5.2.3. Prikaz rješavanja ankete

Pri prvoj prijavi korisnika otvara se anketa čiji odgovori služe za davanje preporuka korisnicima. Prvi zaslon ankete, odnosno sučelja uvoda u anketu, vidljiv je na slici 5.4. Na tom zaslonu korisnik vidi kratki opis ankete i upute koji gumb da pritisne ovisno o tome što želi napraviti. Desni gumb na zaslonu služi za preskakanje ankete, pritisak na gumb „SKIP“ otvara posljednji zaslon ankete, vidljiv na slici 5.5. Pritiskom na gumb „CONTINUE“, korisnik pristaje na rješavanje ankete i otvara mu se zaslon s pitanjem o hobijima sa slike 5.6.

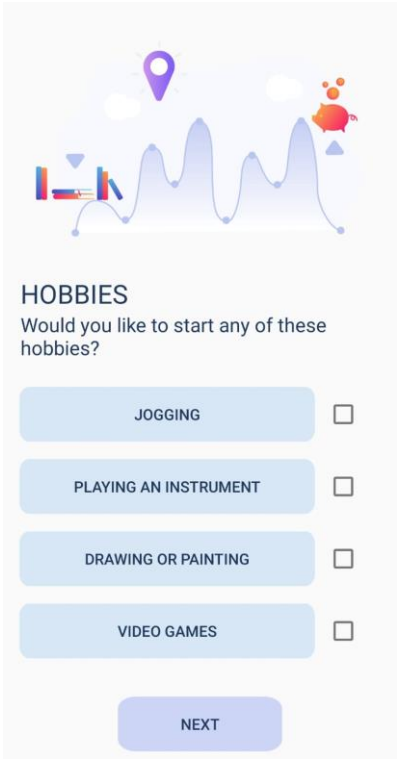


Slika 5.4. Sučelje uvoda u anketu



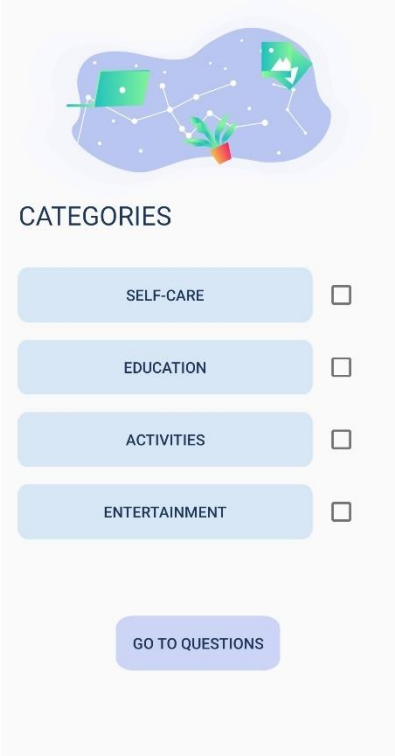
Slika 5.5. Sučelje posljednjeg zaslona ankete

Na zaslonu sa slike 5.6 postavljeno je pitanje koji od ponuđenih hobija interesiraju korisnika, točnije želi li se početi baviti nekim od tih hobija. Korisnik označava što ga zanima pritiskom na kućice pored odgovora i kada je zadovoljan odgovorom, pritišće na gumb „NEXT“ koji otvara zaslon s kategorijama sa slike 5.7.

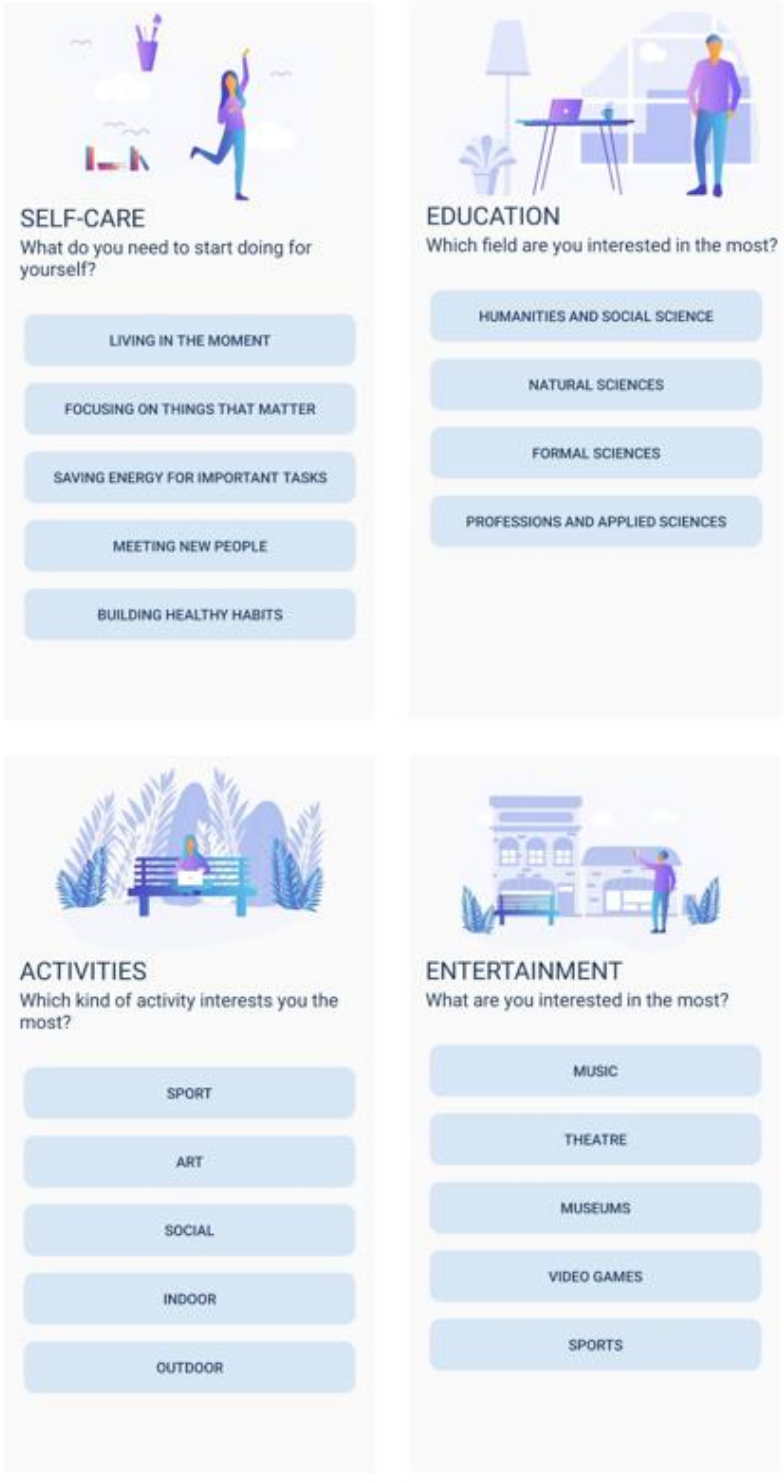


Slika 5.6. Sučelje pitanja o hobijima

Na zaslonu s kategorijama korisnik bira barem jednu kategoriju koja ga zanima i iz koje želi postavljena pitanja. Kategorije koje ga zanimaju označava pritiskom na kvadratić pored imena kategorije. S obzirom na izabrane kategorije, korisniku će se otvoriti pitanja sa slike 5.8. Nakon što korisnik odgovori na sva postavljena pitanja, otvara mu se zaslon sa slike 5.5 i time završava rješavanje ankete.



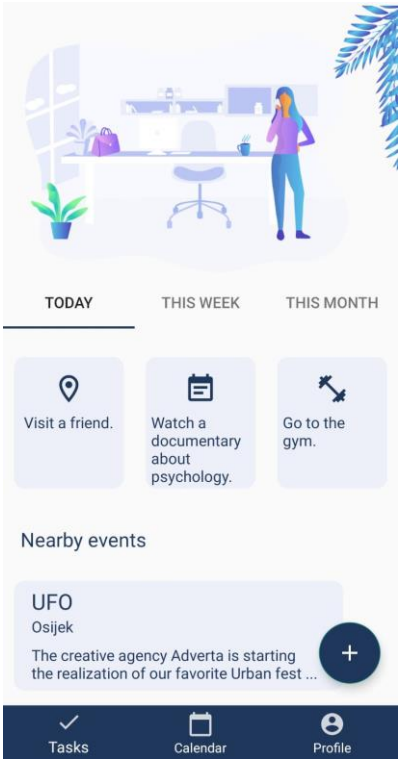
Slika 5.7. Sučelje kategorija



Slika 5.8. Sučelja pitanja po kategorijama

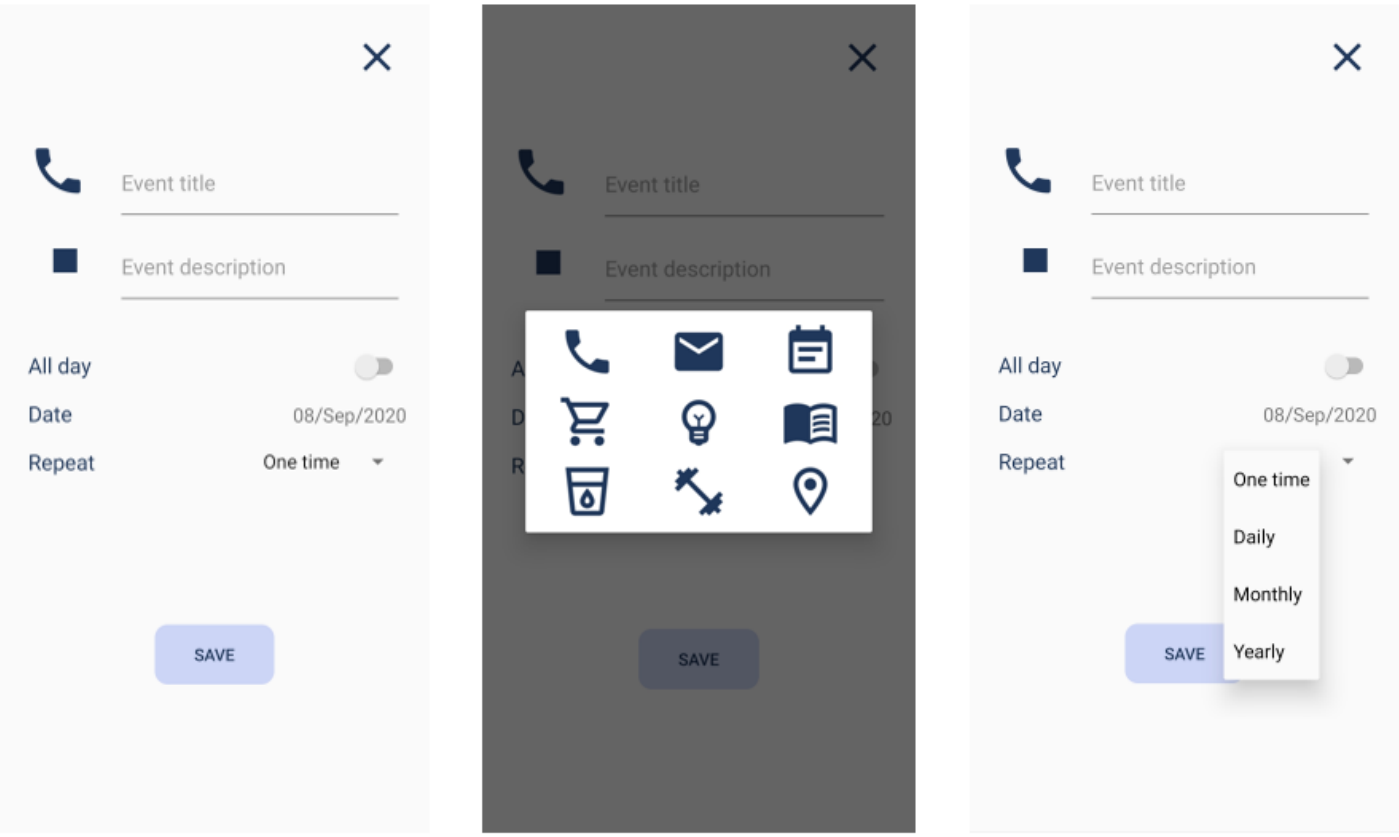
5.2.4. Prikaz zaslona sa zadacima i događajima u blizini

Zaslon koji se prvi otvori nakon ankete je zaslon sa zadacima i događajima u blizini. Zadaci i događaji koji su vidljivi na ovom zaslonu su predloženi korisniku s obzirom na riješenu anketu. U slučaju da korisnik nije htio riješiti anketu, sve ispod prozora „TODAY“ bilo bi prazno dok korisnik ne bi sam dodao neku aktivnost. Sučelje zaslona sa zadacima i događajima u blizini vidljivo je na slici 5.9. Pritiskom na gumb sa znakom plusa u njegovoj sredini koji se nalazi u donjem desnom kutu aplikacije, otvara se sučelje za stvaranje novog zadatka prikazano na slici 5.10 zajedno s opcijama stvaranja. Pritiskom na srednji gumb u navigaciji na dnu zaslona, otvara se zaslon s kalendarom opisan u sljedećem poglavlju.



Slika 5.9. Sučelje zaslona sa zadacima i događajima u blizini

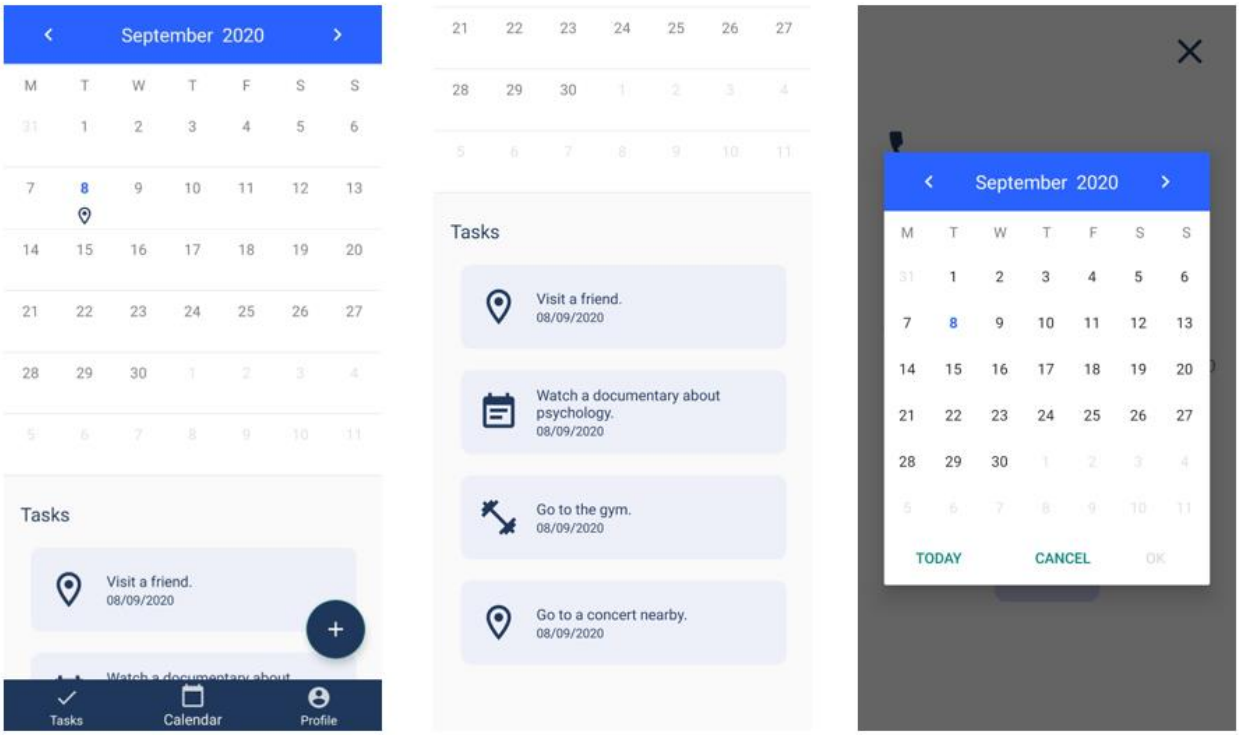
Na zaslonu za dodavanje zadatka, korisnik može izabrati ikonicu zadatka koju želi te boju zadatka. U polja za unos treba unijeti naslov zadatka i njegov opis. Ispod polja za unos korisnik bira želi li da taj zadatak bude cjelodnevan događaj, koji je datum kada se on odvija i želi li korisnik da to bude događaj koji se odvija samo na taj dan, svaki dan, svaki mjesec ili svake godine.



Slika 5.10. Sučelje zaslona za stvaranje zadatka

5.2.5. Prikaz zaslona s kalendarom

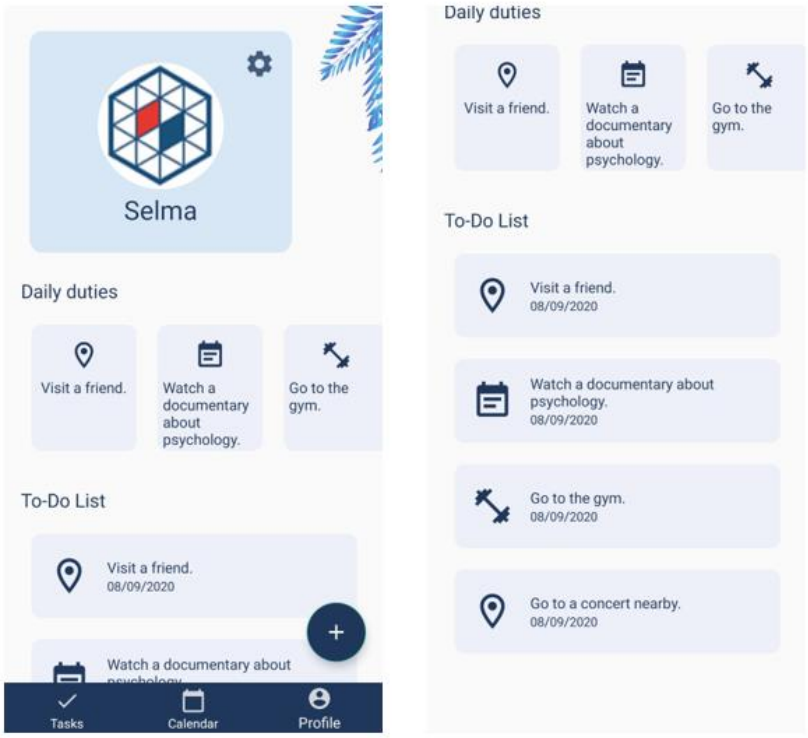
Na slici 5.1 prikazano je sučelje kalendara. Pritiskom na bilo koji dan u kalendaru, ispod njega se otvara popis događaja i aktivnosti koje se odvijaju taj dan. Kada korisnik pritisne na neki od događaja ili aktivnosti, otvara se zaslon za stvaranje i uređivanje tog zadatka koji je prikazan u poglavlju 5.2.4. na slici 5.10. Pritiskom na prvi gumb u navigaciji, korisnik se vraća na prethodno opisani zaslon sa zadacima i događajima u blizini, a pritiskom na treći gumb otvara se sučelje profila koje će biti prikazano i opisano u sljedećem poglavlju.



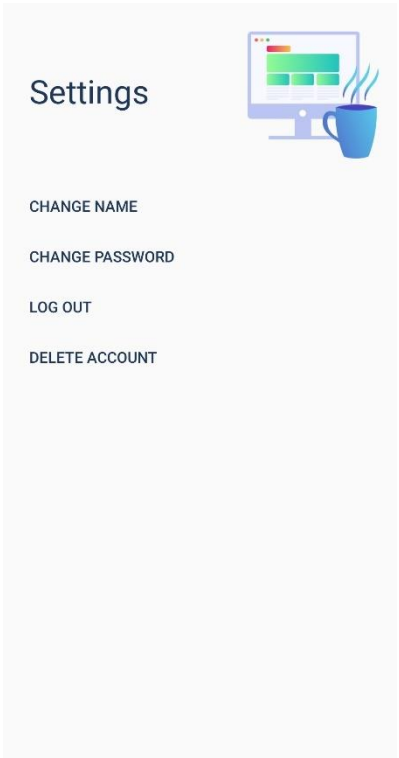
Slika 5.11. Sučelje kalendara

5.2.6. Prikaz korisničkog profila i postavki

Na korisničkom profilu korisnik pritiskom na fotografiju iznad imena ulazi u svoju galeriju i bira profilnu fotografiju koju želi postaviti u aplikaciju. Pritiskom na neku od aktivnosti koje su prikazane na profilu ili gumb u donjem desnom kutu s ikonicom plusa u sredini, otvara se zaslon za stvaranje i uređivanje aktivnosti ili događaja koji je opisan u poglavlju 5.2.4. i prikazan na slici 5.10. Sučelje korisničkog profila prikazano je na slici 5.12. Pritiskom na gumb za postavke otvara se sučelje postavki korisničkog profila prikazano na slici 5.13.

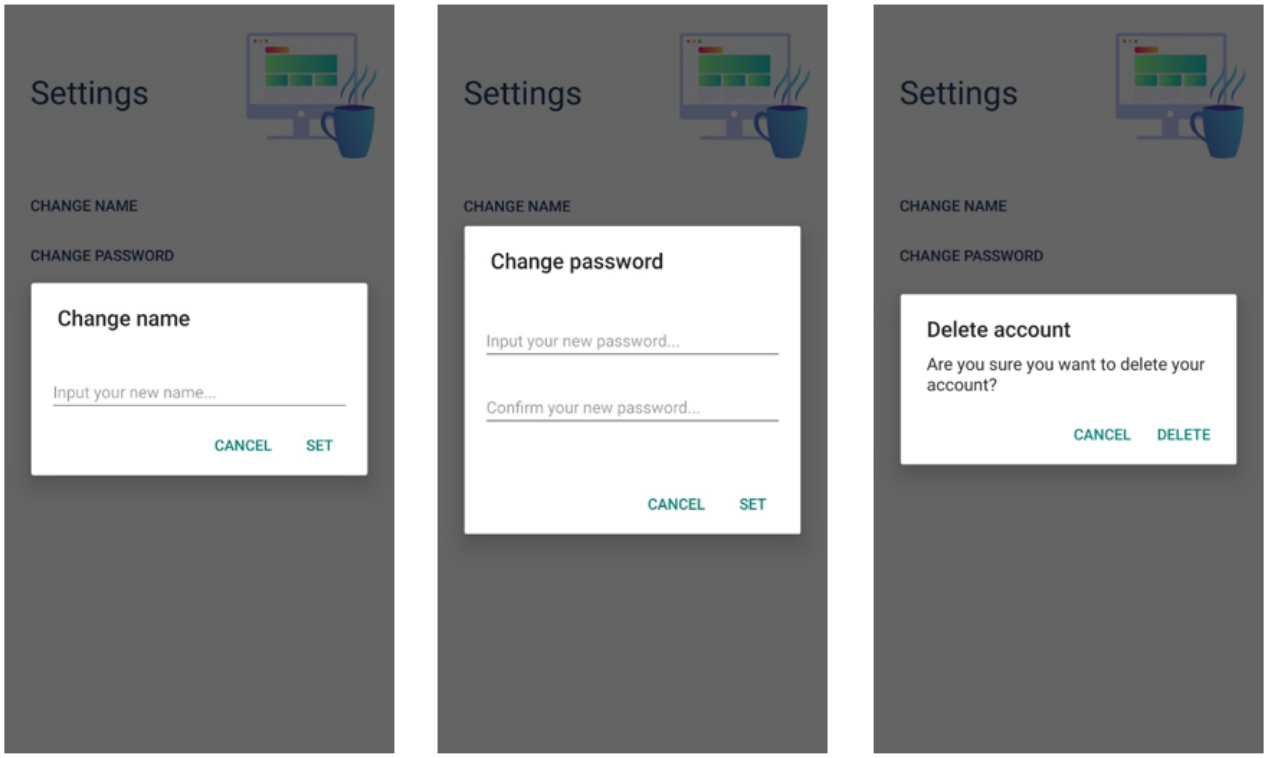


Slika 5.12. Sučelje profila



Slika 5.13. Sučelje postavki korisničkog profila

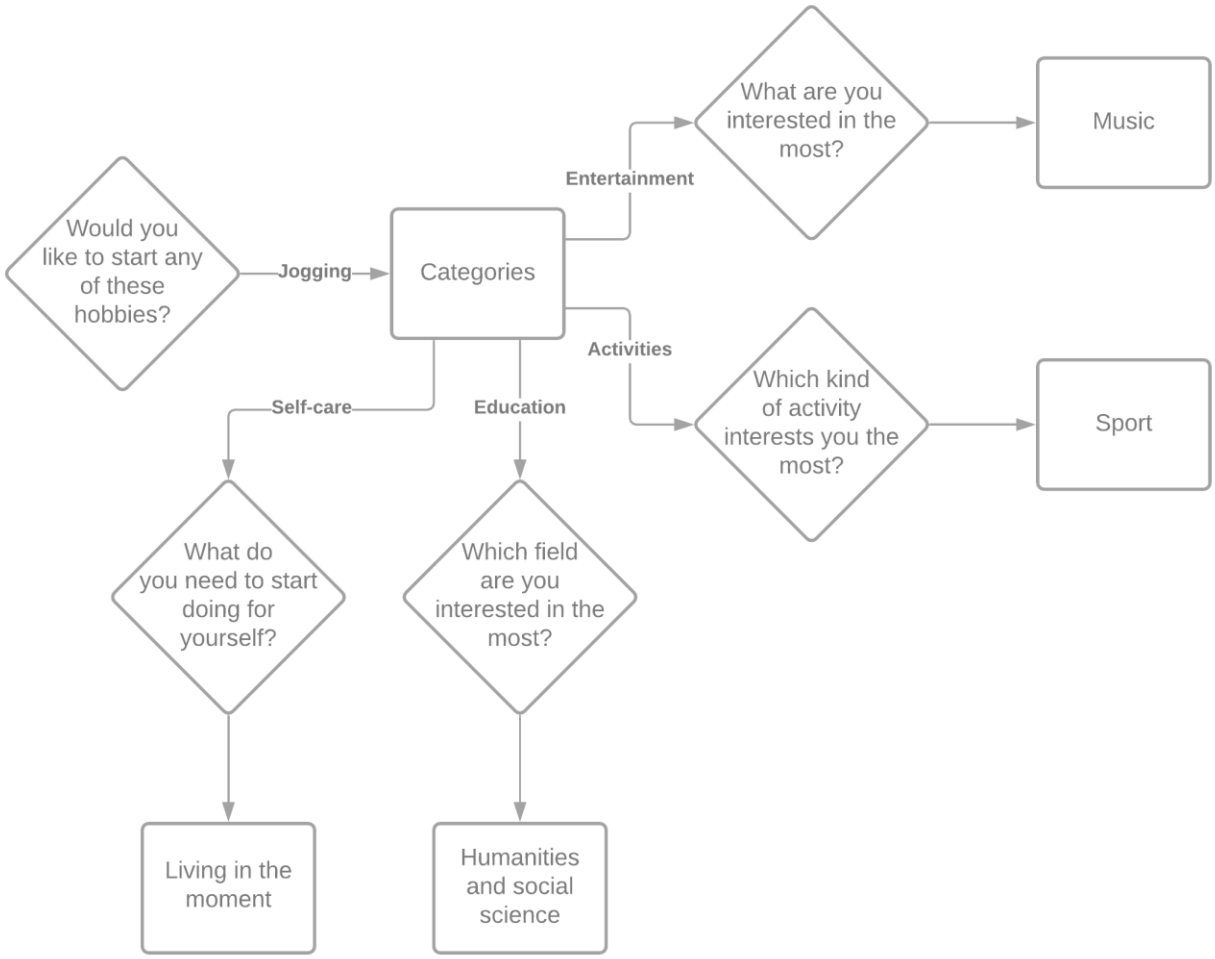
Zaslon s postavkama korisničkog profila ima četiri mogućnosti. Pritiskom na gumb „CHANGE NAME“, otvara se dijalog i pritiskom na „SET“ korisnik mijenja ime na svom profilu. Gumb ispod njega je „CHANGE PASSWORD“ i služi za promjenu lozinke korisnika. Kada korisnik pritisne taj gumb, otvara se dijalog s poljima za unos lozinke i pritiskom na gumb „SET“, nova je lozinka postavljena, ali pod uvjetom da je duža od osam znakova. Sljedeći gumb je „LOG OUT“ koji odjavljuje korisnika kada pritisne na njega i otvara zaslon za prijavu. Posljednji gumb imena „DELETE ACCOUNT“ koristi se za brisanje korisničkog profila te se pritiskom na „DELETE“, korisnički račun koji je trenutno prijavljen, briše iz baze podataka. Dijalozi za promjenu imena, lozinke i brisanje korisničkog računa prikazani su na slici 5.14.



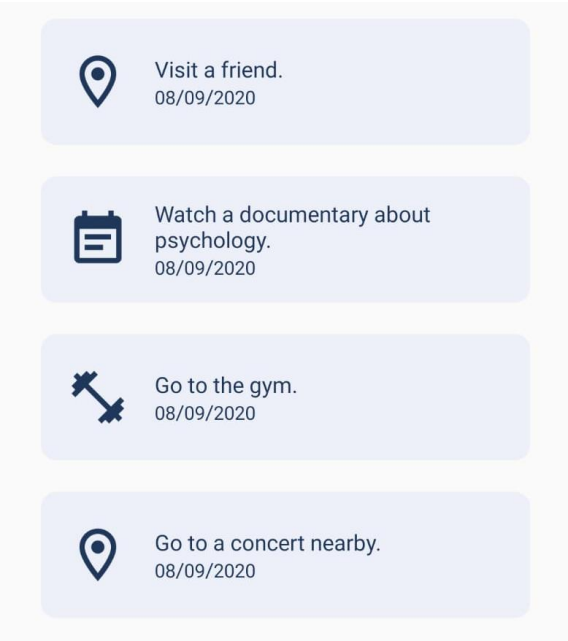
Slika 5.14. Dijalozi postavki korisničkog profila

5.3. Slučaj ispitivanja aplikacije

Prikazan je jedan slučaj predlaganja aktivnosti i događaja korisniku s obzirom na anketu. U slučaju da korisnik prihvati rješavanje ankete i odgovori na postavljena pitanja na način koji je prikazan na slici 5.15, na slici 5.16 su prikazani prijedlozi koje bi korisnik dobio.



Slika 5.15. Odgovori korisnika u anketi



Slika 5.16. Aktivnosti predložene korisniku

Na slici 5.16 prikazane su aktivnosti predložene korisniku. Iz tablica iz poglavlja 3.4.4. vidljivo je kako je korisniku nasumično predložena po jedna aktivnost iz svake izabrane kategorije. Kada korisnik pritisne na te događaje, ima mogućnost urediti ih kako on želi.

6. ZAKLJUČAK

U ovom radu razvijena je mobilna Android aplikacija za planiranje i praćenje obveznih i slobodnih aktivnosti korisnika. Ova aplikacija razvijena je s ciljem lakšeg planiranja vremena i obveza i u svrhu poticanja korisnika na korištenje svog slobodnog vremena na nove načine. Aplikacija omogućuje korisniku jednostavno registriranje i prijavu, rješavanje ankete, dodavanje zadataka i događaja u raspored, uređivanje tih zadataka, njihovo upisivanje u kalendar i uređivanje korisničkog profila. Korištenje ove aplikacije pomaže korisniku u lakšem planiranju i praćenje svojih obaveza i željenih aktivnosti te je prikladna za svakoga tko želi lakše organizirati svaki dan.

Aplikacija je izrađena u razvojnoj okolini Android Studio pri čemu je korišten programski jezik Kotlin, a kao jezikom označavanja podataka XML. Ispitana je funkcionalnost aplikacije i prikazana njena ispravnost na temelju rješavanja ankete i dobivenih aktivnosti koje su korisniku preporučene. Opisane su i prikazane vrste preporuka koje korisnik može dobiti rješavajući anketu. Aplikacija će moći biti poboljšana kada se bude odvijalo više stvarnih događaja što će omogućiti njihov unos s obzirom na lokaciju korisnika.

LITERATURA

[1] N. Osmanagić Bedenik, Operativno planiranje. Zagreb: Školska knjiga, 2002.

[2] P. Gajewska, K. Piskrzynska, *Leisure Time Management,* Forum Scientiae Oeconomia, Vol. 5, No.1, str. 57-69, 2017.

[3] Google Calendar, https://www.google.com/calendar/about/, pristupljeno 06.07.2020.

[4] Google Tasks, https://play.google.com/store/apps/details?id=com.google.android.apps.tasks&hl=hr, pristupljeno 06.07.2020.

[5] Microsoft To Do, https://todo.microsoft.com/tasks/, pristupljeno 06.07.2020.

[6] K. S. Targiel, M. Nowak, T. Trzaskalik, Scheduling non-critical activities using multicriteria approach. Katowice: Department of Operations Research, 2018.

[7] Department for Communities and Local Government, Multi-criteria analysis: a manual. London, 2009.

[8] Figma, https://www.figma.com/, pristupljeno 06.07.2020.

[9] Android, https://developer.android.com/guide/platform, pristupljeno 06.07.2020.

[10] Platform Architecture, https://developer.android.com/guide/platform, pristupljeno 06.07.2020.

[11] Android Studio, https://developer.android.com/studio, pristupljeno 06.07.2020.

[12] Kotlin, https://kotlinlang.org/, pristupljeno 06.07.2020.

[13] XML, https://hr.wikipedia.org/wiki/XML, pristupljeno 06.07.2020.

[14] Firebase, https://firebase.google.com/, pristupljeno 06.07.2020.

[15] Authenticating Users on App Engine Using Firebase, https://cloud.google.com/appengine/docs/standard/python/authenticating-users-firebase-appengine, pristupljeno 06.07.2020.

[16] The Ultimate Guide to Mobile Application Architecture, App Velocity, August 2019, https://www.appvelocity.ca/blog/guide-mobile-application-architecture, pristupljeno 06.07.2020.

[17] Android Architecture Patterns Part 3: Model-View-ViewModel, Florina Muntensecu https://medium.com/upday-devs/android-architecture-patterns-part-3-model-view-viewmodel-e7eeee76b73b, pristupljeno 06.07.2020.

[18] ReactiveX/RxJava, https://github.com/ReactiveX/RxJava, pristupljeno 06.07.2020.

[19] R.C. Martin, Clean Code. Stoughton, Massachusetts: Pearson Education, Inc, 2008.

[20] MVVM Architecture App in Android, https://androidwave.com/mvvm-architecture-app-in-android/, pristupljeno 06.07.2020.

[21] Flow, https://developer.android.com/reference/java/util/concurrent/Flow, pristupljeno 02.09.2020.

[22] Slide between fragments using ViewPager, https://developer.android.com/training/animation/screen-slide, pristupljeno 02.09.2020.

[23] TabLayout, https://developer.android.com/reference/com/google/android/material/tabs/TabLayout, pristupljeno 02.09.2020.

[24] MaterialCalendarView – customizable calendar widget for Android, https://applandeo.com/blog/material-calendar-view-customized-calendar-widget-android/, pristupljeno 02.09.2020.

[25] About Glide, https://bumptech.github.io/glide/, pristupljeno 02.09.2020.

[26] Add a Floating Action Button, https://developer.android.com/guide/topics/ui/floating-action-button, pristupljeno 02.09.2020.

[27] Switch, https://developer.android.com/reference/android/widget/Switch, pristupljeno 02.09.2020.

[28] Spinner, https://developer.android.com/guide/topics/ui/controls/spinner, pristupljeno 02.09.2020.

POPIS SLIKA

POPIS TABLICA

SAŽETAK

U ovom radu razvijena je mobilna Android aplikacija za planiranje i praćenje obveznih i slobodnih aktivnosti korisnika zbog lakšeg planiranja vremena korisnika. U radu je opisan model provedbe aktivnosti te postupak planiranja i stvaranja preporuka. Navedena su i prikazana postojeća rješenja za planiranje aktivnosti i pojašnjene sličnosti i razlike s ovom aplikacijom. Također, predložena je opisana arhitektura aplikacije i potrebne programske tehnologije. Navedene su i objašnjene sve funkcionalnosti aplikacije, od registriranja do korištenja svih mogućnosti planiranja aktivnosti. Korisnik nakon registriranja, prijave i ispunjenja ankete, dobiva određeni broj preporuka aktivnosti. U aplikaciji korisnik može mijenjati i dodavati događaje te podešavati postavke na način na koji mu odgovara. Ispitan je način rada aplikacije i prikazane sve funkcionalnosti. Analiza rada aplikacije prikazala je način na koji su dobiveni potrebni kriteriji za biranje i dodavanje preporuka u korisnikov raspored, a ispitni slučajevi potvrdili su ispravnost funkcionalnosti aplikacije.

Ključne riječi: aktivnosti, Android, mobilna aplikacija, planiranje.

ABSTRACT

In this paper, a mobile Android application is developed for planning and monitoring user obligations and leisure time for easier time planning. The paper describes the model of implementation of activities and the process of planning and making recommendations. Existing solutions for planning activities are listed and presented, and similarities and differences with this application are explained. Also, the described application architecture and required software technologies are proposed. All functionalities of the application are listed and explained, from user registration to the use of all possibilities for planning activities. After registering, signing in and completing the survey, the user receives a number of activity recommendations. In the application, the user can change and add events and adjust settings to suit his needs. The mode of operation of the application was examined and all functionalities were shown. The analysis of the application showed the way in which the necessary criteria for selecting and adding recommendations to the user's schedule is obtained, and test cases confirmed the accuracy of the application's functionality.

Keywords: activities, Android, mobile application, planning.

ŽIVOTOPIS

Selma Tojčić rođena je 11. veljače 1999. godine u Osijeku. Pohađala je Osnovnu školu Vladimir Nazor u Čepinu. Upisuje III. Gimnaziju Osijek 2013. godine te ju završava 2017. godine. Iste godine upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, smjer preddiplomski studij računarstvo.

PRILOZI (na CD-u)

1. Završni rad „Mobilna android aplikacija za planiranje i praćenje obveznih i slobodnih aktivnosti korisnika“ u *.docx* formatu
2. Završni rad „Mobilna android aplikacija za planiranje i praćenje obveznih i slobodnih aktivnosti korisnika“ u *.pdf* formatu
3. Programski kod mobilne aplikacije