Virtualni osobni trener u obliku android aplikacije

Knežević, Dino

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:795102

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-30**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 23.09.2022.

Ime i prezime studenta:	Dino Knežević
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R 4365, 22.07.2019.
Turnitin podudaranje [%]:	11

Ovom izjavom izjavljujem da je rad pod nazivom**: Virtualni osobni trener u obliku android aplikacije**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Irena Galić

i sumentora Marin Benčević, mag. ing. comp.

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:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

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

Osijek, 19.09.2022.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Dino Knežević
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R 4365, 22.07.2019.
OIB Pristupnika:	29049685880
Mentor:	Izv. prof. dr. sc. Irena Galić
Sumentor:	Marin Benčević, mag. ing. comp.
Sumentor iz tvrtke:	
Naslov završnog rada:	Virtualni osobni trener u obliku android aplikacije
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	Napraviti Android aplikaciju koja će na osnovi povijesnih statistika korisnika predložiti optimalan plan vježbi za sljedeći trening. Dinamički prilagoditi mišićne skupine vježbi, težinu i broj ponavljanja i sl. Pratiti i spremati statistike svakog treninga za buduće dinamično predlaganje vježbi. Opisati način prikupljanja i obrade podataka. Istražiti i opisati kakvi algoritmi predlaganja se koriste za slične probleme u znanosti i praksi. Opisati algoritam za predlaganje optimalnog plana vježbi. Usporediti s postojećim rješenjima i opisati
Prijedlog ocjene 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: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	19.09.2022.
Datum potvrde ocjene od strane Odbora:	21.09.2022.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

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

Sveučilišni studij računarstva

VIRTUALNI OSOBNI TRENER U OBLIKU ANDROID APLIKACIJE

Završni rad

Dino Knežević

Osijek, 2022.

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. VAŽNOST SVAKODNEVNE TJELESNE AKTIVNOSTI 3

2.1. Pojam tjelesne aktivnosti 3

2.2. Važnost tjelesne aktivnosti 3

2.3. Nedostatak tjelesne aktivnosti i posljedice 4

2.4. Postojeća rješenja za praćenje i preporučivanje tjelesne aktivnosti 4

2.5. Idejno rješenje vlastite aplikacije za praćenje i preporučivanje tjelesne aktivnosti 7

3. MODEL ANDROID APLIKACIJE ZA PRAĆENJE I PREPORUČIVANJE TJELOVJEŽBE 8

3.1. Korisnički zahtjevi na mobilnu aplikaciju za praćenje i preporučivanje tjelovježbe 8

3.2. Funkcionalni zahtjevi mobilne aplikacije 8

3.3. Sustavski zahtjevi mobilne aplikacije 11

3.4. GRAĐA MOBILNE APLIKACIJE ..... Error! Bookmark not defined.

4. PROGRAMSKO RJEŠENJE APLIKACIJE ZA PRAĆENJE I PREPORUČIVANJE TJELOVJEŽBE 13

4.1. Opis potrebnih alata i tehnologije 13

4.2. Prikaz programskog rješenja osnovnih funkcionalnosti aplikacije 16

4.3. Generiranje plana tjelovježbe 29

4.4. Prikaz i spremanje promjena plana trenutne tjelovježbe 36

4.5. Postojeći algoritmi za predlaganje tjelovježbe 39

5. Primjer korištenja aplikacije 40

5.1. Testiranje rada aplikacije 40

6. Zaključak 45

LITERATURA 46

SAŽETAK **48**

ABSTRACT **49**

ŽIVOTOPIS **50**

1. UVOD

Tjelesna, odnosno fizička aktivnost je sastavni dio života, ali i način održavanja zdravog života. Tjelesna aktivnost su svakodnevne radnje poput šetnje, obavljanje posla, vožnja bicikla i druge radnje. Napretkom tehnologije, ljudi su sve više skloniji sjedilačkom načinu života jer im je mnogo lakše i jednostavnije naručiti hranu iz udobnosti svog doma ili održavati kontakt s poznanicima putem društvenih mreža i slično. Najefikasniji način za podizanje aktivnosti je tjelovježba ili bavljenje sportom. Ona pomaže povećati ili održavati razinu zdravlja tako da smanjuje rizike mnogih bolesti poput kardiovaskularnih, visokog tlaka, otklanja bolne kretnje tako da se jačaju mišići potrebni za njeno obavljanje, također ima i pozitivan psihički učinak [1].

Tema ovog završnog rada je izrada mobilne Android aplikacije koja će korisniku olakšati praćenje aktivnosti, ali i preporučivati istu. Aplikacija posjeduje registraciju za svakog korisnika, potrebni su joj određeni parametri za generiranje preporuka koji se unose pri registraciji. Međutim aplikacija treba dinamički predlagati vježbe što znači da koristi povijest aktivnosti kako bi određivala progresiju i izmjenjivala vježbe. U praktičnom dijelu rada ostvareno je programsko rješenje aplikacije razvijene za Android platformu, u besplatnom obliku namijenjeno svima koji žele održati ili povećati svoju tjelesnu aktivnost.

U drugom poglavlju objašnjava se važnost svakodnevne tjelesne aktivnosti te posljedice neodržavanja iste. Navedeno je vlastito idejno rješenje i nekoliko postojećih rješenja za praćenje i preporučivanje tjelesne aktivnosti. Treće poglavlje opisuje model mobilne Android aplikacije za praćenje i preporučivanje tjelesne aktivnosti. Ono sadrži opisani postupak preporuke, opis parametara i korisničkog sučelja. Četvrto poglavlje opisuje praktično, odnosno programsko rješenje. U njemu su opisani korišteni alati i tehnologije potrebne za izradu mobilne Android aplikacije. Programski kod je objašnjen po svojim dijelovima, uz pripadajuće umetke koda. Peto poglavlje prikazuje rad aplikacije, te je objašnjeno njeno korištenje i ispitivanje njene ispravnosti.

1.1. Zadatak završnog rada

U završnom radu je potrebno objasniti važnost tjelesne aktivnosti, njene prednosti i posljedice njenog neodržavanja. Poboljšanje će se postići na način povećanja mišićne mase, izvodeći vježbe odnosno kretnje koje koristimo u svakodnevnom životu. Također će imati utjecaj na kardiovaskularno zdravlje. Kako bi se opisano lakše ostvarilo, potrebno je izraditi mobilnu Android aplikaciju koja će prikupiti osnovne parametre potrebne za preporuku tjelovježbe, te koja će uvelike olakšati praćenje tjelovježbe uz predlaganje vježbi na osnovu povijesnih statistika

korisnika. Potrebno je dinamički prilagoditi težinu i broj ponavljanja svake vježbe kako bi se osigurao konstantan napredak. Također je potrebno istražiti postojeća rješenja, usporediti ostvareno rješenje s njima, te opisati njegove prednosti i nedostatke.

2. VAŽNOST SVAKODNEVNE TJELESNE AKTIVNOSTI

U ovom poglavlju opisan je pojam tjelesne aktivnosti, njegova važnost u svakodnevnom ljudskom životu te posljedice izostanka tjelesne aktivnosti. Navedeno je nekoliko postojećih rješenja za praćenje i preporučivanje tjelesne aktivnosti koja su ostvarena putem mobilne Android platforme.

2.1. Pojam tjelesne aktivnosti

Svjetska zdravstvena organizacija (SZO) je definirala tjelesnu aktivnost kao [2] „svaki pokret tijela koji se izvodi aktivacijom skeletnih mišića i koji zahtijeva potrošnju energije“, odnosno svako kretanje tijela koje zahtijeva rad mišića uz potrošnju energije. Mnogo bolesti popot kardiovaskularnih se mogu uvelike spriječiti, odnosno može se umanjiti njihov rizik putem tjelovježbe. Tjelovježbom koja se izvodi minimalno dva puta tjedno, koja služi za jačanje mičića, čovjek je osigurao sve pozitivne strane tjelesne aktivnosti.

Osim fizičkih dobrobiti, tjelesna aktivnost donosi i pozitivne učinke na mentalno zdravlje. Uzrokuje i promjenu raspoloženja, prema [3], tjelesna aktivnost značajno smanjuje simptome depresije.

2.2. Važnost tjelesne aktivnosti

Prema [4], u svijetu su glavni problem u zdravstvu kronične bolesti. Izazov s kojim se zdravstvo suočava je loše poznavanje prednosti i nedostataka o zdravim navikama koje uključuju tjelesnu aktivnost i pravilnu prehranu. Redovita tjelesna aktivnost će pomoći smanjiti mnoge rizike. Osim zdravstvenih učinaka, redovitom tjelesnom aktivnosti, pojedinac poprima bolji fizički izgled, što ga dodatno motivira da nastavi provoditi tjelesnu aktivnost. Prema Bergeru [5], uzimajući u obzir mentalno, odnosno psihičko stanje čovjeka, tjelovježba se može odabirati prema sljedećim kriterijima:

- Nekompetitivne aktivnosti, jer kompetitivne aktivnosti mogu imati negativan učinak zbog mogućnosti stvaranja dodatnog stresa na psihu. One omogućuju čovjeku da se fokusira samo na sebe i da mu je jedina referenca on sam, tako će postepeno vidjeti rezultate i napredovati.
- Zatvorene aktivnosti poput vježbanja u teretani ili nekog borilačkog sporta. One će čovjeku osigurati kontinuitet gdje može imati svoj plan tjelovježbe i prilagođeni ritam tjelovježbe.

Navedeni pozitivni fizički i psihički učinci se mogu postići korištenjem teretane koja odgovara Bergerovim kriterijima, i fizičkim zahtjevima pojedinca.

2.3. Nedostatak tjelesne aktivnosti i posljedice

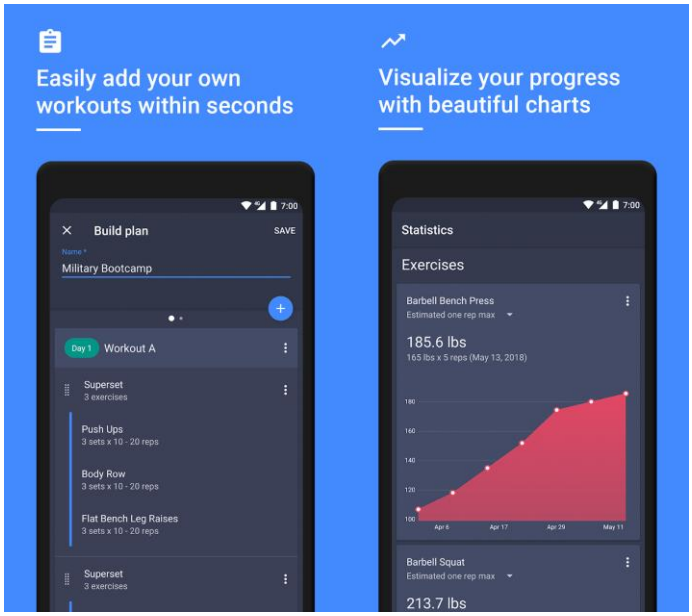
Iako se Republika Hrvatska trudi upozoravati i učiti svoje građane o prednostima ali i nedostatcima tjelesne aktivnosti, većina ljudi nije aktivna [6]. Zdrave navike se trebaju podučavati u ranoj životnoj dobi, te je veliki problem što više od polovice mladih i djece imaju aktivnost manju od poražavajućih 40%. Nedostatak tjelesne aktivnosti je svrstan na čak četvrto mjesto uzroka smrtnosti na svjetskoj razini. Kardiovaskularne bolesti imaju vodeći udio u ukupnoj smrtnosti koja je uzrokovana nedostatkom tjelesne aktivnosti [7].

2.4. Postojeća rješenja za praćenje i preporučivanje tjelesne aktivnosti

Na tržištu postoji mnogo rješenja za praćenje i preporučivanje tjelesne aktivnosti u raznim tehnologijama. Najpopularnija rješenja su razvijena u mobilnim tehnologijama jer je broj korisnika mobilnih uređaja jako velik. U nastavku će biti prikazana tri rješenja koja koriste Android platformu.

2.4.1. Gym Workout Planner & Tracker

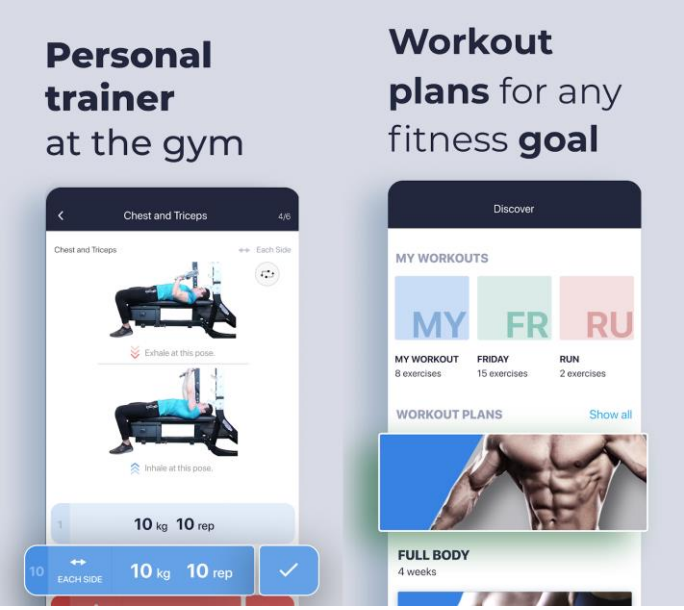
Aplikacija [8] koja omogućuje planiranje i praćenje vlastitog plana tjelovježbe. Primarna svrha je za tjelovježbu u teretani s utezima i spravama, ali moguće je kreirati i plan bez utega i sprava. Također sadrži nekoliko unaprijed osmišljenih planova koje mogu koristiti početnici ali i napredni vježbači. Korisna komponenta aplikacije je statistika gdje korisnik može vidjeti svoj napredak u vježbama, svojoj kilaži te promjene u dimenzijama svoga tijela. Uz sliku vježbe postoji i gumb „notes“ koji sadrži kratki opis izvođenja vježbe. Korisničko sučelje je vrlo jednostavno i intuitivno.



Sl. 2.1. Sučelje aplikacije Gym Workout Planner & Tracker

2.4.2. Gym Workout & Personal Trainer by Balanced

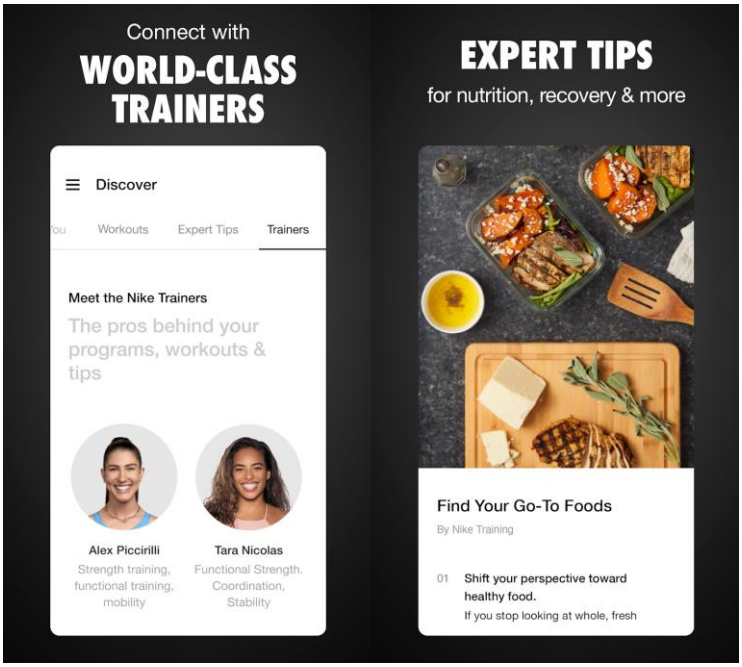
Virtualni osobni trener [9], omogućuje praćenje i planiranje vlastitog plana vježbanja, uz mogućnost odabira virtualnog trenera. Aplikacija nudi programe prema motivaciji korisnika koja može biti gubitak masti ili dobitak kilaže. Namijenjena je za polaznike teretane neovisno o njihovom vježbačkom iskustvu. Također je moguće složiti plan tjelovježbe prema odabranim mišićnim skupinama. Svaka vježba je detaljno objašnjena putem videozapisa i slika uz tekstualni opis. Na slici je prikazano korisničko sučelje koje ima mogućnost mijenjanja izgleda iz tamnog u svijetlo i obrnuto.



Sl. 2.2. Sučelje aplikacije Gym Workout Planner & Tracker by Balanced

2.4.3. Nike Training Club: Fitness

Aplikacija [10], je vrlo detaljna s velikim brojem opcija te je namijenjena svim razinama vježbača. Potpuno je besplatna za korištenje, moguće je vježbati i bez korištenja teretane, te ima veliki broj trenera koji su na svjetskoj razini. Izbor planova je opširan i prilagođen potrebama korisnika. Aplikacija posjeduje veliki broj vježbi za svaku mišićnu skupinu. Također nudi savjete za prehranu, uz preporuke recepata. Korisnik može kreirati svoj plan treninga uz pomoć Nike trenera. Postoje planovi treninga i za kardiovaskularno zdravlje te za mobilnost i elastičnost. Ono po čemu se razlikuje od drugih aplikacija je mogućnost treninga preko videoprijenosa uživo gdje korisnik prati trenera. Aplikacija nudi sinkronizaciju s Google Fit, kako bi korisnik mogao pratiti parametre poput otkucaja srca. Na slici je prikazano korisničko sučelje aplikacije.



Sl. 2.3. Sučelje aplikacije Nike Training Club: Fitness

2.5. Idejno rješenje vlastite aplikacije za praćenje i preporučivanje tjelesne aktivnosti

Glavna motivacija za razvoj mobilne Android aplikacije koja omogućuje praćenje i preporučivanje tjelesne aktivnosti je podizanje osviještenosti o važnosti svakodnevne aktivnosti tako da korisnik ima plan tjelovježbe koji se prilagođava njemu. Iznad navedena postojeća rješenja su poslužila kao ideja za korisničko sučelje i mogućnosti aplikacije. Kako bi korisnik mogao spremati svoj napredak i pristupiti aplikaciji s bilo kojeg Android uređaja, aplikacija će sadržavati registraciju i prijavu korisnika. Ova aplikacija se razlikuje od ostalih po tome što će dinamički predlagati vježbe na osnovu povijesnih statistika korisnika uz opciju da korisnik može korigirati kilažu i broj ponavljanja. Time je korisniku olakšano jer neće morati voditi brigu o samostalnom kreiranju plana treninga. Detaljniji opis modela aplikacije za praćenje i preporučivanje tjelesne aktivnosti je sadržan u poglavlju 3.

3. MODEL ANDROID APLIKACIJE ZA PRAĆENJE I PREPORUČIVANJE TJELOVJEŽBE

U ovom poglavlju je detaljno opisan model Android aplikacije za praćenje i preporučivanje tjelovježbe.

3.1. Korisnički zahtjevi na mobilnu aplikaciju za praćenje i preporučivanje tjelovježbe

Kako bi korisnik mogao čuvati svoje podatke i pristupati im s drugih uređaja od njega se traži da se registrira. Pri registraciji se unosi adresa e-pošte, željena lozinka, korisnikovo ime te početni parametri. Početni parametri se zatim obrade i unose u bazu podataka Firebase Firestore [11] zajedno sa korisnikovim imenom, dok se lozinka i adresa e-pošte odvojeno automatski spremaju na Firebase Auth komponentu [12]. Nakon uspješne registracije, odnosno prijave generira se prilagođeni plan tjelovježbe. Također, korisnik može vidjeti povijest svojih tjelovježbi.

3.2. Funkcionalni zahtjevi mobilne aplikacije

3.2.1. Registracija korisnika

Pri prvom pokretanju aplikacije, korisniku će biti prikazan zaslon za registraciju koji će sadržavati gumb za registraciju, polja za unos adrese e-pošte, željene lozinke, njegovog imena te pet polja gdje korisnik upisuje koju je kilažu mogao izvoditi za traženu vježbu. Korisnik se ne može registrirati ako ne ispuni sva polja, te ako mu adresa e-pošte i lozinka nisu ispravni. Provjera ispravnosti se vrši automatski Firebase Auth komponentom koja će nakon uspješne registracije korisniku dodijeliti njegov jedinstveni ID pomoću kojeg se može identificirati. Korisnikov ID se također koristi kao oznaka dokumenta korisnika u Firebase Firestore bazi podataka kako bi se na jednostavan način povezala baza podataka s korisnikom. Ako je korisnik već registriran, ali je slučajno kliknuo na registraciju, ispod gumba za registraciju je korisniku omogućen povratak na zaslon za prijavu.

3.2.2. Prijava korisnika

Pri svakom pokretanju aplikacije, korisniku se prikazuje zaslon za prijavu koji sadrži gumb za prijavu, dva polja za unos adrese e-pošte i lozinke. Ako korisnik nema račun, ispod gumba za prijavu mu je ponuđena registracija na koju se prebacuje klikom na tekst „Registracija“. Ako je prijava uspješna, korisniku se prikazuje početni zaslon koji sadrži brojač treninga, motivacijski tekst, tekst dobrodošlice i opciju odjave te pokretanja tjelovježbe.

3.2.3. Odjava korisnika

U ovoj aplikaciji, korisnik ima opciju odjave. Korisnikovi podaci poput povijesti vježbanja, predloženi plan tjelovježbe itd. ostaju spremljeni u bazi podataka. Korisnik se može ponovno prijaviti kada želi.

3.2.4. Prikupljanje parametara pri registraciji

Korisnik pri registraciji unosi parametre koji će služiti kao početna točka za kreiranje plana tjelovježbe. Od korisnika se traži da unese težinu s kojom može izvesti najviše jedno ponavljanje tražene vježbe. Parametar koji se odnosi na ime korisnika nije bitan za algoritam, on služi kako bi interakcija s korisnikom bila ugodnija. Ovi parametri se ne koriste direktno, već će biti obrađeni i spremljeni u bazu nakon što korisnik dovrši registraciju. Potrebni parametri su prikazani u tablici 3.1.

Tablica 3.1 Prikaz korisničkih parametara

Parametar	**Tip varijable**
Ime	String
Potisak na ravnoj klupi	Double
Mrtvo dizanje	Double
Čučanj	Integer
Rameni potisak	String
Povlačenje lat sprave	String

3.2.5. Obrada prikupljenih parametara

Prema [13], preporuka za povećanje mišićne mase uključuje umjereno opterećenje koje je u rasponu od 70 do 85% jednog maksimalnog ponavljanja(1RM od engl. one rep maximum) po jednoj seriji vježbe. Svaki parametar se mora obraditi tako da se izračuna 80% kilaže od kilaže 1RM, to će se dobiti iz formule koja glasi:

$$radnaKilaža = \text{kilaža 1RM} * 0.8 \tag{3-1}$$

Formula vrijedi za svaki parametar, tako obrađeni parametri se prosljeđuju u funkciju koja u bazu postavlja sve raspoložive vježbe sa svojim početnim vrijednostima.

3.2.6. Preporuka tjelovježbe na osnovu unesenih parametara i povijesti tjelovježbe

Progresivno opterećenje [14] je vrsta treninga otpora čiji je glavni mehanizam postupno povećavanje količine stresa na tijelu. Navedeno se može postići povećavanjem broja ponavljanja u seriji vježbe koja se izvodi, međutim preporučeni raspon za povećavanje mišićne mase je osam do 12 ponavljanja. Kako bi se moglo optimalno napredovati nakon dvanaest ponavljanja u seriji, potrebno je povećati radnu kilažu koja će povećati količinu stresa izvođene vježbe te će vježbač(korisnik), moći napraviti manje ponavljanja ali će ostati u optimalnom rasponu ponavljanja. Također se preporučuje jedna do tri serije po vježbi za nove i iskusne vježbače. Za postizanje dinamičkog predlaganja vježbi su dovoljne dvije navedene preporuke. Napredak ne može biti striktno linearan, već su moguća poneka odstupanja poput umora vježbača, njegovog raspoloženja i slično, dok algoritam radi prema idealnom principu te će predlagati vježbe tako da se pri svakoj novoj tjelovježbi aktivira progresivno opterećenje. Korisnik može urediti broj ponavljanja i kilažu svake vježbe, što će osigurati pravilno napredovanje jer algoritam za predlaganje tjelovježbe pri svakom generiranju pristupa bazi podataka i dohvaća zadnja četiri treninga. Svaka vježba u sebi sadrži dvije dimenzije pokreta koji su potrebni za izmjenjivanje vježbi, movementType i sectionTargeted. Navedene dimenzije će omogućiti varijaciju tjelovježbe gdje će u svakoj tjelovježbi biti prioritizirana jedna vrsta pokreta, odnosno dio tijela. Dimenzije pokreta su prikazane u tablici 3.2, dok su njihove moguće vrijednosti prikazane u tablicama 3.3, odnosno 3.4.

Tablica 3.2 Prikaz dimenzija vježbi

Parametar	**Tip varijable**
movementType	String
sectionTargeted	String

Tablica 3.3 Moguće vrijednosti parametra „movementType“

Parametar	**Tip varijable**
push	String
pull	String
legs	String

Tablica 3.2 Moguće vrijednosti parametra „sectionTargeted“

Parametar	Tip varijable
upper	String
lower	String

Nakon dohvaćanja povijesti tjelovježbe, potrebno je izbrojati frekvenciju obje dimenzije vježbi, te uz to sortirati vježbe na uzlazni način gdje je kriterij sortiranja frekvencija vježbi. Zatim prema izbrojanim dimenzijama vježbi, algoritam odabire najmanje frekventnu dimenziju te ona postaje prioritetna u novoj tjelovježbi. Odabir vježbi će se vršiti tako da se iz liste vježbi uzmu prve četiri vježbe. Na ovakav način je osigurano pravilno progresivno opterećenje, raspored tjelovježbe u odnosu na mičićne skupine te rotaciju vježbi za pojedinu mišićnu skupinu, odnosno kretnju.

3.2.7. Pregled preporučene tjelovježbe

Kada korisnik pokrene tjelovježbu, prikazuje mu se zaslon koji u sebi sadrži plan tjelovježbe. Svaka vježba ima prikazan naziv, ikonu izvođenja vježbe i upute izvođenja vježbe koje se otvaraju u skočnom prozoru nakon što korisnik klikne na gumb za upute. Svaka vježba je raspoređena po serijama gdje se za svaku seriju nalazi kilaža i broj ponavljanja koji je određen dinamičkim načinom. Korisnik može kliknuti na iznose kilaže ili broja ponavljanja i urediti ih ako je napredovao, stagnirao ili se čak pogoršao u prikazanoj vježbi. Uređivanjem kilaže i broja ponavljanja je omogućeno dinamičko predlaganje tjelovježbe jer će svaka tjelovježba biti predložena na osnovu povijesnih statistika korisnika.

3.3. Sustavski zahtjevi mobilne aplikacije

Kako bi korisničko iskustvo bilo ugodno, potrebno je voditi računa o programskoj podršci. Sustavski zahtjevi su sastavni dio svake aplikacije, neki od zahtjeva koji su bitni za ovu aplikaciju su performanse i sigurnost koji će biti detaljno opisani u nastavku.

3.3.1. Performanse

Razvojno okruženje za izradu aplikacije u sebi sadrži alat naziva Android Profiler koji pruža uvid u performanse cijele aplikacije. Performanse aplikacije se mogu definirati kao učinkovitost sustava da u određenom vremenu obavi traženi zadatak. Najbitniji parametar za korisničko iskustvo je „vrijeme učitavanja“ i odnosi se na vremenski period koji je potreban da se s jednog zaslona

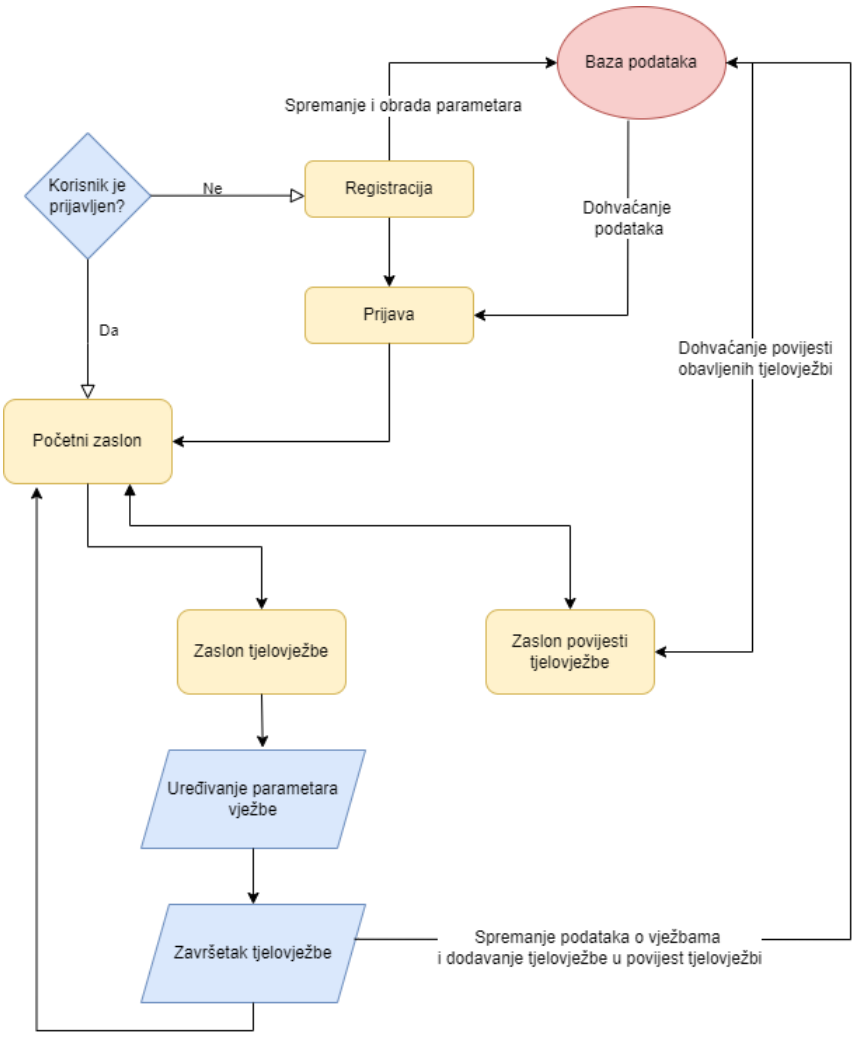
aplikacije prijeđe na drugi. Glavni uzrok dužeg vremena je dohvaćanje i postavljanje podataka. Kako bi se ovaj parametar smanjio, odnosno poboljšao potrebno je što manje pristupati bazi tako da se stvori odvojeni dio programa koje može komunicirati sa zaslonima koji se koriste pri radu aplikacije, navedeno isključuje prijavu i registraciju jer su njihove performanse zanemarive.

3.3.2. Sigurnost

Korištenjem autentikacijske usluge treće strane, korisnik ima opciju prijave i registracije te mu to omogućuje da npr. Svoju povijest tjelovježbe drži privatnom te joj pristupa samo kada i gdje on to želi.

3.4. Građa mobilne aplikacije

Model aplikacije koji odgovara navedenim korisničkim, odnosno funkcionalnim i sustavskim zahtjevima je na slici 3.1.



Sl. 3.1. Model aplikacije za praćenje i preporučivanje tjelovježbi

4. PROGRAMSKO RJEŠENJE APLIKACIJE ZA PRAĆENJE I PREPORUČIVANJE TJELOVJEŽBE

Ovo poglavlje sadrži detaljan opis programskog rješenja aplikacije i korištenih alata i tehnologija. Također je opisano slično postojeće programsko rješenje odnosno algoritam za predlaganje tjelovježbe.

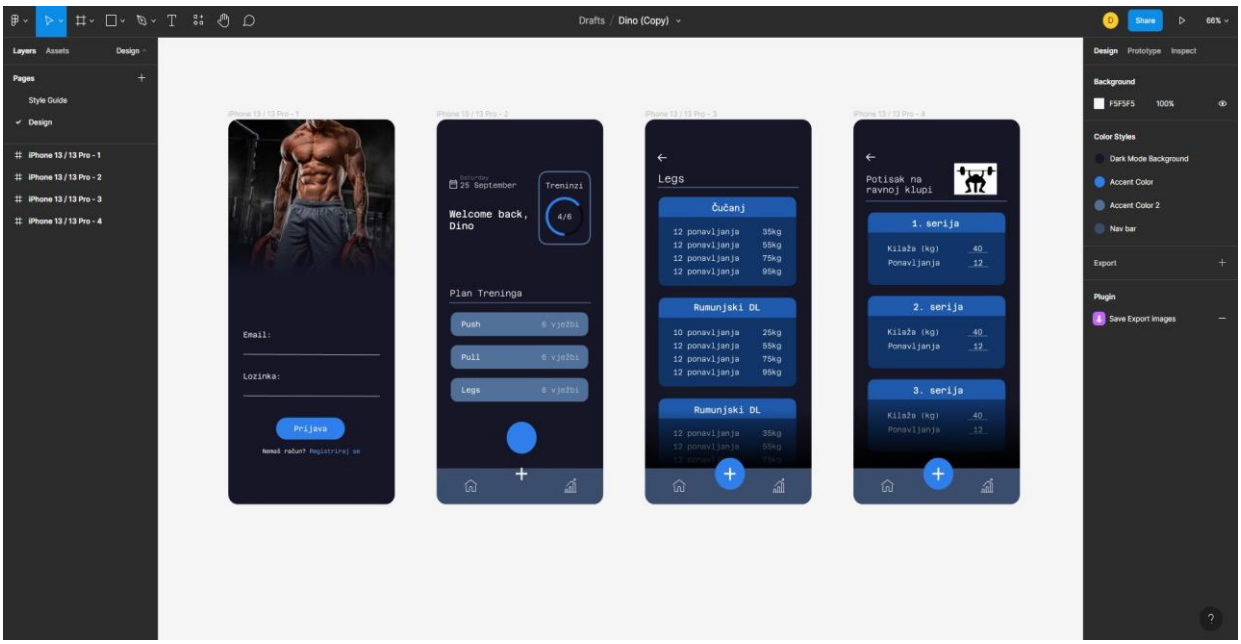
4.1. Opis potrebnih alata i tehnologije

Aplikacija je realizirana u integriranom razvojnom okruženju Android Studio. Za izradu prototipa u svrhu bržeg dizajniranja aplikacije korišten je program Figma. Prijava i registracija korisnika je realizirana putem Firestore Auth platforme, dok se za obradu i spremanje podataka koristi baza podataka Firebase Firestore. Programski kod je pisan u programskom jeziku Kotlin zbog njegove jednostavnosti i mogućnosti. Svi dizajni zaslona su napisani u XML-u.

4.1.1. Figma

Figma [15] je prvi razvojni alat na webu profesionalne razine kreiran za dizajniranje korisničkog sučelja. Omogućuje jednostavno razvijanje izgleda aplikacije i time ubrzava cjeloukupni razvoj.

Na slici 4.1 je prikazano njeno sučelje.



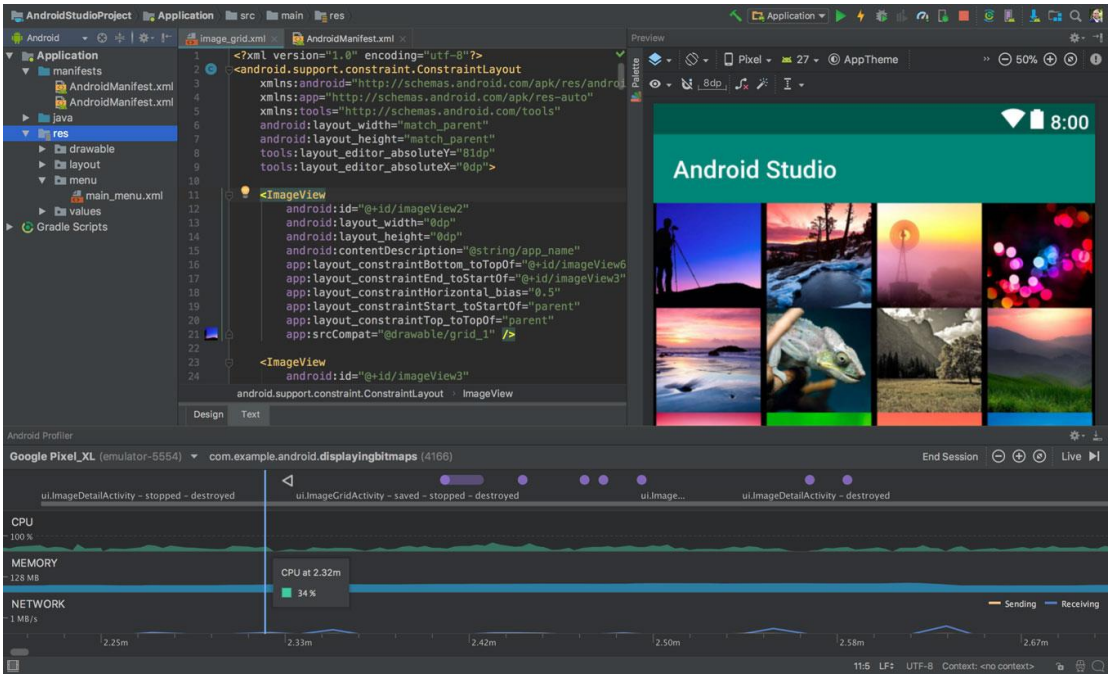
Sl. 4.1. Figma korisničko sučelje

4.1.2. Programski jezik Kotlin

Kotlin [16] je višeplatformski moderan, sažet i siguran programski jezik otvorenog koda. Podržava funkcionalno i objektno orijentiranje te ima odlične mogućnosti testiranja. Google ga je prihvatio kao vodeći jezik za razvoj mobilnih aplikacija na Android platformi 2017. godine, jezik je nastao od strane JetBrains-a u 2011 godini.. Također je interoperabilan s programskim jezikom Java. Njegova sintaksa je slična drugim jezicima poput Java i C#.

4.1.3. Android studio

Android studio [17] je službeno razvojno integrirano okruženje za razvoj Android aplikacija, te se temelji na IntelliJ IDEA [18]. Posjeduje vrlo napredne razvojne alate poput automatske nadopune koda koji pomaže ubrzati razvoj. Osim automatske nadopune, također sadrži sustav ispravljanja i otkrivanja pogrešaka. Aplikacija se može pisati u programskom jeziku Java ili Kotlin koji s vremenom postaje popularniji te je preporučen od strane Google-a. Aplikacije se mogu pokrenuti na virtualnom stroju, odnosno emulatoru koji oponaša stvarni uređaj i uvelike olakšava testiranje aplikacije. Korisničko sučelje je prikazano na slici 4.2.



Sl. 4.2. Android Studio korisničko sučelje

4.1.4. Firebase

Firebase [19] je Google-ova vodeća platforma koja uvelike olakšava realizaciju mobilnih i web aplikacija. Za ovu aplikaciju su korištene dvije komponente koje služe za bazu podataka i provjeru autentičnosti korisnika.

Firebase Auth je dio Firebase platforme koji omogućuje jednostavnu registraciju i prijavu korisnika. Podržava klasične načine prijave poput adrese e-pošte uz lozinku, prijavu putem društvenih mreža ali i višefaktorsku provjeru autentičnosti.

Firebase Firestore je baza podataka tipa noSQL u obliku dokumenata i kolekcija koja omogućuje laganu pohranu, sinkronizaciju i upite za mobilne i web aplikacije. Struktura baze je u obliku stabla, što omogućuje vrlo brze upite neovisno o dubini upita. Baza podataka također nudi jednostavnu integraciju sa Firebase Auth.

4.1.5. XML

XML [20] je jezik za označavanje podataka, kratica dolazi od Extensible Markup Language. Zbog svoje jednostavnosti i čitljivosti je iznimno popularan. Podatke pohranjuje u formatu običnog teksta što omogućuje softverski i hardverski neovisan način pohrane i prijenosa podataka. Korisničko sučelje, odnosno svaki njegov element je opisan putem XML-a, primjer dijela jednog XML dokumenta je prikazan na slici 4.3.

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

    <FrameLayout
        android:id="@+id/fl_authBackground"
        android:layout_width="wrap_content"
        android:layout_height="200dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:contentDescription="Ovo je slika"
            android:scaleType="fitXY"
            android:src="@drawable/auth_background"/>

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:contentDescription="VirtualniTrener"
            android:scaleType="fitCenter"
            android:src="@drawable/auth_icon"
            app:tint="@color/white" />

    </FrameLayout>
```

Sl. 4.3. Primjer XML dokumenta

4.1.6. GIMP

GIMP [21] je skraćenica od GNU Image Manipulation Program. Program je otvorenog koda zbog čega je postao vodeći besplatni program za izradu i uređivanje grafike. Program je korišten za obradu i izvoz slika koje su korištene u aplikaciji. Na slici 4.4 prikazano je korisničko sučelje.

Sl. 4.4. Sučelje programa Gimp

4.1.7. Asinkron rad aplikacije

Glavna nit [22] (engl. Main thread) je primarno zadužena za interakciju s komponentama iz Android korisničkog sučelja. Može se reći kako joj je primarna zadaća prikaz podataka na zaslonu. Poneke funkcionalnosti Firebase platforme zahtijevaju asinkroni rad aplikacije. Asinkroni rad aplikacije se odnosi na situacije kada se nekakva zadaća poput dohvaćanja podataka iz baze treba odvijati u isto vrijeme kada se treba prikazati, odnosno iscrtati zaslon, što je primarna zadaća glavne niti. Kako bi se izbjegao zastoj, odnosno kašnjenje i moguće pogreške nastale kašnjenjem poput krivih kalkulacija i prikaza podataka koji nemaju učitanu vrijednost, takve zadaće treba obavljati u pozadinskoj niti.

Kako bi se posao prebacio u pozadinsku nit, moguće je koristiti „*Thread*“ klasu koju nam pruža Android sustav, te korištenje Kotlin korutina [23]. Kotlin korutine nam omogućuju pozivanje funkcija i pauziranje njihovih izvođenja dok ne obavi posao za koji je zadužena. Za pozivanje funkcija unutar korutine, ona mora sadržavati „suspend“ ključnu riječ, te može biti pozvana samo u „suspend“ funkcijama ili korutinama. Prikaz korištenja korutine se nalazi u poglavlju 4.3.1.

4.2. Prikaz programskog rješenja osnovnih funkcionalnosti aplikacije

U ovom je poglavlju detaljno opisan način rada svakog pojedinog dijela sustava .

4.2.1. Prijava korisnika

Pri pokretanju aplikacije, korisniku se prikazuje zaslon za prijavu. Potrebno je upisati adresu e-pošte i lozinku u odgovarajuća polja. Za dovršetak prijave potrebno je pritisnuti gumb "PRIJAVA", nakon što se korisniku provjeri autentičnost za što se brine komponenta Firebase Auth, zatim ga se šalje na drugi zaslon koji je ujedno i početni zaslon aplikacije. Korisnikova lozinka je zaštićena od pogleda drugih ljudi tako što je svaki znak lozinke prikazan u obliku točke. Postupak prijave korisnika koji trenutno nije prijavljen se pokreće samo ako su polja za unos podataka popunjena, dok se provjera valjanosti unesenih podataka odvija na strani FirebaseAuth komponente.

Ako korisnik nije registriran, potrebno je pritisnuti tekst pod nazivom "Registracija" ispod gumba za prijavu. Zatim se korisnika šalje na zaslon za registraciju. Pri svakom sljedećem pokretanju aplikacije, ako je korisnik ostao prijavljen odmah ga se prebacuje na početni zaslon aplikacije. Zaslon za prijavu se sastoji od slike, teksta koji korisniku daje do znanja da je potrebna prijava, dva polja za unos podataka, gumba za prijavu i teksta za slučaj da korisnik nema račun koji mu nudi opciju odlaska na zaslon za registraciju. Dio koda XML dokumenta kojim je pisano korisničko sučelje nalazi se na slici 4.5. Izgled zaslona je prikazan na slici 4.6.

activity_login.xml

```
87      <Button
88          android:id="@+id/btn_Login"
89          android:layout_width="224dp"
90          android:layout_height="48dp"
91          android:layout_marginTop="32dp"
92          android:background="@color/colorAccent"
93          android:fontFamily="@font/azeretmono_regular"
94          android:foreground="?attr/selectableItemBackground"
95          android:text="@string/login"
96          android:textColor="@color/white"
97          android:textSize="20sp"
98          android:textStyle="bold"
99          app:layout_constraintEnd_toEndOf="parent"
100         app:layout_constraintHorizontal_bias="0.5"
101         app:layout_constraintStart_toStartOf="parent"
102         app:layout_constraintTop_toBottomOf="@+id/til_login_password" />
103
104     <LinearLayout
105         android:layout_width="wrap_content"
106         android:layout_height="wrap_content"
107         android:layout_marginTop="12dp"
108         android:gravity="center_vertical"
109         android:orientation="horizontal"
110         app:layout_constraintEnd_toEndOf="parent"
111         app:layout_constraintHorizontal_bias="0.51"
112         app:layout_constraintStart_toStartOf="parent"
113         app:layout_constraintTop_toBottomOf="@+id/btn_Login">
114
115         <TextView
116             android:id="@+id/tv_don_t_have_an_account"
117             android:layout_width="wrap_content"
118             android:layout_height="wrap_content"
119             android:fontFamily="@font/azeretmono_medium"
120             android:padding="5dp"
121             android:text="@string/textIfNotRegistered"
122             android:textSize="16sp" />
```

```
123
124         <TextView
125             android:id="@+id/tv_register"
126             android:layout_width="wrap_content"
127             android:layout_height="wrap_content"
128             android:fontFamily="@font/azeretmono_semibold"
129             android:padding="5dp"
130             android:text="@string/register"
131             android:textSize="20sp"
132             android:textStyle="bold" />
133     </LinearLayout>
134 </androidx.constraintlayout.widget.ConstraintLayout>
```

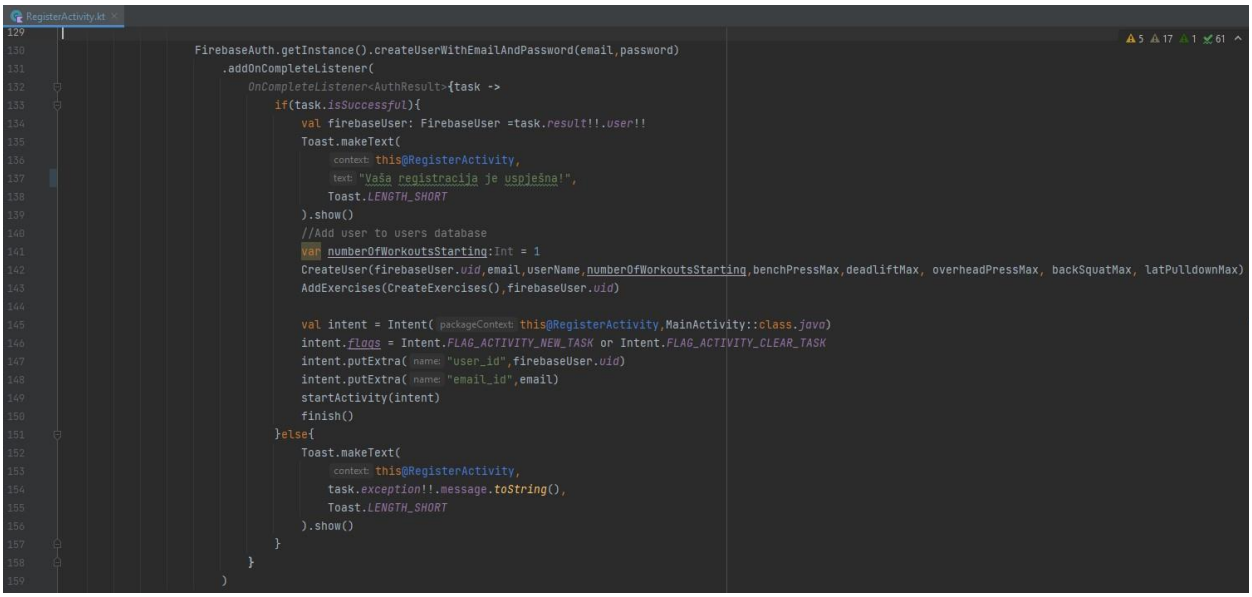
Sl. 4.5. Prikaz dijela XML koda korisničkog sučelja za prijavu

Sl. 4.6. Korisničko sučelje zaslona za prijavu

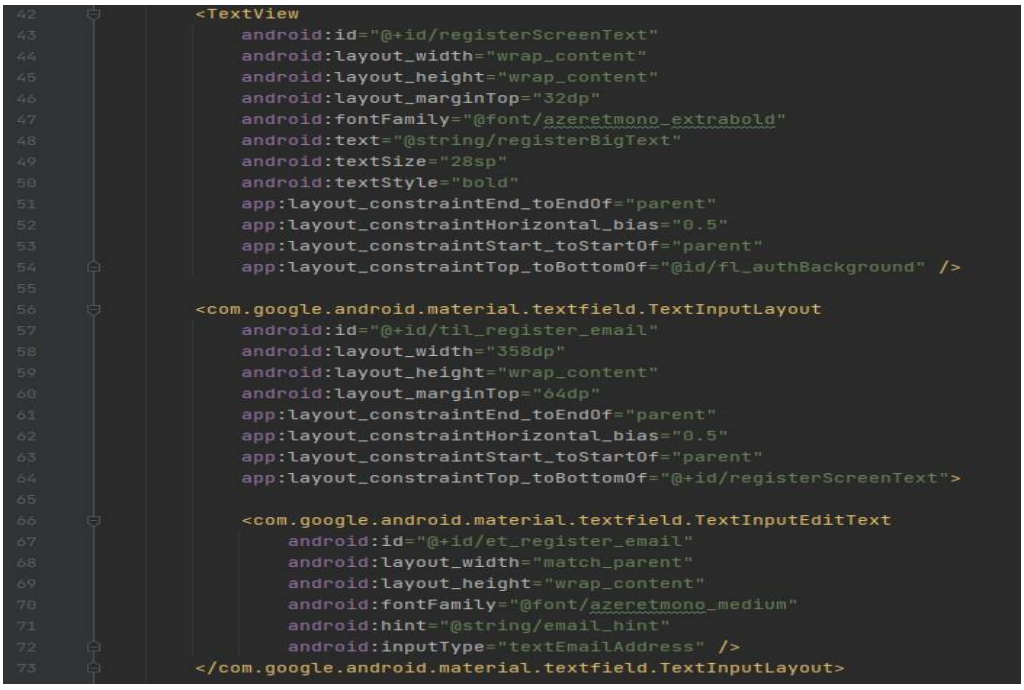
4.2.2. Registracija korisnika

Postupak registracije korisnika se vrši putem Firebase Auth platforme. Za jednostavnost aplikacije koristi se način registracije putem adrese e-pošte i lozinke. Potrebno je koristiti samo jednu metodu iz Firebase Auth platforme naziva „createUserWithEmailAndPassword“ koja kreira korisnika s jedinstvenim ID. Navedena metoda prima parametre koji su adresa e-pošte i lozinka te su oboje tipa String. Metoda kao rezultat vraća objekt tipa „AuthResult“ koji u sebi sadrži metode koje kao povratni tip imaju trenutnog korisnika. Kako bi se sačuvale performanse aplikacije, provjeravaju se polja za unos teksta, odnosno podataka za prijavu prije poziva metode za prijavu koja mora komunicirati sa serverom, te se u slučaju praznog polja korisniku prikazuje kratka nestajuća obavijest koja se kreira korištenjem „Toast“ objekta pozivom metode „makeText“ koja prima kontekst aplikacije, tekst koji bi se trebao prikazati korisniku te duljinu trajanja prikaza poruke. Nakon provjere pravilnog unosa i poziva metode za registraciju korisnika, odvija se zadnja provjera od strane Firebase Auth komponente koja se brine za valjanost unesene adrese e-pošte i pravilne lozinke korisnika. Primjer korištenja metode za registraciju je na slici 4.7. Registracija korisnika je zaseban dio sustava jer prema pravilima urednog koda treba imati samo jednu zadatak

koji je registracija korisnika. Korisničko sučelje je pisano u XML-u, kod je prikazan na slici 4.8 dok je izgled zaslona prikazan na slici 4.9.



Sl. 4.7. Programski kod za pozivanje metode za registraciju korisnika

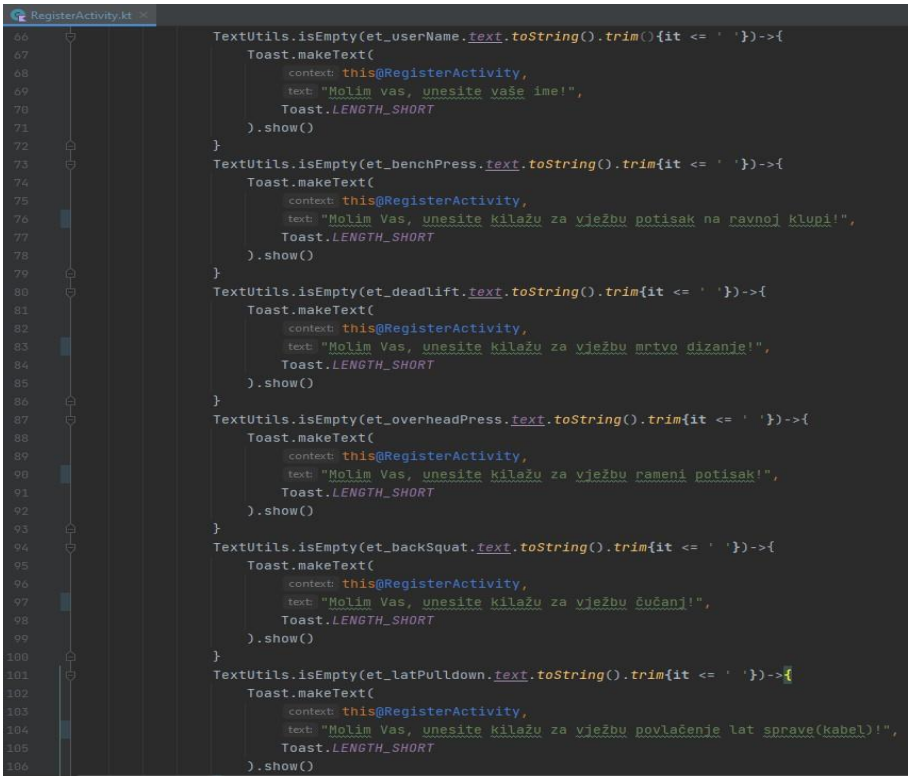


Sl. 4.8. Prikaz dijela XML koda za korisničko sučelje zaslona za registraciju

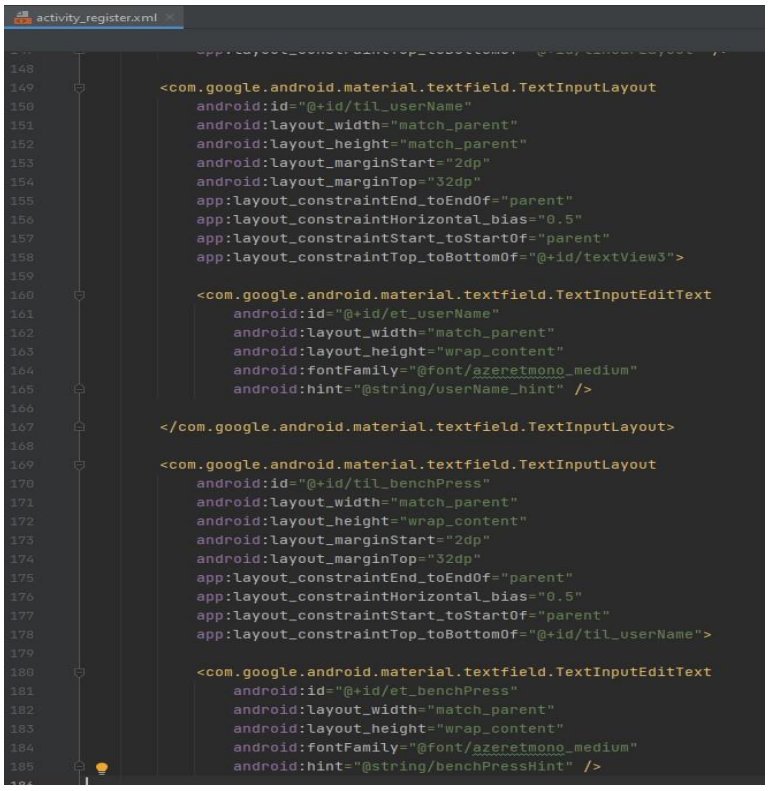
Sl. 4.9. Korisničko sučelje zaslona za registraciju

4.2.3. Unos parametara pri registraciji

U svijetu fitnessa, dobro poznato pravilo je da svaki plan tjelovježbe u teretani treba sadržavati kompleksne vježbe [24] koje zahtijevaju pokrete u više zglobova i time aktiviraju više skupina mišića. Od korisnika se traži da na zaslonu registracije osim podataka za registraciju svog računa unese i podatke o nekim kompleksnim vježbama koje su odabrane tako da pokriju aktivaciju svakog dijela tijela. Prikupljeni podaci nude opciju nadograđivanja aplikacije u budućnosti tako da korisnik može vidjeti graf napretka u pojedinoj vježbi. Za potrebe aplikacije prikupljeni parametri se trenutno koriste samo kako bi sustav mogao predložiti početnu kilažu vježbaču koja se dobiva izračunom čiji formula je prikazana u poglavlju 3. Potrebno je popuniti pet polja za unos, registracija se neće ostvariti ako sva polja nisu popunjena i u točnom formatu koji je tipa Double. Također, od korisnika se traži njegovo ime kako bi aplikacija mogla biti personalizirana korisniku kada mu prikazuje upute, obavijesti i sl. Primjer provjere popunjenosti polja je na slici 4.10, kod za prikaz sučelja je na slici 4.11.



Sl. 4.10. Programski kod za provjeru popunjenosti polja za unos podataka



Sl. 4.11. Programski kod za prikaz sučelja unosa parametara

4.2.4. Obrada unesenih parametara

Da bi se dobila kilaža s kojom će korisnik izvoditi vježbe, potrebno je podatke svake pojedine vježbe pomnožiti sa 0.8 i takve ih spremiti u bazu podataka. Primjer koda sa obradom parametara je na slici 4.12. Parametri se spremaju u objekt tipa User kako bi se prijenos mogao izvršiti odjednom umjesto da se svaki podatak pojedinačno unosi u bazu. Primjer klase „User“ je na slici 4.13. Ime korisnika nije potrebno obrađivati zato jer je ono tipa String te se takvo sprema u bazu.

```
val user=User(userID,email,name,numberOfWorkouts, benchPressMax: benchPressMax*0.8, deadliftMax: deadliftMax*0.8,
    overheadPressMax: overheadPressMax*0.8, backSquatMax: backSquatMax*0.8, latPulldownMax: latPulldownMax*0.8)
```

Sl. 4.12. Programski kod za obradu unesenih parametara

User.kt

```
1   package com.dinoknezevic.virtualnitrener
2
3   data class User(
4       val userID:String,
5       var email:String,
6       var name:String,
7       var numberOfWorkouts:Int,
8       var benchPressMax:Double,
9       var deadliftMax:Double,
10      var overheadPressMax:Double,
11      var backSquatMax:Double,
12      var latPulldownMax:Double,
13  )
14
```

Sl. 4.13. Programski kod klase User

Instanca klase User u sebi također sadrži varijablu „numberOfWorkouts“ tipa „Int“ koja služi kao brojač odrađenih tjelovježbi. Njena početna vrijednost se postavlja prilikom kreacije korisnika te joj je vrijednost 0.

4.2.5. Kreiranje baze podataka

Kako bi svaki korisnik imao svoju bazu podataka potrebno je kreirati dokument u kolekciji „Users“ korištenjem jedinstvenog identifikatora za što može poslužiti jedinstveni ID korisnika koji se kreira prilikom registracije novog korisnika. Za pristup novo-kreiranom trenutnom korisniku potrebno je provjeriti je li korisnik kreiran putem „getResult“ metode, zatim metodom „getUser“ dohvatiti instancu trenutnog korisnika. Zatim je potrebno pozvati metodu „getUid“ koja će kao rezultat vratiti String koji predstavlja jedinstveni ID trenutnog korisnika. Zbog preglednosti koda, kreiranje dokumenta i njegovih potkolekcija svakog pojedinog korisnika je napisano u metodi „CreateUser“. Navedena metoda prima jedinstveni ID korisnika, njegovu adresu e-pošte, početni broj odrađenih treninga koji u trenutku kreiranja ima vrijednost 0 te parametre o kilažama kompleksnih vježbi, način rada metode je prikazan na slici 4.14.

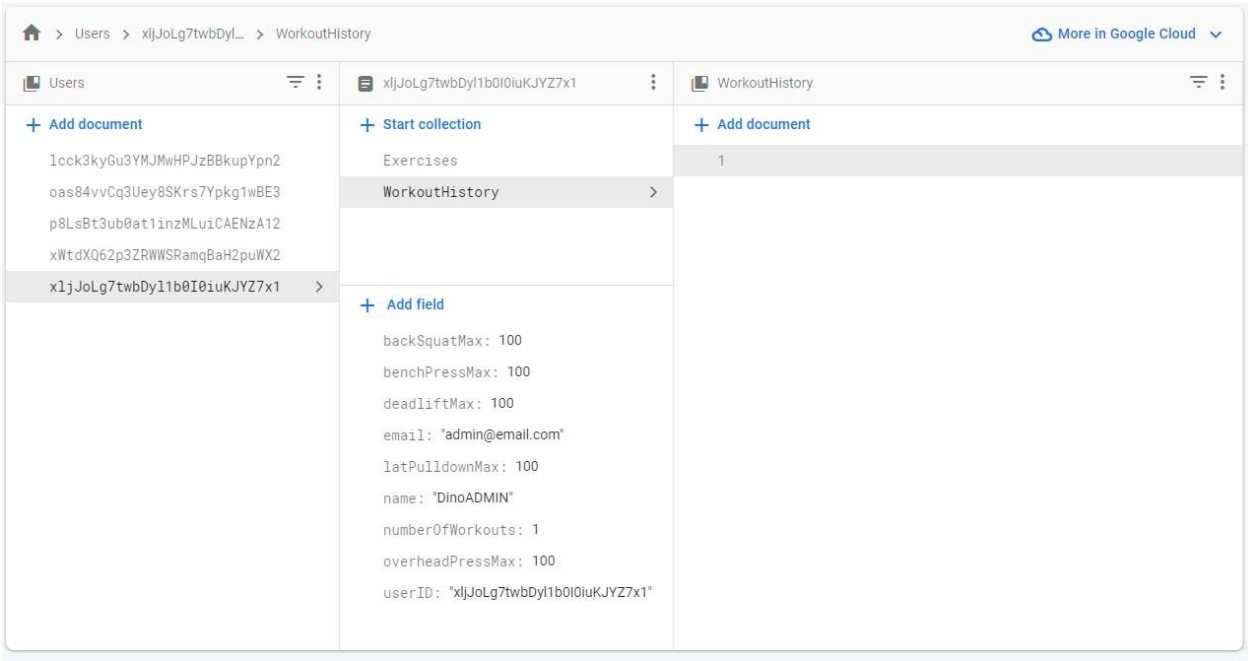
```
195    private fun CreateUser(userID:String,email:String,name:String,numberOfWorkouts:Int,benchPressMax:Double,
196                           deadliftMax:Double,overheadPressMax:Double,backSquatMax:Double,latPulldownMax:Double){
197        val user=User(userID,email,name,numberOfWorkouts, benchPressMax: benchPressMax*0.8, deadliftMax: deadliftMax*0.8,
198             overheadPressMax: overheadPressMax*0.8, backSquatMax: backSquatMax*0.8, latPulldownMax: latPulldownMax*0.8)
199
200        val benchWeight:Double = benchPressMax*0.8
201        val squatWeight:Double = backSquatMax*0.8
202
203        InsertUser(user,userID,benchWeight,squatWeight,numberOfWorkouts)
204    }
```

Sl. 4.14. Programski kod metode CreateUser()

U njoj se kreira objekt tipa User koji se zatim predaje kao argument u pozivu metode „InsertUser“, funkcionalnost metode je prikazana na slici 4.15. Unutar „InsertUser“ metode se pristupa kolekciji „Users“, zatim se referenci kolekcije „Users“ dodaje dokument koji kao argument prima korisnikov jedinstveni ID te se u dokument postavlja objekt tipa User kako bi se svi parametri postavili u istom trenutku. Nakon uspješnog kreiranja dokumenta, potrebno je kreirati potkolekciju „WorkoutHistory“ koja sadrži povijest svih odrađenih tjelovježbi. U nju će se spremiti početne tjelovježbe kako bi algoritam imao početnu točku za stvaranje planova tjelovježbe u budućnosti te ona neće biti ubrojana u odrađene tjelovježbe. Primjer strukture kreirane baze podataka nalazi na slici 4.16.

```
164    private fun InsertUser(user: User,userID: String,weightBench: Double,weightSquat: Double,numberOfWorkouts: Int){
165        val usersReference=db.collection( collectionPath: "Users")
166
167        usersReference.document(userID).set(user).addOnCompleteListener{ it: Task<Void!>
168            when{
169                it.isSuccessful->{
170                    Toast.makeText( context: this, text: "Korisnik uspješno kreiran",Toast.LENGTH_SHORT).show()
171                }
172                else ->{
173                    Toast.makeText( context: this, text: "Korisnik nije kreiran",Toast.LENGTH_SHORT).show()
174                }
175            }
176        }
177
178        val firstWorkoutData = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
179            hashMapOf(
180                "pullCount" to 0,
181                "pushCount" to 4,
182                "legsCount" to 2,
183                "upperCount" to 4,
184                "lowerCount" to 2,
185                "workoutId" to 1,
186                "dateOfWorkout" to LocalDateTime.now()
187            )
188        } else {
189            TODO( reason: "VERSION.SDK_INT < O")
190        }
191        usersReference.document(userID).collection( collectionPath: "WorkoutHistory").document( documentPath: "1").set(firstWorkoutData)
192        AddFirstWorkoutExercises(BuildFirstWorkout(weightBench, weightSquat),userID, numberOfWorkouts)
193    }
```

Sl. 4.15. Programski kod metode InsertUser()



Sl. 4.16. Struktura kreirane baze podataka

4.2.6. Postupak kreiranja početne tjelovježbe

Početna tjelovježba se sprema u potkolekciju „WorkoutHistory“ u dokument koji kao ID ima redni broj tjelovježbe, zatim se u dokumentu kreira potkolekcija naziva „ExercisesPerformed“ gdje je svaka pojedina vježba predstavljena dokumentom koji kao ID ima naziv vježbe. Da bi se navedeno ostvarilo potrebno je pozvati metodu „AddFirstWorkoutExercises“ koja za argumente ima listu objekata tipa ExerciseWorkout, korisnikov jedinstveni ID i broj odrađenih tjelovježbi što je vidljivo na slici 4.17.

```
316     private fun AddFirstWorkoutExercises(exercises:MutableList<ExerciseWorkout>,userID: String, numberOfWorkouts: Int){
317         val userReference= db.collection( collectionPath: "Users")
318         val exercisesSaveLocation = userReference.document(userID).collection( collectionPath: "WorkoutHistory")
319             .document(numberOfWorkouts.toString()).collection( collectionPath: "ExercisesPerformed")
320
321         for(exercise in exercises){
322             exercisesSaveLocation.document(exercise.uniqueId).set(exercise)
323         }
324
325     }
326 }
327
```

Sl. 4.17. Programski kod metode AddFirstWorkoutExercises

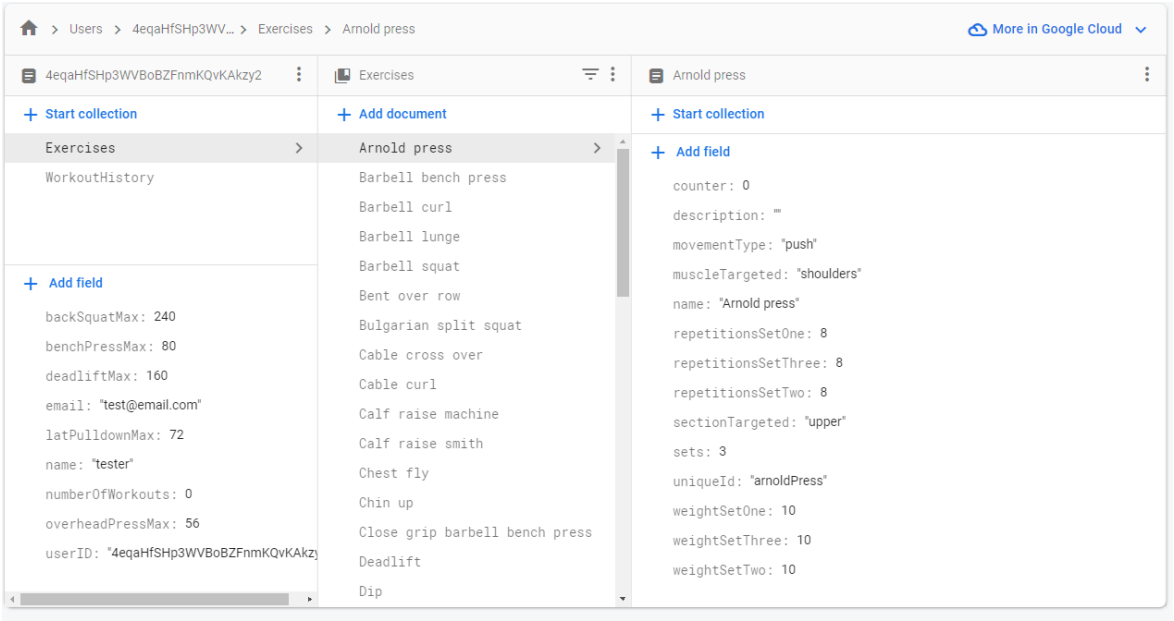
Za stvaranje liste objekata tipa ExerciseWorkout potrebno je pozvati metodu „BuildFirstWorkout“ koja kreira i vraća listu objekata. Izvedba klase ExerciseWorkout se nalazi na slici 4.18. Zatim je potrebno primljenu listu objekata obraditi tako da se petljom svaki element liste spremi u bazu podataka. Svaki dokument koji predstavlja jednu odrađenu tjelovježbu ima parametre koji broje koliko je dimenzija vježbi bilo u tjelovježbi. Navedeni parametri su kasnije potrebni algoritmu za dinamičko preporučivanje tjelovježbi. Programski kod za metodu „BuildFirstWorkout“ je na slici 4.19. Na isti princip rada djeluje i metoda „AddExercises“ koja se poziva nakon kreacije korisnika, odnosno postupka kreiranja dokumenta trenutnog korisnika, razlika je u tome što se objekti nastali u metodi „CreateExercises“, a koja se poziva unutar metode „AddExercises“ sprema u potkolekciju naziva „Exercises“. Navedena potkolekcija služi za spremanje zadnje kilaže s kojom je korisnik izvodio vježbu te sadrži upute o izvođenju vježbe. Izgled kreirane potkolekcije i njenih dokumenata je prikazan na slici 4.20.

```kotlin
data class ExerciseWorkout(
    val name:String?=null,
    val uniqueId:String?=null,
    val description: String?=null,

    val muscleTargeted: String?=null,
    //upper or lower
    val sectionTargeted: String?=null,
    //push or pull or legs
    val movementType:String?=null,

    var weightSetOne: Double?=null,
    var weightSetTwo: Double?=null,
    var weightSetThree: Double?=null,
    var repetitionsSetOne: Int?=null,
    var repetitionsSetTwo: Int?=null,
    var repetitionsSetThree: Int?=null,
    var sets: Int?=null,
    var counter: Int?=null
){
    operator fun iterator(): List<Pair<String, String?>> {
        return listOf("name" to name)
    }
}
```

Sl. 4.18. Izvedba klase ExerciseWorkout

```kotlin
private fun BuildFirstWorkout(weightBench:Double,weightSquat:Double): MutableList<ExerciseWorkout>{
    val exercises: MutableList<ExerciseWorkout> = mutableListOf()
    //Chest triceps calves

    exercises.add(ExerciseWorkout( name: "Barbell bench press", uniqueId: "barbellBenchPress", description: "", muscleTargeted: "chest", sectionTargeted: "upper",
        movementType: "push",weightBench,weightBench,weightBench, repetitionsSetOne: 11, repetitionsSetTwo: 12, repetitionsSetThree: 13, sets: 3, counter: 0))

    exercises.add(ExerciseWorkout( name: "Barbell bench press", uniqueId: "barbellBenchPress", description: "", muscleTargeted: "chest", sectionTargeted: "upper",
        movementType: "push",weightBench,weightBench,weightBench, repetitionsSetOne: 8, repetitionsSetTwo: 8, repetitionsSetThree: 8, sets: 3, counter: 0))

    exercises.add(ExerciseWorkout( name: "Cable cross over", uniqueId: "cableCrossOver", description: "", muscleTargeted: "chest", sectionTargeted: "upper", movementType: "push",
        weightSetOne: 15.0, weightSetTwo: 15.0, weightSetThree: 15.0, repetitionsSetOne: 10, repetitionsSetTwo: 10, repetitionsSetThree: 10, sets: 3, counter: 0))

    exercises.add(ExerciseWorkout( name: "Skull crusher", uniqueId: "skullcrusher", description: "", muscleTargeted: "triceps", sectionTargeted: "upper", movementType: "push",
        weightSetOne: 25.0, weightSetTwo: 25.0, weightSetThree: 25.0, repetitionsSetOne: 10, repetitionsSetTwo: 10, repetitionsSetThree: 10, sets: 3, counter: 0))

    exercises.add(
        ExerciseWorkout( name: "Overhead press", uniqueId: "overheadPress", description: "", muscleTargeted: "shoulders", sectionTargeted: "upper", movementType: "push",
     weightSetOne: 35.0, weightSetTwo: 35.0, weightSetThree: 35.0, repetitionsSetOne: 8, repetitionsSetTwo: 8, repetitionsSetThree: 8, sets: 3, counter: 0))

    exercises.add(ExerciseWorkout( name: "Barbell Squat", uniqueId: "barbellSquat", description: "", muscleTargeted: "quadriceps", sectionTargeted: "lower", movementType: "legs",
        weightSquat,weightSquat,weightSquat, repetitionsSetOne: 8, repetitionsSetTwo: 8, repetitionsSetThree: 8, sets: 3, counter: 0))

    exercises.add(ExerciseWorkout( name: "Barbell lunge", uniqueId: "barbellLunge", description: "", muscleTargeted: "quadriceps", sectionTargeted: "lower", movementType: "legs",
        weightSetOne: 40.0, weightSetTwo: 40.0, weightSetThree: 40.0, repetitionsSetOne: 10, repetitionsSetTwo: 10, repetitionsSetThree: 10, sets: 3, counter: 0))

    return exercises
}
```

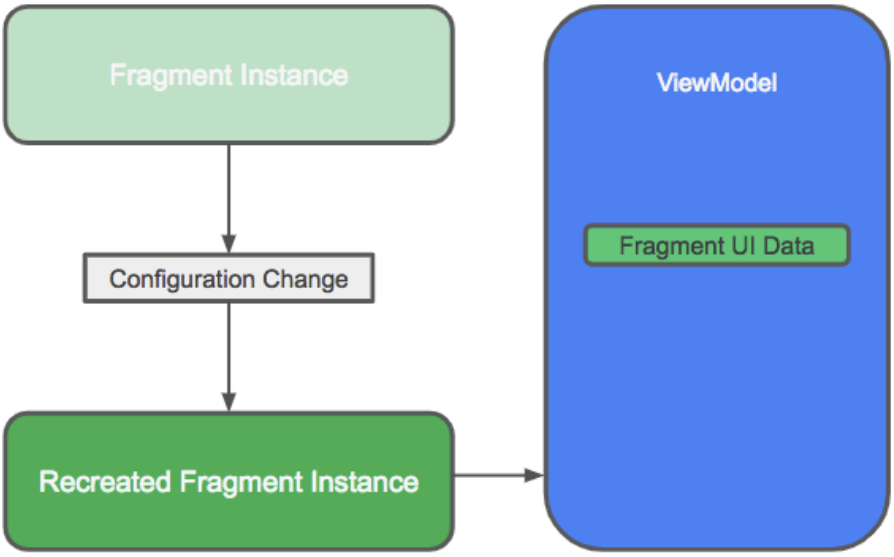
Sl. 4.19. Programski kod metode BuildFirstWorkout



Sl. 4.20. Struktura potkolekcije „Exercises“

4.2.7. ViewModel pristup

ViewModel [25] klasa je dio arhitekturalnog obrasca MVVM (engl. Model-View-ViewModel). Klasa ViewModel dizajnirana je za pohranu i upravljanje podacima povezanima s korisničkim sučeljem na način svjesnog životnog ciklusa, omogućuje podacima da prežive promjene konfiguracije kao što su rotacije zaslona. Prilikom korištenja aplikacije, korisnik će promijeniti više zaslona, svaki zaslon ima svoju zadaću te zahtijeva određene podatke za izvršavanje svojih zadaća. Tijekom rada aplikacije potrebna je komunikacija između fragmenata, poput provjere stanja tjelovježbe kako korisnik ne bi slučajno promijenio zaslon te time prekinuo izvođenje tjelovježbe uz moguće pogreške sustava. Unutar komponente pogled(*ViewModel*) su sadržane varijable poput trenutnog stanja tjelovježbe, liste svih i trenutnih vježbi i slično. Takav pristup omogućava bolje performanse aplikacije ali i lakše razvijanje aplikacije jer su podaci sadržani na jednom mjestu. Također, fragmenti ne moraju znati ništa jedan o drugome što uvelike olakšava razvijanje. Primjer sheme je na slici 4.21. Programska izvedba je prikazana na slici 4.22.



Izvor: https://developer.android.com/codelabs/kotlin-android-training-view-model

Sl. 4.21. MVVM shema

```
class SharedViewModel: ViewModel() {

    var workoutState = MutableLiveData<Boolean>( value: false)
    var completedWorkoutsCount = MutableLiveData<Int>( value: 0)

    var allExercises: MutableList<ExerciseWorkout> = mutableListOf()
    var currentWorkoutExercises: ArrayList<ExerciseWorkout> = arrayListOf()

    fun ProgressiveOverload(){
        for (exercise in currentWorkoutExercises){
            if (exercise.repetitionsSetOne!!.toInt() >12 ){
                exercise.repetitionsSetOne=8
                exercise.weightSetOne=exercise.weightSetOne!! +5.0
            }
            else if (exercise.repetitionsSetTwo!!.toInt()>12){
                exercise.repetitionsSetTwo=8
                exercise.weightSetTwo=exercise.weightSetTwo!! +5.0
            }
            else if (exercise.repetitionsSetThree!!.toInt()>12){
                exercise.repetitionsSetThree=8
                exercise.weightSetThree=exercise.weightSetThree!!+5.0
            }
        }
    }
}
```

Sl. 4.22. Prikaz programske izvedbe ViewModel-a

4.3. Generiranje plana tjelovježbe

Zbog čitljivosti koda, dijelovi koda za preporučivanje tjelovježbe su podijeljeni u nekoliko manjih funkcija koje se zatim pozivaju u metodi „GenerateWorkout“.

4.3.1. Pronalazak najmanje zastupljenog pokreta

Naime, preporučivanje se zasniva na povijesnim karakteristikama korisnika, odnosno povijesti njegove tjelovježbe. Da bi aplikacija mogla točnije predlagati vježbe potrebno je da provjeri nekoliko posljednjih tjelovježbi. Pronalazak se zasniva na pogledu zadnje četiri tjelovježbe.

Za pristup povijesti tjelovježbe je odgovorna metoda naziva „GetWorkoutHistory“, njen programski kod se nalazi na slici 4.23.

Sl. 4.23. Prikaz metode GetWorkoutHistory()

Zbog moguće trenutne nedostupnosti podataka ili dužeg čekanja na odgovor, potrebno je koristiti korutine koje neće blokirati glavnu nit u izvršavanju svojih dužnosti. Metoda pristupa bazi podataka, sortira povijest tjelovježbi prema rednom broju vježbe silazno te uzima četiri vježbe. Nakon toga čeka da se navedeni upit izvrši te kao rezultat vraća *QuerySnapshot?* koji sadrži rezultate upita. Rezultat se obrađuje u metodi „GetDataDimensions“ prikazanoj na slici 4.24. Zadaća metode je prebrojavanje dimenzija odrađenih vježbi iz prošlih tjelovježbi. Funkcija iz svakog dokumenta gdje jedan dokument predstavlja jednu tjelovježbu, taj dokument konvertira u objekt tipa *ExerciseType* te ih dodaje u listu istog tipa. Korištenjem liste je omogućeno jednostavno prebrojavanje svake dimenzije, pozivanjem ugrađene metode „sumOf“. Funkcija vraća rezultat u objektu tipa *ExerciseType* kojemu su parametri postavljeni na prebrojane vrijednosti.

```
179     private suspend fun GetDataDimensions(): ExerciseType {
180
181         val returnValue = ExerciseType( pushCount: 0, pullCount: 0, legsCount: 0)
182         val returns: MutableList<ExerciseType> = mutableListOf()
183         val documentsReference = GetWorkoutHistory()
184         for (document in documentsReference!!){
185             val retrievedDocument = document.toObject<ExerciseType>()
186             returns.add(retrievedDocument)
187         }
188         val pushCount:Long = returns.sumOf { it.pushCount!! }
189         val pullCount:Long = returns.sumOf { it.pullCount!! }
190         val legCount:Long = returns.sumOf { it.legsCount!! }
191
192         returnValue.pushCount = pushCount
193         returnValue.pullCount = pullCount
194         returnValue.legsCount = legCount
195
196         return returnValue
197     }
198
199 }
```

Sl. 4.24. Prikaz metode GetDataDimensions()

4.3.2. Dohvaćanje svih vježbi u bazi podataka

Da bi aplikacija mogla preporučiti vježbe za trenutnu tjelovježbu, potrebna joj je lista svih vježbi jer one u sebi sadrže parametre koji se ažuriraju nakon svake tjelovježbe, te time olakšava implementaciju progresivnog opterećenja.

```
152     private suspend fun GetExercises(){
153         val currentUserReference = FirebaseAuth.getInstance().uid.toString()
154         val currentUser = db.collection( collectionPath: "Users").document(currentUserReference)
155         val exercisesReference = currentUser.collection( collectionPath: "Exercises")
156
157         val documentSnapshot = exercisesReference.get().await()
158
159         for (document in documentSnapshot){
160
161             val retrievedDocument = document.toObject<ExerciseWorkout>()
162             sharedViewModel.allExercises.add(retrievedDocument)
163         }
164     }
```

Sl. 4.25. Prikaz metode GetExercises()

Prema slici 4.25. je vidljivo kako metoda pristupa potkolekciji „Exercises“ u kojoj se nalaze sve vježbe, zatim korištenjem petlje svaki pojedini element konvertira u objekt tipa *ExerciseWorkout* te je time dohvaćena vježba iz baze. Osim što aplikacija treba listu svih vježbi kako bi postavila ispravne vrijednosti na početku tjelovježbe, ona treba prilikom završetka tjelovježbe iste te podatke ažurirati. Kako bi navedeno bilo moguće lista vježbi se nalazi unutar ViewModel-a kojem

može pristupiti svaki fragment. Time smo smanjili dupliciranje koda i poboljšali ukupno korisničko iskustvo jer će se zaslon za odrađivanje tjelovježbe brže prikazati.

4.3.3. Dohvaćanje posljednjih vježbi

Zbog kompleksnosti obavljenog posla, on je izdvojen u metodu naziva „GetLatestExercises“, programski kod se nalazi na slici 4.26.

```
137     private suspend fun GetLatestExercises(
138         uid: String?,
139         listOfExercises: MutableList<ExerciseWorkout>
140     ) {
141         val workoutHistoryDocuments = GetWorkoutHistory()
142         for (document in workoutHistoryDocuments!!) {
143
144             val documentWorkoutId = document.data["workoutId"].toString()
145             val currentCollection: CollectionReference =
146                 db.collection( collectionPath: "Users/$uid/WorkoutHistory/${documentWorkoutId}/ExercisesPerformed")
147
148             currentCollection.get().addOnSuccessListener { result ->
149                 for (document in result) {
150                     val retrievedDocumentData = document.toObject<ExerciseWorkout>()
151                     listOfExercises.add(retrievedDocumentData)
152                 }
153             }
154         }
155     }
```

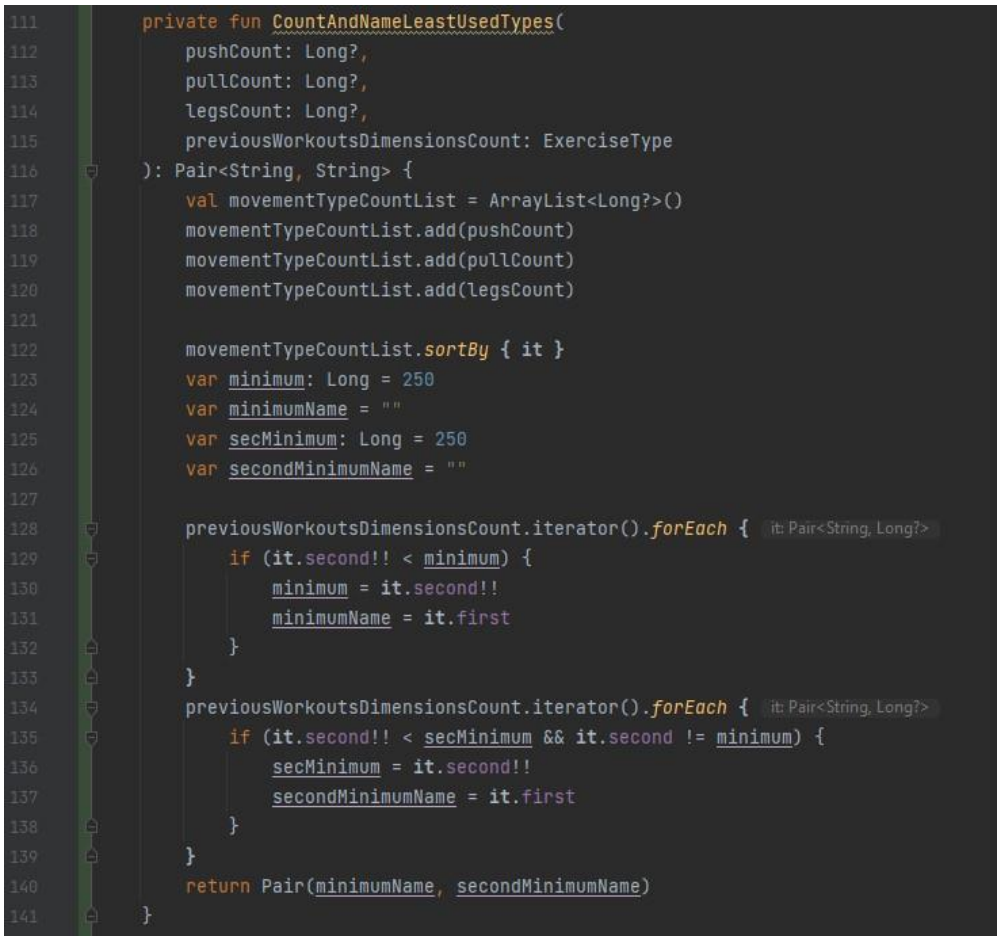
Sl. 4.26. Prikaz metode GetLatestExercises()

Metoda kao parametre prima jedinstveni ID trenutnog korisnika kako bi mogla pristupiti njegovim podacima u bazi podataka i listu vježbi u koju će biti spremljene vježbe iz posljednje četiri tjelovježbe. Funkcija obavlja upit korištenjem metode „GetWorkoutHistory“, zatim svakom dokumentu u povijesti tjelovježbe pristupa i privremeno sprema redni broj tjelovježbe. Redni broj tjelovježbe je potreban kako bi se moglo pristupiti potkolekciji svakog dokumenta tjelovježbe jer su u njoj sadržani dokumenti koji predstavljaju vježbe koje su se izvodile u tjelovježbi. Ako je pristup potkolekciji bio uspješan, što se provjerava korištenjem promatrača „addOnSuccessListener“, svaki dokument u potkolekciji se pretvara(konvertira) u objekt tipa *ExerciseWorkout*, te se dodaje u listu koja predstavlja sve vježbe koje su se izvodile u proteklim tjelovježbama.

4.3.4. Dohvat naziva pokreta potrebnih za preporučivanje vježbi

Kako bi aplikacija znala koje vježbe predložiti, mora imati informacije o zastupljenosti pokreta ali i njihovim nazivima. Radi preglednosti koda, kreirana je metoda

„CountAndNameLeastUsedTypes“ koja pronalazi najmanje i drugu najmanje zastupljenu vrstu pokreta te im pridružuje odgovarajuće nazive.



Sl. 4.27. Prikaz metode CountAndNameLeastUsedTypes()

Prema slici 4.27 je vidljivo da funkcija prima parametar tipa *ExerciseType* koji sadrži prebrojane dimenzije pokreta ali bez njihovih naziva. Također prima frekvenciju svake dimenzije pokreta u obliku zasebnih parametara koji su dobiveni iz parametra tipa *ExerciseType*. Za pronalazak najmanje zastupljene dimenzije pokreta koristi se for petlja koja svaki element uspoređuje s vrijednosti „minimum“ koja je postavljena imajući na vidu da je uvijek veća od moguće koja se dobiva iz povijesti tjelovježbe. Ako je trenutni element u petlji manji od „minimum“ onda njegova vrijednost postaje novi „minimum“, te se njegov naziv sprema u varijablu „minimumName“, koja se vraća kao rezultat. Kako bi se pronašla druga po redu zastupljena dimenzija pokreta potrebno je opet proći kroz listu dimenzija pokreta ali osim usporedbe vrijednost trenutnog elementa s minimalnom potrebno je provjeriti i da trenutna vrijednost ne odgovara najmanjoj koju smo već pronašli. Primjer poziva metode se nalazi na slici 4.28.

```
val pushCount=previousWorkoutsDimensionsCount.pushCount
val pullCount=previousWorkoutsDimensionsCount.pullCount
val legsCount=previousWorkoutsDimensionsCount.legsCount
var (minimumName, secondMinimumName) = CountAndNameLeastUsedTypes(
    pushCount,
    pullCount,
    legsCount,
    previousWorkoutsDimensionsCount
)
```

Sl. 4.28. Prikaz poziva metode CountAndNameLeastUsedTypes()

4.3.5. Preporuka vježbi za trenutnu tjelovježbu

Nakon što se dobiju nazivi pokreta koji će biti preporučeni u tjelovježbi potrebno je iz liste svih vježbi odabrati četiri vježbe koje za dimenziju pokreta odgovaraju najmanje zastupljenom pokretu, te dvije vježbe koje odgovaraju drugom po redu zastupljenom pokretu. Lista svih vježbi se dohvaća iz ViewModel-a, zatim se kroz nju prolazi for petljom te se stvaraju liste za svaku predanu dimenziju pokreta. Novostvorene liste se zatim sortiraju uzlazno prema vrijednosti brojača koji govori koliko puta se pojedina vježba izvela. Nakon toga stvara se lista koja treba sadržavati samo preporučene vježbe.U novostvorenu listu se dodaju vježbe koje po svojoj dimenziji pokreta odgovaraju predanoj dimenziji pokreta. Također redoslijed uzimanja vježbi je uzlazan jer će se tako uzeti najmanje korištene vježbe za predane dimenzije pokreta. Da bi korisnik mogao vidjeti vježbe i odraditi tjelovježbu, lista se sprema u ViewModel, tako da se for petljom svaki element liste vježbi za trenutnu tjelovježbu doda pojedinačno u listu trenutnih vježbi ViewModel-a. Radi preglednosti, kod je napisan u funkciji „RecommendExercises“. Programska izvedba je prikazana na slici 4.29.

```
private fun RecommendExercises(minimumName: String, secondMinimumName: String) {
    val listOfAllExercises = sharedViewModel.allExercises
    val primaryExercisesList: ArrayList<ExerciseWorkout> = arrayListOf()
    val secondaryExercisesList: ArrayList<ExerciseWorkout> = arrayListOf()
    for (exercise in listOfAllExercises) {
        if (exercise.movementType == minimumName) {
            primaryExercisesList.add(exercise)
        } else if (exercise.movementType == secondMinimumName) {
            secondaryExercisesList.add(exercise)
        }
    }
    val listOfRecommendedExercises: ArrayList<ExerciseWorkout> = arrayListOf()
    primaryExercisesList.sortBy { it.counter }
    secondaryExercisesList.sortBy { it.counter }

    listOfRecommendedExercises.add( index: 0, primaryExercisesList[0])
    listOfRecommendedExercises.add( index: 1, primaryExercisesList[1])
    listOfRecommendedExercises.add( index: 2, primaryExercisesList[2])
    listOfRecommendedExercises.add( index: 3, primaryExercisesList[3])
    listOfRecommendedExercises.add(secondaryExercisesList[0])
    listOfRecommendedExercises.add(secondaryExercisesList[1])

    for (element in listOfRecommendedExercises) {
        sharedViewModel.currentWorkoutExercises.add(element)
    }
}
```

Sl. 4.29. Prikaz metode RecommendExercises()

4.3.6. Preporuka tjelovježbe

Zadnji korak je pozivanje svih metoda pravilnim redoslijedom. Ovakav način pisanja koda je uvelike poboljšao preglednost koda, te omogućio lakše izmjene i nadogradnju. Metoda koja objedinjuje sve metode i spaja ih u jednu cjelinu naziva se „GenerateWorkout“, njezina izvedba prikazana je na slici 4.30. Da bi se izbjegle moguće pogreške zbog asinkronog rada metode, korištena je klasa *Thread* te je pomoću nje dodana kratka odgoda koja osigurava ispravan rad aplikacije. Metodu je potrebno pozvati pri stvaranju fragmenta kako bi se prilikom svake prijave korisnika stvorila nova tjelovježba.

```
85    private fun RecommendExercises(minimumName: String, secondMinimumName: String) {
86        val listOfAllExercises = sharedViewModel.allExercises
87        val primaryExercisesList: ArrayList<ExerciseWorkout> = arrayListOf()
88        val secondaryExercisesList: ArrayList<ExerciseWorkout> = arrayListOf()
89        for (exercise in listOfAllExercises) {
90            if (exercise.movementType == minimumName) {
91                primaryExercisesList.add(exercise)
92            } else if (exercise.movementType == secondMinimumName) {
93                secondaryExercisesList.add(exercise)
94            }
95        }
96        val listOfRecommendedExercises: ArrayList<ExerciseWorkout> = arrayListOf()
97        primaryExercisesList.sortBy { it.counter }
98        secondaryExercisesList.sortBy { it.counter }
99
100       listOfRecommendedExercises.add( index: 0, primaryExercisesList[0])
101       listOfRecommendedExercises.add( index: 1, primaryExercisesList[1])
102       listOfRecommendedExercises.add( index: 2, primaryExercisesList[2])
103       listOfRecommendedExercises.add( index: 3, primaryExercisesList[3])
104       listOfRecommendedExercises.add(secondaryExercisesList[0])
105       listOfRecommendedExercises.add(secondaryExercisesList[1])
106
107       for (element in listOfRecommendedExercises) {
108           sharedViewModel.currentWorkoutExercises.add(element)
109       }
110   }
```

Sl. 4.30. Prikaz metode GenerateWorkout()

4.3.7. Progresivno opterećenje

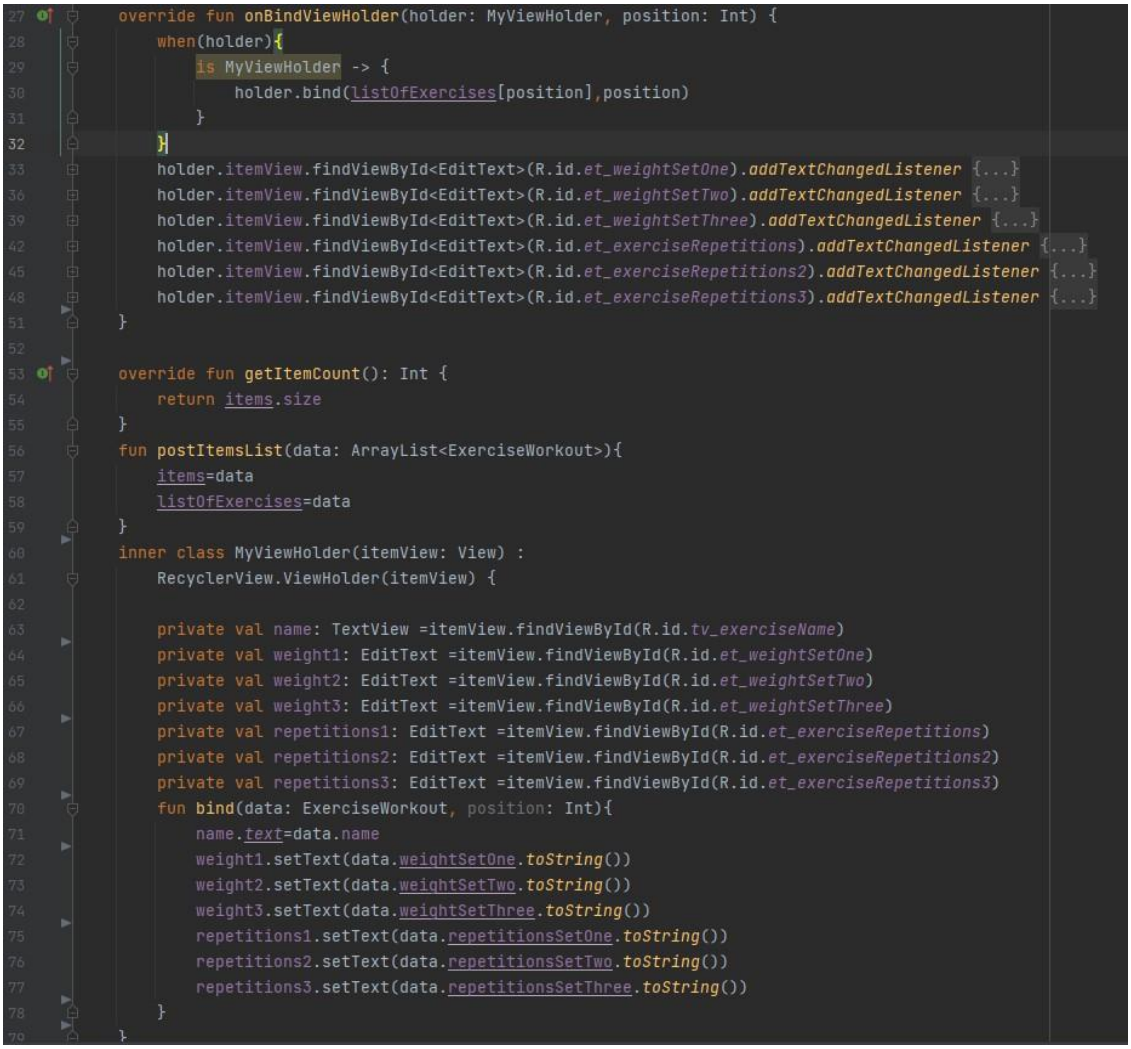
Nakon preporuke tjelovježbe, prije njenog izvođenja je potrebno primijeniti progresivno opterećenje. Na takav način, korisnik ima opciju prilagoditi iznose parametara pojedine vježbe prije spremanja podataka. Progresivno opterećenje se izvodi unutar pogleda, odnosno ViewModel-a, da bi fragment za tjelovježbu imao što manje odgovornosti. Korištenjem for petlje za svaku vježbu u listi vježbi za trenutnu tjelovježbu, za svaku seriju se provjeravaju ponavljanja i ako su ona izvan optimalnih granica, postavljaju se na vrijednost 8 te se dodaje veća kilaža. Na slici 4.31 je prikazana metoda zaslužna za progresivno opterećenje.

Sl. 4.31. Prikaz metode ProgressiveOverload()

4.4. Prikaz i spremanje promjena plana trenutne tjelovježbe

4.4.1. Prikaz trenutne tjelovježbe

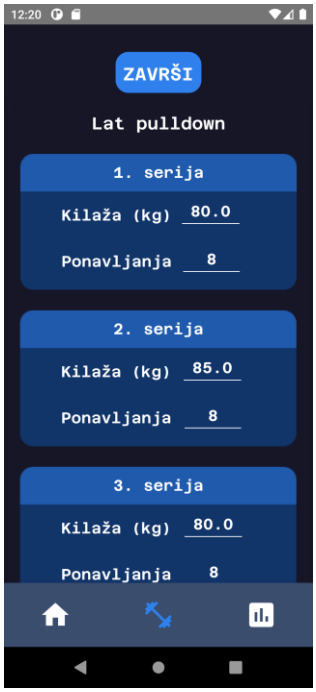
Za prikaz vježbi korištena je komponenta RecyclerView [26] koja dinamički kreira elemente kada su oni potrebni, te se općenito koristi za prikaz veće količine elemenata uz vertikalno pomicanje između elemenata. Ime je dobila po tome što reciklira te pojedine elemente, kada element dođe na poziciju izvan zaslona, neće biti uništen već će njegov izgled biti iskorišten za nove elemente koji su se pojavili na zaslonu. RecyclerView koristi adapter za predstavljanje elemenata, odnosno podataka koje želimo prikazati kao listu. Prikaz koda potrebnog za realizaciju RecyclerView komponente je na slici 4.32.



```
override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
    when(holder){
        is MyViewHolder -> {
            holder.bind(listOfExercises[position],position)
        }
    }
    holder.itemView.findViewById<EditText>(R.id.et_weightSetOne).addTextChangedListener {...}
    holder.itemView.findViewById<EditText>(R.id.et_weightSetTwo).addTextChangedListener {...}
    holder.itemView.findViewById<EditText>(R.id.et_weightSetThree).addTextChangedListener {...}
    holder.itemView.findViewById<EditText>(R.id.et_exerciseRepetitions).addTextChangedListener {...}
    holder.itemView.findViewById<EditText>(R.id.et_exerciseRepetitions2).addTextChangedListener {...}
    holder.itemView.findViewById<EditText>(R.id.et_exerciseRepetitions3).addTextChangedListener {...}
}

override fun getItemCount(): Int {
    return items.size
}
fun postItemsList(data: ArrayList<ExerciseWorkout>){
    items=data
    listOfExercises=data
}
inner class MyViewHolder(itemView: View) :
    RecyclerView.ViewHolder(itemView) {

    private val name: TextView =itemView.findViewById(R.id.tv_exerciseName)
    private val weight1: EditText =itemView.findViewById(R.id.et_weightSetOne)
    private val weight2: EditText =itemView.findViewById(R.id.et_weightSetTwo)
    private val weight3: EditText =itemView.findViewById(R.id.et_weightSetThree)
    private val repetitions1: EditText =itemView.findViewById(R.id.et_exerciseRepetitions)
    private val repetitions2: EditText =itemView.findViewById(R.id.et_exerciseRepetitions2)
    private val repetitions3: EditText =itemView.findViewById(R.id.et_exerciseRepetitions3)
    fun bind(data: ExerciseWorkout, position: Int){
        name.text=data.name
        weight1.setText(data.weightSetOne.toString())
        weight2.setText(data.weightSetTwo.toString())
        weight3.setText(data.weightSetThree.toString())
        repetitions1.setText(data.repetitionsSetOne.toString())
        repetitions2.setText(data.repetitionsSetTwo.toString())
        repetitions3.setText(data.repetitionsSetThree.toString())
    }
}
```

Sl. 4.32. Programski kod adaptera za RecyclerView

U fragmentu koji prikazuje trenutnu tjelovježbu, potrebno je iz pogleda dohvatiti listu vježbi za trenutnu tjelovježbu te ju prikazati korištenjem RecyclerView komponente. Prije nego što se adapteru preda lista vježbi, poziva se metoda za progresivno opterećenje. Izgled zaslona za prikazivanje trenutne tjelovježbe je prikazan na slici 4.33.



Sl. 4.33. Izgled zaslona za prikaz tjelovježbe

4.4.2. Spremanje odrađene izvedene tjelovježbe

Nakon što korisnik klikne na gumb „ZAVRŠI“, inkrementira se broj odrađenih tjelovježbi u pogledu. Kako bi se spremile promjene koje je korisnik unio tijekom izvođenja tjelovježbe, unutar adaptera je za svaki parametar vježbe pozvana metoda „addOnTextChangedListener“ koja nakon promjene vrijednosti vježbe ažurira njene vrijednosti u listi trenutnih vježbi. Dio koda koji je sličan za svaki element je prikazan na slici 4.34.

```
holder.itemView.findViewById<EditText>(R.id.et_weightSetOne).addTextChangedListener { it: Editable?
    listOfExercises[position].weightSetOne=it.toString().toDouble()
}
holder.itemView.findViewById<EditText>(R.id.et_weightSetTwo).addTextChangedListener { it: Editable?
    listOfExercises[position].weightSetTwo=it.toString().toDouble()
}
holder.itemView.findViewById<EditText>(R.id.et_weightSetThree).addTextChangedListener { it: Editable?
    listOfExercises[position].weightSetThree=it.toString().toDouble()
}
holder.itemView.findViewById<EditText>(R.id.et_exerciseRepetitions).addTextChangedListener { it: Editable?
    listOfExercises[position].repetitionsSetOne=it.toString().toInt()
}
holder.itemView.findViewById<EditText>(R.id.et_exerciseRepetitions2).addTextChangedListener { it: Editable?
    listOfExercises[position].repetitionsSetTwo=it.toString().toInt()
}
holder.itemView.findViewById<EditText>(R.id.et_exerciseRepetitions3).addTextChangedListener { it: Editable?
    listOfExercises[position].repetitionsSetThree=it.toString().toInt()
}
```

Sl. 4.34. Izvedba ažuriranja vrijednosti parametara vježbe

4.4.3. Prikaz povijest tjelovježbi

Prikaz povijesti tjelovježbi je izveden na sličan način kao i prikaz trenutne tjelovježbe, korištenjem RecyclerView komponente i prikladnog adaptera. Tjelovježbe su sortirane silaznim redoslijedom kako bi lakše vidio svoj napredak. Slika 4.35. prikazuje izgled zaslona povijesti tjelovježbi.

Sl. 4.35. Prikaz povijesti tjelovježbe

4.5. Postojeći algoritmi za predlaganje tjelovježbe

Ideja ovog rada nije rijetka, ali postoji malo rješenja na tržištu. Jedno od rješenja je ostvareno koristeći algoritam šišmiša (engl. bat algorithm) [27]. Njegova implementacija je vrlo jednostavna i efikasna. Bat algoritam je opisan kao inteligencija roja te se odnosi na kolektivno, novonastalo ponašanje višestrukih i međusobno povezanih agenata koji su sposobni obavljati jednostavne radnje. Iako se svaki agent može smatrati neinteligentnim, cijeli sustav višestrukih agenata pokazuje neke značajke samostalno organiziranog ponašanja, te se tako može ponašati kao neka vrsta kolektivne inteligencije. Ideja algoritma je dobivena inspiracijom iz pojave ekolokacije šišmiša.

Originalni algoritam šišmiša je modificiran tako da je krajnji rezultat plan tjelovježbe koji se sastoji od osam vježbi s odgovarajućim brojem ponavljanja i intenzitetom, odnosno kilažom. Algoritam je testiran i potvrđen od strane ljudskog trenera koji ima više od 20 godina iskustva.

5. Primjer korištenja aplikacije

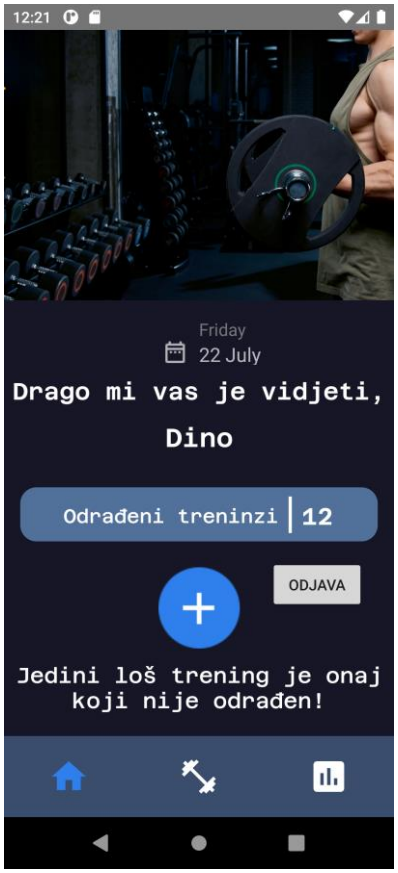
O ovom poglavlju je prikazan izgled gotove funkcionalne aplikacije, te je testirana svaka funkcionalnost.

5.1. Testiranje rada aplikacije

Aplikacija se smatra uspješnom ako pravilno predlaže tjelovježbu te ju sprema nakon što ju korisnik izvede. Također mora prikazivati povijest tjelovježbe silaznim redoslijedom. Zbog unosa početnih parametara prilikom registracije, potrebno je obratiti pozornost na obradu parametara, odnosno jesu li vrijednosti koje su postavljene u bazu ispravne.

Nakon kreiranja računa i unosa potrebnih parametara, aplikacija bi na početnom zaslonu trebala pokazivati ispravno ime korisnika, te bi broj odrađenih tjelovježbi trebao imati vrijednost 0. Zatim aplikacija bi kao prvu tjelovježbu trebala predložiti vježbe s dimenzijom pokreta „pull“ i vježbe s dimenzijom pokreta „legs“. Za provjeru progresivnog opterećenja treba u nasumično odabranu vježbu i njenu seriju upisati broj veći od gornje granice optimalnog broja ponavljanja. Zatim aplikacija treba ispravno spremiti tjelovježbu i prikazati ju u povijesti tjelovježbe. Kako aplikacija dinamički predlaže vježbe na osnovu povijesnih karakteristika potrebno je odraditi nekoliko tjelovježbi dok se ne pojavi vježba kojoj smo uredili broj ponavljanja. Ona bi trebala imati uvećanu vrijednost kilaže, a ponavljanja smanjenja na donju granicu optimalnog broja ponavljanja. Ako su svi uvjeti zadovoljeni, aplikacija radi ispravno.

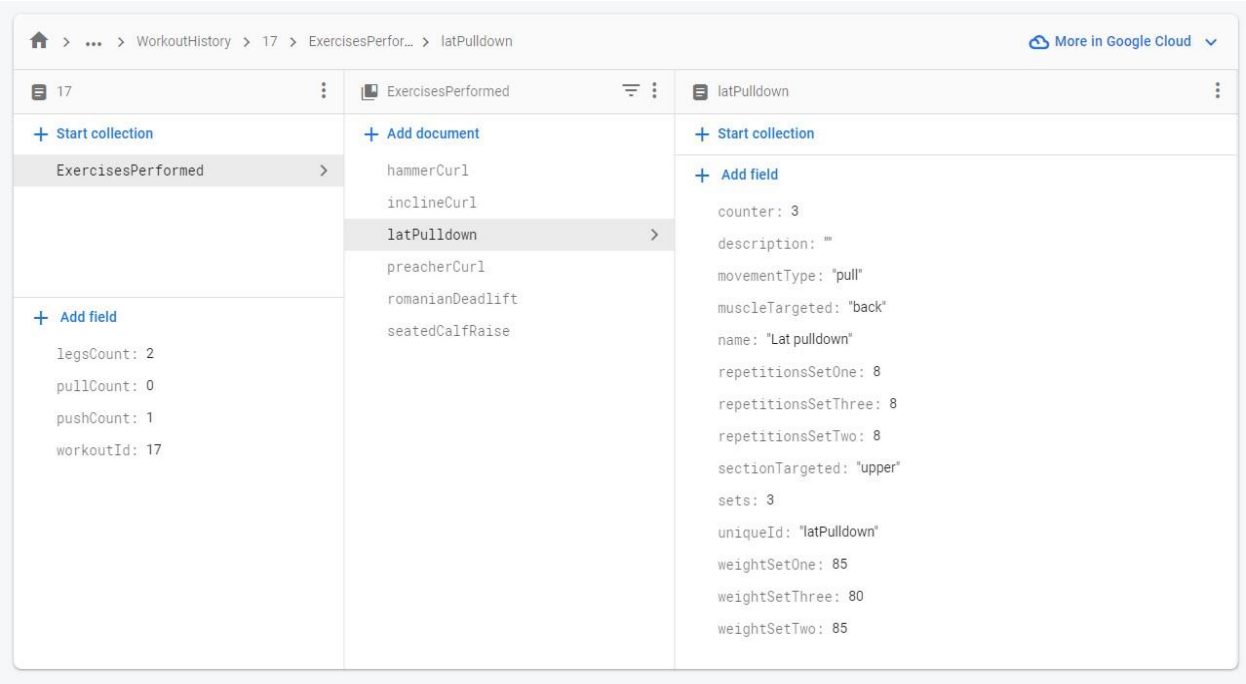
Pravilno prikazivanje imena korisnika i brojač odrađenih tjelovježbi je prikazano na slici 5.1.



Sl. 5.1. Prikaz rada brojača tjelovježbi i imena korisnika

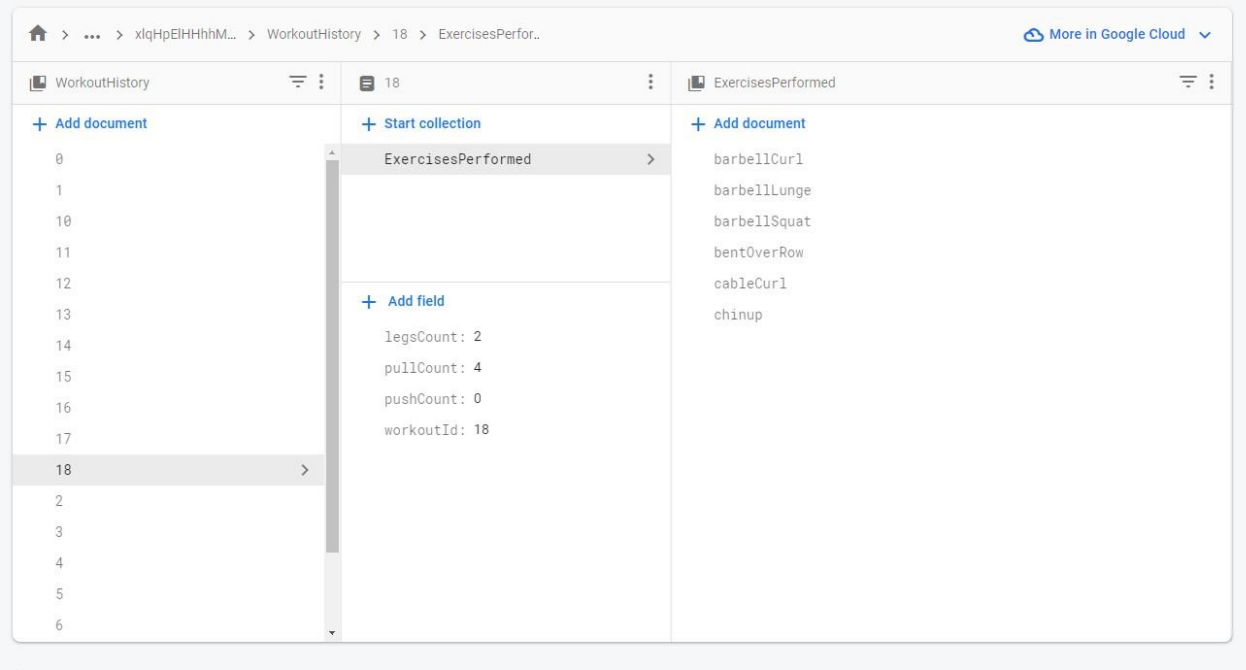
Za potrebe testiranja, vježbi „lat pulldown“ je u prvoj seriji broj ponavljanja postavljen na 13, te se pri sljedećoj tjelovježbi zadovoljava kriterij progresivnog opterećenja. Drugoj seriji je kilaža postavljena na 85, da bi se dodatno provjerila ispravnost spremanja podataka. Postavljanje broja ponavljanja je prikazano na slici 5.2, ažurirana vježba u bazi podataka je prikazana na slici 5.3.

Sl. 5.2. Postavljanje novih vrijednosti vježbe



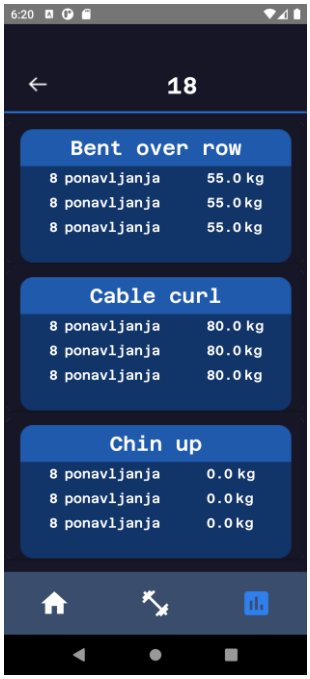
Sl. 5.3. Ažurirana vježba u bazi podataka

Za testiranje prikazivanja povijesti tjelovježbe, jedna tjelovježba je odrađena i spremljena u bazu, zatim se uspoređuje podudarnost vježbi na zaslonu i bazi podataka. Na slici 5.4 je prikazana tjelovježba u bazi podataka, njen prikaz na zaslonu povijest tjelovježbe je na slikama 5.5 i 5.6.



Sl. 5.4. Tjelovježba u bazi podataka

Sl. 5.5. Prikaz tjelovježbe na zaslonu



Sl. 5.6. Prikaz tjelovježbe na zaslonu

Nakon testiranja, utvrđeno je da aplikacija radi ispravno.

6. Zaključak

U ovom završnom radu razvijena je i opisana implementacija mobilne aplikacije za Android platformu koja ima svrhu praćenja i preporučivanja tjelovježbe. Korištenjem programskog jezika Kotlin uz platformu Firebase aplikacija je moderna i osigurana je njena podrška u budućnosti. Korisnik se mora registrirati i unijeti tražene parametre pomoću kojih će dobiti personaliziranu preporuku tjelovježbe koja se izvodi dinamički te sadrži progresivno opterećenje za konstantan napredak korisnika.

Dulji rad na aplikaciji će pomoći da aplikacija bude konkurentna na tržištu, ali će također biti koristan za usavršavanje znanja u izradi aplikacija za Android platformu.

Prilikom testiranja je utvrđeno nekoliko nedostataka što je normalno za svaku aplikaciju. Također je moguće izmijeniti neke funkcionalnosti i dodati nove koje su predviđene tijekom izvedbe projekta. Prvobitno zamišljene funkcionalnosti su ostvarene, dok za dodatne nije bilo mogućnosti zbog roka izrade završnog rada.

Glavni nedostatak aplikacije je vidljiv pad performansi nakon pokretanja početnog zaslona i zaslona tjelovježbe jer aplikacija šalje zahtjev putem interneta kako bi pristupila podacima. Problem se može riješiti naprednijim korištenjem Kotlin korutina. Drugi način pristupa rješavanja problema bi bio korištenje lokalne, vlastite baze podataka koja nema ograničenja poput Firebase-a, već je potpuno prilagođena aplikaciji. Zbog obaveznog korištenja interneta, korisnik je ograničen koristiti aplikaciju samo kada je povezan s internetom što je umanjilo zamišljenu praktičnost.

Za bolje korisničko iskustvo, uz svaku pojedinu vježbu bi trebala biti prikazana skica izvođenja vježbe. Predviđena funkcionalnost koja bi također pridonijela boljem korisničkom iskustvu su upute izvođenja vježbi. Parametar za svaku funkciju već postoji u bazi pa bi ju bilo vrlo jednostavno izvesti u budućnosti.

LITERATURA

[1] D. E. R. Warburton i S. S. D. Bredin, „Health benefits of physical activity: a systematic review of current systematic reviews“, *Curr. Opin. Cardiol.*, sv. 32, izd. 5, str. 541–556, ruj. 2017, doi: 10.1097/HCO.0000000000000437.

[2] Hrvatski zavod za javno zdravstvo, „ZDRAVLJE SRCA I TJELESNA AKTIVNOST“. https://www.hzjz.hr/sluzba-epidemiologija-prevencija-nezaraznih-bolesti/zdravlje-srca-i-tjelesna-aktivnost/ (pristupljeno 11. srpanj 2022.).

[3] S. T. Paluska SA, „Paluska, S A, and T L Schwenk. “Physical activity and mental health: current concepts.” Sports medicine (Auckland, N.Z.) vol. 29,3 (2000): Pages 167-80.“, ožujak 2000.

[4] „Svilar, Luka, et al. ‚TJELESNA AKTIVNOST KAO LIJEK U FUNKCIJI ZDRAVLJA.‘ Hrana u zdravlju i bolesti, vol. Specijalno izdanje, br. Štamparovi dani, 2015, str. 19-22. https://hrcak.srce.hr/157099 (pristupljeno 11.srpanj 2022.).“

[5] „Berger BG. Coping With Stress: The Effectiveness of Exercise and Other Techniques. Laramie, WY: Quest, 1994. (pristupljeno 11.srpanj 2022.).“

[6] D. Jurakić, „TJELESNA NEAKTIVNOST – JAVNOZDRAVSTVENI PRIORITET DANAŠNJICE?“, *Hrana U Zdr. Boles. Znan.-Stručni Časopis Za Nutr. Dijetetiku*, sv. Specijalno izdanje, izd. Štamparovi dani, str. 9–9, pros. 2015.

[7] I. Vuori, „Tjelesna neaktivnost je uzrok, a tjelesna aktivnost lijek za glavne javnozdravstvene probleme“, *Kinesiology*, sv. 36, izd. 2., str. 123–153, pros. 2004.

[8] Daily Strength, „Gym Workout Planner & Tracker, Aplikacije na Google Playu“. https://play.google.com/store/apps/details?id=com.anthonyng.workoutapp&hl=hr&gl=US (pristupljeno 11. srpanj 2022.).

[9] BALANCED, „Gym Workout Planner & Tracker, Aplikacije na Google Playu“. https://play.google.com/store/apps/details?id=com.anthonyng.workoutapp&hl=hr&gl=US (pristupljeno 11. srpanj 2022.).

[10] Inc. Nike, „Nike Training Club: Fitness, Aplikacije na Google Playu“. https://play.google.com/store/apps/details?id=com.exemple.user.crewlettgymapp&hl=hr&gl=US (pristupljeno 11. srpanj 2022.).

[11] „Firestore | Firebase“. https://firebase.google.com/docs/firestore/ (pristupljeno 13. srpanj 2022.).

[12] „Firebase Authentication“, *Firebase*. https://firebase.google.com/docs/auth (pristupljeno 13. srpanj 2022.).

[13] N. I. Mohamad, J. Cronin, i K. Nosaka, „Brief Review: Maximizing Hypertrophic Adaptation—Possible Contributions of Aerobic Exercise in the Interset Rest Period“, *Strength Cond. J.*, sv. 34, izd. 1, str. 8–15, velj. 2012, doi: 10.1519/SSC.0b013e3182308969.

[14] B. S. published, „What is progressive overload?“, *livescience.com*, 26. svibanj 2022. https://www.livescience.com/what-is-progressive-overload (pristupljeno 13. srpanj 2022.).

[15] „Figma: Revenue, Competitors, Alternatives“. https://growjo.com/company/Figma#what-is (pristupljeno 14. srpanj 2022.).

[16] „Kotlin Programming Language“, *Kotlin*. https://kotlinlang.org/ (pristupljeno 14. kolovoz 2022.).

[17] „Meet Android Studio | Android Developers“. https://developer.android.com/studio/intro (pristupljeno 14. kolovoz 2022.).

[18] „IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains“, *JetBrains*. https://www.jetbrains.com/idea/ (pristupljeno 13. srpanj 2022.).

[19] „Firebase“. https://firebase.google.com/ (pristupljeno 13. srpanj 2022.).

[20] „XML Introduction“. https://www.w3schools.com/xml/xml_whatis.asp (pristupljeno 13. srpanj 2022.).

[21] „GIMP“, *GIMP*. https://www.gimp.org/ (pristupljeno 13. srpanj 2022.).
[22] „Processes and threads overview“, *Android Developers*. https://developer.android.com/guide/components/processes-and-threads (pristupljeno 17. srpanj 2022.).
[23] „Coroutines | Kotlin“, *Kotlin Help*. https://kotlinlang.org/docs/coroutines-overview.html (pristupljeno 17. srpanj 2022.).
[24] khore, „What Are Compound Exercises? | AFA Blog“, *Australian Fitness Academy*, 31. listopad 2018. https://www.fitnesseducation.edu.au/blog/education/what-are-compound-exercises/ (pristupljeno 21. srpanj 2022.).
[25] „ViewModel overview“, *Android Developers*. https://developer.android.com/topic/libraries/architecture/viewmodel (pristupljeno 18. srpanj 2022.).
[26] „Create dynamic lists with RecyclerView“, *Android Developers*. https://developer.android.com/guide/topics/ui/layout/recyclerview
[27] „Fister, Iztok & Rauter, Samo & Fister, Karin & Fister, Dušan & Fister jr, Iztok. (2015). Planning fitness training sessions using the bat algorithm. CEUR Workshop Proceedings. 1422. 121-126.“

SAŽETAK

U ovom radu razvijena je Android aplikacija za praćenje i preporuku tjelovježbi. Aplikacija služi za jednostavno zabilježavanje tjelovježbe tako da se vježbe preporučuju dinamički, te se njihovi parametri prilagođuju kako bi se osiguralo progresivno opterećenje potrebno za napredak korisnika. Korisnik se mora registrirati i unijeti osnovne parametre pomoću kojih se stvaraju preporuke tjelovježbi. Svaka vježba sadrži naziv, te radnu kilažu i broj ponavljana za svaku pojedinu seriju koje korisnik može urediti. Aplikacija vodi bilješke o izvedenim tjelovježbama kako bi korisnik mogao vidjeti svoj napredak.

Ključne riječi: android, firebase, mobilna aplikacija, tjelovježba, zdravlje,.

ABSTRACT

Virtual Personal Trainer Android App

In this final paper, an Android application was developed for monitoring and recommending exercises. The application serves to easily record the exercise in such a way that the exercises are recommended dynamically, and their parameters are adjusted to ensure the progressive load necessary for the user's progress. The user must register and enter the basic parameters with which exercise recommendations are created. Each exercise contains a name, working weight and number of repetitions for each series, which the user can edit. The application keeps notes on the exercises performed so that the user can see his progress.

Keywords: android, exercise, firebase, health, mobile app.

ŽIVOTOPIS

Dino Knežević rođen je 13. srpnja 2000. godine u Našicama. Pohađao je osnovnu školu u Zdencima i školi Ivana Gorana Kovačića u razdoblju od 2007. do 2015 godine. Zatim upisuje smjer tehničar za elektroniku u Srednjoj školi Isidora Kršnjavoga u Našicama koju pohađa od 2015. do 2019. godine. Nakon završene srednje škole upisuje sveučilišni preddiplomski studij računarstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku 2019. godine.

Vlastoručni potpis:

_____

Dino Knežević