УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Милан Ајдер

Мобилна апликација за управљање тренинзима снаге

Дипломски рад

- Основне академске студије -

Нови Сад, 2023.

	УНИВЕРЗИТЕТ У НОВОМ САДУ **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Датум:
	ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ (BACHELOR) РАДА	Лист: 1/1

(Податке уноси предметни наставник - ментор)

Врста студија:	**Основне академске студије**
Студијски програм:	**Софтверско инжењерство и информационе технологије**
Руководилац студијског програма:	**проф. др Мирослав Зарић (SW)**

Студент:	**Милан Ајдер**	Број индекса:	**SW 31/2019**
Област:	**Мобилно програмирање**		
Ментор:	**Проф. Горан Савић**		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА ДИПЛОМСКИ РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;
- литература

НАСЛОВ ДИПЛОМСКОГ (BACHELOR) РАДА:

Мобилна апликација за управљање тренинзима снаге

ТЕКСТ ЗАДАТКА:

Задатак рада представља развој андроид апликације за управљање тренинзима снаге и повезивање корисника апликације. Апликација треба да буде развијана у Kotlin програмском језику, користећи MVVM архитектурни образац са Clean Architecture принципима. Кориснички интерфејс апликације треба да буде развијан помоћу Jetpack Compose алата.

Руководилац студијског програма:	Ментор рада:

Примерак за: ☐ - Студента; ☐ - Ментора

САДРЖАЈ

1. УВОД

Савремени начин живота и недостатак физичке активности представљају значајане здравствене ризике савременог човека. Студије показују да прекомерно седење може довести до проблема са циркулацијом, метаболизмом и општим физичким здрављем. Развој апликације која промовише активни начин живота и физичку рекреацију има потенцијал да значајно допринесе смањењу ризика од ових здравствених проблема [18].

Додатно, апликација омогућава развијање позитивног ривалства и међусобног подстицаја. Пратећи активности и постигнућа других корисника, корисник се мотивише да и сам узме учешће у физичкој активности. Ово је важно посебно у условима када друштвено-физичке интеракције могу бити ограничене, како је то данас често случај. Виртуелна подршка и подстицај од стране заједнице може значајно подигнути мотивацију и ангажман корисника.

Решење које је примењено на овај проблем засновано је на напредним *Android* [1] технологијама, укључујући *Kotlin* програмски језик [2], *Jetpack Compose* [3] за дизајн интерфејса и архитектурни модел *MVVM* (*Model-View-ViewModel*) [4] са *Clean Architecture* [5] принципима за ефикасно управљање подацима и корисничким интерфејсом. Додатно, за складиштење података и аутентикацију корисника је коришћен *Firebase* [6].

Размотрићемо постојеће апликације које имају сличну природу и циљеве као приказано решење. Основни критеријуми које су коришћени при избору ових апликација обухватају следеће аспекте:

Друштвене интеракције: Истраживане су апликације које омогућавају корисницима да се повежу, комуницирају и деле активности са другим корисницима.

Тренинг и фитнес функционалности: Анализиране су апликације које пружају садржај и алатке за следење тренинга, вежбања и физичких активности.

Мотивација: Претраживане су апликације које укључују елементе који подстичу и мотивишу кориснике на активност и учешће.

На основу наведених критеријума, издвојено је неколико апликација које деле сличности са овим решењем. У наставку, је приказан детаљан преглед и анализа сваке од ових апликација, истичући њихове добре и мање добре стране у контексту проблема који се решава.

Fitbod [19]: Апликација *Fitbod* фокусира се на персонализоване тренинг планове за вежбаче. Омогућава корисницима да унесу своје циљеве, опрему коју имају на располагању и ниво физичке припремљености. Путем алгоритама, апликација генерише тренинг план подешен за личне потребе. Добра страна овог решења је персонализација тренинга, међутим, ограничена је могућност за друштвену интеракцију и дељење активности.

Sworkit [20]: *Sworkit* је апликација која нуди брзе и ефикасне вежбе које корисници могу радити где год се налазе. Поседује готове тренинге за различите циљеве и стилове тренинга. Предност Sworkit-а је доступност тренинга у сваком тренутку, међутим, ограничена могућност персонализације и друштвене интеракције.

Strava [21]: *Strava* је апликација која нуди опцију праћења активности као што су трчање, вожња бицикла и сл. Омогућава корисницима да прате своје руте, брзину, удаљеност и друге статистике. За разлику од претходно наведених апликација, *Strava* има и друштвену компоненту, кроз коју омогућава корисницима дељење активности са заједницом. Одлична страна Strava-е је фокус на фитнес активностима и подстичање друштвене интеракције, међутим, она не покрива могућност за тренинге у теретани и вежбе снаге.

Instagram [22] и ***Tinder*** [23]: У контексту друштвених мрежа, *Instagram* обухвата разноврстан скуп функционалности које поседују и друге популарне платформе (нпр. дељење слика и статуса, затим *reels, stories* и других формата). *Tinder*, с друге стране, има специфичан концепт спајања корисника на основу међусобних интересова. Иако су ове апликације широко коришћене и успешне, важно је напоменути да њихов фокус није усмерен на тренинг и физичку рекреацију, што представља један од централних аспеката овог решења.

Оно што ово решење чини јединственим јесте синтеза друштвене мреже, платформе за упознавање и тренинга снаге на савременом дигиталном интерфејсу. Ова интеграција ствара иновативно решење које се издваја из понуде сличних апликација.

2. Коришћене софтверске технологије

Ово поглавље представља технологије које су коришћене за развој ове мобилне апликације, како би се боље разумела сама имплементација.

У поглављу 2.1 описан је програмски језик *Kotlin* са акцентом на андроид програмирање, док је у поглављу 2.2 описан Jetpack Compose, модерна библиотека за израду корисничког интерфејса андроид апликација. У поглављу 2.3 описан је *Firebase* и сервиси који су коришћени у овом пројекту. У последњем поглављу 2.4 описани су *plugin*-и и библиотеке које су коришћене у овом пројекту.

2.1 Kotlin

Програмски језик *Kotlin* је савремени, општи наменски програмски језик који је развијен од стране компаније *JetBrains*. Он је дизајниран да буде интероперабилан са *Java* платформом [7] и да омогући брз и сигуран развоју апликација. Котлин се користи за развој великог спектра апликација, укључујући веб, мобилне, *desktop* и серверске апликације.

Kotlin се истиче јасним и статички типизираним програмским кодом. Статичка типизираност и *null safety* доприносе лакшем разумевању и одржавању кода и смањеној вероватноћи *runtime* грешака. *Kotlin* подржава концепте функционалног програмирања, садржи *coroutines* за асинхроно извршавање и има богату стандардну библиотеку. Издваја се као језик који пружа лакшу и ефикаснију синтаксу у односу на *Java* програмски језик, што доприноси бржем развоју апликација.

Један од кључних аспеката *Kotlin* програмског језика, који га чини погодним за андроид развој је интеграција са *Android Studio*, званичним развојним окружењем за андроид платформу. То омогућава програмерима да лако израде апликације користећи котлин и да искористе многе андроид-специфичне функције.

С обзиром на све ове предности, котлин је постао популаран избор за андроид развој. У решењу, *Kotlin* је коришћен као основни програмски језик због његове ефикасности, читљивости и добре интеграције са андроид окружењем**.**

2.2 Jetpack compose

Jetpack Compose [3] представља нови начин (од јула 2021.) израде корисничког интерфејса у андроид апликацијама. Ова модерна

библиотека замењује традиционални *XML* базиран интерфејс и омогућава декларативно дизајнирање корисничког интерфејса путем програмског кода

Главна предност *Jetpack Compose*-а лежи у његовом декларативном приступу. Уместо да се дефинишу хијерархије корисничких елемената као у *XML*-у, програмери описују како треба да изгледа интерфејс коришћењем *Kotlin* кода. Ово омогућава једноставније и читљивије дефинисање интерфејса, као и бржи циклус развоја.

Поред декларативности, *Jetpack Compose* такође нуди реактиван приступ. То значи да се кориснички интерфејс аутоматски ажурира када се стање података промени. Ово значајно олакшава рад са динамичким садржајем и приказом актуелних информација.

Компоненте у *Jetpack Compose*-у су поново употребљиве и могу се лако комбиновати за изградњу сложених интерфејса. Ово побољшава организацију кода и омогућава бржи развој апликација.

У решењу, *Jetpack Compose* је искоришћен за израду корисничког интерфејса апликације. Овај модеран приступ је допринео бржем и лакшем развоју интерфејса, као и омогућио јасно и декларативно дефинисање корисничких елемената.

2.3 Firebase

Firebase је платформа коју је развила компанија *Google* и која омогућава развој апликација уз помоћ различитих *cloud* услуга. У свом арсеналу, Firebase нуди низ алатки које значајно убрзавају процес развоја апликација.

Firestore је база података која омогућава у реалном времену синхронизован приступ подацима различитим уређајима. Ова технологија је корисна за апликације које имају потребу за ажурирањем података у реалном временз.

Authentication (аутентификација) омогућава једноставан и сигуран начин за проверу и управљање идентитетом корисника. Користећи *Firebase Authentication*, апликација може пружити могућност пријаве и регистрације корисника путем различитих метода, као што су корисничко име и лозинка, *Google* налог или путем налога на друштвеним мрежама.

Storage (складиште) омогућава апликацији да сачува и управља датотекама, као што су слике и видео или аудио снимци.

Messaging (поруке) омогућава апликацији да пружи услуге обавештења (*Push notifications*) и упозорења корисницима. Коришћењем *Firebase Cloud Messaging*, апликација може ефикасно

комуницирати са корисницима и обавештавати их о новим активностима, порукама или другим подешавањима.

У пројекту, коришћењем *Firebase* технологија, омогућено је лако управљање подацима (текстом и сликама) и аутентификација корисника.

2.4 Plugin компоненте и библиотеке

У *app* и *project build.gradle* фајловима се налазе зависности и подешавања које се односе на апликацију. У овом поглављу биће више речи о *plugin*-има и најважнијим библиотекама које су коришћене у пројекту.

2.4.1 Plugin компоненте

У овом поглављу су описане најзначајније *plugin* копоненте за израду овог пројекта.

Компонента *com.android.application* омогућава дефинисање пројекта као андроид апликације. Користи се за *build* андроид апликација и управљање ресурсима, зависностима и конфигурацијама.

Компонента *kotlin-android* омогућава коришћење *Kotlin* програмског језика за развој андроид апликације.

Компонента *kotlin-kapt* омогућава коришћење анотација у *Kotlin* порјектима (*KAPT - Kotlin Annotation Processing Tool*). Користи се за генерисање додатног кода на основу анотација које су додате у коду.

Компонента *dagger.hilt.android.plugin* је неопходна за интеграцију *Dagger Hilt* библиотеке у андроид апликацију. *Dagger Hilt* олакшава инјекцију зависности у пројекту, што помаже у бољој организацији кода и упрошћава тестирање.

Компонента *com.google.gms.google-services* се користи за интеграцију *Google Play Services* у апликацију. Он омогућава интеграцију са различитим *Google* сервисима, као што су *Google Analytics, Google Maps* и други.

Компонента *kotlin-parcelize* омогућава коришћење анотације *@Parcelize* за аутоматску серијализацију и десеријализацију *Kotlin* класа. Ова анотација олакшава рад са *Parcelable* објектима, који се користе за пренос података између екрана.

2.4.2 Најзначајније библиотеке

У овом поглављу су описане најзначајније бибилиотеке за израду овог пројекта.

Библиотека *androidx.core:core-ktx* садржи колекцију корисних екстензија (*Kotlin extensions*) које упрошћавају рад са основним функционалностима андроид платформе. На пример, омогућава коришћење функција за рад са *SharedPreferences*, *Resources*, *Bundle*, датумима и сл.

Библиотека *androidx.compose.ui:ui* је део *Android Jetpack Compose* фрејмворка за израду корисничког интерфејса. Садржи основне елементе за израду корисничког интерфејса.

Библиотека *androidx.compose.material:material* садржи имплементацију *Material Design* елемената за *Compose* окружење. Омогућава коришћење стандардних компоненти, као и прилагођавање и измену њихових стилова.

Библиотека *androidx.compose.ui:ui-tooling-preview* садржи различите алате и помоћне функције за убрзан развој и преглед *Compose* корисничног интерфејса. Омогућава бржи дизајн и проналажење грешака, без потребе за покретањем апликације.

Библиотека *androidx.navigation:navigation-compose* омогућава навигацију између екрана у *Compose* апликацији. Пружа декларативни начин дефинисања путања до екрана и прелаза између различитих екрана.

Библиотека *com.google.accompanist:accompanist-swiperefresh* обезбеђује *SwipeRefresh* компоненту која омогућава освежавање садржаја користећи полачење на доле. Ово је честа компонента у апликацијама које приказују спискове података.

Библиотека *org.jetbrains.kotlinx:kotlinx-coroutines-core* садржи основне класе и функције за рад са *Kotlin coroutines*. Омогућава асинхрону и конкурентну обраду задатака на ефикасан начин.

Библиотека *org.jetbrains.kotlinx:kotlinx-coroutines-android* обезбеђује проширења за андроид платформу у оквиру *Kotlin coroutines*. Омогућава манипулацију корисничким интерфејсом и рад са главним нитима.

Библиотека *androidx.lifecycle:lifecycle-viewmodel-ktx* обезбеђује проширење за *AndroidX Lifecycle ViewModel*. Омогућава директан приступ ViewModel-у из активности или фрагмента користећи *Kotlin coroutines*.

Библиотека *androidx.lifecycle:lifecycle-runtime-ktx* обезбеђује проширење за *AndroidX Lifecycle* компоненту. Омогућава упрошћени начин рада са животним циклусом компоненти користећи *Kotlin coroutines*.

Библиотека *com.google.dagger:hilt-android* обезбеђује инфраструктуру за инјекцију зависности у андроид апликације. Чини инјекцију знатно једноставнијом и олакшава организацију кода.

Библиотека *com.squareup.retrofit2:retrofit* омогућава лако прављење HTTP захтева. Користи се за комуникацију са удаљеним API-има.

Библиотека *com.squareup.retrofit2:converter-gson* пружа конвертер за *Gson* који омогућава *Retrofit*-у да аутоматски парсира *JSON* одговоре у одговарајуће моделе.

За подешавање верзија *Firebase* зависности (зависности обезбеђене од стране *Firebase* тима) коришћен је *implementation platform('com.google.firebase:firebase-bom:31.2.0')*.

Библиотека *com.google.firebase:firebase-auth-ktx* пружа аутентикацију корисника преко *Firebase*.

Библиотека *com.google.firebase:firebase-firestore-ktx* омогућава коришћење *Firebase Cloud Firestore* базе података.

Библиотека *com.google.firebase:firebase-storage-ktx* омогућава слање и примање датотека кроз *Firebase Cloud Storage*.

Библиотека *com.google.firebase:firebase-messaging-ktx* пружа функционалности за праћење порука (*push* обавештења) између апликације и *Firebase* сервера.

Библиотека *io.coil-kt:coil-compose* омогућава ефикасно учитавање слика у *Compose* апликацији.

Библиотека *io.coil-kt:coil-gif* омогућава подршку за учитавање и приказивање *GIF* слика у *Compose* апликацији.

Библиотека *io.coil-kt:coil-svg* омогућава подршку за учитавање и приказивање *SVG* слика у *Compose* апликацији.

Библиотека *androidx.compose.material:material-icons-extended* садржи додатне иконице базиране на *Material Design*-у

Библиотека *com.google.android.gms:play-services-auth* омогућава имплементацију аутентикације помоћу *Google Play Services*.

Библиотека *androidx.room:room-runtime* је библиотека за рад са локалном базом података. Ова библиотека пружа *SQL* базирани приступ и управљање подацима у апликацији.

Библиотека *com.alexstyl.swipeablecard:*swipeablecard пружа компоненту за картице које се могу превлачити (*swipeable cards*), често коришћене у апликацијама за упознавање.

Библиотека *com.patrykandpatrick.vico:*core пружа функционалност за израду графикона и дијаграма у *Compose*.

Библиотека *androidx.media:media* пружа функционалности за рад са медијима, као што су аудио и видео.

Библиотека *com.google.android.gms:play-services-location* омогућава приступ информацијама о локацији помоћу *Google Play Services*.

3. СПЕЦИФИКАЦИЈА

3.1 Спецификација захтева

У овом поглављу су описани функционални и нефункционални захтеви које је потребно систем да омогући.

3.1.1 Функционални захтеви

У апликацији постоји само један тип корисника - корисник. Корисник може бити пријављен и непријављен.

Приликом уласка у апликацију, корисник има могућност регистрације или пријаве на систем. Регистрацију и пријаву на систем може да изврши или попуњавањем одговарајућих поља или путем пријаве преко *Google* налога.

Корисник има увид у свој налог. Може да мења своје личне податке попут висине, тежине, годишта.

Корисник има опцију креирања новог шаблона вежбања. Шаблон вежбања садржи информације о тренингу као што су вежбе које су у склопу тренинга, колико понављања и серија вежба садржи и колико дуга треба да буде пауза између серија. Такође корисник има опцију прегледа, измене и брисања својих шаблона за вежбање. Приликом одабира вежби за шаблон, корисник може да филтрира вежбе по називу, мишићној групи која је активирана током извођења вежбе или опреми која је неопходна за вежбу. За сваку вежбу корисник има визуелни приказ извођења вежбе.

Корисник се повезује са другим корисницима преко система превлачења картица на којима су слике других корисника. Уколико корисник жели да се повеже са другом особом треба да превуче картицу на десно, уколико не, треба да превуче картицу на лево. Када оба кориника изразе заинтересованост за другог корисника, они добијају опцију међусобног дописивања. Корисник може да подеси ограничење који тип корисника ће му бити предложен. Може да дефинише пол, године и максималну дистанцу у односу на другог корисника. Корисник може да претражује све повезане кориснике, да се дописује са њима и да види њихове профиле. Такође, корисник има опцију да шаље неки од својих шаблона за вежбање другим корисницима преко поруке, али и да сачува туђ шаблон за вежбање.

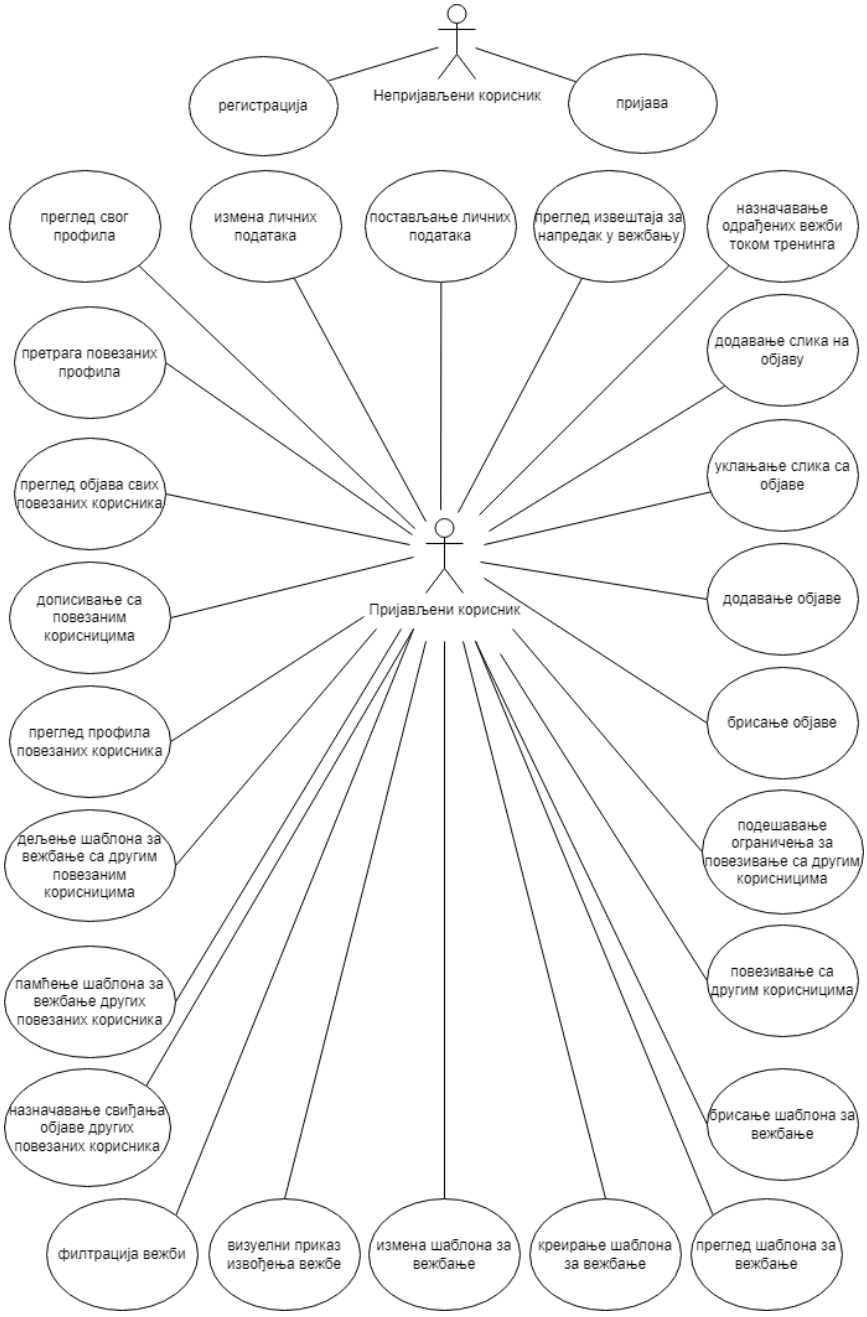
Након избора шаблона за вежбање, корисник може да покрене сам тренинг. Корисник може да назначи коју вежбу и серију је завршио.

Након сваке завршене серије, покреће се часовник који одбројава време до краја паузе и производи звучни ефекат након истека времена. Када је завршио све планиране вежбе, корисник може да назначи крај тренинга и тиме се чувају подаци о тренингу и креира се објава.

Корисник има опцију прегледа објава свих повезаних корисника на једном месту. Такође, може да прегледа све своје досадашње објаве. Објаве су приказане од новијих ка страријим. Објава се састоји из информација о завршеном тренингу. Корисник може додати или уклонити слике са тренинга, за објаву коју је он објавио. Такође, може и уклонити саму објаву. Поред тога, корисник може да назначи свиђање објава других повезаних корисника.

Корисник има опцију прегледа извештаја напретка у вежбању у претходном периоду, за себе и за повезане кориснике. Извештаји се могу правити за одређену групу мишића, вежбу, обиму и интензитету тренинга.

На слици (Слика 1) приказан је дијаграм случајева коришћења на ком су графички приказани функционални захтеви.



Слика 1 - Дијаграм случајева коришћења апликације за управљањем тренинзима

Претходно наведени случајеви коришћења детаљније су описани у табелама испод. Табела 4.1 садржи случајеве коришћења за непријављеног корисника, а табела 4.2 садржи случајеве коришћења за пријављеног корисника.

Назив	Опис	Предуслови
Регистрација	Регистрацију на систем може да изврши или попуњавањем одговарајућих поља или путем пријаве преко *Google* налога.	Корисник није тренутно пријављњен и е-пошта са којим се корисник региструје није у систему.
Пријава	Пријаву на систем може да изврши или попуњавањем одговарајућих поља или путем пријаве преко *Google* налога.	Корисник није тренутно пријављњен.

Табела 4.1 Опис случајева коришћења за непријављеног корисника

Назив	Опис	Предуслови
Преглед свог профила	Профилна слика, висина, тежина, годиште, објаве.	
Измена личних података	Може да мења своје личне податке попут висине, тежине, годишта, слике.	
Креирање шаблона за вежбање	Шаблон садржи вежбе које су у склопу тренинга, колико понављања и серија вежба садржи и колико дуга треба да буде пауза између серија.	
Преглед шаблона за вежбање	Преглед вежби, броја понављања, серија и пауза	

Измена шаблона за вежбање	Измена вежби, броја понављања, серија и пауза	
Брисање шаблона за вежбање	Брисање својих шаблона за вежбање	
Филтрација вежби	Приликом одабира вежби за шаблон, корисник може да филтрира вежбе по називу, мишићној групи која је активирана током извођења вежбе или опреми која је неопходна за вежбу.	
Визуелни приказ извођења вежбе	За сваку вежбу као објашњење.	
Повезивање са другим корисницима	Корисник се повезује са другим корисницима преко система превлачења картица на којима су слике других корисника. Уколико корисник жели да се повеже са другом особом треба да превуче картицу на десно, уколико не, треба да превуче картицу на лево. Када оба кориника изразе заинтересованост за другог корисника, они добијају опцију међусобног дописивања.	Кориснику се препоручују само корисници за које није раније назначио да му се свиђају или не свиђају.
Подешавање ограничења за повезивање са другим корисницима	Корисник може да подеси ограничење који тип корисника ће му бити предложен. Може да дефинише пол, године и максималну дистанцу у односу на другог корисника.	

Претрага повезаних профила	Претрага се врши само међу повезаним корисницима.	
Преглед профила повезаних корисника	Преглед личних података и објава повезаних корисника.	Могућ је преглед само за повезане кориснике.
Дописивање са повезаним корисницима	Слање порука повезаним корисницима.	Корисници морају бити повезани.
Дељење шаблона за вежбање са другим повезаним корисницима	Слање шаблона за вежбање повезаним корисницима.	Корисници морају бити повезани.
Памћење шаблона за вежбање других повезаних корисника	Када други корисник подели свој шаблон за вежбање са нама, можемо да га сачувамо међу својим шаблонима.	Корисници морају бити повезани.
Назначавање одрађених вежби током тренинга	Након избора шаблона за вежбање, корисник може да покрене тренинг. Корисник може да назначи коју вежбу и серију је завршио. Након сваке завршене серије, покреће се часовник који одбројава време до краја паузе и производи звучни ефекат након истека времена.	
Додавање објаве	Када је завршио све планиране вежбе, корисник може да назначи крај тренинга и тиме се чувају подаци о тренингу и креира се објава.	Тренинг је пре тога покренут.

Преглед објава свих повезаних корисника	Преглед објава повезаних корисника на једном месту, објаве су приказане од новијих ка страријим. Објава се састоји из информација о завршеном тренингу.	
Додавање слика на објаву	Слика са тренинга се може додати на своју објаву.	
Уклањање слика са објаве	Слика са тренинга се може уклонити са своје објаве.	
Брисање објаве	Корисник може обрисати своју објаву.	
Назначавање свиђања објаве других повезаних корисника	Корисник може да назначи да му се свиђа објава неког од њему повезаних корисника.	
Преглед извештаја за напредак у вежбању	Корисник има опцију прегледа извештаја напретка у вежбању у претходном периоду, за себе и за повезане кориснике. Извештаји се могу правити за одређену групу мишића, вежбу, обиму и интензитету тренинга.	
Постављање личних података	Корисник може да подеси своје личне податке (године, пол, висину и тежину) приликом првог пријављивања.	Уколико корисник није унео личне податке.

Табела 4.2 Опис случајева коришћења за пријављеног корисника

3.1.2 Нефункционални захтеви

У контексту наше андроид апликације, разматрају се неки аспекти који нису директно функционални захтеви, али имају значајан утицај на корисничко искуство и перформансе система.

Перформансе апликације су од кључног значаја. Систем треба да омогући брз одговор на корисничке акције, осигуравајући глатко коришћење без уских грла.

Корисничко искуство је приоритет. Интуитиван и лак за коришћење кориснички интерфејс са приступачном навигацијом треба да обезбеди лакшу интеракцију са апликацијом.

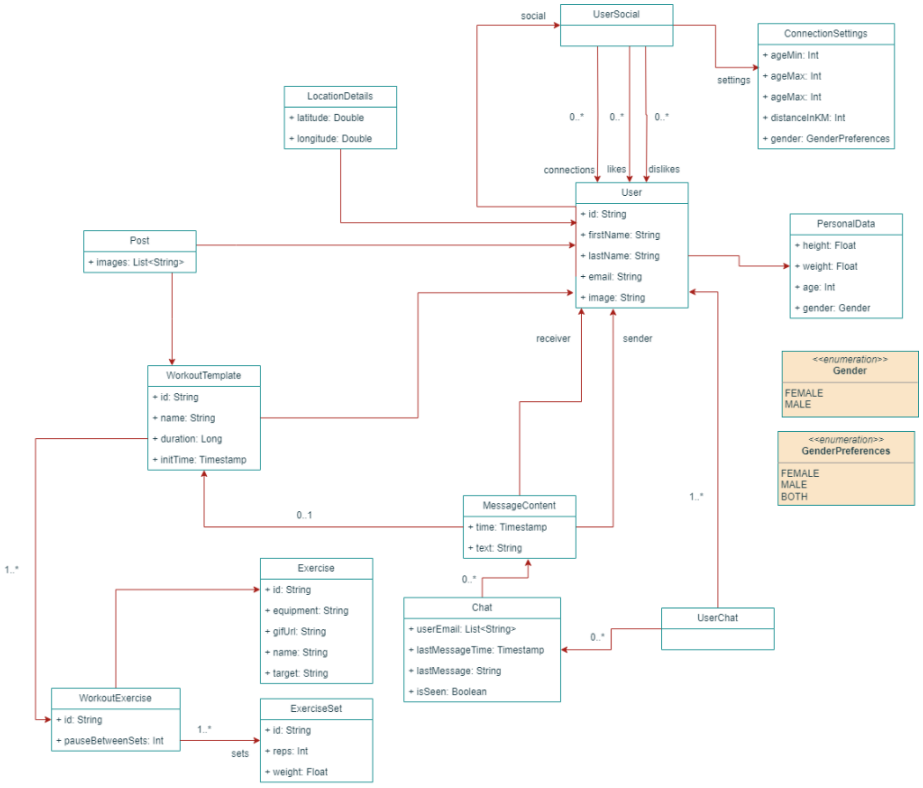
Респонзивни дизајн је од суштинског значаја. Дизајн треба да буде прилагођен различитим уређајима и величинама екрана како би се осигурала добра видљивост и коришћење.

3.2 Спецификација система

У овом поглављу је кроз дијаграм класа приказан модел података 3.2.1, а кроз дијаграм компоненти архитектура система 4.2.2.

3.2.1 Модел података

Класе, атрибути и везе између класа приказани су на слици (Слика 2) помоћу класног дијаграма.



Слика 2 - Модел података апликације за вежбање

Класа `WorkoutTemplate` представља шаблон за вежбање, садржи информације о његовом трајању (`duration`), власнику шаблона (`user`), вежбама које су саставни део шаблона (`workoutExercises`), и времену креирања (`initTime`). Атрибут user представља референцу на објекат класе `User`, а један шаблон може имати 1 или више вежби који су типа `WorkoutExercise`.

Класа `WorkoutExercise` описује вежбу у оквиру тренинга. Садржи информације о идентификационом броју (`id`), референцу на вежбу (типа `Exercise`), серијама (`exerciseSets`) и паузи између серија (`pauseBetweenSets`). Серија мора бити бар једна, а може их бити и више. Серија је типа `ExerciseSet`.

`User` класа представља корисника апликације и садржи основне информације о њему. Атрибути укључују идентификациони број (`id`), име (`firstName`), презиме (`lastName`), е-пошту (`email`), слику профила (`image`, путања до места где се слика складишти) и

социјалне везе (`userSocial`, типа `UserSocial`). Опционално, могу се навести и лични подаци корисника (`personalData` типа `PersonalData`).

Класа `UserSocial` моделира друштвене везе корисника, укључујући референце на пријатеље (`connections`), референце на кориснике који су нам се свидели (`likes`), референце на кориснике који нам се нису свидели (`dislikes`) и подешавања за повезивање (`settings`, инстанца класе `ConnectionSettings`).

Класа `Post` описује објаву, која укључује информације о шаблону тренинга (типа `WorkoutTemplate`), кориснику који је објавио (типа `User`) и сликама (листа путања до места где су складиштене слике).

Enum `Gender` дефинише различите родове, као што су мушки (`MALE`) и женски (`FEMALE`). Enum `GenderPreferences` дефинише преференце по роду приликом повезивања и додаје опцију за оба пола (`BOTH`).

Класа `PersonalData` садржи личне податке о кориснику, укључујући висину (`height`), тежину (`weight`), старост (`age`) и род (`gender`).

Класа `MessageContent` описује садржај поруке, укључујући информације о пошиљаоцу (`sender`), примаоцу (`receiver`), времену поруке (`time`), тексту поруке (`text`) и могућем шаблону тренинга (`workoutTemplate`, типа `WorkoutTemplate`). Уколико се не шаље шаблон кроз поруку, вредност `workoutTemplate` је `null`.

Класа `LocationDetails` садржи информације о локацији, као што су географске координате (`latitude`, `longitude`) и рефеденцу на корисника (`userEmail`).

Класа `ExerciseSet` моделира један сет вежбе, садржи информације о броју понављања (`reps`) и тежини (`weight`).

Класа `Exercise` представља одређену вежбу са информацијама о идентификационом броју (`id`), опреми (`equipment`), мишићној групи која се највише ангажује за дату вежбу (`target`) и називу (`name`).

Класа `ConnectSettings` детаљно описује подешавања за повезивање, укључујући минималну и максималну старост (`ageMin`, `ageMax`), максималну удаљеност (`distanceInKM`) и преферирани род (`gender`). Преферирани род је енумерација `GenderPreferences`.

Класа `UserChat` моделира сва дописивања једног корисника, укључујући референцу на корисника (`userEmail` који референцира на инстанцу `User` класе) и листу његових дописивања са другим

корисницима (`chats`). `Chats` је листа која садржи нула или више дописивања (tipa `Chat`)

Класа `Chat` описује дописивање два корисника, садржећи листу е-пошта учесника (`userEmails`), последњу поруку (`lastMessage`), време последње поруке (`lastMessageTime`), листу порука у чету (`messages`, типа `MessageContent`) и да ли је последња порука погледана од стране корисника (`isSeen`).

3.2.2 Архитектура система

MVVM (*Model-View-ViewModel*) је архитектонски образац који се често користи у андроид развоју. Ова архитектура има за циљ да организује код апликације на начин који омогућава јасно раздвајање одговорности и лакше управљање компонентама.

У *MVVM*-у, ***Model*** представља податке и пословну логику. ***View*** је кориснички интерфејс који приказује податке кориснику и прима корисничке акције. ***ViewModel*** служи као веза између *Model*-а и *View*-а. Он садржи бизнис логику која обрађује податке из *Model*-а и обезбеђује их за приказивање на *View*-у.

Чиста архитектура (***Clean Architecture***) се уклапа у *MVVM* и има за циљ да организује апликацију у слојеве, омогућавајући зависностима да теку у једном смеру. Ова архитектура подразумева три основна слоја: презентациони слој (***presentation layer***), доменски слој (***domain layer***) и слој података (***data layer***).

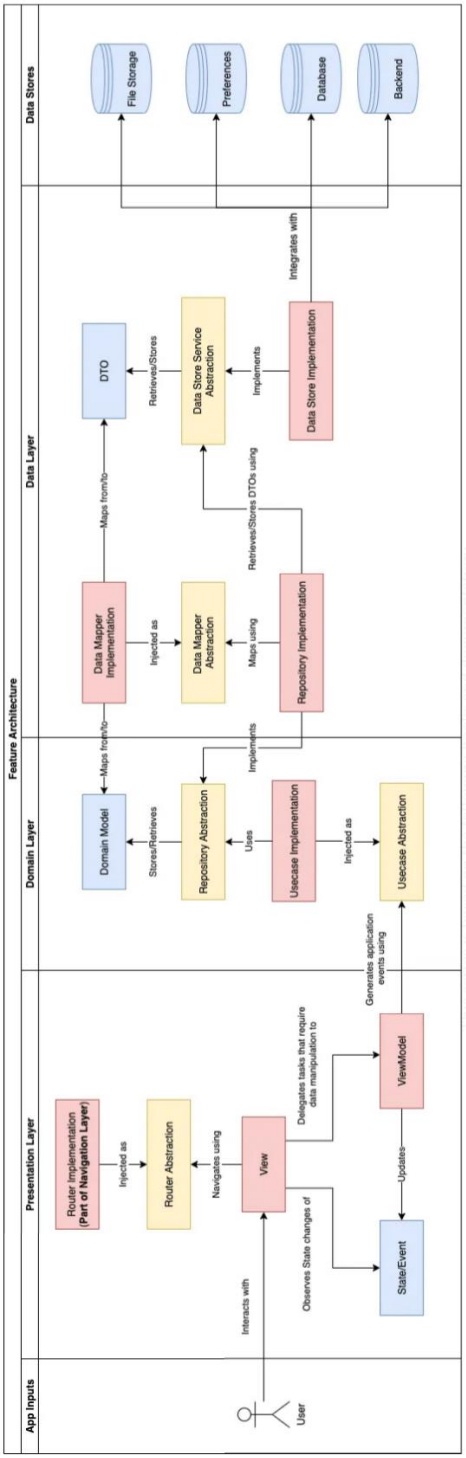
Презентациони слој је слој одговоран за интеракцију са корисницима апликације. Садржи *View* и *ViewModel* компоненте. *View* се бави приказом података кориснику и реагује на корисничке акције. *ViewModel* је мост између *View*-а и доменског слоја.

Доменски слој представља срж апликације и садржи бизнис логику. Употребом *UseCase* апстракција, *ViewModel* генерише догађаје који се приказују у корисничком интерфејсу. *UseCase* служи за изолацију доменске логике од слоја за презентацију. *UseCase* представља једну акцију коју корисник може да уради.

Слој података управља добијањем и складиштењем података. Садржи репозиторијуме и служи као изолација између доменског слоја и различитих извора података. Имплементација репозиторијума комуницира са *Data Mapper*-има за мапирање између доменских модела и *DTO*-а (*Data Transfer Objects*). *DataStoreService* апстракција служи за комуникацију са спољним изворима података, као што су база података мрежни сервиси или други извори пдатака. За те целине се користе библиотеке и технологије као што су *Room* за локалну базу података, *Retrofit* за комуникацију са мрежним сервисима и *Firebase* за спољне изворе података.

У пројекту, *MVVM* архитектурни образац са *Clean Architecture* принципима је примењен да би се допринело смањењу комплексности кода, побољшало управљање зависностима и олакшало развијање и одржавање апликације.

На слици (Слика 3) је представљен дијаграм компоненти у контексту Чисте архитектуре. [8]



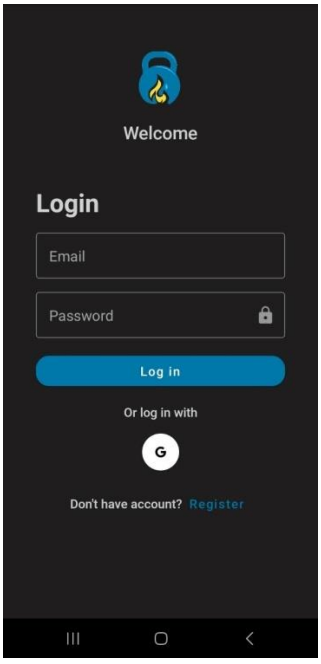
Слика 3 - Дијаграм компоненти Чисте архитектуре

4. ИМПЛЕМЕНТАЦИЈА И ПРИКАЗ АПЛИКАЦИЈЕ

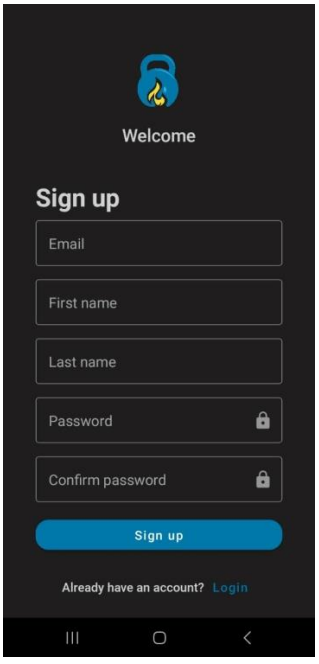
У овом поглављу је детаљно описана имплементација андроид апликације за управљање тренинзима снаге. Приказане су слике екрана и анализа кључних делова.

4.1 Пријава и регистрација

Приликом покретања апликације први пут, пред корисником се налази екран за пријаву. На екрану се налазе две опције за пријаву – помоћу е-поште и лозинке или помоћу *Google* налога (Слика 4). Уколико корисник нема од раније налог, може да се региструје кликом на дугме *Register* којим се редиректује на страницу за регистрацију или путем *Google* налога. Страница за регистрацију садржи поља за унос е-поште, имена, презимена, лозинке и потвде лозинке (Слика 5).



Слика 4 - Екран за пријаву



Слика 5 - Екран за регистрацију

4.1.1 Пријава

Кликом на дугме *Log in* позива се следећа функција (листинг 5.1) класе `AuthRepositoryImpl`:

```
override fun loginUser(email: String, password: String):
Flow<Resource<AuthResult>> {
    return flow {
        emit(Resource.Loading())
        val result = firebaseAuth.signInWithEmailAndPassword(email,
password).await()
        if (firebaseAuth.currentUser?.isEmailVerified == true) {
            emit(Resource.Success(result))
        } else {
            emit(Resource.Error("Please check your email and verify
your account!"))
        }
    }.catch {
        emit(Resource.Error(it.message.toString()))
    }
}
```

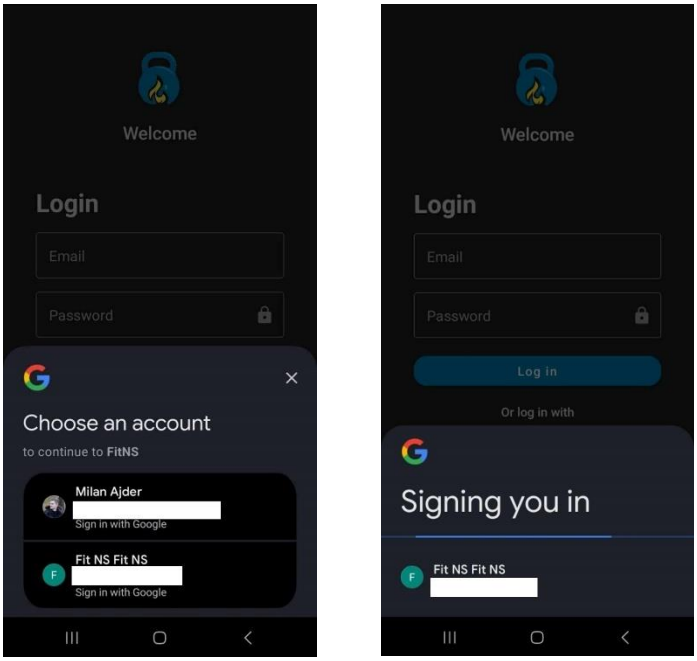
Листинг 5.1 – пријава корисника помоћу *FirebaseAuth* [9]

Функција прима параметре `email` и `password` као корисникове пријавне податке и враћа `Flow` објекат који ће емитовати стања (`Loading`, `Success` или `Error`) аутентикације у облику `Resource` објекта. Прва ствар коју функција ради је емитовање стања `Loading` помоћу `emit` функције. Ово сигнализира да је процес пријављивања започет и да још увек чекамо на резултат.

Користећи *Firebase* аутентикацију, пробамо да пријавимо корисника користећи његову е-пошту и лозинку помоћу методе `signInWithEmailAndPassword(email, password).await()`. Затим проверавамо да ли је корисник верификовао своју е-пошту користећи услов `if (firebaseAuth.currentUser?.isEmailVerified == true)`. Ако јесте, објављујемо статус `Success` са резултатима аутентикације. Ако није, обавештавамо корисника да провери своју е-пошту и верификује свој налог. У случају да се деси грешка при аутентикацији, користимо `catch` блок да обрадимо изузетак и обавестимо корисника о грешци.

4.1.2 Пријава помоћу *Google* налога

`GoogleAuthUiClient` је класа која служи за обављање *Google* аутентикације [10] у апликацији помоћу *One Tap* аутентикације. *One Tap* аутентикација се тако зове јер омогућава кориснику да се пријави кроз само један клик и избор свог налога (Слика 6).



Слика 6 - *One tap* аутентикација

Класа `GoogleAuthUiClient` садржи функционалности које омогућавају пријављивање корисника помоћу *Google* налога и добијање информација о кориснику (листинг 5.2).

```
class GoogleAuthUiClient(
    private val context: Context,
    private val oneTapClient: SignInClient
) {
    private val auth = Firebase.auth

    suspend fun signIn(): IntentSender? {
        val result = try {
            oneTapClient.beginSignIn(
                buildSignInRequest()
            ).await()
        } catch(e: Exception) {
            e.printStackTrace()
            if(e is CancellationException) throw e
            null
        }
        return result?.pendingIntent?.intentSender
    }

    suspend fun signInWithIntent(intent: Intent): SignInResult {
        val credential =
oneTapClient.getSignInCredentialFromIntent(intent)
        val googleIdToken = credential.googleIdToken
```

```
        val googleCredentials =
GoogleAuthProvider.getCredential(googleIdToken, null)
        return try {
            val user =
auth.signInWithCredential(googleCredentials).await().user
            SignInResult(
                data = user?.run {
                    UserData(
                        userId = uid,
                        displayName = displayName.toString(),
                        email = email!!,
                        profilePictureUrl = photoUrl?.toString()
                    )
                },
                errorMessage = null
            )
        } catch(e: Exception) {
            e.printStackTrace()
            if(e is CancellationException) throw e
            SignInResult(
                data = null,
                errorMessage = e.message
            )
        }
    }
```

Листинг 5.2 - пријава корисника помоћу *Google* налога

`Firebase.auth` се користи за аутентикацију корисника помоћу *Firebase Authentication*. Одређене функције у овом контексту укључују иницирање *One Tap* аутентикације помоћу `suspend fun signIn()`, која враћа интерфејс који се користи за пријаву, и `suspend fun signInWithIntent(intent: Intent)`, која се користи након успешне *One Tap* аутентикације за пријаву корисника на систем. Такође, постоји функција `fun buildSignInRequest()` која се користи за креирање захтева за One Tap аутентикацију и подешава конфигурацију за *Google* аутентикацију, укључујући и *Google* `ServerClientId`.

Уколико је корисник успешно аутентификован, врши се провера да ли корисиник већ поседује налог или се сада први пут пријавио са датом е-поштом. Уколико корисник није ранио имао налог, додаје се у *Firestore* базу података. Податке о имену, презимену, е-пошти и слици корисника добијамо из функције: `fun signInWithIntent(intent: Intent): SignInResult.`

4.1.3 Чување пријављеног корисника

Уколико је успешна пријава, како корисник не би морао следећи пут кад уђе у апликацију да се поново логује, потребно је сачувати на

уређају подтаке о пријављеном кориснику. За то се користе функције `SessionCacheImpl` класе (листинг 5.3):

```
class SessionCacheImpl @Inject constructor(
    private val sharedPreferences: SharedPreferences
): SessionCache {

    private val moshi = Moshi.Builder()
        .add(KotlinJsonAdapterFactory())
        .build()
    private val adapter = moshi.adapter(Session::class.java)

    override fun saveSession(session: Session) {
        sharedPreferences.edit()
            .putString("session", adapter.toJson(session))
            .apply()
    }

    override fun getActiveSession(): Session? {
        val json = sharedPreferences.getString("session", null) ?:
return null
        return adapter.fromJson(json)
    }

    override fun clearSession() {
        sharedPreferences.edit().remove("session").apply()
    }
}
```

Листинг 5.3 - чување, брисање и добављање сесије

Функција `override fun saveSession(session: Session) { ... }` је одговорна за складиштење *Session* објекта у кешу. Користи се *SharedPreferences* [11] да би се поставила вредност под називом "*session*" и претворила је у *JSON* помоћу `JsonAdapter`-а.

Функција `override fun getActiveSession(): Session? { ... }` враћа текући `Session` објекат из кеша. Прво се добија *JSON* вредност из `SharedPreferences`, а затим се десеријализује у `Session` објекат користећи `JsonAdapter`. Ако вредност није пронађена, враћа се `null`.

Функција `override fun clearSession() { ... }` брише складиштени `Session` објекат из кеша користећи `SharedPreferences`.

4.1.4 Регистрација

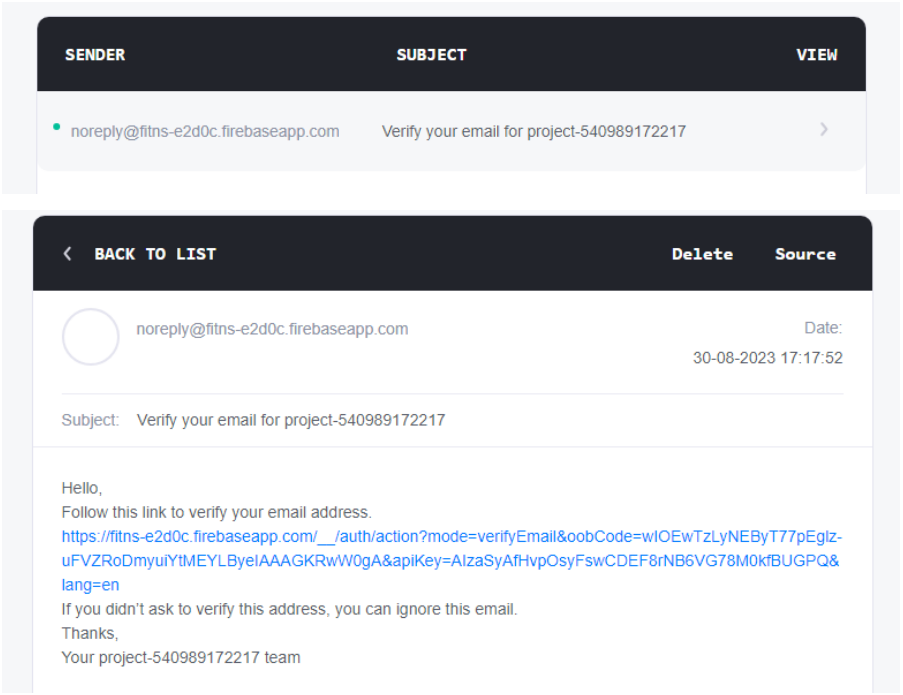
Попуњавањем свих поља и кликом на *Sign up* дугме, на екрану за регистрацију (Слика 5), корисник позива функцију (листинг 5.4) класе `AuthRepositoryImpl`:

```
override fun registerUser(email: String, password: String):
Flow<Resource<AuthResult>> {
    return flow {
        emit(Resource.Loading())
        val result =
firebaseAuth.createUserWithEmailAndPassword(email, password).await()
        firebaseAuth.currentUser?.sendEmailVerification()?.await()
        emit(Resource.Success(result))
    }.catch {
        emit(Resource.Error(it.message.toString()))
    }
}
```

Листинг 5.4 - регистрација новог корисника

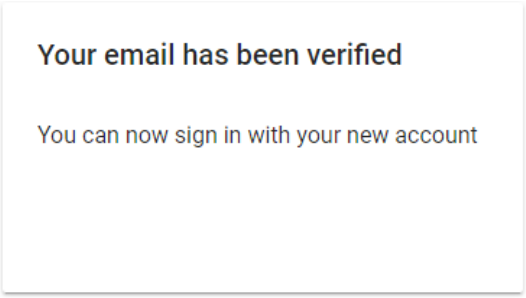
Користећи `FirebaseAuth` инстанцу, региструјемо новог корисника помоћу унете е-поште и лозинке користећи функцију `firebaseAuth.createUserWithEmailAndPassword(email, password).await()`. Ова операција се извршава асинхроно, а `await()` се користи како би се сачекао њен завршетак.

Након успешне регистрације, функција `firebaseAuth.currentUser?.sendEmailVerification()?.await()` шаље верификациони е-мејл на регистровану е-пошту корисника како би потврдио свој налог. Ова операција такође чека на извршење коришћењем `await()`. Ако ова функција врати ресурс са `Success` статусом, додају се информације о кориснику у *Firestore* базу података.



Слика 7 - Мејл за потврду регистрације

Након што корисник кликне на линк са слике (Слика 7) добија поруку о успешном регистровању (Слика 8). Након тога, профил је верификован и корисник може да се пријави на систем.



Слика 8 - Порука о успешном регистровању

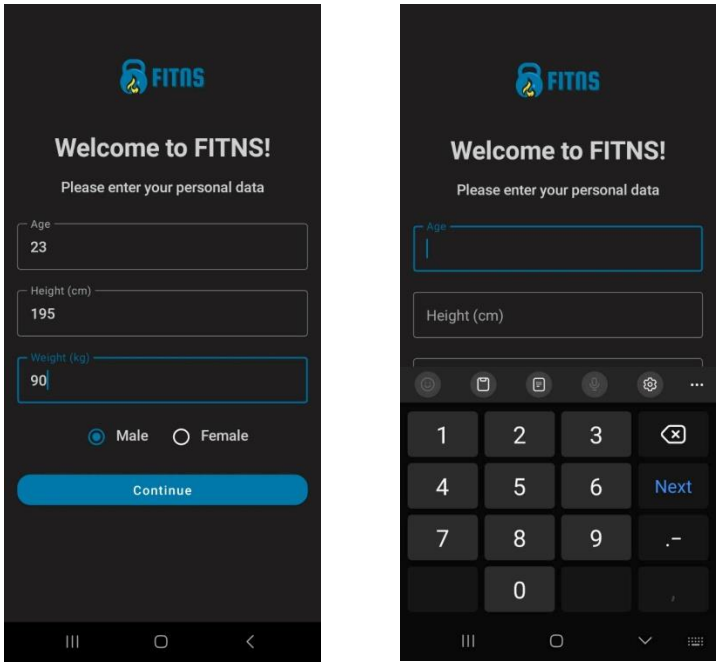
4.1.5 Даља усмерења

Уколико се корисник први пут пријављује на систем било то преко регистрационе форме или *Google* пријаве, усмерава се страницу

добродошлице (поглавље 38), где треба да остави додатне личне податке. Иначе, усмерава се на страницу са новостима (0).

4.2 Страница добродошлице

Приликом првог пријављивања на систем (док год корисник није унео личне податке) корисник се усмерава на страницу добродошлице на којој је неопходно да унесе податке о својим годинама, висини, тежини и полу (Слика 9). Ти подаци су кориснцни због повезивања са другим корисницима (пол и године), али и због личног праћења промена у тежини и висини. Кликом на дугме *Continue* шаљу се подаци у *Firestore* базу података и корисник се усмерава на страницу са новостима (0).



Слика 9 - Приказ екрана добродошлице - унос личних података

4.3 Вежбе

4.3.1 API

Апликација користи ExerciseDB *API* [12]. *API* садржи 1324 вежбе. За добављање вежби је коришћена једна крајња тачка - /exercises (листинг 5.5).

```
@Headers(
    "X-RapidAPI-Key: api key",
    "X-RapidAPI-Host: exercisedb.p.rapidapi.com"
)
@GET("/exercises")
suspend fun getExercises(
): List<ExerciseDTO>

data class ExerciseDTO(
    val bodyPart: String,
    val equipment: String,
    val gifUrl: String,
    val id: String,
    val name: String,
    val target: String
)
```

Листинг 5.5

Из одговора су коришћени сви параметри осим `bodyPart`. Подаци параметра су били доста слични `target` параметру, а `target` је давао детаљније податке.

Вредности атрибута су: `equipment, target, gifUrl, id, name.`

Опрема за вежбање (`equipment`) укључује `body weight, cable, leverage machine, assisted, medicine ball, stability ball, band, barbell, rope, dumbbell, ez barbell, sled machine, upper body ergometer, kettlebell, olympic barbell, weighted, bosu ball, resistance band, roller, skierg machine, hammer, smith machine, wheel roller, stationary bike, tire, trap bar, elliptical machine, stepmill machine`

`target: - abs, quads, lats, calves, pectorals, glutes, hamstrings, adductors, triceps, cardiovascular system, spine, upper back, biceps, delts, forearms, traps, serratus anterior, abductors, levator scapulae.`

Параметар `gifUrl` представља путању до локације до *GIF*-а. Путања до *GIF*-а, се мења на сваких 24 часа. Како аутор рада не би чинио ништа нелегало, нису преузимани *GIF*-ови. Касније у овом поглављу је објашњен принцип рада са *GIF*-овима.

Параметар `id` представља јединствени индетификатор вежбе.

Параметар `name` представља назив вежбе.

У периоду развоја и писања овог рада, *API* дозвољава бесплатну претплату до 400 захтева у месец дана. Како се не би тај број током развоја премашио, подаци о вежбама су складиштени у *Room* бази података. Сваки пут када би база била празна (први пут када корисник захтева вежбе) или када слика/*GIF* вежбе не би могао да се учита из првог покушаја, слао би се позив ка *API*-ју. У другим случајевим би се ти подаци добављали из локалне базе. То значи да би број позива ка бази у месец дана био једнак производу броја корисника и просечног број дана у месецу у којима је коришћена апликација по особи.

4.3.2 Добављање вежби

Списак свих вежби приказан је `ExerciseListScreen` екраном. Приликом покретања екрана добављају се вежбе позивом функције из листинга 5.6

Функција прима вредности као што су: да ли треба преузети податке из *API*-ја (`fetchFromRemote`), текстуални упит за претрагу (`query`), и листе одабраних делова тела (`selectedBodyParts`) и опреме (`selectedEquipment`). `fetchFromRemote` је тачан само у ситуацији када нисмо успешно добили слику за вежбу. Уколико су `selectedBodyParts` или `selectedEquipment` празна листа, подразумева се да није примењен ни један филтер и да се тражи сви елементи из поглавља 4.3.1

Функцијом `dao.searchExercises(...)` се претражује локална база података (листинг 5.7) и филтрирају тражене вежбе.

Ако је скуп добијених вежби након филтрације празан, а при томе није употребљен ни један филтер – значи да је база празна.

Уколико је база празна или `fetchFromRemote` је тачан, онда је неопходно да се позове *API* и сместе добијене вредности у локалну базу података.

```
override suspend fun getExercises(
    fetchFromRemote: Boolean,
    query: String,
    selectedBodyParts: List<String>,
    selectedEquipment: List<String>
): Flow<Resource<List<Exercise>>> {
    return flow {
        emit(Resource.Loading(true))
        val localExercises = dao.searchExercises(
            query,
            determineBodyPartsQuery(selectedBodyParts),
            determineEquipmentQuery(selectedEquipment)
        )
        emit(Resource.Success(data = localExercises.map {
```

```
it.toExercise() }))

        val isDbEmpty = localExercises.isEmpty() && query.isBlank()
        val shouldLoadFromCache = !isDbEmpty && !fetchFromRemote
        if (shouldLoadFromCache) {
            emit(Resource.Loading(false))
            return@flow
        }
        val remoteExercises = try {
            api.getExercises()
        } catch (e: IOException) {
            e.printStackTrace()
            emit(Resource.Error("Could not load data"))
            null
        } catch (e: HttpException) {
            e.printStackTrace()
            emit(Resource.Error("Could not load data"))
            null
        }

        remoteExercises?.let { exercises ->
            dao.clearCompanyListings()
            dao.insertExercises(exercises.map {
it.toExercise().toExerciseEntity() })
            emit(Resource.Success(
                data = dao.searchExercises(
                    "",
                    determineBodyPartsQuery(selectedBodyParts),
                    determineEquipmentQuery(selectedEquipment)
                ).map { it.toExercise() }
            ))
            emit(Resource.Loading(false))
        }
    }
}
```

Листинг 5.6

```
@Query(
    """
    SELECT *
    FROM exerciseentity
    WHERE LOWER(name) LIKE '%' || LOWER(:query) || '%'
    AND (equipment IN (:selectedEquipment))
    AND (target IN (:selectedBodyParts))
    """
)
suspend fun searchExercises(
    query: String,
    selectedBodyParts: List<String>,
    selectedEquipment: List<String>
): List<ExerciseEntity>
```

Листинг 5.7

4.3.3 Приказ

Списак свих вежби приказан је `ExerciseListScreen` екраном. Екран садржи списак вежби, број вежби који је обухваћен тренутним филтером (Слика 10).

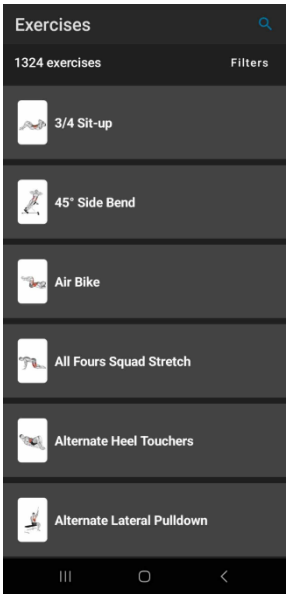
Кликом на *Filters* дугме, отвара се нова страница `ExerciseFilterScreen`. Страница се састоји из слике мишића човека (Слика 11, Слика 13), која може да се окреће напред, назад и списка опреме за тренинг (Слика 14).

На додир на одређену групу мишића на слици човека, та група мишића треба да се означи плавом бојом. За сваку групу мишића је дефинисан регион. Оквир региона се скалира у зависнсти од димензија екрана телефона. Региони се не преклапају. Приликом клика на регион, он се додаје/уклања са списка означених региона и онда се мишићи из региона боје у плаво/бело. За сваку групу мишића је дефинисана по једна тачка унутар сваког мишића из групе. Те тачке се такође скалирају у односу на димензије екрана телефона. Када група мишића треба да промени боју, за сваку тачку из скупа се примењује операција бојења слике, тако да се боје сви пиксели у близини тих тачака који су исте те боје. На тај начин фарбамо мишиће до ивица и добијамо исти ефекат као фарбање у *Paint*-у.

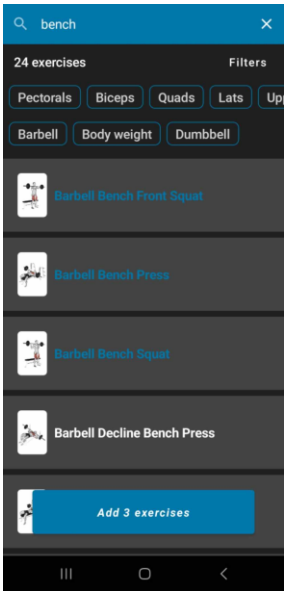
Када се вратимо назад на `ExerciseListScreen` можемо видети изабране мишићне групе и опрему. Такође, можемо претраживати вежбе и по називу. Захтев за претрагу се иницира тек након пола секунде од када корисник престане да укоси карактере у поље за претрагу. Као што видимо на слици (Слика 12) смањио се број вежби након претраге.

Превлачењем вежбе из листе на лево или десно (Слика 16), води нас до екрана са описом вежбе (Слика 15). Екран садржи *GIF*, назив, део тела за који је та вежба и опрему.

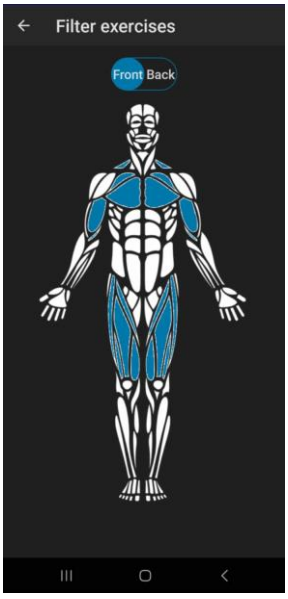
Кликом на вежбе, оне се додају на листу одабраних вежби. Када се кликне на *Add x exercises* дугме, те вежбе се додају на листу вежби екрана који је претходио `ExerciseListScreen` на стеку екрана. Такође, `ExerciseListScreen` се том приликом склања са стека.



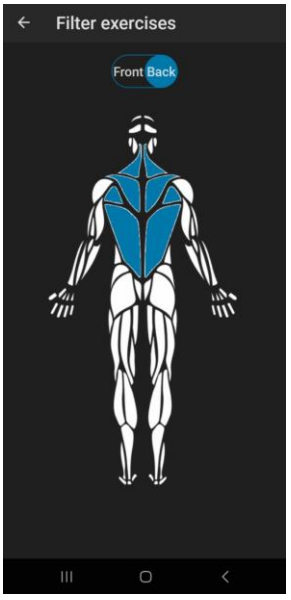
Слика 10 - Списак вежби



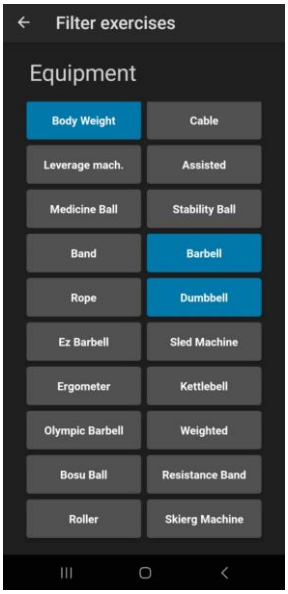
Слика 12 - Списак вежби са филтерима и одабраним вежбама



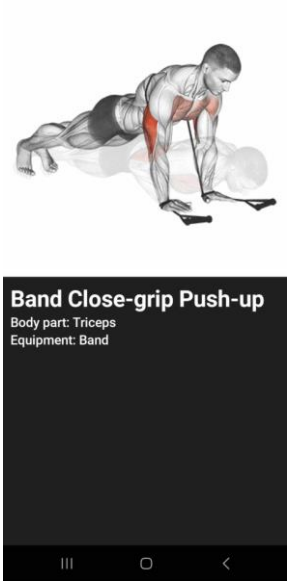
Слика 11 - Мишићне групе - предња страна



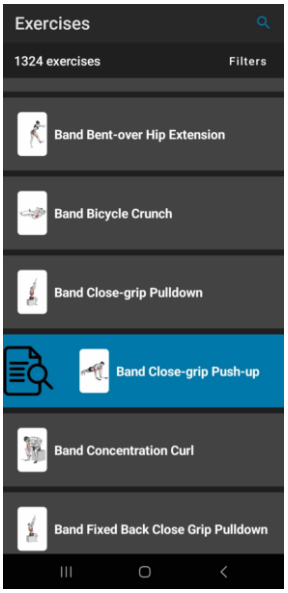
Слика 13 - Мишићне групе - задња страна



Слика 14 - Списак опреме



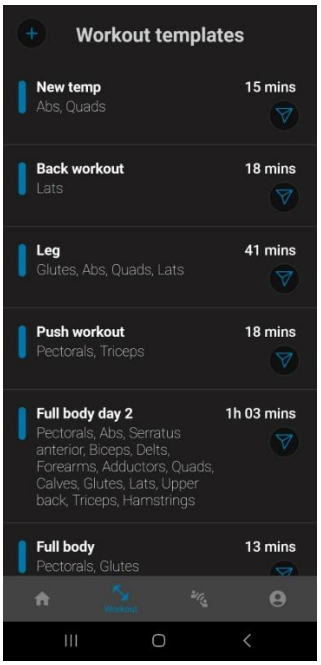
Слика 15 - Приказ извођења вежбе



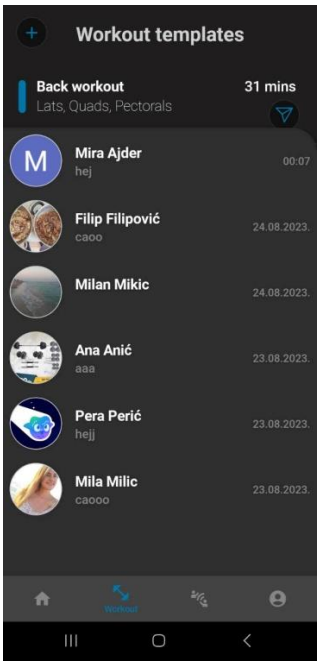
Слика 16 - Превлачење вежбе

4.4 Шаблони за вежбање

Екран шаблона за вежбање, `WorkoutTemplatesScreen`, се састоји из листе шаблона за вежбање. Приликом иницијализације странице прво се добављају сви шаблони за вежбање пријављеног корисника. За сваки шаблон је приказан назив шаблона, мишићне групе које су највише активиране током вежбања и процењено време тренинга (Слика 17). Такође, за сваки шаблон постоји и дугме за дељење шаблона са повезаним корисницима. Притиском на дугме отвара се *BottomSheet* компонента са листом повезаних корисика којима се може послати шаблон (Слика 18). Кликом на неког од повезаних корисника, шаблон се шаље другом кориснику као порука (4.10), а *BottomSheet* компонента се спушта доле.

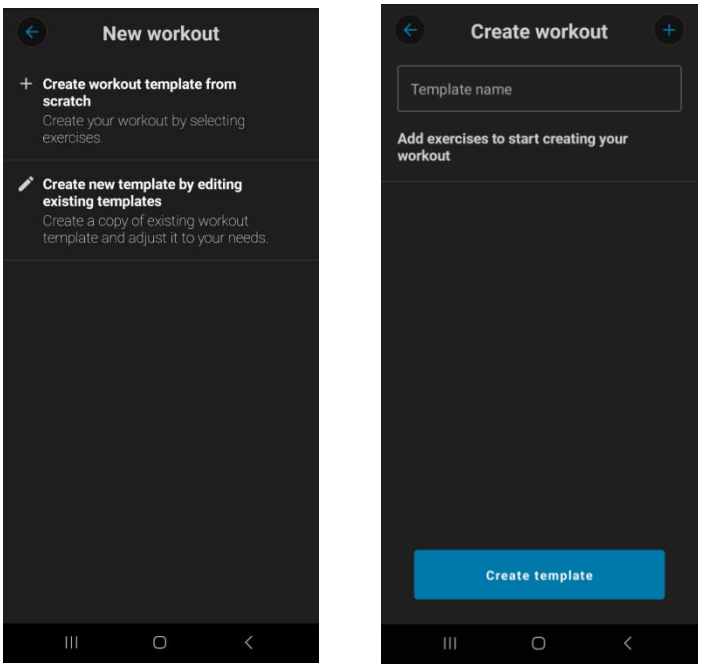


Слика 17 - Листа шаблона за вежбање



Слика 18 - Слање шаблона

Кликом на знак плус у горњем левом углу отвара нови екран за креирање новог шаблона. Екран садржи опцију да се креира нови шаблон од почетка, или да се креира шаблон као копија постојећег шаблона (Слика 19).



Слика 19 - Странице за креирање новог шаблона

Избором креирања скроз новог шаблона позиционирамо се на екран за детаљан приказ шаблона `WorkoutTemplateDetailsScreen` у моду `CREATE`, с тим што шаблон не садржи ни једну вежбу и назив није подешен. (Слика 19).

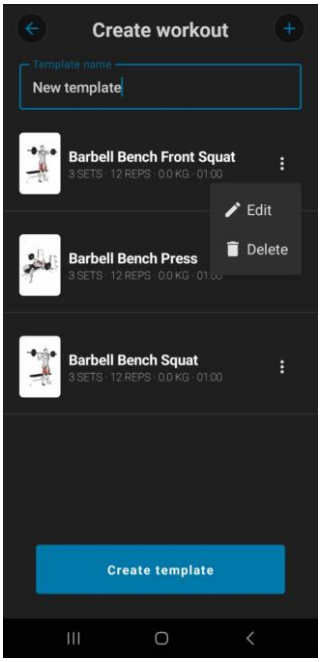
Како би искористили `WorkoutTemplateDetailsScreen` у више различитих ситуација, приликом навигирања прослеђујемо путањи опционе параметре `template_id` и `mode` (листинг 5.8). Постоје 4 различита мода (`CREATE`, `EDIT`, `START`, `SEND`) и у зависности од мода, постоје разлике у корисничком интерфејсу.

```
composable(route = Screen.WorkoutTemplateDetailsScreen.route +
"?template_id={template_id}&mode={mode}") {
    WorkoutTemplateDetailsScreen(navController)
}
```

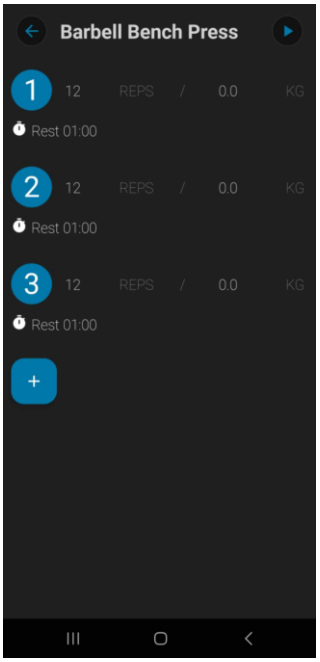
Листинг 5.8 - навигација до екрана за детаље о шаблону

Кликом на плус знак отварамо `ExerciseListScreen` (поглавље 4.3.3.) Када изабреремо жељене вежбе и вратимо се назад на `WorkoutTemplateDetailsScreen`, појавиће се изабране вежбе. Након клика на три тачнице са десне стране вежбе, појављује се падајући мени са опцијама измена и брисања (Слика 20). Клик на брисање, уклања

вежбу из листе из корисничког интерфејса. Клик на измену отвара екран `WorkoutExerciseDetailsScreen` у `EDIT` моду (Слика 21).

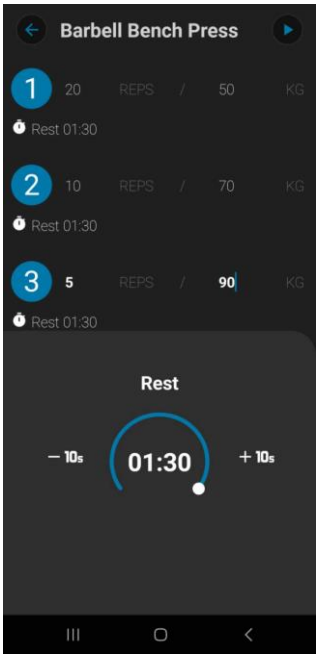


Слика 20 - Шаблон са вежбама

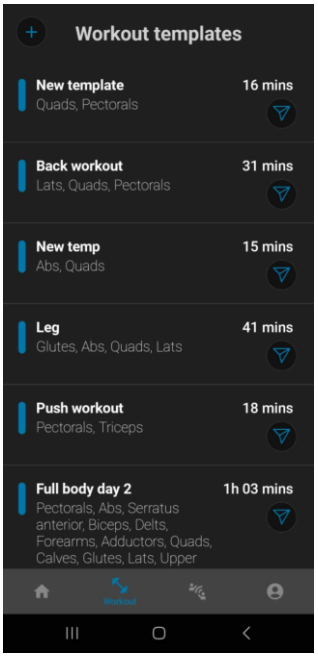


Слика 21 - Детаљан приказ вежбе у тренингу

На `WorkoutExerciseDetailsScreen` се налази списак серија са бројем понављања, килажом за сваки тренинг и пазузом између серија. У горњем десном углу се налази дугме које навигира до екрана са описом вежбе (слика 5.15). Кликом на плус дугме, додаје се копија последње серије као нова последња серија. Дужим држањем на серију вежбе, појављује се опција за брисање вежбе. Кликом на време паузе активира се *BottomSheet* у којем можемо да изменимо време паузе између серија (Слика 22).



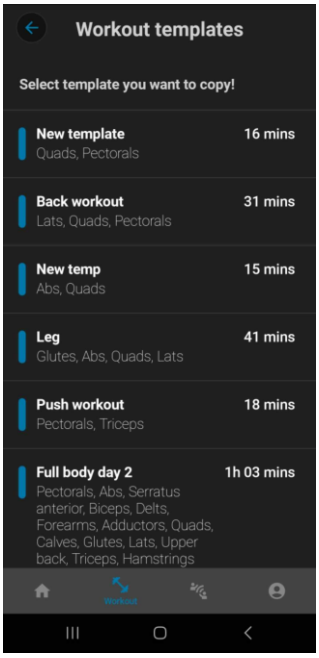
Слика 22 - Измењене вредности понављања, килаже и времена



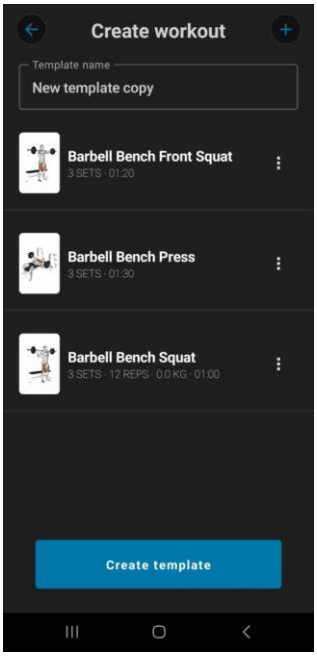
Слика 23 - Шаблон је додат на списак шаблона

Када су све измене завшене и корисник кликне на *Create template* дугме, шаблон се чува у *Firestore* бази података и корисник се навигира на страницу са списком шаблона `WorkoutTemplatesScreen` (Слика 23.)

Уколико бисмо уместо скроз из почтка, креирали нови шаблон као копију постојећег, разлика би била у томе што би имали избор ког шаблона желимо копију (Слика 24). Такође би била разлика то што би се учитале вежбе које су копиране и назив шаблона би био назив оригинала + размак + *copy* (Слика 25).

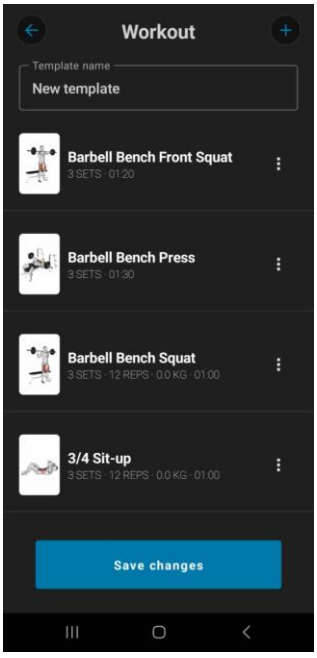


Слика 24 - Приказ избора шаблона за копију

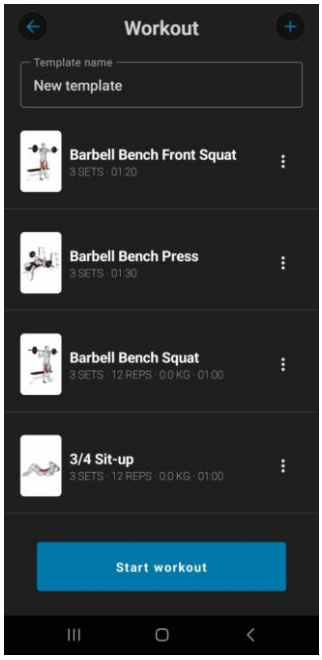


Слика 25 - Приказ креиране копије, без измена

Уклолико бисмо изабрали неки од постојећих шаблона, мод за `WorkoutTemplateDetailsScreen` би био `START`. То значи да уколико бисмо притиснули дугме са текстом *Start workout*, покренули бисмо тренинг (4.7). Уколико бисмо у `START` моду направили неку измену на шаблону, аутоматски бисмо се пребацили у `EDIT` мод и уместо *Start workout* би писало *Save changes* (Слика 26). Кликом на *Save changes* шаље се захтев *Firestore*-у за чување измена. Тек након сачуваних или одбачених измена, мод се поново враћа у `START` и корисник може поново да покрене тренинг (Слика 27).



Слика 26 - Додата је нова вежба и мод је EDIT



Слика 27 - Сачувана је изена и поново смо у START моду

4.5 Повезивање са другим корисницима

4.5.1 Екран за превлачење других корисника

Екран за превлачење служи за повезивање са другим корисницима. Екран садржи профилну слику другог корисника, његово име и презиме и највећу тренажну килажу на три најпопуларније вежбе са оптерећењем (*bench press*, чучањ, и мртво дизање) (Слика 28). Такође садржи два округла дугмета за означавање да ли нам се свиђа или не свиђа дати корисник. Означавање да нам се свиђа или не свиђа корисник се може извршити и путем превлачења картице ка којој се налази слика корисника на десно/лево. Превлачење картица је могуће помоћу *Compose Tinder Card* библиотеке [13].

Функција `getUsersInCloseRangeWithDesiredAgeAndGender(email: String, settings: ConnectSettings): UsersResponse` се користи за добављање корисника који задовољавају `ConnectSettings` параметре за изабраног корисника (`email`). Проверава се да ли је корисник траженог пола, старости и да ли се

налази у довољној близини са изабраним корисником. Удаљеност између корисника се рачуна по *Haversine* формули (листинг 5.9) [14].

```
fun haversineDistance(lat1: Double, lon1: Double, lat2: Double, lon2:
Double): Double {
    val lat1Rad = Math.toRadians(lat1)
    val lon1Rad = Math.toRadians(lon1)
    val lat2Rad = Math.toRadians(lat2)
    val lon2Rad = Math.toRadians(lon2)

    // Haversine formula
    val dLat = lat2Rad - lat1Rad
    val dLon = lon2Rad - lon1Rad
    val a = sin(dLat / 2).pow(2) + cos(lat1Rad) * cos(lat2Rad) *
sin(dLon / 2).pow(2)
    val c = 2 * atan2(sqrt(a), sqrt(1 - a))

    // Earth's radius in kilometers
    val radius = 6371.0

    // Calculate the distance
    val distance = radius * c
    return distance
}
```

Листинг 5.9

`getNonConnectionsInCloseRangeWithDesiredAgeAndGender(email: String): List<User>`. Функција се користи за добављање свих корисника који задовољавају мало пре помињана подешавања и додаје додатне услове. Дати корисници не смеју бити већ повезани са изабраним корисником (`email`) и изанрани корисник већ није реаговао на дате кориснике (дати корисници се не налазе у `likes` или `dislikes` листама `UserSocial` објекта корисника).

На овај начин препоручени корисници су само корисници који се налазе у границама тражене дистанце, траженог пола и година и не понављају се. Једном када корисник назначи да му се неки корисник свиђа или не свиђа, тај корисник му се неће више појављивати у секцији повезивања.

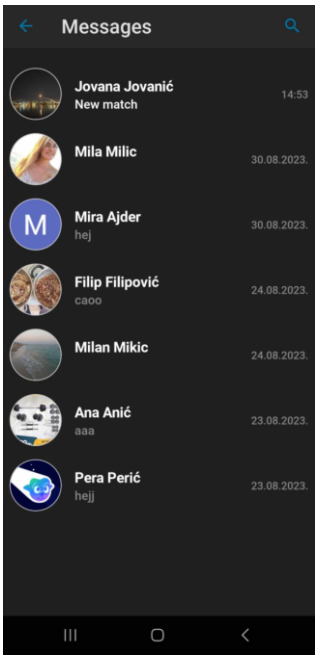
Уколико корисник остане без препоручених корисника, на екрану му се појављује порука да промени подешавања препоруке корисника (1.1.1).

4.5.2 Назначавање да нам се корисник свиђа или не свиђа

Превлачењем на десно/лево или кликом на одговарајуће дугме назначавамо да ли нам се други корисник свиђа или не свиђа. Уколико

назначимо да нам се други корисник свиђа и то је исто други корисник учинио за нас, постајемо повезани (додаје се други корисник у листу `connections`, `UserSocial` обејкта, оба корисника). Такође, креира се ново дописивање за оба корисника са последњом поруком *New match* (Слика 29).

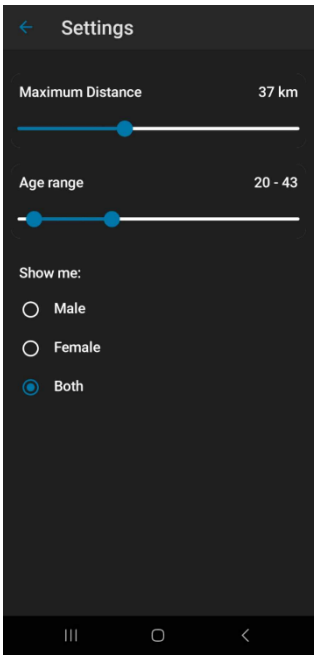
Слика 28 - Екран за повезивање са другим корисницима



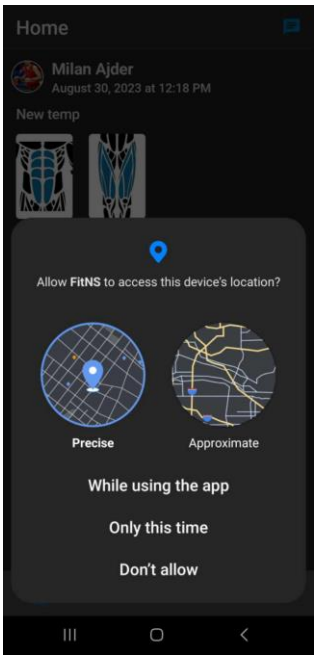
Слика 29 - Додат нов повезан корисник након обостране назнаке свиђања

4.5.3 Подешавања препоруке корисника

Корисник има опцију да дефинише различите параметре који утичу на препоруку других корисника. Ти параметри су максимална удаљеност, опсег година и пол другог корисника Слика 30). Промена било ког од ових параметара иницира функцију: `suspend fun updateSettings(email: String, settings: ConnectSettings)`, која за датог корисника (`email`) ажурира `settings` вредност `UserSocial` објекта у *Firestore* бази података. `Settings` параметар се после користи за филтрацију других корисника.



Слика 30 - Подешавања за препоруку корисника



Слика 31 - Захтев за коришћење локације

4.5.4 Локација уређаја

Како би корисници могли да се повезују са другим корисницима, неопходно је да дозволе прикупљање информација о локацији уређаја [15]. Подаци о локацији се користе за филтрирање препоруке корисника по удаљености. Када корисник покрене апликацију, уколико апликација нема дозволу за коришћење података о локацији, шаље се захтев за дозвовлу коришћења података о локацији (Слика 31). Када корисник прихвати захтев, уређај емитује сваких 10 секунди нову позицију која се чува у *Firestore* бази података као `LocationDetails` објекат.

4.6 Профил корисника

Профил корисника се састоји из основних информација о кориснику, извештаја о тренинзима и објава.

4.6.1 Основне информације о кориснику

Основне информације се састоје од имена и презимена, профилне слике, броја корисника са којима је корисник повезан, броја тренинга, броја узастопних дана које је корисник тренирао, година, тежине и висине.

Профилна слика може да се промени кликом на исту, при чему се отвара `BottomSheet` компонента са галеријом уређаја (Слика 33). Када корисник изабере нову слику, спушта се `BottomSheet` и нова слика се учитава на месту претходне. У позадини, прво се слика дода у *Firebase Storage* (листинг 5.10) [16], а када је то успешно завршено, додаје се путања до слике као параметар `image`, `User` колекције у *Firestore*-у (листинг 5.11).

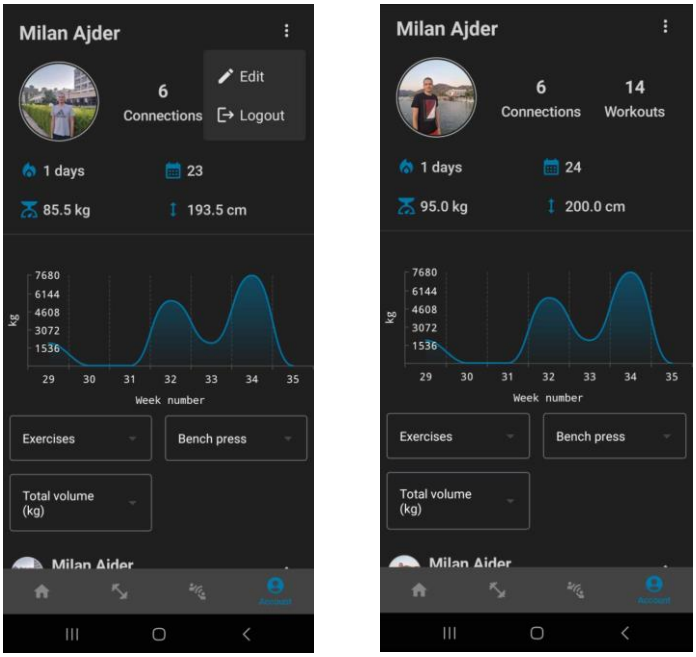
```
override suspend fun addImageToFirebaseStorage(imageUri: Uri, email:
String): AddImageToStorageResponse {
    return try {
        val downloadUrl =
storage.reference.child(IMAGES).child("$email.jpg")
            .putFile(imageUri).await()
            .storage.downloadUrl.await()
        Response.Success(downloadUrl)
    } catch (e: Exception) {
        Response.Failure(e)
    }
}
```

Листинг 5.10

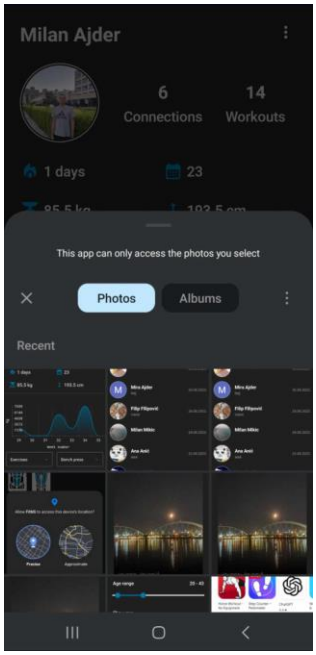
```
override fun updateUserImage(email: String, downloadUrl: Uri)
        : AddImageUrlToFirestoreResponse {
    val userRef = usersRef.whereEqualTo("email", email)
    userRef.get()
        .addOnSuccessListener { querySnapshot ->
            if (!querySnapshot.isEmpty) {
                val documentSnapshot = querySnapshot.documents[0]
                val userDocumentRef =
usersRef.document(documentSnapshot.id)
                userDocumentRef.update("image",
downloadUrl.toString())
                    .addOnSuccessListener {
                        Response.Success(true)
                    }
                    .addOnFailureListener { e ->
                        Response.Failure(e)
                    }
            }
        }
        .addOnFailureListener { e ->
            Response.Failure(e)
        }
    return Response.Loading
}
```

Листинг 5.11

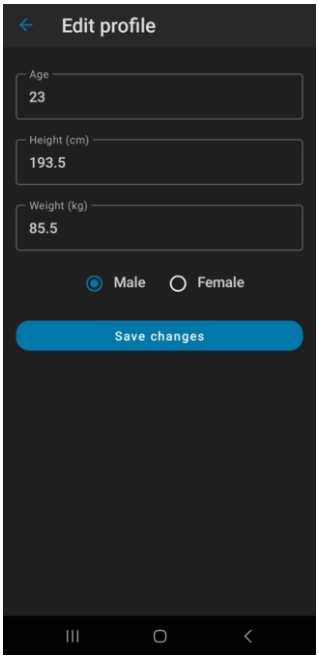
Основне информације као што су године, пол, висина и тежина се такође могу изменити. Кликом на три тачке у горњем десном углу, отвара се падајући мени који садржи опцију измене профила (`Edit`). Екран за измену профила (Слика 34) при покретању учитава тренутне вредности личних информација. Кликом на дугме *Save changes* чувају се измене на *Firestore*-у и корисник се враћа на екран профила (Слика 32).



Слика 32 – Пре и после измене основних информација



Слика 33 - Отворена галерија притиском на профилну слику



Слика 34 - Екран за измену профила

4.6.2 Извештаји о тренинзима

Извештаји о тренинзима приказани су дијаграмом (слика 5.32). Дијаграм је конструисан помоћу *Vico* библиотеке [17].

Испод дијаграма се налази три падајућа менија за избор приказа на дијаграму. У зависности од избора првог падајућег менија, мењају се опције остала два менија. У првом падајућем менију може да се изабере да ли желимо дијаграм за вежбе или мишићне групе.

Уколико је изабран дијаграм за вежбе, у другом падајућем менију може да се изабере која конкретно вежба, из скупа најчешћих вежби, ће се посматрати. У последњем падајућем менију може да се изабере један од 4 избора:

- дијаграм за укупан волумен вежбе у килограмима
- дијаграм за укупан број серија за вежбу
- дијаграм за просечан интензитет на вежби у килограмима
- дијаграм за највећи интензитет на вежби у килограмима

Уколико је изабран дијаграм за мишићну групу, у другом падајућем менију може да се изабере која конкретно мишићна група, из

скупа мишићних група, ће се посматрати. У последњем падајућем менију може да се изабере један од 2 избора:

- дијаграм за укупан волумен мишићне групе у килограмима
- дијаграм за укупан број серија за мишићну групу

4.6.3 Објаве на профилу

Корисник има могућност да прегледа своје објаве на профилу. Сваку објаву објаву може да погледа детаљније и да је обрише. Више о објавама у поглављу 4.8.

4.6.4 Преглед туђих профила

Корисник може да прегледа профиле корисника са којима је повезан. Разлика у односу на преглед свог профила је у томе што кориснсник не може да мења профилну слику, личне податке или да брише објаве корисника (Слика 35).

Слика 35 - Преглед профила повезаног корисника

4.6.5 Одјава

Кликом на три тачке у горњем десном углу, отвара се падајући мени који садржи опцију одјаве са профила (`Logout`). Када корисник

кликне да опцију, позове се функција `clearSession()`, `SessionCacheImpl` класе (Чување пријављеног корисника) и корисник се упућује на страницу за пријаву.

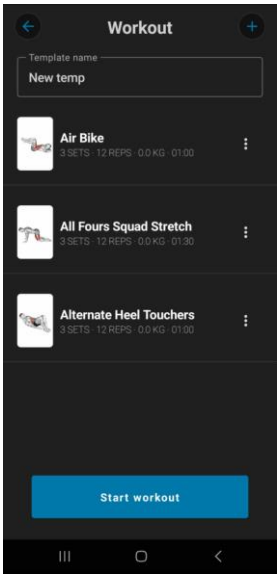
4.7 Вежбање

Избором шаблона за вежбање (Слика 36) и кликом на дугме Start workout корисник започиње тренинг. На страници за тренинг се налазе назив шаблона, протекло време од почетка тренинга, вежбе које су у тренингу и дугме за завршетак тренинга. Нове вежбе се могу додати притиском на плус дугме у горњем десном ћошку. Кликом на три тачкице отвара се падајући мени са три избора (Слика 37):

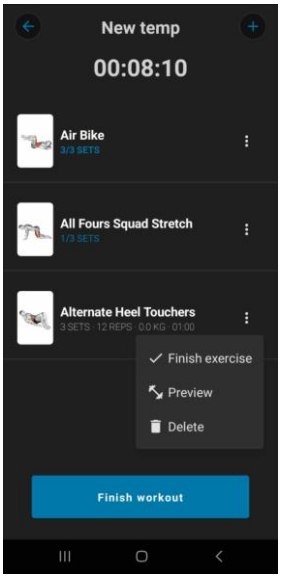
- Завршетак целе вежбе
- Приказ серија вежбе
- Брисање вежбе

Кликом на приказ серија вежбе, приказује се екран са серијама дате вежбе (Слика 39). Број серија, понављања и килажа се могу мењати, исто као и код измене шаблона. Такође корисник може да кликне да дугме у горњем десном ћошку како би погледао *GIF* извођења вежбе. У доњем десном углу се налази дугме *Log set*. Кликом на дугме, завршава се тренутно изабрана серија и покреће се тајмер за паузу (Слика 38). Ако се тајмер не прекине док не истекне време, корисник се обавештава звучним ефектом да је пауза готова. На екрану за тренинг се може видети колико серија је корисник одрадио за коју вежбу.

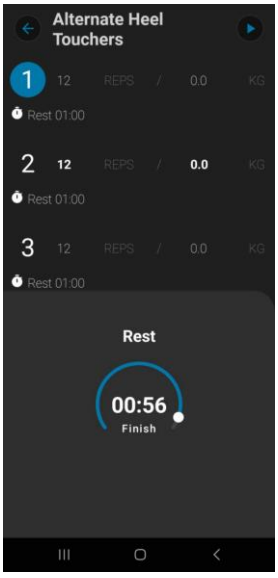
Кликом на *Finish workout* дугме, кориснику се приказује екран са детаљним извештајем његовог тренинга (Слика 40). У извештају су само оне вежбе које су обележене као готове. Одатле корисник може да се врати назад на тренинг, или да заврши тренинг кликом на *Finish workout* дугме. Када је корисник завршио тренинг, подаци о тренингу се чувају у *Firestore* бази података, креира се објава која се може видети на профилу корисника и у новостима.



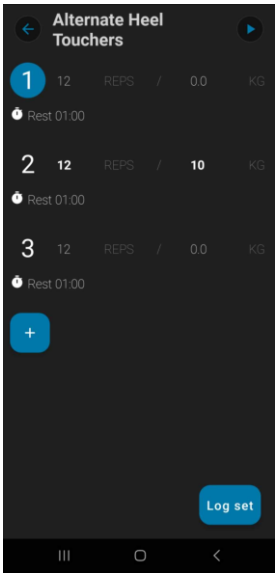
Слика 36 - Шаблон за вежбање, спрена за тренинг



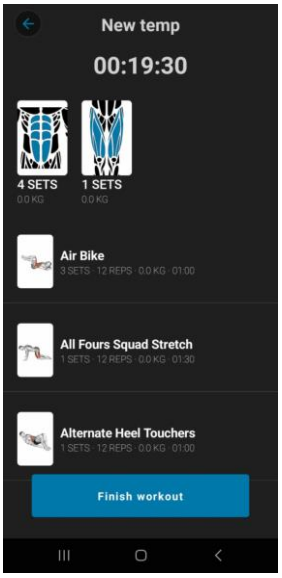
Слика 37 - Екран тренинга, са означеним бројем завршених серија



Слика 38 - Приказ паузе након завршене серије



Слика 39 - Приказ завршених и незавршених серија у вежби

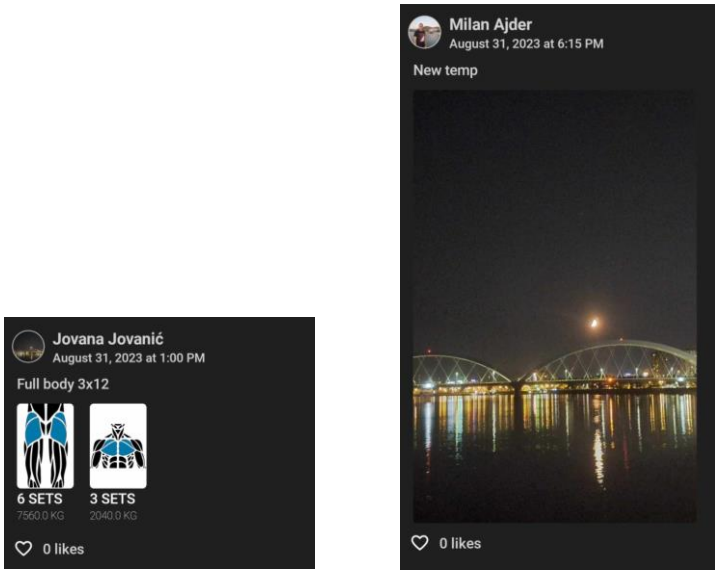


Слика 40 - Извештај о завршеним вежбама на тренингу

4.8 Објава

Након сваког завршеног тренинга, креира се ново објава. Објаву може да види њен оснивач и сви корисници који су повезани са оснивачем. Објаве се налаза на профилу корисника и у новостима.

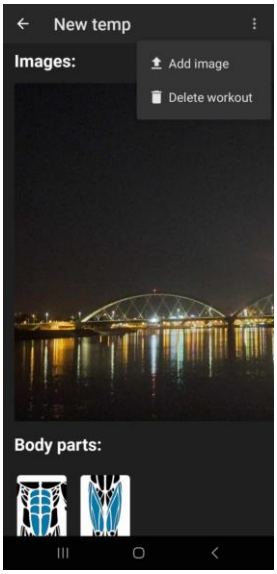
Објава се састоји из слике, имена и презимена оснивача објаве, времена оснивања, назива шаблона вежбања, слика и броја свиђања објаве. Уколико оснивач није додао ни једну слику на објаву, уместо слика се на објави виде мишићне групе које су биле најактивније у тренингу. Ако јесте додао слике, онда се те слике виде на објави (Слика 41). Корисник може да означи да му се свиђа нека објава притиском на иконицу у облику срца.



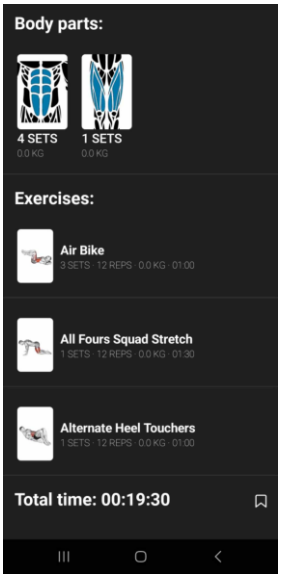
Слика 41 - Објава без слике и објава са сликом

Кликом на објаву корисник се отвара екран са детаљним приказом објаве (Слика 42). Корисник може да види назвив шаблона, слике са објаве, мишићне групе које су биле активне на тренигу, вежбе које се могу и детаљно погледати и време трајања тренинга. Такође, корисник може да сачува шаблон кликом на иконицу у доњем десном ћошку (Слика 43). На тај начин се креира копија тог шаблона и додаје се међу листу шаблона корисника. Након успешног чувања новог шаблона, кориснику се отвара списак шаблона са новим шаблоном на врху, као последње додатим.

Ако је корисник оснивач објаве, у горњем десном углу детаљног приказа може да кликне три тачкице и тиме отвори падајући мени са опцијама додавања слика и брисања објаве. Кликом на додавање слике отвара му се *BottomSheet* који садржи у себи галерију телефона. Када корисник изабрере слику, галерија се затвара и додаје објави. Објаве се додају у *Firebase Storage* и *Firestore* исто као и профилне слике. Свака објава има списак слика које садржи. Слике се могу уклонити са објаве које је корисник оснивач. Уколико корисник дуже држи на слику појављује се поље падајућег менија за брисање слике.



Слика 42 - Детаљан приказ објаве - назив шаблона, слике и додавање слике и брисање објаве за оснивача објаве



Слика 43 - Детаљан приказ објаве - мишићне групе, вежбе, време тренинга и опција чувања шаблона

4.9 Новости

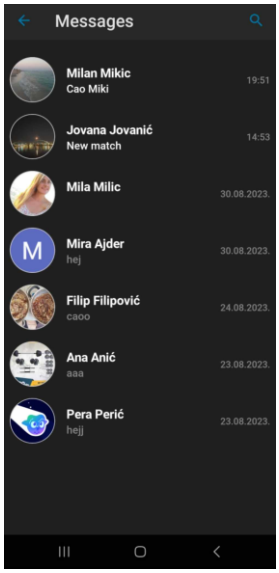
На екрану за новости се налазе објаве корисника и свих његових повезаних корисника, сортираних од новијих ка страијим (Слика 44). Објаве се учитавају тек када је неоходно да се прикажу (*LazyColumn*). У горњем десном ћошку екрана се налази иконица која води корисника до екрана за дописивање. Иконица садржи бец са бројем непрочитаних порука.

4.10 Дописивање

Кликом на иконицу у горњем десном ћошку екрана за новости отвара се екран са свим дописивањима корисника. Корисник је претплаћен на промене у својим дописивањима и када стигне нова порука, аутоматски се појави измена на корисничком интерфејсу (листинг). Корисник може да види кориснике са којима се дописивао, последњу поруку и датум/време последње поруке (Слика 45). Корисник може да

претражује повезане кориснике по иману и презимену. Кликом на неко од дописивања, корисник се преусмерава на детаље дописивања са изабраним корисником. Ако порука није била обележена као прегледана (ако је била болдована), уласком у детаље дописивања је означена као прегледана.

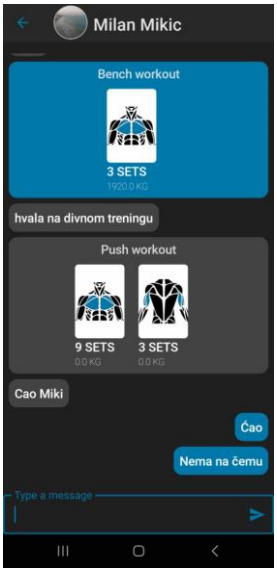
Слика 44 - Екран са новостима



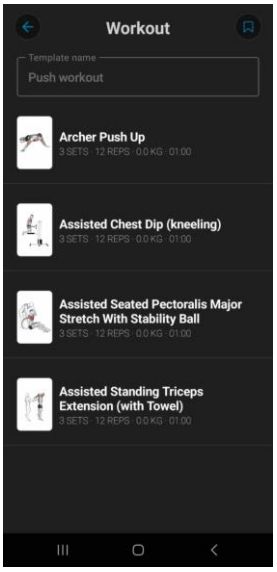
Слика 45 - Екран за дописивања

Детаљи дописивања се састоје из слике, имена и презимена саговорника, порука, прослеђених шаблона за вежбање и поља за унос порука (Слика 46). Порука се шаље путем поља за унос поруке. Шаблон се шаље са списка шаблона (1.1). Клик на податке о саговорник усмерава корисника на страницу профила саговорника.

Клик на подељени шаблон за вежбање усмерава корисника на страницу прегледа шаблона. Ова стараница не може да се мења. Може да се сачува притиском на дугме горе десно и тиме се корисник преусмерава на списак његових шаблона са новододатим шаблоном (Слика 47



Слика 46 – Екран детаља дописивања са корисником



Слика 47 - Екран за преглед послатог шаблона

5. ЗАКЉУЧАК

У овом раду је представљњна апликација за управљање тренинзима снаге кроз развој и имплементацију андроид апликације. Мотивација за решавање овог проблема произилази из потребе за ефикасним и приступачним начином организације тренинга снаге, омогућавајући корисницима да на најбољи начин побољшају своје физичко здравље и форму.

Примена интерфејса за повезивање људи је обогатила корисничко искуство и омогућила забаван начин за упознавање људи из околине који тренирају. Додатно, интеграција социјалних аспеката попут дељења објава и фотографија са тренинга додаје дубљу димензију апликацији.

Унапређења и проширења апликације су многобројна и укључују могућност коришћења *Kotlin Multiplatform* технологије за лакше адаптирање на друге платформе (нпр. *IOS*). Поред тога, тренутно решење зависи од *API*-ја за податке. Како је *API* нешто на шта немамо директан утицај и подложан је променама, а и кошта, било би добро наћи неко друго, дугорочније решење. Додатно, могуће је развијати различите начине интеракције међу корисницима, додатне алатке које обогаћују корисничко искуство и плаћање у апликацији.

Као крајњи резултат, ова апликација нуди иновативан приступ праћења тренинга и повезивања са другим корисницима. Потенцијално има велики утицај на начин на који се људи ангажују у спортским активностима и деле их са другима.

6. ЛИТЕРАТУРА

[1] Google. (2023). Develop for Android. https://developer.android.com/develop

[2] JetBrains. (2023). Kotlin. https://kotlinlang.org/docs/home.html

[3] Google. (2023). Jetpack Compose. https://developer.android.com/jetpack/compose

[4] Anupam Chugh. (2022). Android MVVM Design Pattern. https://www.digitalocean.com/community/tutorials/android-mvvm-design-pattern

[5] Nishchal Visavadiya. (2023). Implement Clean Architecture in Android. https://medium.com/android-dev-hacks/detailed-guide-on-android-clean-architecture-9eab262a9011

[6] Google. (2023). Firebase. https://firebase.google.com/

[7] Baeldung. (2021). Kotlin Java Interoperability. https://www.baeldung.com/kotlin/java-interoperability

[8] Denis Brandi. (2022). The “Real” Clean Architecture in Android: S.O.L.I.D. https://betterprogramming.pub/the-real-clean-architecture-in-android-part-1-s-o-l-i-d-6a661b103451

[9] Google. (2023). FirebaseAuth. https://firebase.google.com/docs/auth

[10] Alex Mamo. (2022). How to authenticate to Firebase using Google One Tap in Jetpack Compose? https://medium.com/firebase-developers/how-to-authenticate-to-firebase-using-google-one-tap-in-jetpack-compose-60b30e621d0d

[11] Google. (2023). Save simple data with SharedPreferences. https://developer.android.com/training/data-storage/shared-preferences

[12] Justin Mozley. (2023). ExerciseDB API.

https://rapidapi.com/justin-WFnsXH_t6/api/exercisedb

[13] Alex Styl. (2023). Compose Tinder Card. https://github.com/alexstyl/compose-tinder-card

[14] Geeks for geeks, prakhar7. (2022). Haversine formula https://www.geeksforgeeks.org/haversine-formula-to-find-distance-between-two-points-on-a-sphere/

[15] Dheeraj Singh Bhadoria. (2023). Mastering Location Permissions and UI in Jetpack Compose https://medium.com/@dheerubhadoria/mastering-location-permissions-and-ui-in-jetpack-compose-847cacd4be2e

[16] Google. (2023). Firebase Storage https://firebase.google.com/docs/storage

[17] Patryk & Patrick. (2023). Vico diagram library. https://github.com/patrykandpatrick/vico

[18] Claude Bouchard-a, Steven N. Blair-a, dr. William Haskell. (2002). Physical Activity and Health. https://books.google.rs/books?hl=sr&lr=&id=tO96DwAAQBAJ&oi=fnd&pg=PT22&dq=Physical+Activity+and+Health&ots=11G-YKubHX&sig=aF5RqEMoTu9UgowEwR3H4yhZkQw&redir_esc=y#v=onepage&q&f=false

[19] Fitbod. (2023). Fitbod. https://fitbod.me/

[20] Sworkit Health. (2023). Sworkit https://sworkit.com/

[21] Strava. (2023). Strava. https://www.strava.com/

[22] Meta. (2023). Instagram. https://www.instagram.com/

[23] Match Group. (2023). Tinder. https://tinder.com/

БИОГРАФИЈА

Милан Ајдер, рођен 20.05.2000. у Новом Саду, где завршава првих 6 разреда у Основној школи „Петефи Шандор“, а 7. и 8. разред у гимназији „Јован Јовановић Змај“, у одељењу ученика обдарених за математику. Средњошколско образовање наставља у гимназији „Јован Јовановић Змај“ на смеру ученика обдарених за математику. По завршетку средње школе, уписује смер Софтверско инжењерство и информационе технологије на Факултету техничких наука у Новом Саду, где студира од 2019. године. Положио је све испите предвиђене планом и програмом и стекао услов за одбрану завршног рада.

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:	
Идентификациони број, **ИБР**:	
Тип документације, **ТД**:	монографска публикација
Тип записа, **ТЗ**:	текстуални штампани документ
Врста рада, **ВР**:	дипломски рад
Аутор, **АУ**:	Милан Ајдер
Ментор, **МН**:	проф. др Горан Савић
Наслов рада, **НР**:	Мобилна апликација за управљање тренинзима снаге
Језик публикације, **ЈП**:	српски
Језик извода, **ЈИ**:	српски / енглески
Земља публиковања, **ЗП**:	Србија
Уже географско подручје, **УГП**:	Војводина
Година, **ГО**:	2023
Издавач, **ИЗ**:	ауторски репринт
Место и адреса, **МА**:	Нови Сад, Факултет техничких наука, Трг Доситеја Обрадовића 6
Физички опис рада, **ФО**:	7 / 68 / 0 / 2 / 47 / 0 / 0
Научна област, **НО**:	Софтверско инжењерство и информационе технологије
Научна дисциплина, **НД**:	Софтверско инжењерство
Предметна одредница / кључне речи, **ПО**:	Андроид, Jetpack Compose, Kotlin
УДК	
Чува се, **ЧУ**:	Библиотека Факултета техничких наука, Трг Доситеја Обрадовића 6, Нови Сад
Важна напомена, **ВН**:	
Извод, **ИЗ**:	Овај рад се бави развојем Android апликације за управљање тренинзима снаге и повезивањем корисника путем Kotlin програмског језика. Апликација је имплементирана користећи MVVM архитектонски образац са Clean Architecture принципима, док је кориснички интерфејс развијен користећи Jetpack Compose алат.
Датум прихватања теме, **ДП**:	
Датум одбране, **ДО**:	
Чланови комисије, **КО**:	
председник	
члан	

ментор	проф. др Горан Савић
	Потпис ментора

KEY WORDS DOCUMENTATION

Accession number, **ANO**:	
Identification number, **INO**:	
Document type, **DT**:	monographic publication
Type of record, **TR**:	textual material
Contents code, **CC**:	Bachelor thesis
Author, **AU**:	Milan Ajder
Mentor, **MN**:	Goran Savić, PhD, prof, FTN Novi Sad
Title, **TI**:	A mobile application for managing strength training
Language of text, **LT**:	Serbian
Language of abstract, **LA**:	Serbian / English
Country of publication, **CP**:	Serbia
Locality of publication, **LP**:	Vojvodina
Publication year, **PY**:	2023
Publisher, **PB**:	author's reprint
Publication place, **PP**:	Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6
Physical description, **PD**:	7 / 68 / 0 / 2 / 47 / 0 / 0
Scientific field, **SF**:	Software Engineering and Information Technologies
Scientific discipline, **SD**:	Software Engineering
Subject / Keywords, **S/KW**:	Android, Jetpack Compose, Kotlin
UDC	
Holding data, **HD**:	Library of the Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad
Note, **N**:	
Abstract, **AB**:	This thesis deals with the development of an Android application for managing strength training and connecting users. It is developed in the Kotlin programming language. The application was implemented using the MVVM architectural pattern with Clean Architecture principles, while the user interface was developed using the Jetpack Compose tool.
Accepted by sci. Board on, **ASB**:	
Defended on, **DE**:	
Defense board, **DB**:	
president	
member	
mentor	Goran Savić, PhD, prof, FTN Novi Sad
	Mentor's signature