Mobilna aplikacija za kreiranje treninga snage

Pintarić, Fran

VELEUČILIŠTE U VIROVITICI

Stručni prijediplomski studij Računarstva

FRAN PINTARIĆ

MOBILNA APLIKACIJA ZA KREIRANJE TRENINGA SNAGE

ZAVRŠNI RAD

VIROVITICA, 2023

VELEUČILIŠTE U VIROVITICI

Stručni prijediplomski studij Računarstva

MOBILNA APLIKACIJA ZA KREIRANJE TRENINGA SNAGE

ZAVRŠNI RAD

Predmet: Projektiranje Informacijskih Sustava

Mentor: Student:

Marko Hajba, mag. math., pred. Fran Pintarić

VIROVITICA, 2023

Veleučilište u Virovitici

Stručni prijediplomski studij Računarstva - Smjer Programsko inženjerstvo

OBRAZAC 1b

ZADATAK ZAVRŠNOG RADA

Student/ica: **PINTARIĆ FRAN** **JMBAG:** **0307017169**

Imenovani mentor: Marko Hajba, mag. math., pred.

Imenovani komentor: -

Naslov rada:

Mobilna aplikacija za kreiranje treninga snage

Puni tekst zadatka završnog rada:

Izraditi mobilnu aplikaciju koja služi za kreiranje treninga snage i praćenje napretka korisnika. Za izradu aplikacije potrebno je koristiti Java programski jezik, Retrofit, te Entity Framework, a za rad s bazom podataka Microsoft SQL server. Bazu je potrebno napuniti testnim podacima. Korisnici mogu biti registrirani korisnici i administratori. Mobilna aplikacija treba imati različite funkcionalnosti, ovisno o tipu korisnika koji ju koristi:

- Omogućiti registraciju i prijavu korisnika.
- Administratori brinu o održavanju funkcionalnosti aplikacije, primaju prijedloge i rješavaju probleme koje mogu prijaviti ostali korisnici. Također, imaju uvid u najbolje korisnike prema zadanom kriteriju.
- Registrirani korisnici mogu pregledavati dostupne vježbe po željenim kriterijima – npr. mišićnoj skupini, te kreirati trening pomoću njih. Moguće je dodijeliti masu zadanim vježbama kako bi se omogućilo praćenje napretka korisnika.
- Registrirani korisnici će imati uvid u osnovnu statistiku o svom napredovanju, npr. ukupni volumen treninga, najbolji rezultati i sl.

Osim programskog rješenja, u pisanom dijelu završnog rada opišite ukratko korištene tehnologije te detaljno opišite arhitekturu sustava i odabrane procese i/ili funkcije unutar same aplikacije. Prilikom opisivanja, osim neformalnih koristite i neke od formalnih metoda koje poznajete.

Datum uručenja zadatka studentu/ici: 31.07.2023.
Rok za predaju gotovog rada: 08.09.2023.

Mentor:

Marko Hajba, mag. math., pred.

M. Hajba

Dostaviti:
1. Studentu/ici
2. Povjerenstvu za završni rad - tajniku

MOBILNA APLIKACIJA ZA KREIRANJE TRENINGA SNAGE

Sažetak

Aplikacija je razvijena za Android platformu koristeći Android Studio i Visual Studio. Programski jezici C# i Java korišteni su tijekom razvoja. Ova aplikacija nudi opcije za registraciju i prijavu. Registrirani korisnik može pretraživati vježbe prema specifičnim kategorijama, kreirati vlastite treninge za jačanje tijela, pristupiti prethodno izrađenim treninzima te pratiti svoj napredak tijekom duljeg razdoblja. Administrator aplikacije ima zadatak dodavanja i brisanja vježbi i kategorija, te prati korisnike s najvećom dodanom težinom za svaku vježbu. Prilikom sastavljanja treninga, korisnik može odabrati specifične vježbe za uključivanje u rutinu, specificirati broj ponavljanja i serija za svaku vježbu, te unijeti željenu težinu za izvođenje vježbi. U slučaju problema, povratnih informacija ili posebnih zahtjeva, korisnicima je omogućeno kontaktiranje administratora putem posebnog obrasca. Nakon što administrator primi poruku, ima uvid u nju i može poduzeti odgovarajuće radnje prema vrsti poruke. Ovakav oblik komunikacije omogućuje daljnji razvoj aplikacije, pri čemu korisnička povratna informacija igra ključnu ulogu u prilagodbi i usavršavanju aplikacije kako bi bolje odgovarala potrebama korisnika. Glavna svrha ove aplikacije jest poticanje korisnika na zdraviji način života i pomoć u ostvarivanju njihovih fitness ciljeva.

Ključne riječi: *C#, Java, mobilna aplikacija, organizacija vježbi, SQL, trening snage*

MOBILE APPLICATION FOR STRENGTH TRAINING CREATION

Abstract

The application was developed for the Android platform using Android Studio and Visual Studio. The programming languages C# and Java were utilized during development. This application offers options for registration and login. Registered users can browse exercises by specific categories, create their own body-strengthening workouts, access previously created workouts, and track their progress over an extended period. The application administrator is responsible for adding and deleting exercises and categories, as well as monitoring users with the highest added weights for each exercise. When composing a workout, users can select specific exercises to include in their routine, specify the number of repetitions and sets for each exercise, and input the desired weight for performing the exercises. In case of issues, feedback, or special requests, users are enabled to contact the administrator through a dedicated form. Once the administrator receives a message, they have visibility into it and can take appropriate actions based on the type of message. This form of communication facilitates further development of the application, with user feedback playing a crucial role in adapting and improving the application to better meet user needs. The primary purpose of this application is to encourage users to lead a healthier lifestyle and assist them in achieving their fitness goals.

Keywords: *C#, Java, mobile application, strength training, SQL, workout organization*

Sadržaj

1. Uvod

U današnjem digitalnom dobu, mobilne aplikacije postaju neizostavni dio svakodnevne rutine, pružajući korisnicima olakšanje pri obavljanju različitih zadataka, pružajući zabavu i omogućujući globalno povezivanje. Mobilni uređaji koriste različite operativne sustave, pri čemu su Android i iOS najistaknutiji. Prema istraživanjima, Android je dominantan operativni sustav u svijetu s udjelom od 70,89%, dok se iOS nalazi na drugom mjestu s udjelom od 28,36%. Međutim, u Sjedinjenim Američkim Državama, iOS preuzima vodeću poziciju s udjelom od 57,39%, dok je Android prisutan na 42,27% tržišta [1].

Briga o vlastitom zdravlju i kondiciji treba biti u središtu pažnje svake osobe, a pristup visokokvalitetnim i korisnim alatima za praćenje tjelesnih aktivnosti igra ključnu ulogu u ostvarivanju fitness ciljeva. Aplikacije za praćenje tjelesnih aktivnosti postale su neizostavan dio svakodnevnog života, pružajući korisnicima potrebne informacije i motivaciju za postizanje vrhunskih rezultata. Unatoč mnogim takvim aplikacijama na tržištu, kao što su Garmin, Fitbit, Polar i druge, suočavaju se s određenim izazovima i ograničenjima. Neki su prekomplicirani za korištenje, dok drugi sadrže nepotrebne funkcionalnosti koje više ometaju nego pomažu u treningu. Fokus ove aplikacije je pružiti korisnicima obilje mogućnosti i funkcija koje će im olakšati stvaranje treninga i praćenje svog napretka.

2. Korišteni alati i tehnologije

Aplikacija je izrađena u Android Studio-u uz korištenje SQL baze podataka, uz podršku Visual Studio-a. Visual Studio je korišten kao *backend* za aplikaciju.

2.1. Android studio

Android Studio (slika 1.) predstavlja integrirano razvojno okruženje IDE (engl. Integrated development environment) koje je specijalizirano za razvoj mobilnih aplikacija na Android platformi. Ovo razvojno okruženje razvio je Google i postalo je centralni alat za programere diljem svijeta koji kreiraju aplikacije za različite Android uređaje, uključujući tablete, pametne telefone, pametne televizore i druge uređaje [3] [4].

Glavne značajke Android Studio-a uključuju:

1. Projektiranje korisničkog sučelja UI (engl. User interface): U okviru Android Studio-a, dostupan je grafički i XML (engl. Extensible Markup Language) uređivač koji omogućuje dizajniranje interaktivnog korisničkog sučelja. Programeri imaju mogućnost povlačiti i spuštati komponente, postavljati rasporede te prilagođavati izgled aplikacije.

2. Razvoj i dizajniranje: Android Studio pruža podršku za programski jezik Java i Kotlin, što omogućava razvoj bržih i učinkovitijih aplikacija. Također, u okviru ovog integriranog razvojnog okruženja dostupni su alati za debagiranje, praćenje performansi i optimizaciju koda

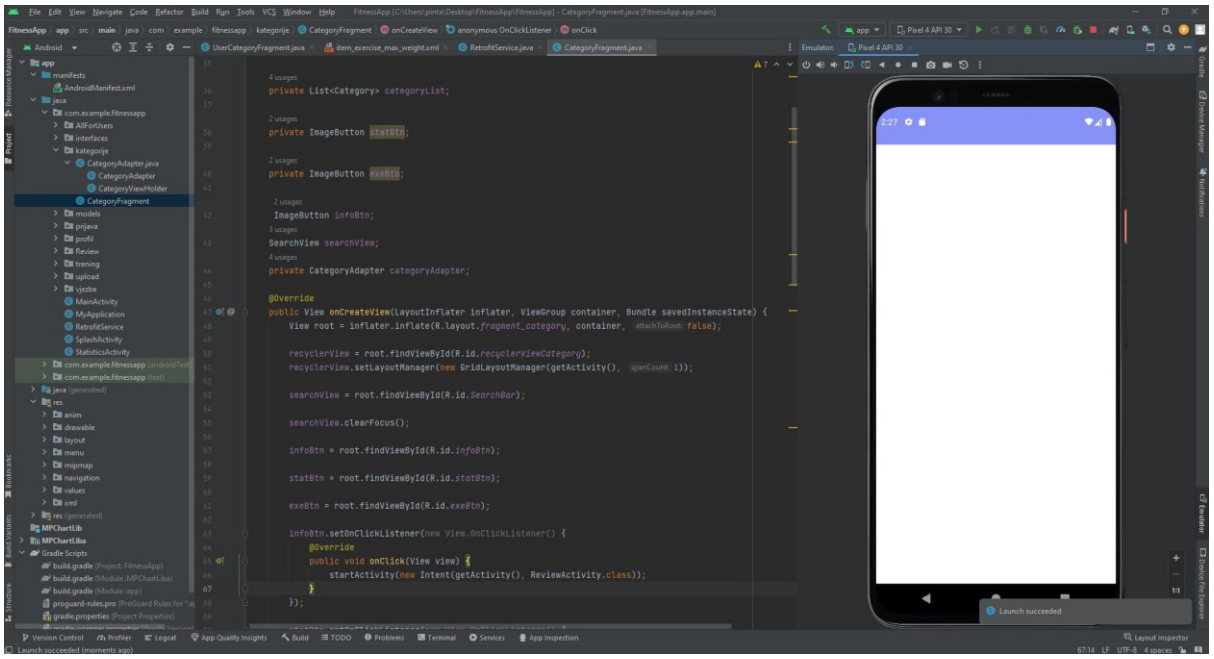
3. Upravljanje resursima: U okviru Android Studio-a, upravljanje resursima kao što su slike, ikone, nizovi i XML datoteke postaje jednostavno i praktično.

4. Emuliranje i testiranje: Android Studio nudi ugrađeni emulator koji omogućava testiranje aplikacija na različitim virtualnim uređajima s različitim verzijama Android operativnog sustava. Također, fizički uređaji se mogu koristiti za testiranje u stvarnim uvjetima.

5. Kreiranje i upravljanje *Gradle* projektima: U Android Studio-u, koristi se *Gradle* kao sustav za izgradnju i upravljanje projektima

6. Integraciju s Google servisima: Android Studio olakšava integraciju s različitim Google uslugama kao što su Firebase, Google Maps, Google Play Services i drugi.

7. Distribuciju aplikacije: Kada je aplikacija spremna za distribuciju, Android Studio olakšava generiranje APK (engl. Android Package Kit) datoteka koje se mogu objaviti na Google Play Store-u ili drugim platformama za aplikacije [5].



Slika 1. Izrada aplikacije i značajke Android Studio-a

Razlozi za korištenje Android Studio-a su:

- Android Studio preporučuje se kao IDE za razvoj Android aplikacija i dostupan je besplatno svima koji razvijaju profesionalne Android aplikacije. Temelji se na JetBrains IntelliJ IDE softveru, što može objasniti zašto je pretpregledna i beta verzija smatrana boljom od Eclipse-a. To je jedan od glavnih razloga zašto ga mnogi Android programeri koriste kao svoj IDE od samog početka [3].
- Prva stabilna verzija Android Studio-a objavljena je u prosincu 2014. godine i zamijenila je Eclipse kao primarni IDE za Android razvoj. Trenutačno, s Android Studio-om, ne samo da imam brži i stabilniji IDE, već također dobivam niz poboljšanih mogućnosti, uključujući korištenje *Gradle-a*, bolje metode refaktoriranja i bogatiji uređivač [3].

2.1.1. Životni ciklus aplikacije

Android aplikacije su kolekcije zadataka, pri čemu svaki zadatak ima svoju funkcionalnost. Funkcionalnosti se mogu podijeliti na četiri Android komponente:

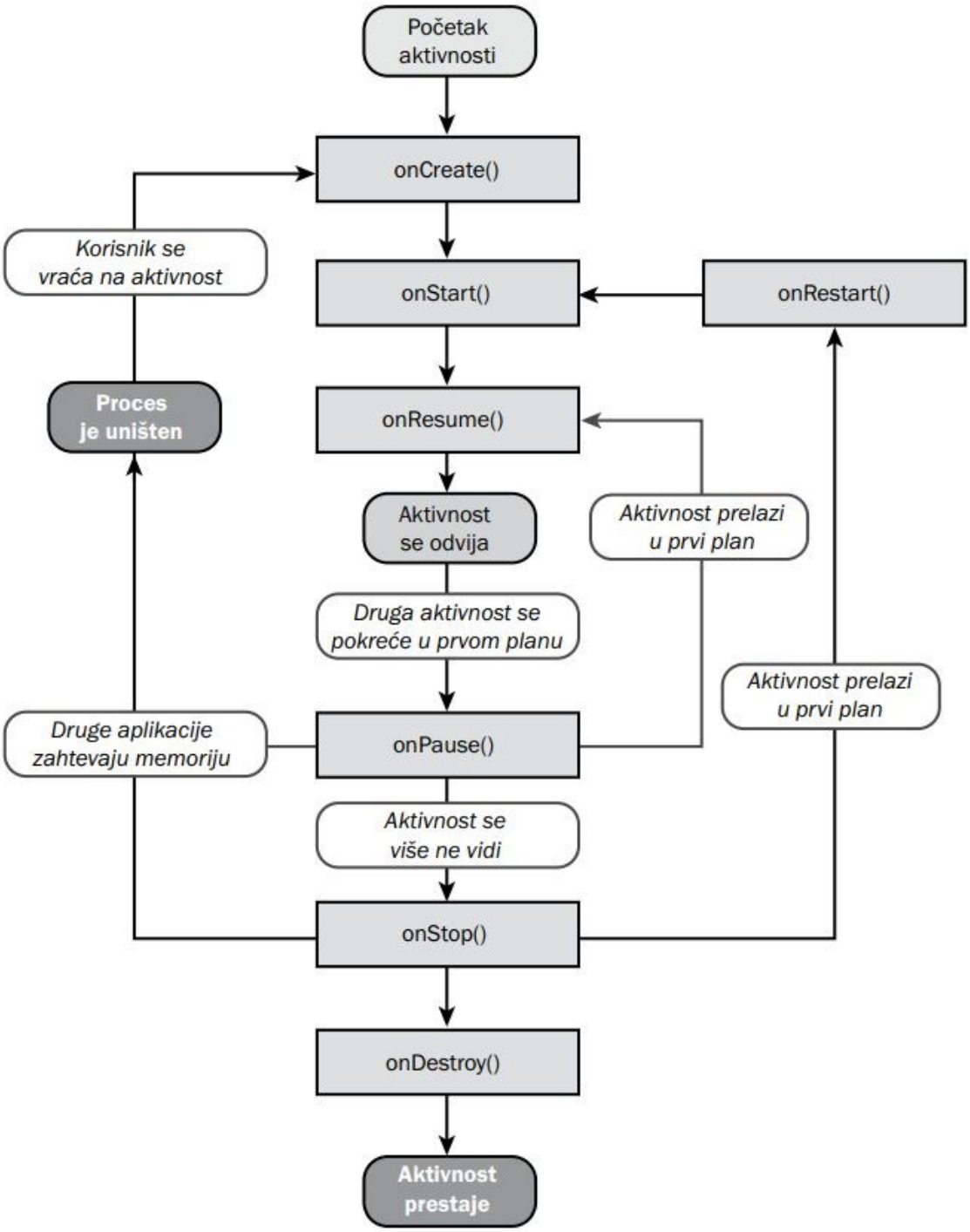
Activity – koristi se za interakciju s korisnicima

Service - predstavlja pozadinski proces koji može dugo trajati, čak i kada korisnik nije izravno u interakciji s aplikacijom. Servis se izvršava sve dok se ne zaustavi sam ili ga neko drugi zaustavi.

BroadcastReciever – koristi se za primanje poruka.

ContentProvider – koristi se za unošenje i očitavanje podataka.

Activity je jedan od načina putem kojeg Android omogućava funkcionalnost aplikacije i služi za interakciju s korisnicima. Svaki *activity* ima svoje korisničko sučelje. Da bi naša klasa predstavljala *activity* u aplikaciji, ona mora naslijediti klasu *„Activity“*. Najčešće se koristi klasa *„AppCompatActivity“,* koja je nasljednik klase *„Activity“* i obuhvaća razne funkcionalnosti. *Activity* je sam po sebi podklasa klase *„Context“,* što znači da svi *activityji* imaju pristup globalnim informacijama o okolini aplikacije. Ako aplikacija ima više *activityja,* uvijek se jedan od njih definira kao *„MainActivity“*. *MainActivity* predstavlja početni zaslon koji se prikaže korisniku pri pokretanju aplikacije. Od trenutka kada se *Activity* prikaže na zaslonu do trenutka kada se sakrije*, Activity* prolazi kroz različite faze svog životnog ciklusa (slika 2.).



Slika 2. Životni ciklus Activityja[6]

Metode životnog ciklusa:

- *onCreate()* – Ova metoda se izvršava prilikom prvog pokretanja aplikacije. Aplikacija nije vidljiva prije izvršenja ove metode i postaje vidljiva tek nakon njenog izvršenja.

- *onStart()* – Nakon završetka metode *onCreate()*, sučelje korisničkog sučelja postaje vidljivo i omogućuje interakciju s korisnikom, što pokreće ovu metodu. Ako je aplikacija već bila pokrenuta i zaustavljena, ova se metoda izvršava nakon završetka metode *onRestart().*
- *onPause()* – Ova metoda se poziva kada se aplikacija pauzira, ali prije nego što se potpuno zaustavi. Ako se aplikacija samo pauzira, nakon ove metode slijedi *onResume().* Ako se aplikacija zaustavi, slijedi *onStop().*
- *onResume()* – Nakon prvog pokretanja aplikacije, ova se metoda izvršava nakon metode *onStart(),* a kasnije se poziva nakon završetka metode *onPause()* svaki put kada se aktivnost nastavi.
- *onStop()* – Ova metoda se poziva kada aktivnost postane nevidljiva, obično zbog toga što je korisnik potisnuo aplikaciju u pozadinu ili neka druga aplikacija. Aplikacija se tada ne vidi, ali i dalje se izvršava u pozadini.
- *onRestart()* – Ova se metoda poziva prilikom ponovnog pokretanja aplikacije prije nego što postane vidljiva. Ova metoda slijedi nakon metode *onStop()* i izvršava se svaki put kad se aktivnost ponovno pokrene, osim prilikom prvog pokretanja aplikacije ili rotacije ekrana.
- *onDestroy()* – Metoda se poziva kada sistem odluči uništiti aplikaciju i prethodi joj metoda *onStop() [6].*

2.2. Programski jezik Java

Za razliku od mnogih drugih računalnih jezika čiji je utjecaj s godinama slabio, Javin je postao jači. Određeno vrijeme Java je bila na samom vrhu internetskog programiranja i očuvala je svoju poziciju s nizom novih verzija. Danas Java više nije prvi i najbolji izbor za razvoj web aplikacija, ali i dalje je pri samom vrhu. Jedan od razloga uspjeha Jave je njezina agilnost. Brzo se prilagodila promjenama u programskom okruženju i promjenama u načinu programiranja programera. Najvažnije je to da je Java prije nekoliko godina ne samo pratila trendove, nego ih i stvarala [7].

Od svog skromnog početka kao verzija 1.02, Java je privukla programere svojom prijateljskom sintaksom, objektno orijentiranim značajkama, upravljanjem memorijom i obećanjem prenosivosti [8].

Java programski jezik ima ogroman značaj zbog svoje objektno orijentirane prirode i neovisnosti o platformi. Karakterizira ga precizna i strukturirana sintaksu, stroga tipitacija podataka i visoka pouzdanost. Zahvaljujući velikoj količini biblioteka i aktivnoj zajednici programera, Java se uspješno koristi za razvoj mobilnih, web i desktop aplikacija diljem svijeta. Njezina funkcionalnost, jednostavno i dokazana trajnost čine je ključnim resursom za programere širom svijeta.

2.3. Entity Framework

Entity framework je objektno-relacijski *maper (ORM)* dizajniran za .NET programere kako bi pojednostavio rad s relacijskim podacima korištenjem objekata specifičnih za domenu (slika 3.). Ovaj alat omogućava programerima da rade s podacima specifičnim za njihovu domenu, kao što su kupci ili adrese kupaca, bez potrebe za rukovanjem osnovnim tablicama i stupcima u bazi podataka. *Entity Framework* efikasno apstrahira složene SQL upite i manipulacije podacima u obliku objekata, čime se smanjuje količina koda koju programer mora napisati.

Korisnik koristi *Entity Framework* tako što slijedi sljedeće korake:

1. Kreira MVC (engl. Model-View-Controller) web aplikaciju kao svoj projekt.

2. Postavi stil i izgled aplikacije prema svojim željama.

3. Instalira *Entity Framework* kao nužan paket u svom projektu.

4. Kreira podatkovni model koji će predstavljati entitete i njihove međusobne veze u aplikaciji.

5. Kreira *Database Context*, što je ključna komponenta za komunikaciju s bazom podataka i pristupanje podacima.

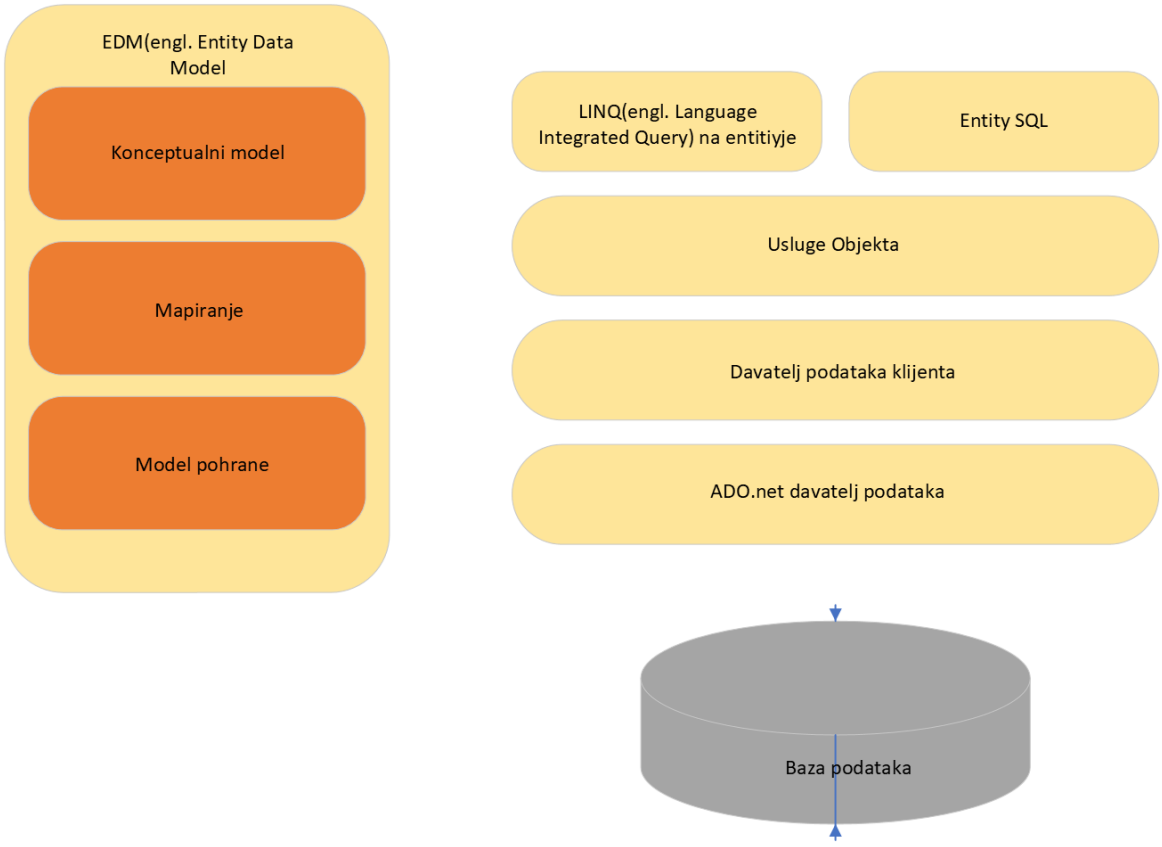
6. Inicijalizira bazu podataka testnim podacima kako bi mogao testirati funkcionalnosti.

7. Konfigurira *Entity Framework* da koristi odgovarajući baza podataka, kao što je LocalDB, SQL Server itd.

8. Kreira kontrolere i pripadajuće viewove koji će omogućiti interakciju korisnika s podacima.

Entity Framework donosi niz prednosti i značajki, uključujući*:*

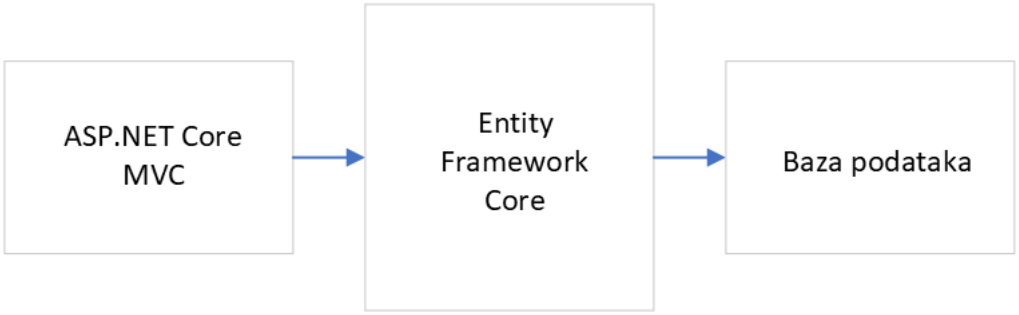
- Lagana i proširiva tehnologija objektno-relacijskog mapiranja
- Podrška za različite platforme, uključujući Windows, Linux i macOS.
- Mogućnost rada s raznim izvorima podataka, uključujući relacijske i ne relacijske baze podataka kao što su *SQL Server, SQLite*, *PostgresSQL, Azure Table Storage i IBM Data Server.*
- Omogućava programerima izvođenje operacija stvaranja, čitanja, ažuriranja i brisanja (engl. CRUD) podržavajući baze podataka. Također olakšava izvođenje unit testova pomoću memorije kao izvora podataka
- Pruža niz naredbi za migraciju koje se mogu pokrenuti iz *NuGet Package Manager* konzole ili sučelja naredbenog retka. Za stvaranje ili upravljanje osnovnom shemom baze podataka [11].



Slika 3. Arhitektura Entity Frameworka izrađeno prema[11]

2.3.1. Entity Framework Core

Entity Framework Core ima jednu ključnu zadaću, a ta zadaća je pohranjivanje .NET objekata u bazu podataka i njihovo kasnije ponovno dohvaćanje (slika 4.). Drugim riječima, *Entity Framework Core* djeluje kao spona između ASP.NET Core MVC aplikacije i baze podataka [12].



Slika 4. Entity Framework Core izrađeno prema[12]

Entity Framework Core je biblioteka koja omogućava pristup bazi podataka programerima. Kreiran je kao objektno-relacijski maper, što znači da povezuje dvije različite domene: relacijsku bazu podataka i objektno orijentirani softver koji uključuje klase i programski kod. Ovaj pristup omogućava brzo pisanje koda za pristup bazi podataka. Microsoft je objavio *Entity Framework Core* 2016. godine, a ovaj alat je dostupan na platformama kao što su Windows, Linux i Apple [10].

2.3.2. DbContext API

Od prvog izdanja, najkritičniji element u *Entity Frameworku* bio je *ObjectContext*. Ova klasa omogućuje interakciju s bazom podataka putem konceptualnog modela. *Context* omogućava izražavanje i izvršavanje upita, praćenje promjena na objektima te pohranjivanje tih promjena i slanje natrag u bazu podataka. *ObjectContext* komunicira s drugim ključnim klasama u *Entity Framework-u,* kao što je *ObjectSet,* koji omogućuje set operacije na *entityju* u memoriji, te *ObjectQuery*, koji je ključan za izvođenje upita. Sve ove klase obiluju različitim značajkama i funkcionalnostima, pri čemu su neke složene, dok se druge koriste samo za specifične scenarije. Nakon dvije nadogradnje (.NET 3.5 SP1 i .NET 4) postalo je jasno da su programeri najviše koristili samo nekoliko značajki. Nažalost, neke od značajki

potrebnih svakodnevne zadataka bile su teške za otkriti i koristiti. Tim *Entity Framework-a* je prepoznao ovaj problem i odlučio olakšati pristup najčešće korištenim uzorcima za rad s objektima unutar *Entity Frameworka*. Kao rješenje, predstavili su novi set klasa povezanih s *ObjectContext-om*. Iako ove nove klase koriste *ObjectContext* „iza scene“, programeri ih mogu koristiti bez potrebe za izravnom upotrebom *ObjectContext-a*, osim ako ne trebaju pristupiti nekim naprednim značajkama. Ovo poboljšanje je predstavljeno u *Entity Framework* 4.1, a ključne klase u ovom novom pristupu su *DbContext*, *DbSet* i *DbQuery*, poznate kao *DbContext* API. Ovaj novi API donosi više od samo klase *DbContext*, ali je ona glavna za upravljanje svim tim novim značajkama (slika 5.) [9].

DbContext API feature	Relevant EF4 feature/class	General purpose	Benefit of DbContext API
DbContext	ObjectContext	Represent a session with the database. Provide query, change tracking and save capabilities.	Exposes and simplifies most commonly used features of ObjectContext.
DbSet	ObjectSet	Provide set operations for entity types, such as Add, Attach and Remove. Inherits from DbQuery to expose query capabilities.	Exposes and simplifies most commonly used features of ObjectSet.
DbQuery	ObjectQuery	Provide querying capabilities.	The query functionality of DbQuery is exposed on DbSet, so you don't have to interact with DbQuery directly.
Change Tracker API	ObjectContext.ObjectStateManager	Get access to change tracking information and operations (e.g., original values, current values) managed by the context.	Simpler and more intuitive API surface.
Validation API	n/a	Provide automatic validation of data at the data layer. This API takes advantage of validation features already existing in .NET 4.	New to DbContext API.
Code First Model Building	n/a	Reads classes and code-based configurations to build in-memory model, metadata and relevant database.	New to DbContext API.

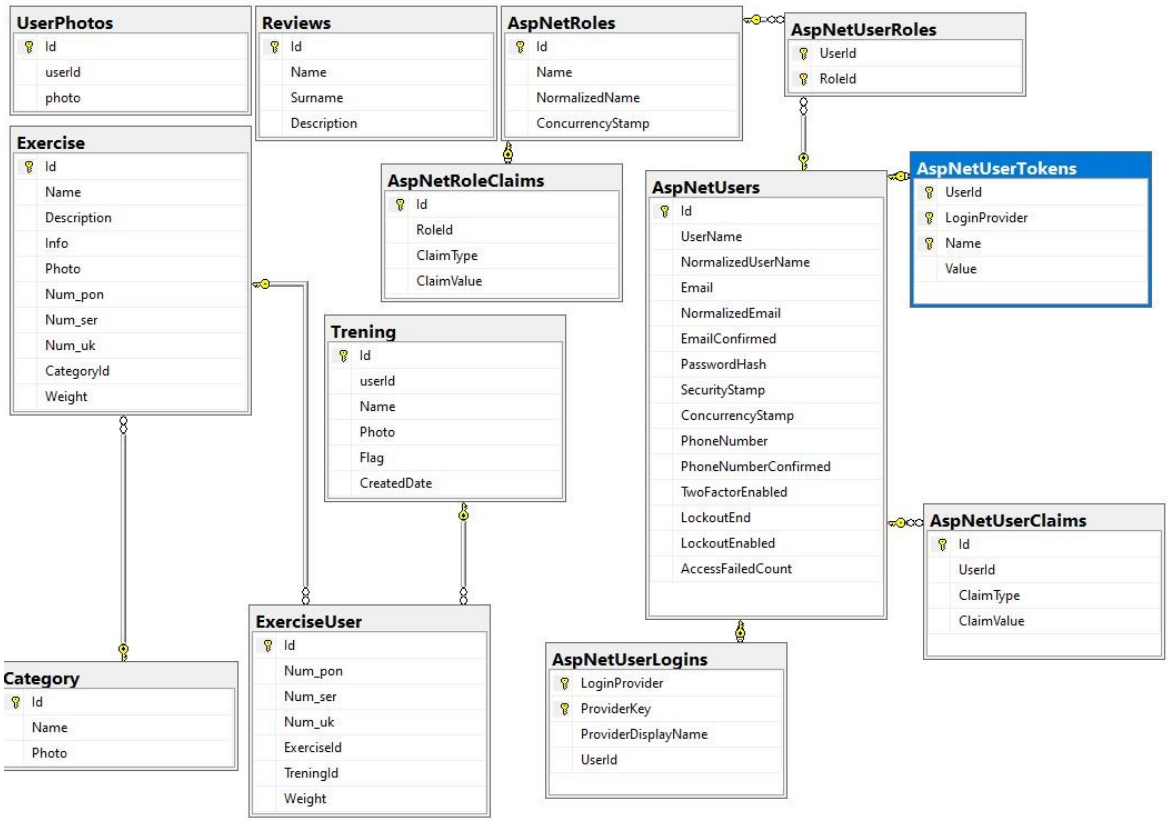
Slika 5. DbContext API značajke[9]

2.4. SQL baza podataka

SQL je industrijski-standardan jezik koji je posebno dizajniran kako bi omogućio ljudima stvaranje baze podataka, dodavanje novih podataka u bazu, održavanje podataka i dohvaćanje odabranih dijelova podataka. Postoje različite vrste baza podataka, a svaka od njih pridržava se različitog modela. SQL je izvorno razvijen za rad s podacima u bazama

podataka koje koriste relacijski model. Nadalje, SQL je uključio i komponentu objektnog modela, što je rezultiralo pojavom hibridnim struktura objektno-relacijskih baza podataka [12].

Neki od uobičajenih sustava za upravljanje relacijskim bazama podataka koji koriste SQL uključuju *Oracle*, *Sybase*, *Microsoft SQL Server*, *Acces itd.* Standardne SQL naredbe kao što su: „*Select*“, „*Insert*“, „*Update*“, „*Delete*“, „*Create*“ i „*Drop*“ mogu se koristiti za postizanje gotovo svih potrebnih operacija s bazom podataka [14]. Shema baze podataka korištena u ovoj aplikaciji prikazana je na slici 6.

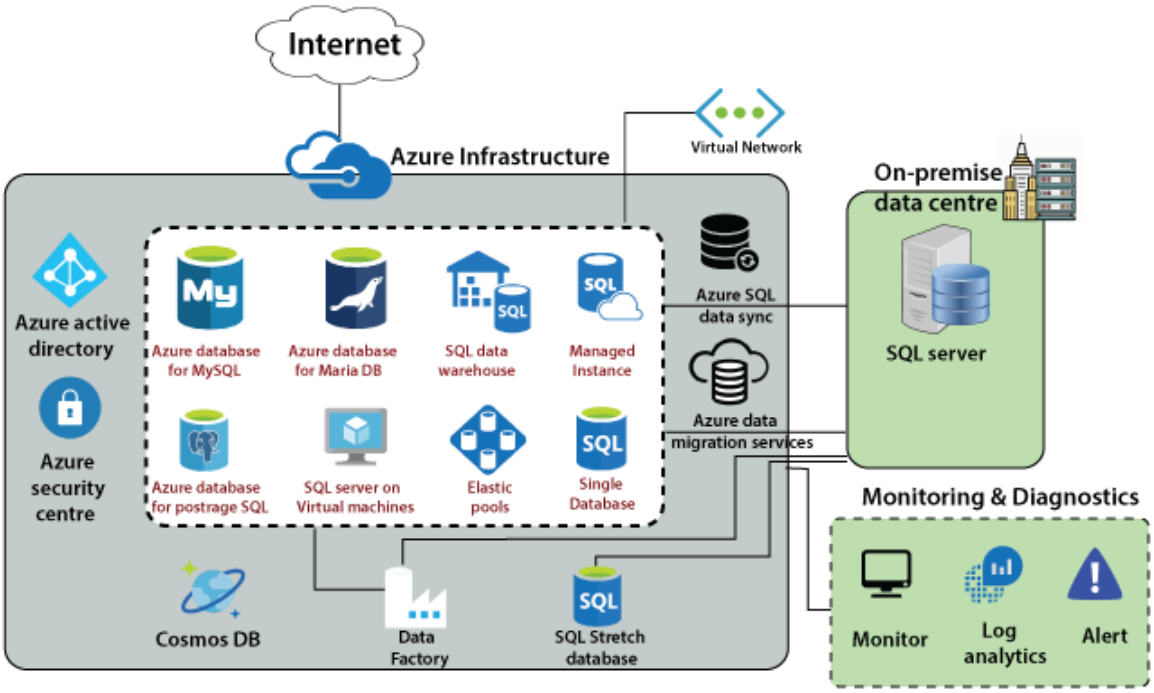


Slika 6. Shema baze podataka

2.5. Azure

Azure je javna platforma za IT u oblaku koja pruža niz rješenja, uključujući *IaaS (engl. Infrastructure as a Service), PaaS (engl. Platform as a Service) i SaaS (engl. Software as a Service)*. Ova platforma omogućuje korisnicima pristup različtim uslugama kao što su analitika, virtualno računarstvo, pohrana podataka, umrežavanje i mnoge druge. Azure se

može koristiti kao zamjene ili nadopunu lokalnim poslužiteljima. Na slici 7. prikazan je izgled Azure baze podataka.



Slika 7. Izgled Azure baze podataka [16]

3. Funkcionalnost aplikacije

Glavna funkcionalnost aplikacije FitnessApp sastoji se u stvaranju samostalnih treninga. Korisnicima se omogućuje pretraživanje kategorija i vježbi unutar svake kategorije. Nakon odabira vježbe, korisnici mogu pregledati detalje o njoj, uključujući sliku, opis, pripadnost određenoj kategoriji i upute o pravilnom izvođenju vježbe. Također, korisnici imaju opciju dodavanja vježbi u svoje treninge. Prilikom odabira vježbe, mogu specificirati broj ponavljanja i serija, te, ako je primjenjivo, težinu s kojom žele izvoditi vježbu.

Korisnicima se pruža mogućnost pregleda statistike njihovih kreiranih treninga tijekom različitih mjeseci i godina. Na taj način, mogu pratiti svoj napredak, uključujući najveću težinu koju su koristili tijekom vježbanja. Nakon što kreiraju trening, korisnici mogu pregledati detalje tog treninga i vidjeti koje vježbe su uključene.

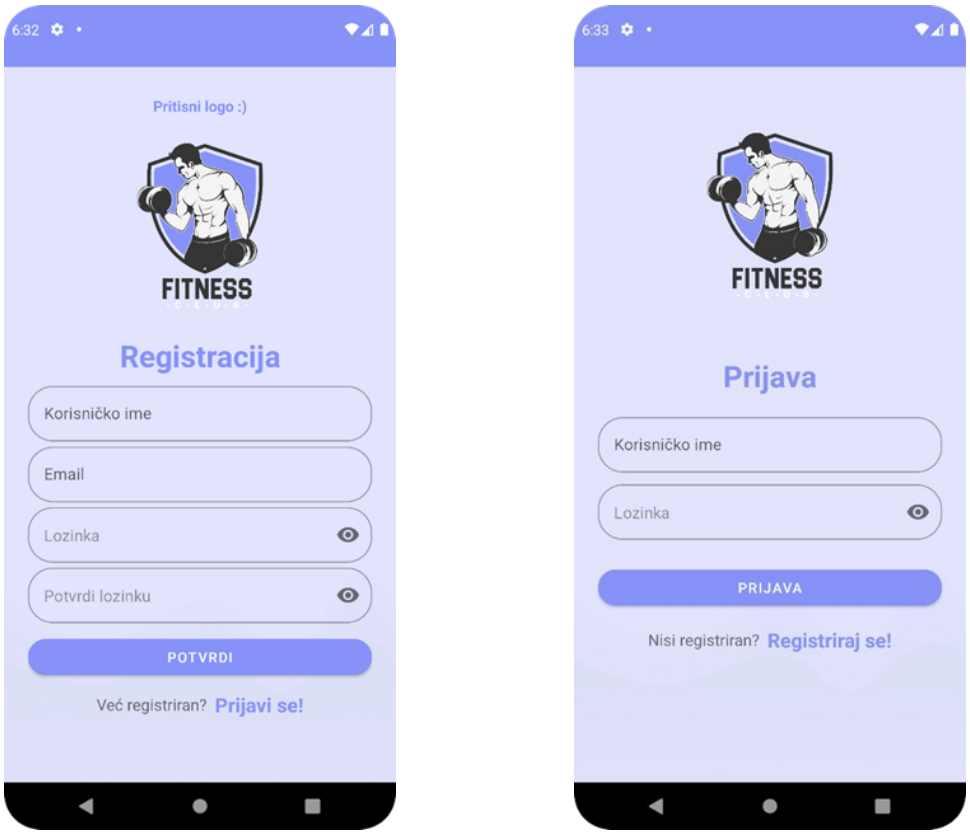
U slučaju problema ili pitanja, korisnici imaju opciju ispunjavanja kratkog obrasca kako bi obavijestili administratora, odgovornog za održavanje aplikacije. Administrator će pregledati te poruke i poduzeti odgovarajuće korake za rješavanje problema. Za razliku od običnih korisnika, administrator ima dodatne ovlasti koje uključuju dodavanje novih kategorija i vježbi, brisanje postojećih vježbi i kategorija, te uređivanje informacija o postojećim vježbama kako bi ažurirao njihove podatke.

3.1. Prijava i kreiranje računa

Prilikom prvog pokretanja aplikacije, korisnicima se prikazuje slika 8 (a). s obrascem za registraciju. Ovaj obrazac zahtijeva unos Korisničkog imena, E-maila, Lozinke i odabir fotografije. Nakon što korisnik unese sve potrebne podatke i pritisne tipku "Potvrdi", izvršava se proces stvaranja korisničkog računa. Ovi podaci se zatim spremaju u bazu podataka, a korisniku se prikazuje poruka da je registracija uspješna.

Proces provjere podataka prvo provjerava je li korisnik unio Korisničko ime. Ako korisnik nije unio Korisničko ime, dobit će poruku "Unesite korisničko ime". Ako je Korisničko ime ispravno uneseno, zatim se provjerava je li unesena Lozinka i ima li Lozinka barem šest znakova. U slučaju da neki od ovih uvjeta nisu zadovoljeni, korisnik će vidjeti poruku da treba unijeti lozinku ili da lozinka mora sadržavati najmanje 6 znakova. Slične provjere provode se i za potvrdu Lozinke, E-maila i slike. Nakon što su sva polja ispravno ispunjena, izvršava se metoda *registerUser().* Ova metoda je odgovorna za spremanje podataka o registraciji u bazu podataka i pozivanje API-ja. U slučaju uspješnog API zahtjeva, podaci se spremaju u bazu podataka, a korisniku se otvara početna stranica aplikacije. Ako API zahtjev nije uspješan, prikazuje se poruka da registracija nije uspjela. Ovdje se također vrši konverzija slike.

Forma za prijavu korisnika uključuje popunjavanje polja Korisničko ime i Lozinka. Prilikom prijave, provjerava se podudaraju li se uneseni podaci s računom u bazi podataka. Ako je korisnik registriran i uneseni podaci se podudaraju, korisniku se otvara početna stranica aplikacije. Proces prijave korisnika prvo izvršava metodu *validateData(),* u kojoj se provjerava je li korisnik popunio polja Korisničko ime i Lozinka. Ako su sva polja ispravno popunjena, pokreće se metoda *loginUser(),* u kojoj se provjerava odgovaraju li uneseni podaci s podacima iz baze podataka koji su se spremili tijekom registracije. Ako provjera uspije, izvršava se konačna provjera kako bi se utvrdilo je li korisnik običan korisnik ili administrator, jer postoje različite početne stranice za ove dvije uloge.

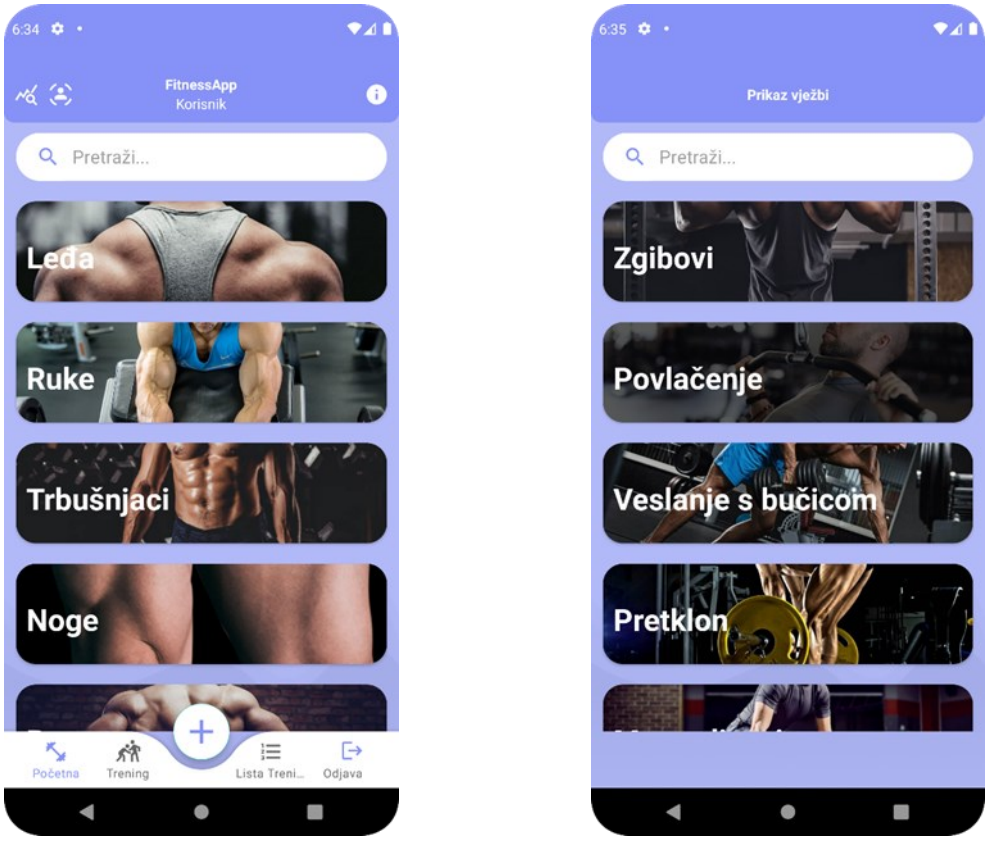


(a) **(b)**

Slika 8. Kreiranje računa (a) i Prijava korisnika u aplikaciju (b)

3.2. Početni zaslon i vježbe

Početni zaslon korisnika prikazuje kategorije vježbi (slika 9 (a)), tražilicu koja omogućuje korisniku pretragu kategorija po imenu, a na samom dnu zaslona izbornik sa četiri gumba koji omogućuje pristup različitim dijelovima aplikacije. Prvi gumb prikazuje početnu stranicu koja je uvijek prva stranica dostupna korisnicima, klikom na gumb trening korisnik otvara zaslon na kojem može potvrditi vježbe koje je dodao u svoj trening, gumb lista treninga vodi korisnika na popis njegovih kreiranih treninga, a gumb odjava omogućuje korisniku da se odjavi iz aplikacije. Na samom vrhu početnog zaslona nalaze se dodatni gumbi. Prvi gumb otvara statistiku kreiranih treninga, drugi gumb otvara statistiku koja prati najveću težinu odrađenu po pojedinoj vježbi koju je korisnik odradio, a treći gumb otvara novi zaslon u kojemu korisnik može ispuniti formu ako naiđe na neki problem u aplikaciji ili ako imamo neku pohvalu ili zahtjev. Odabirom neke od kategorija mišića, korisnik ulazi na izbornik vježbi, pri čemu svaka vježba ima svoju sliku koja reprezentira i demonstrira tu vježbu (slika 9 (b)). Daljnjim odabirom konkretne vježbe, otvara se zaslon koji prikazuje detalje odabrane vježbe.



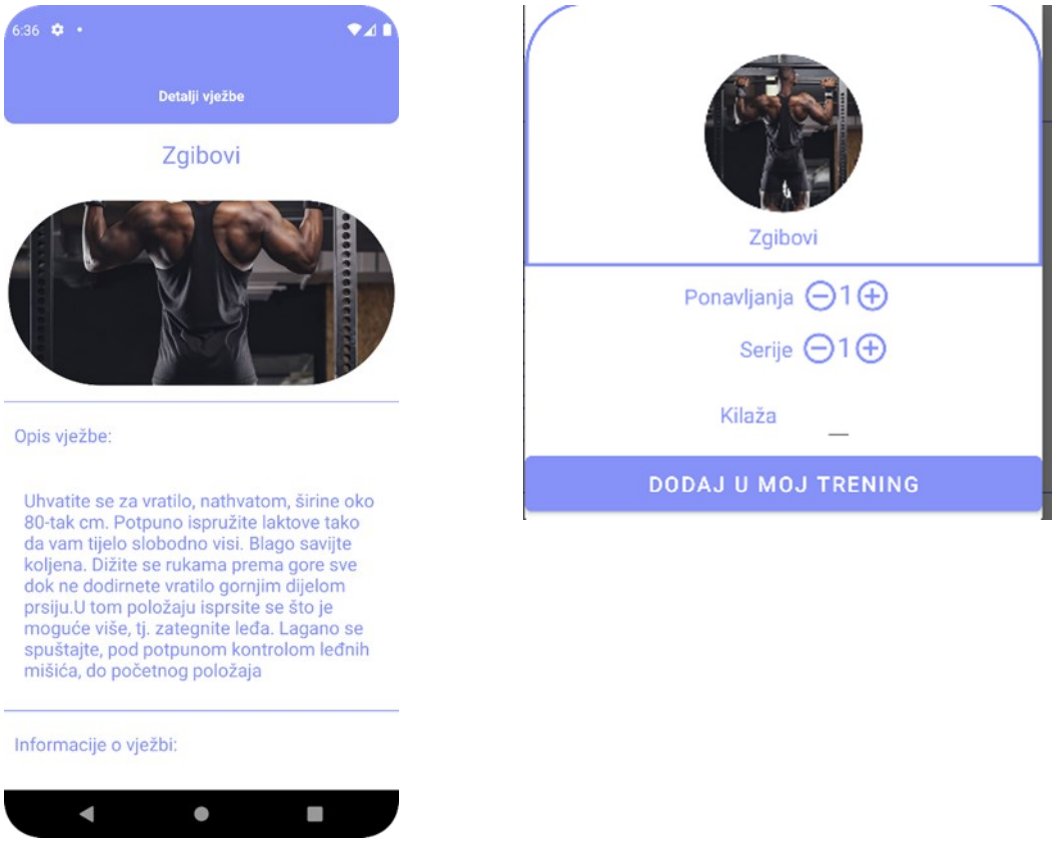
(a) **(b)**

Slika 9. Početni zaslon (a) i Zaslon vježbi (b)

3.3. Detalji vježbe

Zaslon prikazan na slici 10 (a). otvara se klikom na pojedinu vježbu i prikazuje detalje vježbe, unutar detalja vježbe korisnik može vidjeti sliku vježbe, opis vježbe, informacije vježbe i u koju kategoriju vježba spada. Na dnu zaslona nalazi se gumb koji, kad se pritisne, otvara dijalog prikazan na slici 10 (b).

Zaslon sa slike 10 (b). služi kako biste dodali odabranu vježbu u svoj trening, korisnik može odabrati koliko ponavljanja i serija želi odraditi s određenom vježbom, također ako vježba uključuje opterećenje može dodati sa koliko kilograma želi izvesti vježbu. Nakon što odabere ove parametre i pritisne gumb „Dodaj u moj trening“, vježba je dodana unutar treninga. Korisnik može provjeriti svoj odabir unutar trening fragmenta i na kraju potvrditi kreiranje treninga. Podatci se spremaju unutar objekta *exerciseUserRequesta*, a zatim se poziva metoda *addExerciseToTraining* koja prima te podatke i sprema ih u bazu podataka.



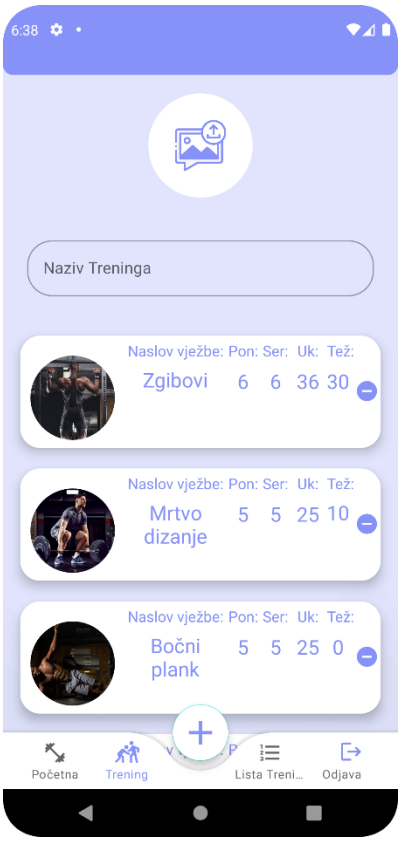
(a) **(b)**

Slika 10. Detalji vježbe (a) i Zaslon dodavanja vježbe u trening (b)

3.4. Trening

Konačna potvrda treninga i njegovo kreiranje prikazani su na slici 11. Korisnik treba unijeti naziv treninga i odabrati sliku treninga koja će predstavljati taj trening. Korisnik ovdje može još jednom provjeriti jesu li uključene sve željene vježbe i odgovara li broj ponavljanja i serija za svaku vježbu, te ako vježba uključuje kilažu, odgovara li kilaža onoj koju je odabrao prilikom dodavanja vježbe unutar treninga. Ako korisnik želi izbrisati neku vježbu iz treninga, to može učinit klikom na gumb koji se nalazi s desne strane ekrana pored težine vježbe. Ako je korisnik zadovoljan sa svim podacima i upisao je ime te postavio sliku i gumb „Potvrdi“ trening će biti izrađen i dodan u bazu podataka.

Ovo je zadnji korak gdje se trening sprema u bazu podataka uz pomoć metode *saveTrainingData()*.

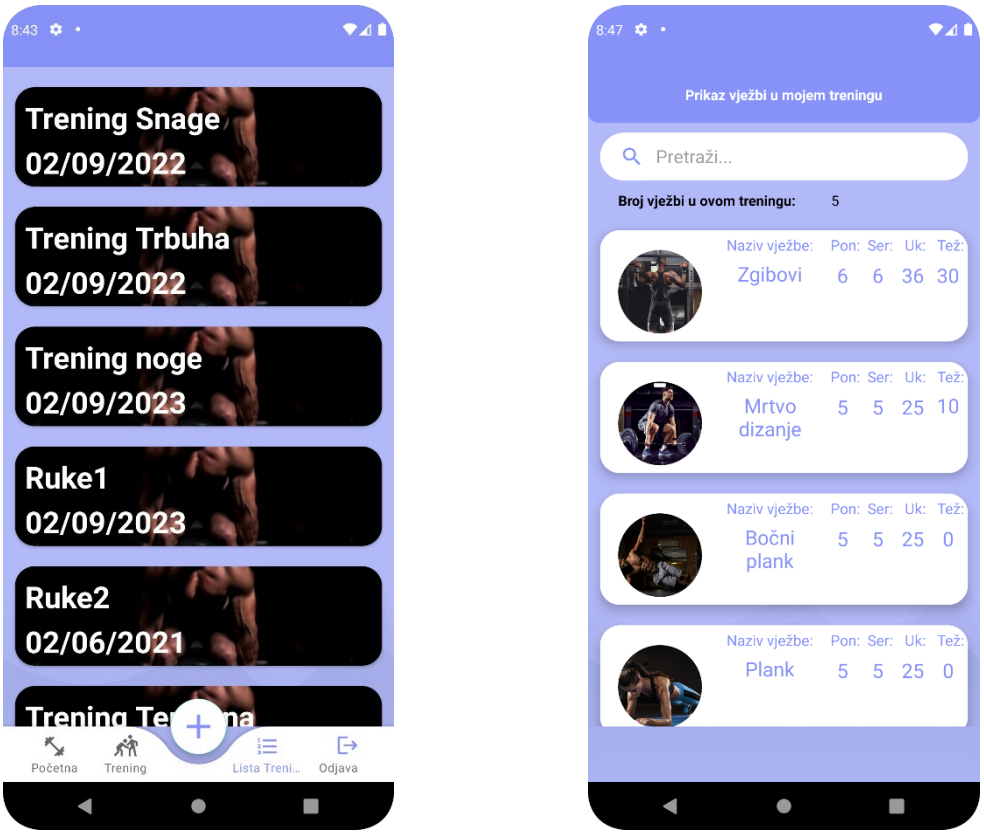


Slika 11. Kreiranje treninga

3.5. Lista treninga i detalji treninga

Lista treninga omogućava korisnicima pregled svih treninga koje su ikad izradili (slika 12 (a)). Ako korisnik želi provjeriti koje vježbe je dodao unutar treninga, jednostavno treba kliknuti na trening. Klikom na trening ulazi u detalje treninga, gdje su mu svi podaci prikazani kao na slici 12 (b).

Unutar detalja treninga korisnik može vidjeti koliko vježbi ima u treningu, ime vježbe, broj ponavljanja, broj serija, ukupan umnožak ponavljanja i serija, te može vidjeti s kojom kilažom je radio svaku vježbu. Ako postoji velik broj vježbi unutar jednog treninga korisnik može koristiti tražilicu kako bi brže pronašao željenu vježbu.



(a) **(b)**

Slika 12. Lista kreiranih treninga (a) i Detalji treninga (b)

Metoda *loadTrainingInfo()* koristi se dohvaćanju informacija o vježbama unutar treninga kako bi korisnik mogao vidjeti koje vježbe su uključene u određeni treninga. Ova metoda poziva API zahtjev, a ako je taj zahtjev uspješan, korisniku se prikazuju sve vježbe koje se nalaze unutar tog treninga. Metoda *searchList()* omogućava korisnicima pretrageu vježbi po imenu – Isječak programskog koda 1.

Isječak programskog koda 1. Metode loadTrainingInfo() i searchList()

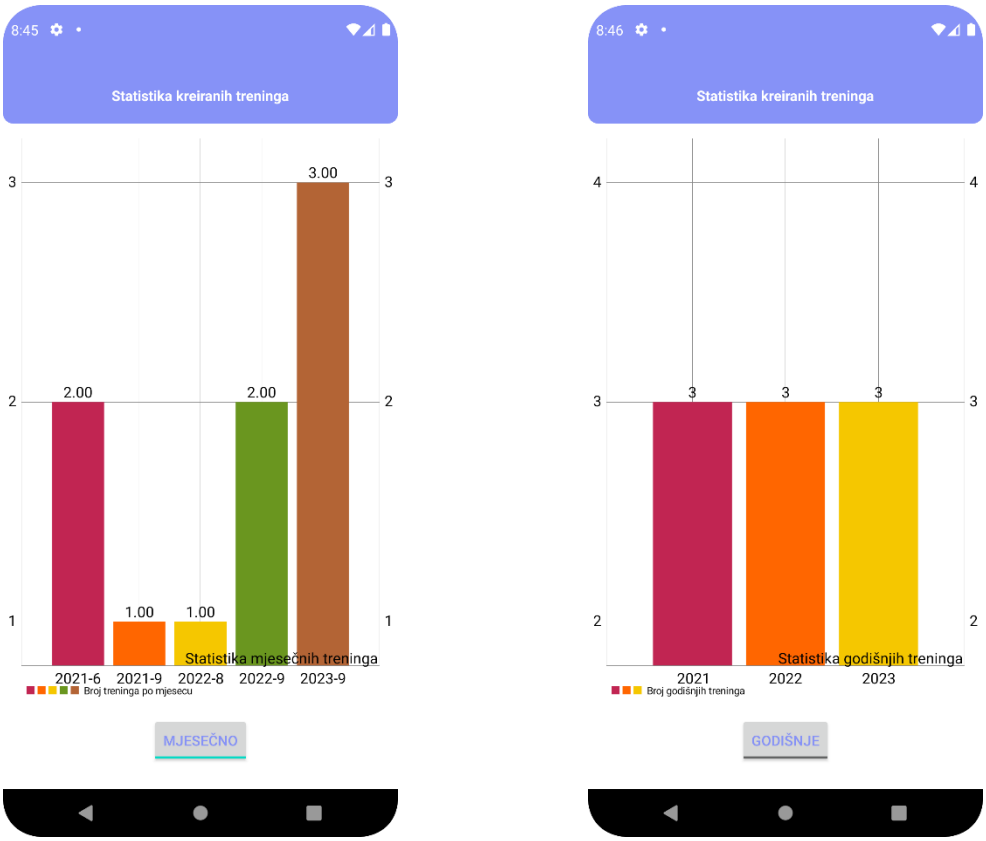
```
1. private void loadTrainingInfo()
 2.     {
 3.         trainingKreiranjeInterface =
RetrofitService.getClient().create(TrainingKreiranjeInterface.class);
 4.         Call<ModelTraining> call =
trainingKreiranjeInterface.getVjezbeUTreningu(trainingId);
 5.     call.enqueue(new Callback<ModelTraining>() {
 6.             @Override
```

```
 7.             public void onResponse(Call<ModelTraining> call, Response<ModelTraining>
response)
 8.     {
 9.         if (response.isSuccessful())
10.         {
11.             ModelTraining training = response.body();
12.             if (training != null)
13.             {
14.                 modelCreatedTrainingArrayList = new ArrayList<>(training.getVjezbe());
15.                 adapter = new TrainingExerciseAdapter(TrainingListDetails.this,
modelCreatedTrainingArrayList, trainingId);
16.                 recyclerView.setAdapter(adapter);
17.                 broj_vjezbi.setText(String.valueOf(modelCreatedTrainingArrayList.size()));
18.             }
19.         }
20.     }
21.
22.     @Override
23.             public void onFailure(Call<ModelTraining> call, Throwable t)
24.     {
25.         // Handle failure
26.     }
27. });
28.     }
29.
30.     public void searchList(String text)
31. {
32.     ArrayList<ExerciseUser> searchList = new ArrayList<>();
33.     for (ExerciseUser exerciseUser : modelCreatedTrainingArrayList) {
34.     if (exerciseUser.getExercise().getName().toLowerCase().contains(text.toLowerCase()))
35.     {
36.         searchList.add(exerciseUser);
37.     }
38. }
39. adapter.searchDataList(searchList);
40.     }
```

3.6. Statistika i pritužbe

Slike 13 (a) i (b). prikazuju statistiku kreiranih treninga u aplikaciji. Statistika uključuje informacije o broju kreiranih treninga svih korisnika aplikacije, što omogućava korisnicima uvid u opseg korištenja aplikacije. Važno je napomenuti da korisnici ne mogu vidjeti detalje o ostalim korisnicima, statistika se odnosi samo na ukupan broj treninga unutar

aplikacije. Korisnik može odabrati želi li pregledati godišnju statistiku ili mjesečnu statistiku. Ova opcija omogućava korisnicima da dobiju bolji uvid u promjene u korištenju aplikacije tijekom određenog vremenskog razdoblja.



(a) **(b)**

Slika 13. Statistika mjesečnih treninga (a) i Statistika godišnjih treninga (b)

Metoda *displayYearlyStatistics()* kreira graf godišnjih treninga. U ovoj metodi možemo podesiti boje grafova, veličinu slova i još brojne druge funkcionalnosti grafa. Ova metoda omogućava prikazivanje i analizu godišnjih statistika o broju kreiranih treninga u aplikaciji – Isječak programskog koda 2.

Isječak programskog koda 2. Metoda displayYearlyStatistics()

```
1. private void displayYearlyStatistics(List<YearlyTrainingStatistic> statistics)
2.     {
3.         List<BarEntry> entries = new ArrayList<>();
4.         List<String> labels = new ArrayList<>();
5.
6.         for (int i = 0; i < statistics.size(); i++)
7.         {
8.             YearlyTrainingStatistic stat = statistics.get(i);
9.             entries.add(new BarEntry(i, stat.getCount()));
10.            labels.add(String.valueOf(stat.getYear()));
11.        }
12.
13.        BarDataSet barDataSet = new BarDataSet(entries, "Broj godišnjih treninga");
14.        barDataSet.setColors(ColorTemplate.COLORFUL_COLORS); // You can set colors as
desired
15.
16.        BarData barData = new BarData(barDataSet);
17.        barData.setValueTextSize(15f);
18.
19.        XAxis xAxis = barChart.getXAxis();
20.        xAxis.setValueFormatter(new IndexAxisValueFormatter(labels));
21.        xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
22.        xAxis.setGranularity(1);
23.        xAxis.setLabelCount(labels.size());
24.        xAxis.setTextSize(15f);
25.
26.        YAxis leftAxis = barChart.getAxisLeft();
27.        YAxis rightAxis = barChart.getAxisRight();
28.
29.        leftAxis.setTextSize(15f); // Set the text size as desired
30.        leftAxis.setGranularity(1.0f);
31.        leftAxis.setGranularityEnabled(true);
32.
```

Korisnik može vidjeti najbolje rezultate za svaku vježbu (slika 14.), a klikom na gumb može provjeriti je li najbolji od svih korisnika aplikacije. Takva vrsta praćenja omogućuje korisniku praćenje njegovog napretka.

Slika 14. Prikaz najvećih težina po vježbi

Isječak koda prikazuje provjeru najveće težine svih korisnika ili samo jednog koji je trenutno ulogiran – Isječak programskog koda 3.

Isječak programskog koda 3. Metoda fetchExerciseMaxWeights()

```
1.    private void fetchExerciseMaxWeights()
2.     {
3.         Call<List<ExerciseMaxWeight>> call;
4.         if (isAllUsers)
5.         {
6.             call = fitnessApi.getExerciseMaxWeights();
7.         }
8.         else
9.         {
10.
11.            call = fitnessApi.getExerciseMaxWeights(SingletonUser.getInstance().getId());
12.        }
```

Na slici 15. prikazana je komunikacija između korisnika i administratora. Korisnik ima mogućnost napisati problem ili prijedlog i nakon toga klikom na gumb „Spremi“ pohranjuje te informacije. Pohranjuju se podaci poput Korisničkog imena i Opisa problema ili prijedloga. Administrator ima mogućnost pregleda poruka koje je korisnik naveo putem ove forme. Važno je napomenuti da korisnik mora unijeti ispravno Korisničko ime kako bi mogao predati svoj problem ili zahtjev – Isječak programskog koda 4.

Slika 15. Forma za prijavu problema ili prijedloga

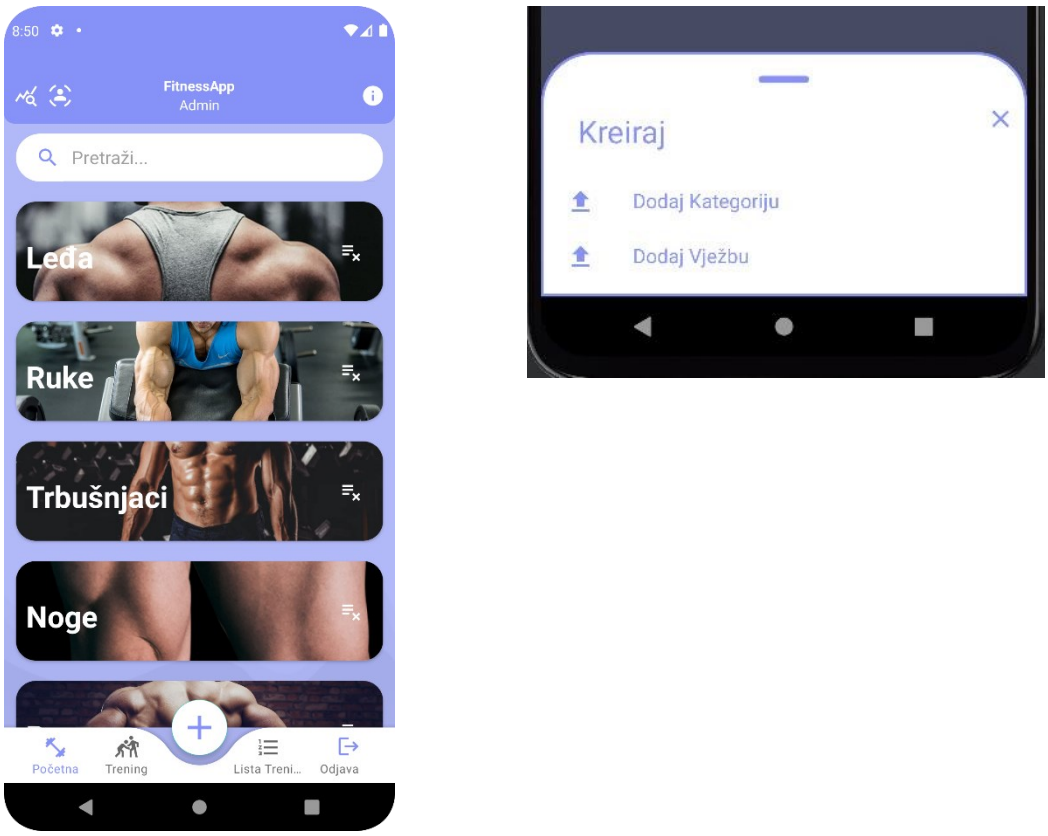
Isječak programskog koda 4. Metoda validateData()

```
1. public void validateData()
2.     {
3.         // Get data
4.         String ime = uploadIme.getText().toString().trim();
5.         String opis = uploadOpis.getText().toString().trim();
6.
7.         // Validate data
8.         if (TextUtils.isEmpty(ime))
9.         {
10.            Toast.makeText(UploadReviewActivity.this, "Upiši ime",
Toast.LENGTH_SHORT).show();
11.        }
12.        else if (!ime.equals(SingletonUser.getInstance().getUserName()))
13.        {
14.            Toast.makeText(UploadReviewActivity.this, "Netočno korisničko ime",
Toast.LENGTH_SHORT).show();
15.        }
16.
```

3.7. Razlika između administratora i korisnika

3.7.1. Početni zaslon

Razlika između običnog korisnika i administratora sastoji se u tome što administrator ima dodatne ovlasti. Administrator, na primjer, ima mogućnost brisanja kategorija. Također, klikom na veliki plus gumb na dnu ekrana otvara se dijalog prikazan na slici 16 (b). Unutar tog dijaloga, administrator ima opciju kreiranja nove kategorije ili nove vježbe. Forma za kreiranje nove vježbe prikazana je na slici 16 (b), dok je forma za kreiranje nove kategorije prikazana na slici 16 (a).

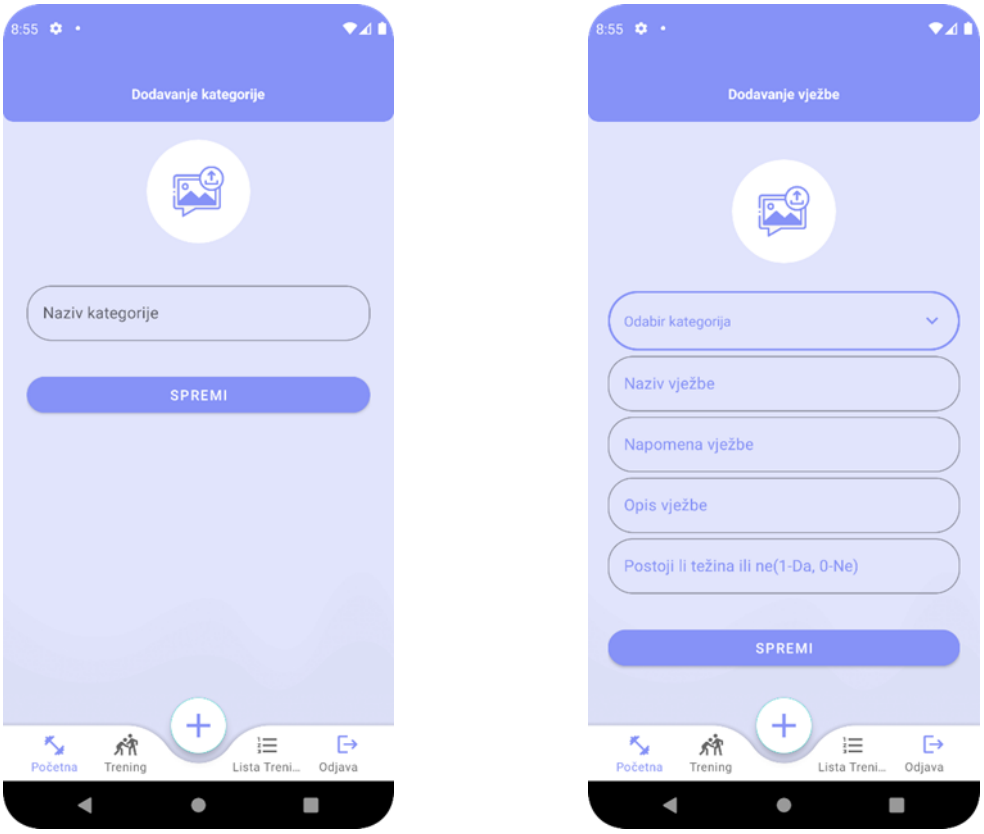


(a) **(b)**

Slika 16. Početni zaslon administratora (a) i Dijalog za kreiranje nove vježbe ili kategorije (b)

Administrator mora ispravno popuniti cijelu formu kako bi mogao dodati vježbu u bazu podataka. Ako ostavi neko polje prazno ili ne unese sve potrebne informacije, neće moći spremiti vježbu u bazu podataka. Forma za dodavanje vježbe uključuje sljedeće elemente: odabir kategorije, unos naziva vježbe, unos napomene o vježbi, opis vježbe, te

odabir postoji li težina za tu vježbu. Također, administrator mora dodati sliku koja reprezentira vježbu (slika 17 (b)). Prilikom dodavanja nove kategorije, administrator treba unijeti naziv kategorije i dodati sliku koja će biti povezana s tom kategorijom. Klikom na gumb "Spremi", informacije o kategoriji će biti pohranjene u bazu podataka (slika 17(a)).



(a) **(b)**

Slika 17. Forma za dodavanje kategorije (a) i Forma za dodavanje vježbe (b)

Metoda validateData() obavlja niz provjera kako bi osigurala da su svi potrebni podaci ispravno uneseni prilikom dodavanja nove vježbe. Ove provjere uključuju:

1. Provjeru je li unesen naziv vježbe.
2. Provjeru je li unesen opis vježbe.
3. Provjeru je li unesena napomena o vježbi.
4. Provjeru je li odabrana težina za vježbu.
5. Provjeru je li odabrana kategorija za vježbu.

6. Provjeru je li težina vježbe dostupna.

7. Provjeru je li odabrana slika vježbe.

Osim ovih provjera, na backend dijelu se također provjerava da naziv vježbe nije već zauzet kako bi se izbjeglo ponavljanje naziva vježbi. Ukoliko su sve ove provjere zadovoljene, vježba se šalje u bazu podataka koristeći funkciju *uploadExerciseInfoToDb()*.
-Isječak programskog koda 5. i 6.

Isječak programskog koda 5. Metoda validateData() za dodavanje vježbe

```
1. public void validateData()
2.     {
3.
4.         int ponavljanja = 0;
5.         int serije = 0;
6.         int ukupno = serije * ponavljanja;
7.
8.
9.         //get data
10.        String desc = uploadDesc.getText().toString().trim();
11.        String exerc = uploadExercise.getText().toString().trim();
12.        String nap = uploadNapomena.getText().toString().trim();
13.        String weight = uploadWeight.getText().toString().trim();
14.
15.        //validate data
16.        if (TextUtils.isEmpty(exerc))
17.        {
18.            Toast.makeText(getActivity(), "Upiši vježbu", Toast.LENGTH_SHORT).show();
19.        }
20.        else if (TextUtils.isEmpty(weight) || Integer.parseInt(weight) != 0 &&
Integer.parseInt(weight) != 1)
21.        {
22.            Toast.makeText(getActivity(), "Možete napisati samo 0 ili 1",
Toast.LENGTH_SHORT).show();
23.        }
24.        else if (TextUtils.isEmpty(desc))
25.        {
26.            Toast.makeText(getActivity(), "Upiši opis", Toast.LENGTH_SHORT).show();
27.        }
28.        else if (TextUtils.isEmpty(nap))
29.        {
30.            Toast.makeText(getActivity(), "Upiši informaciju", Toast.LENGTH_SHORT).show();
31.        }
32.        else if (TextUtils.isEmpty(selectedCategoryName))
```

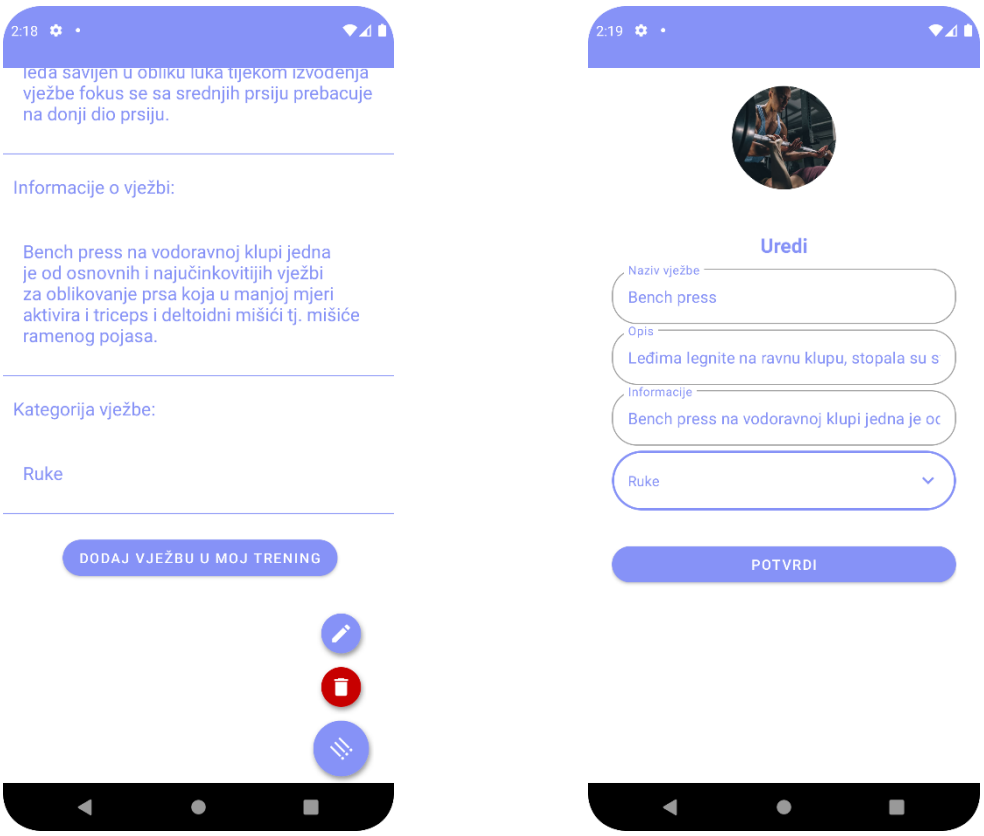
```
33.         {
34.             Toast.makeText(getActivity(), "Izaberi kategoriju",
Toast.LENGTH_SHORT).show();
35.         }
36.         else if (imageUri == null)
37.         {
38.             Toast.makeText(getActivity(), "Izaberite sliku ...",
Toast.LENGTH_SHORT).show();
39.         }
40.         else
41.         {
42.             uploadExerciseInfoToDb(exerc, desc, nap, ponavljanja, serije, ukupno,
Integer.parseInt(weight));
43.         }
44.
45.     }
```

Isječak programskog koda 6. Provjera imena vježbe

```
1. var existingExercise = await _context.Exercise.FirstOrDefaultAsync(e => e.Name ==
exerciseRequest.Name);
2.
3.             if (existingExercise != null)
4.             {
5.                 // Exercise with the same name already exists, return a conflict response
or handle it as needed
6.                 return Conflict("An exercise with the same name already exists.");
7.             }
8.
```

3.7.2. Detalji vježbe(uređivanje i brisanje)

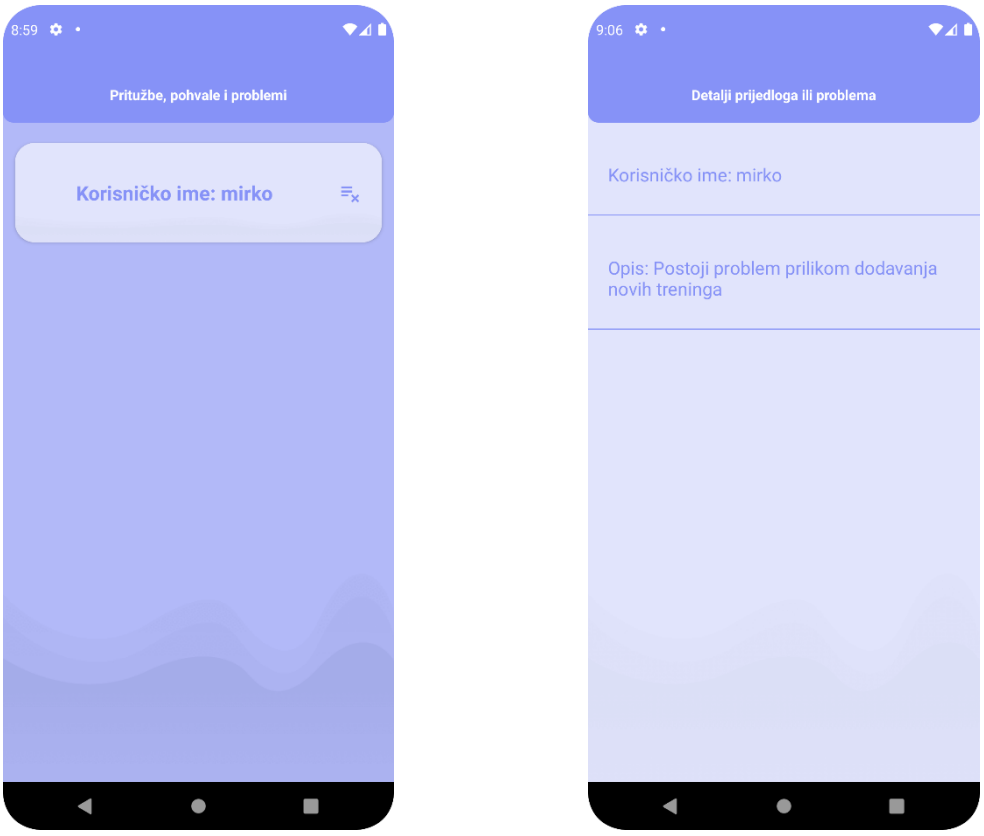
Administrator ima privilegije za brisanje i uređivanje vježbi unutar aplikacije. Klikom na gumb za brisanje (slika 18 (a)), vježba se trajno briše iz baze podataka, a klikom na gumb za uređivanje, otvara se nova forma (slika 18 (b)) kojom administrator može promijeniti informacije o određenoj vježbi. Ova mogućnost uređivanja omogućuje administratoru da ažurira podatke o vježbama kako bi ih prilagodio potrebama ili promjenama.



Slika 18. Detalji vježbe administratora (a) i Uređivanje vježbe (b)

3.7.3. Primanje pritužbi ili pohvala

Administrator ne mora popunjavati formu kako bi odgovorio na problem ili prijedlog korisnika. Umjesto toga, administrator prima Korisničko ime korisnika koji je ispunio formu (slika 19 (a)). Klikom na karticu otvaraju se detalji gdje ponovno piše Korisničko ime korisnika koji je ispunio formu, te se prikazuje opis problema ili prijedloga s kojim se korisnik suočio (slika 19 (b)). Nakon što administrator riješi problem ili razmotri prijedlog, može obrisati zahtjev korisnika kako bi očistio karticu i zadržao preglednost aplikacije.



Slika 19. Administrator poruke (a) i Detalji poruke (b)

Isječak programskog koda 7. Spremanje pritužbe na backendu s korisničkim imenom

```
var review = new Reviews
            {
                username = reviewRequest.username,
                userId = user.Id,
                Description = reviewRequest.Description
            };

            var entity = await _context.Reviews.AddAsync(review);
            await _context.SaveChangesAsync();
            return entity.Entity;

```

Isječak programskog koda 8. Dohvaćanje i zapisivanje korisničkog imena

```
1. String reviewName = reviewList.getUsername();
2.     String reviewDesc = reviewList.getDescription();
3.
4.
5.     //set data
6.     holder.revName.setText("Korisničko ime: " + reviewName);
7.
```

3.7.4. Prikaz najveće težine korisnika

Administrator ima mogućnost pregledati podatke o najvećoj maksimalnoj težini koju je svaki korisnik postigao po svakoj vježbi. Ova funkcionalnost omogućuje administratoru da prati napredak korisnika i utvrdi koji korisnik ima najbolje rezultate u određenim vježbama.

Slika 20. Prikaz najvećih težina po vježbama administrator

4. Zaključak

Svakodnevni život pojedinca je nezamisliv bez pametnih uređaja. U ovom radu predstavljena je aplikacija koja omogućava korisnicima da jednostavno kreiraju vlastite treninge. Proces dodavanja vježbi u trening olakšan je na način koji štedi vrijeme, dok korisnici nakon svakog treninga mogu pratiti svoj napredak. Aplikacija također promiče zdravu konkurenciju i zajednički cilj, budući da najbolji rezultati postaju vidljivi svim korisnicima. Očekuje se da će ovakav pristup korisnicima pružiti dodatnu motivaciju za kontinuirani rad na vlastitom fizičkom zdravlju i fitness ciljevima.

Uloge administratora u aplikaciji su ključne za njen rad i održavanje. Administrator ima mogućnost upravljati sadržajem aplikacije dodavanjem novih vježbi i kategorija te uređivanjem postojećih sadržaja kako bi osigurao svježinu i raznolikost. Osim toga, administrator je osoba koja prima povratne informacije od korisnika putem poruka, u kojima korisnici mogu prijaviti probleme u funkcioniranju aplikacije, predlagati poboljšanja ili izražavati pohvale. Ova razmjena informacija čini aplikaciju prilagodljivijom i usmjerava razvoj prema potrebama korisnika.

Predmet Projektiranje informacijskih sustava pomogao mi je u izradi aplikacije na način da sam unaprijedio svoje znanje korištenja alata kao što je Android Studio, te modernih tehnologija pri razvoju mobilnih aplikacija, poput .NET-a i Retrofita. Nova znanja i vještine planiram iskoristiti u daljnjem razvoju karijere i daljnjem vlastitom napretku.

Popis literature

[1] Bankmycell, Pristupljeno: 20. kolovoza 2023. [Online]. Dostupno na: https://www.bankmycell.com/blog/android-vs-apple-market-share/

[2] Bill Phillips, Chris Stewart, Brian Hardy and Kristin Marsicano (2015): Android Programming: The Big Nerd Ranch Guide, 2nd edition

[3] Mike van Drongelen (2015): Android Studio Cookbook

[4] MeetAndroidStudio, Pristupljeno: 21. kolovoza 2023. [Online]. Dostupno na: *https://developer.android.com/studio/intro*

[5] TechTarget, Pristupljeno: 25. kolovoza 2023. [Online]. Dostupno na: *https://www.techtarget.com/searchmobilecomputing/definition/Android-Studio*

[6] WebProgramiranje, Pristupljeno 25. kolovoza 2023. [Online]. Dostupno na: *https://www.webprogramiranje.org/aktivnost-u-okviru-android-operativnog-sistema/*

[7] Kathy Sierra, Bart Bates (2005): Head First Java, 2nd Edition

[8] Herbert Schildt (2007): Java: The Complete Reference, Seventh Edition

[9] Julia Lerman and Rowan Miller (2012): Programming Entity Framework: DbContext

[10] Adam Freeman (2018): Pro Entity Framework Core 2 for ASP.NET Core MVC

[11] SimpliLearn, Pristupljeno: 26. kolovoza 2023. [Online]. Dostupno na: *https://www.simplilearn.com/tutorials/asp-dot-net-tutorial/entity-framework-in-c-sharp*

[12] Jon P. Smith (2018): Entity Framework Core in Action

[13] Allen G. Taylor (2003): SQL For Dummies, 5th Edition

[14] SqlCourse, Pristupljeno: 27. kolovoza 2023. [Online]. Dostupno na: *https://www.sqlcourse.com/beginner-course/what-is-sql/*

[15] Cccbtechology, Pristupljeno: 27. kolovoza 2023. [Online]. Dostupno na: *https://ccbtechnology.com/what-microsoft-azure-is-and-why-it-matters/*

[16] Javapoint, Pristupljeno: 31. kolovoza 2023. [Online]. Dostupno na: *https://www.javatpoint.com/azure-database-service*

Popis slika

Popis isječaka programskog koda

Veleučilište u Virovitici

OBRAZAC 5

IZJAVA O AUTORSTVU

Ja, FRAN PINTARIĆ

izjavljujem da sam autor/ica završnog/diplomskog rada pod nazivom

Mobilna aplikacija za kreiranje treninga snage

Svojim vlastoručnim potpisom jamčim sljedeće:

- da je predani završni/diplomski rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija,
- da su radovi i mišljenja drugih autora/ica, koje sam u svom radu koristio/la, jasno navedeni i označeni u tekstu te u popisu literature,
- da sam u radu poštivao/la pravila znanstvenog i akademskog rada.

Potpis studenta/ice

Pintarić T.

Veleučilište u Virovitici

OBRAZAC 6

ODOBRENJE ZA OBJAVLJIVANJE ZAVRŠNOG/DIPLOMSKOG RADA U DIGITALNOM REPOZITORIJU

Ja FRAN PINTARIĆ

dajem odobrenje za objavljivanje mog autorskog završnog/diplomskog rada u javno dostupnom digitalnom repozitoriju Veleučilišta u Virovitici sadržanom u Dabar (Digitalni akademski arhivi i repozitoriji) te u javnoj internetskoj bazi završnih radova Nacionalne i sveučilišne knjižnice bez vremenskog ograničenja i novčane nadoknade, a u skladu s odredbama članka 58. stavka 5., odnosno članka 59. stavka 4. Zakona o visokom obrazovanju i znanstvenoj djelatnosti (NN 119/22).
Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog završnog/diplomskog rada. Ovom izjavom, kao autor navedenog rada dajem odobrenje i da se moj rad, bez naknade, trajno javno objavi i besplatno učini dostupnim na sljedeći način:

a) Rad u otvorenom pristupu
b) Rad dostupan nakon: _____ (upisati datum nakon kojeg želite da rad bude dostupan)
c) Pristup svim korisnicima iz sustava znanosti i visokog obrazovanja RH
d) Pristup korisnicima matične ustanove
e) Rad nije dostupan (*u slučaju potrebe dodatnog ograničavanja pristupa Vašem završnom/diplomskom radu, podnosi se pisani obrazloženi zahtjev*).

Potpis studenta/ice

Pintarić F.

U Virovitici, 8.9.2023.