Android aplikacija kao simulator nogometnog izbornika

Ocelić, Patrik

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

Sveučilišni studij računarstva

ANDROID APLIKACIJA KAO SIMULATOR NOGOMETNOG IZBORNIKA

Završni rad

Patrik Ocelić

Osijek, 2022.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

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

Osijek, 19.09.2022.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Patrik Ocelić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R 4409, 22.07.2019.
OIB Pristupnika:	10708843176
Mentor:	Izv. prof. dr. sc. Irena Galić
Sumentor:	Marin Benčević, mag. ing. comp.
Sumentor iz tvrtke:	
Naslov završnog rada:	Android aplikacija kao simulator nogometnog izbornika
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	Razviti Android aplikaciju za simulaciju umjetne nogometne lige gdje svaki korisnik može odabrati svoje igraće koji čine momčad. Nakon odabira, aplikacija računa rang ljestvicu svakog izbornika na osnovu stvarnih statistika odigranih utakmica svakog izabranog igraća. Opisati implementaciju aplikacije, način dohvaćanja i obrade podataka te algoritam za izračun ukupnih bodova na osnovu statistike svakog
Prijedlog ocjene završnog rada:	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	19.09.2022.
Datum potvrde ocjene od strane Odbora:	21.09.2022.

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

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 24.09.2022.

Ime i prezime studenta:	Patrik Ocelić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R 4409, 22.07.2019.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Android aplikacija kao simulator nogometnog izbornika**

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

i sumentora Marin Benčević, mag. ing. comp.

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

Potpis studenta:

Patrik Ocelić

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

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja Patrik Ocelić, OIB: 10708843176, student/ica Fakulteta elektrotehnike,
računarstva i informacijskih tehnologija Osijek na studiju Preddiplomski sveučilišni studij Računarstvo, kao autor/ica ocjenskog rada pod naslovom: Android aplikacija kao simulator nogometnog izbornika,
dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor/ica ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

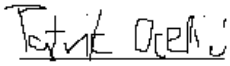
a) široj javnosti

b) studentima/icama i djelatnicima/ama ustanove

c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

Osijek, 24.09.2022.



(mjesto i datum)

(vlastoručni potpis studenta/ice)

SADRŽAJ

1. UVOD

Kada je riječ o nogometu, s jedne strane imamo ljude koji će reći kako je to samo sport i trčanje za loptom, no s druge strane imamo ljude koji govore kako je to najvažnija sporedna stvar na svijetu. Za potrebe ovog rada napravljena je aplikacija koja bi zaljubljenicima u nogomet mogla biti prava zanimacija. U svakoj ekipi, pa tako i u nogometnoj, jednu od najvažnijih uloga, ako ne i onu najvažniju ima trener. Cilj ove igre je korisniku što je bolje moguće približiti i dočarati ulogu trenera u ekipi.

Tema ovog završnog rada je izrada mobilne Android aplikacije koja će korisniku pružiti mogućnost stvaranja svoje ekipe od igrača iz Njemačke 1.nogometne lige, Bundeslige[1]. Nakon što korisnik izabere svoju ekipu od 11 igrača, pritiskom na određeni gumb provjerava koliko je dobru ekipu sastavio, te u ovisnosti o tome dobiva određen broj bodova.

1.1. Zadatak završnog rada

Cilj završnog rada je predočiti i približiti opis posla trenera/izbornika. Cilj je također i ostvariti mogućnost korisniku slobodan odabir igrača za svoju ekipu. Uz sve te mogućnosti, ova aplikacija bi trebala biti jako dobra za interakciju i zabavu među mlađim uzrastima. Aplikacija se može nadograditi tako da se kreira liga i da se otvori natjecanje, što bi vjerujem bilo vrlo popularno među svim vršnjacima.

Kako bi korisnicima uspio prenijeti te sve osjećaje i omogućiti im zabavu, potrebno je izraditi Android aplikaciju koja će moći dohvatiti sve potrebne podatke o igračima, o utakmicama, te koja će im omogućiti samo sučelje za korištenje svih funkcionalnosti.

2. SMISAO I NAČIN RADA SIMULATORA

2.1. Smisao simulatora

Bundesliga je ime za sva prvoligaška natjecanja u svim sportovima u Njemačkoj ili Austriji.

Bundesliga je vrlo zanimljiva za ova projekt zbog svoje napetosti u zadnjih par godina te zbog velikog broja vrlo dobrih igrača. Smisao ovog simulatora je prikazati i približiti populaciji posao nogometnog trenera/menadžera.

2.2. Način rada simulatora

Ovaj simulator/aplikacija je osmišljena tako da svaki korisnik odabire svoje igrače iz Njemačke 1.lige/Bundeslige , za koje misli da su dobro odigrali određenu utakmicu za svoj tim. Korisnik ima mogućnost biranja između dvije formacije: „4-3-3“ i „4-4-2“, nakon odabira formacije, potrebno je odabrati 11 igrača. Naravno, korisnik ne može npr. na mjesto golmana postaviti napadača i obrnuto, ograničeno je mijenjanje pozicija igračima. Na primjer, pritiskom na poziciju golmana, otvara se lista za odabir na kojoj su samo golmani Njemačke Bundeslige. Nakon odabira svih 11 igrača, pritiskom na „kvačicu“ pokreće se proces izračuna bodova, prvo se na slučajan način odabire „kolo“ lige i računaju se bodovi za svakog igrača.

2.3. Način računanja bodova

Bodovi se dobivaju ovisno o tome kako su igrači ekipe koju je korisnik složio odigrali određenu utakmicu. Za svaki zgoditak napadača koji se nalazi u ekipi, korisnik dobiva 3 boda. Dakle, broj golova korisnikovih napadača pomnožen s 3 pokazuje nam koliko je korisnik bodova dobio od napadačkog dijela. Kod veznog reda situacija je nešto drugačija, korisnik će od veznih igrača zarađivati bodove na osnovu njihovih asistencija. Dakle, za svaku asistenciju nekog od veznih igrača odabranih od strane korisnika, korisnik dobiva 2 boda. Bodovi koje je korisnik zaradio na osnovu odabranog veznog reda računat će se kao umnožak broja asistencija s 2. Nadalje, ostaje nam obrambeni dio. U ovom dijelu korisnik dobiva bodove na osnovu duela njegovih obrambenih igrača, za svaki osvojeni duel obrambenog igrača korisnik dobiva jedan bod. Dakle, obrambeni dio korisniku donosi (broj osvojenih duela)*1 bodova. Svi bodovi se zbrajaju i daju konačan rezultat. U teoriji nema maksimalnog broja bodova, no u praksi nije realno očekivati previše bodova.

2.4. Slična i postojeća rješenja za simulaciju nogometnog trenera

Naravno, na *Google Playu* postoji već nekoliko rješenja za ovakav tip aplikacije.

Neka rješenja su napravljena za neke druge natjecateljske lige, a ima rješenje čak i za Bundesligu.

Teško je za očekivati da već ne postoji rješenje za ovakav problem s obzirom na to koliko je nogomet popularan i rasprostranjen. Nadalje će biti prikazana neka od tih rješenja.

2.4.1. Fantasy Premier League

Ova aplikacija [2] jedna je od najpoznatijih ovog tipa, ako ne i najpoznatija. Koristi podatke i igrače najpoznatije lige na svijetu, Engleske „Premijer Lige" (eng. *Premier League*) .

Uz dostupnost na WEB pregledniku, također je dostupna i na Android uređajima. Ova aplikacija ima jako puno opcija, jako je detaljna te su dostupni svi klubovi i igrači ove lige. Sami dizajn aplikacije jako lijepo izgleda, te je aplikacija vrlo jednostavna za korištenje. Smisao aplikacije je da korisnici biraju svoje ekipe prije nego se odigra utakmica, dakle korisnici predviđaju koji igrači bi trebali odigrati dobro svoju narednu utakmicu. Kada utakmice završe bodovi se računaju, a korisnici se pomiču gore/dolje na ligaškoj tablici ovisno o osvojenim bodovima.

Slika 2.4 Izgled sučelja aplikacije

2.4.2. Fantasy Bundesliga

Navedena aplikacija[3] slična je našem rješenju/aplikaciji. Koristi se ista liga, tj. Njemačka Bundesliga, koriste se isti klubovi i igrači. Aplikacija je jako dobro optimizirana, te također ima jako lijepo sučelje za korisnika. Naše rješenje razlikuje se od ove aplikacije po tome što ova aplikacija također kao i „*Fantasy Premier League*", traži odabir ekipe prije početka utakmice. Dakle, princip rada te dvije aplikacije je vrlo sličan.

Commented [MB1]: Nemoj citirati Google Play, ili Wikipedia clanak za Google Play, jer je manje vise poznato sto je google play. Ako postoji neki popis fantasy football aplikacija, mozed njega citirati ali ako ne onda mozes samo obrisati ovaj citat.

Commented [MB2]: Slicno kao i gore, ne moras citrate Premier ligu.

Slika 2.5. Izgled sučelja aplikacije

3. MODEL ANDROID APLIKACIJE ZA SIMULACIJU NOGOMETNOG TRENERA/MENADŽERA

U ovom dijelu detaljno je opisan model Android aplikacije za simulaciju nogometnog trenera/menadžera.

3.1. Korisnički zahtjevi na mobilnu aplikaciju za simulaciju nogometnog izbornika

Od korisnika se ne traži niti registracija niti prijava pri ulazu u aplikaciju, aplikacija nema takve zahtjeve. Od korisnika se traži odabir igrača uz uvjet da odabere svih 11 igrača. To je zapravo jedini korisnički zahtjev uz odabir formacije. Iako odabir formacije nije nužan jer je korisniku pri odabiru opcije „CREATE LINEUP“ automatski postavljena osnovna formacija. Također korisnik ne može ni mijenjati pozicije igračima, tako da ni u tom dijelu nema nekakvih posebnih zahtjeva.

3.2. Funkcionalni zahtjevi mobilne aplikacije

3.2.1. Mijenjanje formacija u ovisnosti o želji korisnika

Kada korisnik odabere opciju „CREATE LINEUP“ tada se otvara zaslon za kreiranje ekipe.

Na ovom zaslonu moguće je odabrati opciju za promjenu formacije, korisnik ima dvije mogućnosti, formacija „4-3-3“ i formacija „4-4-2“. Prva formacija nam govori da u ekipi imamo 4 obrambena igrača, 3 vezna igrača te 3 igrača u napadu. Druga formacija postavlja 4 igrača u obranu, 4 igrača u veznu liniju i 2 igrača u sami vrh napada. Za ovaj zahtjev bilo je potrebno kreirati 2 formacije s različitim funkcionalnostima. Potreban je drugačiji razmještaj igrača za svaku od ove dvije formacije. Broj zahtjeva na bazu podataka je drugačiji, ako odaberemo formaciju „4-3-3“ tada imamo 4 poziva za dohvaćanje obrambenih igrača iz baze podataka, 3 poziva za dohvaćanje veznih igrača, te 3 poziva za dohvaćanje napadača.

3.2.2. Dohvaćanje podataka o igračima pomoću Retrofita

Commented [MB3]: Stavi veliko pocetno slovo svuda gdje spominjes (Retrofit, ne retrofit)

Pomoću Retrofita[4] potrebno je dohvatiti sve igrače Bundeslige, ovisno o odabranoj poziciji. Za dohvaćanje podataka korišten je API[5] (eng. *Application Programming Interface*) pod nazivom „RapidApi“[6]. Ovaj API vrlo je koristan zbog svojih mnogobrojnih mogućnosti, neke od njih su:

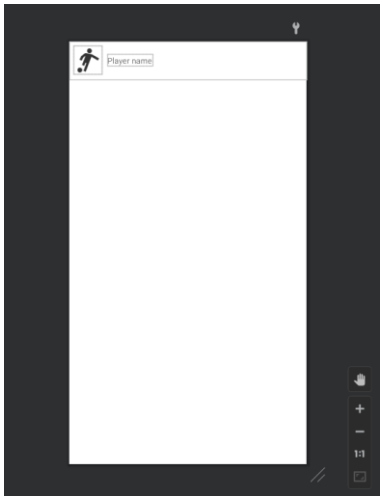
Commented [MB4]: API

- dohvaćanje podataka o utakmicama
- dohvaćanje podataka o igračima s obzirom na utakmicu, sezonu, godinu…
- dohvaćanje podataka za cijelu ligu
- dohvaćanje podataka za određeni klub

Postoje još mnogi drugi zahtjevi koje je moguće ostvariti pomoću ovog API-ja. Za naš problem koristili smo dohvaćanje podataka o igraču s obzirom na odigranu utakmicu, i tako za svakog od igrača. Dohvaćanje igrača obavili smo pomoću Retrofita koji zapravo pretvara naš HTTP API u Java sučelje. Omogućava nam komunikaciju s udaljenim serverom tj. bazom podataka.

3.2.3. Postvaljanje dohvaćenih podataka/igrača u RecyclerView

Naravno nakon dohvaćanja podataka i igrača, iste je potrebno spremiti u nekakvu listu kako bi korisnik mogao proći kroz sve igrače i odabrati one koji njemu odgovaraju. Ovu funkcionalnost ostvarili smo pomoći RecyclerView-a[7]. RecyclerView nam omogućava stvaranje dinamičke liste, kada „View“ ode izvan zaslona, tj. kada postane nevidljiv za korisnika, neće se uništiti, ponovno će iskoristiti isti „View“. Kako bi kreirali listu igrača koja nam se treba pojaviti klikom na neku od pozicija, potrebno je imati artikl(eng. *item*) tj. element kako će izgledati svaki od igrača kada ga dodamo u listu. Lista se sastoji od artikala, a svaki artikl ima svoj izgled. Slika nam prikazuje izgled artikla prije no što se dohvate podaci o igračima i spreme u listu.



Slika 3.2. Izgled jednog artikla

3.2.4. Izračun rezultata/bodova

Po završetku sastavljanja ekipe i odabira igrača, korisnik ima opciju za izračun ostvarenih bodova svoje ekipe. Naravno, to je i smisao aplikacije, da korisnik vidi koliko je dobru ekipu sastavio. Kako bi navedeno bilo ostvareno, potrebno je za svakoga igrača dohvatiti podatke za određenu utakmicu. To nije sve, treba provjeriti poziciju igrača te određeni statistički podatak(gol/asistenciju/osvojeni duel) pomnožiti s određenim koeficijentom(1/2/3). Ovo je potrebno napraviti za sve igrače u ekipi. Na kraju svi se bodovi zbrajaju i ispisuju na idućem zaslonu, uz to spremaju se u lokalnu bazu i prikazuju na profilu.

4. PROGRAMSKO RJEŠENJE APLIKACIJE ZA SIMULACIJU NOGOMETNOG TRENERA/MENADŽERA

U nadolazećem tekstu bit će detaljno objašnjeno što se i kako odvija u implementiranom programskom rješenju. Aplikacija se sastoji od jedne aktivnost (eng. *activity*), na toj jednoj aktivnosti mijenja se nekoliko *fragmenata*, od kojih svaki ima neku svoju određenu funkcionalnost. Ova aplikacija je izrađena koristeći MVVM arhitekturalni obrazac (eng. *architectural pattern*), fragmenti su zaduženi za prikaz podataka, a logika je smještena u *ViewModelima* i repozitorijima (eng. *repositories*).

Commented [MB5]: architectural

4.1. Opis alata i tehnologije

Aplikacija je napravljena u integriranom razvojnom okruženju Android Studio. Prototip aplikacije je izrađen u programu Figma. Za dohvaćanje podataka o igračima i ligi korišten je Rapid Api. Programski kod je pisan u programskom jeziku Kotlin zbog njegove jednostavnosti i mogućnosti. XML je korišten za izradu dizajna zaslona.

4.1.1. Figma

Figma[8] je prvi razvojni alat na webu profesionalne razine kreiran za dizajniranje korisničkog sučelja. Omogućuje jednostavno razvijanje izgleda aplikacije i time ubrzava cjelokupni razvoj.

Slika 4.1. Izgled aplikacije u Figmi

4.1.2. Kotlin

Kotlin[9] je programski jezik otvorenog koda, moderan je , siguran i višeplatformski. Podržava i objektno orijentiranje te ima odlične opcije testiranja. Od strane Google-a prihvaćen je kao vodeći jezik za razvoj Android mobilnih aplikacija. Ima vrlo sličnu sintaksu kao Java i C#. Razvijen je od strane JetBrains-a.

```
fun refreshPlayers()=viewModelScope.launch {
    _playersLoading.value=true
    PlayersRepository.getPlayers()
    _playersLoading.value=false
}
```

Slika 4.2. Primjer koda u Kotlinu

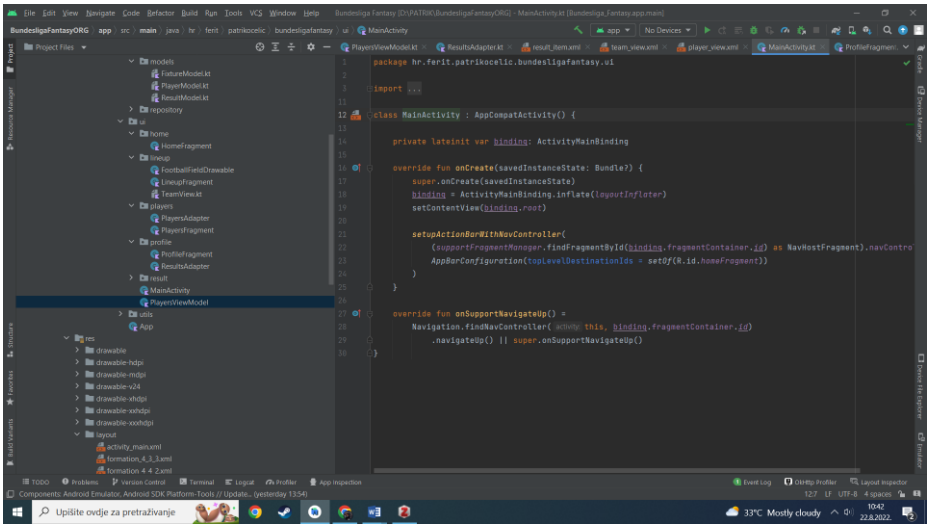
4.1.3. Android Studio

Commented [MB6]: Veliko S svuda u tekstu.

Android Studio[10] razvojno je okruženje koje se koristi za razvoj Android aplikacija. Najpoznatiji je i najbolji izbor za Android platformu. Riječ je o integriranom razvojnom okruženju (engl. *integrated development environment* - IDE) zasnovanom na IntelliJ IDEA platformi koje nudi niz razvojnih alata. Alati su:

- sustav za prevođenje
- emulator (služi za virtualno pokretanje aplikacija)
- alati za statističku analizu
- uređivač teksta

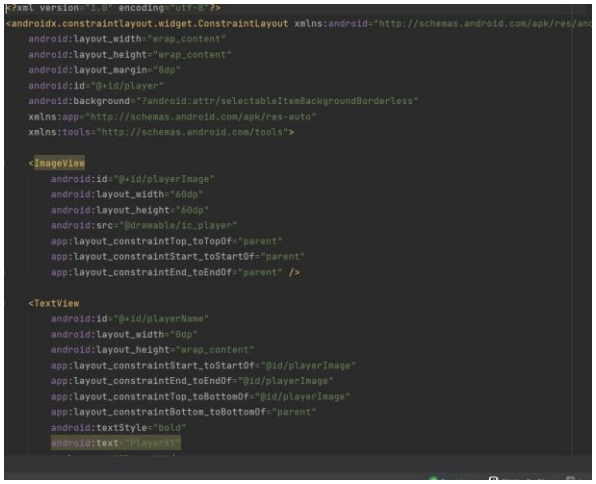
Pri razvoju aplikacija potrebna su nam dva paketa, a to su: JDK (eng. *Java development kit*) i Android SDK (eng. *Software development kit*) . JDK se za razliku od SDK ne preuzima automatski sa instaliranjem Android studija, pa ga je potrebno dodatno preuzeti.



Slika 4.3. Izgled sučelja Android studija

4.1.4. XML

XML(eng. *Extensible Markup Language*) je jezik za označavanje podataka. Zbog svoje jednostavnosti i čitljivosti je iznimno popularan. Podatke pohranjuje u formatu običnog teksta što omogućuje softverski i hardverski neovisan način pohrane i prijenosa podataka.



Slika 4.4. Primjer XML dokumenta

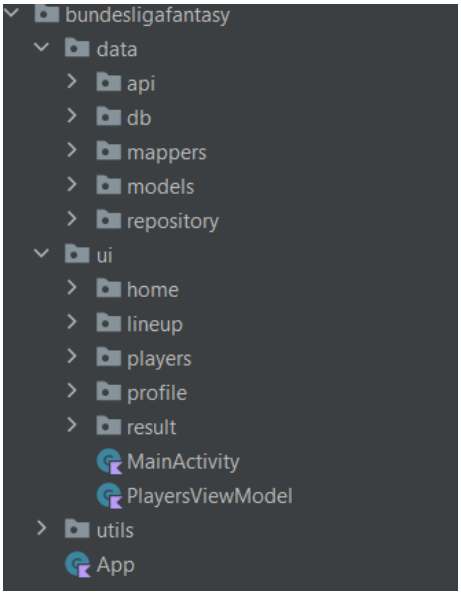
4.2. Dijagram tijeka

Dijagram tijeka (eng. *flowchart*) grafički je prikaz algoritma . Pomoćni je alat, vizualizira zadatak ili problem kako bi isti postao razumljiviji, jednostavniji za obradu te pregledniji. Čine ga geometrijski likovi, od kojih svaki ima svoje zasebno značenje. Na primjer, romb predstavlja uvjetno grananje, ovalni lik predstavlja početak/kraj programa, itd. Povezani su usmjerenim crtama i čine jednu cjelinu. Na slici prikazan je dijagram tijeka ove aplikacije.

4.3. Struktura projekta

Većina ozbiljnih projekata se nakon nekog vremena „pretrpa“ datotekama klasa i ostalim dijelovima koda. Radi bolje organizacije u kodu i strukture projekta Android Studio nam omogućava grupiranje srodnih datoteka unutar paketa (eng. *packages*). To su zapravo direktoriji.

Commented [MB7]: Android



Slika 4.5. Struktura projekta

Na slici prikazana je struktura projekta ove aplikacije, Aplikacija je podijeljena u nekoliko paketa, a neki od njih sadrže podpakete. Glavni paketi (eng. *main packages*) su:

- *data* – sadrži POJO(eng. *Plain old Java object*) klase i datoteke za lokalnu bazu podataka

- *ui* – korisničko sučelje (eng. *user interface*), sastoji se od klasa i fragmenata vidljivih korisnicima
- *utils* – sadrži datoteke sa ekstenzijskim i običnim funkcijama te ostale pomoćne klase koje služe za obradu podataka

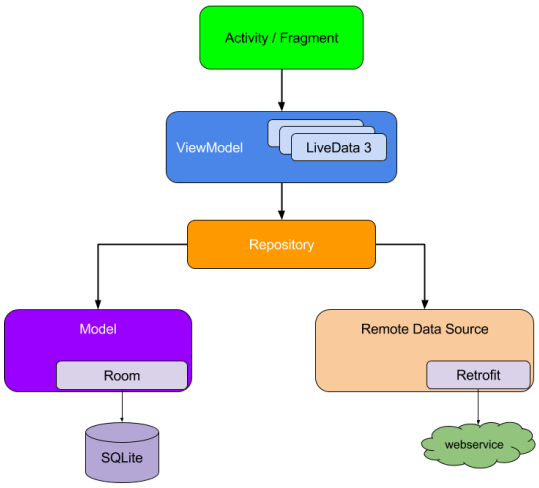
4.4. Korištene tehnologije

4.4.1. MVVM

Ova aplikacija je izrađena koristeći MVVM(eng. *Model-View-ViewModel*) obrazac.

Većina novih aplikacija izrađena je pomoću ovog obrasca. Google u posljednje vrijeme izbacuje jako puno novih biblioteka koje olakšavaju implementaciju ovog obrasca. Neke od tih biblioteka su biblioteka za navigaciju (eng. *Navigation library*), „JankStats“ biblioteka[11] pomaže nam da pratimo i analiziramo probleme pri izvođenju aplikacije u korisničkom sučelju, i mnoge druge biblioteke. Na slici 4.6. je prikazan MVVM obrazac.

Commented [MB8]: Citiraj odakle je slika kao i neke bibliotele koje je Google izbacio da potkrijepis ove recenice dokazima. Takodjer i u captionu slike ubaci citat na izvor.



Slika 4.6. MVVM shema [12]

Activity/fragment predstavlja pogled (eng. *view*), *ViewModel* govori sam za sebe, a repozitorij, koji koristi lokalne i podatke s vanjskog izvora predstavlja model jer služi za obradu podataka. Na shemi je vidljivo kako su komponente povezane. Pri korištenju aplikacije od strane korisnika, pozivaju se metode koje se nalaze unutar komponente pogleda(eng. *view*) . Ova komponenta poziva potrebne metode (odgovarajuće metode s obzirom na zahtjeve korisnika) u *ViewModelu* , dok on prosljeđuje zahtjev repozitoriju. Podaci se vraćaju obrnutim redoslijedom do pogleda, tada se podaci prikazuju na zaslonu.

Commented [MB9]: Ne mozes za MVVM rec da view model govori sam za sebe kad je to vjerojatno najbitniji dio koda. :) Malo opisi koji dijelovi koda se stavljaju u VM.

4.4.2. Kotlin Coroutine

Pružanje što boljeg korisničkog iskustva (eng. *user-experience - UX*) vrlo je važno pri izradi mobilnih aplikacija. Vrlo česta pogreška prilikom implementacije programskog rješenja je prepuštanje obavljanja kompliciranih i teških zadataka glavnoj niti (eng. *main thread*). Neki takvi zadaci su na primjer *API* pozivi (na neki udaljeni server), rad sa bazom podataka ili sa datotekama, i tako dalje. Prikaz podataka na zaslonu trebao bi biti jedini i glavni zadatak glavne niti, dakle glavna niti ne bi trebala obavljati nikakve teške/kompliciranе zadatke jer se tada zaslon ne može osvježavati, to može uzrokovati zastajkivanje aplikacije i automatski loše korisničko iskustvo, a to nam nije cilj. Svi nam je ljepše i ugodnije koristiti aplikacije koje rade glatko, dakle ovakva pojavljivanja zastajkivanja mogu udaljiti korisnika od korištenja aplikacije.

Jedan od rješenja ovog problema je prebacivanje poslova na pozadinske niti. *Thread*[13] klasa koju nam omogućuje Android sustav nije dobra i korisna u velikim projektima. Ako želimo dohvatiti podatke sa udaljenog servera (što je slučaj u ovoj aplikaciji), proslijediti ih dalje i odraditi još neki posao, možemo koristit *callback* pristup. Dakle, funkcija se poziva tek kada se posao završi, tj. ostali dio koda/programa se ne blokira dok se čeka rezultat, time smo ostvarili asinkron rad aplikacije. U slučaju da imamo više funkcija unutar *callbacka*, tada kod postaje teži za razumijevanje i dosta kompliciraniji.

Commented [MB10]: Moras ovdje ili dodati citat (tko je to rekao i zasto, npr. Neki dobar clanak na tu temu) ili sam reci zasto.

Ovaj problem rješava se vanjskim bibliotekama RxJava, Kotlin korutine (eng. *Kotlin coroutines*) itd. RxJava nam omogućuje jednostavno prebacivanje između niti, olakšava obradu i prikaz podataka na zaslonu.

Commented [MB11]: korutine

```
Observable.just( item1: 1,  item2: 2,  item3: 3)
    .doOnNext { printThread( prefix: "1") }
    .subscribeOn(s1)
    .observeOn(s2)
    .doOnNext { printThread( prefix: "2") }
    .flatMap { it: Int
        Observable.just( item1: 1,  item2: 2,  item3: 3)
            .doOnNext { printThread( prefix: "inner 1") }
            .subscribeOn(s3)
            .observeOn(s4)
            .doOnNext { printThread( prefix: "inner 2") }
    }
    .doOnNext { printThread( prefix: "3") }
    .observeOn(main)
    .subscribe { printThread( prefix: "end") }
```

Slika 4.7. Primjer korištenja RxJave [14]

Commented [MB12]: Ako je ovaj kod od negdje kopiran od rijeci do rijeci, onda ubaci citat na izvor.

Kao što vidimo na slici 4.8. RxJava koristi puno operatora (*doOnNext, subscribeOn, flatMap*..) za transformaciju podataka, znatno pomaže pri radu sa velikom količinom podataka.

Osnovica Kotlin korutina nije u pristupu rada sa operatorima već omogućuju pisanje asinkronog koda na sinkroni način. Korutine su slične nitima, ali ne koriste toliko puno resursa kao niti. Ključna riječ za pozivanje neke funkcije unutar korutine je *suspend* . Naredba *suspend* omogućuje pauziranje funkcije dok se ne izvrši posao za koji je zadužena. Funkcija ne blokira nit na kojoj je pozvana, te je to prednost.

```kotlin
suspend fun getPlayers() {
    val initialResponse = fetchPlayers( page: 1) ?: return
    val currentPage = initialResponse.paging.current
    val lastPage = initialResponse.paging.total
    val fetchMore = currentPage == 1 && currentPage < lastPage

    val otherResponse = if (!fetchMore) {
        emptyList()
    } else {
        ((currentPage + 1)..lastPage)
            .mapNotNull { page -> fetchPlayers(page) }
    }

    val total = listOf(initialResponse) + otherResponse
    total
        .flatMap { it.toPlayerUiModel() }
        .distinctBy { it.id }
        .let { AppDatabase.getInstance().getPlayerDao().replaceData(it) }
}
```

Slika 4.8. Primjer korištenja Kotlin korutina

4.5. Početni zaslon

Na slici 4.9. je prikazan početni/prvi zaslon. Kao što se može vidjeti nema login/register zaslona, naravno može se bez problema dodati, ali bi tada bilo potrebno kreirati svoju bazu podataka u koju će se spremati informacije korisnika za login. Za potrebe ovog rada nije bilo potrebno kreirati mogućnost registriranja korisnika, ali smatram da bi aplikacija time dobila novu dimenziju. Trenutno se ne mogu kreirati lige u kojima se korisnici mogu natjecati jer se svi rezultati korisnika spremaju lokalno, no ako bi se aplikacija proširila autentifikacijom (*Firebase*) [15] igrica bi dobila novi smisao.

Commented [MB13]: Kao što se može vidjeti,

Slika 4.9. Početni zaslon

Kao što se vidi na slici 4.9. na početnom zaslonu nalaze se dva gumba(eng. *button*). Gornji gumb CREATE LINEUP vodi na novi zaslon za kreiranje nove ekipe, dok nas donji gumb PROFILE vodi na naš profil gdje su prikazani naši prijašnji rezultati.

Commented [MB14]: o što se vidi na slici 4.9, …

Nemoj se obracati direktno citateljima u radu.

Commented [PO15R14]:

```
binding.btnLineup.setOnClickListener {
            findNavController().navigate(
HomeFragmentDirections.actionHomeFragmentToLineupFragment())
        }
```

Slika 5. Primjer korištenja gumba „btnLineup“

Iz koda sa slike je vidljivo kako nam „btnLineup“ tj. CREATE LINEUP gumb služi samo za prijelaz na drugi fragment.

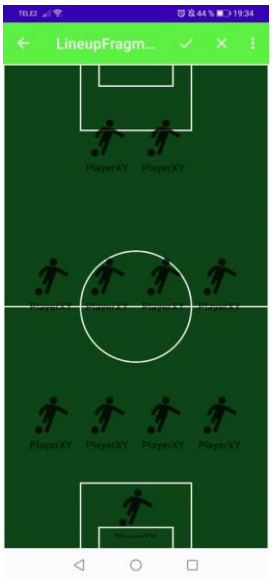
4.6. Zaslon za kreiranje ekipe

Sa slike 5.1. možemo vidjeti kako izgleda zaslon za kreiranje ekipe („LineupFragment“).

Kao što vidimo, na samom vrhu nalazi se navigacijska traka (eng. *navigation toolbar*).

Pritiskom na „kvačicu“ na navigacijskoj traci pokušavamo saznati broj bodova koje smo dobili za našu ekipu (uvjet izračuna je da je odabrano 11 igrača). Ako želimo ukloniti sve igrače iz postave možemo kliknuti na „iksić“ pored „kvačice“ na navigacijskoj traci, a za promjenu formacije pritisnemo „tri točkice“.

Commented [MB16]: Nema potrebe da ovo bude 3 odvojena odlomka.



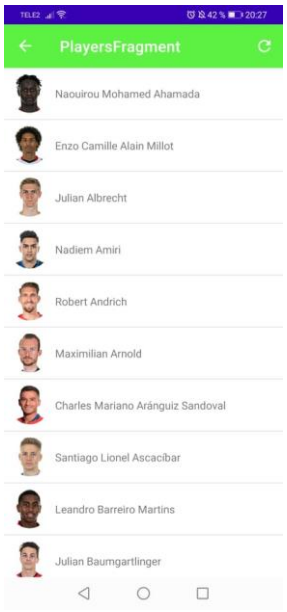
Slika 5.1.Zaslon za kreiranje ekipe

Pritiskom na svakog od „PlayerXY“/igrača otvara nam se lista sa svim igračima iz „Bundeslige“, ali samo za odabranu poziciju. Lista je implementirana koristeći *RecyclerView* .

Za razliku od *ListViewa* koji je korišten u prošlosti , ova komponenta stvara *view holdera* [16] koliko nam treba da se zaslon popuni te da se pripremi za pomicanje kroz listu.

4.7. Zaslon za prikaz igrača

Zaslon za prikaz igrača odnosno „PlayersFragment“ služi nam za prikaz igrača pri odabiru istih. Dakle, kada se nalazimo u „LineupFragmentu“ te tada pritisnemo na „PlayerXY“, otvara se ovaj zaslon.



Slika 5.2. Zaslon za prikaz igrača

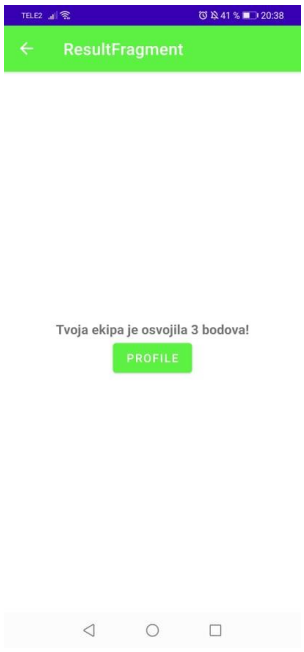
Na zaslonu se tada u obliku liste prikazuju igrači za odabranu poziciju, kao što je gore navedeno, korišten je *RecyclerView* za prikaz igrača.

```
private fun initRecycler() {
        binding.playersList.apply {
            layoutManager = LinearLayoutManager(requireContext())
            adapter = this@PlayersFragment.adapter
            setHasFixedSize(true)
            addItemDecoration(DividerItemDecoration(requireContext(),
(layoutManager as LinearLayoutManager).orientation))
        }
    }
```

Slika 5.3. Primjer korištenja *RecyclerViewa*

4.8. Zaslon za prikaz rezultata

„ResultFragment“ odnosno zaslon za prikazivanje rezultata zaslužan je za prikaz osvojenih bodova sa odabranom ekipom. Kada je odabrano svih jedanaest igrača, moguće je pokrenuti izračun bodova koje smo osvojili sa odabranim igračima. Izračun se pokreće pritiskom na „kvačicu“ na „LineupFragmentu“. Slika 5.4. nam prikazuje način na koji se prikazuju izračunati bodovi.

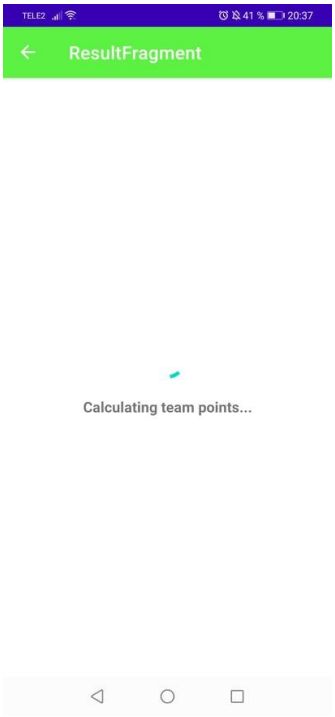


Slika 5.4. Zaslon za prikaz rezultata

Dok proces izračunavanja bodova traje, jer ponekada je moguće da treba pričekati po nekoliko sekundi, otvara nam se „loading screen“ koji nas obavještava da je izračun u tijeku.

Slika 5.5. prikazuje „loading screen“.

Commented [MB17]: Ubaci zadnje poglavlje Zakljucak (vise u mailu)



Slika 5.5. Prikaz „loading screena“

5. ZAKLJUČAK

Za potrebe ovog završnog rada napravljena je aplikacija za simulaciju nogometnog trenera/menadžera. Koristeći se logikom i algoritmom za izračun bodova, uz statistiku za svakog igrača i za svaku ekipu omogućen je izračun bodova. Aplikacija je napravljena koristeći Android studio, napisana je u programskom jeziku Kotlin.

Ova aplikacija pruža nekoliko opcija, odabir formacije, odabir igrača, odabir pozicija i još nekoliko sličnih opcija. Aplikacija korisniku pruža slobodu odabira pri svakoj opciji. Za potrebe dohvaćanja igrača i podataka o istim korišten je Retrofit. Za prikaz igrača u obliku dinamičke liste korišten je *RecyclerView*. *RecyclerView* nam omogućava spremanje dohvaćenih podataka u listu na način na koji korisnik želi.

Kao jednu od prednosti ove aplikacije naveo bih svakako i korištenje stvarnih podataka o igračima i klubovima u kojim isti igraju. Dakle, svaki podatak je provjeren i dohvaćen iz baze podataka koja koristi stvarne podatke Bundeslige. Ovakav tip aplikacije sa ažuriranim podacima trebao bi biti zanimljiv svakom korisniku.

Ovakav tip aplikacije izrazito je popularan među mlađim korisnicima. Kao dodatno proširenje i poboljšanje uz daljnji razvoj naveo bih kreiranje opcije Logina. Ova opcija bi uvelike poboljšala i proširila mogućnosti korisniku. Korisnik bi imao mogućnost spremanja osobnih podataka u bazu, također aplikacija bi postala kompetitivna, svaki korisnik bi se mogao natjecati u osvajanju bodova sa drugim korisnicima.

LITERATURA

[1] „Bundesliga“. 19. prosinac 2021. Pristupljeno: 19. kolovoz 2022. [Na internetu]. Dostupno na: https://hr.wikipedia.org/w/index.php?title=Bundesliga&oldid=6157094

[2] „Fantasy Premier League“. https://fantasy.premierleague.com/ (pristupljeno 21. kolovoz 2022.).

[3] „Bundesliga Fantasy Manager, Aplikacije na Google Playu“. https://play.google.com/store/apps/details?id=com.bundesliga.fantasy&hl=hr&gl=US (pristupljeno 21. kolovoz 2022.).

[4] „Retrofit“. https://guides.codepath.com/android/consuming-apis-with-retrofit (pristupljeno 22. kolovoz 2022.).

[5] „API“. https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces (pristupljeno 22. kolovoz 2022.).

[6] „RapidApi“. https://rapidapi.com/hub (pristupljeno 21. kolovoz 2022.).

[7] „RecyclerView“, *Android Developers*. https://developer.android.com/guide/topics/ui/layout/recyclerview (pristupljeno 22. kolovoz 2022.).

[8] „Figma (software)“, *Wikipedia*. 31. srpanj 2022. Pristupljeno: 21. kolovoz 2022. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Figma_(software)&oldid=1101521958

[9] „Kotlin Programming Language“, *Kotlin*. https://kotlinlang.org/ (pristupljeno 22. kolovoz 2022.).

[10] „Android Studio“, *Wikipedia*. 05. kolovoz 2022. Pristupljeno: 22. kolovoz 2022. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Android_Studio&oldid=1102445910

[11] „Google libraries“, *Android Developers Blog*. https://android-developers.googleblog.com/2022/05/whats-new-in-jetpack.html (pristupljeno 24. kolovoz 2022.).

[12] „MVVM image“. https://www.pinterest.com/pin/702631979347824648/ (pristupljeno 24. kolovoz 2022.).

[13] „Threading“. https://developer.android.com/topic/performance/threads (pristupljeno 24. kolovoz 2022.).

[14] „RxJava“. https://krossovochkin.com/posts/2020_01_25_from_rxjava_2_to_kotlin_flow_threading/ (pristupljeno 24. kolovoz 2022.).

[15] „Firebase“. https://firebase.google.com/?gclid=Cj0KCQjw0oyYBhDGARIsAMZEuMso9Oe1R2P6PtUVVpMKhLPWUIo6RZetJ-TrAI6sAJX6iTt5Vr48w28aArsCEALw_wcB&gclsrc=aw.ds (pristupljeno 22. kolovoz 2022.).

[16] „ViewHolder“. https://developer.android.com/reference/kotlin/androidx/recyclerview/widget/RecyclerView.ViewHolder (pristupljeno 22. kolovoz 2022.).

SAŽETAK

U ovom završnom radu kreirana je aplikacija koja se koristi kao simulator funkcije nogometnog trenera.

Opisan je postupak dohvaćanja i korištenja podataka koji se koriste za rad aplikacije.

Spomenuti su i također opisani alati koji se koriste za izradu aplikacije. Važnu ulogu u ostvarivanju funkcionalnosti aplikacije imali su *ViewModel* i *RecyclerView*. *ViewModel* je obrazac koji korisniku olakšava odvajanje korisničkog sučelja od logike koja je potrebna za pravilan rad aplikacije. *RecyclerView* korisniku omogućava postavljanje elemenata u dinamičku listu na način na koji on to želi, tj. korisnik kreira izgled elementa liste, naravno svi elementi moraju imati isti izgled, mijenjaju se samo određeni podaci. Naravno kao i kod svake druge aplikacije korisnik vidi samo korisničko sučelje i omogućeno mu je korištenje aplikacije.

Ključne riječi: nogometni trener, Bundesliga, Android, ViewModel

ABSTRACT

Football Manager Simulator Android Application

In this final paper, an application was created that is used as a simulator of the function of a football coach. The process of retrieving and using the data used for the operation of the application is described. The tools used to create the application are mentioned and also described. ViewModel and RecyclerView played an important role in realizing the functionality of the application. ViewModel is a pattern that makes it easy for the user to separate the user interface from the logic that is needed for the application to work properly. RecyclerView allows the user to place elements in a dynamic list in the way he wants, i.e. the user creates the appearance of the list element, of course all elements must have the same appearance, only certain data are changed. Of course, as with any other application, the user sees only the user interface and is enabled to use the application.

Key words: football coach, Bundesliga, Android, ViewModel

ŽIVOTOPIS

Commented [MB18]: Treba donate sažetak (više u mailu) i prilog s kodom

Patrik Ocelić rođen je 27. ožujka 2001. godine u Našicama. Od 2007. do 2011. godine pohađa Područnu Školu u Bokšiću. Po završetku 4.razred osnovne škole, školovanje nastavlja u Đurđenovcu. Od 5. do 8. razreda pohađao je Osnovnu školu Josipa Jurja Strossmayera u Đurđenovcu. Godine 2015. upisuje Prirodoslovnu matematičku gimnaziju u Srednjoj školi Isidora Kršnjavog u Našicama. Nakon što je završio srednju školu, 2019. godine upisao je preddiplomski studij Računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.