Aplikacija za Prvu nogometnu ligu

Stanković, Luka

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

APLIKACIJA ZA PRVU NOGOMETNU LIGU

Završni rad

Luka Stanković

Osijek, 2024.

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

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Luka Stanković
Studij, smjer:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R4714, 28.07.2021.
JMBAG:	0165091889
Mentor:	izv. prof. dr. sc. Josip Balen
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Aplikacija za Prvu nogometnu ligu
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada:	Potrebno je razviti mobilnu aplikaciju u razvojnom okruženju Android Studio koristeći Kotlin programski jezik, JetPack Compose, Firebase, RESTful API i ostale alate. Aplikacija treba sadržavati prikaz osnovnih informacija o nogometnoj ligi, trenutnu tablicu, raspored utakmica, vijesti i rezultate najnovijih utakmica. Nadalje, treba prikazati raspored svih nadolazećih utakmica u ligi (sa datumima, vremenima, lokacijama i informacijama o klubovima), prikaz trenutne tablice lige s osnovnim informacijama o svakom klubu (broj odigranih utakmica, pobjeda, neriješenih rezultata, poraza, gol razlike,

Datum prijedloga ocjene završnog rada od strane mentora:	04.09.2024.
Prijedlog ocjene završnog rada od strane mentora:	Izvrstan (5)
Datum potvrde ocjene završnog rada od strane Odbora:	11.09.2024.
Ocjena završnog rada nakon obrane:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	13.09.2024.

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

IZJAVA O IZVORNOSTI RADA

Osijek, 13.09.2024.

Ime i prezime Pristupnika:	Luka Stanković
Studij:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R4714, 28.07.2021.
Turnitin podudaranje [%]:	12

Ovom izjavom izjavljujem da je rad pod nazivom**: Aplikacija za Prvu nogometnu ligu**

izrađen pod vodstvom mentora izv. prof. dr. sc. Josip Balen

i sumentora

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

Potpis pristupnika:

SADRŽAJ

1. UVOD

U današnjem digitalnom vremenu, mobilne aplikacije igraju ključnu ulogu u različitim aspektima svakodnevnog života, od komunikacije i zabave do obrazovanja i poslovanja. Jedno od područja koje je doživjelo značajan rast je sportski segment, posebno nogomet, koji privlači milijune korisnika širom svijeta. Sa sve većom popularnošću nogometa, postoji potreba za aplikacijama koje omogućuju korisnicima pristup ažuriranim informacijama, statistikama, transferima i vijestima vezanim uz kako svjetske tako i hrvatske nogometne lige.

Glavni cilj ovog završnog rada je razvoj mobilne aplikacije koja će omogućiti korisnicima praćenje najnovijih informacija o Prvoj hrvatskoj nogometnoj ligi. Aplikacija će korisnicima pružiti mogućnost pregleda trenutnih rezultata, statistika, informacija o igračima i klubovima, te vijesti i transfere unutar liga. Na ovaj način, korisnici će imati sve potrebne informacije na dohvat ruke.

Ovaj rad podijeljen je u nekoliko poglavlja. U prvom poglavlju, daje se pregled trenutnih dostignuća u području mobilnih aplikacija za praćenje kako europskih tako i hrvatskih nogometnih liga. Drugo poglavlje opisuje tehnologije i korištene alate pri razvoju aplikacije, uključujući razvojno okruženje Android Studio, Kotlin programski jezik, *JetPack Compose, Firebase* te RESTful API. Zatim u trećem poglavlju detaljno se opisuje razvoj aplikacije, od dizajna korisničkog sučelja do implementacije funkcionalnosti kao što su autentifikacija korisnika, prikaz statistika i upravljanje podacima. U četvrtom poglavlju prikazuju se rezultati testiranja aplikacije te se raspravlja o prednostima i nedostacima razvijenog rješenja. Na kraju se dolazi do zaključka gdje se sumiraju postignuti rezultati i daju prijedlozi za budući razvoj aplikacije.

1.1. Zadatak završnog rada

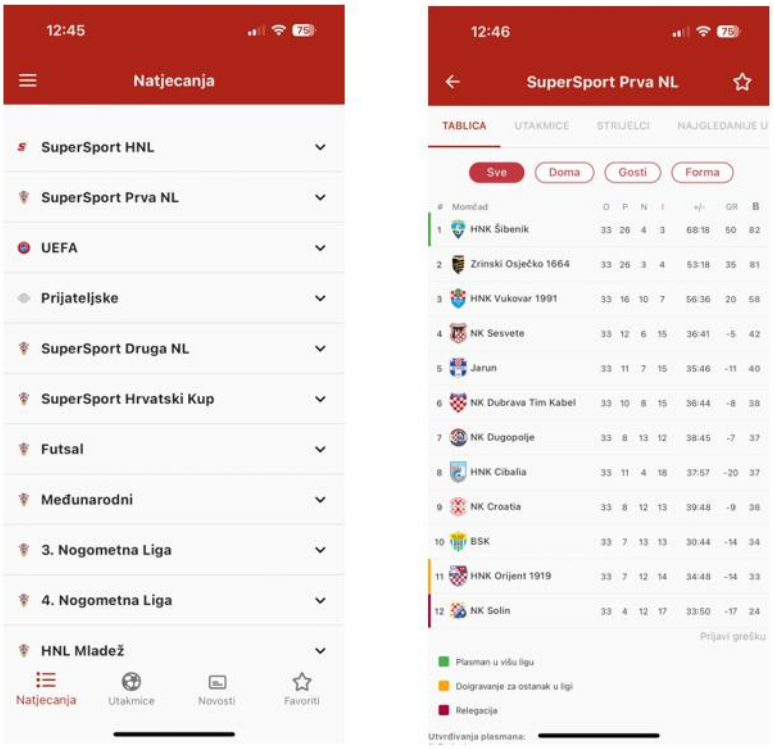
Potrebno je razviti mobilnu aplikaciju u razvojnom okruženju Android Studio koristeći Kotlin programski jezik, *JetPack Compose, Firebase*, RESTful API i ostale alate. Aplikacija treba sadržavati prikaz osnovnih informacija o nogometnoj ligi, trenutnu tablicu, odigrane utakmice, vijesti i sažetke utakmica. Nadalje, treba prikazati trenutnu tablice lige s osnovnim informacijama o svakom klubu (broj odigranih utakmica, pobjeda, neriješenih rezultata, poraza, gol razlike, bodovi itd.), prikaz statistike igrača i klubova u ligi (najbolji strijelci, najviše žutih kartona i sl.), vijesti (analize, vijesti o transferima ili ozljedama) te sve ostale relevantne informacije o ligi. Korisnicima treba omogućiti registraciju te mogućnost komentiranja i ocjenjivanja utakmica.

2. PREGLED SLIČNIH POSTOJEĆIH RJEŠENJA

U ovom poglavlju, daje se pregled postojećih mobilnih aplikacija za praćenje nogometnih liga, s naglaskom na hrvatske i međunarodne aplikacije. Cilj je analizirati trenutno dostupna rješenja, njihove funkcionalnosti i performanse koje će služiti kao uzor u izradi programskog rješenja ovog završnog rada. Aplikacije koje će biti analizirane su HRnogomet, SofaScore te Rezultati.hr.

2.1. HRnogomet

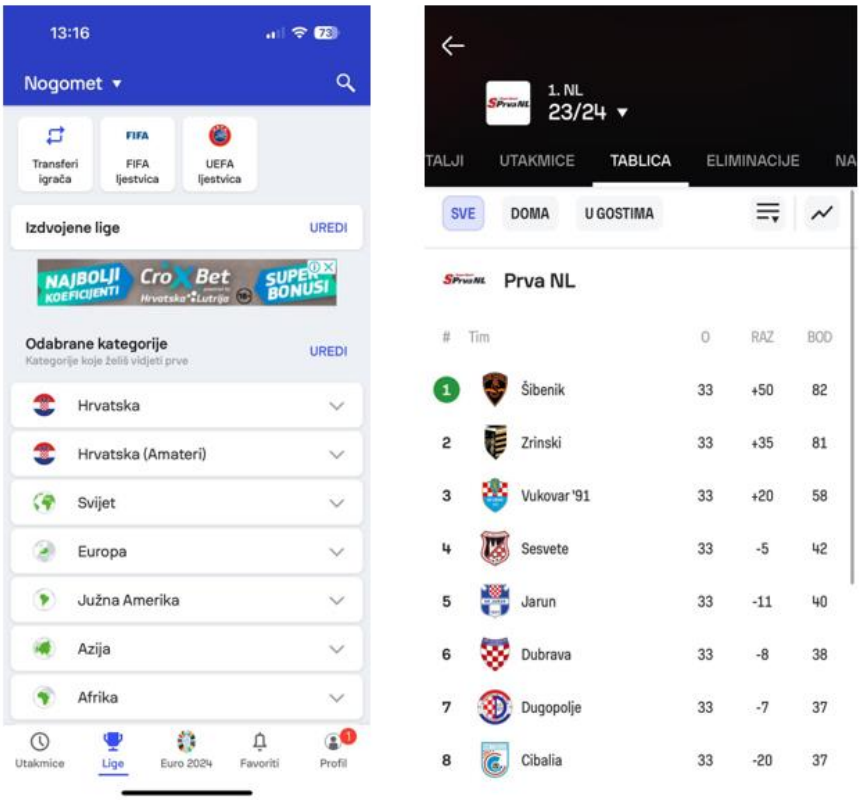
HRnogomet je aplikacija specijalizirana za praćenje hrvatskih nogometnih liga. Korisnicima omogućuje pregled rezultata, tablica, rasporeda utakmica i vijesti vezanih uz hrvatski nogomet. Ono što ovu aplikaciju čini posebnijom od ostalih jest to da pruža i mnogo informacija o hrvatskom niželigaškom nogometu te izbor liga koje žele promatrati grupiranih po županijama (slika 2.1). Unutar ove aplikacije naravno se nalazi i liga obrađena u ovom završnom radu, Prva hrvatska nogometna liga [1].



Slika 2.1 Zasloni unutar aplikacije HRnogomet

2.2. SofaScore

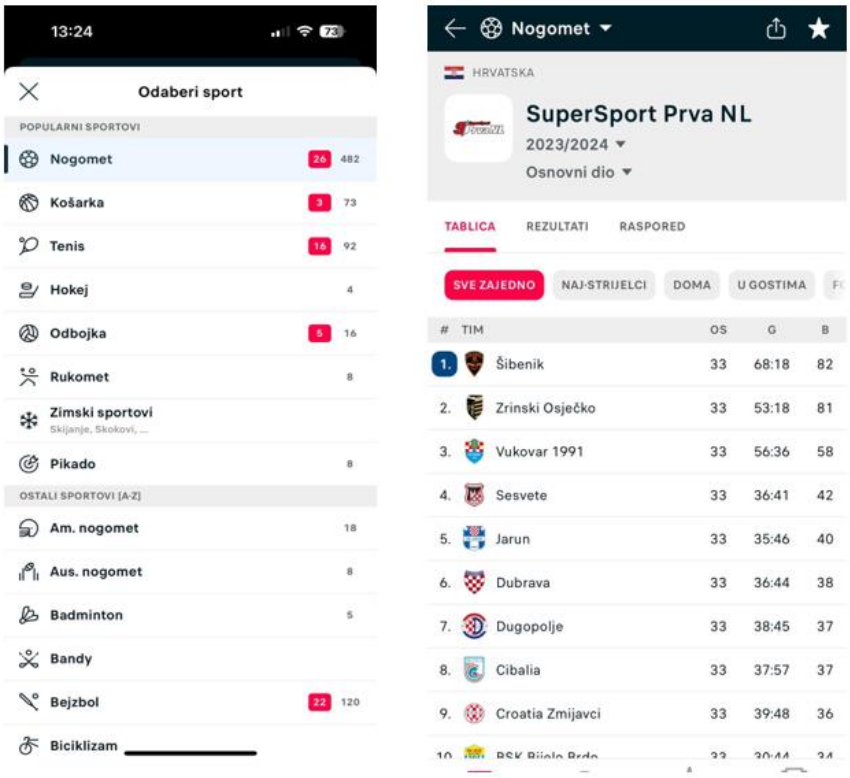
SofaScore je aplikacija koja pokriva širok spektar sportova, uključujući nogomet, te nudi detaljne statistike, rezultate uživo, analize i vijesti (slika 2.2). Posebno je popularna među korisnicima koji žele pratiti rezultate hrvatskih nogometnih liga. Prednosti koje donosi su širok spektar sportova, detaljne analize i statistike, označavanje omiljenih timova te brze i točne obavijesti. Nedostaci su previše informacija koje nisu nužno relevantne za svakog korisnika te broj reklama unutar aplikacije [2].



Slika 2.2. Zasloni unutar SofaScore aplikacije

2.3. Rezultati.hr

Rezultati.hr je aplikacija koja nudi rezultate uživo, tablice i rasporede za razne sportove, uključujući nogomet. Korisnicima također omogućuje praćenje rezultata hrvatskih i međunarodnih nogometnih liga. Prednosti su pružanje rezultata uživo i brze obavijesti, mogućnost praćenja više sportova, vijesti koje uključuju najave i izvještaje te odabir omiljenih ekipa (slika 2.3). Nedostaci u odnosu na konkurente su manje intuitivno sučelje za nove korisnike te nedostatak detaljnih statistika [3].



Slika 2.3. Zasloni unutar aplikacije Rezultati.hr

3. TEHNOLOGIJE I ALATI KORIŠTENI ZA RAZVOJ APLIKACIJE

Mobilne aplikacije su softverske aplikacije razvijene posebno za uporabu na malim bežičnim računalnim uređajima kao poput pametnih telefona i tableta. Mobilne aplikacije dizajnirane su za rad na mobilnim operacijskim sustavima kao što su Android, iOS i Windows Phone. Kada se nativna mobilna aplikacija preuzme i instalira na uređaj, ona se pohranjuje u memoriju uređaja i pokreće unutar operacijskog sustava uređaja [4]. To vrijedi za aplikacije koje su specifično razvijene za određeni operativni sustav. Nasuprot tome, postoje i web aplikacije koje se ne instaliraju na uređaj, već se pokreću putem internetskog preglednika, dok hibridne aplikacije kombiniraju karakteristike nativnih i web aplikacija. Razvoj aplikacije zahtijeva integraciju nekoliko ključnih tehnologija kako bi se osigurala funkcionalnost, učinkovitost i korisničko iskustvo.

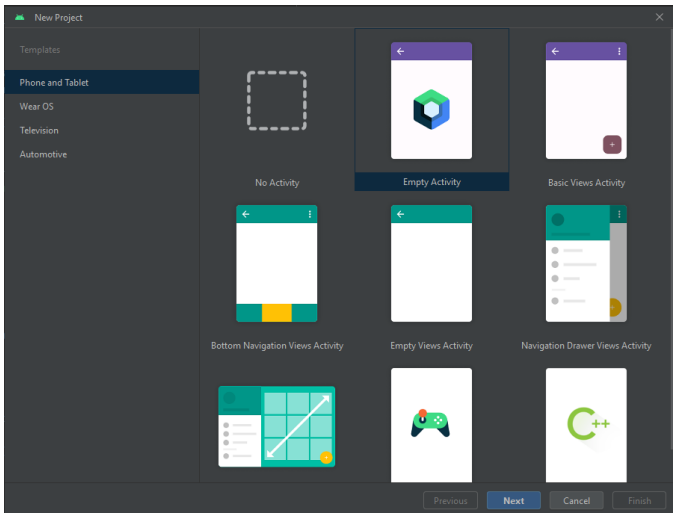
Korišteni jezici i alati za izradu ove aplikacije su:

- Razvojno okruženje Android Studio,
- Kotlin,
- Firebase,
- JetPack Compose,
- RESTful API.

Sve navedene tehnologije detaljnije su objašnjene u sljedećim potpoglavljima.

3.1. Razvojno okruženje Android Studio

Android Studio je službeno integrirano razvojno okruženje (IDE) za razvoj Android aplikacija. Primjer kreiranja jednostavnog projekta prikazan je slikom 3.1. Android Studio nudi fleksibilan sustav gradnje baziran na *Gradle-u*, brze i raznovrsne emulatore, ujedinjeno okruženje u kojem možete razvijati za sve Android uređaje, *Live Edit* za ažuriranje kompozicija u emulatorima i fizičkim uređajima u stvarnom vremenu, *Lint* alate za hvatanje problema s performansama, upotrebljivošću, kompatibilnošću verzija i drugih problema te mnoge druge mogućnosti [5].



Slika 3.1. Primjer kreiranja projekta u razvojnome okruženju

3.2.Programski jezik Kotlin

Kotlin je moderni programski jezik razvijen od tvrtke *JetBrains*, a koji je službeno podržan od strane *Google-a* za razvoj Android aplikacija [6]. Kotlin ima bogat skup značajki koje od samog početka podržavaju funkcionalno programiranje. Te značajke uključuju:

- Tipovi funkcija - Omogućuju funkcijama da primaju druge funkcije kao argumente ili vraćaju funkcije.
- Lambda izrazi - Omogućuju prosljeđivanje blokova koda s minimalnim viškom koda.
- Referencije članova - Dopuštaju korištenje funkcija kao vrijednosti, primjerice, prosljeđivanje funkcija kao argumenata.
- Podatkovne klase - Pružaju sažet sintaktički način za kreiranje klasa koje mogu pohranjivati nepromjenjive podatke.
- API standardne biblioteke - Bogat set API-ja u standardnoj biblioteci za rad s objektima i kolekcijama u funkcionalnom stilu [7].

3.3. JetPack Compose

Jetpack Compose je preporučeni moderni alat za kreiranje izvornog korisničkog sučelja na Androidu. Pojednostavljuje i značajno ubrzava razvoj korisničkog sučelja na Androidu. Omogućava brzo unaprjeđenje dizajna aplikacije s manje koda, moćnim alatima i intuitivnim Kotlin API-jevima [8].

Funkcija kompozicije je Kotlin funkcija koja je označena anotacijom *@Composable*. Sve kompozicijske funkcije moraju biti označene na ovaj način jer anotacija obavještava *Compose* prevoditelj da funkcija pretvara podatke u UI elemente.

Potpis Kotlin funkcije sastoji se od sljedećih dijelova:

- Opcionalni modifikator vidljivosti (private, protected, internal ili public)
- Ključna riječ *fun*
- Naziv funkcije
- Popis parametara (može biti prazan) ili, opcionalno, zadana vrijednost
- Opcionalni povratni tip
- Blok koda [9].

Kada se kreira korisničko sučelje (UI), potrebno je definirati gdje će se elementi pojaviti i kolike će im biti dimenzije. *Jetpack Compose* pruža nekoliko osnovnih rasporeda koji postavljaju sadržaj duž jedne glavne osi. Svaka osa predstavljena je rasporedom. *Row()* postavlja svoj sadržaj horizontalno, dok *Column()* postavlja sadržaj vertikalno. *Box()* i *BoxWithConstraints()* slažu svoje sadržaje jedan iznad drugoga [9].

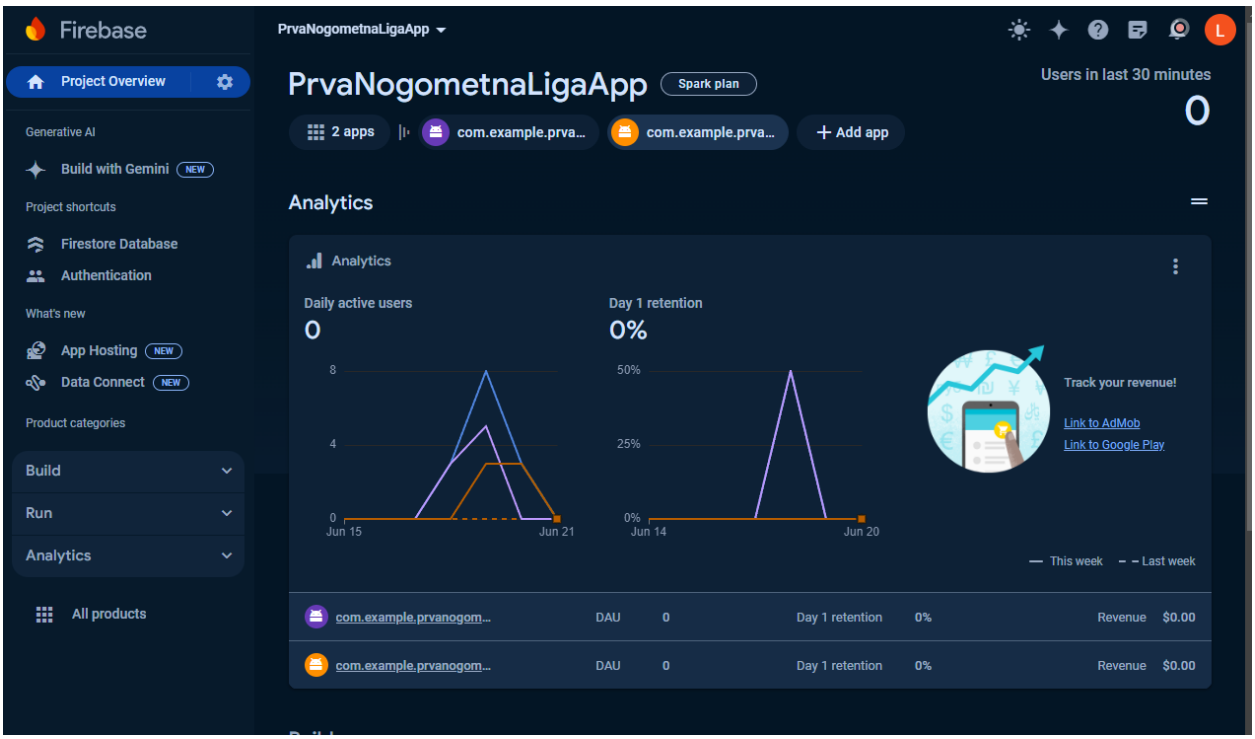
3.4. Firebase

Firebase je razvijen od strane Google-a kako bi pomogao programerima da lakše izgrade, upravljaju i razvijaju svoje aplikacije (slika 3.2). Na strani *Firebase-a* nije potrebno programiranje, što olakšava korištenje njegovih značajki učinkovitije. Pruža usluge za Android, iOS, web i Unity [10]. Pruža pohranu u oblaku. Za pohranu podataka koristi *NoSQL* bazu podataka.

Ova tehnologija uglavnom uključuje *backend* usluge koje pomažu programerima da bolje izgrade i upravljaju svojim aplikacijama. Usluge uključene u ovu značajku su:

- *Realtime Database: Firebase Realtime Database* je oblačna *NoSQL* baza podataka koja upravlja vašim podacima nevjerojatnom brzinom od milisekundi. U najjednostavnijim terminima, može se smatrati velikom *JSON* datotekom.
- *Cloud Firestore: Cloud Firestore* je *NoSQL* dokumentna baza podataka koja pruža usluge kao što su pohrana, sinkronizacija i upiti kroz aplikaciju na globalnoj razini. Pohranjuje podatke u obliku objekata poznatih kao dokumenti. Ima par ključ-vrijednost i može pohraniti sve vrste podataka, poput stringova, binarnih podataka, pa čak i JSON stabala.

- Autentifikacija: Sustav za autentifikaciju korisnika koji omogućuje jednostavnu integraciju prijave i registracije e-pošte i lozinke, društvenih mreža (kao što su Google, Facebook i Twitter), jednokratne prijave (OTP) i drugih metoda autentifikacije.
- *Remote Config*: Usluga daljinske konfiguracije pomaže u objavljivanju ažuriranja korisnicima odmah. Promjene mogu uključivati promjenu komponenti korisničkog sučelja ili promjenu ponašanja aplikacija. Često se koriste prilikom objavljivanja aktualnih ponuda i sadržaja u aplikaciji koji imaju ograničen vijek trajanja [11].



Slika 3.2 Isječak zaslona Firebase konzole

3.5. RESTful API

API što dolazi kao skraćenica od *Application Programming Interface* je skup pravila i protokola koji definira način na koji aplikacijski softver komunicira i integrira se. Često se opisuje kao ugovor između pružatelja i potrošača informacija - u kojemu potrošač definira zahtjev (engl. *request*), a pružatelj odgovara na taj zahtjev (engl. *response*).

REST je skup arhitektonskih ograničenja, najčešće primijenjenih putem HTTP-a kao osnovnog transportnog protokola. Sam pojam REST, skraćenica od *Representational State Transfer*, uveo je računalni znanstvenik Roy Fielding.

Kako bi API bio smatran RESTful-om, mora ispunjavati sljedeće uvjete:

- Interakcija između proizvođača i potrošača je modelirana, pri čemu proizvođač modelira resurse s kojima potrošač može komunicirati.
- Zahtjevi između proizvođača i potrošača moraju biti neovisni, što znači da proizvođač ne pohranjuje detalje prethodnih zahtjeva. Kako bi potrošač stvorio niz zahtjeva za određeni resurs, mora poslati sve potrebne informacije proizvođaču za obradu.
- Zahtjevi moraju biti predpohranjeni, što znači da proizvođač može pružiti smjernice potrošaču kada je to prikladno. U HTTP-u se to često omogućuje putem informacija u zaglavlju.
- Svim potrošačima treba biti pruženo jednoliko sučelje.
- REST mora biti slojevit sustav, apstrahirajući složenost sustava koji se nalaze iza REST sučelja. Na primjer, potrošač ne bi trebao znati niti brinuti komunicira li s bazom podataka ili s nekim drugim uslugama [12].

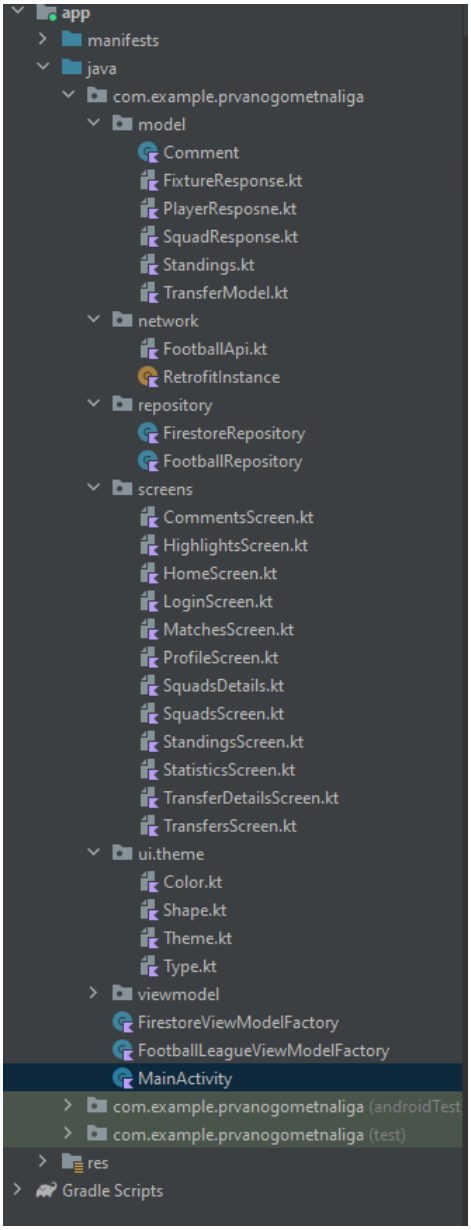
Kada klijent pošalje zahtjev putem RESTful API-ja, prenosi reprezentaciju stanja resursa prema odredišnoj točki odnosno poslužitelju. Ova reprezentacija, ili prikaz, dostavlja se u jednom od nekoliko formata putem HTTP-a, kao što su JSON (JavaScript Object Notation), HTML, XML, Python, PHP ili običan (engl. plain) tekst. JSON je najčešće korišten format jer, unatoč svom nazivu, nije ograničen na određeni programski jezik i podjednako je lako razumljiv i ljudima i strojevima [13].

4. PRIKAZ PROGRAMSKOG RJEŠENJA

U ovom poglavlju, detaljno će se opisati razvoj mobilne aplikacije "Prva Nogometna Liga". Pokazat će se dizajn korisničkog sučelja, implementacija ključnih funkcionalnosti poput autentifikacije korisnika, prikaza pojedinih ekrana te upravljanja podacima

4.1. Struktura projekta

Struktura projekta je organizirana na način koji omogućava jednostavno upravljanje, proširenje i održavanje aplikacije. Projekt je podijeljen u nekoliko paketa, od kojih svaki ima specifičnu ulogu u aplikaciji. Detaljna struktura projekta po paketima prikazana je na slici 4.1.



Slika 4.1 Prikaz strukture projekta

4.2. Prijava i registracija korisnika

Početni ekran aplikacije služi za autentifikaciju korisnika, omogućujući im prijavu ili registraciju za pristup aplikaciji. Koristeći *AuthViewModel*, ekran omogućuje upravljanje korisničkim stanjima prijave i registracije, čime se osigurava da samo ovlašteni korisnici mogu pristupiti aplikaciji. Metode unutar klase prikazane su programskim kodom 4.1, dok je UI tog ekrana prikazan programski kodom 4.2.

```
fun signIn(email: String, password: String) {
    viewModelScope.launch {
        auth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    _user.value = auth.currentUser
                    _authError.value = null
                } else {
                    _user.value = null
                    _authError.value = task.exception?.message
                }
            }
    }
}

fun signUp(email: String, password: String) {
    viewModelScope.launch {
        auth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    _user.value = auth.currentUser
                    _authError.value = null
                } else {
                    _user.value = null
                    _authError.value = task.exception?.message
                }
            }
    }
}

fun signOut() {
    auth.signOut()
    _user.value = null
}
```

Programski kod 4.1 Metode signIn, signUp i signOut unutar AuthViewModela

```
        Image(
            painter = painterResource(id = R.drawable.app_logo),
            contentDescription = null,
            modifier = Modifier.size(150.dp),
            contentScale = ContentScale.Crop
        )
        Spacer(modifier = Modifier.height(32.dp))
        OutlinedTextField(
            value = email,
            onValueChange = { email = it },
            label = { Text( text: "Email") },
            modifier = Modifier.fillMaxWidth(),
            keyboardOptions = KeyboardOptions.Default.copy(
                imeAction = ImeAction.Next
            ),
            keyboardActions = KeyboardActions(
                onNext = { keyboardController?.hide() }
            )
        )
        Spacer(modifier = Modifier.height(16.dp))
        OutlinedTextField(
            value = password,
            onValueChange = { password = it },
            label = { Text( text: "Password") },
            modifier = Modifier.fillMaxWidth(),
            keyboardOptions = KeyboardOptions.Default.copy(
                imeAction = ImeAction.Done
            ),
            keyboardActions = KeyboardActions(
                onDone = { keyboardController?.hide() }
            )
        )
        Spacer(modifier = Modifier.height(16.dp))
        Button(
            onClick = {
                if (isSignUp) {
                    authViewModel.signUp(email, password)
                } else {
                    authViewModel.signIn(email, password)
                }
            },
            modifier = Modifier.fillMaxWidth()
        ) {
            Text(if (isSignUp) "Sign Up" else "Sign In")
        }
        TextButton(onClick = { isSignUp = !isSignUp }) {
            Text(if (isSignUp) "Already have an account? Sign In" else "Don't have an account? Sign Up")
        }
    }
```

Programski kod 4.2 UI ekrana za prijavu i registraciju

Početni ekran prikazuje polja za unos emaila odnosno lozinke te gumbe za prijavu i registraciju. Koristi se *remember* za praćenje unosa emaila i lozinke. Kada korisnik klikne na gumb za prijavu ili registraciju, pozivaju se odgovarajuće metode iz *AuthViewModel* (*signIn* ili *signUp*). Ako je prijava ili registracija uspješna, *LaunchedEffect* preusmjerava korisnika na glavni ekran (engl. *HomeScreen*).

4.3. API Sučelje

Sučelje prikazano programskim kodom 4.3 *ApiFootballService* je sučelje koje definira sve *REST API* pozive koje aplikacija koristi.

```
interface ApiFootballService {
    @Headers(
        "x-rapidapi-key: ac3d1aca25msh93b53f0edb584a3p1d60eajsn11a7967b27d9",
        "x-rapidapi-host: api-football-v1.p.rapidapi.com"
    )
    @GET("v3/standings")
    suspend fun getStandings(
        @Query("league") leagueId: Int,
        @Query("season") season: Int
    ): Response<ApiResponse>

    @Headers(
        "x-rapidapi-key: ac3d1aca25msh93b53f0edb584a3p1d60eajsn11a7967b27d9",
        "x-rapidapi-host: api-football-v1.p.rapidapi.com"
    )
    @GET("v3/fixtures")
    suspend fun getFixtures(
        @Query("league") leagueId: Int,
        @Query("season") season: Int
    ): Response<FixturesResponse>
```

Programski kod 4.3.Dio programskog koda sučelja ApiFootballService

Metode *GET* i *Headers* koriste *Retrofit* za definiranje HTTP zahtjeva. *Headers* metoda u *Retrofit-u* se koristi za specificiranje statičkih zaglavlja koja će biti uključena u svaki HTTP zahtjev prema API-ju. Zaglavlja mogu sadržavati informacije kao što su API ključ za autentifikaciju, *Content-Type, User-Agent* i slično. *GET* metoda u *Retrofit-u* se koristi za definiranje *HTTP GET* zahtjeva. *GET* zahtjevi se koriste za dohvaćanje podataka sa servera. U kontekstu *Retrofit-a,* koristi se za definiranje krajnje točke (engl. end-point) API-ja i parametara koji će biti proslijeđeni serveru.

Kada se žele dohvatiti informacije o stanju na tablici za određenu ligu, *FootballApi* sučelje koristi definiran *GET* zahtjev da bi poslao *HTTP GET* zahtjev prema *v3/standings* krajnjoj točki, uz specificiranje *leagueId* i *season* parametara.

4.4. Retrofit Instance

Klasa *RetrofitInstance* služi za konfiguriranje i inicijalizaciju *Retrofit* objekta, koji se koristi za obavljanje mrežnih zahtjeva prema REST API-ju. Programsko rješenje ove klase prikazano je programskim kodom 4.4.

```
object RetrofitInstance {
    private val client = OkHttpClient.Builder().build()

    val api: ApiFootballService by lazy {
        Retrofit.Builder() Retrofit.Builder
            .baseUrl("https://api-football-v1.p.rapidapi.com/") Retrofit.Builder
            .client(client)
            .addConverterFactory(GsonConverterFactory.create())
            .build() Retrofit
            .create(ApiFootballService::class.java)
    }
}
```

Programski kod 4.4. Klasa Retorfit Instance

Ovo programsko rješenje osigurava centraliziranu konfiguraciju za obavljanje svih mrežnih zahtjeva prema API-ju za nogometne podatke, što uključuje korištenje *OkHttpClient* za osnovne mrežne operacije i *Gson* za pretvaranje *JSON* odgovora u odgovarajuće Kotlin objekte.

4.5. FootballLeague ViewModel

FootballLeagueViewModel je klasa koja upravlja poslovnom logikom i podacima aplikacije. Uključuje metode za dohvaćanje podataka s mreže, upravljanje stanjem podataka, te komunikaciju s korisničkim sučeljem putem *LiveData* objekata. *ViewModel* omogućava aplikaciji da „preživi“ promjene konfiguracije, kao što su rotacije ekrana, bez gubitka podataka. Programski kod 4.5 prikazuje metodu za dohvaćanje podataka o trenutnom poretku nogometne lige.

```
fun fetchStandings(leagueId: Int, season: Int) {
    viewModelScope.launch {
        try {
            val response :Response<ApiResponse> = repository.getStandings(leagueId, season)
            if (response.isSuccessful && response.body() != null) {
                _standings.postValue(response.body()!!.response.flatMap { it.league.standings.flatten() })
            }
        } catch (e: Exception) {
            Log.e( tag: "FootballLeagueViewModel", msg: "Error fetching standings", e)
        }
    }
}
```

Programski kod 4.5. Metoda za dohvaćanje unutar klase

4.6.FootballLeague ViewModel Factory

Klasa *FootballLeagueViewModelFactory* se koristi za stvaranje instance *FootballLeagueViewModel* s potrebnim parametrima. Ovo je posebno korisno kada *ViewModel* zahtijeva određene ovisnosti koje nisu jednostavni tipovi (npr. Repozitoriji ili API servisi) koji se ne mogu direktno proslijediti putem konstruktora *ViewModela* (programski kod 4.6).

```
class FootballLeagueViewModelFactory(
    private val repository: FootballRepository
) : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        return FootballLeagueViewModel(repository) as T
    }
}
```

Programski kod 4.6. Klasa FootballLeagueViewModelFactory

Ovaj pristup omogućuje ubrizgavanje ovisnosti u *ViewModel* koristeći *ViewModelFactory*, što je ključna tehnika za održavanje čiste arhitekture i boljeg testiranja *ViewModela*.

4.7. FootballRepository

FootballRepository je klasa koja služi kao posrednik između *ViewModel*-a i izvora podataka (API-ja ili lokalne baze podataka). Ova klasa enkapsulira logiku za dohvaćanje podataka i pruža čiste metode koje se mogu koristiti u *ViewModel*-ima. Na taj način, kod za dohvaćanje podataka je odvojen od poslovne logike aplikacije, što omogućava lakšu održivost i testiranje. Dio metoda unutar klase prikazan je programski kodom 4.7.

```
class FootballRepository(private val api: ApiFootballService) {

    suspend fun getStandings(leagueId: Int, season: Int): Response<ApiResponse> {
        return api.getStandings(leagueId, season)
    }

    suspend fun getFixtures(leagueId: Int, season: Int): Response<FixturesResponse> {
        return api.getFixtures(leagueId, season)
    }

    suspend fun getTransfers(teamId: Int): Response<TransfersResponse> {
        return api.getTransfers(teamId)
    }
```

Programski kod 4.7 Metode unutar klase FootballRepository

4.8. FirestoreRepository

Klasa *FirestoreRepository* u aplikaciji je odgovorna za rad s podacima o komentarima i njihovo upravljanje putem *Firestore* baze podataka. Ova klasa koristi *Firestore* API kako bi omogućila dinamičko dohvaćanje i spremanje komentara vezanih uz odabranu utakmicu.

Funkcionalnost klase uključuje metodu za dohvaćanje komentara koja koristi *snapshot listener* kako bi stalno pratila promjene u bazi podataka u stvarnom vremenu. Kada se promjene dogode, podaci se ažuriraju i prikazuju na korisničkom sučelju, što osigurava da aplikacija uvijek prikazuje najnovije komentare korisnicima. Osim toga, klasa omogućava dodavanje novih komentara u *Firestore* bazu pomoću metode koja prima identifikator utakmice i tekst komentara te ga dodaje u odgovarajuću kolekciju unutar baze podataka. Metode unutar ove klase prikazane su programskim kodom 4.8.

```
fun getComments(matchId: String): LiveData<List<Comment>> {
    val commentsLiveData = MutableLiveData<List<Comment>>()
    firestore.collection( collectionPath: "matches").document(matchId).collection( collectionPath: "comments") CollectionReference
        .orderBy( field: "timestamp") Query
        .addSnapshotListener { snapshot, e ->
            if (e != null) {
                Log.w( tag: "FirestoreRepository", msg: "Listen failed.", e)
                return@addSnapshotListener
            }

            if (snapshot != null && !snapshot.isEmpty) {
                val comments : (Mutable)List<Comment!> = snapshot.toObjects(Comment::class.java)
                commentsLiveData.postValue(comments)
            } else {
                commentsLiveData.postValue(emptyList())
            }
        }
    return commentsLiveData
}


fun addComment(matchId: String, comment: Comment) {
    firestore.collection( collectionPath: "matches").document(matchId) DocumentReference
        .collection( collectionPath: "comments") CollectionReference
        .add(comment) Task<DocumentReference!>
        .addOnSuccessListener { Log.d( tag: "FootballRepository", msg: "Comment added successfully!") }
        .addOnFailureListener { e -> Log.w( tag: "FootballRepository", msg: "Error adding comment", e) }
}
```

Programski kod 4.8 Funkcije unutar klase FirestoreRepository

4.9. Podatkovne klase

Podatkovne klase u Kotlinu služe za pohranu podataka i omogućuju jednostavnu manipulaciju s njima. Kod dohvaćanja podataka iz API-ja, podatkovne klase igraju ključnu ulogu u strukturiranju i rukovanju tim podacima.

```kotlin
data class ApiResponse(
    @SerializedName("response")
    val response: List<StandingsResponse>
)

data class StandingsResponse(
    @SerializedName("league")
    val league: League
)

data class League(
    @SerializedName("id")
    val id: Int,
    @SerializedName("name")
    val name: String,
    @SerializedName("country")
    val country: String,
    @SerializedName("logo")
    val logo: String,
    @SerializedName("flag")
    val flag: String?,
    @SerializedName("season")
    val season: Int,
    @SerializedName("standings")
    val standings: List<List<TeamStanding>>
)

data class TeamStanding(
    @SerializedName("rank")
    val rank: Int,
    @SerializedName("team")
    val team: Team,
    @SerializedName("points")
    val points: Int,
    @SerializedName("goalsDiff")
    val goalsDiff: Int,
    @SerializedName("group")
    val group: String,
    @SerializedName("form")
    val form: String,
    @SerializedName("status")
    val status: String,
    @SerializedName("description")
    val description: String?,
    @SerializedName("all")
    val all: AllStats
)
```

```kotlin
data class AllStats(
    @SerializedName("played")
    val played: Int,
    @SerializedName("win")
    val win: Int,
    @SerializedName("draw")
    val draw: Int,
    @SerializedName("lose")
    val lose: Int,
    @SerializedName("goals")
    val goals: TeamGoals
)

data class Team(
    @SerializedName("id")
    val id: Int,
    @SerializedName("name")
    val name: String,
    @SerializedName("logo")
    val logo: String
)

data class TeamGoals(
    @SerializedName("for")
    val goalsFor: Int,
    @SerializedName("against")
    val goalsAgainst: Int
)
```

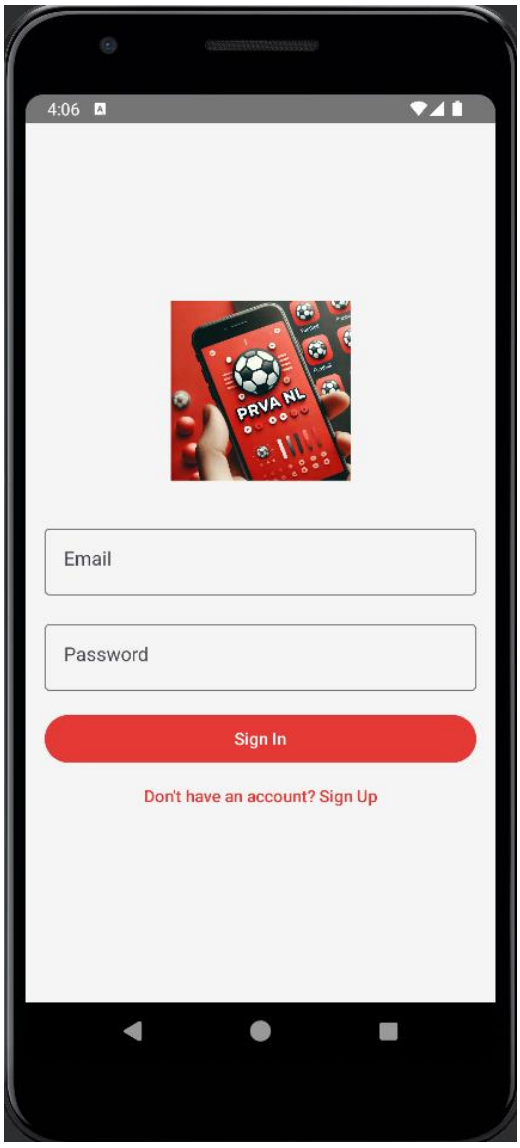
Programski kod 4.9 Podatkovne klase korištene za ekran Standings.kt

Podaci iz podatkovnih klasa popunjavaju se kroz nekoliko koraka, koristeći *Retrofit* za dohvaćanje podataka iz API-ja, te *ViewModel* i *LiveData* za upravljanje podacima i njihovim prikazom u korisničkom sučelju. *Retrofit* obavlja dohvaćanje podataka, repozitorij upravlja podacima, *ViewModel* omogućuje komunikaciju između podataka i UI-a, a *composable* funkcije prikazuju podatke korisniku.

5. KONAČAN IZGLED I FUNKCIONALNOSTI APLIKACIJE

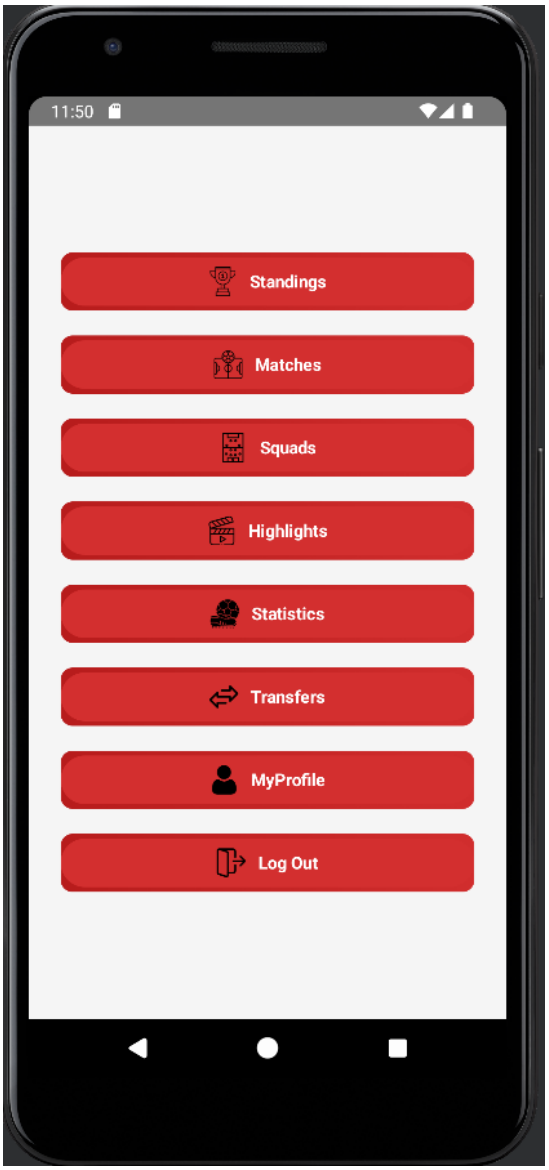
U ovom poglavlju prikazat će se konačan izgled i funkcionalnosti mobilne aplikacije, pomoću slika zaslona i objašnjenja, biti će prikazano kako aplikacija omogućuje korisnicima praćenje najnovijih informacija, statistika i rezultata iz Prve hrvatske nogometne lige.

Ekran za prijavu i registraciju prikazan na slici 5.1 je prvi zaslon aplikacije koji korisnicima omogućava prijavu ili registraciju pomoću email adrese i lozinke. Korisnici se mogu prijaviti ako već imaju račun ili se mogu registrirati ako su novi korisnici.



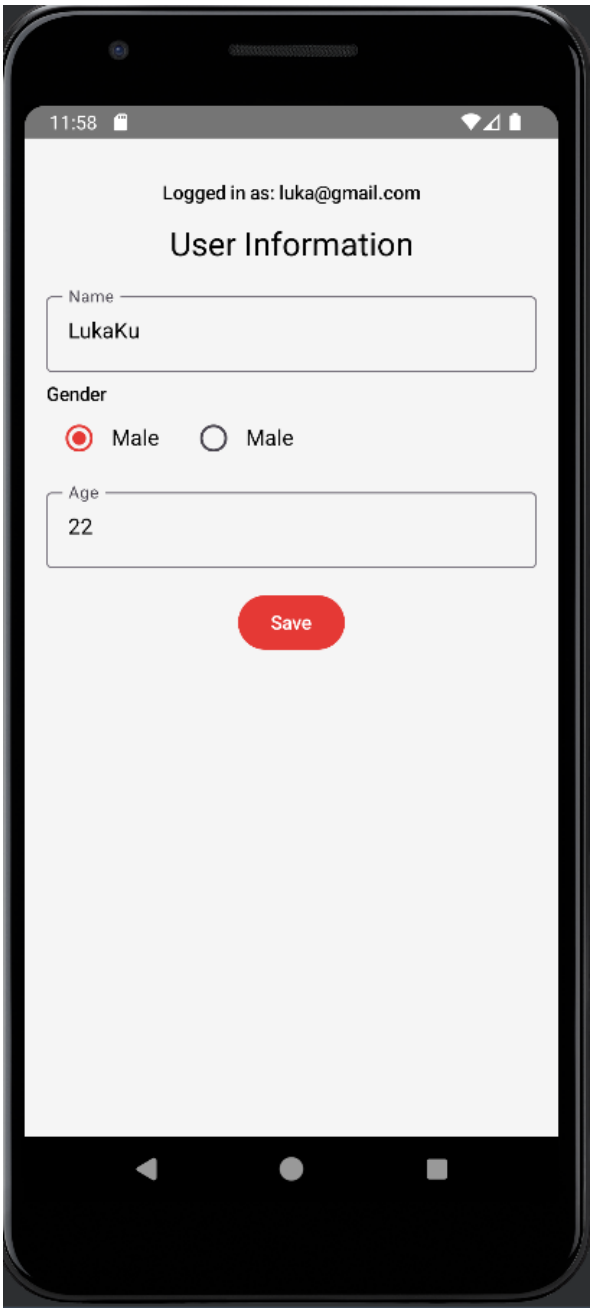
Slika 5.1. Ekran za prijavu i registraciju

Početni zaslon aplikacije (slika 5.2) prikazuje glavne kategorije i funkcionalnosti koje su korisnicima dostupne. Nakon uspješne prijave odnosno registracije, korisnici dolaze na ovaj zaslon gdje mogu birati između različitih opcija kao što su pregled trenutnog poretka, rezultate odigranih utakmica, transfera i statistike.



Slika 5.2 Početni ekran aplikacije

Ono što se prvo korisniku preporučuje jest da posjeti ekran *MyProfile* koji omogućava korisnicima da unesu svoje osnovne podatke poput korisničkog imena, spola i godina (slika 5.3), te da ih ažuriraju po potrebi. Sve promjene koje korisnik unese na ovom ekranu automatski se ažuriraju u stvarnom vremenu, zahvaljujući integraciji s *Firebaseom*.



Slika 5.3 Ekran za uređivanje profila korisnika

Također, *MyProfile* ekran sadrži prikaz poruke o uspješnom ažuriranju podataka, poboljšavajući korisničko iskustvo potvrdom o uspješnom unosu i ažuriranju informacija.

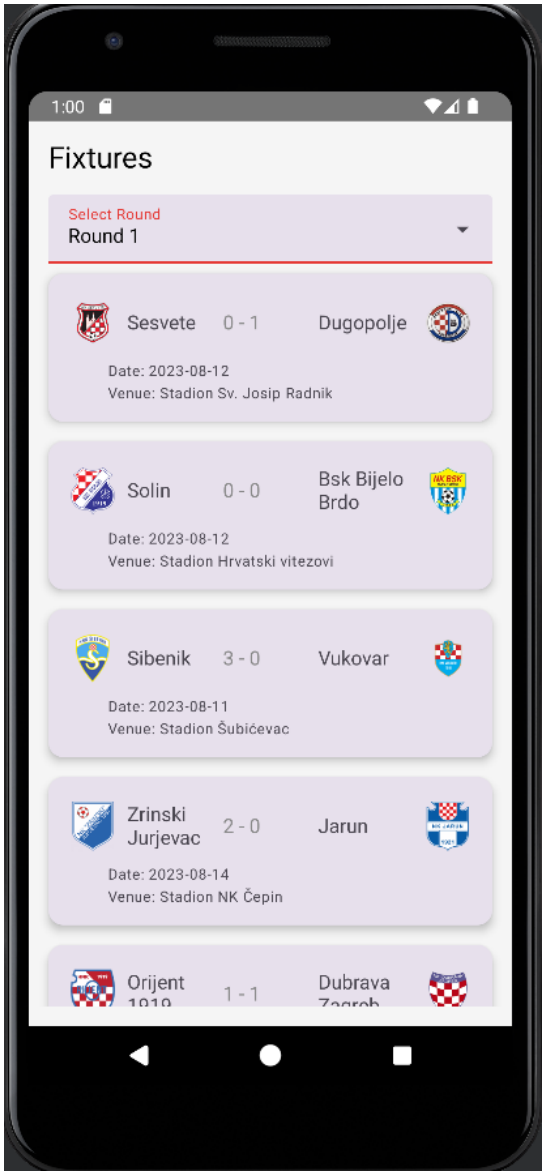
Ekran Standings u aplikaciji prikazuje poredak nogometnih timova koristeći podatke preuzete putem API-ja (slika 5.4). Na ovom ekranu korisnik može pregledati trenutni poredak timova za odabranu sezonu, kao i broj osvojenih bodova pojedinog tima te gol razliku.

Slika 5.4 Ekran za prikaz trenutnog poretka klubova

Aplikacija koristi gumb za odabir sezone, što korisniku omogućava da jednostavno prebacuje između različitih sezona. Kada korisnik odabere željenu sezonu, aplikacija šalje zahtjev API-ju za preuzimanje podataka o poretku za tu sezonu. Dobiveni podaci se dinamički prikazuju u aplikaciji, ažurirajući informacije u stvarnom vremenu.

Ekran *Matches* prikazuje popis svih utakmica za sezonu 2023./24. pri čemu korisnik može izabrati točno određeno kolo koje ga zanima putem padajućeg izbornika. Podaci o utakmicama se

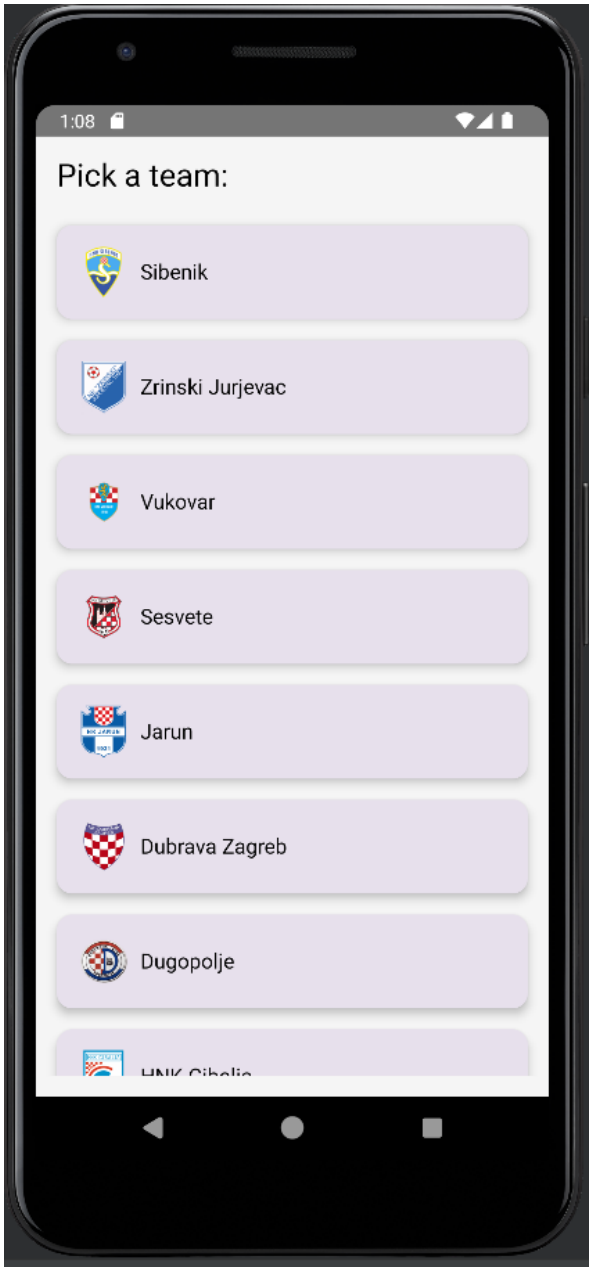
prikazuju u obliku liste, a svaka utakmica uključuje informacije o domaćinu i gostujućoj ekipi, rezultatima, datumu i mjestu održavanja (slika 5.5). Klikom na određenu utakmicu korisnik se preusmjerava na ekran gdje može dodati komentare vezane uz tu utakmicu koji će biti prikazani uz komentare svih drugih korisnika (slika 5.6).



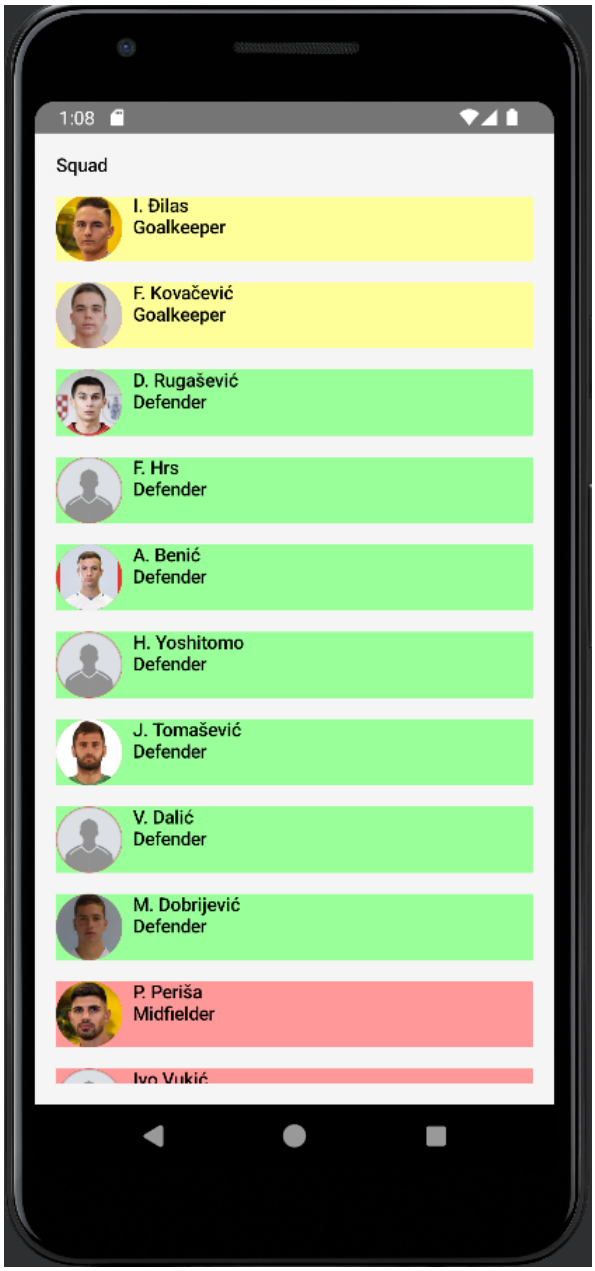
Slika 5.5 Ekran za prikaz utakmica prvog kola

Slika 5.6 Primjer ekrana za komentare na utakmicu

Squads je ekran koji omogućuje korisnicima popisa timova unutar prve hrvatske nogometne lige. Ekran prikazuje sve timove koji sudjeluju u ligi, koristeći podatke dobivene s API-ja (slika 5.7). Korisnici mogu kliknuti na bilo koji tim kako bi dobili popis svih igrača tog tima, što ih vodi do ekrana *SquadDetails* (slika 5.8). Popis igrača određenog tima poredan je od golmanske pozicije pa sve do napadača pri čemu je svaka pozicija obojana određenom bojom (npr. braniči – zelena, veznjaci – crvena itd.).

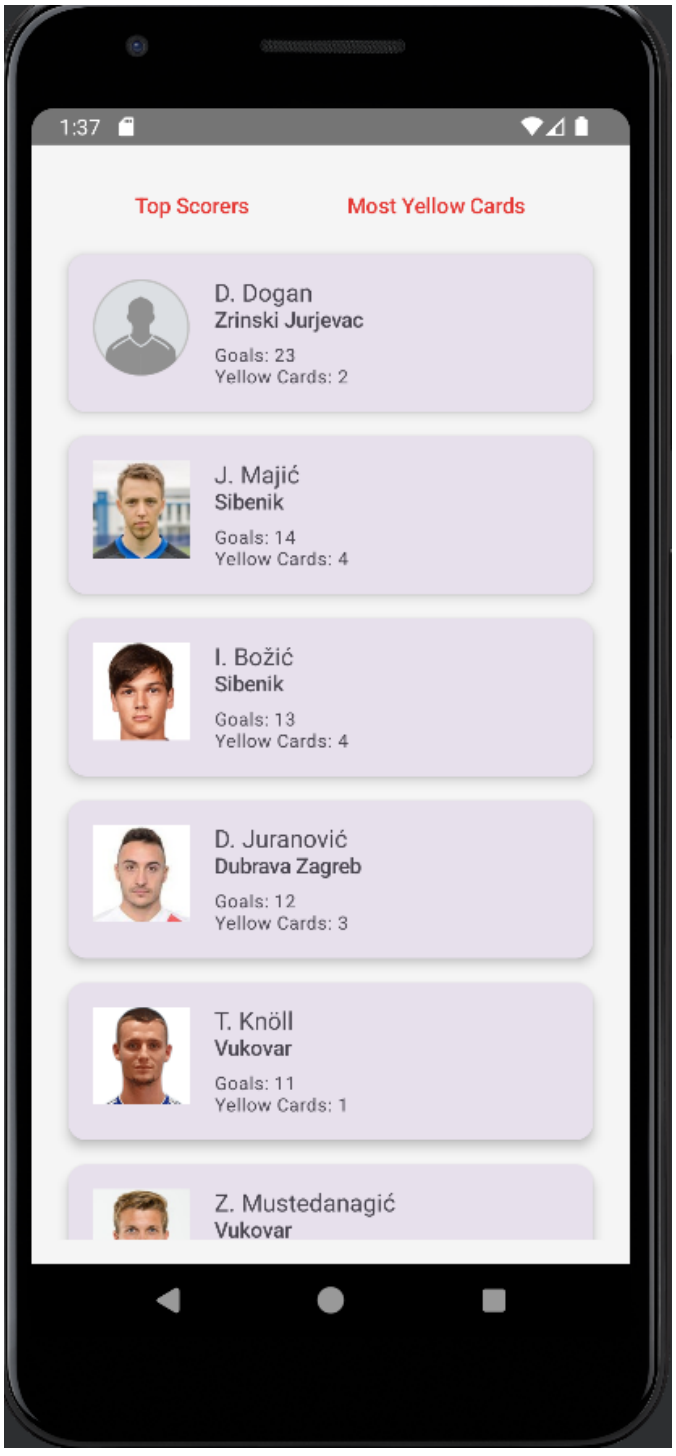


Slika 5.7 Ekran sa popisom timova



Slika 5.8 Primjer sastava za izabrani tim

Ekran *Statistics* u aplikaciji za praćenje nogometne lige prikazuje statističke podatke povezane s ligom ili ekipama unutar lige. Cilj ekrana je omogućiti korisnicima pregled ključnih statistika kao što su najbolji strijelci te igrači sa najvećim brojem žutih kartona u sezoni (slika 5.9). Korisnik putem gumbova bira statistiku koju želi vidjeti.



Slika 5.9 Prikaz najboljih strijelaca unutar ekrana za statistiku

6. ZAKLJUČAK

Aplikacija za završni rad "Prva Nogometna Liga" uspješno je ispunila svoj glavni cilj - pružanje korisnicima ažurnih informacija o Prvoj hrvatskoj nogometnoj ligi, uključujući rezultate, statistike, tablicu poretka te transfere. Kroz razvoj aplikacije korištene su moderne tehnologije kao što su Android Studio, Kotlin, *Jetpack Compose*, *Firebase* i RESTful API, što je omogućilo kreiranje intuitivnog korisničkog sučelja i stabilnog *backend* sustava.

Međutim, tijekom razvoja aplikacije identificirani su i određeni nedostaci. Jedan od glavnih izazova bio je rad s podacima iz različitih sezona, što je zahtijevalo dodatnu logiku za obradu i prikaz informacija iz API-ja. Također, upravljanje korisničkim podacima i autentifikacijom preko Firebasea pokazalo se kao područje koje zahtijeva daljnju optimizaciju i poboljšanja kako bi se osigurala maksimalna sigurnost i jednostavnost korištenja. Daljnji rad na aplikaciji trebao bi se fokusirati na proširenje funkcionalnosti i povećanje korisničkog iskustva uključujući i unaprjeđenje forme za unos komentara i ocjenjivanja utakmica, čime bi se osigurala njezina relevantnost i korisnost za širu bazu korisnika.

LITERATURA

[1] "HRnogomet" [online], https://play.google.com/store/apps/details?id=com.hr.nogomet&hl=hr, datum pristupa: 2.9.2024

[2] "Sofascore - rezultati uživo" [online], https://play.google.com/store/apps/details?id=com.sofascore.results&hl=hr&gl=AL , datum pristupa: 2.9.2024

[3] "Rezultati.hr" [online], https://www.rezultati.com/ , datum pristupa: 2.9.2024

[4] K.T.Hanna, I.Wigmore, "What is a mobile app (mobile application)?" [online], https://www.techtarget.com/whatis/definition/mobile-app, datum pristupa: 23.6.2024

[5] Andorid Developers, "Meet Android Studio" [online], https://developer.android.com/studio/intro, datum pristupa: 23.6.2024

[6] KotlinLang, "Kotlin" [online], https://kotlinlang.org/, datum pristupa: 23.6.2024

[7] R. Elizarov,S. Aigner, S. Isakova, 2024., "Kotlin in Action , Second Edition", https://learning.oreilly.com/library/view/kotlin-in-action/9781617299605/, datum pristupa: 13.9.2024

[8] Andoroid Developers, "JetPack Compose" [online], https://developer.android.com/develop/ui/compose, datum pristupa: 23.6.2024

[9] T. Künneth, 2023., "Android UI Development with Jetpack Compose - Second Edition", https://learning.oreilly.com/library/view/android-ui-development/9781837634255/ , datum pristupa: 12.9.2024

[10] GeekForGeeks, "Firebase – Introduction" [online], https://www.geeksforgeeks.org/firebase-introduction/, datum pristupa: 23.6.2024

[11] G. Batschinski, "What is Firebase? All secrets unlocked" [online], https://blog.back4app.com/firebase/, datum pristupa: 23.6.2024

[12] J. Gough, D. Bryant, M. Auburn, 2022., "Mastering API Architecture", https://learning.oreilly.com/library/view/mastering-api-architecture/9781492090625/ , datum pristupa: 13.9.2024

[13] RedHat, "What is a REST API? " [online], https://www.redhat.com/en/topics/api/what-is-a-rest-api, datum pristupa: 23.6.2024

SAŽETAK

Glavni problem obrađen u ovom završnom radu bio je razvoj mobilne aplikacije koja korisnicima pruža aktualne informacije o Prvoj hrvatskoj nogometnoj ligi na intuitivan i jednostavan način. Aplikacija "Prva Nogometna Liga" razvijena je korištenjem suvremenih tehnologija, uključujući Android Studio, Kotlin, Jetpack Compose i Firebase, s ciljem olakšavanja praćenja rezultata utakmica, statistika, trenutne tablice i transfera unutar lige. U procesu razvoja, posebna pažnja posvećena je dizajnu korisničkog sučelja te implementaciji funkcionalnosti kao što su autentifikacija korisnika, upravljanje podacima, mogućnost komentiranja utakmica, slanje korisničkih unosa na Firebase te dohvat informacija putem RESTful API-ja. Rješenje je testirano i evaluirano u nekoliko različitih ekrana, čime je postignuta stabilna aplikacija koja korisnicima omogućuje jednostavan pristup potrebnim informacijama u stvarnom vremenu. Prepoznati nedostaci aplikacije uključuju ograničene interaktivne mogućnosti te nedostatak naprednih statističkih prikaza, što pruža prostor za buduća unaprjeđenja.

Ključne riječi: Android Studio, autentifikacija, Kotlin, mobilna aplikacija, nogomet

ABSTRACT

The main problem addressed in this final thesis was the development of a mobile application that provides users with up-to-date information about the Croatian First Football League in an intuitive and straightforward manner. The application, "Prva Nogometna Liga," was developed using modern technologies, including Android Studio, Kotlin, Jetpack Compose, and Firebase, with the goal of facilitating the tracking of results, statistics, current league standings, and transfers. During development, special attention was given to the design of the user interface and the implementation of functionalities such as user authentication, data management, match commenting, user input submission to Firebase, and information retrieval via RESTful API. The solution was tested and evaluated across various screens, resulting in a stable application that provides users with easy access to necessary real-time information. Identified shortcomings of the application include limited interactive features and a lack of advanced statistical displays, leaving room for future improvements.

Keywords: Android Studio, authentication, football, Kotlin, mobile application

PRILOG

GitHub poveznica na kojoj se nalazi cijelo programsko rješenje aplikacije: https://github.com/stankela8/Zavrsni-rad

ŽIVOTOPIS

Luka Stanković, rođen je 7. kolovoza 2002. u Osijeku. Pohađao osnovnu školu Milka Cepelića u Vuki. Srednju školu upisao je u Osijeku, smjer tehničar za računalstvo u Elektrotehničkoj i prometnoj školi Osijek. Godine 2021. nakon uspješno položene mature upisuje fakultet te postaje student na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku na sveučilišnom preddiplomskom studiju Računarstva, smjer računalno inženjerstvo. Kroz srednju školu i fakultet upoznao se s programskim jezicima C,C++,C#, Java, Kotlin, PYTHON te alatima i jezicima za izradu web aplikacija kao što su HTML, CSS i PHP. U slobodno vrijeme rado prati i igra nogomet te je iz toga stekao i motivaciju za izradu ove aplikacije za završni rad.