Android aplikacija za praćenje rezultata Formule 1

Kostić, Mislav

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

Sveučilišni studij

ANDROID APLIKACIJA ZA PRAĆENJE REZULTATA FORMULE 1

Diplomski rad

Mislav Kostić

Osijek, 2023.

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

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 12.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Mislav Kostić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1213R, 07.10.2021.
OIB studenta:	44052557220
Mentor:	prof. dr. sc. Krešimir Nenadić
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Ivica Lukić
Član Povjerenstva 1:	prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 2:	Robert Šojo, mag. ing. comp.
Naslov diplomskog rada:	Android aplikacija za praćenje rezultata Formule 1
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Opisati funkcionalnosti Android aplikacije koja bi se koristila za praćenje rezultata i informacija utrka Formule 1. Dati opis dizajna baze podataka ili nekog drugog načina lokalne pohrane podataka. Omogućiti korisnicima postavljanje notifikacija za nadolazeće događaje koji se mogu dohvatiti preko vanjskog API-ja. Izraditi Android aplikaciju prema navedenim specifikacijama i opisati postupak izrade. Tema rezervirana: Mislav Kostić
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
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: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	12.09.2023.

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, 03.10.2023.

Ime i prezime studenta:	Mislav Kostić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1213R, 07.10.2021.
Turnitin podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Android aplikacija za praćenje rezultata Formule 1**

izrađen pod vodstvom mentora prof. dr. sc. Krešimir Nenadić

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 studenta:

SADRŽAJ

1. UVOD

Formula 1, često nazivana „kraljicom automobilskog sporta“, predstavlja jedan od najuzbudljivijih i najprestižnijih sportskih događaja na svijetu. Sa svojom bogatom poviješću, inovacijama u inženjeringu i brzinama koje pomiču granice ljudske sposobnosti, Formula 1 je privukla milijune obožavatelja širom svijeta. Pošto je broj novih obožavatelja sporta iz godine u godinu sve veći, postoji potreba za sve inovativnijim načinima praćenja ovog sporta, osim klasičnog prijenosa glavnih utrka uživo.

Motivacija za izradu ovog diplomskog rada potječe od dugogodišnjeg praćenja utrka Formule 1 i ljubavi prema tom sportu. Tradicionalni način praćenja ovog sporta je gledanje prijenosa utrka uživo putem medija, no želja mi pružiti obožavateljima jedan od modernijih načina praćenja utrka, putem Android mobilne aplikacije. Korištenjem ove aplikacije korisnici mogu pratiti sve utrke putem dostupnih rezultata ili postavljanjem podsjetnika za početak prijenosa u živo kako ne bi propustili gledanje utrke.

U drugom poglavlju opisana su slična rješenja te je napravljena kratka analiza i usporedba s aplikacijom ovoga rada. U trećem poglavlju opisan je proces izrade same aplikacije te je napravljen kratak osvrt na alate i tehnologije koje su korištene prilikom izrade. Konačno, u četvrtom poglavlju detaljno su opisane i slikovno prikazane sve funkcionalnosti aplikacije uz popratne upute za korištenje.

2. PREGLED PODRUČJA TEME

U ovome poglavlju prikazano je nekoliko istraženih Android aplikacija koje svojom tematikom rješavaju istu problematiku kojom se bavi ovaj diplomski rad. Osim toga, prikazane su ključne razlike između istraženih aplikacija te izrađene aplikacije u ovome diplomskome radu. Za početak, ukratko je opisano i pojašnjeno što uopće jest sport Formula 1.

2.1. Formula 1

Formula 1 je najprestižnije i najpopularnije automobilističko natjecanje na svijetu. To je jedna vrsta automobilističkog sporta koji privlači milijune obožavatelja diljem svijeta. U Formuli 1 vozači se natječu u posebno dizajniranim bolidima, razvijenim za vrhunske performanse i brzinu.

Utrke Formule 1 odvijaju se na stazama diljem svijeta, a vozači se natječu u seriji utrka tijekom sezone koja traje od ožujka do studenog. Natjecanje je organizirano u skladu s pravilima Međunarodne automobilističke federacije FIA *(franc. Fédération Internationale de l'Automobile)* koje reguliraju sve aspekte utrka, sigurnost vozača i tehničke specifikacije bolida. [1]

2.2. F1 Race Guide

Aplikacija *F1 Race Guide* korisnicima služi kao vodič kroz sve trkaće staze u sezoni Formule 1. Na interaktivnoj karti pruža pregled pojedine odabrane trkaće staze te daje dodatne informacije o povijesti staze, tehničkim informacijama i slično.

Razlika ove aplikacije od aplikacije kojom se bavi ovaj diplomski rad su dostupnost podataka te njihov prikaz na ekranu. F1 Race Guide aplikacija je orijentirana na prikaz podataka uz interaktivnu kartu te ne sadrži informacije o trenutnom bodovnom poretku momčadi i vozača.

Na slici 2.1 prikazana je interaktivna karta trkaće staze u Australiji.

Slika 2.1. *Prikaz interaktivne karte u aplikaciji F1 Race Guide* [2]

2.3. Racing Calendar 2023

Aplikacija *Racing Calendar 2023* svojim korisnicima na vrlo pregledan i minimalistički način prikazuje sezonski kalendar utrka Formule 1. Osim kalendara, aplikacija pruža podatke o trenutnom bodovnom stanju svih vozača u trenutnoj trkaćoj sezoni.

Ključna razlika između aplikacije *Racing Calendar 2023* i aplikacije ovog diplomskog rada je prikaz podataka na ekranu. Osim načina na koji su podaci prikazani, ova aplikacija ne sadrži tehničke podatke o pojedinoj trkaćoj stazi dok su ti podaci dostupni u aplikaciji diplomskoga rada.

Na slici 2.2. prikazan je glavni zaslon aplikacije Racing Calendar 2023.

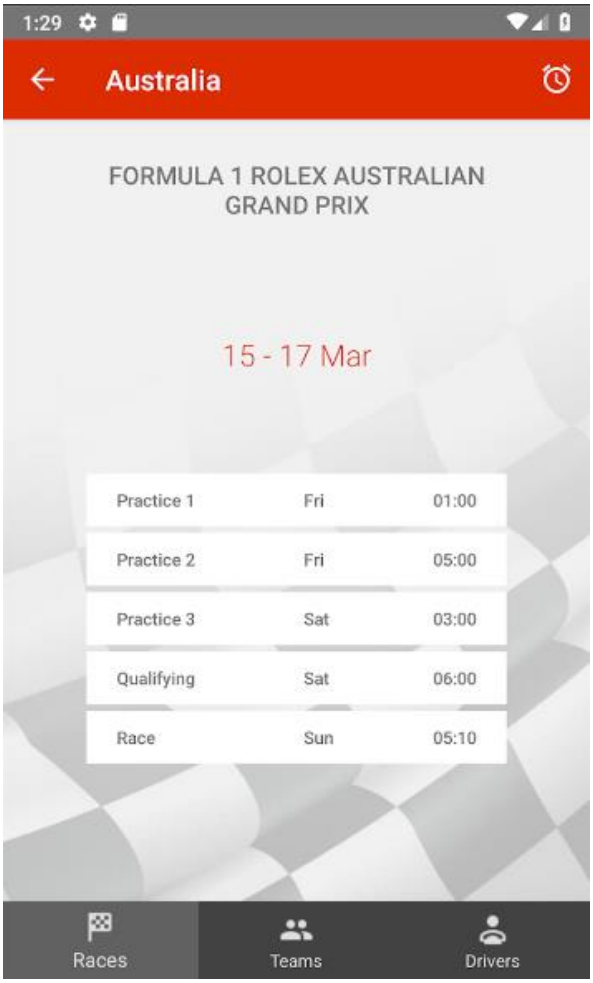
Slika 2.2. *Prikaz kalendara utrka u aplikaciji Racing Calendar 2023* [3]

2.4. Formula 2023 CalendarStandings

Aplikacija *Formula 2023 CalendarStandings* prikazuje sažeti raspored događanja jedne utrke sa sezonskog kalendara utrka. Osim rasporeda događanja, korisnicima je omogućen momčadski bodovni prikaz unutar aktivne sezone te bodovni poredak vozača.

U ovoj aplikaciji ne postoji mogućnost prikaza podataka o trenutnoj trkaćoj stazi koja je sljedeća na redu u kalendarskom prikazu, dok u aplikaciji ovog diplomskog rada, postoji. Osim toga, aplikacije se razlikuju i po dizajnu ekrana koji se prikazuju.

Na slici 2.3. prikazan je sažeti prikaz rasporeda događanja jedne od utrka.



Slika 2.3. *Prikaz događanja u aplikaciji Formula 2023 CalendarStandings* [4]

2.5. FORONE: F1 Calendar 2023

Aplikacija *FORONE: F1 Calendar 2023* osim prikaza sezonskog trkaćeg kalendara, korisnicima daje mogućnost prikaza informacija o pojedinom vozaču. Na slici 2.4. prikazan je ekran s podacima o jednom od vozača Formule 1 te se ondje mogu vidjeti podaci poput osobnih podataka, trkaćih postignuća kroz karijeru i slično.

Glavna razlika između aplikacija je dostupnost podataka. Ovdje je fokus na prikazu informacija o samim vozačima, dok podaci o trenutnoj trkaćoj stazi gotovo ne postoje. Osim toga, velika je razlika u tematskom dizajnu aplikacija.

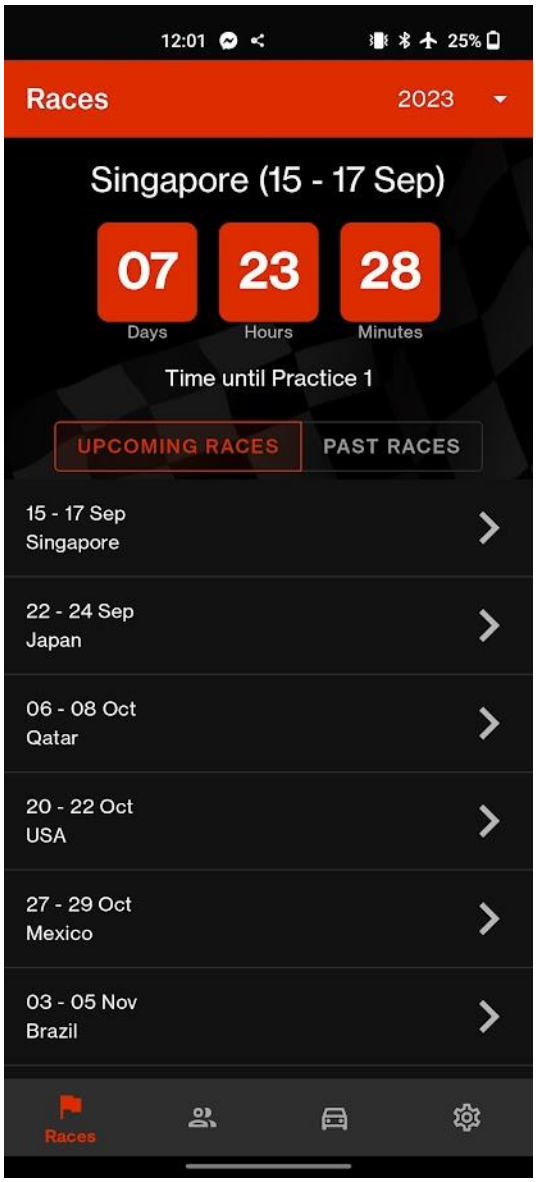
Slika 2.4.. *Prikaz podataka o vozaču u aplikaciji FORONE: F1 Calendar 2023* [5]

2.6. Formula 2023 Calendar

Aplikacija *Formula 2023 Calendar* svojim minimalističkim sučeljem prikazuje nadolazeće utrke s aktualnog trkaćeg kalendara te uz to odbrojava dane, sate i minute do početka utrke. Točno odbrojavanje vremena postignuto je s Android umetkom sata koji broji vrijeme unatrag.

Osnovna razlika između ove aplikacije i aplikacije ovog diplomskog rada jesu podaci koji su dostupni korisnicima. U ovoj aplikaciji postoji kalendar i sat koji odbrojava vrijeme do početka sljedeće utrke, dok podaci o vozačima i trkaćim stazama uopće ne postoje.

Na slici 2.5. prikazan je početni zaslon aplikacije Formula 2023 Calendar gdje se može vidjeti vremensko odbrojavanje nadolazeće utrke, a odmah ispod sata nalazi se sortirani popis slijedećih utrka sa sezonskog kalendara.



Slika 2.5..*Prikaz kalendara u aplikaciji Formula 2023 Calendar* [6]

3. ANDROID APLIKACIJA

U ovome poglavlju opisan je proces izrade same aplikacije, te nešto detaljnije pojasniti pojedine korištene Android komponente. Također, detaljno su obrazloženi razlozi zašto su pojedini alati korišteni i na koji način su bili upotrijebljeni u izradi ove aplikacije.

3.1. Izrada aplikacije

Ova Android aplikacija pisana je Kotlin programskim jezikom. Kotlin je Googleov programski jezik koji je nastao kao alternativa jeziku Java za razvoj Android aplikacija. Najveća prednost kod Kotlina je interoperabilnost s Javom. Na taj se način može koristiti postojeći Java kôd unutar aplikacija pisanih u Kotlinu [7].

Cjelokupni razvojni proces ove aplikacije bio je moguć uz korištenje Android Studio razvojnog okruženja, koje je dizajnirano za razvoj aplikacija za Android platformu. Android Studio je Googleovo razvojno okruženje koje programerima pruža sve alate i resurse potrebne za izgradnju visokokvalitetnih mobilnih aplikacija. Podržava različite programske jezike poput Java, Kotlin i C++, omogućavajući programerima fleksibilnost pri odabiru jezika koji će koristiti u svojim projektima [8].

3.1.1. Arhitektura

Aplikacija je razvijena slijedeći MVVM (*engl. Model View ViewModel)* arhitekturni obrazac.

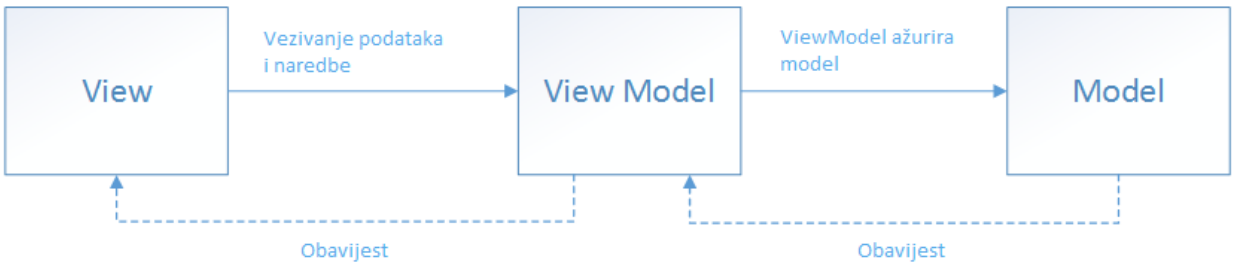
U MVVM arhitekturi, *Model* predstavlja podatke i poslovnu logiku aplikacije. To može biti baza podataka, web API (*engl. Application Programming Interface*), lokalni spremnik podataka ili bilo koji drugi izvor podataka. Model ne ovisi o korisničkom sučelju i ne zna ništa o njemu.

View predstavlja korisničko sučelje, poput prozora, obrazaca ili elemenata sučelja. *View* je pasivan i ne sadrži poslovnu logiku. Umjesto toga, on prikazuje podatke iz *ViewModela* i reagira na korisničke interakcije.

ViewModel je veza između Modela i *Viewa*. *ViewModel* sadrži podatke koje prikazuje *View*, kao i logiku koja upravlja tim podacima. *ViewModel* se također brine o obradi korisničkih interakcija i komunikaciji s modelom. *ViewModel* također može sadržavati povratne veze koje povezuju *View* s podacima i osiguravaju ažuriranje prikaza kada se podaci promijene.

Glavna prednost MVVM arhitekture je jasna odvojenost odgovornosti između komponenti, što olakšava razumijevanje i održavanje kôda. Također pruža dobru podršku za testiranje, jer se poslovna logika nalazi u *ViewModelu* koji se može testirati odvojeno od korisničkog sučelja.

Na slici 3.1. prikazan je shematski prikaz principa rada MVVM arhitekture.



Slika 3.1..*Shematski prikaz MVVM arhitekture*

3.1.2. Fragmenti

Aplikacija je razvijena prateći obrazac jedne aktivnosti (*engl. Single Activity Pattern).* Postoji jedna osnovna aktivnost te je svaki ekran u aplikaciji predstavljen jednim fragmentom. Fragmenti su jedan od osnovnih građevnih blokova Android aplikacija koji omogućuju organizaciju i upravljanje dijelovima korisničkog sučelja. Oni predstavljaju manje, modularne komponente koje se mogu koristiti unutar aktivnosti kako bi se poboljšala modularnost i prilagodljivost aplikacije.

Fragmenti imaju vlastiti životni ciklus koji je sličan životnom ciklusu aktivnosti. To uključuje metode poput *onCreate(), onStart(), onPause(), onResume(), onStop(),* itd. Ovo omogućuje fragmentima da reagiraju na događaje kao što su promjene konfiguracije uređaja ili prelazak između fragmenata.

Većina podataka koji se dohvaćaju u ovoj aplikaciji su u obliku popisa i listi, a gotovo svaki fragment mora prikazati takve oblike podataka na ekranu pa se u tu svrhu koristi Android komponenta *RecyclerView.* To je komponenta koja se koristi za prikazivanje velikih skupova podataka koji je učinkovit po pitanju resursa. Svaki RecyclerView mora imati svoj *Adapter*. Adapter je odgovoran za povezivanje podataka s prikazom [9].

Na slici 3.2. prikazana je jedna od osnovnih metoda životnog ciklusa fragmenta, *onViewCreated().* U toj metode se izvršava inicijalizacija i postavljanje korisničkog sučelja. Na linijama 40 – 42

izvršava se inicijalizacija *RecyclerViewa*. U slučaju ovog fragmenta, adapteru se u argumentu predaje popis vozača koji se trebaju prikazati na ekranu na liniji 48.

```
37  override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
38      super.onViewCreated(view, savedInstanceState)
39
40      binding.rvDrivers.adapter = adapter
41      val layoutManager = LinearLayoutManager(context, LinearLayoutManager.VERTICAL,  reverseLayout: false)
42      binding.rvDrivers.layoutManager = layoutManager
43
44      viewModel.uiState.observe(viewLifecycleOwner) { state →
45          when (state) {
46              is DriversFragmentUiState.Success → {
47                  hideProgressBar()
48                  adapter.submitList(state.drivers)
49              }
50              is DriversFragmentUiState.Loading → {
51                  showProgressBar()
52              }
53              else → {
54                  Log.d( tag: "response",  msg: "onViewCreated: error")
55              }
56          }
57      }
58  }
```

Slika 3.2..*Prikaz onViewCreated() metode*

3.1.3. Udaljeni podaci

Svi podaci koji se prikazuju unutar aplikacije, dohvaćaju se s udaljenog poslužitelja *ErgastAPI*. ErgastAPI je API koji pruža programsko sučelje za pristup i dohvaćanje podataka iz svjetskog automobilističkog prvenstva Formule 1. Ovaj API sadrži različite podatke o utrkama, vozačima, momčadima, stazama i mnogim drugim aspektima Formule 1. ErgastAPI omogućava programerima pristup i korištenje tih podataka u svojim aplikacijama, web stranicama ili sustavima [10].

Primjeri informacija koje se mogu dobiti korištenjem ErgastAPI-ja uključuju rezultate utrka, statistiku vozača, povijest timova, raspored utrka i mnoge druge relevantne podatke za ljubitelje Formule 1.

Kako bi aplikacija mogla funkcionirati, potrebno je putem *Retrofit* biblioteke dohvatiti tražene podatke s ErgastAPI udaljenog poslužitelja. Nakon uspješnog slanja i obrade zahtjeva, API vraća tražene podatke u JSON (*engl. JavaScript Object Notation*) transportnom obliku koje je moguće dalje obrađivati lokalno, unutar memorije samog mobilnog uređaja. [11]

Na slici 3.3. prikazani su podaci prve utrke s trkaćeg kalendara 2023. godine u JSON obliku. Objekt utrke je složeni objekt koji sadrži atribute i druge objekte unutar sebe. Aplikacija od

poslužiteljske strane prima JSON zapis podataka te iz njega uzme potrebne atribute čije se vrijednosti onda prikazuju na predviđenim mjestima na ekranu.

Slika 3.3. *Prikaz JSON podataka jedne utrke*

Kao što je spomenuto, u aplikaciji se koristi Retrofit biblioteka za dohvaćanje podataka s udaljenog poslužitelja, u ovome slučaju ErgastAPI. Retrofit je popularna biblioteka za Android programiranje koja omogućava jednostavnu izradu HTTP (*engl. Hypertext Transfer Protocol*) zahtjeva i komunikaciju s web uslugama. Ova biblioteka olakšava rad s RESTful API-jima (*engl. Representational State Transfer Application Programming Interface*) na način da programerima omogući definiranje HTTP zahtjeva kao sučelja (*engl. interfaces*) u programskom jeziku Java ili Kotlin.

Za potrebe ove aplikacije napisane su metode koje dohvaćaju podatke s udaljenog API-ja.

Na slici 3.4. prikazane su metode za komunikaciju s udaljenim API-jem koje se koriste unutar ove aplikacije. Sve metode po svojoj funkciji dohvaćaju podatke te zbog toga sadrže oznaku (*engl. annotation) @GET*. Nazivi metoda su opisni te se, prijevodom s engleskog jezika, iz imena metode može pročitati što pojedina metoda radi.

```
10      @GET("current.json")
11      suspend fun getCalendarData(): Response<ApiCalendarResponse>
12
13      @GET("current/driverStandings.json")
14      suspend fun getDriverStandings(): Response<ApiDriverStandingsResponse>
15
16      @GET("current/constructorStandings.json")
17      suspend fun getConstructorStandings(): Response<ApiConstructorStandingsResponse>
18
19      @GET("current/{round}/qualifying.json")
20      suspend fun getQualifyingResults(@Path("round") round: String): Response<ApiQualifyingResultsResponse>
21
22      @GET("current/{round}/results.json")
23      suspend fun getRaceResults(@Path("round") round: String): Response<ApiRaceResultsResponse>
```

Slika 3.4. *Prikaz metode za dohvaćanje podataka s udaljenog poslužitelja*

3.1.4. Prikaz podataka

U prethodnom poglavlju opisan je način dohvaćanja podataka, no potrebno je te iste podatke prikazati na ekranu. Svrha aplikacije je prikazati složeni skup podataka na jednostavan način, razumljiv svim korisnicima Android aplikacije. To se postiglo analizom i raščlanjivanjem (*engl. parsing)* dohvaćenog skupa podataka s udaljenog poslužitelja te tematskom podjelom istih podataka na više cjelina.

Kako je spomenuto, skup podataka koji se dohvaća u sebi sadrži mnoštvo podataka koji nisu potrebni za svrhu ove Android aplikacije. Raščlanjivanje korisnih informacija postiglo se korištenjem tzv. objektnog mapera (*engl. Object Mapper)*. Kako se u ovoj aplikaciji koristi MVVM arhitektura, vrlo je korisno imati objektne mapere kojima se na efikasan i ljudskom oku pregledan način mogu pridružiti vrijednosti podataka s poslužiteljske strane (*engl. backend)* na korisničko sučelje.

Na slici 3.5. prikazan je izvorni kôd objektnog mapera za utrke u ovoj Android aplikaciji. Radi se o objektnom maperu koji kao jedini parametar prima ulazne podatke, dohvaćene od API poslužitelja, a kao krajnji rezultat vraća novi objekt koji sadrži samo proizvoljne atribute iz izvornog objekta, koji se dalje u aplikaciji koristiti ovisno o potrebi.

Također na slici 3.5., na liniji 25 se stvara privremena varijabla *domainRaces* koja je svojim tipom prazna dinamična lista (*engl. Mutable List)*. Nadalje, na linijama 26 - 39, program ulazi u petlju te

iterira kroz elemente dohvaćene liste utrka od udaljenog API-ja. Prilikom svake iteracije petlje, u varijablu *domainRaces* se dodaje novi objekt utrke *Race* te se njemu pridružuju samo odabrani atributi koji će kasnije biti korišteni u aplikaciji. Na ovaj način se izostavljaju nepotrebni podaci koji su dohvaćeni API-jem.

Konačno, na liniji 40, funkcija vraća novi objekt *Calendar* kojemu se, prilikom stvaranja, predaje parametar popis utrka *races.* Na to mjesto predaje se spomenuta privremena varijabla *domainRaces* koja sada, nakon izlaska iz petlje, sadrži popis svih utrka bez nepotrebnih parametara za daljnje korištenje unutar aplikacije.

```
24  fun mapCalendar(response: Response<ApiCalendarResponse>): Calendar {
25      val domainRaces = mutableListOf<Race>()
26      for (race in response.body()!!.MRData.RaceTable.Races)
27          domainRaces.add(
28              Race(
29                  round = race.round,
30                  raceName = race.raceName,
31                  circuit = mapCircuit(race.Circuit),
32                  country = race.Circuit.Location.country,
33                  dateTime = parseDateTime(race.date, race.time),
34                  firstPractice = mapFirstPractice(race.FirstPractice),
35                  secondPractice = mapSecondPractice(race.SecondPractice),
36                  thirdPracticeOrSprint = mapThirdPracticeOrSprint(race),
37                  qualifying = mapQualifying(race.Qualifying)
38              )
39          )
40      return Calendar(races = domainRaces)
41  }
```

Slika 3.5. *Prikaz izvornog koda objektniog mapera za utrke*

Aplikacija bi radila jednako i bez korištenja objektnih mapera, ali njihova vrijednost je vidljiva ponajviše programerima, tj. maperi povećavaju čitkost i preglednost kôda te na taj način omogućuje lakše održavanje i proširivanje izvornog kôda.

Objektni maperi se mogu koristiti i ulančano. Potreba za ulančavanjem objektnih mapera proizlazi iz kompleksne prirode izvornih podataka koji imaju takvu, složenu, strukturu koja je netrivijalna za preslikavanje iz jednog objekta u drugi te je potrebno raditi određene transformacije podataka.

3.1.5. Obavijesti

Jedna od glavnih značajki ove aplikacije jesu obavijesti (*engl. notifications).* Jedna od stvari koju nudi ova aplikacija jest kalendarski prikaz nadolazećih utrka Formule 1. Korisnicima je

omogućeno postavljanje podsjetnika na neki od događaja iz kalendarskog prikaza te se taj podsjetnik manifestira u obliku obavijesti na mobilnom uređaju.

Korisnik je u mogućnosti birati između nekoliko uobičajenih vremenskih intervala za pristizanje obavijesti o željenom događaju. Ti vremenski intervali su sljedeći:

- Točno vrijeme događanja (bez vremenske odgode)
- 30 minuta prije događanja
- Jedan dan prije događanja
- Proizvoljan datum i vrijeme

Osim predviđenih vremenskih intervala odgode obavijesti, korisnik može proizvoljno odabrati kada želi biti obavješten o odabranom događaju. Korisnikovi podsjetnici se pohranjuju unutar *SharedPreferences* objekta putem čije instance se kroz aplikaciju dohvaća popis postavljenih podsjetnika koji su potrebni za prikaz na ekranu [12].

Na slici 3.6. Prikazan je izvorni kôd za provjeru prethodno postavljenih obavijesti za pojedini događaj u jednom trkaćem vikendu koji su istekli te više nisu aktualni. Na samome početku, na liniji 253, dohvaća se instanca *SharedPreferences* objekta u kojemu se pohranjuju svi podsjetnici koji su uključeni od strane korisnika. Na liniji 255 dohvaća se točno, trenutno, korisnikovo vrijeme koje se dalje koristiti unutar metode. Nakon toga, na liniji 257 – 268, program ulazi u petlju kojom iterira kroz dohvaćeni popis podsjetnika iz *sharedPref* varijable. Na liniji 260 se dohvaća ciljano vrijeme za aktivaciju podsjetnika i pohranjuje u varijablu *reminderDate*, te se na liniji 261 uspoređuje trenutno vrijeme korisnika, *currentDate* i *reminderDate*. Nakon te provjere, ukoliko se utvrdi da je ciljano vrijeme manje nego trenutno korisnikovo vrijeme, može se zaključiti da je taj podsjetnik više nije aktualan te ga više nije potrebno prikazivati korisniku na ekranu. Na linijama 262 – 265 obavlja se brisanje trenutnog podsjetnika.

Slika 3.6. *Prikaz izvornog koda za ažuriranje obavijesti*

3.1.6. Ubrizgavanje ovisnosti

Aplikacija je razvijena uz pomoć Hilt biblioteke. Hilt je Googleova biblioteka koja olakšava implementaciju za ubrizgavanje ovisnosti (*engl. Dependency Injection*). Ubrizgavanje ovisnosti je obrazac dizajna koji se koristi za upravljanje ovisnostima između različitih komponenata u aplikaciji, kao što su aktivnosti, fragmenti, servisi i druge klase. Hilt čini ubrizgavanje ovisnosti jednostavnim i učinkovitim za Android razvoj [13].

Hilt koristi anotacije za označavanje klasa koje trebaju biti dostupne za ubrizgavanje ovisnosti. Oznake olakšavaju deklariranje i upravljanje ovisnostima. Također, Hilt generira dio kôda koji obavlja ubrizgavanje ovisnosti. To uključuje stvaranje instanci, povezivanje komponenti i upravljanje životnim ciklusima objekata.

Na slici 3.7. prikazana je implementacija Hilt biblioteke unutar *ViewModela* vozača. Na samome početku, na liniji 18, nalazi se oznaka *@HiltViewModel* koja klasu *DriverViewModel* označava kao dostupnu generatoru kôda za ubrizgavanje ovisnosti. Osim toga, prilikom pisanja konstruktora potrebno je staviti oznaku *@inject*, a to je vidljivo na liniji 19. Tom oznakom je omogućeno ubrizgavanje drugih klasa i servisa kroz konstruktor klase.

Slika 3.7. *Prikaz označavanja Hilt ViewModela*

Ova aplikacija koristi obrazac repozitorija te je potrebno ubrizgati *F1ApiRepository* u *ViewModelu* vozača. To se postiže kroz konstruktor kao što je objašnjeno na slici 3.7.. Kako bi Hilt biblioteka mogla generirati potreban kôd za ubrizgavanje ovisnosti, potrebno je označiti s oznakom *@Module* objekt repozitorija koji je ubrizgan prilikom kreiranja ViewModela.

Na slici 3.8. prikazano je označavanje repozitorija za automatsko generiranje kôda Hilt biblioteke. Na liniji 11 nalazi se spomenuta, potrebna anotacija *@Module*. Osim toga, još je potrebno naznačiti što je ubrizgano prilikom poziva ovog repozitorija. To se postiglo s anotacijom *@Provides* na liniji 15.

```
@Module
@InstallIn(SingletonComponent::class)
object RepositoryModule {

    @Provides
    fun provideRepository(api: F1DataApi): F1ApiRepository{
        return F1ApiRepositoryImpl(api = api)
    }

}
```

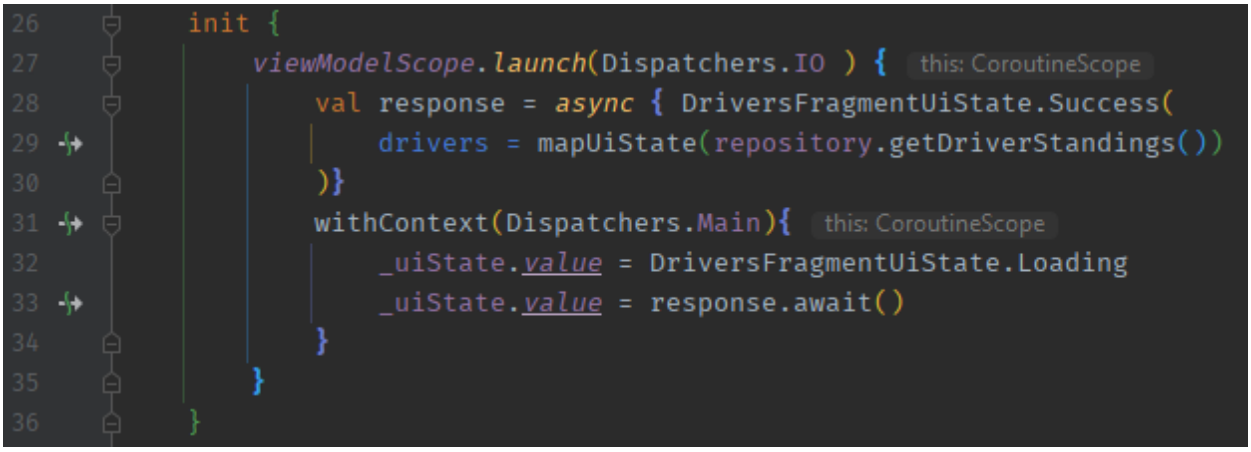
Slika 3.8. *Prikaz Hilt modula*

Nakon postavljanja potrebnih oznaka, Hiltov generator kôda ima sve potrebno za ubrizgavanje kôda prilikom izvođenja. Na taj način se ubrizgavaju svi potrebni servisi i klase unutar aplikacije.

3.1.7. Izvedba i optimiziranje

Kako je do sada pokazano, ova aplikacija sadrži resursno zahtjevne pozive funkcija kao što je dohvaćanje podataka putem HTTP protokola, mapiranje objekata iz jednog u drugi i slično. Kako bi aplikacija radila bez poteškoća i bez blokiranja korisničkog sučelja, koristi se višenitnost za takve, zahtjevne, operacije.

U Kotlinu postoje tzv. korutine (*engl. coroutines)* koje omogućuju asinkrono programiranje [14]. Te je na slici 3.9. prikazano pokretanje korutine. Na liniji 27. pokreće se korutina na *IO* dretvi koja dohvaća popis vozača (linija 28), te na liniji 29 zove objektni maper kojim se izostavljaju nepotrebni podaci dohvaćeni od strane API poslužitelja. Osim toga, na liniji 31, pokreće se korutina, na glavnoj dretvi *Main.* Na *Main* dretvi se odvijaju sve operacije koje su vezan za korsničko sučelje te se u toj korutini, na liniji 32, prikazuje standardna Android animacija za učitavanje podataka (*engl. Loading)* dok *IO* korutina završi sa svojim poslom.



Slika 3.8. *Prikaz izvornog koda za pokretanje korutina*

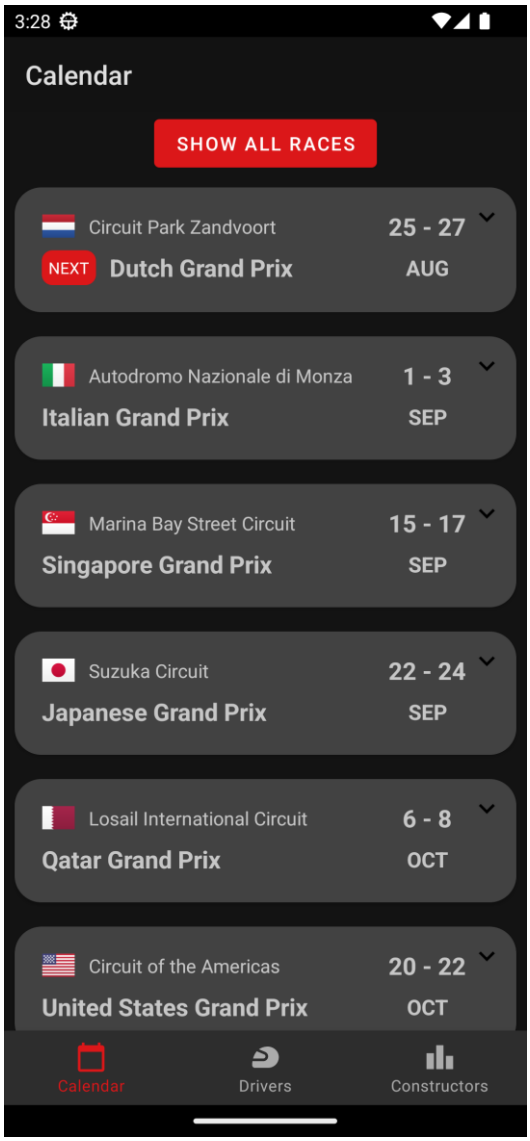
4. PRIKAZ RADA APLIKACIJE

U ovome poglavlju detaljno su prikazani svi ekrani u sklopu aplikacije i objašnjena njihova uloga. Uz to, napisana je kratka uputa za korištenje prikazanih ekrana te su dani opisi pojedinih elemenata aplikacije.

4.1. Kalendar utrka

Prilikom pokretanja aplikacije, prikazuje se početni ekran na kojemu je središnji sadržaj kalendarski prikaz utrka trenutno aktivne sezone Formule 1. Na dnu ekrana nalazi se navigacijski izbornik putem kojeg se prelazi s jednog ekrana na ostale ponuđene destinacije unutar aplikacije.

Na slici 4.1. prikazan je početni zaslon aplikacije prilikom prvog pokretanja.

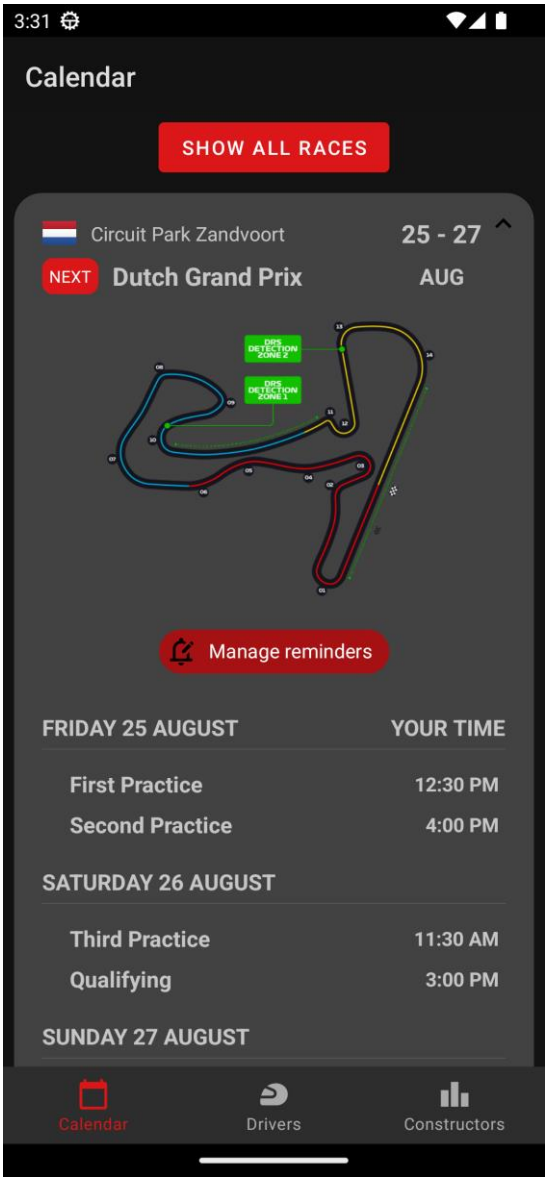


Slika 4.1. *Prikaz početnog ekrana aplikacije*

Popis utrka koji se prikazuje mijenja se dinamički, ovisno o trenutnom datumu kada se aplikacija pokreće. To znači da se utrke koje su završile, unutar trenutno aktivne sezone, ne prikazuju prilikom pokretanja aplikacije. One se mogu prikazati pritiskom na crvenu kontrolu *Show all races* na vrhu ekrana.

Svaka prikazan utrka na popisu je interaktivni element na ekranu koji u sebi sadrži popratne detalje pojedine utrke koji se mogu prikazati i ponovno sakriti pritiskom na jednu od utrka s popisa.

Na slici 4.2. prikazan je detaljan opis Nizozemske utrke Formule 1 gdje se mogu vidjeti podaci poput datumskih i vremenskih termina početka pojedinog događaja tijekom trkaćeg vikenda, izgled staze i najvažnije, crvena kontrola za postavljanje podsjetnika s naslovom *Manage reminders*.



Slika 4.2. *Prikaz ekrana s detaljima utrke*

4.1.1. Podsjetnici

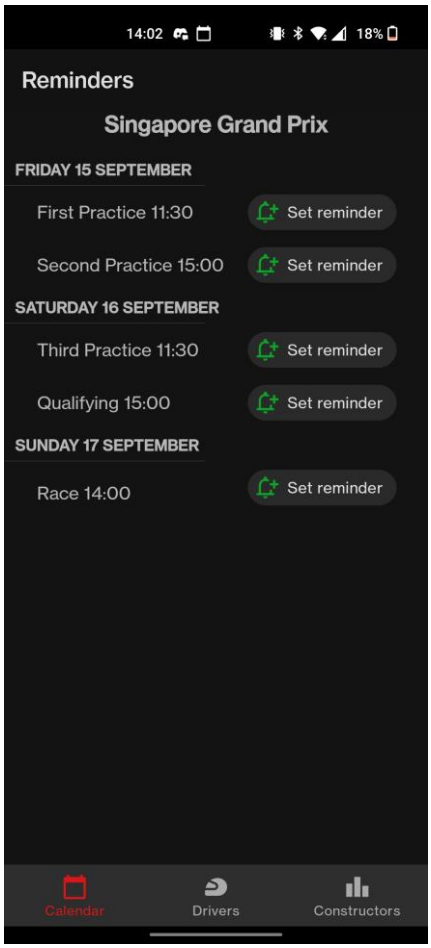
U prethodnome poglavlju, točnije na slici 4.2. prikazana je kotnrola za podsjetnike. Ta kontrola korisnike odvodi na poseban ekran gdje je moguće postaviti podsjetnike za početak pojedinog događaja unutar jednog vikenda.

Postoji pet događaja koji su ključni za gledatelje i protežu se kroz tri dana, a to su:

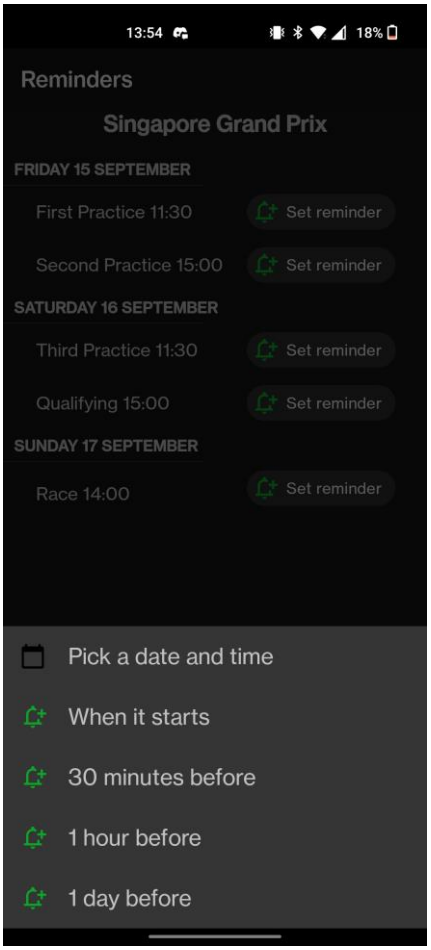
- Prvi i drugi slobodni trening (petak)
- Treći slobodni trening i kvalifikacije (subota)
- Utrka (nedjelja)

Na slikama 4.3. i 4.4. prikazano je sučelje za postavljanje podsjetnika. Na slici 4.3. korisnik ima mogućnost postavljanja podsjetnika za sve ponuđene događaje unutar jednog trkaćeg vikenda. Pritiskom na jednu od pet kontrola *Set reminder* prikaže se izbornik (slika 4.4.) putem kojeg se postavlja željeno vrijeme podsjetnika.

Na slici 4.4. prikazan je spomenuti izbornik gdje se nalaze unaprijed ponuđene vremenske odgode uz posebnu kontrolu *Pick a date and time* kojom se postavlja proizvoljno vrijeme kada aplikacija treba obavijestiti korisnika.



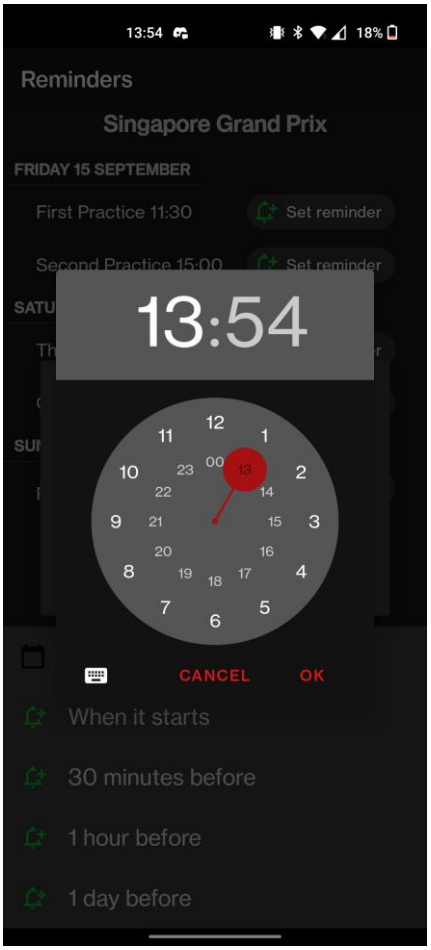
Slika 4.3. *Prikaz svih podsjetnika*



Slika 4.4. *Prikaz izbornika za podsjetnike*

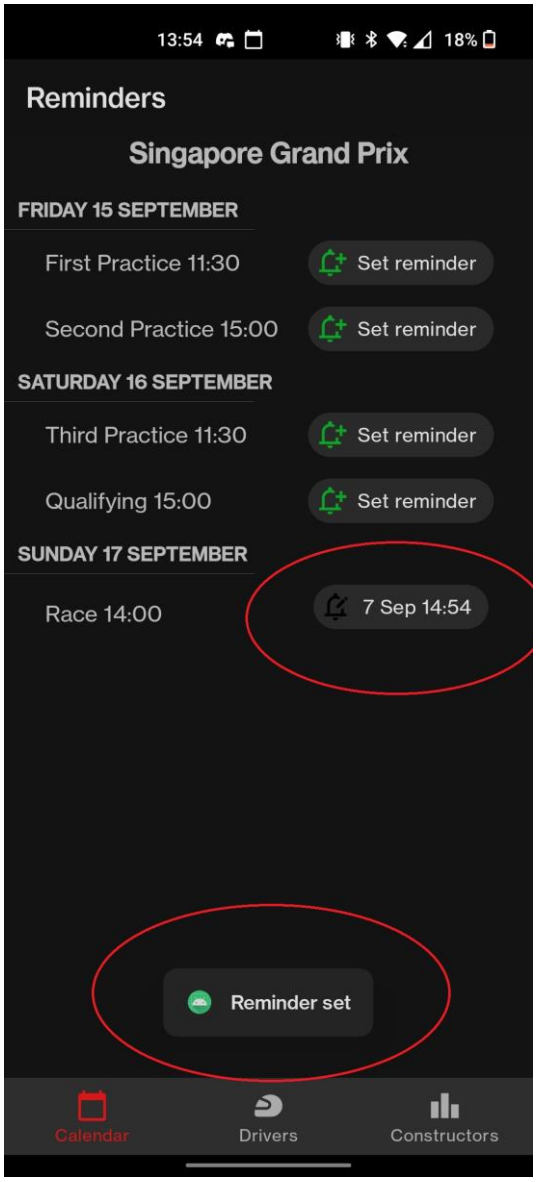
Osim predodređenih vremenskih intervala za podsjetnike, korisnik također može proizvoljno postaviti podsjetnike te se na slikama 4.5. i 4.6. može vidjeti kako izgleda sučelje u tome slučaju. Prvo se od korisnika traži unos datuma (slika 4.5.), a nakon toga unos vremena (slika 4.6.). Aplikacija korisnicima ne dopušta unošenje nedozvoljenog datuma i vremena. Što znači da se ne dopušta unos datuma i vremena ranijeg od trenutka unosa i kasnijeg od termina događaja za kojeg se podsjetnik postavlja.

Slika 4.5. *Prikaz ekrana za unos datuma*



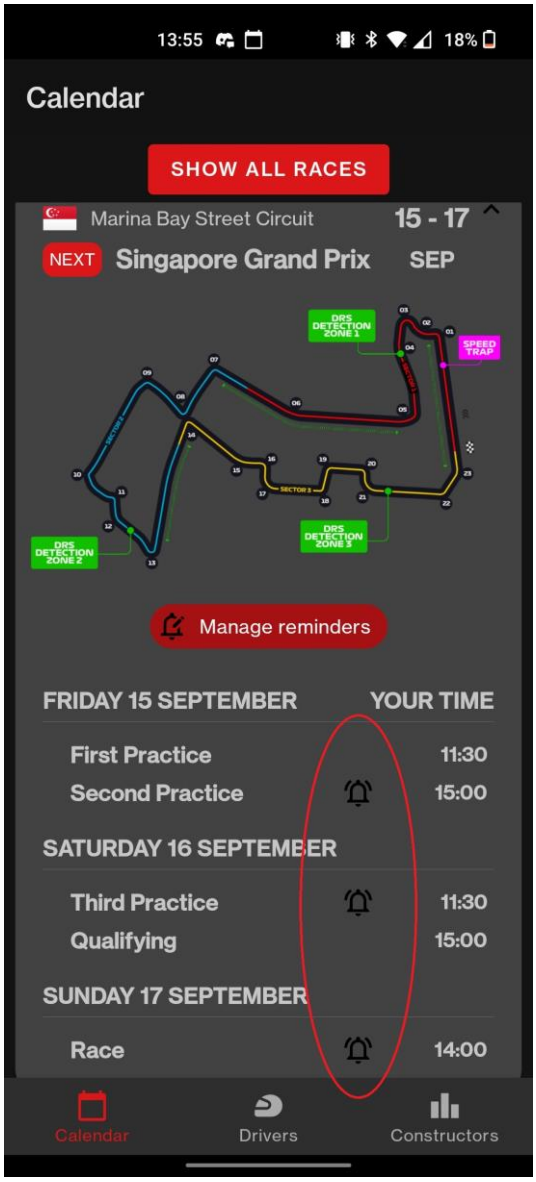
Slika 4.6. *Prikaz ekrana za unos vremena*

Nakon uspješnog odabira datuma i vremena, aplikacija korisnika vraća na prethodni ekran te se prikazuje kratka obavijest o uspješnom rezultatu. Na slici 4.7. prikazano je uspješno postavljanje obavijesti o početku utrke uz kratku obavijest na dnu ekrana s tekstom *Reminder set.* Osim kratke obavijesti, postavljeni podsjetnici mijenjaju ikonu na kontroli za postavljanje te sada umjesto *Set reminder* na toj kontroli piše datum i vrijeme kada će taj podsjetnik biti aktiviran.



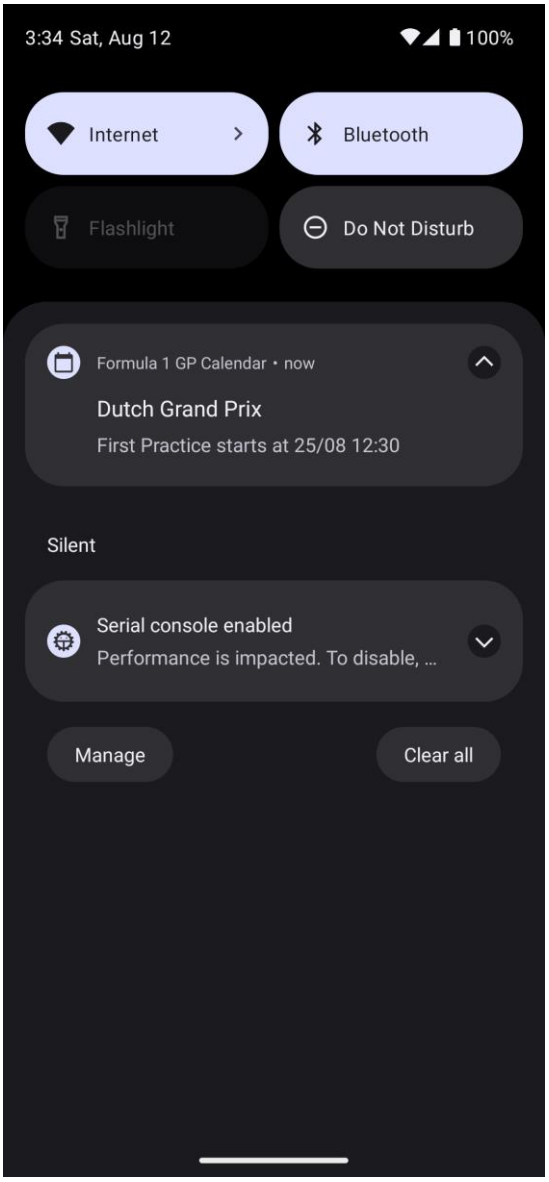
Slika 4.7. *Prikaz postavljenog podsjetnika*

Na početnom ekranu unutar prikaza detalja utrke, nakon uspješnog postavljanja obavijesti vidljiva je Android ikona za obavijesti. Na slici 4.8. prikazano je kako izgleda nekoliko aktivnih podsjetnika unutar jednog trkaćeg vikenda.



Slika 4.8. *Prikaz više postavljenih podsjetnika*

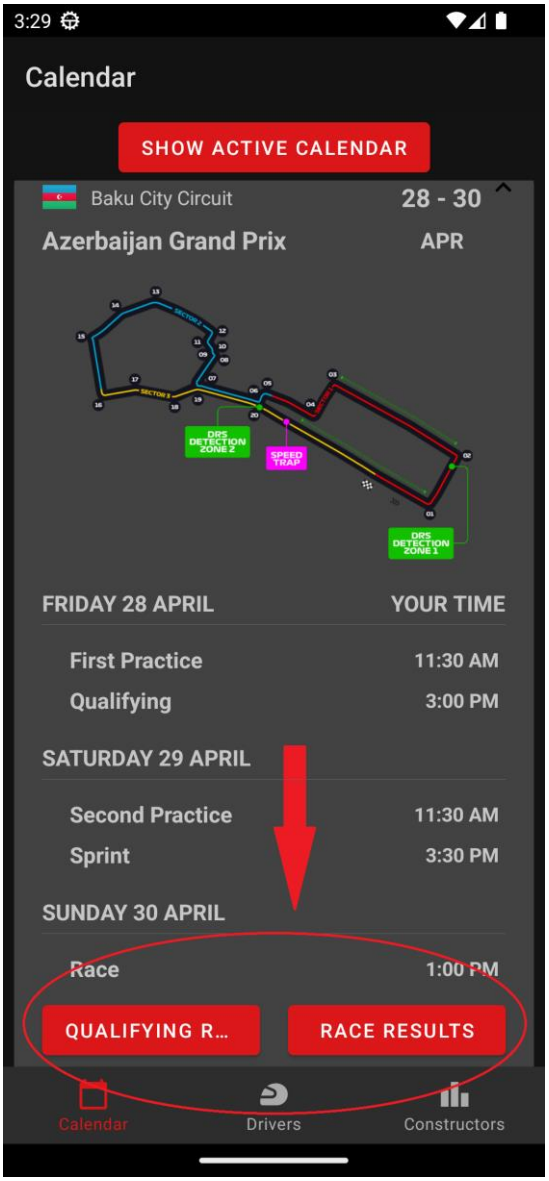
Kada dođe ciljano vrijeme aplikacije za prikaz podsjetnika koji je korisnik postavio, na Android operacijskom sustavu se korisniku šalje podsjetnik u obliku Android notifikacije koji nije ovisan o radu aplikacije. Notifikacija se korisniku šalje na posebnoj sustavskoj dretvi koja ne ovisi o radu aplikacije. Na slici 4.9. može se vidjeti podsjetnik koji se prikazao korisniku u traženo vrijeme za početak prvog slobodnog treninga.



Slika 4.9. *Prikaz aplikacijskog podsjetnika u ciljano vrijeme*

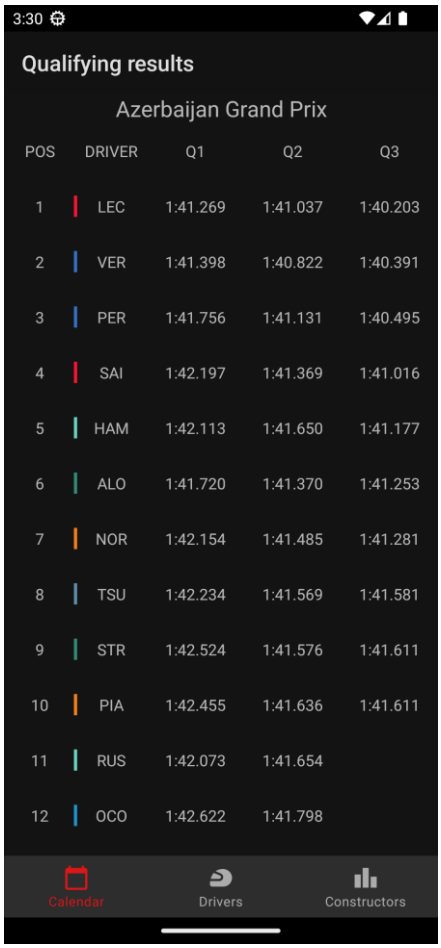
4.1.2. Rezultati utrke

Osim detalja o utrci, moguće je dohvatiti i prikazati podatke o prošlim utrkama. Aplikacija podržava dohvaćanje i prikazivanje rezultata kvalifikacija za utrku i krajnjih rezultata pojedine utrke. Na slici 4.10. Prikazane su kontrole za prikaz rezultata kvalifikacija (lijevo) i konačnih rezultata utrke (desno).

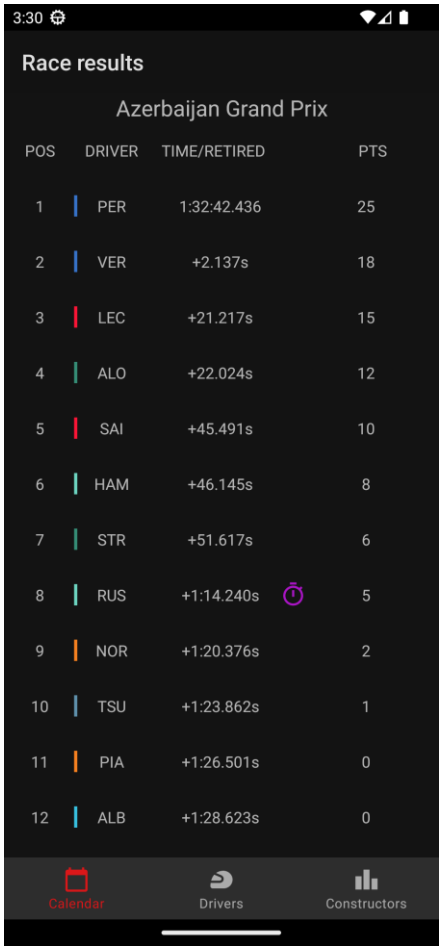


Slika 4.10. *Prikaz kontrola za prikaz rezultata utrke*

Imena vozača su prikazana sa službenom skraćenicom od tri slova koju propisuje FIA. Također, ukoliko je netko od vozača odustao tijekom utrke, prikazan je razlog odustajanja, npr. *kvar na automobilu*. Na slikama 4.11. i 4.12. mogu se vidjeti prikazi rezultata kvalifikacija i konačnih rezultata utrke u Azerbajdžanu.



Slika 4.11. *Prikaz rezultata kvalifikacija*

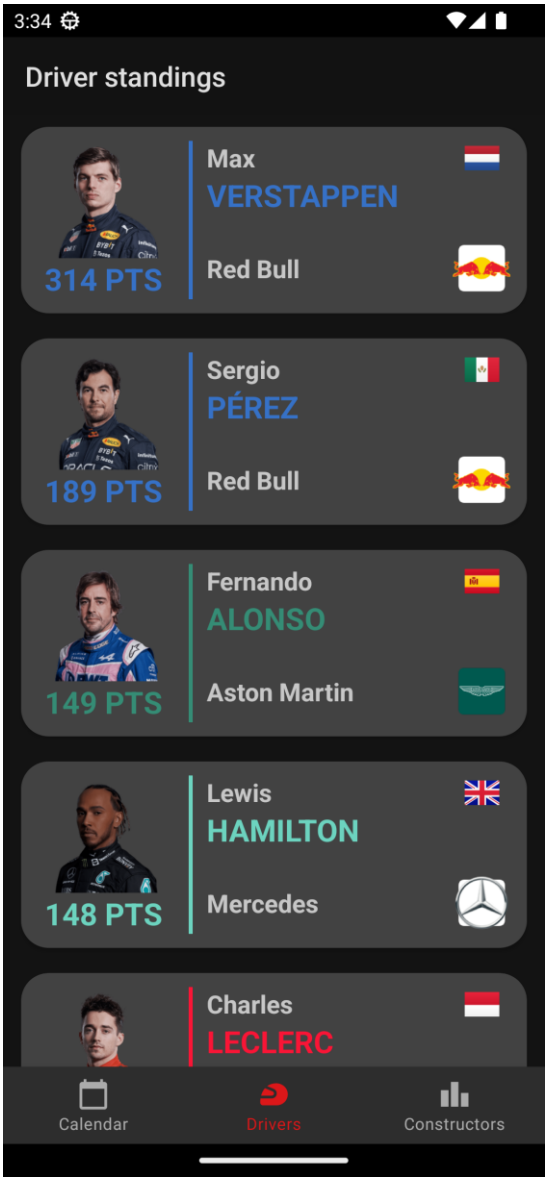


Slika 4.12. *Prikaz konačnih rezultata utrke*

4.2. Vozački poredak

Druga po redu od tri destinacije na navigacijskom izborniku jest trenutni poredak vozača prema osvojenim bodovima. Na ovome ekranu se prikazuje bodovno sortirani popis vozača (od većeg prema manjem) koji sudjeluju u trenutnoj sezoni Formule 1. Osim broja ostvarenih bodova prikazan je portret vozača, nacionalnost i logotip momčadi za koju voze.

Na slici 4.13. prikazan je ekran s vozačkim poretkom.

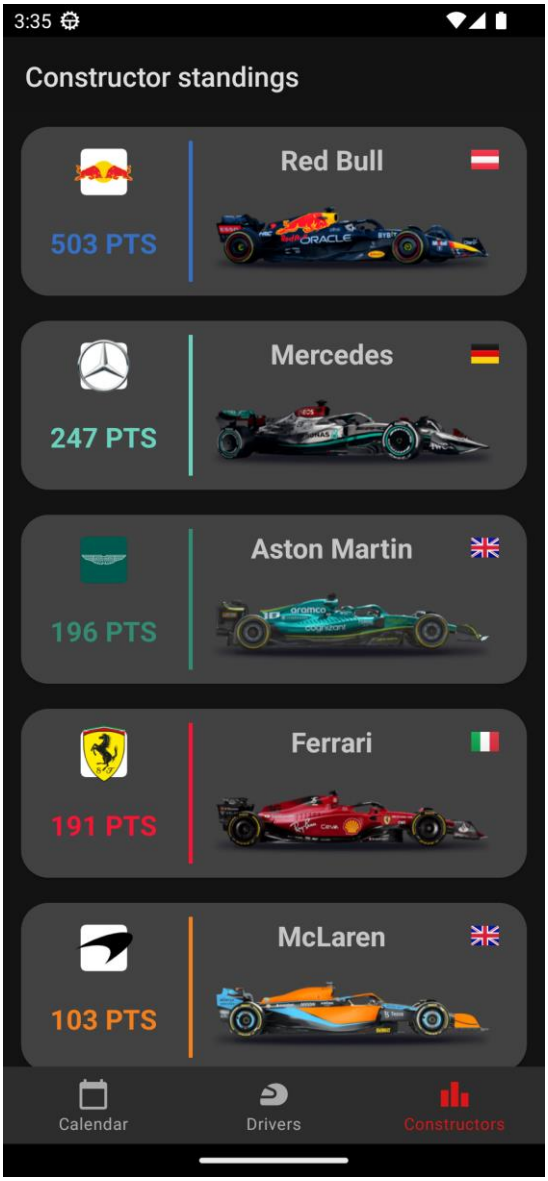


Slika 4.13. *Prikaz vozačkog poretka*

4.3. Ekipni poredak

Na posljednjoj destinaciji na navigacijskom izborniku nalazi se prikaz ekipnog poretka u trenutnoj trkaćoj sezoni. Unutar ekipe nalaze se dva vozača koji zbrojem svojih bodova čine ukupno ekipno bodovanje. Na ekranu se osim broja ekipnih bodova može vidjeti službeni model trkaćeg auta, logotip ekipe i država otkud ekipa potječe.

Na slici 4.14. prikazan je ekran ekipnog poretka.



Slika 4.14. *Prikaz ekipnog poretka*

5. ZAKLJUČAK

Android aplikacija za praćenje rezultata Formule 1 je aplikacija koja obožavateljima Formule 1 omogućuje praćenje rezultata i aktivnosti na moderan način. Aplikacija je razvijena korištenjem razvojnog okruženja Android Studio te je sva logika unutar aplikacije pisana Kotlin programskim jezikom. Za korištenje ove aplikacije nije potrebno kreiranje korisničkog računa, već je potrebno aplikaciju instalirati na svoj uređaj i početi koristiti.

Aplikacija uzima u obzir vremenske zone te korisnicima pruža prikaz svih prijašnjih i nadolazećih utrka u lokalnom vremenu. Osim samih termina i detalja o utrkama, dostupni su rezultati prošlih utrka te pogled na bodovni poredak vozača i momčadi posebno.

Kako korisnici ne bi zaboravili i propustili početak neke od utrka, aplikacija ima mogućnost postavljanja obavijesti o početku događanja. Korisnici mogu postaviti neke od prije definiranih termina za podsjetnik ili postaviti proizvoljnu obavijest kada žele biti informirani o događanju.

Ograničenje ove aplikacije se ponajviše manifestira u prikazu jedne, trenutno aktivne, sezone u Formuli 1. Ovisno o datumu korištenja, aplikacija će od API-ja zatražiti najsvježije podatke i prikazati ih korisnicima na isti način, neovisno o godini. Osim toga, pristupom projektu sa gledišta čistog kôda i lake proširivosti, ostavljeno je prostora za daljnje nadogradnje na trenutno funkcioniranje aplikacije.

LITERATURA

[1] Formula 1, https://corp.formula1.com/about-f1/, (datum zadnje posjete: 12. kolovoza 2023.)

[2] F1 Race Guide, https://play.google.com/store/apps/details?id=com.formula1.event, (datum zadnje posjete: 12. kolovoza 2023.)

[3] Racing Calendar 2023, https://play.google.com/store/apps/details?id=nl.deepapp.RaceCalendar, (datum zadnje posjete: 12. kolovoza 2023.)

[4] Formula 2023 CalendarStandings, https://play.google.com/store/apps/details?id=xyz.romanello.formulacalendar, (datum zadnje posjete: 12. kolovoza 2023.)

[5] FORONE: F1 Calendar 2023, https://play.google.com/store/apps/details?id=com.aspargus.formulaguan, (datum zadnje posjete: 12. kolovoza 2023.)

[6] Formula 2023 Calendar, https://play.google.com/store/apps/details?id=com.android.apps.f1calendarandremainder, (datum zadnje posjete: 12. kolovoza 2023.)

[7] Kotlin, https://kotlinlang.org/, (datum zadnje posjete: 12. kolovoza 2023.)

[8] Android Studio, https://developer.android.com/about, (datum zadnje posjete: 12. kolovoza 2023.)

[9] RecyclerView, https://developer.android.com/develop/ui/views/layout/recyclerview, (datum zadnje posjete: 12. kolovoza 2023.)

[10] Ergast API, http://ergast.com/mrd/, (datum zadnje posjete: 12. kolovoza 2023.)

[11] Retrofit, https://square.github.io/retrofit/, (datum zadnje posjete: 12. kolovoza 2023.)

[12] SharedPreferences, https://developer.android.com/reference/android/content/SharedPreferences, (datum zadnje posjete: 12. kolovoza 2023.)

[13] Hilt, https://developer.android.com/training/dependency-injection/hilt-android#kts, (datum zadnje posjete: 12. kolovoza 2023.)

[14] Coroutines, https://kotlinlang.org/docs/coroutines-overview.html, (datum zadnje posjete: 12. kolovoza 2023.)

SAŽETAK

Android aplikacija za praćenje rezultata Formule 1 namijenjena je svim obožavateljima ovog automobilskog sporta. Svojim korisnicima na prvi pogled daje sažeti prikaz cjelokupne aktualne trkaće sezone, no može prikazati detaljne informacije o pojedinoj utrci. Osim toga, korisnici imaju mogućnost postavljanja obavijesti o početku svih događanja u Formuli 1. Nadalje, sve informacije o rezultatima prošlih utrka su sadržane unutar aplikacije kao i ukupni bodovni poredak vozača i ekipa.

KLJUČNE RIJEČI: Android, Formula 1, Kotlin, mobilna aplikacija, MVVM arhitektura

ABSTRACT

Android application for tracking Formula 1 results

Android application for tracking Formula 1 results is designed for all kinds of fans of this motorsport. At first glance, it provides its users with a concise overview of the entire current racing season, but it can also display detailed information about individual races. Furthermore, users have the option to set up notifications for the start of all Formula 1 events. Moreover, all information about the results of past races is contained within the application, as well as the overall driver and team standings.

KEYWORDS: Android, Formula 1, Kotlin, mobile application, MVVM architecture

ŽIVOTOPIS

Mislav Kostić rođen je 1. travnja 1997. godine u Požegi. Pohađao je osnovnu školu Dobriše Cesarića u Požegi. Nakon završenog osnovnoškolskog obrazovanja, upisao je klasičnu Katoličku gimnaziju u Požegi gdje je maturirao 2016. godine. Iste godine upisao je Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. 2021. godine sječe naziv Sveučilišni prvostupnik (*baccalaureus*) inženjer računarstva. Nastavlja obrazovanje upisom diplomskog studija računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. 2022.. godine obavlja stručnu praksu u tvrtki COBE, Hrvatske Republike 33, Osijek, gdje trenutno radi studentski posao izrade mobilnih aplikacija u Flutter razvojnom okviru.

_____

Mislav Kostić

6. PRILOZI

Prilog 1. CD/DVD disk s izvornim kodom i diplomskim radom u .docx i .pdf formatu