

# Oblikovanje programske potpore

Ak. god. 2019./2020.

## Zabava.NET

Dokumentacija, Rev. 2

Grupa: *KrimTim3*

Voditelj: *Ivona Martinović*

Datum predaje: *16. 01. 2020.*

Nastavnik: *Nikolina Frid*

# Sadržaj

<b>1 Dnevnik promjena dokumentacije</b>	<b>3</b>
<b>2 Opis projektnog zadatka</b>	<b>9</b>
<b>3 Specifikacija programske potpore</b>	<b>14</b>
3.1 Funkcionalni zahtjevi . . . . .	14
3.1.1 Obrasci uporabe . . . . .	16
3.1.2 Sekvencijski dijagrami . . . . .	29
3.2 Ostali zahtjevi . . . . .	33
<b>4 Arhitektura i dizajn sustava</b>	<b>34</b>
4.1 Baza podataka . . . . .	37
4.1.1 Opis tablica . . . . .	38
4.1.2 Dijagram baze podataka . . . . .	41
4.2 Dijagram razreda . . . . .	42
4.3 Dijagram stanja . . . . .	56
4.4 Dijagram aktivnosti . . . . .	57
4.5 Dijagram komponenti . . . . .	58
<b>5 Implementacija i korisničko sučelje</b>	<b>59</b>
5.1 Korištene tehnologije i alati . . . . .	59
5.2 Ispitivanje programskog rješenja . . . . .	60
5.2.1 Ispitivanje komponenti . . . . .	60
5.2.2 Ispitivanje sustava . . . . .	64
5.3 Dijagram razmještaja . . . . .	71
5.4 Upute za puštanje u pogon . . . . .	72
5.4.1 Kreiranje baze podataka . . . . .	72
5.4.2 Spajanje s bazom podataka . . . . .	72
5.4.3 Punjenje baze podataka . . . . .	73
5.4.4 Pokretanje Android aplikacije . . . . .	73

<b>6 Zaključak i budući rad</b>	<b>74</b>
<b>Popis literature</b>	<b>76</b>
<b>Dodatak: Prikaz aktivnosti grupe</b>	<b>77</b>
<b>Dodatak 2: Kodovi ispitivanja sustava</b>	<b>83</b>
<b>Indeks slika i dijagrama</b>	<b>104</b>

# 1. Dnevnik promjena dokumentacije

Tablica 1.1: Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak	Martinović	04.10.2019.
0.2	Opis projektnog zadatka dodan u dokumentaciju	Ivandić	12.10.2019.
0.3	Uključena specifikacija	Popović	15.10.2019.
0.4	Dodani <i>Use Case</i> dijagrami	Samodol	22.10.2019.
0.5	Dodani obrasci uporabe u dokumentaciju Ispravljene pogreške u obrascima uporabe	Harbaš	22.10.2019.
0.6	Dodani dijagrami sa generalizacijama za korisnika i organizatora	Samodol	23.10.2019.
0.7	Dodan dionik Google i aktor neverificirani korisnik Dodan dodatak - dokumentacija sastanaka	Popović	24.10.2019.
0.8	Preimenovan klijent u posjetitelj i dodan Google kao aktor	Samodol	25.10.2019.
0.8	Dodan sekvencijski dijagram za UC-10-13	Ivandić	05.11.2019.
0.9	Unesen prethodni sastanak Aktor 'Korisnik' preimenovan u 'Posjetitelj' Dodani 'Ostali zahtjevi'	Popović	05.11.2019.
0.10	Dodani sekvencijski dijagrami za UC-3 i UC-6 Izbrisani duplicirani dijagrami te ispravljene pogreške	Samodol	06.11.2019.
0.11	Ispravci u UC-10-13	Ivandić	06.11.2019.
0.12	Manje izmjene u sekvencijskom dijagramu UC-3	Gorup	07.11.2019.

Rev.	Opis promjene/dodatka	Autori	Datum
0.13	Dodana zadnja dva sastanka Dodan aktor Google i opisi sekvencijskih dijagrama	Popović	07.11.2019.
0.14	Promjene u obrascima uporabe, naziv i brisanje duplikata Promjena redoslijeda obrazaca uporabe	Ivandić	11.11.2019.
0.15	Izmjene u opisima sekvencijskih dijagrama	Popović	11.11.2019.
0.16	Dodana generalizacija u obrascu uporabe za posjetitelja Uređeni obrasci uporabe Preimenovani dijagrami u smislenije nazive, popravljen redoslijed	Martinović	11.11.2019.
0.17	Dodan dijagram za UC-10-13 Dodana slika za UC-10-13	Harbaš	12.11.2019.
0.18	Izmjenjen opis sekvencijskog dijagrama UC-10-13 Manje izmjene u opisima obrazaca uporabe i funkcionalnostima	Popović	12.11.2019.
0.19	Dodan opis dijagrama razreda Dodana slika dijagrama razreda Dodane slike dijagrama obrazaca uporabe i sekvencijskih dijagrama Promijenjena slika dijagrama razreda iz 'Promoter' u 'Organizer'	Gorup	12.11.2019.
0.20	Dodan opis arhitekture sustava i opis baze podataka Dodan datum predaje	Martinović	13.11.2019.
0.21	Manje izmjene Promijenjen opis sekvencijskog dijagrama UC-17 Dodan opis novog sastanka Promjena opisa sekvencijskog dijagrama UC-10-13	Popović	14.11.2019.
0.22	Ispravljen UC-17 Izvezena slika za UC-17	Ivandić	14.11.2019.

Rev.	Opis promjene/dodatka	Autori	Datum
0.23	Izmjene u sekvencijskim dijagramima Dodana ilustracija 'viewModel' životnog ciklusa	Harbaš	14.11.2019.
0.24	Dodan dijagram baze	Gorup	14.11.2019.
0.25	Dodana slika 'ViewModelScope'	Harbaš	15.11.2019.
0.26	Dodana slika za MVVM Manje ispravke u gramatici	Ivandić	15.11.2019.
0.27	Dodani ostali dijagrami razreda Dodani opisi dijagrama razreda Ispravak opisa projektnog zadatka	Pranklin	15.11.2019.
0.28	Izmijenjen dijagram baze	Gorup	15.11.2019.
0.29	Dodan dnevnik promjena dokumentacije	Martinović Samodol	15.11.2019.
0.30	Manje ispravke u gramatici	Popović	15.11.2019.
1.0	Finalna verzija za prvu predaju	Martinović	15.11.2019.
1.1	Zaključak i budući rad	Ivandić	24.12.2019.
1.2	Promjena iz attending u going	Popović	26.12.2019.
1.3	Dodan dijagram aktivnosti i dijagram stanja	Samodol	28.12.2019.
1.4	Dodan opis novog sastanka	Popović	29.12.2019.
1.5	Dodan include zaključka	Ivandić	29.12.2019.
1.6	Male promjene u rasporedu linija Dodane slike testiranja za pisanje dokumentacije	Gorup	3.1.2020.
1.7	Dodani opisi testova Dodano poglavlje implementacija	Gorup	5.1.2020.
1.8	Sitne izmjene Dodan prostor za indekse tablica	Gorup	7.1.2020.

Rev.	Opis promjene/dodatka	Autori	Datum
1.9	Dodane korištene tehnologije i alati Dodane slike dijagrama Dodane upute za puštanje u pogon Dodana slike za BuildGradleProject i BuildGradleModule Dodana slika primjera punjenja baze Izmjena tablica Dodana tablica organizations Dodan dijagram baze kao trenutno stanje	Gorup	8.1.2020.
1.10	Dodan opis novog sastanka Ažuriranje tablica Promjena tablice aktivnosti	Popović	8.1.2020.
1.11	Sitne izmjene Promjena opisa mogućih opcija za posjetitelje Filipovi ispravci Promjena prikaza Izvezena nova slika za UC-17 Promjena iz attending u going Promjena dijagrama razreda Izvezena slika za dijagram razreda	Ivandić	8.1.2020.
1.12	Dodane slike dijagrama u dokumentaciju	Martinović	8.1.2020.
1.13	Ažuriran dijagram razreda komponenti Ažuriran dijagram razreda modula Ažuriran dijagram razreda Utilities Ažuriran dijagram razreda View	Samodol	8.1.2020.
1.14	Dodane slike dijagrama razreda modula Dodane slike dijagrama razreda Components, Utilities, View	Samodol	10.1.2020.

Rev.	Opis promjene/dodatka	Autori	Datum
1.15	Promjene u specifikacijiPP da bude u skladu s funkcionalnostima aplikacije Obrisan UC za kreiranje korisničkog računa Sitne izmjene Promjene dijagrama da budu u skladu s brojevima UC Promjena naslova i opisa sekvencijskih dijagrama Promjena dijagrama UC-19 Izvezene nove slike uc dijagrama, sekvencijskih dijagrama i dijagrama razreda	Martinović	10.1.2020.
1.16	Dodani indexi i naslovi tablica Promjena naslova tablica	Popović	10.1.2020.
1.17	Ažuriran dijagram razreda ViewModel Izvezena slika za dijagram razreda ViewModel	Samodol	11.1.2020.
1.18	Dodani opisi ostalih funkcionalnosti Firestore koje koristimo Poboljšane tablice podataka i njihovi opisi	Martinović	11.1.2020.
1.19	Promjena naziva Map u String Izmjena dijagrama baze	Gorup	12.1.2020.
1.20	Promjena tablice aktivnosti	Popović	12.1.2020.
1.21	Ažuriran dnevnik promjena	Ivandić	14.1.2020.
1.22	Ažurirana tablica aktivnosti Dodan opis dijagrama stanja i dijagrama aktivnosti	Popović	14.1.2020.
1.23	Poboljšanje dijagrama stanja Promjene u estetici dijagrama razmještaja	Martinović	14.1.2020.
1.24	Dodane slike uspješnosti unit testova Dodan opis ispitivanje komponenti	Gorup	14.1.2020.



Rev.	Opis promjene/dodatka	Autori	Datum
1.25	Dodan novi dijagram stanja Dodana slika dijagrama stanja Izmjene u dijagramu razreda komponenti Izmjene u dijagramu razreda Utilities Izmjene u dijagramu razreda View Izmjene u dijagramu razreda modula Izmjene u dijagramu razreda ViewModel Dodana slika za dijagram razreda Utilities Dodana slika za dijagram razreda View Dodana slika za dijagram razreda modula Dodana slika za dijagram razreda ViewModel	Samodol	15.1.2020.
1.26	Dodan opis dijagrama razmještaja	Popović	15.1.2020.
1.27	Dodan dijagram komponenti Dodana slika dijagrama komponenti Izmijenjen dijagram razreda Izmijenjen dijagram komponenti Izvezena slika dijagrama komponenti Dodan UserTest i njegov opis Dodani kodovi za ispitivanje komponenti i njihov opis Dodani kodovi za ispitivanje sustava Dodan opis dijagrama komponenti	Gorup	16.1.2020.
1.28	Promjena dijagrama razreda Izvezena slika dijagrama razreda Uređivanje teksta Ispravak dijagrama razmještaja Izvezena slika dijagrama razmještaja Ažuriran dnevnik promjena	Ivandić	16.1.2020.
1.29	Upisan zadnji sastanak Promjena opisa dijagrama razmještaja Mala promjena imena	Popović	16.1.2020.
1.30	Završne pripreme	Martinović	16.1.2020.
2.0	Finalna verzija za drugu predaju	Martinović	16.1.2020.

## 2. Opis projektnog zadatka

Cilj ovog projekta je razvoj aplikacije "Zabava.NET". Koristeći svoja znanja iz oblikovanja programske potpore, tim će u obliku mobilne aplikacije stvoriti personaliziranu platformu koja će reklamirati zabavna događanja u gradu. Organizatori će oglašavati događanja poput koncerata i predstava, dok će posjetitelji moći najaviti svoj dolazak, a kasnije i dati recenziju.

Kako bi korištenje aplikacije bilo moguće, korisnik se treba prijaviti unosom e-mail adrese i lozinke. Ukoliko još nema račun na aplikaciji, korisnik se mora registrirati. Za registraciju kao **posjetitelj** potrebni su sljedeći podaci:

- e-mail adresa
- lozinka

Kasnije, nakon verifikacije e-mail adrese, posjetitelj može nadopuniti sljedeće informacije:

- ime
- prezime
- korisničko ime
- datum rođenja

Ako posjetitelj želi postati **organizator**, treba odabrati tu opciju te platiti mjesečnu članarinu putem kartice ili PayPala. Za posjetitelje nema naplate usluge korištenja aplikacije. Organizator treba dodatno nadopuniti svoje informacije:

- adresa organizacije
- poveznica na web ili Facebook stranice

Organizacije imaju javne profile kako bi posjetitelji mogli pregledati događaje koje je pojedina organizacija organizirala u posljednje dvije godine. Vidljivi su im i osnovni podaci organizacije. Za svaki događaj koji organizator oglasi, obavezno je navesti sljedeće podatke:

- naziv

- opis događaja
- lokacija
- vrsta
- vrijeme početka
- vrijeme završetka
- fotografija

Uz navedene podatke prikazuje se i popis svih recenzija za odabrano događanje. Dodatno, kako bi privukli što više posjetitelja, organizatori mogu objaviti fotografije i jedan video na kojima se vidi kako se gosti zabavljaju na njihovim događanjima.

Posjetitelji mogu birati vremensko razdoblje u kojem ih zanimaju događanja u gradu. Na taj način omogućava se brži pristup željenom sadržaju.

Posjetitelju se na početnom zaslonu prikazuje popis aktualnih događanja. Za svako događanje posjetitelj može izraziti interes. Za to mu se nude tri mogućnosti i to su: **"sigurno dolazim"**, **"možda dolazim"** i **"ne dolazim"**. Nakon što se posjetitelj izjasni, može i promijeniti mišljenje.

Na detaljima događaja, javno je vidljiv i broj zainteresiranih posjetitelja. Organizatori mogu vidjeti i popis korisničkih imena i slika profila zainteresiranih posjetitelja. Imajući u vidu te informacije, organizatori mogu pretpostaviti broj gostiju koji će stvarno doći.

Posjetitelji unutar 48 sati od događanja mogu napisati recenziju za događaj kojem su prisustvovali. Na taj način informiraju ostale korisnike o kakvoj se zabavi radi i što mogu očekivati od određenog organizatora. Time poboljšavaju kvalitetu aplikacije i nude bolju uslugu ostalim korisnicima.

Posjetiteljima se nudi mogućnost prikaza događaja prema određenim kriterijima. Također mogu odabrati postavke da im aplikacija šalje obavijesti o novim događajima prema zadanim kriterijima. Za odabir su ponuđena tri kriterija: vrijeme održavanja događaja, vrsta događaja i područje. Tako se korisnika potiče da što više koristi aplikaciju te da mu se ističe sadržaj primjeren njegovim željama.

Osim organizatora i posjetitelja, u sustavu postoje i **administratori sustava**. Oni imaju najveće ovlasti - postavljaju cijenu članstva i upravljaju korisnicima.

- Aplikacija Zabava.NET iznimno je korisna za ljude željne dobre zabave. Svakom posjetitelju nudi personaliziranu ponudu. Bilo da korisnika zanimaju zabave u klubu ili baletni nastupi, aplikacija ima nešto za svakoga. Umjesto dugotrajnog pretraživanja interneta i pokušavanja pronalaska odgovarajućeg tipa zabave, korisniku je sve pruženo na jednom mjestu. U par klikova moguće je pronaći idealnu zabavu. Uz mnogobrojne mogućnosti poput davanja recenzija i objave fotografija, posjetitelji se mogu informirati je li događanje prikladno onome što traže.
- Osim samih posjetitelja, aplikacija uvelike olakšava posao organizatorima različitih događanja. Više ne moraju dijeliti letke ili lijepiti plakate po gradu kako bi reklamirali događaj, već ga mogu objaviti u našoj aplikaciji. Osim uštede vremena i novca, na ovaj način dolaze do više ljudi koji su zainteresirani baš za takav tip zabave.
- Mogućnost recenziranja događaja vrlo je korisna kako za posjetitelje, tako i za organizatore. Posjetiteljima recenzije pružaju bolju sliku o predstojećim događanjima istog organizatora. Mogu doći do savjeta ljudi sličnog ukusa i tako procijeniti žele li ići na određeni događaj ili ne. S druge strane, recenzije potiču organizatore na kvalitetniji rad. Znajući da će svi potencijalni posjetitelji vidjeti recenzije, organizatori će se truditi pružiti što bolju uslugu svojim gostima i time prikupiti dobre recenzije. Tako će privući još puno novih posjetitelja i ostvariti veći profit.
- Rješenje slično aplikaciji Zabava.NET je mogućnost objave događanja na Facebooku. Tamo organizatori također objavljuju zabave koje organiziraju, te stavljaju adresu i fotografije. Ostali korisnici mogu pratiti događanje, izjasniti se dolaze li i ostaviti komentar. Razlika u odnosu na našu aplikaciju je u tome što Facebook svima omogućuje kreiranje događaja te se često događa da je objava izmišljena pa se događanje uopće neće održati. Također na Facebooku nije uvijek moguće ostaviti recenziju tako da korisnik ne zna što može očekivati. U našoj aplikaciji zabave mogu oglašavati isključivo organizatori te su recenzije uvijek omogućene. Za razliku od Facebooka koji je besplatan za sve, Zabava.NET organizatorima naplaćuje mjesečnu članarinu. Iako obje aplikacije imaju dobrih i loših strana, naša aplikacija je specijalizirana

baš za događanja tako da korisniku omogućava direktan pristup traženoj informaciji. Također, omogućava posjetiteljima da odabirom kriterija primaju obavijesti samo za one događaje koji bi im odgovarali po stilu zabave.

- S obzirom na široki spektar ponude, aplikacija bi mogla zainteresirati sve ljude koji koriste mobilne uređaje. Za starije korisnike nude se predstave i klasični koncerti, a za mlađe događanja u različitim klubovima i raznovrsni koncerti modernih glazbenika. Za korisnike posjetitelje iznimno je privlačna i činjenica da je aplikacija besplatna te da je za korištenje potrebna samo registracija koja traje par sekundi. Iako su organizatori obavezni plaćati mjesečnu članarinu, Zabava.NET mogla bi ih privući zbog svoje jednostavnosti korištenja i pristupa velikom broju ljudi koji traže zabavu. S vremenom bi se aplikacija proširila jer bi već nakon nekoliko događanja imala veću ponudu sadržaja korisnicima.
- Rješenje se može prilagoditi za različite namjene. Npr. aplikacija se može prenamijeniti za objave sajмова, tečajeva, prosvjeda ili aukcija. Također, aplikacija bi se mogla prenamijeniti kao reklamna za tvrtke koje iznajmljuju automobile. Umjesto posjetitelja bili bi korisnici koji žele iznajmiti auto, a mogu i postavljati recenzije nakon najma.
- Cilj nam je izgraditi aplikaciju koja će omogućiti jednostavan i brzi pristup traženim podacima. Želimo da bude jednostavna tako da je prikladna za svakog korisnika. Uz mogućnosti koje ćemo ugraditi, ističemo kako je aplikacija uvijek orijentirana prema korisniku te kako pruža sadržaj prilagođen baš njemu. Davanjem informacije ljudima koji traže zabavu, olakšati ćemo snalaženje u bespućima interneta i omogućiti posjetitelju da u par minuta nađe željenu vrstu događanja.
- Aplikaciju Zabava.NET moguće je nadograditi na brojne načine. U budućnosti bi se mogla dodati mogućnost da posjetitelji odaberu oblik javnog profila te da se vidi na kojim događanjima su bili i na koja planiraju ići. Na taj način bi se njihovi prijatelji informirali koja su dobra mjesta za izlazak jer, ako su se svidjela njihovim prijateljima, mogla bi se i njima. Također, mogli bi se ugraditi dodatni filteri za odabir obavijesti. Ako bismo omogućili da korisnici

posjetitelji mogu međusobno postati prijatelji, mogli bismo dodati i kriterij obavijesti da je netko od bliskih prijatelja označio svoju zainteresiranost za događaj. Tako bi se bez pretraživanja lako saznalo za popularna događanja. Aplikacija bi se mogla nadograditi i za organizatore. Mogli bismo razviti mogućnost da se organizatori međusobno prate, tj. da mogu vidjeti što nudi konkurencija te prema tome poboljšati kvalitetu usluge. Na taj način povisio bi se standard svih vrsta zabave. Ulagalo bi se više truda u izgled stranice jer bi organizatori bili svjesni što drugi nude. Događanje s više fotografija i s boljim recenzijama puno će lakše privući posjetitelja.

## 3. Specifikacija programske potpore

### 3.1 Funkcionalni zahtjevi

#### Dionici:

1. Organizator događanja (naručitelj)
2. Zaposlenici na događanju
3. Posjetitelji događanja
4. Administrator
5. Razvojni tim
6. Google

#### Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
  - (a) se registrirati i stvoriti novi korisnički račun za koji su mu potrebni e-mail adresa i lozinka, te čekati e-mail verifikaciju
  - (b) se registrirati se ili prijaviti pomoću Google računa koristeći "Sign in with Google"
  - (c) se prijaviti ako već ima korisnički račun
2. Neverificirani korisnik (inicijator) može:
  - (a) se prijaviti i aktivirati korisnički račun verificiranjem email adrese
3. Posjetitelj (inicijator) može:
  - (a) vidjeti sva prošla i buduća događanja
  - (b) filtrirati događanja prema lokaciji(gradu), vrsti događanja, te vremenskom razdoblju
  - (c) dobiti obavijesti o najnovijim događanjima odabrane vrste, lokacije ili vremenskog razdoblja
  - (d) vidjeti informacije o događanju kao što su naziv, vrsta, lokacija, vrijeme početka, vrijeme završetka, foto/video galerija i broj zainteresiranih

- (e) pregledati recenzije događaja
- (f) označiti i promijeniti hoće li doći na događanje
- (g) pregledati i mijenjati osobne podatke
- (h) pisati recenzije i ocijeniti događanja na kojima je bio u proteklih 48 sati
- (i) odjaviti se

4. Organizator (inicijator) može:

- (a) dodati ili izbrisati vlastito događanje
- (b) dodati, promijeniti i izbrisati informacije o svojim događanjima
- (c) dodati ili izbrisati slike i video isječke događanja
- (d) platiti mjesečnu članarinu

5. Administrator (inicijator) može:

- (a) postaviti i promijeniti iznos mjesečne članarine
- (b) vidjeti popis svih registriranih korisnika i njihove osobne podatke
- (c) brisati korisničke račune i promijeniti im razinu pristupa (posjetitelj, organizator)
- (d) brisati događanja
- (e) brisati recenzije koje nisu u skladu s pravilima aplikacije

6. Baza podataka (sudionik):

- (a) pohranjuje osobne podatke i ovlasti korisnika
- (b) pohranjuje podatke o detaljima događanja i recenzije na njih
- (c) pohranjuje profilne slike korisnika, slike organizacija, fotografije i videa događanja

7. Google (sudionik):

- (a) pruža pristup Google računima i njihovim podacima koji postoje na mobilnom uređaju



### 3.1.1 Obrasci uporabe

#### Opis obrazaca uporabe

##### UC 1 - Registracija

- **Glavni sudionik:** korisnik
- **Cilj:** stvoriti račun
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. korisnik upisuje potrebne podatke
  2. korisnik prima obavijest o uspješnoj registraciji
  3. sustav zadržava korisnika na posebnom zaslonu i čeka potvrdu e-mail adrese
- **Opis mogućih odstupanja:**
  - 2.a odabir zauzetog maila, unos lozinke ili maila neispravnog formata
    1. sustav obavještava korisnika o unosu neispravnih podataka
    2. korisnik mijenja podatke i završava unos ili odustaje od registracije
  - 3.a unos nepostojeće e-mail adrese:
    1. korisnik nema pristup aplikaciji - aplikacija ostaje na prozoru za verifikaciju

##### UC 2 - Prijava u sustav

- **Glavni sudionik:** posjetitelj
- **Cilj:** dobiti pristup korisničkom sučelju
- **Sudionici:** baza podataka
- **Preduvjet:** korisnik je registriran i verificiran
- **Opis osnovnog tijeka:**
  1. unos e-mail adrese i lozinke
  2. potvrda o ispravnosti unesenih podataka
  3. pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
  - 2.a neispravna e-mail adresa ili lozinka
    1. sustav obavještava korisnika o neuspjelom upisu i ostaje na stranici za prijavu
  - 2.b e-mail adresa nije potvrđena

1. sustav zadržava korisnika na posebnom zaslonu i čeka potvrdu e-mail adrese

### **UC 3 - Registracija/prijava pomoću Google računa**

- **Glavni sudionik:** korisnik/posjetitelj
- **Cilj:** stvoriti račun ako već ne postoji, u suprotnom dobiti pristup korisničkom sučelju
- **Sudionici:** baza podataka
- **Preduvjet:** posjedovanje Google računa (automatski ispunjeno, inače nije moguće skinuti mobilnu aplikaciju)
- **Opis osnovnog tijeka:**
  1. korisnik odabire opciju "Sign in with Google"
  2. odabir jednog od Google računa prisutnih na mobilnom uređaju
  3. pristup korisničkom sučelju

### **UC 4 - Unos osobnih podataka**

- **Glavni sudionik:** posjetitelj
- **Cilj:** unijeti osobne podatke
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je prijavljen pomoću e-mail adrese
- **Opis osnovnog tijeka:**
  1. posjetitelj unosi osobne podatke
  2. posjetitelj sprema unesene podatke
  3. baza podataka se ažurira
- **Opis mogućih odstupanja:**
  - 2.a nisu uneseni obavezni podaci
    1. sustav obavještava korisnika da nije unio obavezne podatke

### **UC 5 - Pregled osobnih podataka**

- **Glavni sudionik:** posjetitelj
- **Cilj:** pregledati osnovne podatke
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je prijavljen
- **Opis osnovnog tijeka:**
  1. korisnik odabire opciju "moj profil"
  2. aplikacija prikazuje osobne podatke korisnika

**UC 6 - Promjena osobnih podataka**

- **Glavni sudionik:** posjetitelj
- **Cilj:** promijeniti osobne podatke
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je prijavljen
- **Opis osnovnog tijeka:**
  1. posjetitelj odabire opciju za promjenu podataka na svom profilu
  2. posjetitelj mijenja osobne podatke
  3. posjetitelj sprema promjene
  4. baza podataka se ažurira
- **Opis mogućih odstupanja:**
  - 3.a posjetitelj promijeni svoje osobne podatke, ali pri tome ostavi neko od obaveznih polja prazno
    1. sustav obavještava korisnika da nije unio obavezne podatke

**UC 7 - Odjava iz sustava**

- **Glavni sudionik:** posjetitelj
- **Cilj:** odjaviti se iz sustava
- **Sudionici:** -
- **Preduvjet:** posjetitelj je prijavljen
- **Opis osnovnog tijeka:**
  1. posjetitelj je na svom profilu
  2. otvara se stranica s osobnim podacima posjetitelja
  3. posjetitelj odabire opciju odjavljivanja
  4. otvara se stranica za prijavu

**UC 8 - Odabir filtera**

- **Glavni sudionik:** posjetitelj
- **Cilj:** odabrati filtere za pretraživanje događaja
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je prijavljen
- **Opis osnovnog tijeka:**
  1. posjetitelj je izabrao opciju "filteri"
  2. posjetitelj bira filtere prema kojima mu se prikazuje samo relevantan sadržaj - vrsta događaja, područje i vremenski raspon

3. posjetitelj potvrđuje odabir

#### **UC 9 - Pregled filtriranih događaja**

- **Glavni sudionik:** posjetitelj
- **Cilj:** pregledati događaje
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je prijavljen
- **Opis osnovnog tijeka:**
  1. baza podataka vraća sve događaje koji zadovoljavaju zadane filtere
  2. posjetitelj pregledava događaje koji zadovoljavaju filtere

#### **UC 10 - Pregled detalja događaja**

- **Glavni sudionik:** posjetitelj
- **Cilj:** pregledati detalje određenog događaja
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je prijavljen
- **Opis osnovnog tijeka:**
  1. posjetitelj odabire događaj čije detalje želi pregledati
  2. otvara se zaslon s detaljima događaja
  3. posjetitelj pregledava detalje događaja

#### **UC 11 - Iskazivanje interesa za određeni događaj**

- **Glavni sudionik:** posjetitelj
- **Cilj:** izraziti interes za određeni događaj
- **Sudionici:** baza podataka
- **Preduvjet:** događaj nije završio, posjetitelj je prijavljen
- **Opis osnovnog tijeka:**
  1. posjetitelj odabire događaj
  2. otvara se zaslon s detaljima događaja
  3. posjetitelj odabire jednu od opcija: "sigurno dolazim", "možda dolazim", "ne dolazim"

#### **UC 12 - Mijenjanje interesa za određeni događaj**

- **Glavni sudionik:** posjetitelj
- **Cilj:** promijeniti interes za određeni događaj
- **Sudionici:** baza podataka

- **Preduvjet:** događaj nije završio, posjetitelj je prijavljen i već je iskazao interes za taj događaj
- **Opis osnovnog tijeka:**
  1. posjetitelj odabire događaj za koji želi promijeniti interes
  2. otvara se zaslon s detaljima događaja
  3. posjetitelj odabire jednu od opcija: "sigurno dolazim", "možda dolazim" ili "ne dolazim"

### UC 13 - Pregled recenzija

- **Glavni sudionik:** posjetitelj
- **Cilj:** pregledati recenzije za određeni događaj
- **Sudionici:** baza podataka
- **Preduvjet:** događaj je završio, posjetitelj je prijavljen
- **Opis osnovnog tijeka:**
  1. posjetitelj bira događaje čije ga recenzije zanimaju
  2. otvara se zaslon s detaljima događaja
  3. posjetitelj pregledava skraćene recenzije
  4. pritišće recenziju koju želi u potpunosti vidjeti
  5. otvara se skočni prozor s detaljima

### UC 14 - Pisanje recenzija

- **Glavni sudionik:** posjetitelj
- **Cilj:** napisati recenziju
- **Sudionici:** baza podataka
- **Preduvjet:** događaj je počeo ili je završio u proteklih 48 sati, posjetitelj je za taj događaj izrazio interes "sigurno dolazim", posjetitelj je prijavljen
- **Opis osnovnog tijeka:**
  1. posjetitelj pod svi događaji bira događaj na kojem je prisustvovao
  2. otvara se zaslon s detaljima događaja
  3. posjetitelj bira opciju "dodaj recenziju"
  4. posjetitelj piše recenziju
  5. posjetitelj objavljuje recenziju
  6. otvara se zaslon s detaljima događaja

**UC 15 - Pregled dobivene obavijesti**

- **Glavni sudionik:** posjetitelj
- **Cilj:** pregledati dobivenu obavijest
- **Sudionici:** baza podataka
- **Preduvjet:** na mobitel je došla obavijest o dodanom događaju koji zadovoljava kriterije postavljenog filtra za obavijesti
- **Opis osnovnog tijeka:**
  1. posjetitelj otvara obavijest
  2. otvara se zaslona s detaljima događaja
- **Opis mogućih odstupanja:**
  - 1.a na tom mobilnom uređaju nitko nije prijavljen u aplikaciju
    1. otvara se ekran za prijavu
    2. nakon uspješnog prijavljivanja, nastavlja se korak 2. osnovnog tijeka

**UC 16 - Postajanje organizatorom**

- **Glavni sudionik:** posjetitelj
- **Cilj:** postati organizator
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je prijavljen
- **Opis osnovnog tijeka:**
  1. posjetitelj pregledava svoj profil
  2. posjetitelj odabire opciju postati organizator
  3. otvara se skočni prozor za odabir načina plaćanja
    - (a) plaćanje PayPalom, on unosi e-mail adresu i lozinku svog PayPal računa
    - (b) kartično plaćanje, on unosi podatke o kartici
  4. posjetitelj unosi podatke o organizaciji
  5. posjetitelj sprema podatke
  6. baza podataka se ažurira
- **Opis mogućih odstupanja:**
  - 3.a neispravan unos podataka
    1. sustav obavještava korisnika, posjetitelj ne može nastaviti dok ih ne unese ili odustane od postajanja organizatorom
  - 3.b plaćanje nije provedeno
    1. sustav obavještava korisnika o neuspjehom plaćanja

5.a nisu uneseni obavezni podaci

1. sustav obavještava korisnika da nije unio obavezne podatke

#### UC 17 - Pregled podataka svoje organizacije

- **Glavni sudionik:** organizator
- **Cilj:** pregledati osnovne podatke o svojoj organizaciji
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je organizator koji je platio mjesečnu članarinu, organizator je prijavljen
- **Opis osnovnog tijeka:**
  1. korisnik na svom profilu odabire opciju informacije o organizaciji
  2. na novootvorenom ekranu pregledava podatke o organizaciji

#### UC 18 - Promjena podataka o organizaciji

- **Glavni sudionik:** organizator
- **Cilj:** promijeniti podatke svoje organizacije
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je organizator koji je platio mjesečnu članarinu, organizator je prijavljen
- **Opis osnovnog tijeka:**
  1. posjetitelj odabire opciju za promjenu podatka na ekranu s informacijama o svojoj organizaciji
  2. posjetitelj mijenja podatke o organizaciji
  3. posjetitelj sprema promjene
  4. baza podataka se ažurira
- **Opis mogućih odstupanja:**
  - 3.a posjetitelj promijeni svoje podatke o organizaciji, ali pri tome ostavi neko od obaveznih polja prazno
    1. sustav obavještava korisnika da nije unio obavezne podatke

#### UC 19 - Dodavanje događaja

- **Glavni sudionik:** organizator
- **Cilj:** dodati događaj
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je organizator koji je platio mjesečnu članarinu, organizator je prijavljen

- **Opis osnovnog tijeka:**
  1. organizator odabire opciju dodavanje novog događaja
  2. na novom zaslonu organizator unosi podatke o događaju
  3. organizator sprema podatke
  4. baza podataka se ažurira
- **Opis mogućih odstupanja:**
  - 3.a organizator nije unio potrebne podatke
    1. sustav prikazuje poruku da je potrebno unijeti potrebne podatke

#### UC 20 - Uređivanje događaja

- **Glavni sudionik:** organizator
- **Cilj:** promijeniti podatke o događaju
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je organizator koji je platio mjesečnu članarinu, organizator je prijavljen
- **Opis osnovnog tijeka:**
  1. na svojem profilu, organizator bira događaj koji želi promijeniti
  2. organizator bira opciju unesi promjene
  3. organizator unosi podatke
  4. organizator sprema podatke
  5. baza podataka se ažurira
- **Opis mogućih odstupanja:**
  - 4.a organizator je obrisao neki od potrebnih podataka
    1. sustav prikazuje poruku da je potrebno unijeti potrebne podatke

#### UC 21 - Brisanje događaja

- **Glavni sudionik:** organizator
- **Cilj:** obrisati događaj
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je organizator, organizator je prijavljen
- **Opis osnovnog tijeka:**
  1. organizator pod moji događaji bira događaj koji želi obrisati
  2. organizator drži događaj koji želi obrisati
  3. organizator potvrđuje odabir
  4. događaj se briše iz baze podataka



**UC 22 - Pregled zainteresiranih za događaj**

- **Glavni sudionik:** organizator
- **Cilj:** pregledati zainteresiranost za događaj
- **Sudionici:** baza podataka
- **Preduvjet:** posjetitelj je organizator, organizator je prijavljen
- **Opis osnovnog tijeka:**
  1. na svojem profilu, organizator bira događaj za koji želi pregledati zainteresirane
  2. otvara se zaslon s detaljima događaja
  3. organizator pritišće broj zainteresiranih
  4. otvara se lista posjetitelja

**UC 23 - Ažuriranje cijene članstva**

- **Glavni sudionik:** administrator
- **Cilj:** postaviti cijenu članstva
- **Sudionici:** baza podataka, Firebase
- **Preduvjet:** korisniku su dodijeljena prava administratora
- **Opis osnovnog tijeka:**
  1. administrator otvara bazu podataka na Firebase konzoli
  2. odabire kolekciju "constants"
  3. otvara dokument "prices"
  4. administrator mijenja vrijednost atributa "monthlyMembership"
  5. prihvaća vrijednost pritiskom na gumb "Update" ili klikom na enter
- **Opis mogućih odstupanja:**
  - 4.a administrator nije unio broj
    1. Firebase javlja grešku

**UC 24 - Pregled posjetitelja**

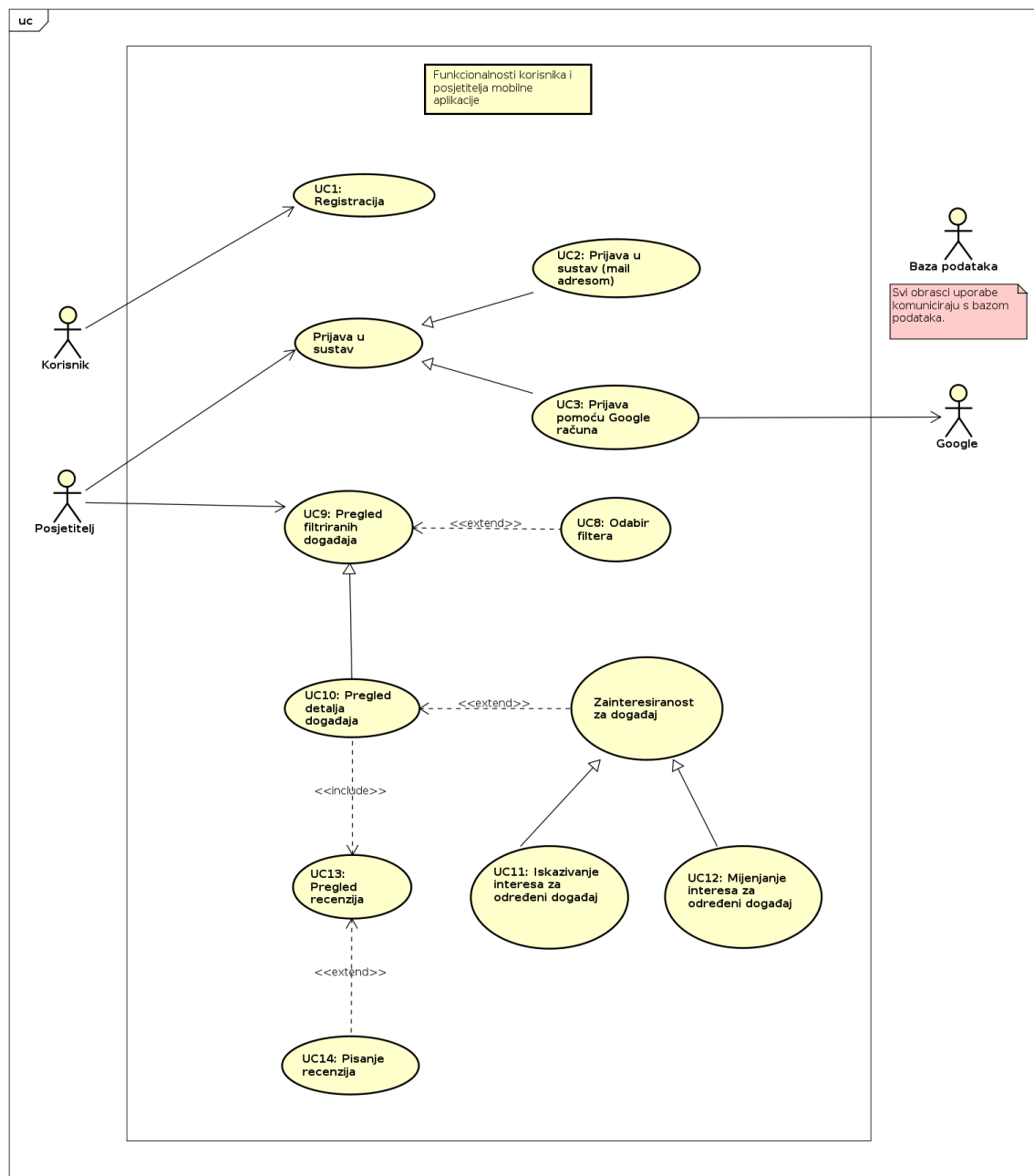
- **Glavni sudionik:** administrator
- **Cilj:** pregledati posjetitelja
- **Sudionici:** baza podataka
- **Preduvjet:** korisniku su dodijeljena prava administratora
- **Opis osnovnog tijeka:**
  1. administrator otvara bazu podataka na Firebase konzoli
  2. odabire kolekciju "users"

3. otvara dokument onog posjetitelja čije podatke želi pregledati

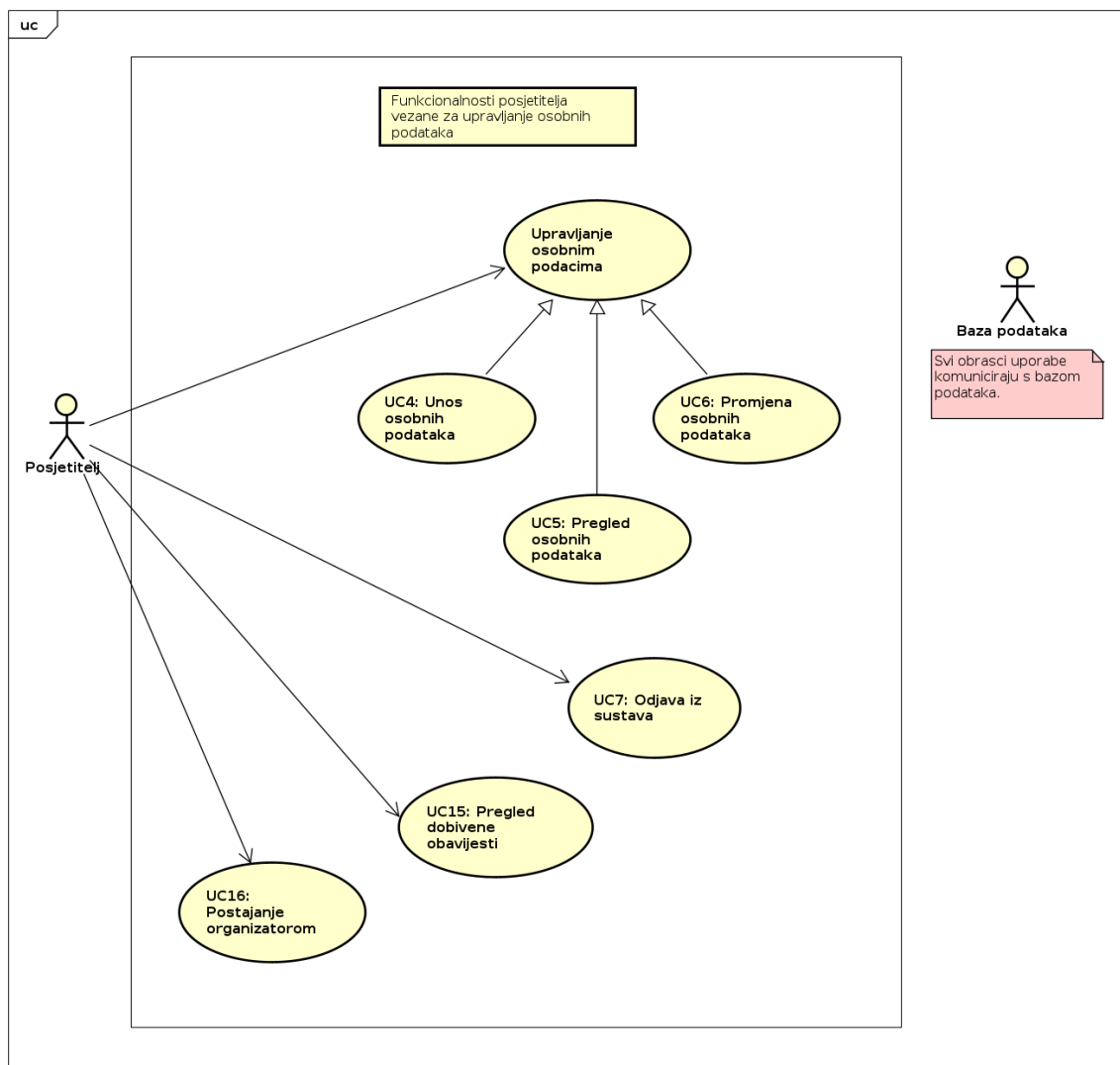
### UC 25 - Brisanje posjetitelja

- **Glavni sudionik:** administrator
- **Cilj:** obrisati posjetitelja
- **Sudionici:** baza podataka
- **Preduvjet:** korisniku su dodijeljena prava administratora
- **Opis osnovnog tijeka:**
  1. administrator otvara "Authentication" na Firebase konzoli
  2. kod posjetitelja kojeg želi obrisati odabire tri točkice
  3. odabire "Delete account"
  4. na skočnom prozoru odabire "Delete"
  5. administrator otvara bazu podataka ("Database") na Firebase konzoli
  6. odabire kolekciju "users"
  7. otvara dokument onog posjetitelja čije podatke želi obrisati
  8. kod posjetitelja kojeg želi obrisati odabire tri točkice
  9. odabire "Delete document"
  10. na skočnom prozoru odabire "Start delete"
- **Opis mogućih odstupanja:**
  - 7.a nema dokumenta posjetitelja jer nikad nije verificiran

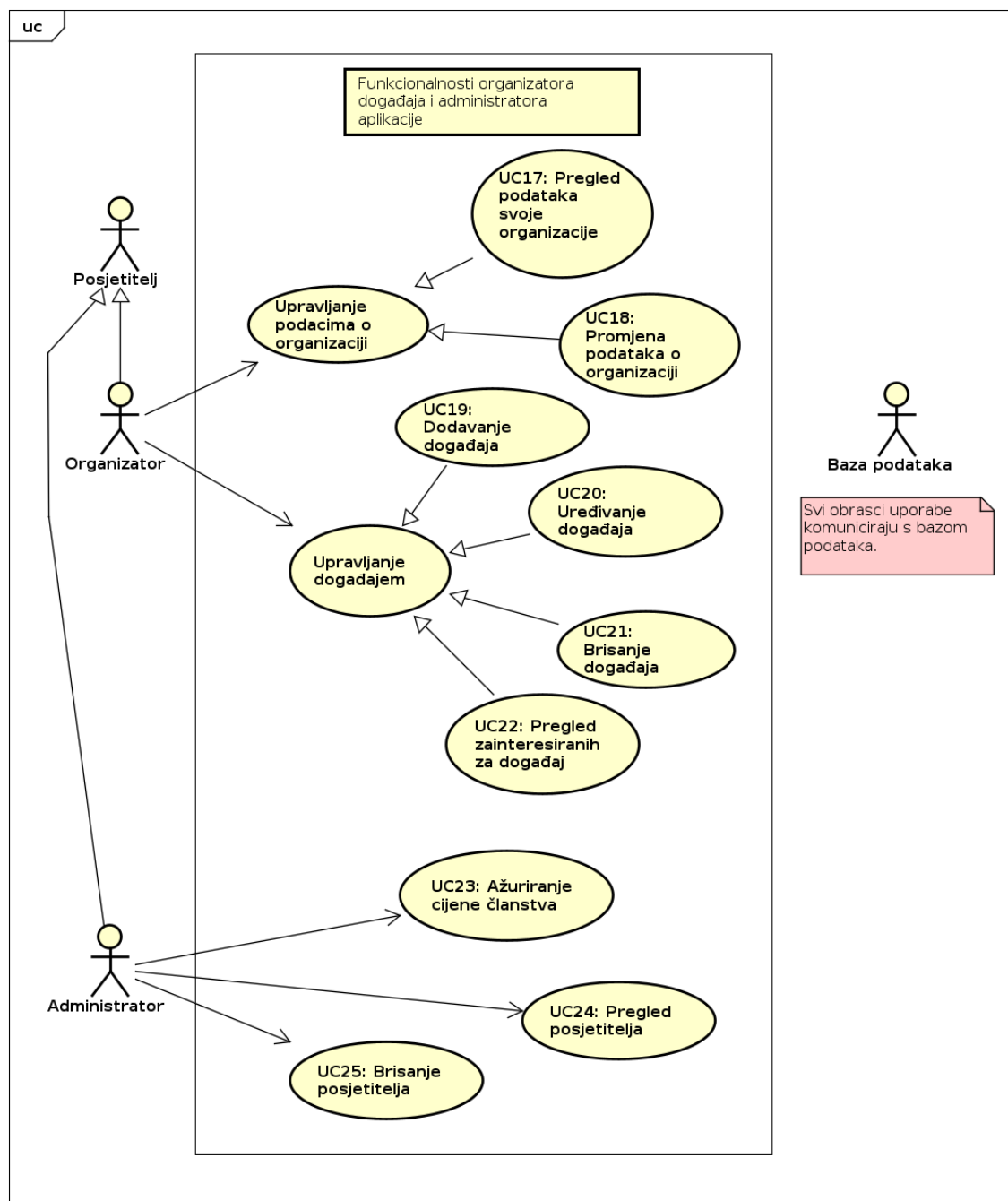
## Dijagrami obrazaca uporabe



Slika 3.1: Dijagram korisnik i posjetitelj



Slika 3.2: Dijagram posjetitelj

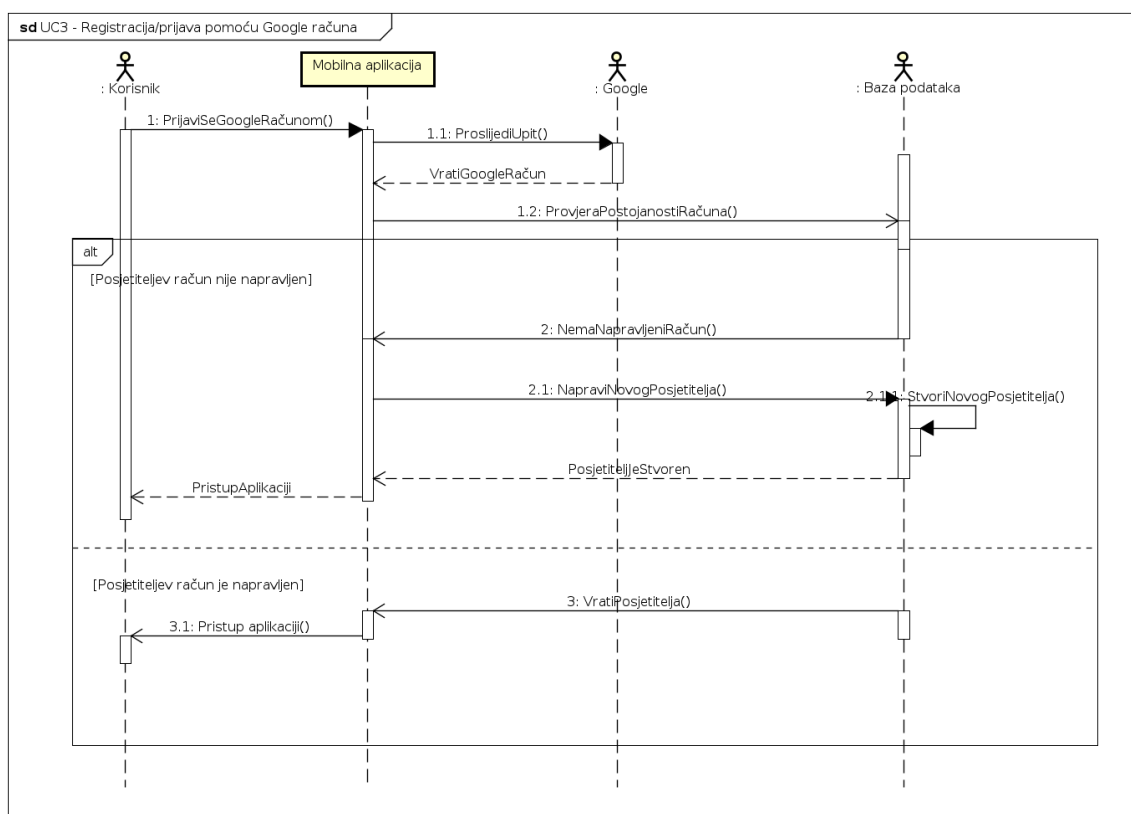


Slika 3.3: Dijagram organizator i administrator

### 3.1.2 Sekvencijski dijagrami

#### Obrazac uporabe UC3 - Registracija/prijava pomoću Google računa

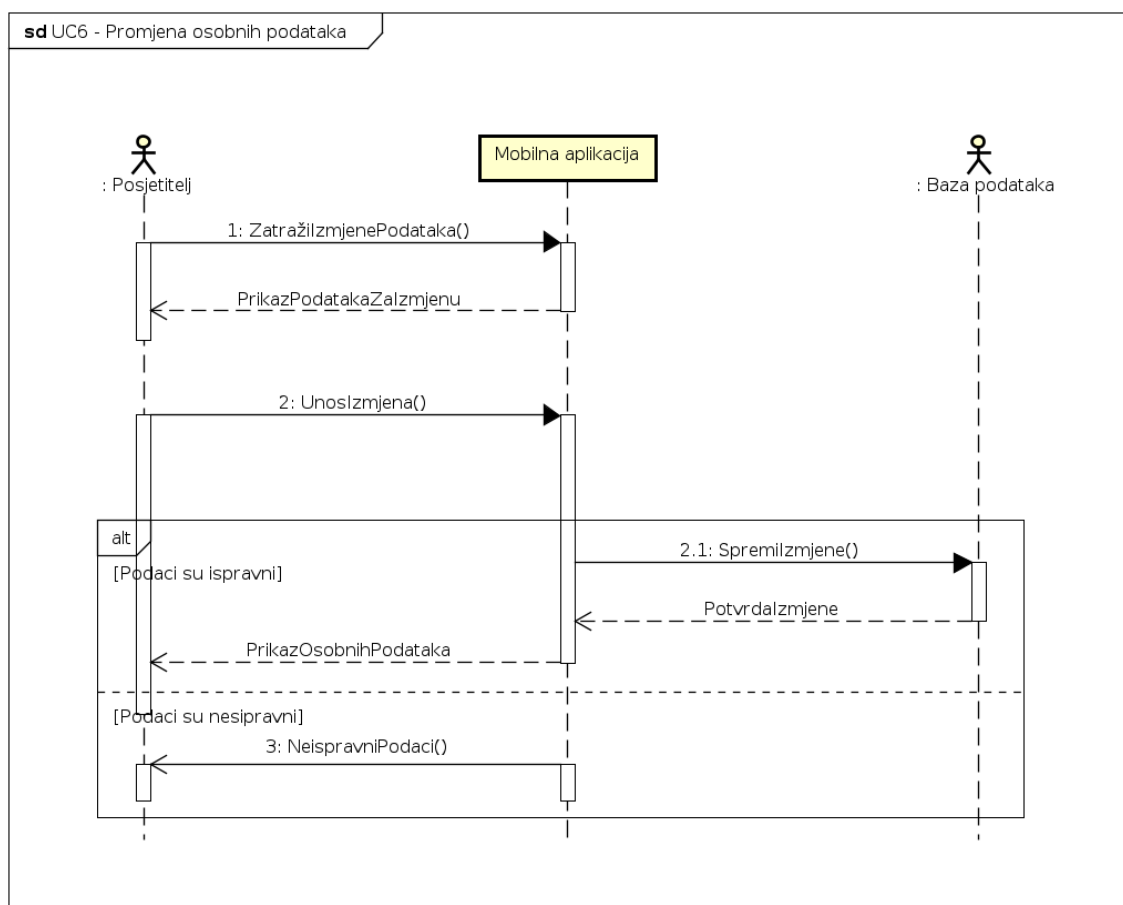
Korisnik odabire opciju "Sign in with Google". Mobilna aplikacija prosljeđuje Googleu upit o postojanosti Google računa. Google vraća aplikaciji Google račun. Mobilna aplikacija prosljeđuje informacije bazi podataka. Ukoliko korisnikov račun nije napravljen, baza podataka obavještava aplikaciju o tome. Mobilna aplikacija obavještava bazu podataka da napravi novog posjetitelja. Baza podataka stvara novog posjetitelja i javlja to aplikaciji, koja korisniku omogućuje pristup aplikaciji. Ukoliko je korisnikov račun već napravljen baza podataka vraća korisnički račun mobilnoj aplikaciji, koja korisniku omogućuje pristup aplikaciji.



Slika 3.4: Prijava pomoću Google računa

**Obrazac uporabe UC6 - Promjena osobnih podataka**

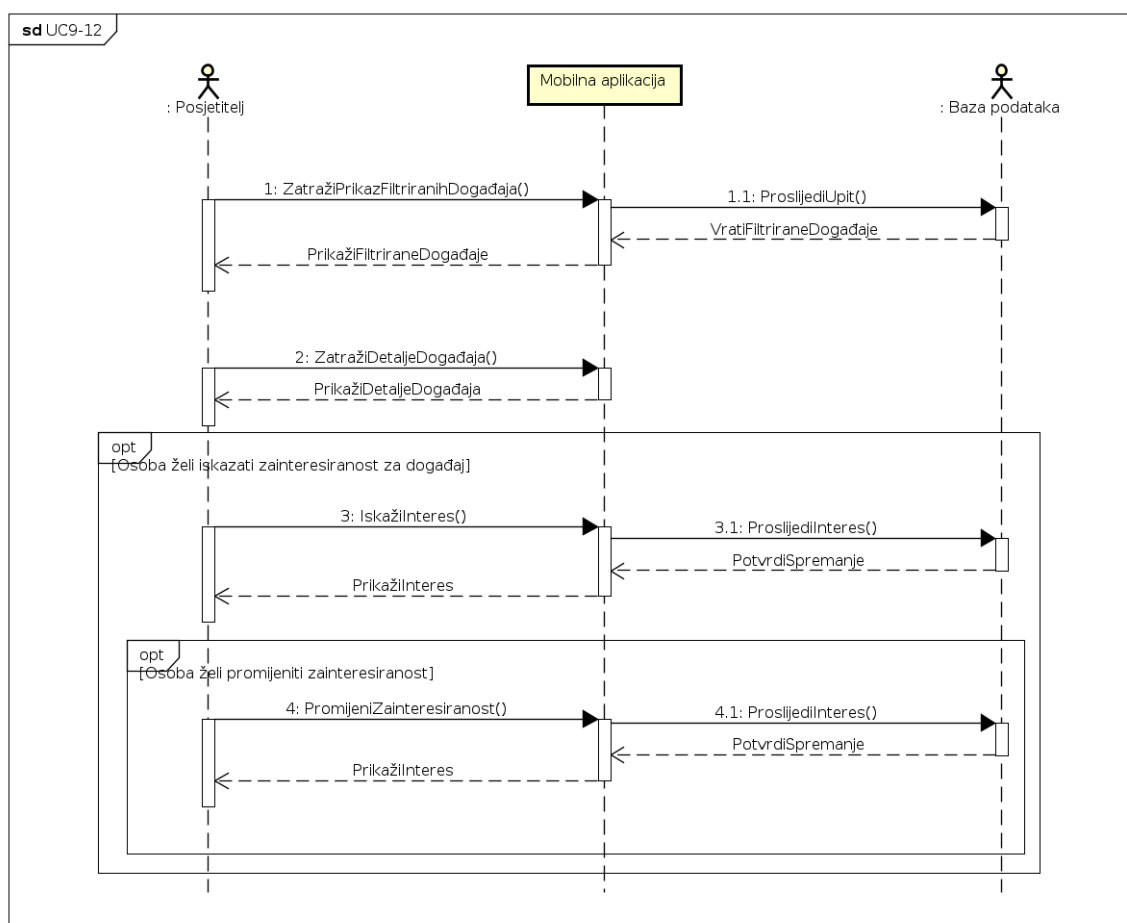
Posjetitelj šalje mobilnoj aplikaciji zahtjev za izmjenu osobnih podataka. Mobilna aplikacija prikazuje posjetitelju podatke za izmjenu. Posjetitelj unosi izmjene. Ukoliko su ispravne, mobilna aplikacija izmijeni podatke u bazi podataka koja vraća potvrdu. Aplikacija prikazuje izmijenjene osobne podatke posjetitelju. Ukoliko su izmijenjeni podaci neispravni ispiše se poruka.



Slika 3.5: Promjena osobnih podataka

### Obrazac uporabe UC9-12 - Pregled filtriranih događaja, pregled detalja događaja, iskazivanje i mijenjanje interesa za određeni događaj

Posjetitelj šalje mobilnoj aplikaciji zahtjev za prikaz filtriranih događaja. Mobilna aplikacija dohvaća listu filtriranih događaja iz baze podataka. Ukoliko postoji barem jedan filtrirani događaj mobilna aplikacija ga vraća posjetitelju. Posjetitelj traži od mobilne aplikacije pregled detalja događaja koje aplikacija tada prikazuje. Posjetitelj može iskazati interes za događaj. Mobilna aplikacija interes proslijedi bazi podataka. Baza podataka sprema posjetiteljev interes i šalje potvrdu aplikaciji koja posjetitelju prikazuje njegov interes. Nakon što je iskazao interes, posjetitelj može promijeniti interes za događaj. Mobilna aplikacija promjenu interesa proslijedi bazi podataka. Baza podataka izmijeni posjetiteljev interes i šalje potvrdu aplikaciji koja posjetitelju prikazuje njegov novi interes.

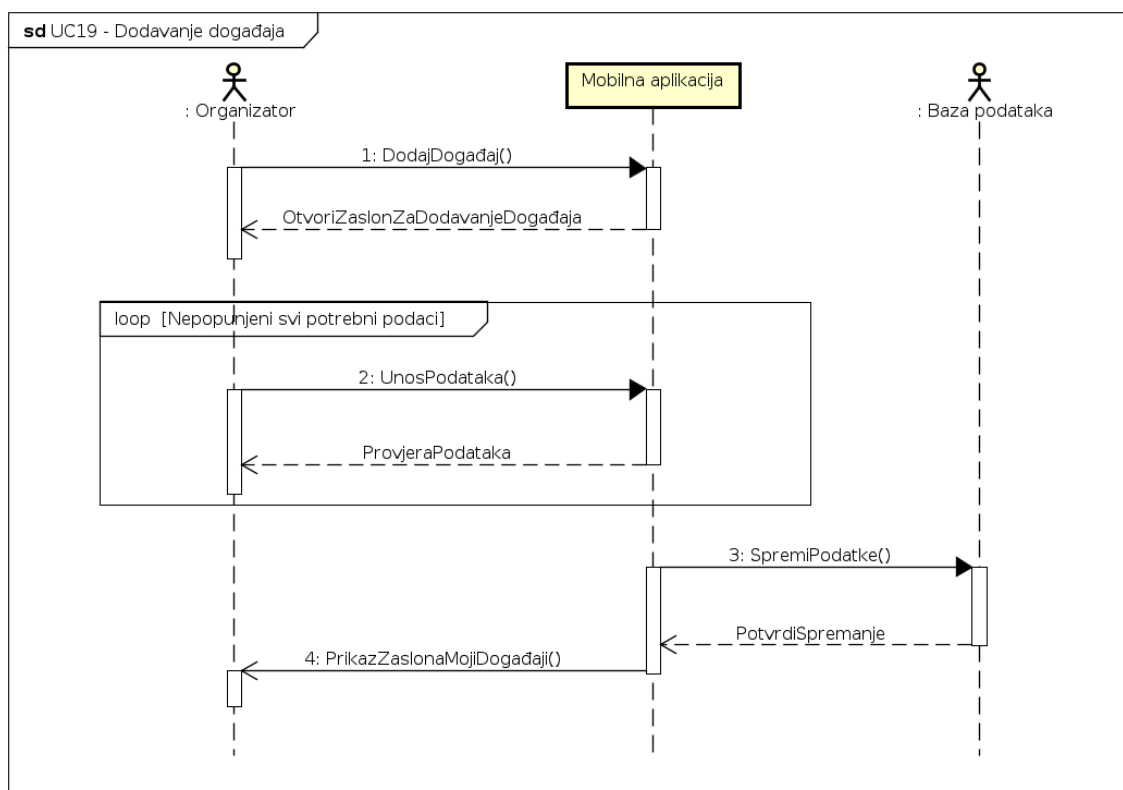


Slika 3.6: Pregled filtriranih događaja, pregled detalja događaja, iskazivanje interesa za određeni događaj, mijenjanje interesa za određeni događaj



**Obrazac uporabe UC19 - Dodavanje događaja**

Organizator odabire opciju "Dodaj događaj". Mobilna aplikacija otvara zaslon za dodavanje događaja. Organizator unosi podatke dok ne popuni sve potrebne podatke o događaju. Nakon svakog unosa mobilna aplikacija provjeri jesu li uneseni podaci. Mobilna aplikacija prosljeđuje događaj u bazu podataka koja ga sprema i vraća potvrdu.



Slika 3.7: Dodavanje događaja

## 3.2 Ostali zahtjevi

- Sustav treba biti implementiran kao mobilna aplikacija koristeći objektno-orijentirane jezike
- Sustav kao valutu članarine koristi HRK
- Sustav treba omogućiti istovremeni rad više korisnika
- Sustav treba biti intuitivan za korištenje
- Tekstualni sadržaj sustava podržava hrvatsku abecedu i dijakritičke znakove
- Pristup bazi podataka ne smije trajati duže od nekoliko sekundi
- Funkcionalnosti sustava ne smiju biti narušene prilikom nadogradnje
- Funkcionalnosti sustava ne smiju biti narušene neispravnim korištenjem sučelja
- Veza sustava s bazom podataka treba biti brza i dobro zaštićena

## 4. Arhitektura i dizajn sustava

Arhitektura se može podijeliti na dva glavna podsustava:

- mobilna (Android) aplikacija
- Firebase - od kojeg koristimo Authentication, Cloud Firestore, Cloud Storage, Cloud Functions, Cloud Messaging, Crashlytics, Performance Monitoring i App Distribution

Mobilna aplikacija je program koji se instalira na mobilni uređaj. Aplikacija se može izvoditi na različitim verzijama Android operacijskih sustava. Izvodi se već prevedeni kod. Korisniku se putem intuitivnog korisničkog sučelja omogućava pristup sadržaju koji se nudi u aplikaciji.

Firebase je razvojna platforma koju je razvio Firebase, Inc 2011., a preuzeo Google 2014. Cilj platforme je izgradnja bržih, sigurnijih i skalabilnijih aplikacija. Uz svu funkcionalnost pruža i uvide u razne statistike dobivene temeljem korištenja aplikacije.

Firebase Authentication pruža backend usluge za autentifikaciju korisnika. Podržava nekoliko različitih načina autentifikacije od kojih mi koristimo onaj pomoću lozinke i Google računa. Za provjeru autentičnosti koristi industrijske standarde poput OAuth 2.0 i OpenID Connect što ga čini jednim od sigurnijih načina autentifikacije. Za administratore sustava pruža jednostavan uvid u korisnike i njihovu aktivnost.

Cloud Firestore je NoSQL baza podataka koja se nalazi u oblaku. Pruža mogućnost postavljanja sigurnosnih pravila, slično, ali šire nego obične SQL baze podataka. Usluga je razvijena s ciljem da upiti dohvaćanja budu jednostavni, efikasni i fleksibilni. Nadalje, brzina izvođenja upita ne ovisi o veličini baze podataka. Glavna značajka je korištenje NoSQL baze koja, umjesto u tablice, podatke sprema u dokumente koji se grupiraju u kolekcije. U dokumentima se mogu nalaziti i kompleksnije strukture podataka, poput nizova i mapa što uklanja potrebu za spajanjem tablica poput one u klasičnim SQL bazama. Zbog toga su upiti čitanja puno brži, uz cijenu sporijih upita ažuriranja i pisanja. Međutim, u današnje vrijeme upiti

čitanja mnogo su češći od upita pisanja. Također, ovakvo oblikovanje ponekad uzrokuje redundantne podatke što ne stvara nikakav problem jer nam Firebase eliminira brigu o fizičkoj implementaciji. Administratori sustava u Firebase consoli vrlo jednostavno pristupaju bazi podataka (Database) te imaju mogućnost upravljanja svim podacima unesenim u bazu.

Cloud Storage je baza podataka namijenjena za spremanje većih datoteka, za razliku od Cloud Firestore čija je primarna namjena spremanja teksta ili objekata u JSON formatu. Ovdje se spremaju slike i videa događaja te profilne slike posjetitelja i organizacija.

Cloud Functions pozivaju se prilikom nekih događaja koji se dogode u drugim Firebase servisima. Konkretno, mi koristimo Funkcije da bi saznali kad je stvoren novi događaj u bazi podataka, pronašli korisnike koji su se pretplatili na obavijesti o tom tipu događaja i poslali im obavijest. Sav kod koji se izvršava u Funkcijama mora biti pisan u JavaScriptu ili TypeScriptu.

Firebase Cloud Messaging pruža pouzdanu vezu između poslužitelja i mobilnog uređaja koja omogućuje slanje poruka i obavijesti na mobilni uređaj. Pomoću ove usluge dostavljaju se obavijesti posjetiteljima ako je stvoren novi događaj koji odgovara tipu događaja na koji se posjetitelj pretplatio.

Firebase Crashlytics služi za praćenje problema koji se javljaju prilikom razvoja aplikacije na način da razvojni tim može putem web sučelja pregledati kada, kako i na kojim uređajima je došlo do rušenja aplikacije. Također, mogu se slati i posebne log poruke koje nisu nužno vezane uz poteškoće u radu aplikacije, već su to neke informacije bitne razvojnom timu.

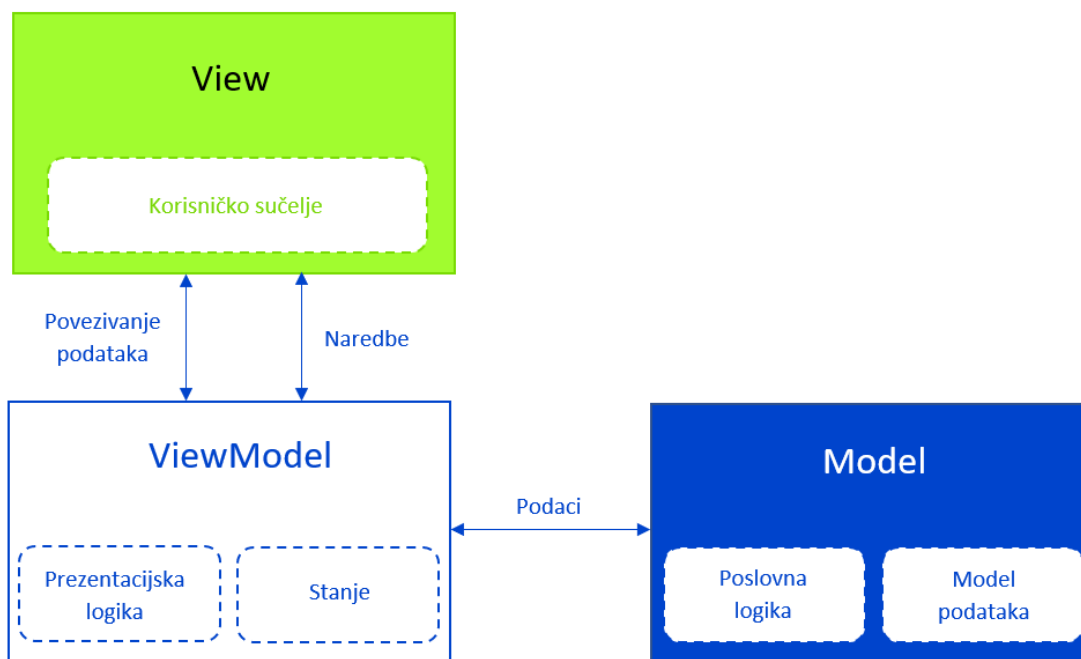
Firebase Performance Monitoring služi za praćenje performansi naše aplikacije odnosno daje nam uvid u trajanje mrežnih zahtjeva, trajanje učitavanja ekrana, koliko vremena aplikacija provodi budna u pozadini i slično.

Firebase App Distribution služi za slanje aplikacije testerima prije produkcije da bi se na vrijeme saznali određeni problemi. Korisnici putem posebne aplikacije i putem maila dobivaju obavijesti kad razvojni tim objavi novu verziju i potom je instaliraju na uređaj. U Crashlyticsu se vidi iz koje verzije aplikacije dolaze podaci.

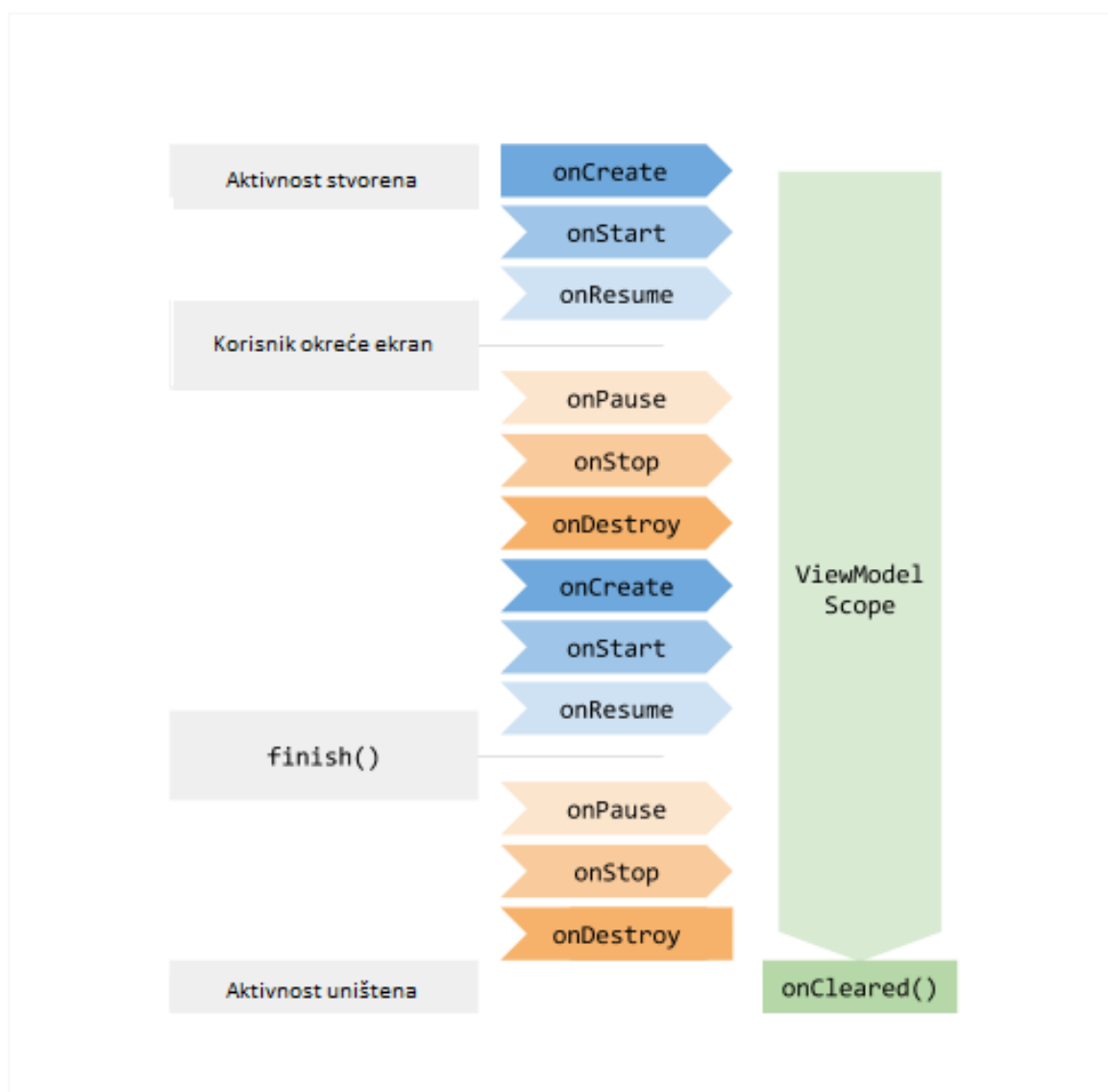
Programski jezik koji smo odabrali za izradu mobilne aplikacije je Kotlin, a odabrano razvojno okruženje je Android Studio. Kotlin je nedavno postao preporučeni jezik za izradu Android aplikacija, dok je Android Studio oduvijek bio službeni alat za izradu istih.

Arhitektura koju koristimo je MVVM (Model-View-ViewModel). MVVM je najnovija i trenutno preporučena arhitektura za Android aplikacije. MVVM sastoji se od:

- **Model** - služi za opis podatkovnih objekata i popratnih metoda potrebnih za funkcioniranje aplikacije.
- **View** - služi za upravljanje grafičkim sučeljem i reagiranje na događaje.
- **ViewModel** - služi za obradu podataka koje koristi View. Pri izradi mobilnih aplikacija valja paziti na potrošnju baterije, mobilnih podataka i slično pa pri dohvaćanju i obradi podataka moramo biti pažljivi. Iz tog razloga, podatke koji se ne mijenjaju često smijemo dohvaćati/obrađivati samo kada je to nužno. Svaki View, odnosno Activity ima svoj životni ciklus u kojem može prolaziti kroz različita stanja, ali ViewModel se ne mijenja od stvaranja do uništavanja Viewa što omogućava da svi podaci potrebni za neki View budu dostupni dok taj View postoji bez nepotrebnog dohvaćanja/obrađivanja.



Slika 4.1: Arhitektura MVVM



Slika 4.2: Životni ciklus ViewModela

## 4.1 Baza podataka

Kao što je prethodno spomenuto, koristimo NoSQL bazu podataka u kojoj se podaci spremaju u dokumente koji se grupiraju u kolekcije. Dodatno, svaki dokument može sadržavati podkolekciju sa svojim dokumentima. Naša baza sastoji se od sljedećih kolekcija:

- users
- events

- constants

Svaki dokument ima svoj jedinstveni id koji je automatski generiran osim u kolekciji **users** u kojoj je id dokumenta e-mail tog usera.

#### 4.1.1 Opis tablica

Svaki dokument kolekcije **users** sadrži attribute: email, firstName, lastName, username, dateOfBirth, zodiacSign, role, deviceToken.

Tablica 4.1: Atributi dokumenata kolekcije users

users		
email	string	e-mail adresa posjetitelja/organizatora
firstName	string	ime posjetitelja/organizatora
lastName	string	prezime posjetitelja/organizatora
username	string	korisničko ime posjetitelja/organizatora
dateOfBirth	timestamp	datum rođenja posjetitelja/organizatora
zodiacSign	string	horoskopski znak posjetitelja/organizatora
role	string	uloga - posjetitelj ili organizator
membershipDuration	timestamp	do kada traje članarina (ako je korisnik organizator)
deviceToken	string	identifikator mobilnog uređaja

Svaki user koji ima postavljeno da želi da mu dolaze obavijesti za određeni filter, ima podkolekciju notifications.

Svaki dokument kolekcije **notifications** sadrži attribute: title, content, eventType, location, after, before.

Tablica 4.2: Atributi dokumenata kolekcije notifications

notifications		
title	string	naslov obavijesti
content	string	tekst obavijesti
eventType	string	vrsta događaja za koju posjetitelj želi primati obavijesti

notifications		
location	string	područje za koje posjetitelj želi primati obavijesti
after	timestamp	vrijeme nakon kojeg počinje događaj
before	timestamp	vrijeme prije kojeg počinje događaj

Svaki user koji je organizator, ima podkolekciju organizations.

Svaki dokument kolekcije **organizations** sadrži attribute: organizationName, webPage, address.

Tablica 4.3: Atributi dokumenata kolekcije organizations

organizations		
organizationName	string	ime organizacije
webPage	string	web/Facebook stranica organizacije
address	string	adresa organizacije

Svaki dokument kolekcije **events** sadrži attribute: organizer, title, description, startTime, endTime, eventType, location, going, maybeGoing, notGoing.

Tablica 4.4: Atributi dokumenata kolekcije events

events		
organizer	reference	referenca na organizatora
title	string	ime događaja
description	string	opis događaja
startTime	timestamp	vrijeme početka događaja
endTime	timestamp	vrijeme završetka događaja
createdAt	timestamp	vrijeme kreiranja događaja
eventType	array<string>	vrsta događaja - može spadati u više kategorija
location	string	lokacija događaja - područje (grad)
address	string	egzaktna adresa događaja
going	array<reference>	posjetitelji koji su odabrali opciju "going"



<b>events</b>		
maybeGoing	array<reference>	posjetitelji koji su odabrali opciju "maybe going"
notGoing	array<reference>	posjetitelji koji su odabrali opciju "not going"

Svaki event koji ima recenzije sadrži podkolekciju reviews.

Svaki dokument kolekcije **reviews** sadrži attribute: userRef, createdAt, text, rating.

Tablica 4.5: Atributi dokumenata kolekcije reviews

<b>reviews</b>		
userRef	reference	referenca na posjetitelja koji je napisao recenziju
createdAt	timestamp	vrijeme pisanja recenzije
text	string	tekst recenzije
rating	number	ocjena događaja

Kolekcija **constants** sadrži dokumente s raznim konstantama. Dokument prices sadrži atribut monthlyMembership.

Tablica 4.6: Atributi dokumenata prices

<b>prices</b>		
monthlyMembership	number	cijena mjesečne članarine

Dokument cities sadrži atribut croatia koji je lista gradova u Hrvatskoj.

Tablica 4.7: Atributi dokumenata cities

<b>cities</b>		
croatia	array<string>	gradovi u Hrvatskoj

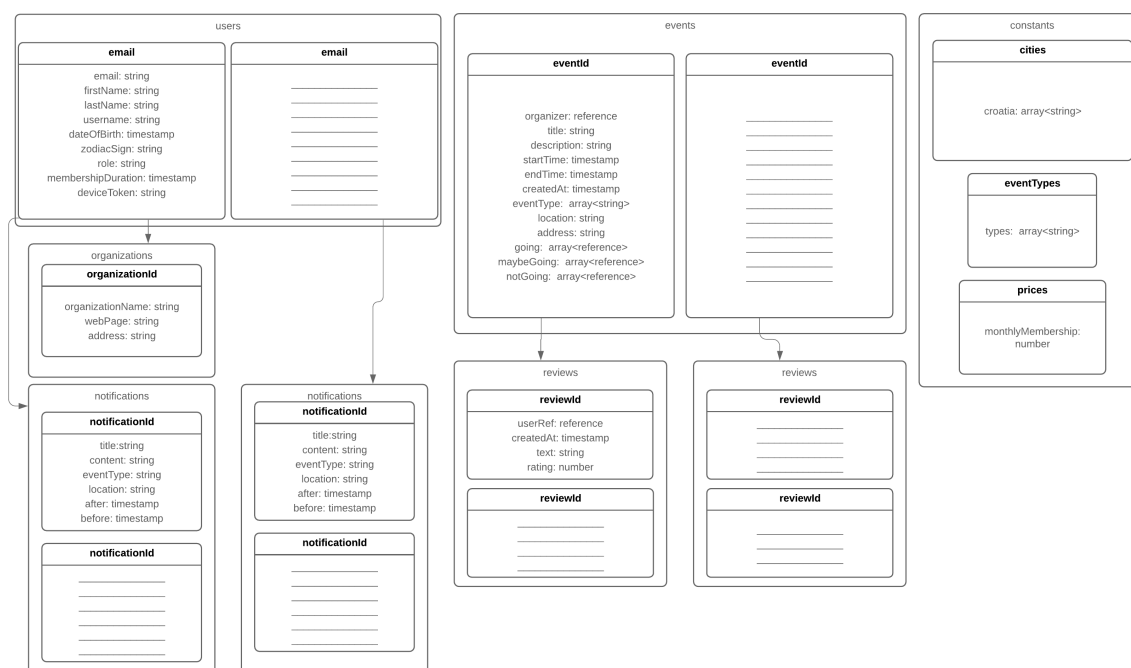
Dokument eventTypes sadrži atribut types koji je lista tipova događaja.

Tablica 4.8: Atributi dokumenata eventTypes

eventTypes		
types	array<string>	tipovi događaja

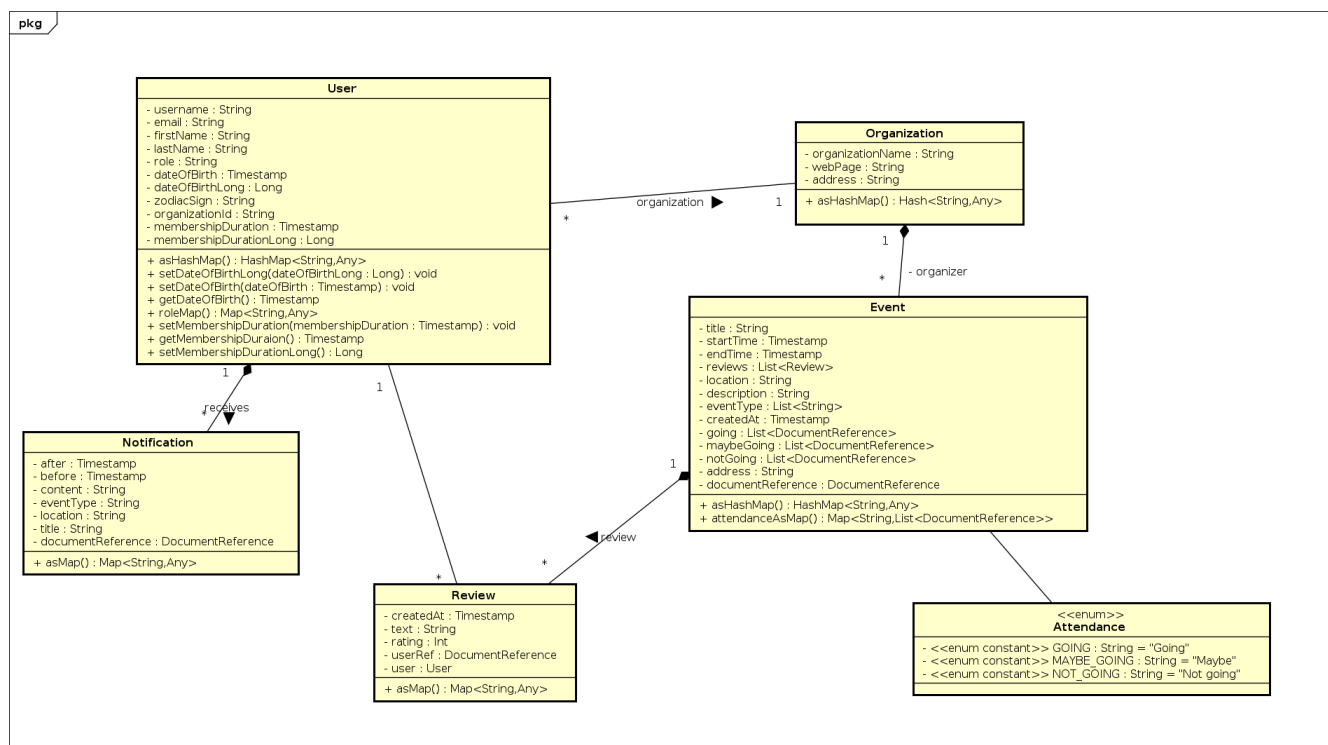
U Firebase Storageu nalaze se mape images i videos. U obje mape nalaze se mape čija su imena id dokumenta događaja za koji su te slike i videi stavljeni. U mapi images nalaze se i mape čija su imena e-mailovi korisnika aplikacije, te se u takvim mapama nalazi profilna slika korisnika s tim e-mailom, a takva mapa postoji isključivo ako je korisnik postavio profilnu sliku (više nema default sliku).

### 4.1.2 Dijagram baze podataka



Slika 4.3: Dijagram baze podataka

## 4.2 Dijagram razreda

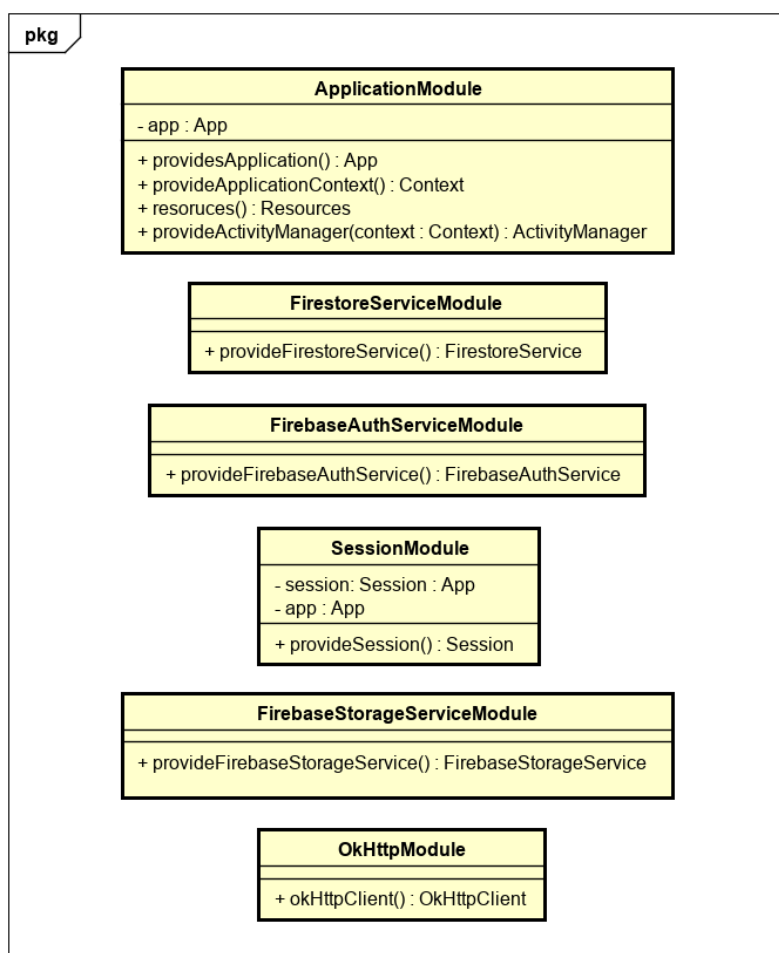


Slika 4.4: Dijagram razreda - model

Na slici 4.4 prikazan je dijagram razreda pri čemu metoda `asHashMap` pretvara sadržaj objekta u mapu što nam je potrebno za spremanje u bazu podataka. Razred **User** predstavlja korisnika koji može biti organizator ili posjetitelj. Razred **Organizations** predstavlja organizaciju. Razred **Event** predstavlja pojedini događaj kojeg stvara organizator, dok ostali korisnici mogu pokazati zainteresiranost jednom od tri mogućnosti (dolazim, možda dolazim, ne dolazim). Enumeracija **Attendance** koristi se za prikaz teksta na gumbima za iskazivanje zainteresiranosti za događaj. Razred **Review** služi posjetiteljima nekog događaja da iskažu svoje mišljenje o tom događaju kako bi ljudi ubuduće znali kakvi su događaji njegovog organizatora. Razred **Notifications** sprema podatke o postavljenim obavijestima koje aplikacija treba poslati.

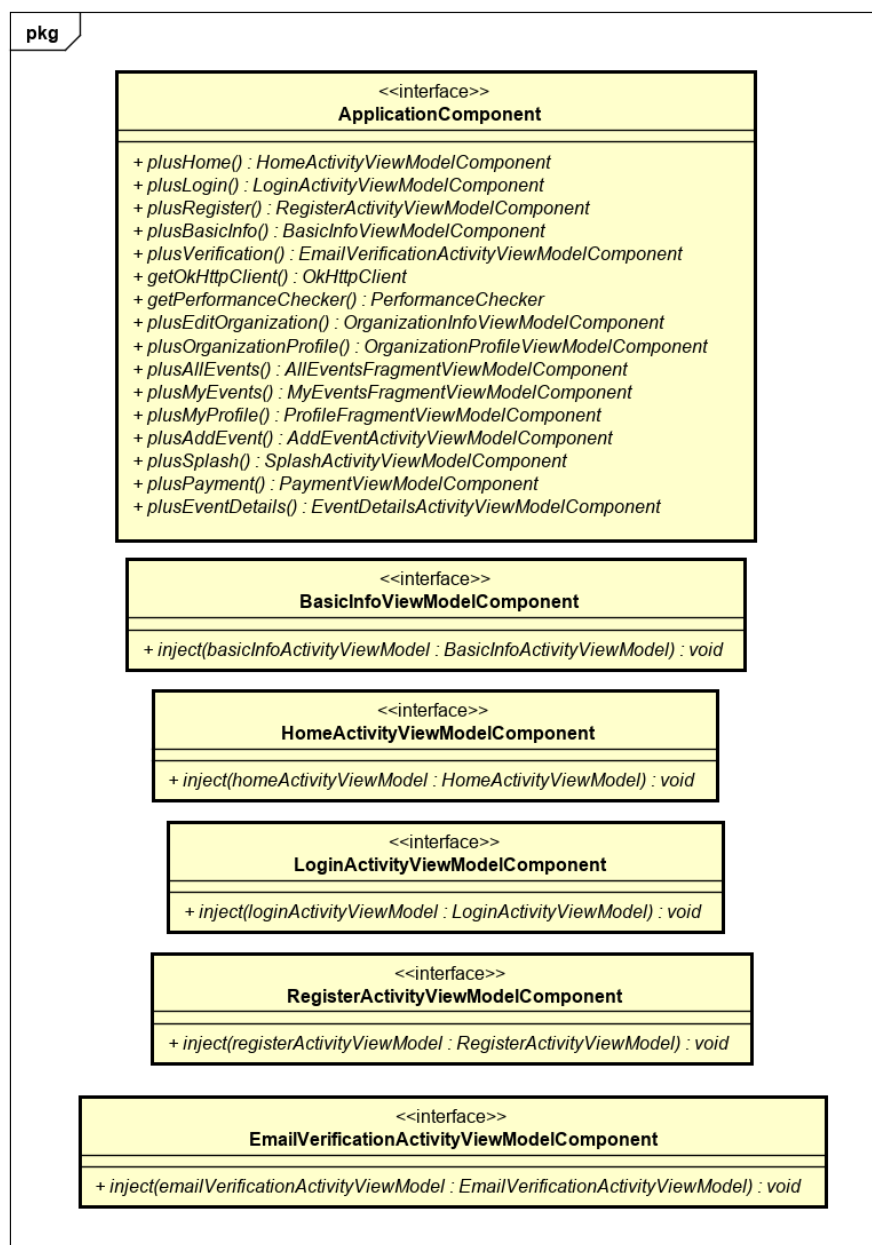
Dependency injection (di) oblikovni je obrazac koji omogućava ubacivanje (engl. injection) podataka koji bi se inače trebali nalaziti u konstruktoru. Postoje klase ko-

jima ne možemo mijenjati konstruktore pa je potreban drugi način za ubacivanje. Također, di omogućava da se neke klase instanciraju samo jednom u cijeloj aplikaciji. Na primjer, za cijelu aplikaciju dovoljna nam je samo jedna instanca klase koja omogućava slanje i primanje HTTP zahtjeva te nije potrebno svaki put kad želimo slati ili primiti HTTP zahtjev stvarati novu instancu. Dependency injection popularno je rješenje za ove probleme. Za dependency injection koristimo Dagger2.

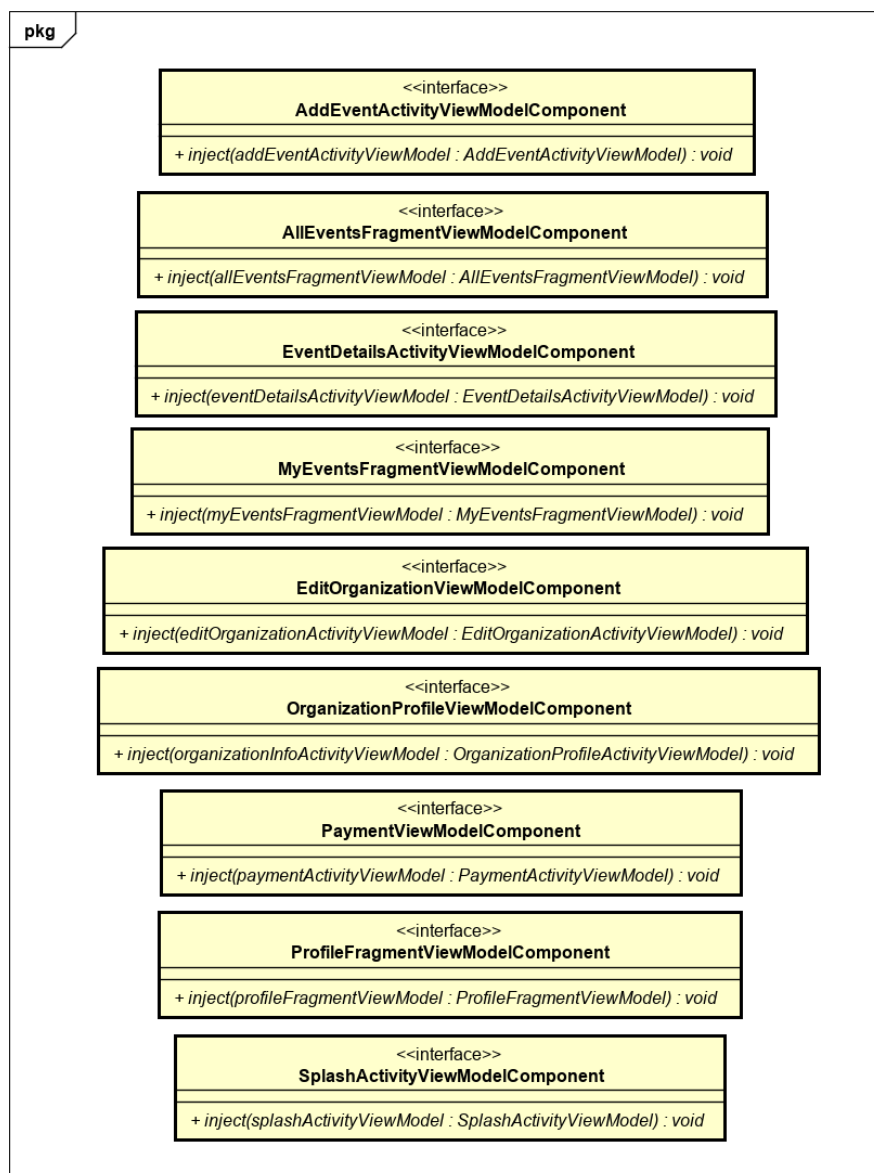


Slika 4.5: Dijagram razreda modula

Modul je klasa potrebna za di i u njoj se nalaze metode u kojima je opisano kako se stvara instanca neke klase. Potrebno je napraviti metodu za svaki tip objekta koji na ovaj način želimo instancirati. Metode su grupirane u različite module, ovisno o tome uz što koji tip objekata povezujemo. Moduli se mogu koristiti na različitim razinama pa tako jedan modul može služiti za instanciranje na razini cijele aplikacije, a drugi samo na razini ViewModela.

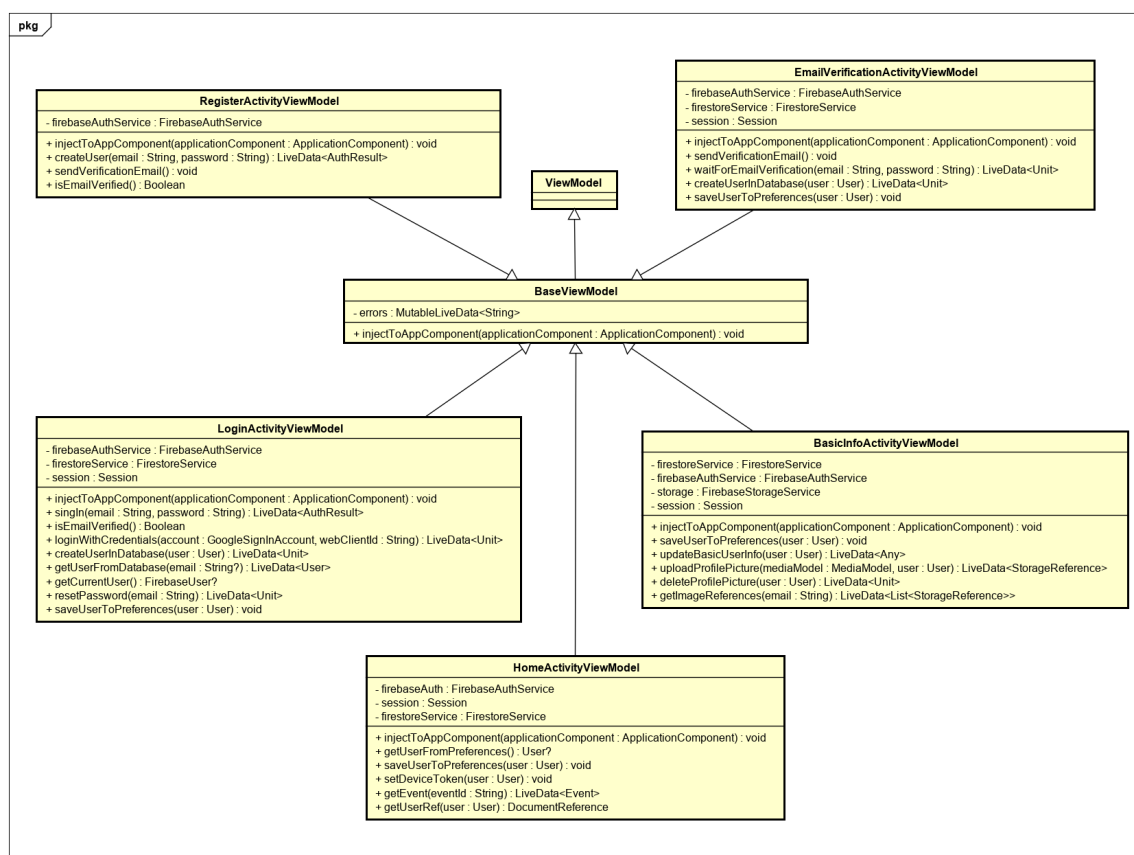


Slika 4.6: Dijagram razreda komponenti

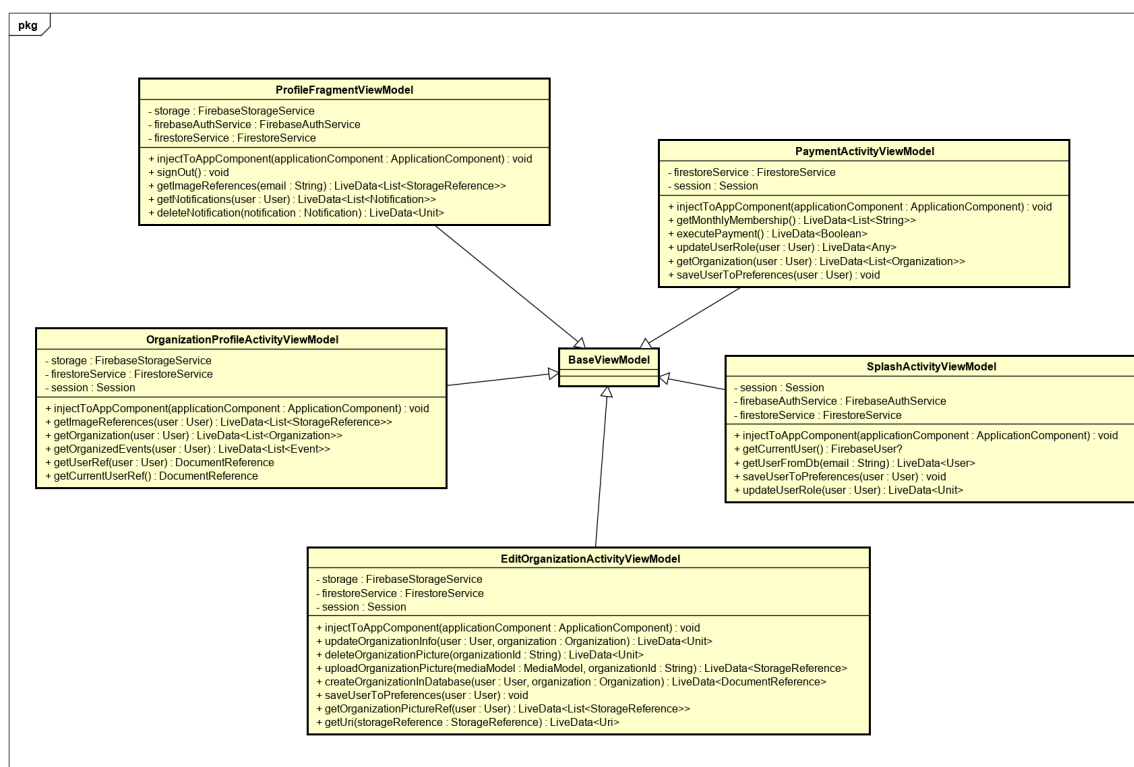


Slika 4.7: Dijagram razreda komponenti - drugi dio

Komponenta je sučelje u kojem se navodi koji moduli (pomoću anotacija) se koriste za pojedinu klasu pa tako svaki ViewModel ima popratnu Component klasu u kojoj su navedene te informacije. Moduli koji su uključeni na razini aplikacije (sučelje ApplicationComponent) ne moraju se i ne smiju ponovno navoditi za pojedinu komponentu.

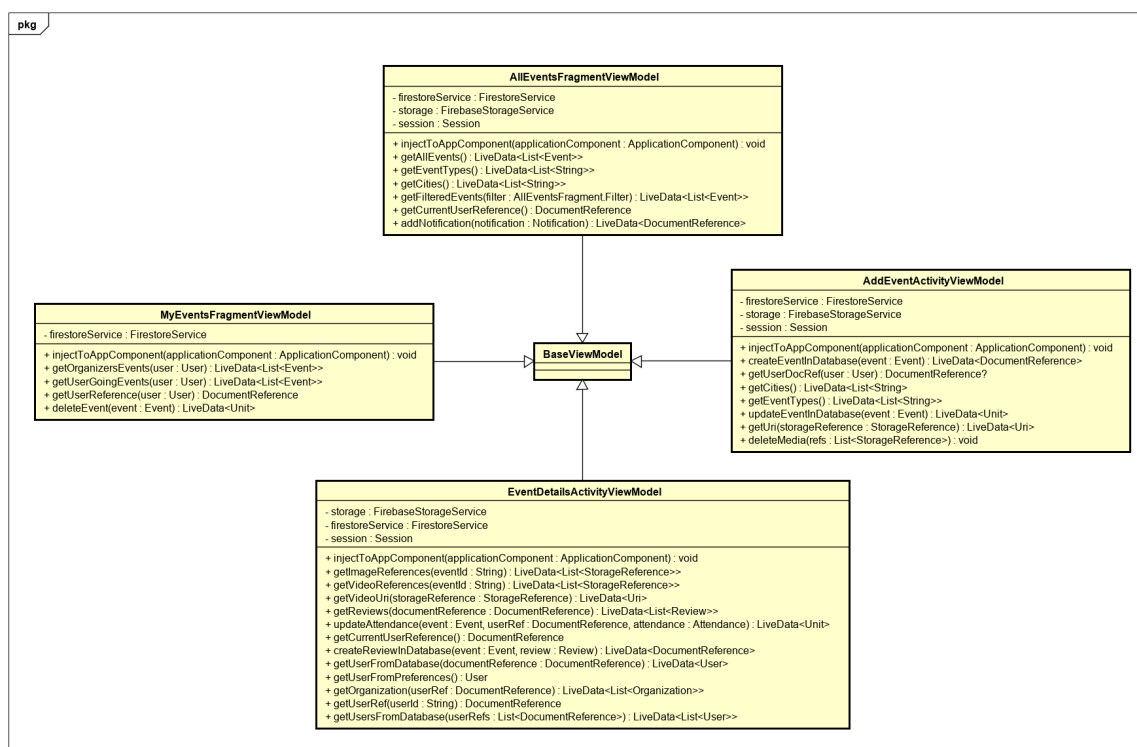


Slika 4.8: Dijagram razreda ViewModela



Slika 4.9: Dijagram razreda ViewModela - drugi dio

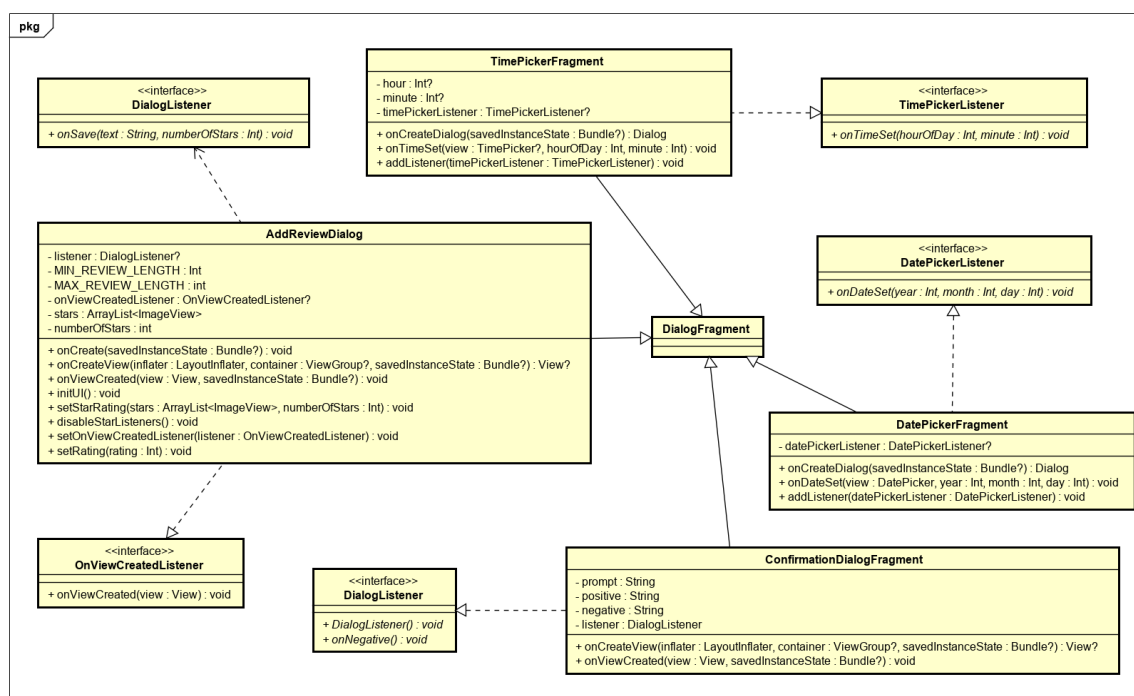




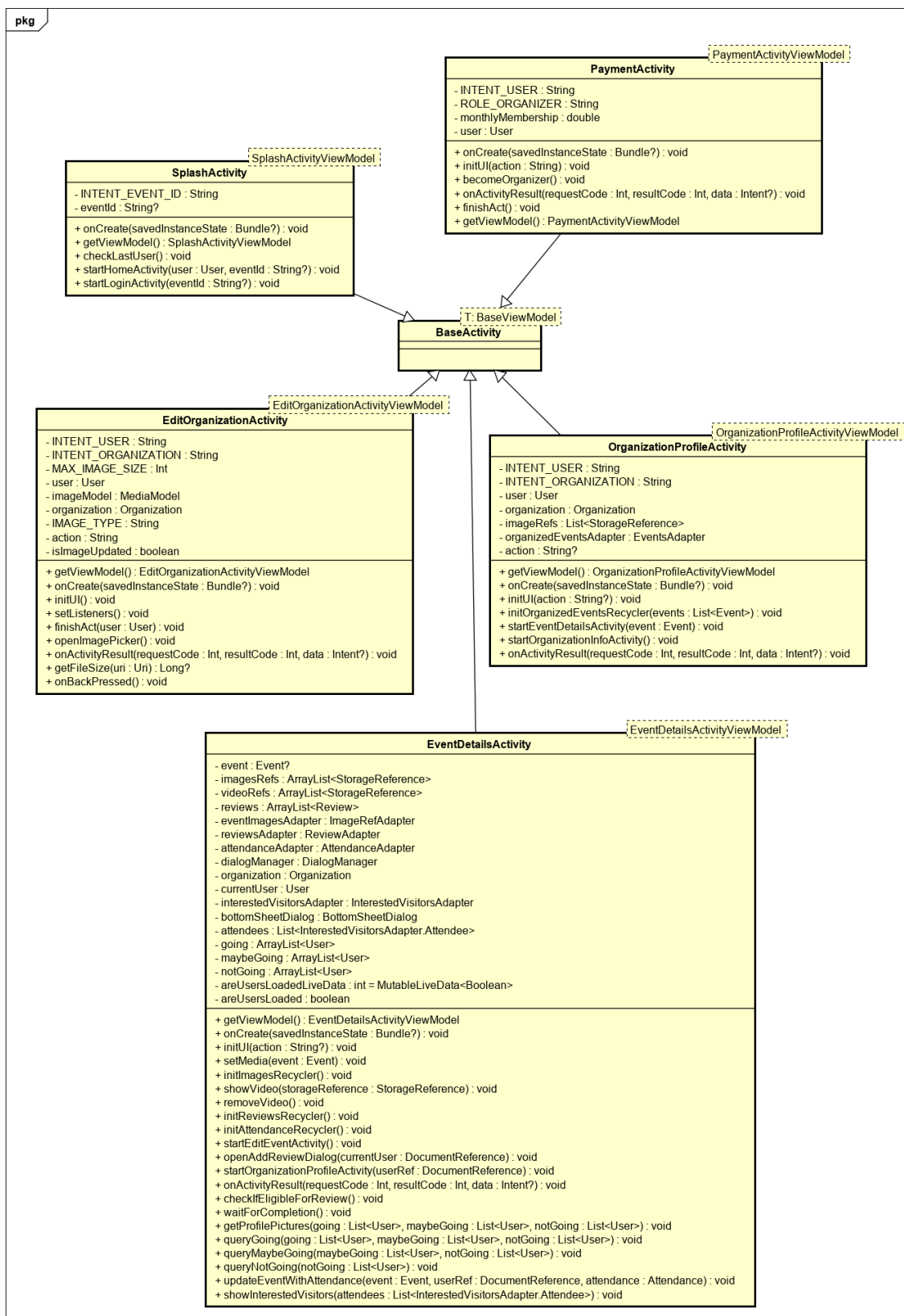
Slika 4.10: Dijagram razreda ViewModela - treći dio

Svi ViewModeli koji se koriste nasljeđuju klasu BaseViewModel u kojoj se nalazi varijabla errors koja služi kako bi se sve poruke o greškama u aplikaciji objavljivale na jednom mjestu i onda kasnije obrađivale na uglavnom jednak način. Također, tu se nalazi još metoda injectToAppComponent koja služi za povezivanje pojedine komponente potrebne za dependency injection i pojedinog ViewModela. Ostale metode u ViewModelima uglavnom služe za komunikaciju s poslužiteljem preko metoda prikazanih u dijagramu razreda Utilities.

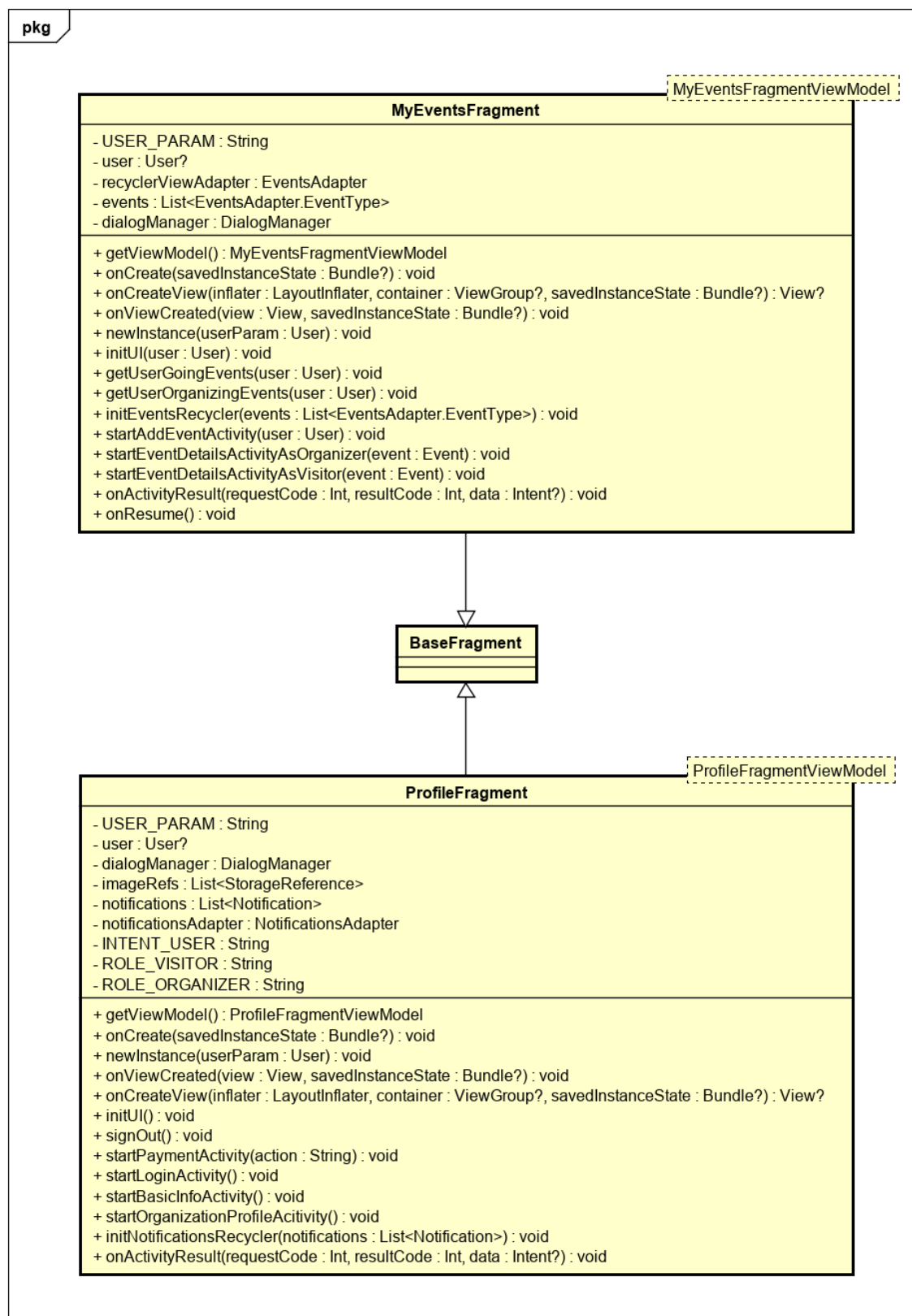




Slika 4.13: Dijagram razreda View - treći dio

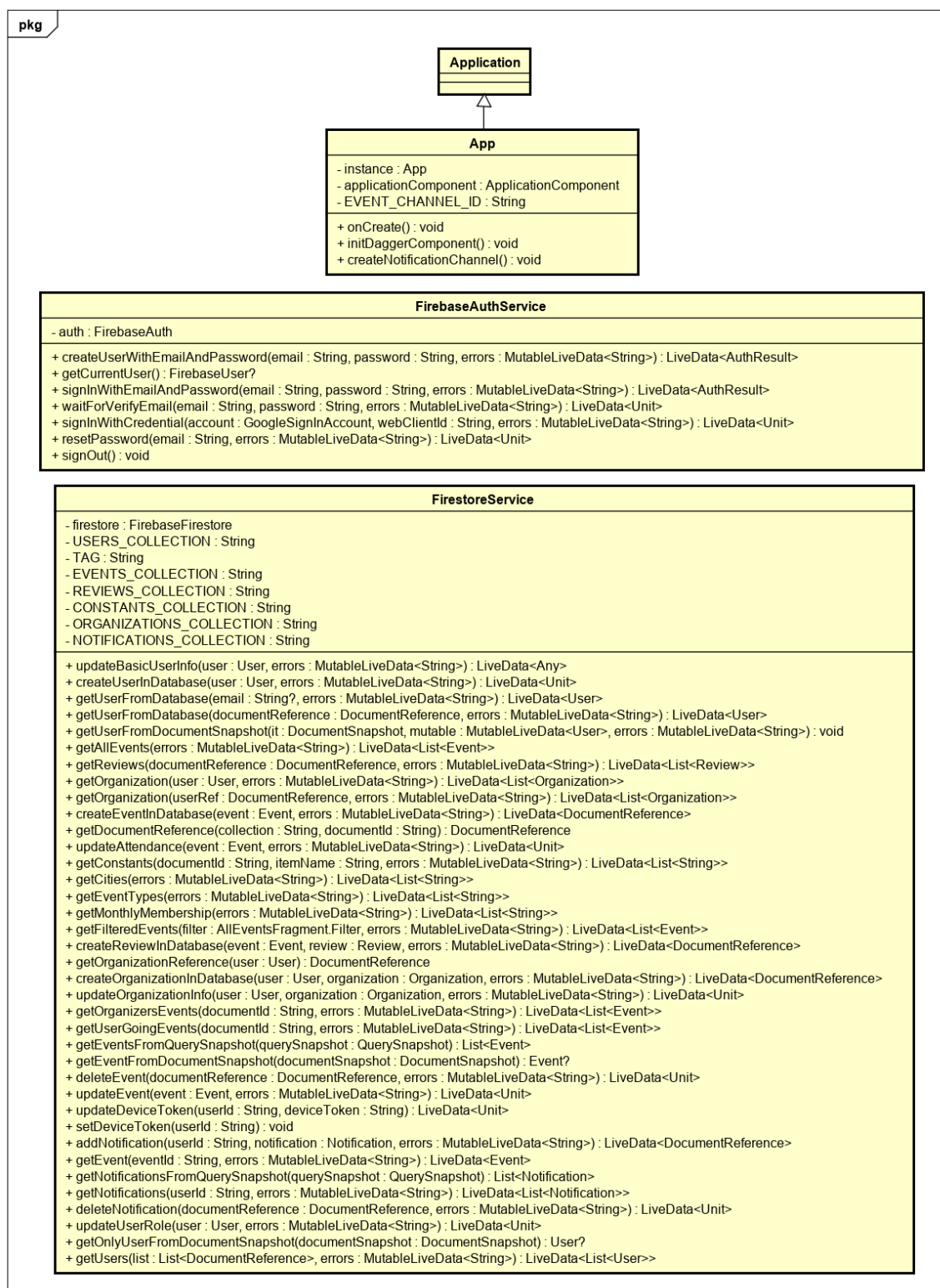


Slika 4.14: Dijagram razreda View - četvrti dio

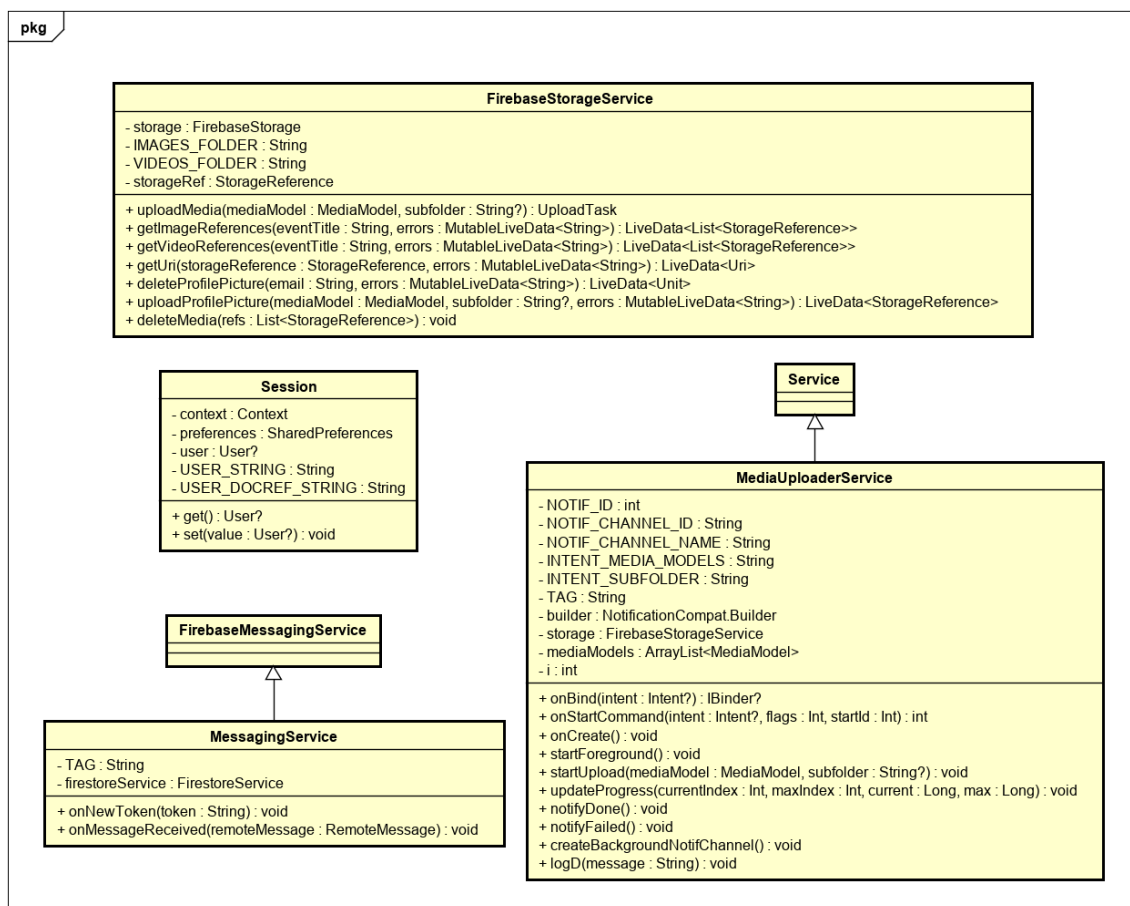


Slika 4.15: Dijagram razreda View - peti dio

Sve aktivnosti nasljeđuju klasu BaseActivity u kojoj se nalaze neke metode poput showMessage za prikazivanje poruka korisniku ili show/hideLoading za prikazivanje odnosno skrivanje poruke o učitavanju, logD/W za prikazivanje poruka u logu i slično. Ta klasa ima parametar koji mora nasljeđivati BaseViewModel i popratnu metodu koja inicijalizira varijablu viewModel preko koje se pristupa funkcionalnostima dodijeljenog ViewModela. To nas prisiljava da konzistentno koristimo arhitekturu MVVM, i, dodatno, ViewModele koji koriste dependency injection i varijablu errors za objavljivanje grešaka.



Slika 4.16: Dijagram razreda Utilites



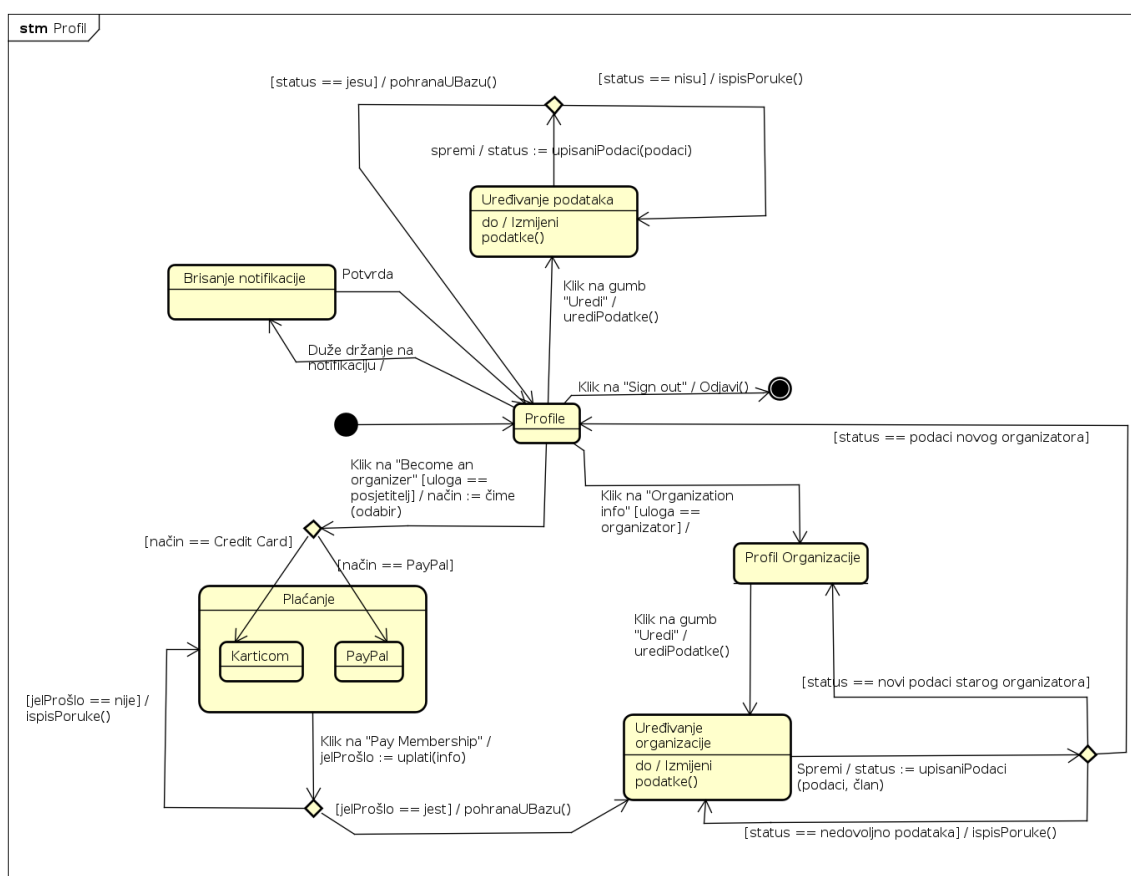
Slika 4.17: Dijagram razreda Utilites - drugi dio

U svakoj aplikaciji postoje klase koje se redovito koriste, a nisu eksplicitno dio neke arhitekture. U ovom slučaju, to su nekakvi "servisi" preko kojih pristupamo Firebaseu i Session preko kojeg spremamo podatke u podijeljenu memoriju uređaja. Podijeljena memorija pogodna je za spremanje podataka koji su često potrebni u aplikaciji. Klasa App ulazna je točka aplikacije i služi za inicijalizaciju dependency injectiona.



### 4.3 Dijagram stanja

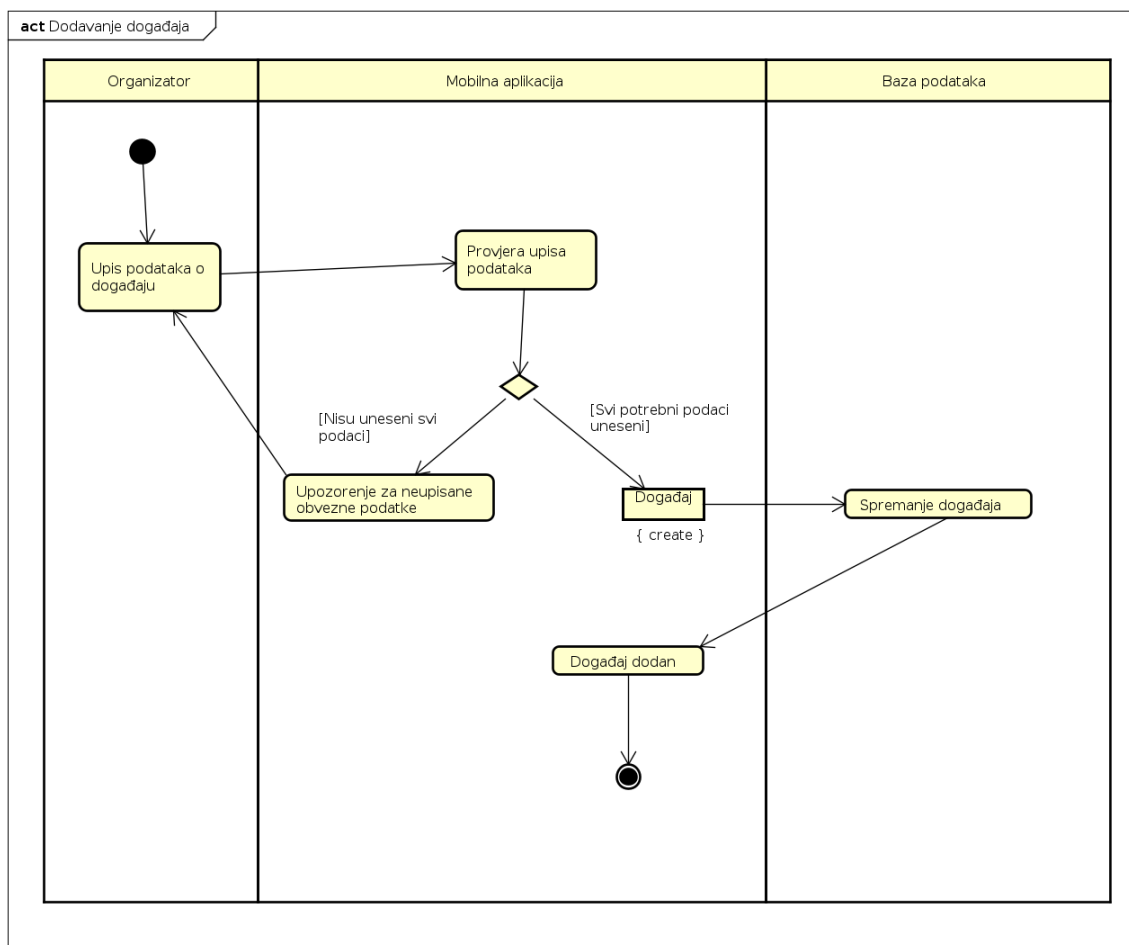
Na slici 4.18 prikazan je dijagram stanja za pregled profila registriranog korisnika. Nakon odabira pregleda svojeg profila, korisnik može vidjeti vlastite informacije i obavijesti o događajima. Obavijest se briše držanjem obavijesti koju želimo ukloniti, nakon čega se otvara skočni prozor na kojem potvrđujemo ili odustajemo od brisanja. Korisnik može urediti svoje podatke. Ako su podaci neispravni, korisnik je zadržan na ekranu za uređivanje podataka. Posjetitelj ima opciju postati organizator. Posjetitelj tada bira način plaćanja (kreditna kartica ili PayPal) i upisuje potrebne informacije. Ako su podaci ispravni, korisnik upisuje podatke o organizaciji i vraća se na pregled profila. Organizator može pogledati profil svoje organizacije. Može i mijenjati podatke o svojoj organizaciji te se vratiti na profil organizacije nakon pohrane promjena u bazu. Odabirom "Sign out" korisnik se može odjaviti iz aplikacije.



Slika 4.18: Dijagram stanja

## 4.4 Dijagram aktivnosti

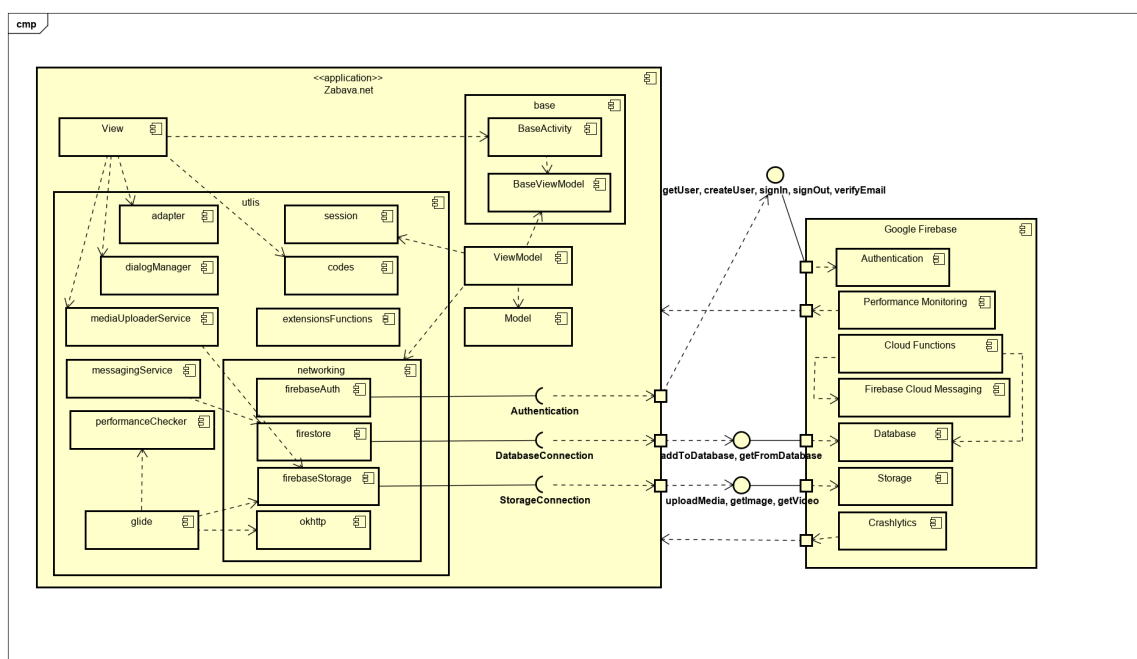
Na dijagramu 4.19 prikazan je proces dodavanja događaja. Organizator upisuje podatke o događaju. Mobilna aplikacija provjerava jesu li uneseni svi potrebni podaci. Ako nisu, aplikacija upozorava organizatora o nedostatku obaveznih informacija. Ako su uneseni svi potrebni podaci, aplikacija stvara događaj. Pri tome se novi događaj sprema u bazu podataka i dodaje u aplikaciju.



Slika 4.19: Dijagram aktivnosti

## 4.5 Dijagram komponenti

Na dijagramu 4.20 prikazan je dijagram komponenti naše aplikacije i Google Firebase-a koji koristimo. U komponenti Google Firebase prikazane su sve komponente koje koristimo od Google Firebase-a. Database nam služi za pohranu svih podataka dok nam Storage služi za pohranu slika i videa. Authenticate nam služi za autentifikaciju pojedinog korisnika. Cloud Functions koriste se prilikom događaja u drugim Firebase servisima, Firebase Cloud Messaging koristimo za vezu između poslužitelja i mobilnog uređaja da bismo mogli korisnicima slati obavijesti. Firebase Performance Monitoring koristimo da bismo vidjeli ponašanje aplikacije kod korisnika, za moguće unaprjeđenje aplikacije i otkrivanja nekih pogrešaka ako postoje. Aplikacija u sebi sadrži tri glavne komponente View, ViewModel i Model od kojih svaki radi svoj posao kako je to definirano u arhitekturi. Komponenta utlis u sebi sadrži sve komponente potrebne za rad aplikacija a nisu dio arhitekture. Konkretno komponenta networking služi za svu komunikaciju s bazom.



Slika 4.20: Dijagram komponenti

## 5. Implementacija i korisničko sučelje

### 5.1 Korištene tehnologije i alati

Komunikacija u timu realizirana je korištenjem aplikacije Slack<sup>1</sup>. Za izradu UML dijagrama korišten je alat Astah Professional<sup>2</sup>, a kao sustav za upravljanje izvornim kodom Git<sup>3</sup>. Udaljeni repozitorij projekta dostupan je na web platformi GitLab<sup>4</sup>.

Sva dokumentacija pisana je u alatu TEXstudio<sup>5</sup> prema zadanom predlošku.

Kao razvojno okruženje korišten je Android Studio<sup>6</sup> koji se najčešće koristi za izradu mobilnih Android aplikacija. Za izradu aplikacije mogu se koristiti dva jezika, Kotlin<sup>7</sup> i Java<sup>8</sup>. Mi smo odabrali Kotlin zbog veće prilagođenosti pisanju Android aplikacija.

Za testiranje smo koristili Appium<sup>9</sup> koji služi za automatiziranje testiranja, dok su testovi napisani u IntelliJ Idea<sup>10</sup> koristeći jezik Javu uz dodatne Selenium<sup>11</sup> biblioteke.

Baza podataka nalazi se na Google Firebase<sup>12</sup> kojega koristimo i za autentifikaciju korisnika.

---

<sup>1</sup><https://slack.com/intl/en-hr/>

<sup>2</sup><http://astah.net/editions/professional>

<sup>3</sup><https://git-scm.com/>

<sup>4</sup><https://gitlab.com>

<sup>5</sup><https://www.texstudio.org/>

<sup>6</sup><https://developer.android.com/studio>

<sup>7</sup><https://kotlinlang.org/>

<sup>8</sup><https://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html>

<sup>9</sup><http://appium.io/>

<sup>10</sup><https://www.jetbrains.com/idea/>

<sup>11</sup><https://selenium.dev/>

<sup>12</sup><https://firebase.google.com/>

## 5.2 Ispitivanje programskog rješenja

### 5.2.1 Ispitivanje komponenti

Ispitivanje komponenti provodi se s ciljem ispitivanja pojedinih metoda i klasa kod korištenja složenijih algoritama kako bismo se uvjerali daje li metoda ispravan i očekivan rezultat. Ispitivanje komponenti provodi se tako da se ispituju rubni slučajevi, što znači da se ispituje s ulaznim vrijednostima koje u pravilu nisu očekivane. Testira se kako se komponente ponašaju ako dobiju null vrijednost, krivi tip podataka ili, na primjer, ako metoda na ulazu očekuje String, kako se ponaša ako je ulaz prazan String ili unos neki drugi tip podatka.

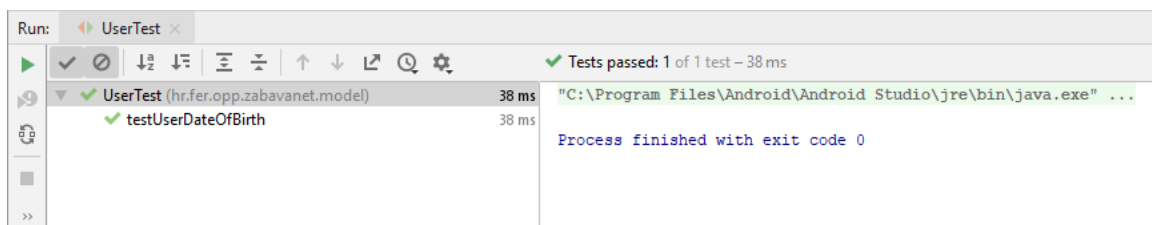
Prilikom testiranja klase User testirali smo gettere i settere za varijablu dateOfBirth, kao što je prikazano u kodu 5.1. Najprije se kreira varijabla user s potrebnim atributima. Prva dva testa provjeravaju je li se dateOfBirth postavio sam od sebe. Isto tako se provjerava i dateOfBirthLong koji predstavlja vrijednost varijable dateOfBirth koja je tipa Timestamp u sekundama. Zatim kreiramo varijablu timestamp koju pridružimo varijabli dateOfBirth. Očekuje se da se istovremeno postavi varijabla dateOfBirthLong na istu vrijednost koju sadrži timestamp.seconds. Također, pokušavamo postaviti vrijednost varijable na null i vidimo da se varijabla ne mijenja ukoliko je pokušamo postaviti na null. Na slici 5.1 vidimo da metoda uspješno prolazi test.

Kod 5.1: dateOfBirthTest

```
class UserTest {  
  
    @Test  
    fun testUserDateOfBirth() {  
        var user : User = User("david", "je",  
                                "bio@tu", "doktor", "bog")  
  
        assertNull(user.dateOfBirthLong)  
        assertNull(user.dateOfBirth)  
  
        var stamp = createTimestamp(2019, 11,  
                                    15, 12, 12)  
        user.dateOfBirth = stamp  
    }  
}
```

```
        assertEquals (user . dateOfBirthLong ,
                        stamp . seconds )

        user . dateOfBirth = null
        assertNotNull (user . dateOfBirth )
        assertEquals (user . dateOfBirthLong ,
                        stamp . seconds )
    }
}
```



Slika 5.1: Uspješan test datuma rođenja

Osim klase User, testirali smo i neke metode iz datoteke ExtensionFunctions. Testirali smo 4 metode koje su bitne za ispravan rad aplikacije.

U kodu 5.2 testira se metoda createTimestamp koja prima cjelobrojne vrijednosti godine, mjeseca, dana, sata i minute. Metoda toPattern iz Timestampa daje String koji sadrži format datuma zadan argumentom. Na slici 5.2 vidi se da metoda prolazi zadane testove.

Kod 5.2: createTimestamp

```
@Test
fun createTimestamp () {

    val stampTime = createTimestamp (
        year = 2019,
        month = 11,
        day = 15,
        hour = 12,
        minute = 12
```

```
)  
//12. je jer metoda koja se  
    koristi broji mjesece od 0  
assertEquals("12:12 15.12.2019" ,  
    stamptime.toPattern(null))  
assertEquals("12:12", createTimeStamp(2020,  
    0, 15, 12, 12).toPattern("HH:mm"))  
}
```

U kodu 5.3 testira se metoda koja vraća razliku u danima između dva predana Timestampa. Važno je pripaziti da metoda createTimeStamp broji mjesece od 0, pa tako 0 predstavlja siječanj, 1 veljaču, ..., 11 prosinac. Na slici 5.2 vidimo da metoda prolazi sve testove.

#### Kod 5.3: timeDifferenceDays

```
@Test  
fun timeDifferenceDays() {  
    assertEquals(2.0, timeDifferenceDays(  
        createTimeStamp(2019, 12, 15, 12, 12),  
        createTimeStamp(2019, 12, 17, 12, 12)), 0.1)  
    assertEquals(1.0, timeDifferenceDays(  
        createTimeStamp(2019, 11, 31, 12, 12),  
        createTimeStamp(2020, 0, 1, 12, 12)), 0.1)  
    assertEquals(1.0, timeDifferenceDays(  
        createTimeStamp(2020, 1, 29, 12, 12),  
        createTimeStamp(2020, 2, 1, 12, 12)), 0.1)  
    assertEquals(1.0, timeDifferenceDays(  
        createTimeStamp(2020, 4, 30, 12, 12),  
        createTimeStamp(2020, 5, 1, 12, 12)), 0.1)  
    assertEquals(1.0, timeDifferenceDays(  
        createTimeStamp(2020, 4, 20, 12, 12),  
        createTimeStamp(2020, 4, 21, 12, 12)), 0.1)  
    assertEquals(1.0, timeDifferenceDays(  
        createTimeStamp(2020, 5, 30, 12, 12),  
        createTimeStamp(2020, 6, 1, 12, 12)), 0.1)  
}
```

U kodu 5.4 testira se metoda koja zadani Timestamp pretvara u String koji sadrži zapis Timestampa u formatu zadanom u argumentu funkcije. Ako metodi predamo null, ona vraća datum oblika "HH:mm dd.MM.yyyy". Metodi predajemo različite Timestampove i uspoređujemo rezultat metode s očekivanim rezultatom (Stringom). Na slici 5.2 vidimo da metoda uspješno prolazi sve testove.

Kod 5.4: toPattern

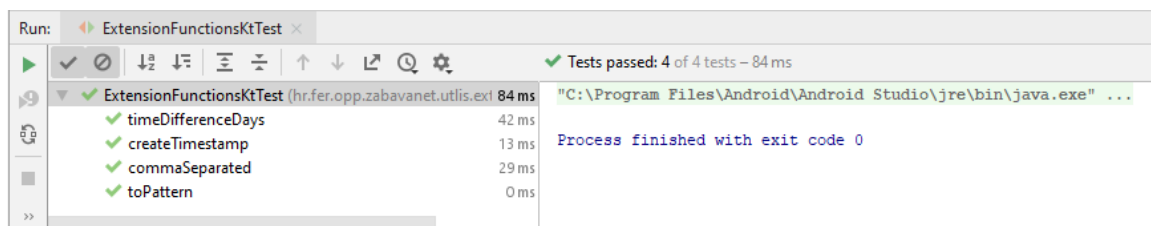
```
@Test
fun toPattern() {
    assertEquals("12:12 15.01.2020",
        createTimestamp(2020, 0, 15, 12, 12)
            .toPattern(null))
    assertEquals("12:12",
        createTimestamp(2020, 0, 15, 12, 12)
            .toPattern("HH:mm"))
    assertEquals("15.01.2020",
        createTimestamp(2020, 0, 15, 12, 12)
            .toPattern("dd.MM.yyyy"))
    assertEquals("2020.12:12",
        createTimestamp(2020, 0, 15, 12, 12)
            .toPattern("yyyy.HH:mm"))
}
```

U kodu 5.5 testira se metoda commaSeparated koja za zadanu listu kreira String koji sadrži sve vrijednosti liste odvojene zarezom. Metoda je testirana s brojevima, null vrijednošću i praznim Stringom. Na slici 5.2 vidi se da i ova metoda prolazi sve testove.

Kod 5.5: commaSeparated

```
@Test
fun commaSeparated() {
    assertEquals("1, 2, 3", listOf(1,2,3)
        .commaSeparated())
    assertEquals("null", listOf(null)
        .commaSeparated())
    assertEquals("", listOf(" ").commaSeparated())
}
```





Slika 5.2: Uspješan test dodatnih funkcija

## 5.2.2 Ispitivanje sustava

Za ispitivanje sustava koristili smo Appium i IntelliJ s korištenjem Selenium klijenta i standalone .jar datoteka te još nekoliko biblioteka. Ispitivanje smo proveli na dijelu aplikacije za dodavanje događaja. Provedeno je nekoliko testova kod kojih su ispitani granični slučajevi tog dijela aplikacije.

U prvom testu provjeravamo bez kojih je podataka moguće kreirati događaj. Najprije pokušamo kreirati događaj koji završava prije nego što počinje postavljanjem vremena završetka na manju vrijednost od vremena početka. Na to aplikacija ispisuje poruku na slici 5.3.

17:48

Event title:

Description:

City:

Exact address:

Event type:

Start time:

End time:

ADD IMAGES (0/10)

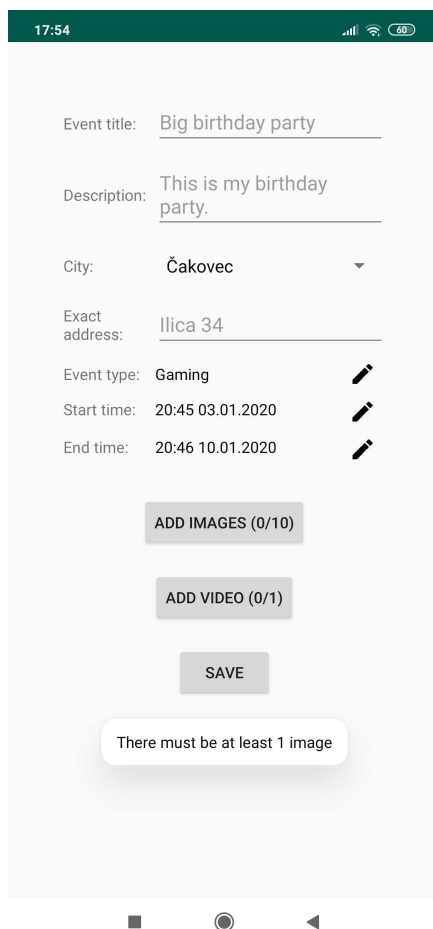
ADD VIDEO (0/1)

SAVE

End time must be greater than start time

Slika 5.3: Krivo vrijeme završetka

Nadalje, postavimo valjano vrijeme početka i završetka događaja te pokušamo spremiti ovakav događaj. Na to aplikacija ispisuje poruku na slici 5.4.



17:54

Event title: Big birthday party

Description: This is my birthday party.

City: Čakovec

Exact address: Ilica 34

Event type: Gaming

Start time: 20:45 03.01.2020

End time: 20:46 10.01.2020

ADD IMAGES (0/10)

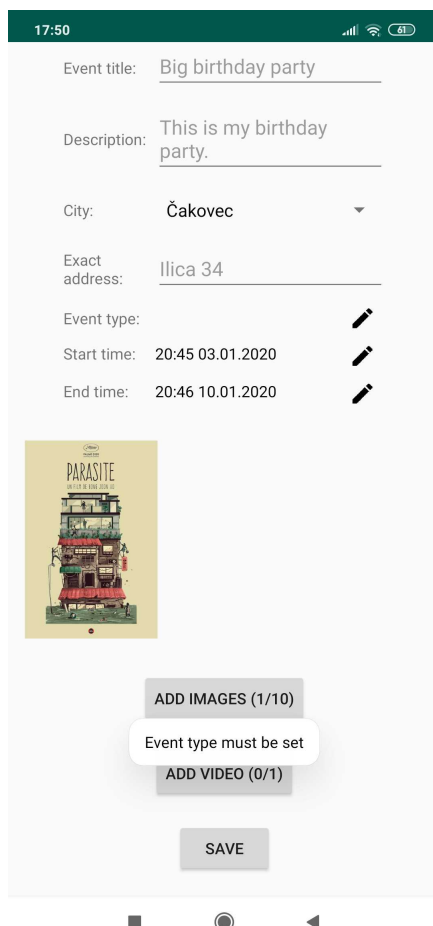
ADD VIDEO (0/1)

SAVE

There must be at least 1 image

Slika 5.4: Događaj bez slike

Sada dodajemo sliku tom događaju i pokušamo ga spremiti. Aplikacija ispisuje poruku prikazanu na slici 5.5.



Slika 5.5: Nema tipa događaja

Dodajemo tip događaja i pokušamo sada spremi događaj.

17:50

Event title: Big birthday party

Description: This is my birthday party.

City: Čakovec

Exact address: Ilica 34

Event type: Gaming

Start time: 20:45 03.01.2020

End time: 20:46 10.01.2020

PARASITE

ADD IMAGES (1/10)

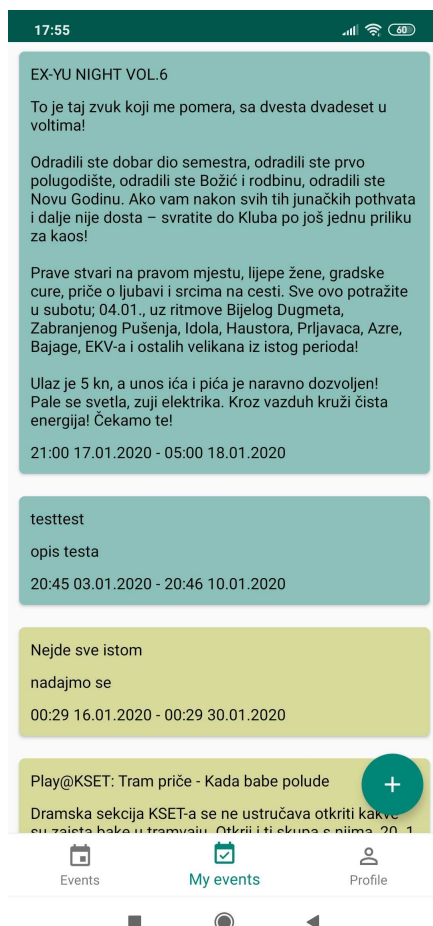
All fields are mandatory

ADD VIDEO (0/1)

SAVE

Slika 5.6: Nedostaje naziv događaja

Vidimo da još nismo dodali najbitnije, naziv, opis i adresu događaja. Dodajemo naziv, opis i adresu događaja i spremimo sada događaj.



Slika 5.7: Prikaz dodanog događaja

Na slici 5.7 vidi se da je dodan događaj "testtest" koji smo kreirali u ovom testu. Drugi provedeni test je test kod kojeg se pokušava dodati previše slika na događaj te nam sustav ispisa poruku kao što je prikazano na slici 5.8. Treći provedeni test je test kod kojeg se događaj uspješno kreira s dodanim svim obaveznim podacima. Svi kodovi testova sustava priloženi su u poglavlju Dodatak 2: Kodovi ispitivanja sustava.

17:51

Event title: Big birthday party

Description: This is my birthday party.

City: Čakovec

Exact address: Ilica 34

Event type: Gaming

Start time: 20:45 03.01.2020

End time: 20:46 10.01.2020

PARASITE

기생충

PROJEKCIJA VOĐA 领风者

ADD IMAGES (10/10)

Can't add more than 10 images

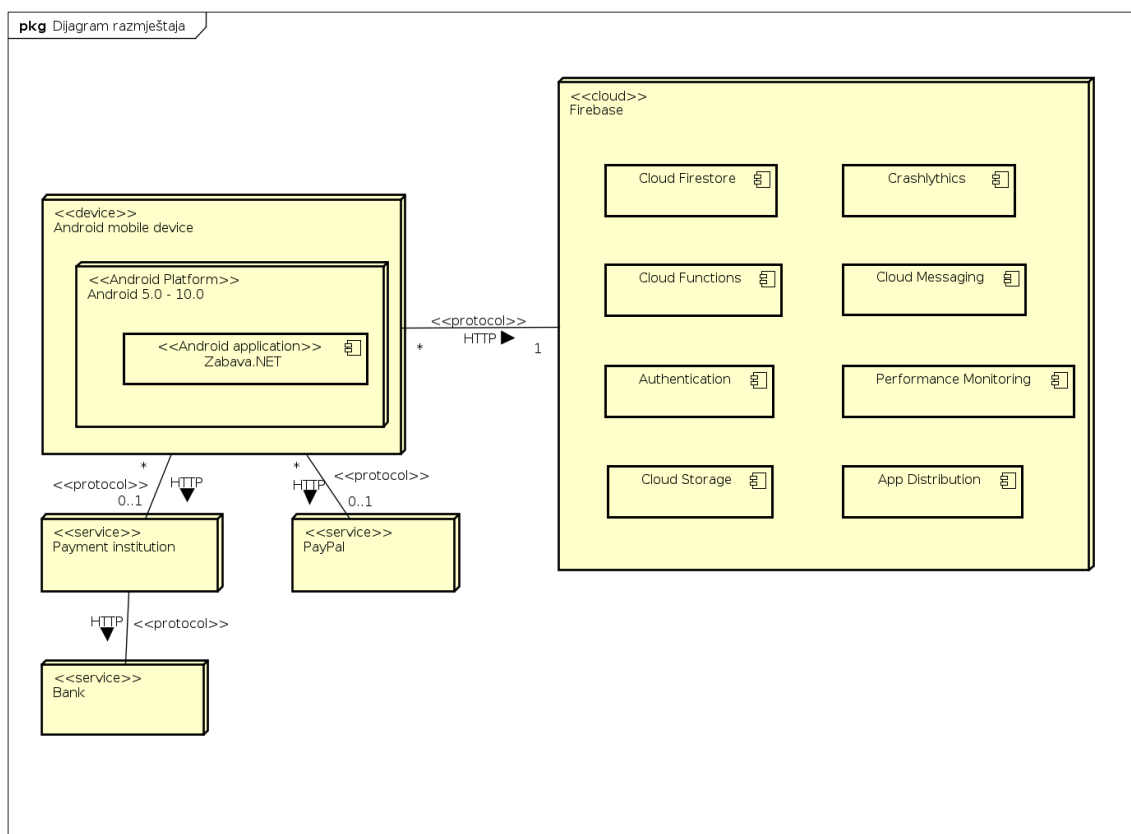
ADD VIDEO (0/1)

SAVE

Slika 5.8: Dodano previše slika

### 5.3 Dijagram razmještaja

Dijagram na slici 5.9 opisuje programsku potporu korištenu u implementaciji sustava. Aplikaciji Zabava.net pristupa se na Android mobilnom uređaju koji koristi Android platformu verzije 5.0 – 10.0. Plaćanje unutar aplikacije odvija se uz pomoć usluge PayPal i kartičnim plaćanjem preko platnih institucija koje su povezane s bankama. Komunikacija se odvija putem protokola HTTP. Platforma Firebase pruža aplikaciji usluge svojih komponenata. Komponente koje aplikacija koristi su Cloud Firestore, Cloud Functions, Cloud Messaging, Crashlytics, Authentication, Performance Monitoring, Cloud Storage i App Distribution. Komunikacija između mobilnog uređaja korisnika i Firebasea odvija se putem protokola HTTP.



Slika 5.9: Dijagram razmještaja



## 5.4 Upute za puštanje u pogon

### 5.4.1 Kreiranje baze podataka

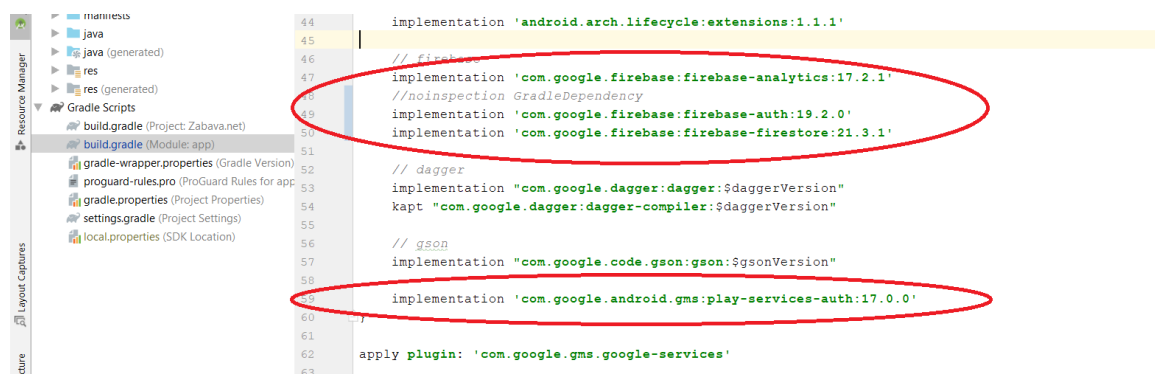
Potrebno je kreirati Google račun i zatim otići na Google Firebase konzolu. Klikom na tipku "Add project" kreiramo novi projekt. Imenujemo ga i završavamo kreiranje baze podataka.

### 5.4.2 Spajanje s bazom podataka

Na Google Firebase konzoli bismo spajanje s Androidom. Popunimo polje "Android package name" i kliknemo tipku "Register app". Nakon toga Firebase kreira datoteku "google-services.json" koju je potrebno preuzeti i smjestiti u direktorij "app/" naše aplikacije. Nakon toga potrebno je podesiti "build.gradle" datoteke da koriste preuzetu datoteku na način da se doda dio koda kao na slikama 5.10 i 5.11.



Slika 5.10: Datoteka Project



Slika 5.11: Datoteka Module

### 5.4.3 Punjenje baze podataka

Baza podataka puni se tako da joj se kolekcija `Map<String, Any?>` šalje izravno iz koda. Primjer koda za punjenje baze podataka dan je na slici 5.12.

```
fun createEventInDatabase(
    event: Event,
    errors: MutableLiveData<String>
): LiveData<DocumentReference> {
    val mutable = MutableLiveData<DocumentReference>()

    firestore.collection(EVENTS_COLLECTION).add(event.asHashMap())
        .addOnSuccessListener { it: DocumentReference!
            mutable.postValue(it)
        }.addOnFailureListener { it: Exception
            errors.postValue(it.message)
        }

    return mutable
}
```

Slika 5.12: Primjer punjenja baze

### 5.4.4 Pokretanje Android aplikacije

Potrebno je s GitLab repozitorija projekta na mobilni uređaj preuzeti datoteku "app-release.apk". Njenim pokretanjem započinje instalacija same aplikacije. Nakon završene instalacije, aplikacija se pokreće klikom na ikonu.

## 6. Zaključak i budući rad

Naša grupa imala je zadatak izraditi mobilnu aplikaciju za oglašavanje zabavnih događanja. Osim same izrade aplikacije, puno se pažnje posvetilo izradi kvalitetne projektne dokumentacije. Projekt je bio podijeljen u dvije faze kroz 15 tjedana.

Prva faza izrade projekta trajala je 9 tjedana. Uključivala je okupljanje tima, dogovor načina komunikacije, dodjelu projektnog zadatka, rad na dokumentaciji i implementaciju osnovnih funkcionalnosti aplikacije. Tim je kroz redovite sastanke i komunikaciju preko Slacka podijelio zadatke. Specificirani su zahtjevi sustava, definirani su obrasci uporabe te su izrađeni dijagrami obrazaca uporabe i sekvencijski dijagrami. Odabrana je MVVM arhitektura, programski jezik Kotlin, razvojno okruženje Android Studio i razvojna platforma Firebase. Izrađeni su dijagram baze podataka i dijagrami razreda. Dobra provedba prve faze projekta bila je ključna za izradu kvalitetne aplikacije. Pomoću izrađenih dijagrama jasnije je prikazano što je sve potrebno implementirati za ispravan rad aplikacije.

Druga faza projekta trajala je 6 tjedana. Unatoč kraćem vremenu rada u drugoj fazi, ona je bila puno zahtjevnija od prve. Od članova tima tražilo se da upoznaju tehnologije koje još nisu koristili i stečeno znanje primijene u izradi aplikacije. Nakon implementacije odabranog rješenja, provedeno je testiranje aplikacije. Iako testovima nije moguće dokazati da u aplikaciji ne postoji problem, vrlo ih je važno provesti. Ako aplikacija prođe sve izrađene testove, velika je vjerojatnost da će ispravno raditi i nakon puštanja u pogon. Kako nije moguće testirati aplikaciju u svim uvjetima, zadovoljili smo se činjenicom da aplikacija prolazi sve testove koje smo nad njom proveli. Osim programiranja aplikacije, izrađeni su dijagram stanja, dijagram aktivnosti, dijagram komponenti i dijagram razmještaja. Dovršena je dokumentacija cijelog projekta.

Rad na projektu bio je iznimno koristan za sve članove tima. Naučeno je što znači rad u timu i koliko je važna dobra organizacija i podjela posla. Shvaćena je važnost kvalitetne dokumentacije projekta i testiranja izrađene aplikacije. Također, tim se upoznao s brojnim tehnologijama s kojima se prije nije imao prilike susresti. Vjerujemo da će nam stečeno znanje i iskustvo biti od velike koristi kako za studij tako i za rad na stvarnim projektima.

Za brže i kvalitetnije ostvarenje projekta najvažnije je iskustvo. Kako se tim tek upoznao s radom u određenim alatima i podjelom posla, izrada aplikacije i dokumentacije bila je nešto sporija. Sada, kad tim ima iskustva, vjerujemo da bi izrada neke nove aplikacije bila puno brža.

Aplikacija Zabava.NET mogla bi se nadograditi na brojne načine. Moguće je ugraditi dodatne funkcionalnosti ili potpuno prenamijeniti aplikaciju za neku drugu svrhu. Time bismo mogli privući više korisnika i povećati uspješnost aplikacije.

# Popis literature

1. Oblikovanje programske potpore, FER ZEMRIS, <http://www.fer.hr/predmet/opp>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/books/SE>
5. The Unified Modeling Language, <https://www.uml-diagrams.org/>
6. Astah Community, <http://astah.net/editions/uml-new>

# Dodatak: Prikaz aktivnosti grupe

## Dnevnik sastajanja

### 1. sastanak

- Datum: 9. listopada 2019.
- Prisustvovali: D.Gorup, M.Harbaš, H.Ivandić, I.Martinović, E.Popović, F.Pranklin, A.Samodol
- Teme sastanka:
  - sastanak s asistenticom
  - objašnjenje zadatka

### 2. sastanak

- Datum: 14. listopada 2019.
- Prisustvovali: D.Gorup, M.Harbaš, H.Ivandić, I.Martinović, E.Popović, F.Pranklin, A.Samodol
- Teme sastanka:
  - definiranje funkcionalnih zahtjeva
  - diskusija o bazi podataka

### 3. sastanak

- Datum: 21. listopada 2019.
- Prisustvovali: D.Gorup, M.Harbaš, H.Ivandić, I.Martinović, E.Popović, F.Pranklin, A.Samodol
- Teme sastanka:
  - definiranje obrazaca uporabe
  - podjela zadataka(programiranje + nastavak dokumentacije)

### 4. sastanak

- Datum: 24. listopada 2019.
- Prisustvovali: D.Gorup, M.Harbaš, H.Ivandić, I.Martinović, E.Popović, F.Pranklin, A.Samodol
- Teme sastanka:
  - sastanak s asistenticom-prodiskutirati dosadašnji rad(fokus na opi-

sima obrazaca uporabe i login aplikacije)

5. sastanak

- Datum: 5. studenoga 2019.
- Prisustvovali: D.Gorup, H.Ivandić, I.Martinović, E.Popović, F.Pranklin, A.Samodol
- Teme sastanka:
  - podjela zadataka (sekvencijski dijagrami + nefunkcionalni zahtjevi)
  - opis preostalih obrazaca uporabe

6. sastanak

- Datum: 7. studenoga 2019.
- Prisustvovali: D.Gorup, I.Martinović, E.Popović, A.Samodol
- Teme sastanka:
  - sastanak s demosom - komentiranje dijagrama obrazaca, sekvencijskih dijagrama i baze podataka

7. sastanak

- Datum: 11. studenoga 2019.
- Prisustvovali: D.Gorup, M.Harbaš, H.Ivandić, I.Martinović, E.Popović, F.Pranklin, A.Samodol
- Teme sastanka:
  - osmišljavanje dijagrama razreda
  - diskutiranje o obrascima uporabe i sekvencijskim dijagramima

8. sastanak

- Datum: 14. studenoga 2019.
- Prisustvovali: D.Gorup, M.Harbaš, H.Ivandić, I.Martinović, E.Popović, F.Pranklin, A.Samodol
- Teme sastanka:
  - sastanak s asistenticom - evaluacija dosadašnjeg rada

9. sastanak

- Datum: 2. prosinca 2019.
- Prisustvovali: D.Gorup, M.Harbaš, H.Ivandić, I.Martinović, E.Popović, F.Pranklin, A.Samodol
- Teme sastanka:
  - komentiranje dokumentacije
  - podjela zadataka (testiranje + dijagram stanja + dijagram aktivnosti + dijagram komponenti)

## 10. sastanak

- Datum: 8. siječnja 2019.
- Prisustvovali: D.Gorup, M.Harbaš, H.Ivandić, I.Martinović, E.Popović, F.Pranklin, A.Samodol
- Teme sastanka:
  - aktualizacija dokumentacije
  - podjela zadataka (implementacija i korisničko sučelje)

## 11. sastanak

- Datum: 9. siječnja 2019.
- Prisustvovali: D.Gorup, M.Harbaš, H.Ivandić, I.Martinović, E.Popović, F.Pranklin, A.Samodol
- Teme sastanka:
  - sastanak s demosom - demonstracija alfa inačice

## 12. sastanak

- Datum: 16. siječnja 2019.
- Prisustvovali: D.Gorup, I.Martinović, E.Popović
- Teme sastanka:
  - sastanak s asistenticom - razjašnjenje pitanja vezana uz dijagrame



## Tablica aktivnosti

Tablica 6.1: Tablica aktivnosti

	Ivona Martinović	Filip Franklin	Ema Popović	Miro Harbaš	Hana Ivandić	David Gorup	Andrija Samodol
Upravljanje projektom	6	0.5	-	-	-	-	-
Opis projektnog zadatka	0.3	-	-	-	3	-	-
Funkcionalni zahtjevi	0.2	-	2	0.2	-	0.2	0.2
Opis pojedinih obrazaca	1.3	-	0.5	3	1	-	1
Dijagram obrazaca	0.2	-	-	-	-	-	3
Sekvencijski dijagrami	0.5	-	2	1.5	1.5	-	2
Opis ostalih zahtjeva	-	-	1.5	-	-	-	-
Arhitektura i dizajn sustava	3	1.5	-	1.5	1.3	-	-
Baza podataka	2	1	-	-	-	2	-
Dijagram razreda	0.5	2	-	-	2	2.5	4.5
Dijagram stanja	0.5	-	2	-	-	-	1
Dijagram aktivnosti	-	-	1	-	-	-	1.5
Dijagram komponenti	-	-	1	-	-	2	1
Korištene tehnologije i alati	-	1.5	-	1	1	1	-
Ispitivanje programskog rješenja	-	-	-	-	1	1.5	-
Dijagram razmještaja	0.5	-	1	-	2	-	1
Upute za puštanje u pogon	-	-	-	-	1.3	1.5	-
Dnevnik sastajanja	-	-	2	-	-	-	-
Zaključak i budući rad	-	-	-	-	2.7	-	-
Popis literature	0.5	-	0.5	-	-	-	-
Dnevnik promjena	0.5	-	0.5	-	1.5	-	1
Tablica aktivnosti	0.5	-	1.5	-	-	-	-
Lektoriranje i strukturiranje	-	-	2	-	2.5	-	-
Izrada baze podataka	-	-	-	-	-	1.5	-

	Ivona Martinović	Filip Pranklin	Ema Popović	Miro Harbaš	Hana Ivandić	David Gorup	Andrija Samodol
Pristupanje bazi podataka	-	-	-	-	-	1	-
Spajanje s Firebaseom	-	0.5	-	-	-	0.5	-
Autentifikacija korisnika	-	0.5	-	-	-	1	-
Izrada login ekrana	0.5	2	-	-	-	-	-
Izrada registration ekrana	-	1.5	-	-	-	-	-
Izrada basicInfo ekrana	3	-	-	-	-	-	-
Izrada home ekrana	-	0.5	-	-	-	-	-
Izrada verification ekrana	-	-	-	-	-	1	-
Izrada razreda potrebnih za ostvarivanje arhitekture	-	2	-	-	-	-	-
Izrada payment ekrana	2	-	-	-	-	-	-
Izrada profile ekrana	2	-	-	-	-	-	-
Izrada addEvent ekrana	-	2	-	-	-	-	-
Izrada addReview ekrana	-	2	-	-	-	-	-
Izrada allEvents ekrana	-	3	-	-	-	-	-
Izrada conformationDialog ekrana	-	2	-	-	-	-	-
Izrada eventDetails ekrana	-	1.5	-	-	-	-	-
Izrada myEvents ekrana	-	3	-	-	-	-	-
Izrada editOrganization ekrana	2	-	-	-	-	-	-
Izrada organizationProfile ekrana	2	-	-	-	-	-	-
Izrada splash ekrana	-	1	-	-	-	-	-
Izrada dodatnih razreda	1	6	-	-	-	-	-
Dizajn aplikacije	-	-	-	9.5	-	-	-
Testiranje aplikacije	-	-	-	-	-	4.5	-

## Dijagrami pregleda promjena



Slika 6.1: Dijagram pregleda promjena

## Dodatak 2: Kodovi ispitivanja sustava

Kod 6.1: AllTheMessages Code

```
public class Automation {

    public static void main(String args[]) throws
        MalformedURLException, InterruptedException {

        //setup

        DesiredCapabilities dc = new DesiredCapabilities();

        dc.setCapability(MobileCapabilityType.DEVICE_NAME,
            "emulator-5554");
        dc.setCapability("platformName", "android");
        dc.setCapability("appPackage", "hr.fer.opp.zabavanet");
        dc.setCapability("appActivity",
            ".view.splash.SplashActivity");

        AndroidDriver<AndroidElement> ad =
            new AndroidDriver<AndroidElement>
                (new URL("http://127.0.0.1:4723/wd/hub"),
                 dc);

        //login

        MobileElement el1 = (MobileElement)
            ad.findElementById("hr.fer.opp.zabavanet
                :id/text_email");
        el1.sendKeys("dgb.extra000@gmail.com");
        MobileElement el2 = (MobileElement)
            ad.findElementById("hr.fer.opp.zabavanet
```

```
                :id/text_password");
el2.sendKeys("budala555");
MobileElement el3 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_login");
el3.click();
Thread.sleep(3000);

//start creating an event

MobileElement el4 = (MobileElement)
    ad.findElementByAccessibilityId("My events");
el4.click();
MobileElement el5 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/fab_add_event");
el5.click();
Thread.sleep(1000);

//add start time

MobileElement el6 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_edit_start_time");
el6.click();
MobileElement el7 = (MobileElement)
    ad.findElementById("android:id/button1");
el7.click();
Thread.sleep(500);
MobileElement el8 = (MobileElement)
    ad.findElementById("android:id/button1");
el8.click();

//what happens if we try putting lower date as end
    time than the start time
```

```
//add end time

MobileElement el9 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_edit_end_time");
el9.click();
Thread.sleep(500);
MobileElement el10 = (MobileElement)
    ad.findElementByAccessibilityId("02 January 2020");
el10.click();
MobileElement el11 = (MobileElement)
    ad.findElementById("android:id/button1");
el11.click();
MobileElement el12 = (MobileElement)
    ad.findElementById("android:id/button1");
el12.click();
//we get the message on the screen that we can't do this

//we fill in the end time

MobileElement el13 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_edit_end_time");
el13.click();
Thread.sleep(500);
MobileElement el14 = (MobileElement)
    ad.findElementByAccessibilityId("31 January 2020");
el14.click();
MobileElement el15 = (MobileElement)
    ad.findElementById("android:id/button1");
el15.click();
MobileElement el16 = (MobileElement)
    ad.findElementById("android:id/button1");
el16.click();
```

```
//we try saving the event with just start time and end time

MobileElement el17 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_save_event");
el17.click();

//we also need at least one picture

MobileElement el18 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el18.click();
Thread.sleep(500);
MobileElement el19 = (MobileElement)
    ad.findElementById("com.android.documentsui
        :id/icon_thumb");
el19.click();

//we scroll down to see the save button

Thread.sleep(1500);
TouchAction touchAction1 = new TouchAction(ad);
touchAction1.press(ElementOption.element(el13))
    .waitAction(WaitOptions
        .waitOptions(Duration.ofMillis(500)))
        .moveTo(ElementOption.element(el6))
            .release().perform();

//we try saving the event

MobileElement el20 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
```

```
                :id/btn_save_event");
el20.click();

//we also need to have event type

MobileElement el21 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_edit_event_types");
el21.click();
MobileElement el22 = (MobileElement)
    ad.findElementByXPath("/hierarchy/android.widget
        .FrameLayout/" + "android.widget.FrameLayout/
        android.widget.FrameLayout/android.widget
        .LinearLayout/" + "android.widget.FrameLayout
        /android.widget.ListView
        /android.widget.CheckedTextView[3]");
el22.click();
MobileElement el23 = (MobileElement)
    ad.findElementById("android:id/button1");
el23.click();

//we try saving the event
Thread.sleep(500);
MobileElement el24 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_save_event");
el24.click();

//add exact address
MobileElement el50 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/text_event_address");
el50.sendKeys("Iz grada");

// we need to fill all fields , we add event title
```



and description

```
MobileElement el25 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/text_event_title");
el25.sendKeys("testtest");
MobileElement el26 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/text_event_description");
el26.sendKeys("opis testa");

//we save the event now

MobileElement el27 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_save_event");
el27.click();
}
}
```

## Kod 6.2: TooManyPictures Code

```
public class TooManyPictures {

    public static void main(String args[]) throws
        MalformedURLException, InterruptedException {

        //setup

        DesiredCapabilities dc = new DesiredCapabilities();

        dc.setCapability(MobileCapabilityType.DEVICE_NAME,
            "emulator-5554");
        dc.setCapability("platformName", "android");
        dc.setCapability("appPackage", "hr.fer.opp.zabavanet");
        dc.setCapability("appActivity",
            ".view.splash.SplashActivity");

        AndroidDriver<AndroidElement> ad =
            new AndroidDriver<AndroidElement>
                (new URL("http://127.0.0.1:4723/wd/hub"),
                 dc);

        //login

        MobileElement el1 = (MobileElement)
            ad.findElementById("hr.fer.opp.zabavanet
                :id/text_email");
        el1.sendKeys("dgb.extra000@gmail.com");
        MobileElement el2 = (MobileElement)
            ad.findElementById("hr.fer.opp.zabavanet
                :id/text_password");
        el2.sendKeys("budala555");
        MobileElement el3 = (MobileElement)
            ad.findElementById("hr.fer.opp.zabavanet
                :id/btn_login");
```

```
el3.click();
Thread.sleep(3000);

//start creating an event

MobileElement el4 = (MobileElement)
    ad.findElementByAccessibilityId("My events");
el4.click();
MobileElement el5 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/fab_add_event");
el5.click();
Thread.sleep(1000);

//let's add 10 pictures

MobileElement el6 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el6.click();
Thread.sleep(500);
MobileElement el7 = (MobileElement)
    ad.findElementByXPath("/hierarchy/android.widget
        .FrameLayout/" +
        "android.widget.FrameLayout/android.widget
        .FrameLayout/android.view.ViewGroup/" +
        "android.support.v4.widget.DrawerLayout/android
        .widget.LinearLayout/android.widget.FrameLayout/"
        + "android.widget.LinearLayout/android.widget
        .FrameLayout/android.widget.LinearLayout/" +
        "android.view.ViewGroup/android.support.v7.widget
        .RecyclerView/android.widget.LinearLayout[1]/"
        + "android.widget.RelativeLayout/android
        .widget.FrameLayout/android.widget
        .ImageView");
```

```
el7.click();
Thread.sleep(1000);
MobileElement el8 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el8.click();
Thread.sleep(500);
MobileElement el9 = (MobileElement)
    ad.findElementByXPath("/hierarchy/android.widget.
        FrameLayout/" +
        "android.widget.FrameLayout/android.widget
        .FrameLayout/android.view.ViewGroup/" +
        "android.support.v4.widget.DrawerLayout/android
        .widget.LinearLayout/android.widget.FrameLayout
        /" + "android.widget.LinearLayout/android.widget
        .FrameLayout/android.widget.LinearLayout/" +
        "android.view.ViewGroup/android.support.v7.widget
        .RecyclerView/android.widget.LinearLayout[2]/" +
        "android.widget.RelativeLayout/android.widget.
        FrameLayout/android.widget.ImageView[1]");
el9.click();
Thread.sleep(1000);
MobileElement el10 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el10.click();
Thread.sleep(500);
MobileElement el11 = (MobileElement)
    ad.findElementByXPath("/hierarchy/android.widget.
        FrameLayout/" +
        "android.widget.FrameLayout/android.widget
        .FrameLayout/android.view.ViewGroup/" +
        "android.support.v4.widget.DrawerLayout/android
        .widget
        .LinearLayout/android.widget.FrameLayout/" +
```

```
        "android.widget.LinearLayout/android.widget
        .FrameLayout/android.widget.LinearLayout/" +
        "android.view.ViewGroup/android.support.v7.widget
        .RecyclerView/android.widget.LinearLayout[1]/" +
        "android.widget.RelativeLayout/android.widget
        .FrameLayout/android.widget.ImageView[1]");
el11.click();
Thread.sleep(1000);
MobileElement el12 = (MobileElement)
        ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el12.click();
Thread.sleep(500);
MobileElement el13 = (MobileElement)
        ad.findElementByXPath("/hierarchy/android.widget
        .FrameLayout/" +
        "android.widget.FrameLayout/android.widget
        .FrameLayout/android.view.ViewGroup/" +
        "android.support.v4.widget.DrawerLayout
        /android.widget.LinearLayout/android.widget
        .FrameLayout/" +
        "android.widget.LinearLayout/android.widget
        .FrameLayout
        /android.widget.LinearLayout/" +
        "android.view.ViewGroup/android.support.v7.widget
        .RecyclerView/android.widget.LinearLayout[1]/" +
        "android.widget.RelativeLayout/android.widget
        .FrameLayout/android.widget.ImageView[1]");
el13.click();
Thread.sleep(1000);
MobileElement el14 = (MobileElement)
        ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el14.click();
Thread.sleep(500);
```

```
MobileElement el15 = (MobileElement)
    ad.findElementByXPath("/hierarchy/android.widget
        .FrameLayout/" +
        "android.widget.FrameLayout/android.widget
        .FrameLayout/android.view.ViewGroup/" +
        "android.support.v4.widget.DrawerLayout
        /android.widget.LinearLayout/android.widget
        .FrameLayout/" + "android.widget.LinearLayout
        /android.widget.FrameLayout/android.widget
        .LinearLayout/" +
        "android.view.ViewGroup/android.support.v7.widget
        .RecyclerView/android.widget.LinearLayout[4]/" +
        "android.widget.RelativeLayout/android.widget
        .FrameLayout/android.widget.ImageView[1]");
el15.click();
Thread.sleep(1000);
MobileElement el16 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el16.click();
Thread.sleep(500);
MobileElement el17 = (MobileElement)
    ad.findElementByXPath("/hierarchy/android.widget
        .FrameLayout/" +
        "android.widget.FrameLayout/android.widget.
        FrameLayout/android.view.ViewGroup/" +
        "android.support.v4.widget.DrawerLayout/
        android.widget.LinearLayout/android.widget
        .FrameLayout/" +
        "android.widget.LinearLayout/android.widget
        .FrameLayout/android.widget.LinearLayout/" +
        "android.view.ViewGroup/android.support.v7
        .widget.RecyclerView/android.widget.LinearLayout[3]/"
    + "android.widget.RelativeLayout/android.widget
        .FrameLayout/android.widget.ImageView[1]");
```

```
el17.click();
Thread.sleep(1000);
MobileElement el18 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el18.click();
Thread.sleep(500);
MobileElement el19 = (MobileElement)
    ad.findElementByXPath("/hierarchy/android.widget
        .FrameLayout/" + "android.widget.FrameLayout
        /android.widget.FrameLayout
        /android.view.ViewGroup/" +
        "android.support.v4.widget.DrawerLayout
        /android.widget.LinearLayout/android.widget
        .FrameLayout/" + "android.widget.LinearLayout
        /android.widget.FrameLayout
        /android.widget.LinearLayout/" +
        "android.view.ViewGroup/android.support.v7.widget
        .RecyclerView/android.widget.LinearLayout[4]/" +
        "android.widget.RelativeLayout/android.widget
        .FrameLayout/android.widget.ImageView[1]");
el19.click();
Thread.sleep(1000);
MobileElement el20 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el20.click();
Thread.sleep(500);
MobileElement el21 = (MobileElement)
    ad.findElementByXPath("/hierarchy/android.widget
        .FrameLayout/" +
        "android.widget.FrameLayout/android.widget
        .FrameLayout/android.view.ViewGroup/" +
        "android.support.v4.widget.DrawerLayout/android
        .widget.LinearLayout/android.widget.FrameLayout/" +
```

```
        "android.widget.LinearLayout/android.widget
        .FrameLayout/android.widget.LinearLayout/" +
        "android.view.ViewGroup/android.support.v7
        .widget.RecyclerView/android.widget.LinearLayout[1]/"
        + "android.widget.RelativeLayout/android
        .widget.FrameLayout/android.widget.ImageView[1]");
el21.click();
Thread.sleep(1000);
MobileElement el22 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el22.click();
Thread.sleep(500);
MobileElement el23 = (MobileElement)
    ad.findElementByXPath("/hierarchy/android.widget
        .FrameLayout/" +
        "android.widget.FrameLayout/android.widget
        .FrameLayout/android.view.ViewGroup/" +
        "android.support.v4.widget.DrawerLayout/android
        .widget.LinearLayout/" +
        "android.widget.FrameLayout/android.widget
        .LinearLayout/android.widget.FrameLayout/" +
        "android.widget.LinearLayout/android.view
        .ViewGroup/android.support.v7.widget.RecyclerView/" +
        "android.widget.LinearLayout[3]/android.widget
        .RelativeLayout/android.widget.FrameLayout/" +
        "android.widget.ImageView[1]");
el23.click();
Thread.sleep(1000);
MobileElement el24 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el24.click();
Thread.sleep(500);
MobileElement el25 = (MobileElement)
```



```
ad.findElementByXPath("/hierarchy/android.widget
    .FrameLayout/" +
    "android.widget.FrameLayout/android.widget
    .FrameLayout/android.view.ViewGroup/" +
    "android.support.v4.widget.DrawerLayout/android
    .widget.LinearLayout/android.widget.FrameLayout/" +
    "android.widget.LinearLayout/android.widget
    .FrameLayout/android.widget.LinearLayout/" +
    "android.view.ViewGroup/android.support.v7
    .widget.RecyclerView/android.widget
    .LinearLayout[3]/" +
    "android.widget.RelativeLayout/android.widget
    .FrameLayout/android.widget.ImageView[1]");
el25.click();
Thread.sleep(1000);

//now let's try adding another one

MobileElement el26 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el26.click();
Thread.sleep(500);
MobileElement el27 = (MobileElement)
    ad.findElementByXPath("/hierarchy/android.widget
    .FrameLayout/" +
    "android.widget.FrameLayout/android.widget
    .FrameLayout/android.view.ViewGroup/" +
    "android.support.v4.widget.DrawerLayout/android
    .widget.LinearLayout/android.widget.FrameLayout/" +
    "android.widget.LinearLayout/android.widget
    .FrameLayout/android.widget.LinearLayout/" +
    "android.view.ViewGroup/android.support.v7
    .widget.RecyclerView/android.widget.LinearLayout[1]/"
    + "android.widget.RelativeLayout/android.widget
```

```
        .FrameLayout/android.widget.ImageView[1]");  
el27.click();  
//we get a message that we can't add more pictures  
}  
}
```

## Kod 6.3: GoodTest Code

```
public class GoodExample {

    public static void main(String args[])
        throws MalformedURLException, InterruptedException {

//setup

        DesiredCapabilities dc = new DesiredCapabilities();

        dc.setCapability(MobileCapabilityType.DEVICE_NAME,
            "emulator-5554");
        dc.setCapability("platformName", "android");
        dc.setCapability("appPackage", "hr.fer.opp.zabavanet");
        dc.setCapability("appActivity",
            ".view.splash.SplashActivity");

        AndroidDriver<AndroidElement> ad =
            new AndroidDriver<AndroidElement>
                (new URL("http://127.0.0.1:4723/wd/hub"),
                 dc);

//login

        MobileElement el1 = (MobileElement)
            ad.findElementById("hr.fer.opp.zabavanet
                :id/text_email");
        el1.sendKeys("dgb.extra000@gmail.com");
        MobileElement el2 = (MobileElement)
            ad.findElementById("hr.fer.opp.zabavanet
                :id/text_password");
        el2.sendKeys("budala555");
        MobileElement el3 = (MobileElement)
            ad.findElementById("hr.fer.opp.zabavanet
                :id/btn_login");
```

```
el3.click();
Thread.sleep(3000);

//start creating event

MobileElement el4 = (MobileElement)
    ad.findElementByAccessibilityId("My events");
    el4.click();
MobileElement el5 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/fab_add_event");
el5.click();
Thread.sleep(1000);

//add picture

MobileElement el6 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_add_images");
el6.click();
Thread.sleep(1000);
MobileElement el7 = (MobileElement)
    ad.findElementById("com.android.documentsui
        :id/icon_thumb");
el7.click();
Thread.sleep(1000);

//add title and description

MobileElement el8 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/text_event_title");
el8.sendKeys("test");
MobileElement el9 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
```

```
                :id/text_event_description ");
el9.sendKeys("test");

//add exact address
MobileElement el50 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/text_event_address ");
el50.sendKeys("Iz grada");

//add event type

MobileElement el10 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_edit_event_types ");
el10.click();
MobileElement el11 = (MobileElement)
    ad.findElementByXPath("/hierarchy/android.widget
        .FrameLayout/" + "android.widget.FrameLayout
        /android.widget.FrameLayout/android.widget
        .LinearLayout/" + "android.widget.FrameLayout
        /android.widget.ListView
        /android.widget.CheckedTextView[3]");
el11.click();
MobileElement el12 = (MobileElement)
    ad.findElementById("android:id/button1");
el12.click();

//add start time

MobileElement el13 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_edit_start_time ");
el13.click();
```

```
MobileElement el14 = (MobileElement)
    ad.findElementByAccessibilityId("14 January 2020");
el14.click();
MobileElement el15 = (MobileElement)
    ad.findElementById("android:id/button1");
el15.click();
MobileElement el16 = (MobileElement)
    ad.findElementById("android:id/button1");
el16.click();

//add end time

MobileElement el17 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_edit_end_time");
el17.click();
MobileElement el18 = (MobileElement)
    ad.findElementByAccessibilityId("22 January 2020");
el18.click();
MobileElement el19 = (MobileElement)
    ad.findElementById("android:id/button1");
el19.click();
MobileElement el20 = ad.findElementById("android
    :id/button1");
el20.click();
Thread.sleep(1000);
//scroll to see the save button

TouchAction touchAction = new TouchAction(ad);
    touchAction.press(ElementOption.element(el17))
        .waitAction(WaitOptions.waitOptions
            (Duration.ofMillis(500)))
        .moveTo(ElementOption.element(el13))
        .release().perform();
```

```
//save the created event

MobileElement el21 = (MobileElement)
    ad.findElementById("hr.fer.opp.zabavanet
        :id/btn_save_event");
el21.click();
Thread.sleep(1000);

MobileElement el22 = (MobileElement)
    ad.findElementByXPath("//android.widget.FrameLayout"
        + "[@content-desc=\"Events\"]
            /android.widget.ImageView");
el22.click();
}
}
```

# Indeks slika i dijagrama

3.1	Dijagram korisnik i posjetitelj . . . . .	26
3.2	Dijagram posjetitelj . . . . .	27
3.3	Dijagram organizator i administrator . . . . .	28
3.4	Prijava pomoću Google računa . . . . .	29
3.5	Promjena osobnih podataka . . . . .	30
3.6	Pregled filtriranih događaja, pregled detalja događaja, iskazivanje interesa za određeni događaj, mijenjanje interesa za određeni događaj	31
3.7	Dodavanje događaja . . . . .	32
4.1	Arhitektura MVVM . . . . .	36
4.2	Životni ciklus ViewModela . . . . .	37
4.3	Dijagram baze podataka . . . . .	41
4.4	Dijagram razreda - model . . . . .	42
4.5	Dijagram razreda modula . . . . .	43
4.6	Dijagram razreda komponenti . . . . .	44
4.7	Dijagram razreda komponenti - drugi dio . . . . .	45
4.8	Dijagram razreda ViewModela . . . . .	46
4.9	Dijagram razreda ViewModela - drugi dio . . . . .	47
4.10	Dijagram razreda ViewModela - treći dio . . . . .	48
4.11	Dijagram razreda View . . . . .	49
4.12	Dijagram razreda View - drugi dio . . . . .	49
4.13	Dijagram razreda View - treći dio . . . . .	50
4.14	Dijagram razreda View - četvrti dio . . . . .	51
4.15	Dijagram razreda View - peti dio . . . . .	52
4.16	Dijagram razreda Utilites . . . . .	54
4.17	Dijagram razreda Utilites - drugi dio . . . . .	55
4.18	Dijagram stanja . . . . .	56
4.19	Dijagram aktivnosti . . . . .	57
4.20	Dijagram komponenti . . . . .	58
5.1	Uspješan test datuma rođenja . . . . .	61



5.2	Uspješan test dodatnih funkcija . . . . .	64
5.3	Krivo vrijeme završetka . . . . .	65
5.4	Događaj bez slike . . . . .	66
5.5	Nema tipa događaja . . . . .	67
5.6	Nedostaje naziv događaja . . . . .	68
5.7	Prikaz dodanog događaja . . . . .	69
5.8	Dodano previše slika . . . . .	70
5.9	Dijagram razmještaja . . . . .	71
5.10	Datoteka Project . . . . .	72
5.11	Datoteka Module . . . . .	72
5.12	Primjer punjenja baze . . . . .	73
6.1	Dijagram pregleda promjena . . . . .	82

## Indeks tablica

1.1	Dnevnik promjena dokumentacije . . . . .	3
4.1	Atributi dokumenata kolekcije users . . . . .	38
4.2	Atributi dokumenata kolekcije notifications . . . . .	38
4.3	Atributi dokumenata kolekcije organizations . . . . .	39
4.4	Atributi dokumenata kolekcije events . . . . .	39
4.5	Atributi dokumenata kolekcije reviews . . . . .	40
4.6	Atributi dokumenata prices . . . . .	40
4.7	Atributi dokumenata cities . . . . .	40
4.8	Atributi dokumenata eventTypes . . . . .	41
6.1	Tablica aktivnosti . . . . .	80