

Programsko inženjerstvo

Ak. god. 2023./2024.

# ConnectiNET

Dokumentacija, Rev. 2

Grupa: *Eventio*

Voditelj: *Gašpar Haramija*

Datum predaje: *19. siječanj 2024.*

Nastavnik: *Nikolina Frid*

# Sadržaj

<b>1 Dnevnik promjena dokumentacije</b>	<b>3</b>
<b>2 Opis projektnog zadatka</b>	<b>6</b>
<b>3 Specifikacija programske potpore</b>	<b>11</b>
3.1 Funkcionalni zahtjevi . . . . .	11
3.1.1 Obrasci uporabe . . . . .	13
3.1.2 Sekvencijski dijagrami . . . . .	27
3.2 Ostali zahtjevi . . . . .	30
<b>4 Arhitektura i dizajn sustava</b>	<b>31</b>
4.1 Baza podataka . . . . .	32
4.1.1 Opis tablica . . . . .	33
4.1.2 Dijagram baze podataka . . . . .	37
4.2 Dijagram razreda . . . . .	38
4.3 Dijagram stanja . . . . .	45
4.4 Dijagram aktivnosti . . . . .	46
4.5 Dijagram komponenti . . . . .	48
<b>5 Implementacija i korisničko sučelje</b>	<b>49</b>
5.1 Korištene tehnologije i alati . . . . .	49
5.2 Ispitivanje programskog rješenja . . . . .	51
5.2.1 Ispitivanje komponenti . . . . .	51
5.2.2 Ispitivanje sustava . . . . .	55
5.3 Dijagram razmještaja . . . . .	58
5.4 Upute za puštanje u pogon . . . . .	60
<b>6 Zaključak i budući rad</b>	<b>65</b>
<b>Popis literature</b>	<b>67</b>
<b>Indeks slika i dijagonama</b>	<b>69</b>

**Dodatak: Prikaz aktivnosti grupe**

**70**

# 1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Gašpar Haramija	24.10.2023.
0.2	Ažuriranje dnevnika sastajanja.	Filip Buhiniček	26.10.2023.
0.3	Dodavanje općeg UC dijagrama te pojedinačnih UC dijagrama. Dodane reference.	Sandro Boka	30.10.2023.
0.4	Izrada i dopuna funkcionalnih zahtjeva. Dodane reference.	Mateo Grdić	31.10.2023.
0.5	Dodani tekstualni opisi obrazaca uporabe	Marko Sr- sic, Jakov Šarolić	31.10.2023.
0.6	Dopisan opis projekta te ažuriran dnevnik promjena dokumentacije	Filip Bu- hiniček, Gašpar Haramija	01.11.2023.
0.7	Sekvencijski dijagrami	Gašpar Haramija	02.11.2023.
0.8.1	Prva verzija dijagrama razreda	Sandro Boka, Marko Sršić	03.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

<b>Rev.</b>	<b>Opis promjene/dodataka</b>	<b>Autori</b>	<b>Datum</b>
0.8.2	Nastavak i ažuriranje dijagrama razreda	Sandro Boka, Marko Sršić	15.11.2023.
0.9	Ažuriranje dokumentacije, brisanje template dijelova	Marko Sršić	16.11.2023.
<b>1.0</b>	Verzija samo s bitnim dijelovima za 1. ciklus	Gašpar Haramija	17.11.2023.
1.1	Ažuriranje obrazaca uporabe prema uputama iz 1. revizije	Sandro Boka, Marko Sršić	9.12.2023.
1.2	Brisanje template dijelova	Marko Sršić	18.12.2023.
1.3	Dijagram stanja	Marko Sršić	19.12.2023.
1.4	Dijagram aktivnosti	Marko Sršić	20.12.2023.
1.5.1	Razni ispravci (funkcionalni zahtjevi, obrasci uporabe, arhitektura i dizajn sustava, itd.)	Marko Sršić	4.1.2024.
1.5.2	Ažuriranje dnevnika sastajanja, tablice aktivnosti	Marko Sršić	11.1.2024.
1.6	Dijagram komponenti, dijagram razmještaja	Gašpar Haramija	13.1.2024.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

<b>Rev.</b>	<b>Opis promjene/dodataka</b>	<b>Autori</b>	<b>Datum</b>
1.7	Korištene tehnologije, ispitivanje komponenti/sustava, opći ispravci i ažuriranja	Gašpar Haramija, Marko Sršić	17.1.2024.
1.8	Upute za puštanje u pogon	Gašpar Haramija	17.1.2024.
1.9	Dijagrami pregleda promjena, zaključak i budući rad	Gašpar Haramija	17.1.2024.
2.0	Konačni tekst predloška dokumentacije	Gašpar Haramija, Marko Sršić	18.1.2024.

## 2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za stvaranje web aplikacije "Eventio", koja će korisnicima omogućiti stvaranje, oglašavanje i sudjelovanje na različitim događanjima. Ova inovativna web aplikacija će korisnicima omogućiti lakši pristup društvenim događanjima koja odgovaraju njihovim interesima te olakšati organizaciju i promociju istih.

Postoje mnoge društvene platforme koje već imaju razvijene mreže korisnika i nude vlastite implementacije sličnih rješenja za događanja. Međutim, ono što "Eventio" čini drugačijim jest njegov pristup. Dok druga rješenja ovise o korisnicima koji stvaraju različite grupe i objavljaju događanja na koje se korisnici prijavljuju ili postaju članovi tih grupa (primjer toga je Facebook), "Eventio" nudi alternativni pristup. Korisnici će moći vidjeti sve dostupne događaje i filtrirati ih prema vlastitim preferencijama, kao što su vrsta događaja, lokacija, datum i druge karakteristike. Na taj način, korisnici će imati veće šanse saznati za događanja koja najbolje odgovaraju njihovim interesima, čime se rješava glavni problem: pronalaženje događanja koja odgovaraju njihovim željama i potrebama.

Ova aplikacija je namijenjena svima - bez obzira na dob, spol ili interes. Svatko tko želi sudjelovati na događanjima bilo kojeg opsega i pronaći svoju zajednicu s istim interesima može pronaći korist u "Eventio". Osim što korisnicima pruža bolji pristup događanjima, aplikacija također nudi organizatorima priliku da promoviraju svoje događaje i privuku veći broj sudionika.

Unatoč prednostima, postoji nekoliko problema koji će se morati rješavati kako bi se projekt uspješno realizirao i u konačnici našao među krajnjim korisnicima. Aplikacija ovisi o velikom broju korisnika i organizatora kako bi postala funkcionalna i privukla interes. Potrebno je privući dovoljan broj organizatora s raznovrsnim događanjima kako bi se zadovoljili interesi široke publike. Također, potrebno je privući i potaknuti korisnike da koriste aplikaciju i aktivno sudjeluju na događanjima. Konkurenčija s postojećim sličnim rješenjima kao što su Facebook, Meetup i Eventbrite ima prednost zbog svojih već razvijenih korisničkih baza koje mogu koristiti za testiranje i lansiranje ovakvog proizvoda.

U ovom kontekstu, "Eventio" se postavlja kao platforma koja će pružiti jed-

nostavan i učinkovit način povezivanja organizatora i posjetitelja događanja te olakšati promociju i sudjelovanje u raznovrsnim događanjima. Kroz inovativan pristup i kontinuirane nadogradnje, "Eventio" ima potencijal promijeniti način na koji korisnici pronalaze i sudjeluju na događanjima u svom okruženju. Primarni cilj ovog zadatka je razviti Minimalan vitalni proizvod (Minimum viable product, MVP), rješenje koje će na najjednostavniji i pristupačan način rješavati problem pronalaženja i sudjelovanja na događanjima prema osobnim preferencijama. MVP će osigurati osnovne funkcionalnosti aplikacije kako bi se zadovoljile osnovne potrebe korisnika i omogućila provedba testiranja i daljnog razvoja.

U sljedećem dijelu opisa projekta bit će detaljno razrađena problematika zadatka. Također, bit će opisani korisnički zahtjevi i predložena moguća rješenja za svaku komponentu aplikacije. Sve ovo ima za cilj definirati temelje i smjernice za daljnji razvoj "Eventio" aplikacije.

Sljedeći dio opisa projekta detaljnije će razmotriti problematiku projektnog zadatka, identificirati glavne aktore i dionike te opisati korisničke zahtjeve i funkcionalnosti aplikacije "Eventio".

Glavni aktori uključuju:

- *Administrator: Osoba s najvišim ovlastima u sustavu, odgovorna za upravljanje korisnicima, događajima, i finansijskim aspektima*
- *Korisnik (Organizator): Osoba koja može stvarati i oglašavati svoje događaje, upravljati svojim profilom, i platiti mjesecnu članarinu za objavu plaćenih događaja*
- *Korisnik (Posjetitelj): Osoba koja može pretraživati i iskazivati zainteresiranost za događaje, ostavljati recenzije i upravljati svojim profilom.*
- *Banka i PayPal: Pružatelji usluga plaćanja koji omogućuju korisnicima plaćanje mjesечne članarine i ulaznica za događaje.*
- *Baza podataka: Skladište podataka o korisnicima, događajima, recenzijama, i finansijskim transakcijama*

U nastavku su detaljno razmotreni korisnički zahtjevi i funkcionalnosti aplikacije "Eventio".

Prilikom pokretanja aplikacije, korisniku će se prikazati jedna od dvije mogućnosti.

- *Prozor sa pozdravom: Korisnik je već stvorio profil i ranije se prijavio u sustav*
- *Prozor sa formom za prijavu te gumbom koji vodi na kreiranje novog računa: Ovisno o tome posjeduje li korisnik račun će ili ispuniti prijavu u sustav ili pak kliknuti na gumb za stvaranje računa.*

Kako bi korisnik mogao uspješno kreirati račun potrebni su sljedeći podaci:

- *Korisničko ime*
- *Lozinka*
- *E-mail adresa*

Korisnik tijekom registracije isto tako može birati koju ulogu želi imati unutar aplikacije. Uloge su posjetitelj i organizator. Ako je korisnik odabrao ulogu organizatora, tada još dodatno uz navedene podatke treba upisati i sljedeće:

- *Naziv organizacije*
- *Adresa*
- *Link/Url svoje stranice*

Nakon kreiranja samog računa kao jedna uloga, neće biti omogućeno mijenjanje te uloge. Za prijavu će biti potrebno unijeti korisničko ime i lozinku postojećeg računa. Za odjavu će korisnici imati gumb odjavi koji je potrebno pritisnuti kako bi se korisnik odjavio sa računa te se nakon toga vraća na početni zaslon sa formom za prijavu.

Nakon uspješne prijave/registracije kao organizator događaja, korisnik može krenuti sa oglašavanjem. Prilikom unosa novog događaja, organizator mora unijeti podatke kao što su:

- *Naziv događaja*
- *Vrsta događaja*
- *Lokacija događaja*
- *Opis lokacije*
- *Vrijeme početka*
- *Trajanje*
- *Cijena ulaznice*
- *Opis događaja*

Gore navedeni podaci su obavezni, te uz obavezne podatke organizator može dodati još slike ili video snimke u galeriju samog događaja. Kada završi sa kreiranjem događaja, njega je moguće vidjeti na profilu organizatora zajedno sa svim njegovim podacima te događajima koji su se zbili unatrag dvije godine ili drugim događajima koji su stvorenih i tek dolaze.

Neki organizatori će organizirati događaje s besplatnim ulazom, no neki će vjerojatno htjeti naplatiti ulaz na događaj. Za mogućnost objavljivanja budućih

događaja za koje se plaća ulaz organizator mora biti preplatnik mjesecne članarine na web aplikaciji. Prilikom objave prvog takvog događaja organizator je obvezan platiti članarinu.

Članarina se može platiti na dva načina:

- *PayPal*
- *Bankovna kartica*

Članarina će se obnavljati na mjesecnoj bazi sve dokle organizator sadrži neki događaj za koji je potrebno plaćanje ulaza. Također, prilikom objave novog događaja od strane organizatora za koji se naplaćuje ulaz, a organizator je u tom trenutku preplatnik, slobodan je objaviti događaj normalnim procesom bez dodatnih obaveza.

Nakon uspješne prijave/registracije kao posjetitelj, korisnik može krenuti tražiti događaje koji mu odgovaraju te iskazivati svoju zainteresiranost za sami dolazak na te događaje. Zainteresiranost se dijeli u tri skupine:

- *Sigurno dolazim*
- *Možda dolazim*
- *Ne dolazim*

Kada korisnik stisne na bilo koju grupu, na samom događaju će biti vidljiva zainteresiranost. Ako je odabrana grupa „Sigurno dolazim“ ili „Možda dolazim“, tada se na profilu posjetitelja pojavi taj događaj.

Posjetitelj isto tako može ostaviti recenziju na događaju koja će biti dodana u listu svih recenzija pod tim događajem. Kako bi ju uspješno dodoao u listu recenzija, svaka recenzija mora sadržavati kratki opis recenzije te opcionalno može sadržavati ocjenu od 1 do 10.

Posjetitelji unutar aplikacije mogu filtrirati sve događaje. Mogu ih filtrirati po nazivu ili vrsti događaja, njihovom vremenu početka, po lokaciji na kojoj se događaj odvija te po samom trajanju događaja. Na taj način posjetitelj može naći one događaje koji mu najviše odgovaraju i koji su mu najpogodniji. Kako bi se najvećim fanovima olakšala potražnja događaja njihovih najdražih organizatora, svaki posjetitelj moći će pretraživati isto tako i po korisničkom imenu organizatora. Kao što je navedeno, na profilu organizatora navedeni su svi prethodni događaji te svi događaji u budućnosti koji se također mogu filtrirati.

Svaki korisnik će isto tako imati mogućnosti osvježavanja i uređivanja vlastitog profila. Posjetitelji će moći izmijeniti svoje podatke te uvijek mogu promijeniti

zainteresiranost za neki event. Osim toga, posjetitelji mogu odabrati postavke da im aplikacija automatski šalje obavijesti o najnovijim događanjima prema zadanim kriterijima: vrsta događanja i područje. Organizatori uz promjene vlastitih podataka mogu isto tako promijeniti podatke događaja koji će se tek dogoditi, a svi oni već prošli događaji neće se moći mijenjati od strane organizatora.

Kako bi posjetiteljima dodatno olakšali korištenje aplikacije, događaji na kojima će zainteresiranost biti „Sigurno dolazim“ ili „Možda dolazim“ poslati će e-mail poruku na posjetiteljev račun 24 sata prije početka događaja. Isto tako, kada dođe do neke promjene na događaju, a koje nisu došle sa strane posjetitelja, ponovno stiže notifikacija u obliku e-mail poruke o samoj promjeni.

Uz dvije uloge korisnika, organizator i posjetitelj, unutar sustava postoji i uloga administratora. Administrator se kao i ostali korisnici može prijaviti te odjaviti iz sustava te unutar sustava on je korisnik s najvišim ovlastima.

Neće imati mogućnosti iskazati zainteresiranost za neki event te neće moći stvarati događaje kao što to organizator može, no ima druge ovlasti. On može pristupiti bazi podataka te može pregledati profil bilo kojeg drugog korisnika. Ukoliko smatra potrebnim, on može mijenjati podatke o događajima te može brisati recenzije ukoliko ih smatra nevaljanim. Isto tako može brisati same događaje, ali i korisnike ako se ne pridržavaju pravila unutar aplikacije. Zajedno sa time, on je taj koji određuje kolika će biti cijena mjesecne članarine koju plaćaju organizatori koji posjeduju neki događaj za koji se plaća upad.

Za ovakav tip proizvoda može se stalnim nadogradnjama i poboljšanjima privući još i šira publika. Ubrzo bi se na sami prototip aplikacije mogli dodati i chatovi unutar svakog događaja kako bi ljudi mogli komunicirati i prije samog događaja te na taj način zapravo započeli sami događaj i prije njegovog stvarnog početka. Uz ovaj dodatak mogli bi se na vrh svakog pretraživanja dodati i oni događaji za koje smatramo da su najpogodniji posjetitelju ovisno o njegovim prošlim izborima događaja.

# 3. Specifikacija programske potpore

## 3.1 Funkcionalni zahtjevi

Dionici:

1. Administrator
2. Korisnik/Posjetitelj
3. Organizator
4. Baza podataka
5. PayPal
6. Banka

Aktori i njihovi funkcionalni zahtjevi:

1. Administrator (inicijator) može:
  - (a) prijaviti se i odjaviti sa stranice
  - (b) pregledavati događaje
    - i. filtrirati događaje po vremenskom razdoblju
  - (c) upravljati korisnicima
    - i. urediti ili izbrisati korisničke profile
    - ii. urediti ili izbrisati objave korisnika
  - (d) postavljati cijene članstva
2. Korisnik/Posjetitelj (inicijator) može:
  - (a) prijaviti se i odjaviti sa stranice
  - (b) napraviti profil
  - (c) prikazati profil i urediti ga
  - (d) pregledavati događaje
    - i. filtrirati događaje po vremenskom razdoblju i ostalim parametrima
  - (e) uključiti obavijesti o najnovijim događajima po kriterijima: vrsta događanja i po-dručje

- (f) iskazati interes za događaj („sigurno dolazim“, „možda dolazim“, „ne dolazim“)
- (g) recenzirati događaj

3. Organizator (inicijator) može:

- (a) postaviti događaj
  - i. urediti ili izbrisati događaj
  - ii. postaviti događaj sa plaćanjem ulaznice i platiti članarinu (Paypalom ili bankovnom karticom)
- (b) prijaviti se i odjaviti sa stranice
- (c) napraviti profil
- (d) prikazati profil i urediti ga
- (e) pregledavati događaje
  - i. filtrirati događaje po vremenskom razdoblju i ostalim parametrima

4. Baza podataka (sudionik):

- (a) pohranjuje zapise o korisnicima i događajima
- (b) pohranjuje iskazane interese za događaje
- (c) pohranjuje recenzije događaja
- (d) pohranjuje informacije o cijenama članstva
- (e) omogućuje prijavu i odjavu sa stranice (token)

5. PayPal (sudionik):

- (a) Odobrava transakcije potrebne za platiti članarinu

6. Banka (sudionik):

- (a) Odobrava transakcije potrebne za platiti članarinu

### 3.1.1 Obrasci uporabe

#### UC1 - Registracija posjetiteljskog računa

- **Glavni sudionik:** Korisnik
- **Cilj:** Izrada korisničkog računa za potrebe pregledavanja i posjećivanja događaja
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik izabire opciju registracije
  2. Korisnik unosi korisničke podatke koji se od njega traže
  3. Sustav provjerava ispravnost unesenih podataka
  4. Sustav šalje korisnika na prozor za prijavu na stranicu
- **Opis mogućih odstupanja:**
  - 2.a Korisnik odabire već zauzeto korisničko ime ili e-mail
    1. Sustav obavještava korisnika o neuspješnoj registraciji i navodi razlog zašto registracija nije uspjela
    2. Korisnik mijenja potrebne podatke, završi registraciju ili odustane od registracije

#### UC2 - Registracija organizatorskog računa

- **Glavni sudionik:** Korisnik
- **Cilj:** Izrada korisničkog računa za potrebe stvaranja/organiziranja događaja
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik izabire opciju registracije uz odabir polja "Organizator"
  2. Korisnik unosi korisničke podatke koji se od njega traže
  3. Sustav provjerava ispravnost unesenih podataka
  4. Sustav šalje korisnika na prozor za prijavu na stranicu
- **Opis mogućih odstupanja:**
  - 2.a Korisnik odabire već zauzeto korisničko ime ili e-mail
    1. Sustav obavještava korisnika o neuspješnoj registraciji i navodi razlog zašto registracija nije uspjela
    2. Korisnik mijenja potrebne podatke, završi registraciju ili odustane od registracije

#### UC3 - Prijava na stranicu

---

- **Glavni sudionik:** Korisnik, Administrator
- **Cilj:** Dobiti pristup stranici
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik ima registrirani račun ili ima račun s pravima administratora
- **Opis osnovnog tijeka:**
  1. Korisnik/Administrator otvara stranicu
  2. Korisnik/Administrator unosi korisničke podatke za prijavu na stranicu
  3. Sustav prepoznaje je li prijavljeni račun korisnički ili administratorski te otvara odgovarajuću početnu stranicu
- **Opis mogućih odstupanja:**
  - 2.a Korisnik unosi pogrešnu kombinaciju imena i lozinke za prijavu
    1. Sustav javlja korisniku da je prijava neuspješna, nudi ponovnu prijavu ili stvaranje novog korisničkog računa

#### UC4 - Pregled događaja

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati dostupne događaje
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik se prijavio na stranicu
- **Opis osnovnog tijeka:**
  1. Nakon prijave korisnika se prebacuje na stranicu za pregledavanje događaja  
- "Svi događaji"
  2. Korisniku su događaji izlistani jedan ispod drugog
  3. Kod svakog događaja prikazane su sve informacije o događaju (npr. datum održavanja, broj zainteresiranih itd.) te pripadajuća fotografija

#### UC5 - Filtriranje događaja

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati događaje prema željenim kriterijima
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen i nalazi se na stranici za pregled događaja
- **Opis osnovnog tijeka:**
  1. Korisnik na stranici za pregledavanje događaja bira između više ponuđenih opcija za filtriranje događaja
- **Opis mogućih odstupanja:**

- 1.a Ne postoji događaj koji odgovara korisnikovoj kombinaciji filtera
  1. Sustav korisniku prikazuje stranicu bez događaja

### UC6 - Prikazivanje osobnog profila

- **Glavni sudionik:** Korisnik
- **Cilj:** Pregledati osobni profil
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen na stranici
- **Opis osnovnog tijeka:**
  1. Korisnik klikne na ime svog računa te odabire opciju "Profil"
  2. Sustav prikazuje stranicu korisnikovog osobnog profila

### UC7 - Uređivanje profila

- **Glavni sudionik:** Korisnik
- **Cilj:** Izmjena osobnih podataka
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen na stranici i odabrao je opciju "Profil" (UC6)
- **Opis osnovnog tijeka:**
  1. Korisnik na stranici osobnog profila pored podatka koji želi promijeniti pritisne ikonu olovke
  2. Sustav omogućuje izmjenu podatka
  3. Korisnik mijenja podatak
  4. Korisnik odabire ikonu kvačice
  5. Korisnik odabire opciju "SPREMI"
  6. Sustav korisniku javlja da su promjene uspješno pohranjene
- **Opis mogućih odstupanja:**
  - 4.a Korisnik je promijenio korisničko ime ili e-mail u nešto što već postoji u sustavu
    1. Sustav javlja korisniku da je došlo do pogreške
    2. Korisnik ponovno upisuje ime/e-mail ili odustaje od promjena te ih briše pritiskom na ikonu križića

### UC8 - Odjava sa stranice

- **Glavni sudionik:** Korisnik, Administrator
- **Cilj:** Odjaviti se sa stranice
- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik/Administrator je prijavljen na stranici
- **Opis osnovnog tijeka:**
  1. Korisnik/Administrator nakon klika na korisničko ime odabire opciju "Odjava"
  2. Sustav odjavljuje korisnika/administratora sa stranice

### UC9 - Izražavanje interesa

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Izraziti interes za neki ponuđeni događaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen na stranici i njegov račun je posjetiteljski
- **Opis osnovnog tijeka:**
  1. Posjetitelj pregledava događaje (UC4) te dolazi do događaja koji ga zanimaju
  2. Posjetitelj odabire jednu od tri ponuđene opcije interesa: "DOLAZIM", "MOŽDA", "NE DOLAZIM"
  3. Sustav pohranjuje podatak o posjetiteljevom interesu, a u slučaju da je posjetitelj odabrao "DOLAZIM" ili "MOŽDA", ažurira odgovarajući brojač kod prikaza događaja te dodaje taj događaj na popis posjetiteljevih interesnih događaja
- **Opis mogućih odstupanja:**
  - 2.a Posjetitelj odabrao pogrešnu opciju ili želi promijeniti interes
    1. Posjetitelj odabire neku drugu ponuđenu opciju

### UC10 - Pregled posjetiteljevih interesnih događaja

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Pregledati sve događaje koje posjetitelja zanimaju
- **Sudionici:** Baza podataka
- **Preduvjet:** Posjetitelj je prijavljen i na početnoj je stranici
- **Opis osnovnog tijeka:**
  1. Posjetitelj na početnoj stranici odabire opciju "MOJI DOGAĐAJI"
  2. Sustav posjetitelju otvara stranicu na kojoj je popis svih događaja za koje je postavio interes "DOLAZIM" ili "MOŽDA"
- **Opis mogućih odstupanja:**
  - 2.a Posjetitelj nije odabrao odgovarajuću opciju niti za jedan događaj
    1. Sustav korisniku prikazuje stranicu bez događaja

## UC11 - Recenziranje događaja

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Ostaviti recenziju na posjećen događaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Posjetitelj je prijavljen na stranicu i nalazi se na stranici pregleda interesnih događaja (UC10)
- **Opis osnovnog tijeka:**
  1. Posjetitelj dolazi do događaja kojeg želi recenzirati
  2. Ako je događaj završio i nije prošlo 48 sati od njegovog završetka, posjetitelju je otvoren obrazac za recenziranje u kojem može odabrati ocjenu te napisati recenziju
  3. Posjetitelj završava recenziju i odabire opciju "Pohrani recenziju"
  4. Sustav pohranjuje recenziju
  5. Ako događaj nije završio, ili je prošlo više od 48 sati od njegovog završetka, posjetitelju nije otvoren obrazac za recenziranje
- **Opis mogućih odstupanja:**
  - 2.a Posjetitelj napisao opis recenzije, no nije izabrao ocjenu
    1. Sustav ne dozvoljava pohranjivanje recenzije dok ocjena nije odabrana
    2. Posjetitelj odabire ocjenu i pohranjuje recenziju, ili odustaje

## UC12 - Automatsko slanje obavijesti

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Kroz aplikaciju postaviti automatsko primanje obavijesti za najnovije događaje na temelju zadanih kriterija
- **Sudionici:** Baza podataka
- **Preduvjet:** Posjetitelj prijavljen i nalazi se na stranici osobnog korisničkog računa
- **Opis osnovnog tijeka:**
  1. Korisnik ispod naznake "Preplate na obavijesti" zadaje kriterij notifikacija - vrstu događaja te lokaciju
  2. Korisnik spremi pretplatu na obavijesti pritiskom na ikonu plusa te sustav pohranjuje korisnikov odabir
- **Opis mogućih odstupanja:**
  - 2.a Korisnik odabrao pretplatu na obavijesti, ali se predomislio i ne želi je

1. Sustav omogućuje korisniku brisanje pretplate klikom na ikonu koša za smeće

### UC13 - Postavljanje događaja

- **Glavni sudionik:** Organizator
- **Cilj:** Postaviti novi događaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen na stranici i ima organizatorski račun
- **Opis osnovnog tijeka:**
  1. Organizator započinje proces izrade novog događaja pritiskom na ikonu plusa u desnom donjem kutu
  2. Organizatoru se otvara obrazac za dodavanje događaja
  3. Organizator popunjava sve potrebne podatke o događaju te postavlja događaj odabirom opcije "DODAJ"
  4. Sustav pohranjuje novi događaj te vraća organizatora na stranicu "Moji događaji"
- **Opis mogućih odstupanja:**
  - 4.a Organizator odustaje od objave događaja
    1. Organizator ima opciju "Odustani" čijim odabirom prekida proces stvaranja novog događaja

### UC14 - Postavljanje događaja sa plaćanjem ulaznice

- **Glavni sudionik:** Organizator
- **Cilj:** Postaviti događaj na kojem će biti naplaćivanje ulaznica
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik prijavljen na stranicu, ima organizatorski korisnički račun i plaćenu pretplatu
- **Opis osnovnog tijeka:**
  1. Organizator pri ispunjavanju podataka o događaju kojeg želi postaviti stavlja željenu cijenu događaja (veću od 0)
  2. Organizator odabire opciju "DODAJ", sustav pohranjuje novi događaj te vraća organizatora na stranicu "Moji događaji"
- **Opis mogućih odstupanja:**
  - 2.a Organizator nema plaćenu članarinu
    1. Sustav javlja organizatoru kako nema plaćenu članarinu i ne objavljuje događaj

### UC15 - Plaćanje članarine

- **Glavni sudionik:** Organizator
- **Cilj:** Platiti članarinu kako bi se mogao postaviti događaj na kojem će se naplaćivati ulaznice
- **Sudionici:** Baza podataka, PayPal, Banka
- **Preduvjet:** Organizator se nalazi na stranici osobnog korisničkog računa
- **Opis osnovnog tijeka:**
  1. Organizator odabire opciju "PRETPLATA"
  2. Organizator odabire način plaćanja članarine - plaćanje PayPal-om ili plaćanje bankovnom karticom, ispunjava podatke i odabire opciju "Pretplati se"
  3. Sustav javlja organizatoru da je plaćanje članarine uspješno
- **Opis mogućih odstupanja:**
  - 2.a Transakcija je neuspješna
    1. Sustav javlja organizatoru kako transakcija nije prošla
    2. Organizator ponavlja postupak ili odustaje od plaćanja odabirom opcije "Povratak na profil"

### UC16 - Pregled vlastitih događaja

- **Glavni sudionik:** Organizator
- **Cilj:** Pregledati vlastite postavljene događaje
- **Sudionici:** Baza podataka
- **Preduvjet:** Organizator prijavljen na stranicu
- **Opis osnovnog tijeka:**
  1. Organizator na početnoj stranici odabire opciju "MOJI DOGAĐAJI"
  2. Sustav otvara organizatoru stranicu na kojoj prikazuje sve njegove postavljene događaje
- **Opis mogućih odstupanja:**
  - 2.a Organizator nema niti jedan postavljen događaj
    1. Sustav korisniku prikazuje stranicu bez događaja

### UC17 - Uređivanje/brisanje događaja

- **Glavni sudionik:** Organizator
- **Cilj:** Urediti ili obrisati postavljeni događaj
- **Sudionici:** Baza podataka

- **Preduvjet:** Organizator prijavljen, nalazi se na stranici "MOJI DOGAĐAJI" te ima postavljeni događaj
- **Opis osnovnog tijeka:**
  1. Organizator pronađe događaj kojeg želi uređiti ili izbrisati i pritisne na ikonu olovke, odnosno koš za smeće
  2. Ako je izabrana ikona olovke, sustav otvara organizatoru obrazac za postavljanje događaja ispunjen s njegovim podacima i dozvoljava uređivanje; organizator zatim uređuje podatke i bira opciju "Spremi"
  3. Ako je izabrana ikona koš za smeće, sustav briše događaj iz baze podataka
- **Opis mogućih odstupanja:**
  - 2.a Organizator greškom unio krive podatke
    1. Sustav dozvoljava izmjenu podataka neograničen broj puta

#### UC18 - Administrativni pregled događaja

- **Glavni sudionik:** Administrator
- **Cilj:** Pregledati događaje kao administrator
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran, ima prava administratora i uspješno se prijavio
- **Opis osnovnog tijeka:**
  1. Administrator na početnoj stranici odabire opciju "SVI DOGAĐAJI"
  2. Sustav administratora prebacuje na stranicu s popisom događaja

#### UC19 - Postavljanje cijene pretplate

- **Glavni sudionik:** Administrator
- **Cilj:** Postaviti ili promijeniti cijenu pretplate
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran, ima prava administratora i nalazi se na stranici osobnog profila
- **Opis osnovnog tijeka:**
  1. Administrator ispod naznake "Cijena pretplate" ažurira cijenu pretplate i odabire opciju "Spremi"

#### UC20 - Upravljanje korisnicima

- **Glavni sudionik:** Administrator

- **Cilj:** Pregledati korisničke račune i/ili njihove objave
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i ima prava administratora
- **Opis osnovnog tijeka:**
  1. Administrator na početnoj stranici odabire opciju "SVI KORISNICI"
  2. Sustav administratora prebacuje na stranicu s popisom korisnika i njihovim podacima

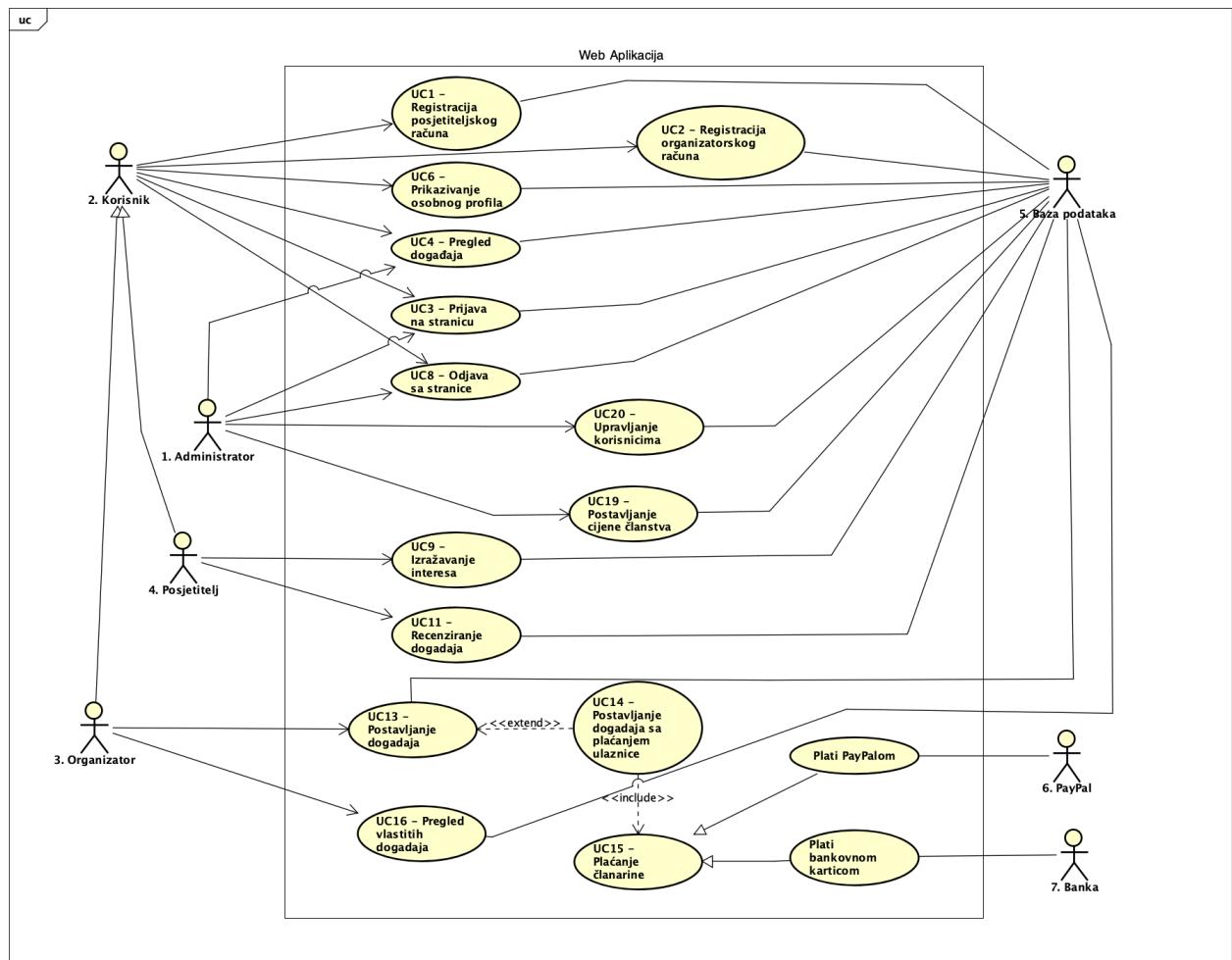
#### UC21 - Uređivanje/brisanje korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Urediti ili obrisati korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i ima prava administratora
- **Opis osnovnog tijeka:**
  1. Administrator pronalazi željenog korisnika
  2. Administrator odabire ikonu koša za smeće te briše njegove podatke iz baze podataka

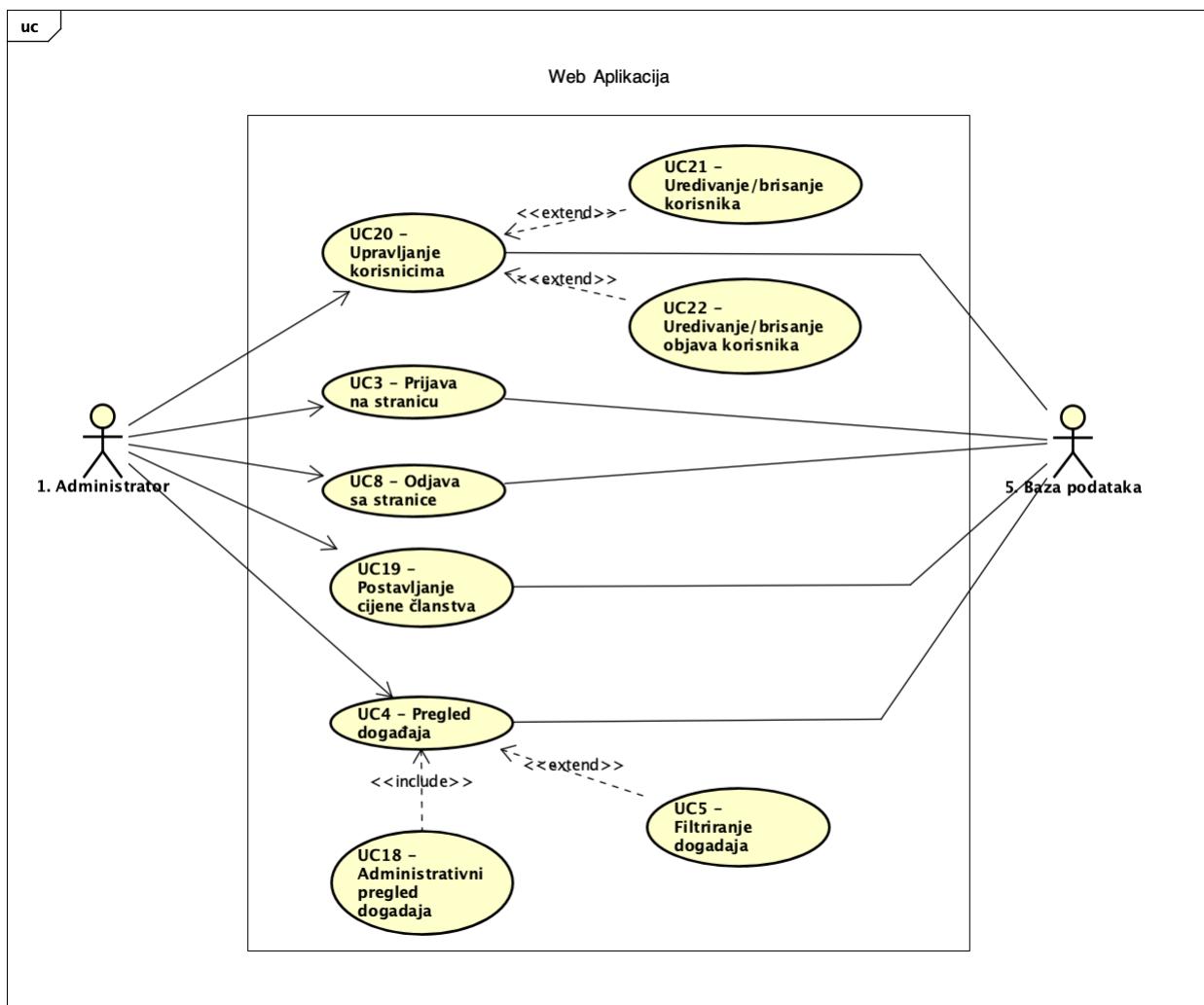
#### UC22 - Uređivanje/brisanje objava korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Urediti/obrisati objave korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran, ima prava administratora i pregledava sve događaje
- **Opis osnovnog tijeka:**
  1. Administrator pronalazi događaj kojeg želi urediti ili izbrisati te pritisne opciju na ikonu olovke ili ikonu koša za smeće
  2. Ako je odabrana ikona olovke, sustav otvara obrazac za postavljanje događaja ispunjen s njegovim podacima i dozvoljava uređivanje; administrator zatim uređuje podatke i bira opciju "Spremi"
  3. Ako je izabrana ikona koša za smeće, sustav briše događaj iz baze podataka

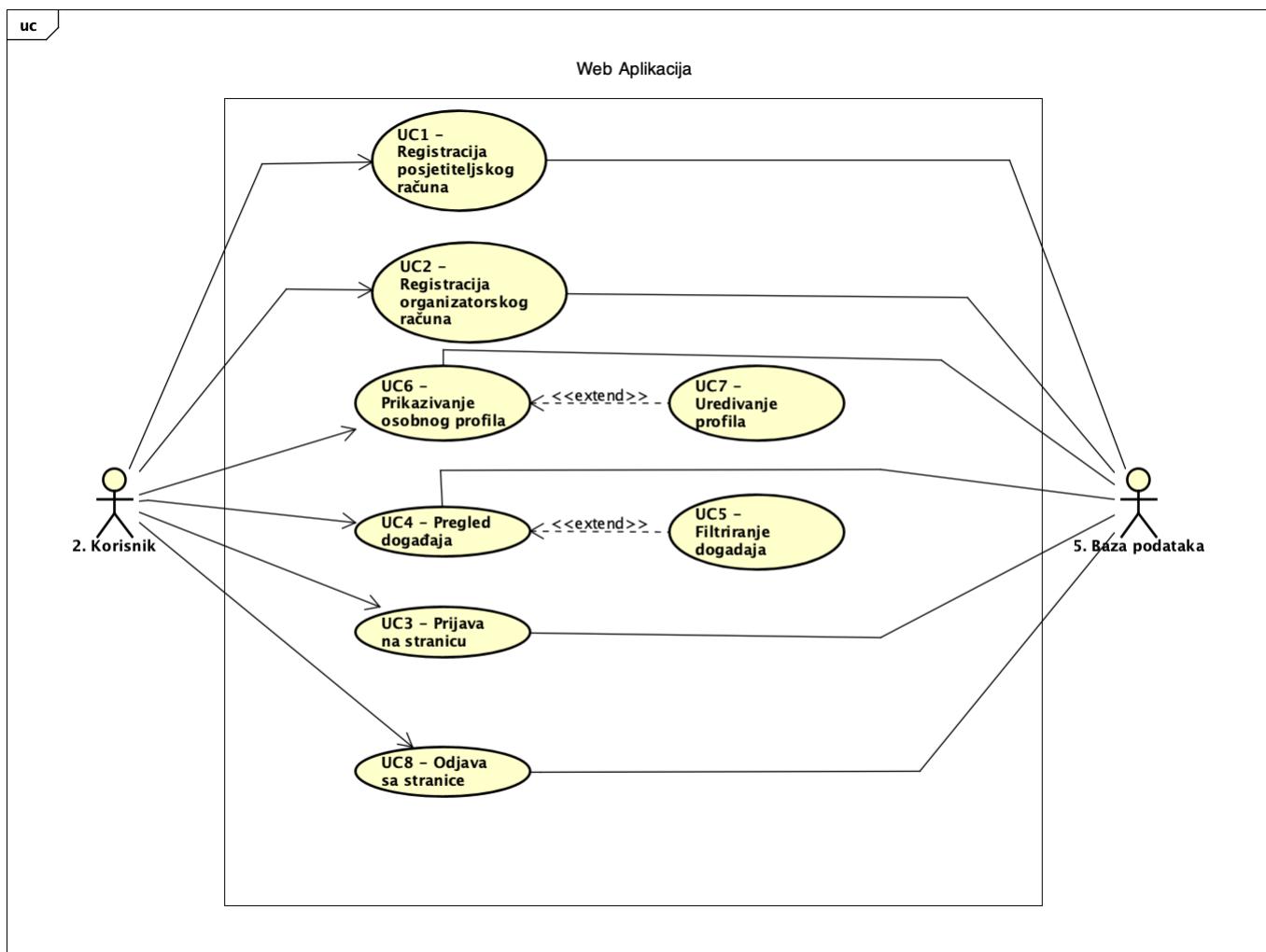
## Dijagrami obrazaca uporabe



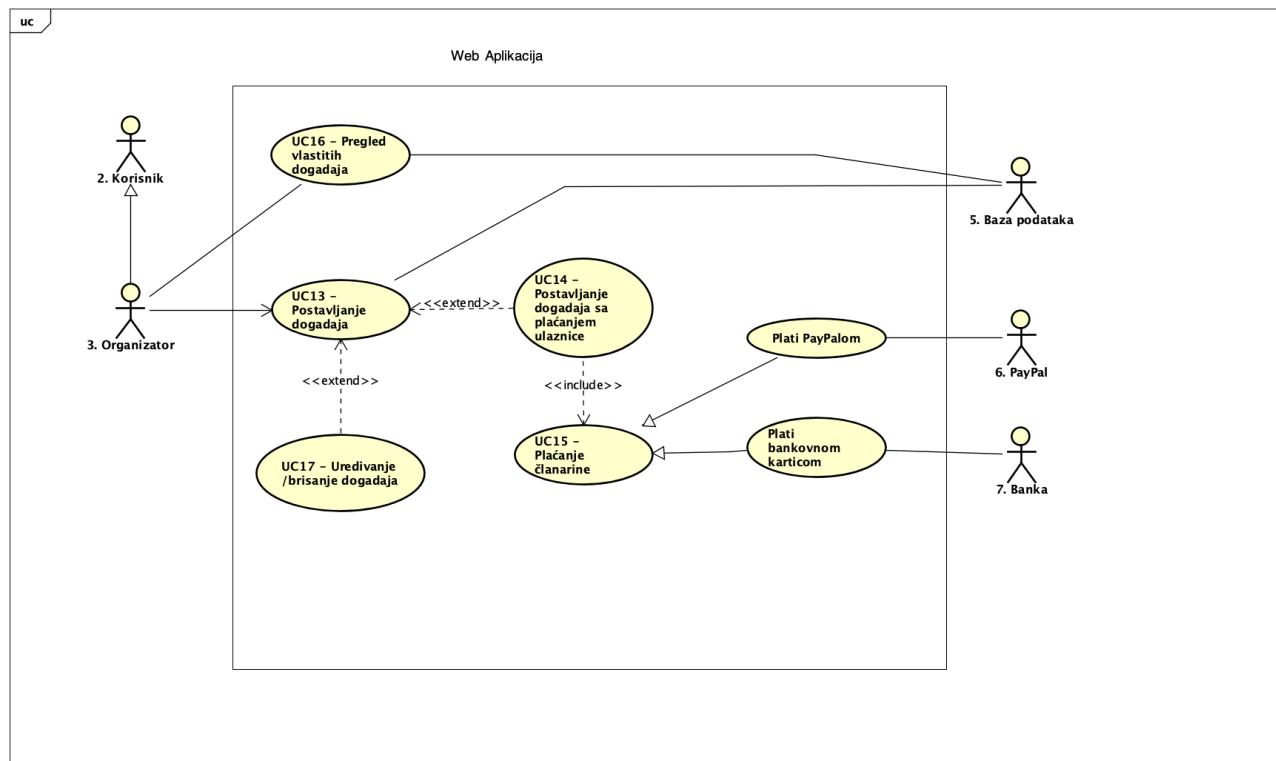
Slika 3.1: Dijagram obrasca uporabe, opća funkcionalnost aktora



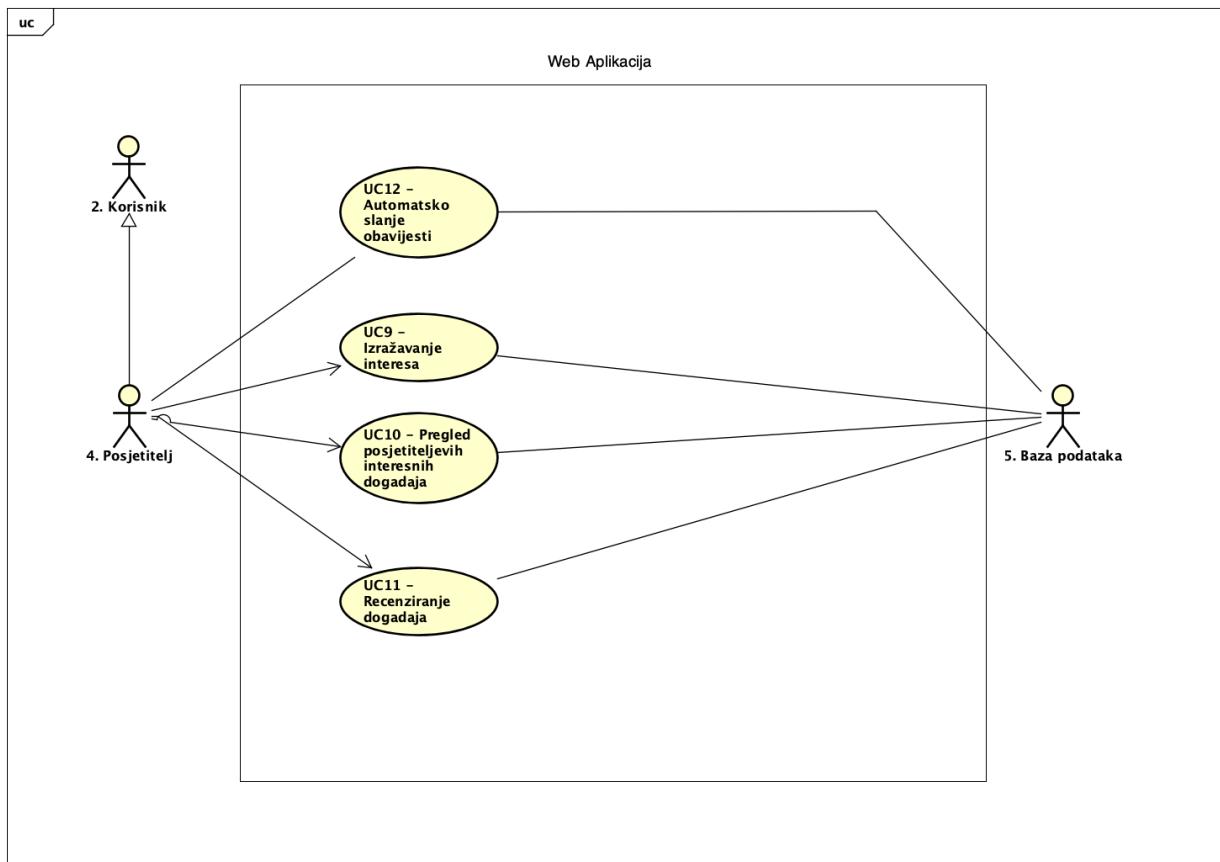
Slika 3.2: Dijagram obrasca uporabe, funkcionalnost administratora



Slika 3.3: Dijagram obrasca uporabe, funkcionalnost korisnika



Slika 3.4: Dijagram obrasca uporabe, funkcionalnost organizatora



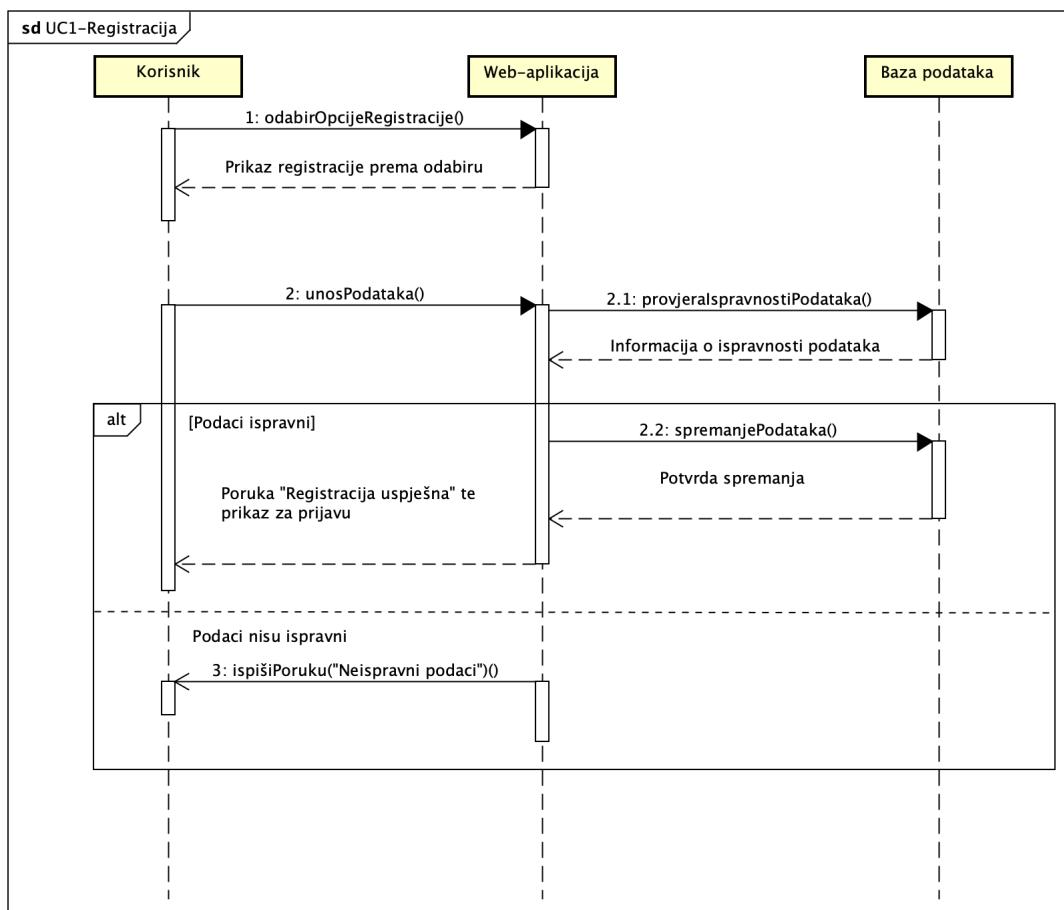
Slika 3.5: Dijagram obrasca uporabe, funkcionalnost posjetitelja

### 3.1.2 Sekvencijski dijagrami

#### Obrazac uporabe UC1 - Registracija

Korisnik prilikom procesa registracije odabire opciju registracije: Posjetitelj ili Organizator. Na temelju odabira, korisnik dohvaća obrazac za registraciju. Nakon ispunjavanja obrasca, korisnik šalje zahtjev za registraciju. Nakon provjere ispravnosti unesenih podataka moguće su dvije opcije:

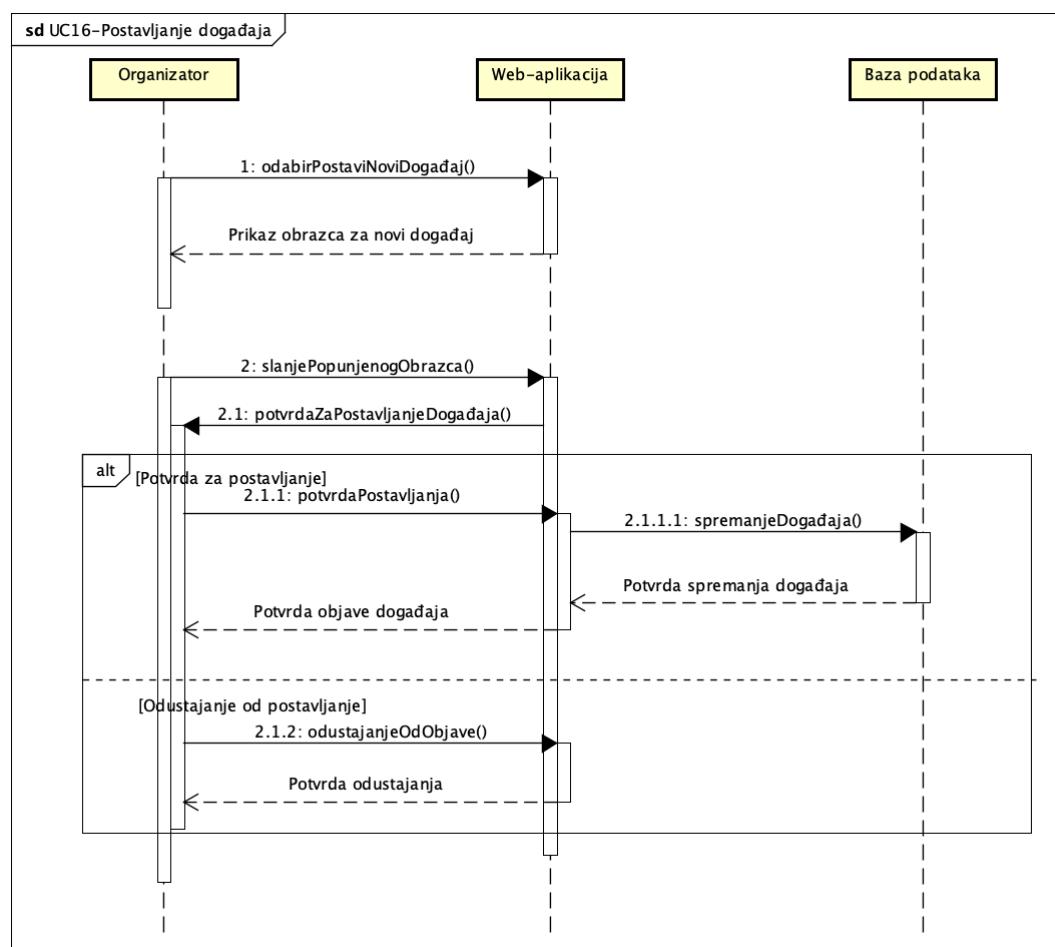
- Podaci ispravni: u slučaju ispravnih podataka vrši se spremanje korisničkih podataka te korisnik prima poruku "Registracija uspješna" te slijedi prikaz za prijavu
- Podaci neispravni: Korisnik prima poruku da su podaci neispravni te može unijeti podatke ispočetka



Slika 3.6: Sekvencijski dijagram za UC1

## Obrazac uporabe UC16 - Postavljanje događaja

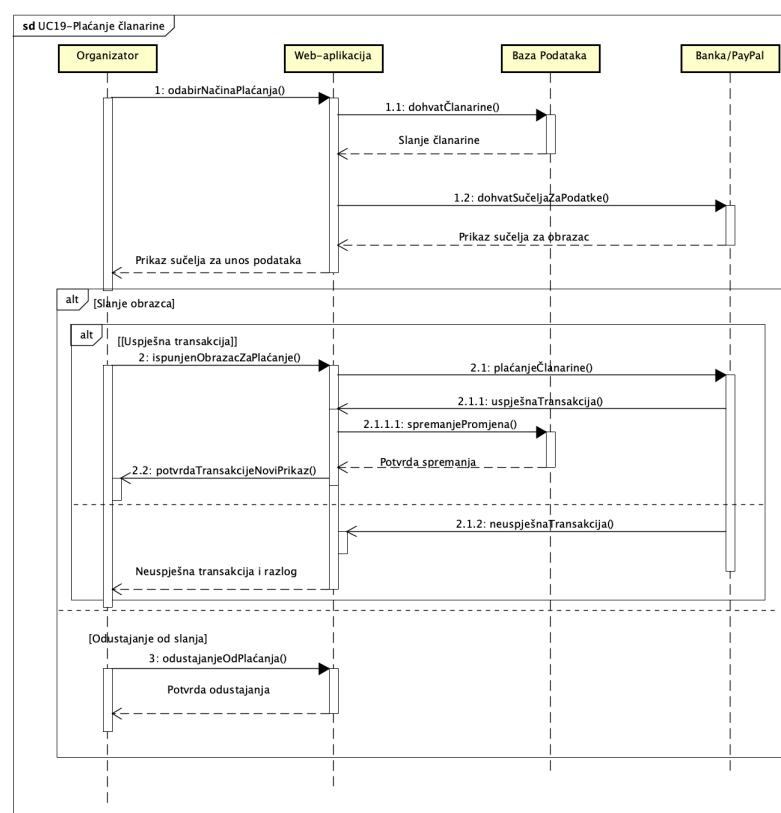
Tijekom procesa postavljanja događaja Organizator ima mogućnost izabrati objavu bez plaćanje ulaznice za događaj ili uz plaćanje. U slučaju postavljanja događaja bez plaćanja ulaznice odmah dohvaća obrazac za novi događaj. Nakon slanja popunjenoj obrasca sa svim potrebnim informacijama, Organizator zaprima potvrdu za postavljanje događaja te ima mogućnost potvrditi ili odustati od objave. Ukoliko dođe do potvrde, informacije o događaju se spremaju te se isti objavljuje, u suprotnom Organizator dobiva potvrdu odustajanja.



Slika 3.7: Sekvencijski dijagram za UC16

## Obrazac uporabe UC19 - Plaćanje članarine

Tijekom procesa objave događaja moguće je izabrati opciju za plaćanje ulaznice te u tom slučaju prije slanja obrasca za postavljanje novog događaja dolazi do plaćanje članarine ukoliko je potrebno. Organizator odabire način plaćanja te mu se prikazuje sučelje za unos podataka od izabranog izvora sa trenutnom cijenom članarine. U ovom trenutku Organizator može nastaviti sa procesom plaćanja ili odustati. U slučaju odustajanja dobiva potvrdu, a u suprotnom šalje ispunjen obrazac za plaćanje. Na temeju poslanog obrasca se izvršava plaćanje koje može biti uspješno ili neuspješno. Nakon uspješnog plaćanja se navedene promjene spremaju te Organizator dobiva potvrdu transakcije i novi prikaz. U slučaju neuspješne transakcije Organizator dobiva poruku i razlog neuspješne transakcije.



Slika 3.8: Sekvencijski dijagram za UC19

### 3.2 Ostali zahtjevi

- Sustav treba omogućiti brzo i učinkovito pregledavanje aktualnih događanja, uz minimalno vrijeme odziva aplikacije.
- Korisničko sučelje i sustav moraju podržavati različite vrste događanja i inicijativa, prilagođene različitim interesima zajednice.
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od nekoliko sekundi kako bi se osigurala brza dostupnost informacija.
- Sustav treba biti implementiran kao web-aplikacija koristeći suvremene tehnologije i objektno-orientirane jezike.
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava, a sustav treba pružiti jasne upute korisnicima o načinu korištenja.
- Sustav treba podržavati različite načine plaćanja članarine, uključujući PayPal i kreditne kartice, te osigurati sigurnost transakcija.
- Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške kako bi se očuvala integritet podataka.
- Pristup sustavu mora biti omogućen iz javne mreže pomoću HTTPS kako bi se osigurala sigurna komunikacija između korisnika i sustava.
- Administratori sustava trebaju imati alate za postavljanje cijena članstva, upravljanje korisnicima i održavanje cjelokupnog sustava.

## 4. Arhitektura i dizajn sustava

Arhitektura našeg sustava temeljiti će se na kombinaciji modernih tehnologija kako bismo ostvarili funkcionalan i skalabilan sustav. Sustav će se podijeliti na tri ključna podsustava:

- *Web poslužitelj*
- *Web klijent*
- *Baza podataka*



Slika 4.1: Arhitektura sustava

Organizacija će biti usklađena s MVC (Model-View-Controller) konceptom, što će omogućiti neovisnost između različitih dijelova sustava te olakšati razvoj, ispitivanje i dodavanje novih funkcionalnosti.

*Web preglednik*, kao program koji omogućuje korisnicima pregled web-stranica i multimedijalnih sadržaja, igra ključnu ulogu kao sučelje između web klijenta i web aplikacije. Klijenti, putem web preglednika, šalju zahtjeve web poslužitelju kako bi pristupili željenim resursima.

*Web poslužitelj* je osnova rada web aplikacije i odgovoran je za komunikaciju između klijenta i aplikacije. Komunikacija se odvija putem HTTP protokola, koji omogućuje prijenos informacija na webu. Poslužitelj pokreće web aplikaciju i prosljeđuje joj zahtjeve koje prima od klijentata.

*Web aplikacija*, smještena na poslužitelju, obrađuje zahtjeve korisnika. Ovisno o tim zahtjevima, pristupa poslužitelju baze podataka i na temelju dobivenih podataka, putem web poslužitelja, šalje odgovor korisnicima. Aplikacija se sastoji od backend i frontend dijela, koji se razvijaju koristeći različite tehnologije u skladu s

opisanom arhitekturom sustava. Konkretno, za backend dio koristit ćemo Spring, dok će za frontend biti korišten React, što je u skladu s MVC načelima.

Ključna karakteristika arhitekturnog obrasca MVC-a je nezavisan razvoj pojedinih dijelova aplikacije. Ova karakteristika rezultira jednostavnijim ispitivanjem, kao i olakšanim razvojem i dodavanjem novih svojstava u sustav.

MVC koncept sastoji se od tri osnovna dijela:

- **Model:** središnja komponenta sustava koja predstavlja dinamičke strukture podataka neovisne o korisničkom sučelju. Model upravlja podacima, logikom i pravilima aplikacije, primajući ulazne podatke od Controllera
- **View:** odgovara za prikaz podataka, poput grafičkih elemenata. Moguće je različiti prikaz istih informacija, kao što su grafički ili tablični prikazi podataka
- **Controller:** primarno prima ulazne podatke od korisnika i prilagođava ih za daljnju interakciju s Modelom ili Viewom. Kontrolira korisničke zahtjeve i izvodi daljnju interakciju s ostalim elementima sustava

Ovaj pristup omogućuje jasnu organizaciju sustava i olakšava daljnji razvoj i održavanje.

## 4.1 Baza podataka

Kao sustav za upravljanje bazama podataka za našu aplikaciju odabrali smo PostgreSQL. To je otvoren sustav za upravljanje bazama koji omogućuje pohranu, upravljanje i analizu podataka.

Sama baza podataka sadržavat će sedam tablica. One će služiti za pohranu podataka o korisnicima te njihovom korištenju aplikacije. Korištenje će se spremati u obliku događaja koje stvaraju ili posjećuju te recenzija koje pišu za određene događaje. Isto tako spremat će se sama zainteresiranost za događaje te notifikacije koje posjetitelji žele primati za događaje s određenim karakteristikama.

Sami odabir PostgreSQL-a je u njegovoj pouzdanosti, skalabilnosti i podrške za napredne SQL funkcionalnosti, što će nam omogućiti lakše te učinkovitije upravljanje podacima. Baza podataka ove aplikacije sastoji se od sljedećih entiteta:

- Korisnik
- Organizator

- Događaj
- Recenzija
- Zainteresiranost
- Notifikacija
- Misc

#### 4.1.1 Opis tablica

**Korisnik** - ovaj entitet sadržava sve važne informacije o korisniku aplikacije. Sadrži atribute: korisnikId koji je automatski dodijeljen, username, password, email i uloga u aplikaciji. Ovaj entitet u vezi je Many-to-One s entitetom organizator preko atributa organizerId.

korisnik		
korisnikId	INT	Primarni ključ tablice, dodjeljuje se automatski
username	VARCHAR	Jedinstveno ime koje korisnik odabire tijekom registracije
password	VARCHAR	Lozinka koju korisnik odabire tijekom registracije
email	VARCHAR	Jedinstveni email korisnika
uloga	SMALL INT/INT	Uloga koju korisnik ima unutar aplikacije(admin, posjetitelj, organizator)

**Organizator** - Ovaj entitet sadržava informacije o organizatoru događaja. Sadrži atribute: organizerId, nazivOrganizacije, adresa, poveznica, clanarina. Ovaj entitet u vezi je One-to-Many s entitetom dogadaj preko atributa organizerId.

organizator		
organizerId	INT	Primarni ključ tablice, ujedno i strani ključ iz tablice korisnik

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

organizator		
nazivOrganizacije	VARCHAR	Naziv organizacije kojoj pripada organizator
adresa	VARCHAR	Adresa organizacije
poveznica	VARCHAR	Poveznica na web stranicu organizacije
clanarina	BOOLEAN	Plaćena članarina

**Događaj** - ovaj entitet sadržava informacije o događaju. Sadrži atribute: dogadajId, organizatorId, nazivDogadaja, vrsta, lokacija, trajanje, vrijemePocetka, cijenaUlaznice, opis, galerija. Ovaj entitet u vezi je Many-to-One s entitetom Organizator preko atributa organizatorId te u vezi One-to-Many s entitetom recenzija preko atributa dogadajId.

dogadaj		
dogadajId	INT	Primarni ključ tablice, dodjeljuje se automatski
organizatorId	INT	Strani ključ iz tablice organizator koji se odnosi na njegov ID
nazivDogadaja	VARCHAR	Naziv događaja
vrsta	VARCHAR	Vrsta događaja(koncert, kazališna predstava ...)
lokacija	VARCHAR	Lokacija događaja
opisLokacije	VARCHAR	Kratki opis lokacije, adresa, kat ...
trajanje	INTERVAL	Trajanje događaja u danima
vrijemePocetka	TIMESTAMP	Datum i vrijeme početka događaja
cijenaUlaznice	NUMERIC	Cijena ulaznice na događaj
opis	VARCHAR	Kratki opis samog događaja
galerija	VARCHAR	Link do web stranice gdje se nalaze video snimke ili fotografije

**Recenzija** - ovaj entitet sadržava informacije o recenziji događaja. Sadrži atri-

bute: recenzijaId, korisnikId, dogadajId, tekst, ocjena. Ovaj entitet u vezi je Many-to-One s entitetom korisnik preko atributa korisnikId te u vezi Many-to-One s entitetom događaj preko atributa dogadajId.

recenzija		
recenzijaId	INT	Primarni ključ tablice, dodjeljuje se automatski
korisnikId	INT	Strani ključ iz tablice korisnik koji se odnosi na njegov ID
dogadajId	INT	Strani ključ iz tablice dogadaj koji se odnosi na njegov ID
tekst	VARCHAR	Kratki tekst recenzije
ocjena	SMALL INT/INT	Ocjena od 1-5, opcionalna

**Zainteresiranost** - ovaj entitet sadržava informacije o zainteresiranosti posjetitelja za određeni događaj. Sadrži atrbute: zainteresiranostId, posjetiteljId, dogadajId, kategorija. Ovaj entitet u vezi je Many-to-One s entitetom korisnik preko atributa posjetiteljId te u vezi Many-to-One s entitetom dogadaj preko atributa dogadajId.

zainteresiranost		
zainteresiranostId	INT	Primarni ključ tablice, dodjeljuje se automatski
posjetiteljId	INT	Strani ključ iz tablice korisnik koji se odnosi na njegov ID
dogadajId	INT	Strani ključ iz tablice dogadaj koji se odnosi na njegov ID
kategorija	SMALL INT/INT	Jedna od tri kategorije: Sigurno dolazim, možda dolazim ili ne dolazim

**Notifikacija** - ovaj entitet sadržava informacije o notifikacijama koje će posjetitelji primati vezano uz tražene događaje filtrirane po vrsti ili lokaciji. Sadrži atrbute: notifikacijaId, posjetiteljId, vrsta, lokacija. Ovaj entitet u vezi je Many-to-One

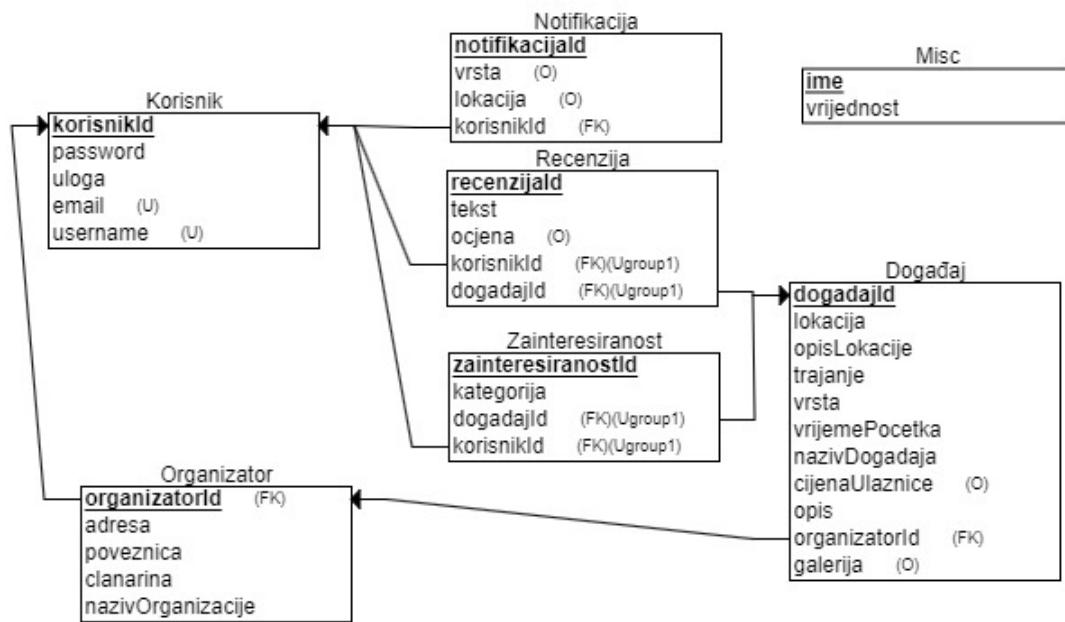
s entitetom korisnik preko atributa korisnikId te u vezi Many-to-One s entitetom dogadaj preko atributa vrsta i lokacija.

<b>notifikacija</b>		
notifikacijaId	INT	Primarni ključ tablice, dodjeljuje se automatski
posjetiteljId	INT	Strani ključ iz tablice korisnik koji se odnosi na njegov ID
vrsta	VARCHAR	Vrsta događaja za koji posjetitelj želi primiti obavijest o njegovu stvaranju, optionalno
lokacija	VARCHAR	Lokacija događaja za koji posjetitelj želi primiti obavijest o njegovu stvaranju, optionalno

**misc** - skraćeno od "miscellaneous", ovaj entitet služi za spremanje globalnih postavki aplikacija u obliku ključ-vrijednost. Sadrži attribute ime i vrijednost.

<b>misc</b>		
ime	VARCHAR	Ime (ključ) svojstva
vrijednost	VARCHAR	Vrijednost svojstva

#### 4.1.2 Dijagram baze podataka



Slika 4.2: Dijagram baze podataka

## 4.2 Dijagram razreda

Na slikama 4.3, 4.4, 4.5 i 4.6 su prikazani razredi koji pripadaju *backend* dijelu projekta te oni predstavljaju i odgovaraju stvarnom stanju implementacije sustava te su generirani iz same implementacije.

Razredi prikazani na slici 4.3 su razredi kontroleri. Ti razredi su ključni dijelovi koji upravljaju pristiglim zahtjevima, određuju rute te obraduju logiku aplikacije. Obično se koriste za implementaciju HTTP metoda poput **GET**, **POST**, **PUT** i **DELETE** za obrađivanje zahtjeva koji dolaze od klijenata. Razredi koji imaju svoje kontrolere su **Korisnik**, **Organizator**, **Događaj**, **Notifikacija** i **Transakcije**. Na primjeru KorisnikControllera ćemo detaljnije opisati metode i članske varijable koje su korištene za spajanje sa servisnim slojem.

**KorisnikController** ima metode:

- getAll - vraća listu Korisnika
- delete - kao argument prima ID Korisnika te ga briše ukoliko postoji
- validate - prima Korisnika te vraća ResponseEntity, služi validaciji Korisnika
- register - kao argument prima KorsnikDTO, a vraća ResponseEntity koji je string
- update - služi promjeni i ažuriranju podataka Korisnika

Isto ima i članske varijable tipa KorisnikService i OrganizatorServiceJpa preko kojih se obavlja poslovna logika (operacije nad podacima, pristupanje bazi podataka ili komunikacija s vanjskim servisima).

Razredi prikazani na slici 4.4 su razredi servisi. Razredi servisi u Spring Bootu predstavljaju komponente koje sadrže poslovnu logiku aplikacije, pristupaju podacima i obavljaju operacije koje nisu specifične za obradu HTTP zahtjeva. Oni često služe kao posrednici između kontrolera i sloja pristupa podacima (Repository sloj) te obavljaju ključne funkcije za obradu podataka, poslovnu logiku i vanjske integracije. U našem primjeru ostvarili smo prvo sučelja koja smo zatim implementirali. Razredi koji imaju servise su **Korisnik**, **Organizator**, **Događaj**,

**Notifikacija, Zainteresiranost, EmailSender, Misc i Recenzija.** Na primjeru KorisnikServicve ćemo detaljnije opisati metode i članske varijable.

**KorisnikService** sučelje ima metode:

- `listAll` - vraća listu Korisnika
- `findById` - prima id kao argument, a ako postoji Korisnik s tim id, njega vraća
- `findByUsername` - prima username kao argument, a ako postoji korisnik s tim usernamom njega vraća
- `findByEmail` - prima email kao argument, a ako postoji korisnik s tim emailem njega vraća
- `registerUser` - kao argument prima KorsnikDTO, a boolean ovisno o uspješnosti
- `updateUser` - kao argument prima KorsnikDTO i ID Korisnika, a boolean ovisno o uspješnosti
- `deleteUser` - kao argument prima Korisnika i briše ga ukoliko postoji

Implementacija sučelja KorisnikService, KorisnikServiceJpa, isto ima i članske varijable tipa KorisnikRepository i PasswordEncoder. Razred repozitorija u Spring Bootu predstavlja sloj pristupa podacima (data access layer) i obično se koristi za komunikaciju s bazom podataka. Ovi repozitoriji pružaju apstrakciju za operacije nad podacima, kao što su dohvaćanje, spremanje, ažuriranje ili brisanje podataka. Instanca PasswordEncoder služi nam za registraciju korisnika te za ažuriranje njegovih podataka kada dolazi do promjene lozinke.

Razredi prikazani na slici 4.5. su razredi modeli. Struktura baze podataka u aplikaciji odražava se upravo model razredima; osim što imaju svoje pripadne atribute, razredi također implementiraju metode koje ostvaruju interakciju s bazom podataka.

Razred Korisnik reprezentira općeg korisnika aplikacije. Specificira ga atribut "uloga" koja određuje je li korisnik administrator, posjetitelj ili organizator.

Razred Organizator reprezentira korisnika s organizatorskim računom. Organizatori mogu pregledavati svoje postojeće događaje i postavljati nove događaje.

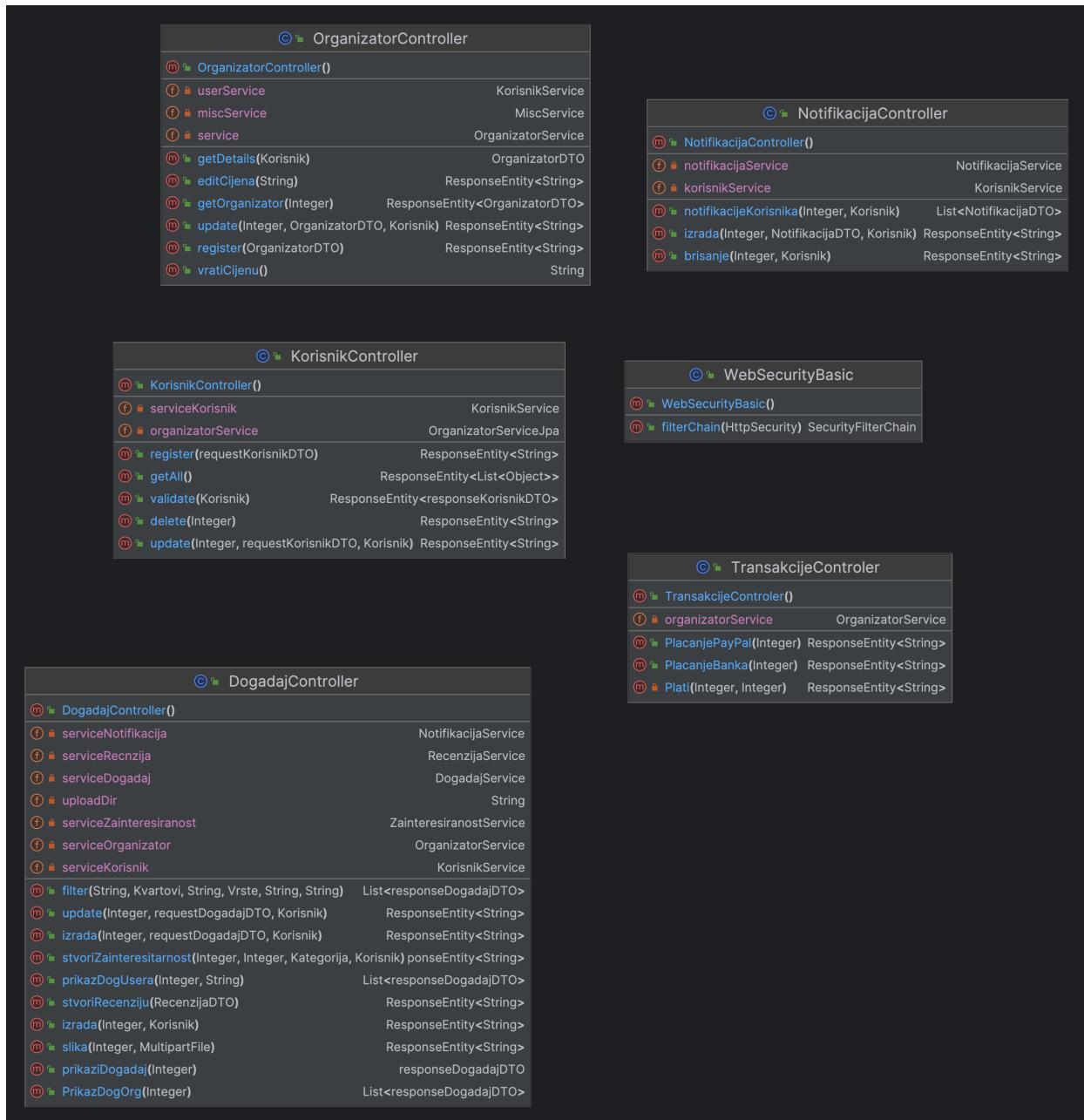
Razred Recenzija reprezentira korisnikov osvrt na događaj u obliku ocjene i komentara.

Razred Notifikacija reprezentira obavijest koja se šalje korisniku. Korisnik ima mogućnost uključiti notifikacije za određenu vrstu događaja te za lokaciju na kojoj se događaji organiziraju.

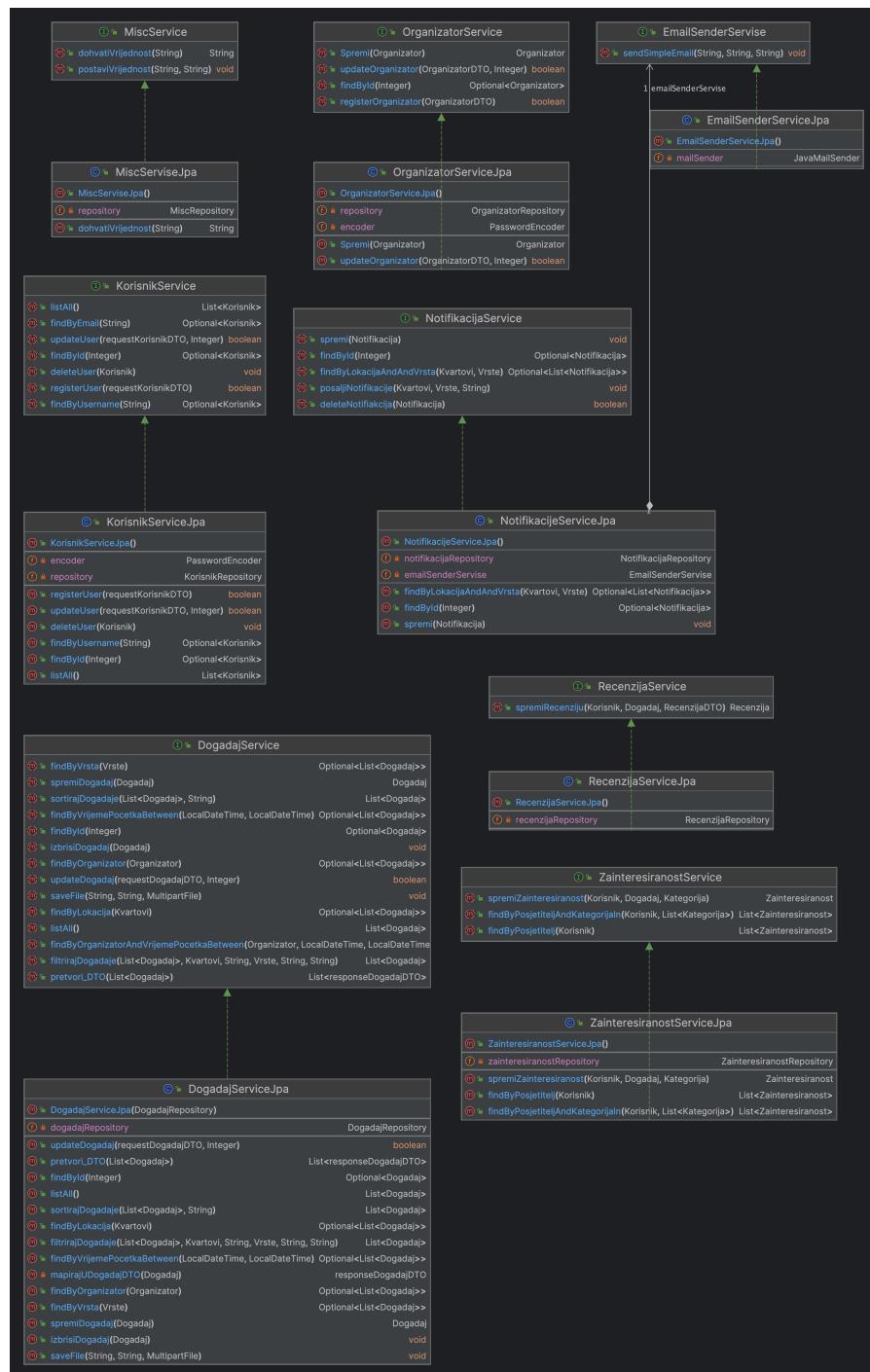
Razred Zainteresiranost reprezentira korisnikovu interakciju s postavljenim događajima. Korisnikova zainteresiranost ima tri razine: "sigurno dolazim", "možda dolazim", "ne dolazim".

Razred Dogadaj reprezentira događaj koji je postavio organizator. Lokacija događaja predstavljena je gradskom četvrti - kvartom.

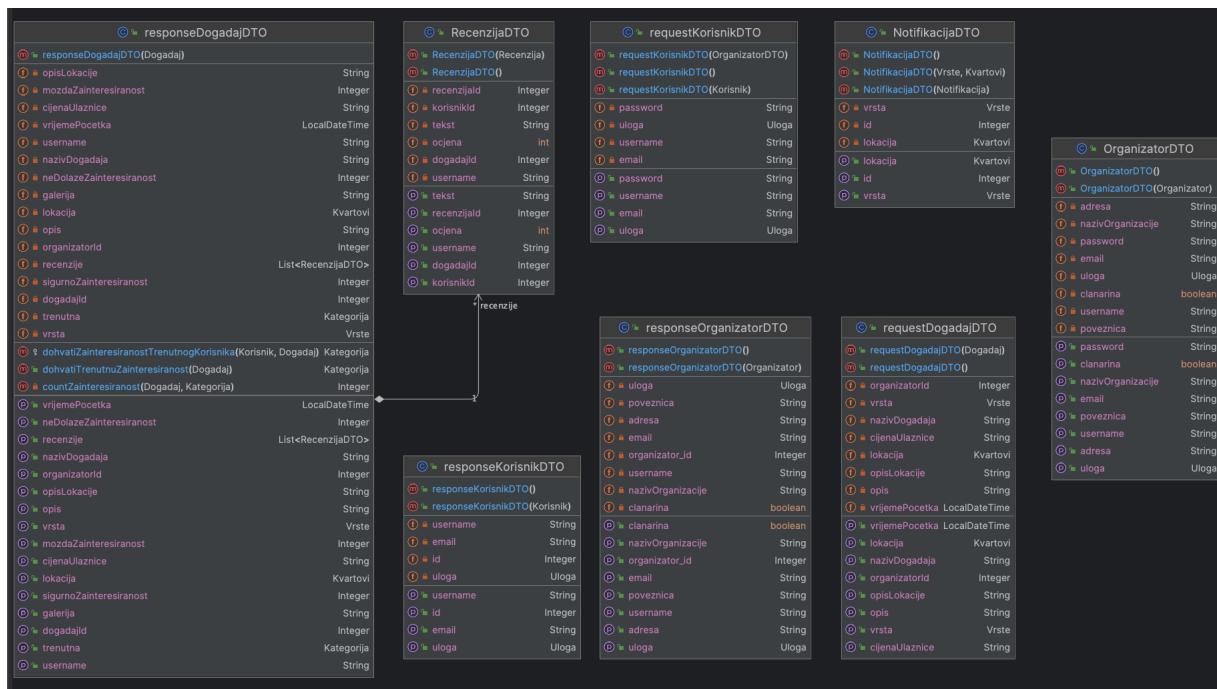
Razred Misc skraćeno od "miscellaneous", ovaj razred služi za spremanje globalnih postavki aplikacija u obliku ključ-vrijednost. Sadrži atribute ime i vrijednost. Osim navedenih razreda na slici su također prikazane enumeracije: Kategorija - kategorija zainteresiranosti korisnika za događaj, Kvartovi - kvartovi grada Zagreba gdje se organiziraju događaji, Uloga - uloga korisnika i Vrste - različite vrste događaja



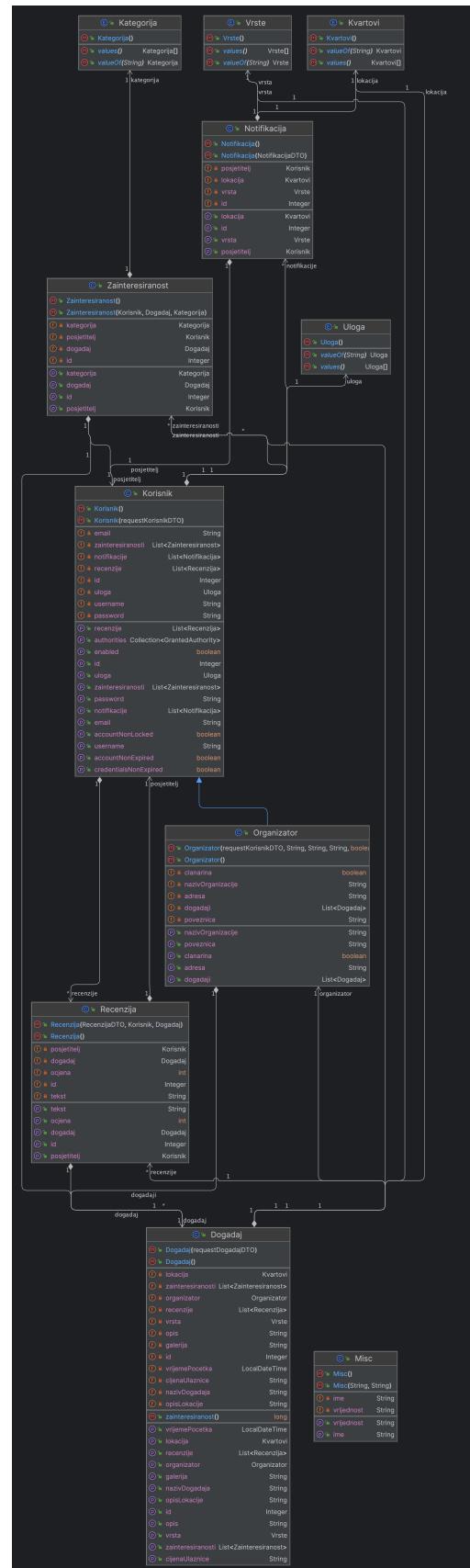
Slika 4.3: Dijagram razreda - Kontroleri



Slika 4.4: Dijagram razreda - Servisi



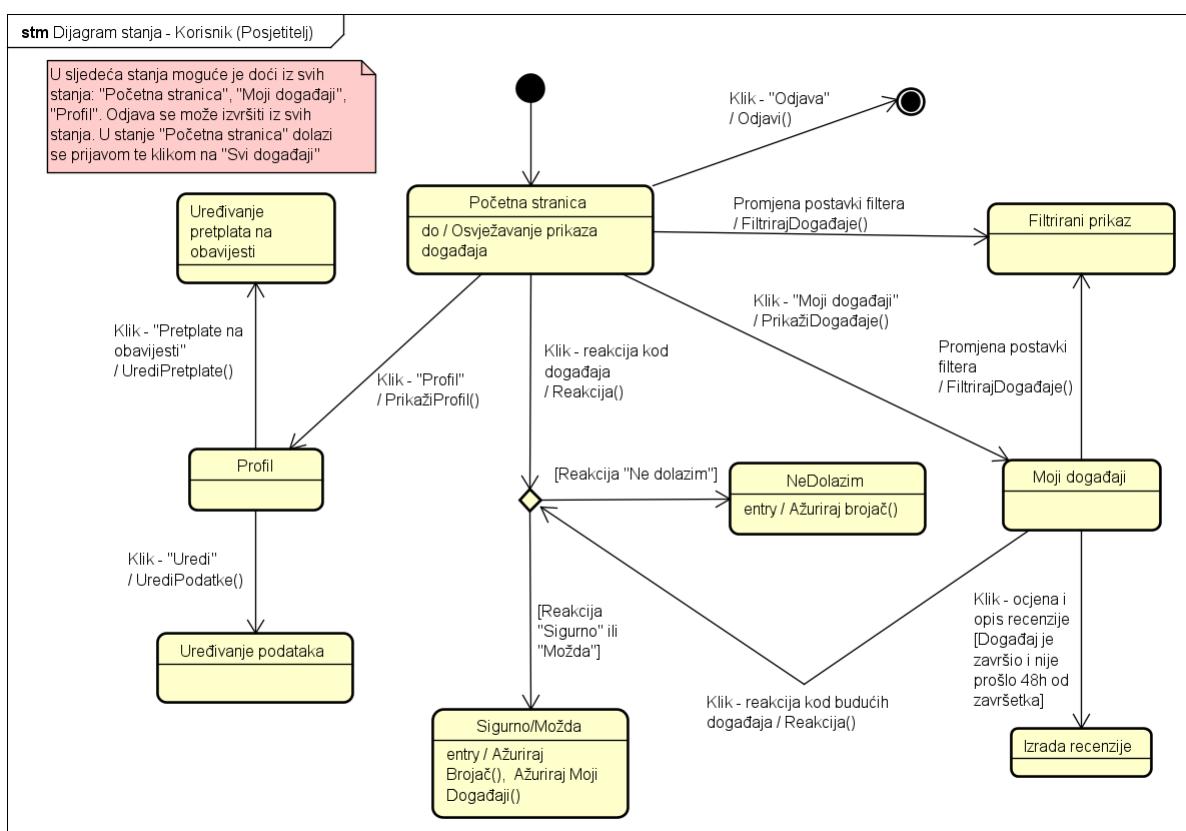
Slika 4.5: Dijagram razreda - Data Transfer Objects



Slika 4.6: Dijagram razreda - Models

## 4.3 Dijagram stanja

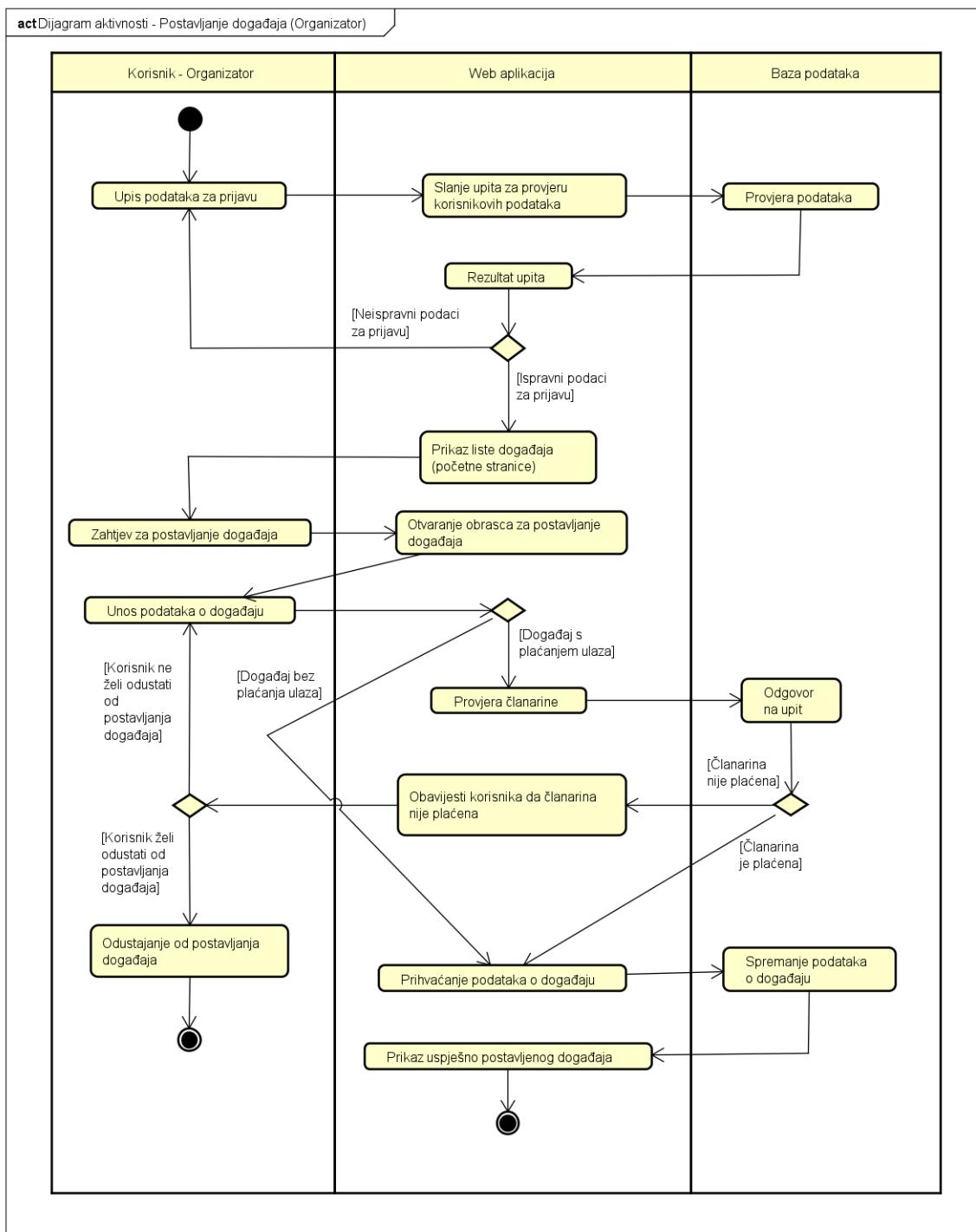
Dijagram stanja opisuje dinamičko ponašanje dijela sustava u vremenu; prikazuje stanja objekta i prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici 4.7 prikazan je dijagram stanja za korisnika - posjetitelja. Kada se korisnik prijavlji prikazuje mu se početna stranica na kojoj su vidljivi događaji, filter pomoću kojeg može odlučiti koje događaje želi pregledati, te navigacijska traka preko kojeg pristupa svim događajima, događajima za koje je označio "Sigurno dolazim" ili "Možda dolazim" te profilu. Klikom na "Moji događaji" prikazuju mu se događaji na koje je prethodno reagirao sa "Sigurno dolazim" ili "Možda dolazim". Ako među ovim događajima postoji događaj koji je završili i nije prošlo 48 sati od njihovog završetka, korisnik za njih može izraditi recenziju. Klikom na korisničko ime - profil prikazuje mu se stranica profila na kojoj može promijeniti osobne podatke te urediti postavke pretplate na obavijesti.



Slika 4.7: Dijagram stanja - Korisnik (Posjetitelj)

## 4.4 Dijagram aktivnosti

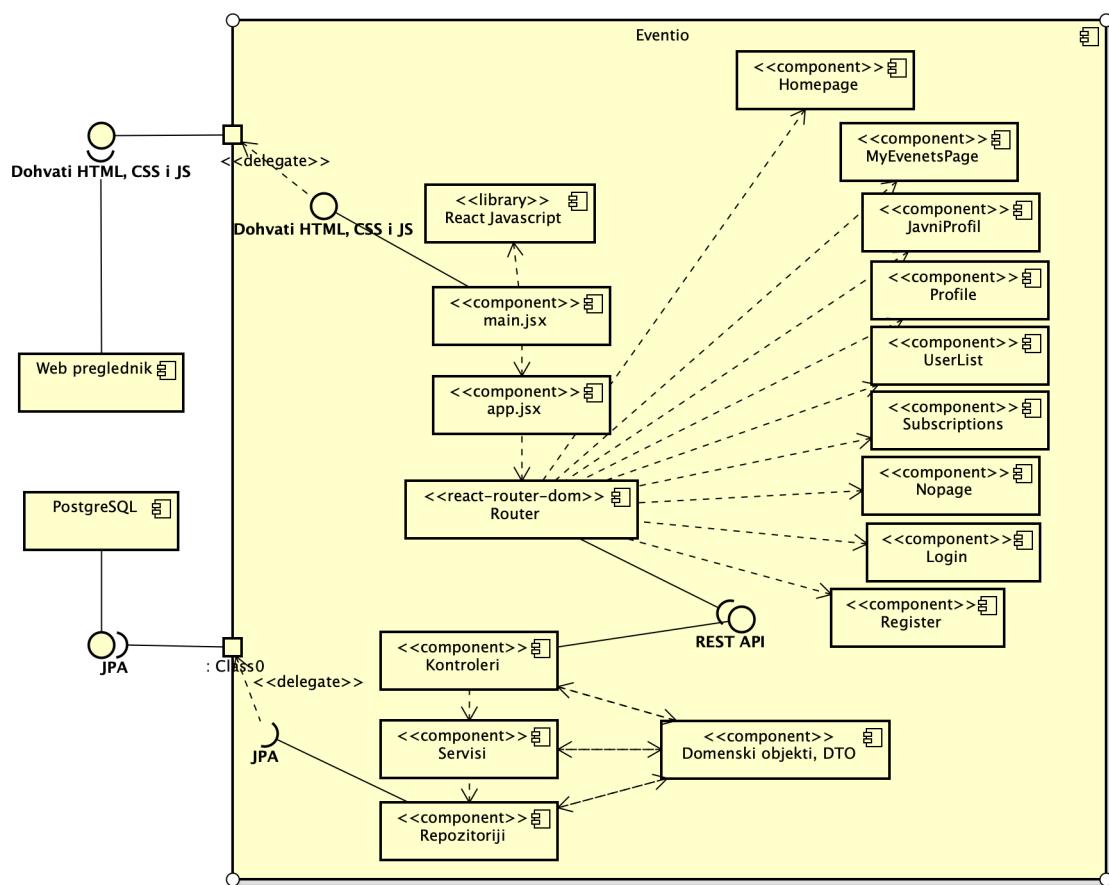
Dijagram aktivnosti primjenjuje se za modeliranje poslovnih procesa, upravljačkog i podatkovnog toka. Na slici 4.8 prikazan je dijagram aktivnosti za proces stvaranje novog događaja. Nakon prijave organizator na početnoj stranici odabire opciju postavljanja događaja. Otvara mu se stranica za postavljanje događaja na kojoj unosi sve potrebne podatke o događaju. Ako organizator želi događaj bez plaćanja ulaza, podaci o događaju se odmah procesiraju, spremaju u bazu te se organizatoru prikazuje potvrda o uspješno postavljenom događaju. Ako organizator želi događaj koji će imati plaćanje ulaza, provodi se provjera članarine. Ako je članarina plaćena, proces se nastavlja jednako kao i u slučaju besplatnog događaja. Ako nije, aplikacija obavješta organizatora da članarina nije plaćena te organizator ima dvije mogućnosti: ponovno unijeti podatke o događaju i početi proces postavljanja ispočetka ili odustati od postavljanja događaja.



Slika 4.8: Dijagram aktivnosti - Postavljanje događaja (Organizator)

## 4.5 Dijagram komponenti

Dijagram komponenti na slici opisuje organizaciju, međuovisnost komponenti i odnose prema okolini - prikazuje komponente aplikacije i glavna korištena sučelja. Web aplikacija Korisnik pristupa aplikaciji koristeći web preglednik te preko sučelja za dohvaćanje HTML-a, CSS i JS-a poslužuje mu se frontend dio aplikacije. Pomoću REST API sučelja komunicira te mu se poslužuju podaci povezani s backend dijelom. Svaka od opisanih komponenti može imati ovisnosti prema drugim resursima, tako je backend dio povezan sa bazom podataka korištenjem Jakarta Persistence API - JPA. Web aplikacija Eventio prati MVC arhitekturu te stoga postoji distinkcija između frontend i backend dijela aplikacije što je vidljivo i na ovom dijagramu. "View" dio arhitekture je frontend u kojem koristimo React library koji nam omogućuje podjelu na više komponenata te se brine o njihovom spašanju. App.jsx komponenta web aplikacije brine o prijavi korisnika te ovisno o ulozi korisnika Router prikazuje potrebne komponente. Također su prikazane komponente backend sloja, redom: Kontroleri, Servisi i Repozitoriji. Predstavljaju "Model" i "Controller" dio MVC arhitekture te je u dijagramu prikazana njihova međuovisnost.



Slika 4.9: Dijagram komponenti

# 5. Implementacija i korisničko sučelje

## 5.1 Korištene tehnologije i alati

Za međusobnu komunikaciju unutar tima koristili smo aplikacije WhatsApp i Discord. *WhatsApp* je popularna platforma koja omogućava korisnicima slanje poruka i medijskih datoteka te obavljanje glasovnih/video poziva. *Discord* je aplikacija za stvaranje servera na kojima članovi mogu komunicirati putem poruka i poziva.

Za modeliranje UML dijagrama koristili smo *Astah UML*, alat koji omogućava korisnicima da kreiraju različite vrste UML dijagrama. Za pripremu i uređivanje dokumentacije koristili smo *TeXstudio*, specifično prilagođeno okruženje za učinkovit rad s LaTeX sustavom.

Po pitanju baze podataka i svega vezanog uz nju koristili smo nekoliko alata. Alatom *ERDplus*, popularnim online alatom za izradu ER dijagrama, modelirali smo ER dijagram. Sama baza podataka temeljena je na *PostgreSQL*-u, poznatom sustavu za upravljanje bazama podataka. *Firebase* je platforma koja nam služi za pohranu slika koje organizatori postavljaju uz svoje događaje. Alat *Liquibase* koristili smo za upravljanje promjenama u shemi baze podataka.

Za ispitivanje komponenti iskoristili smo *JUnit*, okvir za testiranje osmišljen za podršku automatskom testiranju Java aplikacija, i *Mockito*, koji nam je poslužio za stvaranje lažnih, mock objekata. Za ispitivanje sustava upotrijebili smo *Selenium IDE*, alat koji omogućuje snimanje, uređivanje i reprodukciju testova jednostavnim klikanjem i snimanjem korisničkih radnji na web stranici.

Za upravljanje izvornim kodom primijenili smo sustav *Git*, dok se sam kod nalazi na *GitHub* platformi unutar udaljenog repozitorija. GitHub je jedna od najpoznatijih web platformi koja pruži usluge za upravljanje projektima temeljenima na Git sustavu.

Naposjetku, za izradu aplikacije upotrijebili smo mnogo različitih alata. Za frontend smo koristili *React.js* uz *JavaScript*, *Vite* te *Material UI*. React.js je popularna JavaScript biblioteka koja se koristi za izgradnju korisničkih sučelja u web aplikacijama. Vite je alat čija je osnovna svrha pojednostaviti i ubrzati razvoj web

aplikacija. Material UI je poslužio za dizajniranje osnovnih komponenti frontenda. Za backend smo koristili *Javu* i *Spring Boot*, okvir za izgradnju Java aplikacija temeljenih na Springu, a koji je posebno koristan jer omogućuje programerima da se fokusiraju na poslovnu logiku aplikacije umjesto na konfiguraciju i upravljanje okolinom

Deployment frontenda, backenda te baze podataka postigli smo uz pomoć platforme *Render* koja služi za deployment i hosting web aplikacija.

---

- Whatsapp <https://www.whatsapp.com/>
- Discord <https://discord.com/>
- Astah UML <https://astah.net/products/astah-uml/>
- TeXstudio <https://www.texstudio.org/>
- ERDplus <https://erdplus.com/>
- PostgreSQL <https://www.postgresql.org/>
- Firebase <https://firebase.google.com/>
- Liquibase <https://www.liquibase.org/>
- JUnit <https://junit.org/junit5/>
- Mockito <https://site.mockito.org/>
- Selenium IDE <https://www.selenium.dev/selenium-ide/>
- Git <https://git-scm.com/>
- GitHub <https://github.com/>
- React.js <https://reactjs.org/>
- Vite <https://vitejs.dev/>
- JavaScript <https://www.javascript.com/>
- Material UI <https://mui.com/>
- Java <https://www.java.com/en/>
- SpringBoot <https://spring.io/projects/spring-boot/>
- Render <https://render.com/>

## 5.2 Ispitivanje programskog rješenja

U ovom poglavlju usredotočili smo se na testiranje programskog rješenja testirajući cijeli sustav, ali i komponente istog. Tijekom ispitivanja komponenti sustava korišteni su JUnit za izvršavanje testova te Mockito za stvaranje mock objekata korištenih u testiranju. Selenium IDE korišten je u ispitivanju sustava te nam je omogućio brzo i precizno provjeravanje funkcionalnosti web aplikacije.

### 5.2.1 Ispitivanje komponenti

U poglavlju o ispitivanju komponenti, razvijeni su i izvršeni testovi kako bi se osigurala kvaliteta i funkcionalnost komponenata sustava, a istovremeno se time osigurava pouzdanost i održivost koda. U nastavku su prikazani primjeri nekoliko testova koji su implementirani kao dio ispitivanja komponenti sustava:

**Test sortiranja i ispisivanja događaja** provjerava ispravnu funkcionalnost sortiranja događaja prema određenom filtru, u ovom slučaju prema vremenu početka događaja. Također testirano je izlistavanje događaja, je li broj događaja jednak očekivanom broju. Korišten je JUnit za pisanje i izvršavanje testova te Mockito za stvaranje mock objekta i postavljanje očekivanja.

**Test registracije korisnika** ispituje uspješnost registracije korisnika. Također se koriste Mock objekti koji simuliraju ponašanje komponenti tijekom registracije. Prikazana su dva testa, u jednom se testira registracija koja je uspješna, u drugom se testira neuspješna registracija korisnika.

U nastavku su još prikazani sljedeći testovi: **Test slanja e-maila**, **Test spremanja recenzije**, **Test spremanja zainteresiranosti**. Oni redom provjeravaju točnost slanja e-maila, ispravnu funkcionalnost spremanja recenzije i zainteresiranosti. U navedenim testovima se također koriste spomenute tehnologije za provjeravanje ispravnosti. Osim testova, prikazano je i njihovo izvođenje te rezultat izvođenja.

```
@Test
public void testListAll() {
    List<Dogadaj> dummyDogadaji = Arrays.asList(new Dogadaj(), new Dogadaj());
    when(dogadajRepository.findAll()).thenReturn(dummyDogadaji);

    List<Dogadaj> result = dogadajService.listAll();

    assertEquals(expected: 2, result.size());
}

▲ Filip Buhiniček
@Test
public void testSortirajDogadaje() {
    Dogadaj dogadaj1 = new Dogadaj();
    dogadaj1.setVrijemePocetka(LocalDateTime.now().plusDays(2));

    Dogadaj dogadaj2 = new Dogadaj();
    dogadaj2.setVrijemePocetka(LocalDateTime.now().plusDays(1));

    Dogadaj dogadaj3 = new Dogadaj();
    dogadaj3.setVrijemePocetka(LocalDateTime.now().plusDays(3));

    List<Dogadaj> dummyDogadaji = Arrays.asList(dogadaj1, dogadaj2, dogadaj3);

    when(dogadajRepository.findAll()).thenReturn(dummyDogadaji);

    List<Dogadaj> result = dogadajService.sortirajDogadaje(dogadajService.listAll(), sort: "vrijeme-silazno");

    assertEquals(dogadaj3, result.get(0));
    assertEquals(dogadaj1, result.get(1));
    assertEquals(dogadaj2, result.get(2));
}
```

Slika 5.1: Test za sortiranje i ispisivanje događaja

```

@Test
void testRegisterUser_Success() {
    requestKorisnikDTO mockDTO = new requestKorisnikDTO();
    mockDTO.setUsername("user");
    mockDTO.setPassword("password");
    mockDTO.setEmail("test@example.com");

    when(encoder.encode(mockDTO.getPassword())).thenReturn( t: "encodedPassword");
    when(repository.saveAndFlush(any(Korisnik.class))).thenReturn(new Korisnik(mockDTO));

    boolean result = korisnikService.registerUser(mockDTO);

    assertTrue(result);
}

▲ Filip Buhiniček
@Test
void testRegisterUser_Failure() {
    requestKorisnikDTO mockDTO = new requestKorisnikDTO();
    mockDTO.setUsername("user");
    mockDTO.setPassword("password");
    mockDTO.setEmail("test@example.com");

    when(encoder.encode(mockDTO.getPassword())).thenReturn( t: "encodedPassword");
    doThrow(new RuntimeException("Simulated failure")).when(repository).saveAndFlush(any(Korisnik.class));

    boolean result = korisnikService.registerUser(mockDTO);

    assertFalse(result);
}

```

Slika 5.2: Test registracije korisnika

```

@Test
void testSendSimpleEmail() {
    String toEmail = "test@example.com";
    String subject = "Test Subject";
    String body = "Test Body";

    emailSenderService.sendSimpleEmail(toEmail, subject, body);

    verify(mailSender).send(createExpectedSimpleMailMessage(toEmail, subject, body));
}

1 usage ▲ Filip Buhiniček
private SimpleMailMessage createExpectedSimpleMailMessage(String toEmail, String subject, String body) {
    SimpleMailMessage expectedMessage = new SimpleMailMessage();
    expectedMessage.setFrom("eventio.progi@gmail.com");
    expectedMessage.setTo(toEmail);
    expectedMessage.setSubject(subject);
    expectedMessage.setText(body);
    return expectedMessage;
}

```

Slika 5.3: Test slanja e-maila

```
@Test
void testSpremiRecenziju() {
    Korisnik korisnik = new Korisnik();
    Dogadaj dogadaj = new Dogadaj();
    RecenzijaDTO recenzijaDTO = new RecenzijaDTO();
    recenzijaDTO.setOcjena(5);
    recenzijaDTO.setTekst("Ovo je sjajan dogadjaj!");

    when(recenzijaRepository.save(any(Recenzija.class))).thenReturn(new Recenzija(recenzijaDTO,korisnik,dogadaj));

    Recenzija savedRecenzija = recenzijaService.spremiRecenziju(korisnik, dogadaj, recenzijaDTO);

    assertEquals(korisnik, savedRecenzija.getPosjetitelj());
    assertEquals(dogadaj, savedRecenzija.getDogadaj());
    assertEquals(recenzijaDTO.getOcjena(), savedRecenzija.getOcjena());
    assertEquals(recenzijaDTO.getTekst(), savedRecenzija.getTekst());

    verify(recenzijaRepository).save(any(Recenzija.class));
}
```

Slika 5.4: Test spremanja recenzije

```
@Test
void testSpremiZainteresiranost_Nova() {
    Korisnik posjetitelj = new Korisnik();
    Dogadaj dogadaj = new Dogadaj();
    Kategorija kategorija = Kategorija.SIGURNO;

    when(zainteresiranostRepository.findByPosjetiteljAndDogadaj(posjetitelj, dogadaj)).thenReturn(null);
    when(zainteresiranostRepository.save(any(Zainteresiranost.class)))
        .thenReturn(new Zainteresiranost(posjetitelj,dogadaj, kategorija));

    Zainteresiranost savedZainteresiranost = zainteresiranostService.spremiZainteresiranost(posjetitelj, dogadaj, kategorija);

    assertEquals(posjetitelj, savedZainteresiranost.getPosjetitelj());
    assertEquals(dogadaj, savedZainteresiranost.getDogadaj());
    assertEquals(kategorija, savedZainteresiranost.getKategorija());

    verify(zainteresiranostRepository).findByPosjetiteljAndDogadaj(posjetitelj, dogadaj);
    verify(zainteresiranostRepository).save(any(Zainteresiranost.class));
}
```

Slika 5.5: Test spremanja nove zainteresiranosti

```

@Test
void testSpremiZainteresiranost_Postojeća() {
    Korisnik posjetitelj = new Korisnik();
    Dogadaj dogadaj = new Dogadaj();
    Kategorija novaKategorija = Kategorija.SIGURNO;
    Zainteresiranost postojećaZainteresiranost = new Zainteresiranost(posjetitelj, dogadaj, Kategorija.MOZDA);

    when(zainteresiranostRepository.findByPosjetiteljAndDogadaj(posjetitelj, dogadaj))
        .thenReturn(postojećaZainteresiranost);
    when(zainteresiranostRepository.save(any(Zainteresiranost.class)))
        .thenReturn(new Zainteresiranost(posjetitelj, dogadaj, novaKategorija));

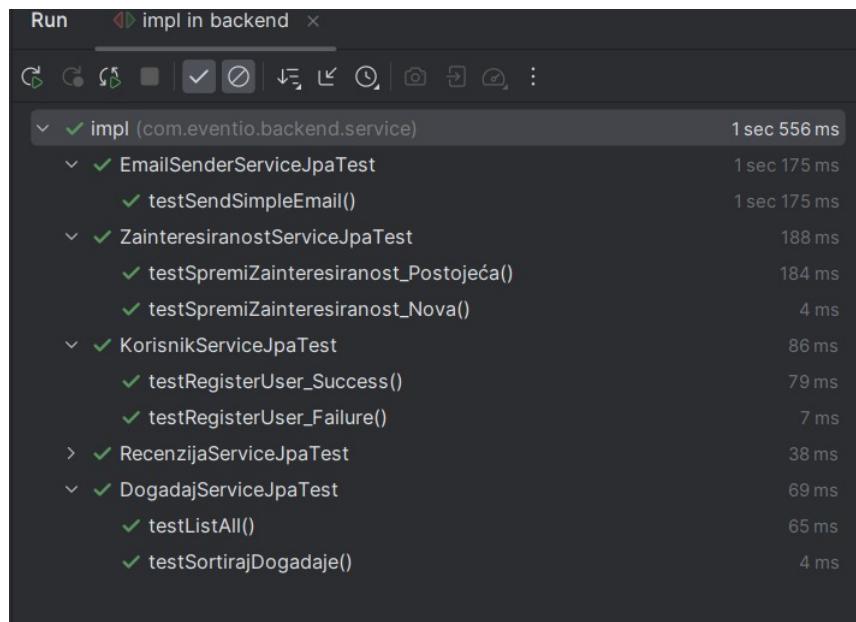
    Zainteresiranost savedZainteresiranost = zainteresiranostService.spremiZainteresiranost(posjetitelj, dogadaj, novaKategorija);

    assertEquals(posjetitelj, savedZainteresiranost.getPosjetitelj());
    assertEquals(dogadaj, savedZainteresiranost.getDogadaj());
    assertEquals(novaKategorija, savedZainteresiranost.getKategorija());

    verify(zainteresiranostRepository).findByPosjetiteljAndDogadaj(posjetitelj, dogadaj);
    verify(zainteresiranostRepository).save(any(Zainteresiranost.class));
}

```

Slika 5.6: Test spremanja postojeće zainteresiranosti



Slika 5.7: Izvršavanje i rezultati testova

### 5.2.2 Ispitivanje sustava

U procesu ispitivanja sustava koristili smo Selenium IDE koji nam je omogućio precizno i brzo testiranje funkcionalnosti web aplikacije. Selenium IDE predstavlja moćan alat za testiranje web aplikacija te se istovremeno olakšava proces automatizacije.

Na sljedećim slikama prikazani su ispitni slučajevi kojima smo testirali funkcionalnost aplikacije. **Test prijave organizatora** provjerava uspješnost prijave korisnika koji je organizator. **Test neuspješne prijave organizatora** provjerava neuspješnu prijavu korisnika koji je organizator. **Test stvaranja novog događaja** provjerava funkcionalnost stvaranja novog događaja sa svim postavkama. Prikazano je i izvršavanje **Testa promjene cijene pretplate** i **Testa stvaranja nove recenzije**.

Project: Eventio*			
Tests	+	Run	Stop
Search tests... <input type="text"/>			
AdminLogin			
✓ FailedOrgLogin*			
NovaRecenzija			
NoviDogadjaj			
✓ OrgLogin*			
PosLogin			
PromjenaCijenePreplate			
1	✓ open	/	
2	✓ set window size	973x685	
3	✓ click	id=r1	
4	✓ type	id=r1	org
5	✓ mouse over	css=MuiButton-contained	
6	✓ type	id=r3	org
7	✓ click	css=MuiButton-contained	
8	✓ click	css=MuiContainer-maxWidthLg	

Slika 5.8: Test uspješne prijave organizatora

Project: Eventio*			
Tests	+	Run	Stop
Search tests... <input type="text"/> Run current test Ctrl+R			
AdminLogin			
✓ FailedOrgLogin*			
NovaRecenzija			
NoviDogadjaj			
✓ OrgLogin*			
PosLogin			
PromjenaCijenePreplate			
1	✓ open	/	
2	✓ set window size	973x685	
3	✓ mouse down	id=r1	
4	✓ mouse up	id=r1-label	
5	✓ click	css=MuiFormControl-root:nth-child(1)	
6	✓ type	id=r1	org
7	✓ type	id=r3	Org
8	✓ click	css=MuiButton-contained	
9	✓ click	css=css-1y3h0mg	

Slika 5.9: Test neuspješne prijave organizatora

Project: Eventio*		
Executing	Run current test Ctrl+R	
<b>✓ NoviDogadjaj*</b>		
Command	Target	Value
1 ✓ open	http://localhost:3000/moji	
2 ✓ set window size	1936x1048	
3 ✓ click	css=MuiContainer-maxWidthLg	
4 ✓ click	css=MuiFab-root	
5 ✓ click	name=nazivDogadaja	
6 ✓ type	name=nazivDogadaja	Dogadjaj
7 ✓ mouse down	css=mui-component-select-vrsta	
8 ✓ click	css=body	
9 ✓ click	css=#3Afv3A > MuiButtonBase-root:nth-child(2)	SPORT
10 ✓ click	id=mui-component-select-lokacija	
11 ✓ click	css=body	
12 ✓ click	css=#3Afv113A > .MuiButtonBase-root:nth-child(2)	
13 ✓ click	name=opisLokacije	
14 ✓ type	name=opisLokacije	Ulica Doma Sportova
15 ✓ click	name=vrijemePocetka	
16 ✓ type	name=vrijemePocetka	2024-01-14T15:58
17 ✓ click	name=cijenaUlaznice	
18 ✓ type	name=cijenaUlaznice	0
19 ✓ click	name=opis	
20 ✓ type	name=opis	Evo opis
21 ✓ click	css=MuiButton-containedSizeLarge	

Slika 5.10: Test stvaranja novog događaja

Project: Eventio*		
Tests	+	
Search tests... <input type="text"/>		
✓ AdminLogin		
✓ NovaRecenzija*		
✓ NoviDogadjaj*		
✓ OrgLogin		
✓ PosLogin		
✓ PromjenaCijenePreplate		
Command	Target	Value
1 ✓ open	http://localhost:3000/moji	
2 ✓ set window size	1936x1048	
3 ✓ mouse over	css=MuiButton-contained	
4 ✓ click	css=MuiButton-contained	
5 ✓ mouse out	css=MuiButton-contained	
6 ✓ click	css=MuiButtonBase-root:nth-child(1) > MuiTypography-root	
7 ✓ mouse over	css=MuiButtonBase-root:nth-child(4)	
8 ✓ click	css=MuiButtonBase-root:nth-child(4)	
9 ✓ mouse out	css=MuiButtonBase-root:nth-child(5)	
10 ✓ type	id:rb:	10.99
11 ✓ click	id:rb:	
12 ✓ click	css=MuiButtonBase-root:nth-child(6)	

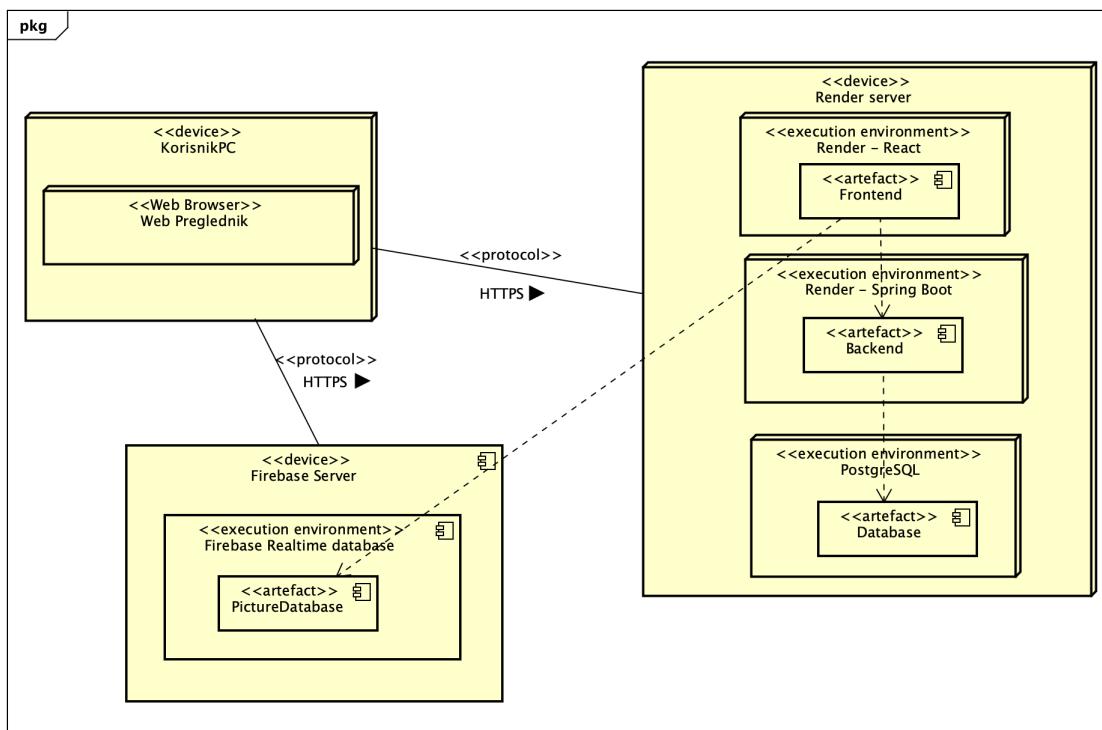
Slika 5.11: Test promjene cijene pretplate

Selenium IDE - Eventio*		
Project:	Executing	Run current test Ctrl+R
<b>NovaRecenzija*</b>		
Command	Target	Value
1 ✓ open	http://localhost:3000/moji	
2 ✓ set window size	1936x1048	
3 ✓ click	css=MuiContainer-maxWidthLg	
4 ✓ click	css=MuiBadge-root:nth-child(3) .MuiTypography-root:nth-child(3)	
5 ✓ click	css=MuiBadge-root:nth-child(3) .css-dgr9h-MuiRating-label:nth-child(7)	
6 ✓ click	id=r13:	Recenzija
7 ✓ type	css=MuiBadge-root:nth-child(3) .MuiCardActions-spacing	
8 ✓ click	css=MuiBadge-root:nth-child(3) .MuiCardActions-root:nth-child(5) > .MuiButtonBase-root	
9 ✓ click	css=MuiBadge-root:nth-child(3) .MuiCardActions-root > .MuiButtonBase-root > .MuiSvgIcon-root	
10 ✓ click	css=MuiContainer-maxWidthLg	
11 ✓ click		

Slika 5.12: Test stvaranja nove recenzije

### 5.3 Dijagram razmještaja

Dijagrami razmještaja opisuju topologiju sklopolja i programsku potporu koja se koristi u implementaciji sustava u njegovom radnom okruženju. Korisnici pristupaju web aplikaciji putem web preglednika na svojim osobnim računalima. Arhitektura sustava temelji se na jasno odvojenim slojevima: frontendu, backendu i bazi podataka koji su na zasebnim posužiteljima na Render-u - pružatelj poslužiteljskih usluga. Frontend poslužitelj posvećen je izvršavanju i posluživanju resursa vezanih uz korisničko sučelje. Također je povezan sa Firebase poslužiteljem koji smo koristili za spremanje slika. Firebase poslužitelj također je u komunikaciji sa web preglednikom korisnika kako bi poslužio potrebnu sliku. Backend poslužitelj fokusiran je na izvršavanje poslovne logike, pružanje API-ja te komunikaciju s bazom podataka. Na ovom sloju obavljaju se operacije i logika sustava, pružajući funkcionalnosti koje podržavaju rad aplikacije. Baza podataka poslužitelj je posvećen upravljanju bazom podataka. Ovdje se nalazi sama baza podataka i sustav za pohranu i dohvata podataka. Komunikacija između korisnika i poslužitelja odvija se preko HTTPS veze.



Slika 5.13: Dijagram razmještaja

## 5.4 Upute za puštanje u pogon

U ovom poglavlju opisat ćemo postupak puštanja web aplikacije u pogon - na koji način se od izvornog koda dolazi do potpuno postavljene baze podataka i poslužitelja koji odgovara na upite korisnika. U ovom postupku posebno ćemo opisat postavljanje baze podataka, backend poslužitelja i frontend poslužitelja. Za puštanje aplikacije u pogon preporuča se korištenje pružatelja servera te smo se odlučili za Render.

### Kreiranje baze podataka

Za postavljanje baze podataka u Render-u potrebno je pratiti sljedeće korake:

1. U padajućem izborniku "New" izabrati opciju "PostgreSQL"
2. Postaviti ime baze podataka te korisničko ime (username) za korisnika baze, lozinka je automatski generirana
3. U padajućem izborniku "Region" izabrati opciju "Frankfurt"
4. Odabratи opciju "Create database", u slučaju odabira besplatnog plana valja napomenuti da ima maksimalnu pohranu podataka do 1GB te da se baza podataka briše nakon 90 dana

Connections

---

Hostname	①	<input type="text"/>
Port	5432	
Database	dbeventio	
Username	dbeventio_user	
Password	<input type="password"/> ⚡	.....
Internal Database URL	<input type="text"/> ⚡	.....
External Database URL	<input type="text"/> ⚡	.....
PSQL Command	<input type="text"/> ⚡	.....

Slika 5.14: Postavke PostgreSQL baze podataka nakon kreiranja

## Kreiranje backenda

Na slici 5.13. također su prikazane varijable koje će se koristiti za spajanje na backend poslužitelj što je opisano u sljedećim koracima.

1. U padajućem izborniku "New" izabrati opciju "Web Service"
2. Povezati GitHub račun te u odabrat odgovarajući projekt za povezivanje
3. Postaviti ime koje će također biti postati i dio web adrese
4. Opciju "Root directory" ostaviti praznom te postaviti "Environment - Docker"
5. U padajućem izborniku "Region" izabrati opciju "Frankfurt"
6. Na dnu stranice proširiti opciju "Advanced"
7. Dodati potrebne environment varijable koje se trebaju kopirati iz postavki baze podataka na Render, slika 5.13.

The screenshot shows the Render interface for a project named "eventio-be". The "Environment" tab is active. It displays three environment variables:

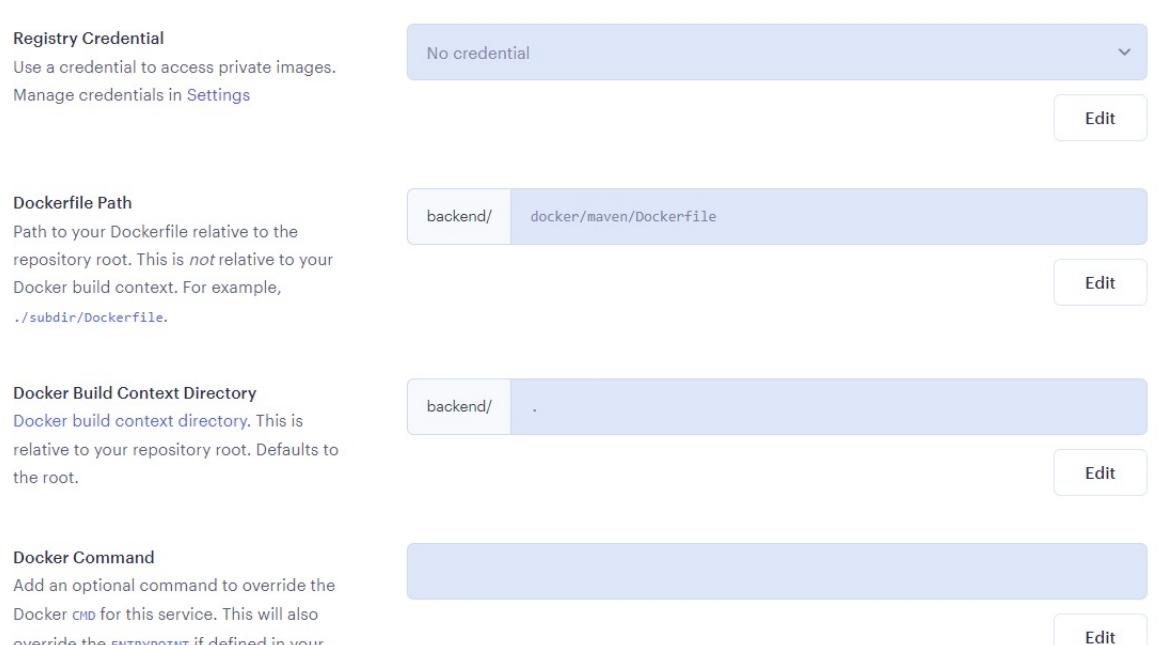
- DB\_PASSWORD**: Value: \*\*\*\*\*
- DB\_URL**: Value: postgresql://@dpq-clq8omhjvg7s73e4h6t0-a.frankfurt-...
- DB\_USERNAME**: Value: eventio\_db\_user

At the bottom right, there are buttons for "+ Add Environment Variable" and "Save Changes".

Slika 5.15: Environment varijable za backend

8. Postaviti putanju na Dockerfile (ovisno o packet manageru), u našem slučaju to je "./docker/maven/Dockerfile"

Ovdje prilažemo slike koje prikazuju konfiguracijske postavke na Renderu, Dockerfile na spomenutoj putanji te dio datoteke



Slika 5.16: Konfiguracijske postavke backend poslužitelja

```
# Dockerfile
1 # Container za izgradnju (build) aplikacije
2 > FROM openjdk:17-alpine AS builder
3
4 # Kopiranje izvornog koda u container
5 COPY ../../.mvn .mvn
6 COPY ../../mvnw .
7 COPY ../../pom.xml .
8 COPY ../../src src
9 RUN chmod +x mvnw
10
11 # Pokretanje builda i skipanje testova
12 RUN ./mvnw clean package -DskipTests
13
14 # Stvaranje containera u kojem ce se vrtiti aplikacija
15 FROM openjdk:17-alpine
16
17 ## Ovdje je moguce instalirati alete potrebne za rad aplikacije. Vjerojatno vam nece trebati, no dobro je znati.
18 ## Linux distro koji se koristi je Alpine, stoga se kao package manager koristi apk
19 #RUN apk install <nesto>
20
21 # Kopiranje izvrsnog JAR-a iz build containera u izvrsni container
22 COPY --from=builder target/*.jar /app.jar
23
24 # Izlaganje porta
25 EXPOSE 8080
26
27 # Naredba kojom se pokreće aplikacija
28 ENTRYPOINT ["java","-jar","/app.jar"]
```

Slika 5.17: Dockerfile na putanju /docker/maven/Dockerfile

## 9. Odabratи opciju "Create Web Service"

Ovdje napominjemo da u slučaju korištenja besplate verzije, nakon neaktivnosti aplikacije ona će biti automatski ugašena te ponovno podignuta zaprimanjem prvog zahtjeva. Drugim riječima, ovo može rezultirati sa nekoliko minuta nedostupnosti aplikacije nakon perioda neaktivnosti.

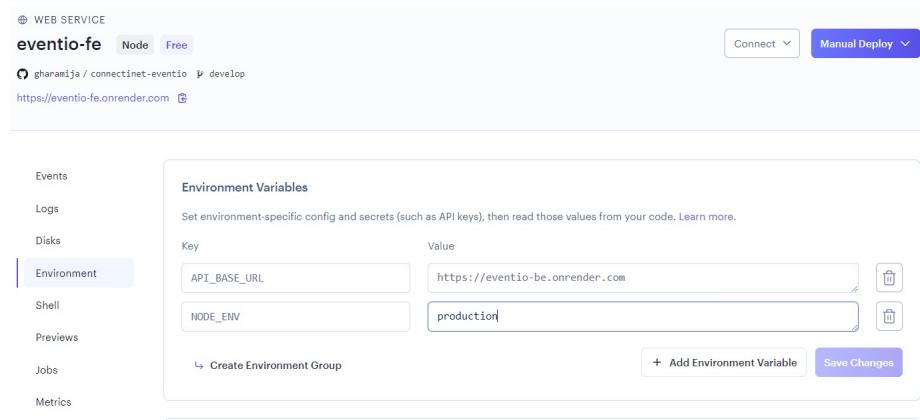
## Kreiranje frontenda

1. U padajućem izborniku "New" izabrati opciju "Web Service"
2. Povezati GitHub račun te u odabrat odgovarajući projekt za povezivanje
3. Postaviti ime koje će također biti postati i dio web adrese
4. Opciju "Root directory" ostaviti praznom te postaviti "Environment - Node"
5. U padajućem izborniku "Region" izabrati opciju "Frankfurt"
6. Opciju "Build command" postaviti na "npm run build"
7. Opciju "Start command" postaviti na "npm run start-prod"



Slika 5.18: Konfiguracijske postavke frontend poslužitelja

8. Na dnu stranice proširiti opciju "Advanced"
9. Dodati potrebne environment varijable, "API-BASE-URL" postaviti na adresu podignutog backenda što je prikazano Render Dashboardu



Slika 5.19: Environment varijable za frontend

#### 10. Odabri opciju "Create Web Service"

## 6. Zaključak i budući rad

Cilj našeg projekta bio je razviti programsku podršku za web aplikaciju "Eventio", inovativnu platformu za promociju zabavnih događanja u gradu, koja bi korisnicima omogućila jednostavno stvaranje i sudjelovanje na događanjima. Ova inovativna web aplikacija je koncipirana kako bi korisnicima pružila lakši pristup društvenim događanjima koja odgovaraju njihovim interesima, istovremeno olakšavajući organizaciju i promociju tih događanja.

Formiran je tim od sedam članova koji su u razdoblju od 17 tjedana surađivali na ovom projektu. Za svakodnevnu komunikaciju koristili smo WhatsApp, Discord za održavanje remote sastanaka te česte timske sastanke uživo kako bismo zajedno raspravljali o tijeku projekta, izazovima, i mogućim unapređenjima. Razvili smo internu raspodjelu zadataka koji su bili podijeljeni članovima tima što je osiguralo konstantan rad na projektu. Tome je pomogla i interna struktura tima podijeljena u podtimove s većim fokusom na drukčijim dijelovima projekta, uključujući backend, frontend i dokumentaciju. Iskusniji članovi su pružali mentorstvo i podršku onima s manje iskustva, olakšavajući im savladavanje novih tehnologija i koncepta.

Prva faza projekta, počevši s okupljanjem tima, bila je ključna za razvoj međusobnih odnosa, što je rezultiralo učinkovitijom suradnjom u kasnijim fazama. Osim toga, faza je obuhvatila rad na dokumentaciji, uključujući izradu obrazaca uporabe, sekvencijske dijagrame, model baze podataka, dijagrame razreda. Izrada navedenih dijagrama, ali i vizualnih rješenja programskog zadatka bile su od ključne važnosti za usmjeravanje razvoja te su uvelike olakšale implementaciju rješenja.

Druga faza projekta bila je usmjerenja na konkretan razvoj web aplikacije. Osim samog programiranja, ova faza uključivala je detaljno dokumentiranje projekta, uključujući UML dijagrame (dijagram stanja, aktivnosti, komponenti i razmještaja) te izradu popratne dokumentacije koja je obuhvatila opis testiranja, deployment procese i druge važne aspekte implementacije.

Gledajući prema budućnosti, projekt pruža niz mogućnosti za daljnje unapređenje. Jedna od mogućnosti je razvoj mobilne aplikacije kako bi se omogućilo korisnicima pristup događanjima putem mobilnih uređaja. Također, smatramo da bi

uvodenje sustava za chat na događanjima dodatno obogatilo korisničko iskustvo omogućavajući sudionicima međusobnu komunikaciju. Osim toga, daljnje širenje na područja izvan Zagreba, primjerice na razini Republike Hrvatske, predstavlja logičan korak u razvoju projekta.

Iskustvo rada na ovom projektu bilo je izuzetno korisno za sve članove tima. Poboljšali smo ne samo tehničke vještine već i sposobnosti komunikacije, planiranja i timskog rada. Sudjelovanje u projektu pružilo nam je uvid u dinamiku rada u timu te nas pripremilo za buduće izazove u našim karijerama. Zadovoljni smo rezultatom ovog projekta i što smo bili njegov dio te se veselimo nastavku učenja i rasta u budućnosti.

# Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinzh>
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>
7. Stack Overflow, <http://https://stackoverflow.com/>
8. React dokumentacija <https://legacy.reactjs.org/>
9. PostgreSQL dokumentacija <https://www.postgresql.org/>
10. W3Schools <https://www.w3schools.com/>
11. GitHub <https://github.com/>

# Indeks slika i dijagrama

3.1 Dijagram obrasca uporabe, opća funkcionalnost aktora . . . . .	22
3.2 Dijagram obrasca uporabe, funkcionalnost administratora . . . . .	23
3.3 Dijagram obrasca uporabe, funkcionalnost korisnika . . . . .	24
3.4 Dijagram obrasca uporabe, funkcionalnost organizatora . . . . .	25
3.5 Dijagram obrasca uporabe, funkcionalnost posjetitelja . . . . .	26
3.6 Sekvencijski dijagram za UC1 . . . . .	27
3.7 Sekvencijski dijagram za UC16 . . . . .	28
3.8 Sekvencijski dijagram za UC19 . . . . .	29
4.1 Arhitektura sustava . . . . .	31
4.2 Dijagram baze podataka . . . . .	37
4.3 Dijagram razreda - Kontroleri . . . . .	41
4.4 Dijagram razreda - Servisi . . . . .	42
4.5 Dijagram razreda - Data Transfer Objects . . . . .	43
4.6 Dijagram razreda - Models . . . . .	44
4.7 Dijagram stanja - Korisnik (Posjetitelj) . . . . .	45
4.8 Dijagram aktivnosti - Postavljanje događaja (Organizator) . . . . .	47
4.9 Dijagram komponenti . . . . .	48
5.1 Test za sortiranje i ispisivanje događaja . . . . .	52
5.2 Test registracije korisnika . . . . .	53
5.3 Test slanja e-maila . . . . .	53
5.4 Test spremanja recenzije . . . . .	54
5.5 Test spremanja nove zainteresiranosti . . . . .	54
5.6 Test spremanja postojeće zainteresiranosti . . . . .	55
5.7 Izvršavanje i rezultati testova . . . . .	55
5.8 Test uspješne prijave organizatora . . . . .	56
5.9 Test neuspješne prijave organizatora . . . . .	56
5.10 Test stvaranja novog događaja . . . . .	57
5.11 Test promjene cijene pretplate . . . . .	57
5.12 Test stvaranja nove recenzije . . . . .	58

5.13 Dijagram razmještaja . . . . .	59
5.14 Postavke PostgreSQL baze podataka nakon kreiranja . . . . .	60
5.15 Environment varijable za backend . . . . .	61
5.16 Konfiguracijske postavke backend poslužitelja . . . . .	62
5.17 Dockerfile na putanju /docker/maven/Dockerfile . . . . .	62
5.18 Konfiguracijske postavke frontend poslužitelja . . . . .	63
5.19 Environment varijable za frontend . . . . .	64
6.1 Dijagram pregleda promjena . . . . .	74

# Dodatak: Prikaz aktivnosti grupe

## Dnevnik sastajanja

### 1. sastanak

- Datum: 19. listopada 2023.
- Prisustvovali: F. Buhiniček, M. Grdić, M. Sršić, N. Kušen, J. Šarolić, S. Boka, G. Haramija
- Teme sastanka:
  - upoznavanje sudionika
  - dogovaranje prvih koraka

### 2. sastanak

- Datum: 26. listopada 2023.
- Prisustvovali: F. Buhiniček, M. Grdić, M. Sršić, N. Kušen, J. Šarolić, S. Boka, G. Haramija
- Teme sastanka:
  - podijelili dijelove dokumentacije članovima za rad (2,3.1,3.1.1 )
  - složen osnovni Spring Boot server

### 3. sastanak

- Datum: 2. studenoga 2023.
- Prisustvovali: F. Buhiniček, M. Grdić, M. Sršić, N. Kušen, J. Šarolić, S. Boka, G. Haramija
- Teme sastanka:
  - podijelili dijelove dokumentacije (4, 4.1, 4.1.1, 4.1.2, 4.2)
  - započeta izrada registracije i login-a (backend, frontend)

### 4. sastanak

- Datum: 10. studenoga 2023.
- Prisustvovali: F. Buhiniček, M. Grdić, M. Sršić, N. Kušen, J. Šarolić, S. Boka, G. Haramija
- Teme sastanka:
  - pregled napravljenog, dogovor za ispravke, poboljšanja, deploy

## 5. sastanak

- Datum: 13. prosinca 2023.
- Prisustvovali: F. Buhiniček, M. Grdić, M. Sršić, N. Kušen, J. Šarolić, S. Boka, G. Haramija
- Teme sastanka:
  - pregled napravljenog
  - izrada skica screen-ova
  - podjela frontend komponenti

## 6. sastanak

- Datum: 22. prosinca 2023.
- Prisustvovali: F. Buhiniček, M. Grdić, M. Sršić, N. Kušen, J. Šarolić, S. Boka, G. Haramija
- Teme sastanka:
  - pregled napravljenog
  - dogovor o radu za vrijeme praznika

## 7. sastanak

- Datum: 11. siječnja 2024.
- Prisustvovali: F. Buhiniček, M. Grdić, M. Sršić, N. Kušen, J. Šarolić, S. Boka, G. Haramija
- Teme sastanka:
  - pregled napravljenog
  - podjela završnih zadataka

## 8. sastanak

- Datum: 18. siječnja 2024.
- Prisustvovali: F. Buhiniček, M. Grdić, M. Sršić, N. Kušen, J. Šarolić, S. Boka, G. Haramija
- Teme sastanka:
  - pregled i finalizacija aplikacije i dokumentacije
  - priprema za predaju

## Tablica aktivnosti

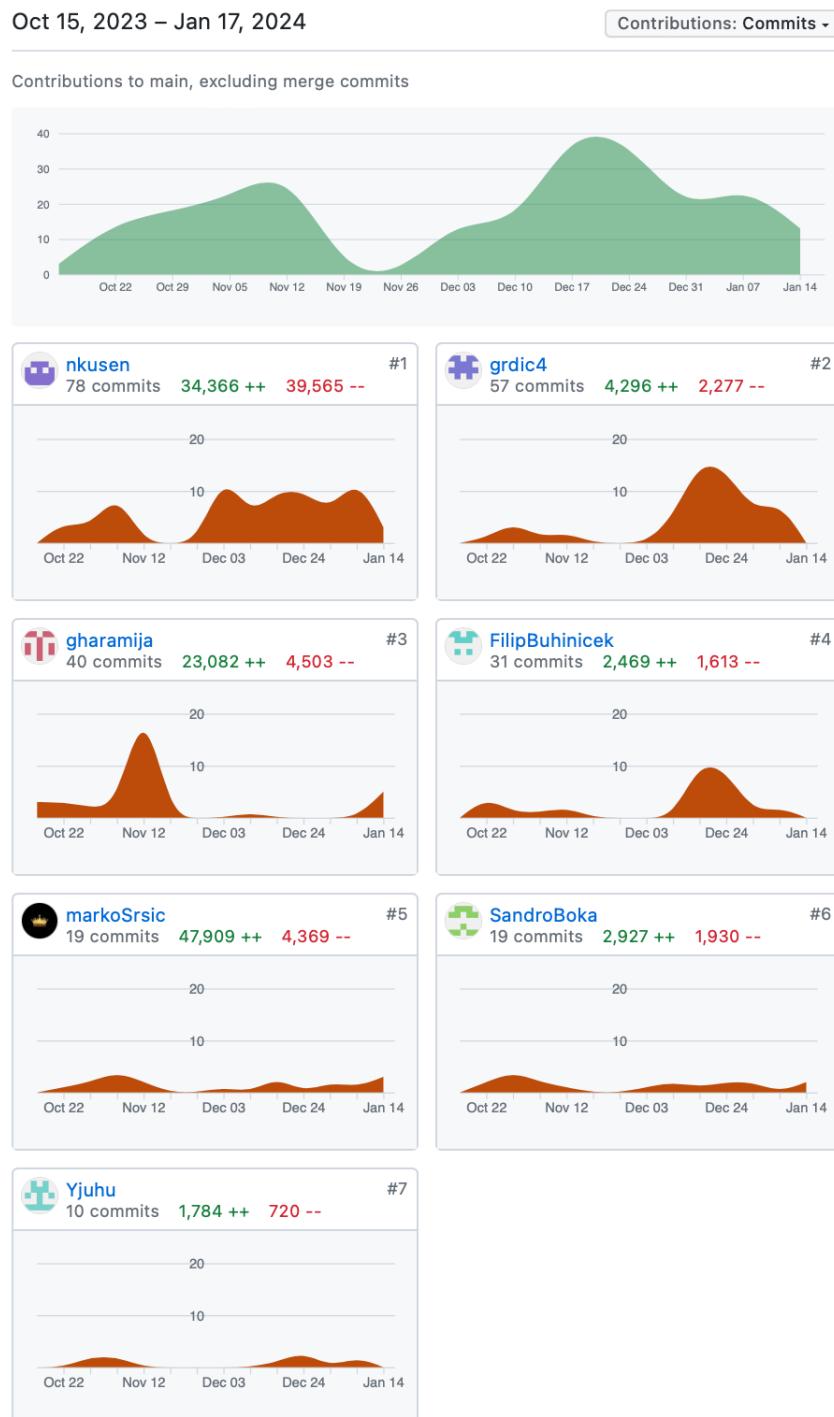
	Gašpar Haramija	Filip Buhiniček	Sandro Bokta	Marko Sršić	Mateo Grdić	Nikola Kušen	Jakov Šarolić
Upravljanje projektom	40						
Opis projektnog zadatka	8	7					
Funkcionalni zahtjevi				2.5	5		
Opis pojedinih obrazaca				18			
Dijagram obrazaca			7				
Sekvencijski dijagrami	5			1			
Opis ostalih zahtjeva	2						
Arhitektura i dizajn sustava	3			2			
Baza podataka		9		1.5	10		
Dijagram razreda	2		6	8			
Dijagram stanja				7			
Dijagram aktivnosti				7			
Dijagram komponenti	7						
Korištene tehnologije i alati	7	5	3	7	5	5	5
Ispitivanje programskog rješenja	7	7			2.5	2.5	2.5
Dijagram razmještaja	7						
Upute za puštanje u pogon	6.5	1.5	1.5	1.5	1.5	1.5	1.5
Dnevnik sastajanja	3	3		4			
Zaključak i budući rad	2						
Popis literature				1.5			

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Gašpar Haramija	Filip Buhiniček	Sandro Boka	Marko Sršić	Mateo Grdić	Nikola Kušen	Jakov Šarolić
Frontend			25		35	40	30
Backend	10	37			48	40	
Deploy	15	8			8	15	
Opće ažuriranje dokumentacije	10	1.5		17			
GitHub	10	8			5		
Readme, upute	2			1			

## Dijagrami pregleda promjena



Slika 6.1: Dijagram pregleda promjena