

Programsko inženjerstvo

Ak. god. 2023./2024.

ConnectiNET

Dokumentacija, Rev. 1

Grupa: *Spajalice*

Voditelj: *Mario Mrvčić*

Datum predaje: *17. studenog 2023.*

Nastavnik: *Nikolina Frid*

Sadržaj

1 Dnevnik promjena dokumentacije	3
2 Opis projektnog zadatka	6
3 Specifikacija programske potpore	10
3.1 Funkcionalni zahtjevi	10
3.1.1 Obrasci uporabe	12
3.1.2 Sekvencijski dijagrami	25
3.2 Ostali zahtjevi	32
4 Arhitektura i dizajn sustava	33
4.1 Baza podataka	36
4.1.1 Opis tablica	37
4.1.2 Dijagram baze podataka	42
4.2 Dijagram razreda	44
4.3 Dijagram stanja	47
4.4 Dijagram aktivnosti	49
4.5 Dijagram komponenti	52
5 Implementacija i korisničko sučelje	54
5.1 Korištene tehnologije i alati	54
5.2 Ispitivanje programskog rješenja	56
5.2.1 Ispitivanje komponenti	56
5.2.2 Ispitivanje sustava	63
5.3 Dijagram razmještaja	73
5.4 Upute za puštanje u pogon	74
6 Zaključak i budući rad	77
Popis literature	79
Indeks slika i dijagrama	81

Dodatak: Prikaz aktivnosti grupe

82

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak	Josip Du- vančić	30.10.2023.
0.2	Dodani funkcionalni zahtjevi	Josip Du- vančić	02.11.2023.
0.3	Dodan opis projektnog zadatka	Istok Kor- kut	07.11.2013.
0.4	Dodana većina obrazaca uporabe	Josip Du- vančić	07.11.2023.
0.4.1	Dodani svi obrasci uporabe	Josip Du- vančić	08.11.2023.
0.5	Napravljeni dijagrami obrazaca uporabe	Josip Du- vančić	09.11.2023.
0.6	Napisani opisi sekvencijskih dijagrama	Istok Kor- kut	09.11.2023.
0.6.1	Prepravljanje obrazaca uporabe i izrada sekvencijskih dijagrama	Josip Du- vančić	10.11.2023.
0.7	Početak izrade baze podataka	Josip Du- vančić, Mario Mrvčić	13.11.2023.
0.7.1	Napravljene tablice za bazu podataka	Josip Du- vančić	15.1.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
0.7.2	Dovršeni opisi tablica za bazu podataka	Josip Duvančić	15.1.2023.
0.7.3	Napravljen dijagram baze podataka	Josip Duvančić	15.1.2023.
0.8	Napisan opis arhitekture sustava	Domagoj Capar, Duje Jurić	15.1.2023.
0.9	Napravljeni dijagrami razreda	Istok Korkut	16.1.2023.
0.10	Zadnje preinake u dokumentaciji	Josip Duvančić	17.1.2023.
1.0	Ispravak grešaka iz prve revizije	Josip Duvančić	8.12.2023.
1.1	Napravljen dijagram stanja	Josip Duvančić	20.12.2023.
1.2	Napravljen dijagram aktivnosti	Josip Duvančić	21.12.2023.
1.3	Napisan opis korištenih tehnologija i alata	Istok Korkut	6.1.2024.
1.4	Napravljen dijagram komponenti	Josip Duvančić	9.1.2024.
1.5	Napravljen dijagram razmještaja	Josip Duvančić	10.1.2024.
1.6.1	Napravljeni testovi za ispitivanje programskog rješenja	Domagoj Capar, Toma Žulj	15.1.2024.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.6.2	Opisani testovi za ispitivanje programskog rješenja	Domagoj Capar, Toma Žulj, Josip Duvančić	15.1.2024.
1.7	Napisane upute za puštanje u pogon	Istok Kor- kut	17.1.2024.
1.8	Napisan zaključak i nadopunjavanje aktivnosti	Josip Du- vančić	19.1.2024.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti programsku podršku za izradu web aplikacije koja će korisnicima omogućiti objaviti ili pronaći događaj u svojoj zajednici. U aplikaciji će organizatori događanja postavljati najave za događanja koja organiziraju, a zainteresirani posjetitelji će moći najaviti svoj dolazak te pisati recenzije.

Kako bi korisnik pristupio funkcionalnostima aplikacije, prvo će se trebati registrirati kao posjetitelj ili kao organizator. Organizatori će svoja događanja koja promoviraju putem aplikacije moći naplaćivati ili ne naplaćivati. Ako organizatori naplaćuju svoja događanja, plaćat će korištenje aplikacije na mjesečnoj bazi. To će raditi putem PayPala ili pomoću kreditnih kartica. Za razliku od njih, posjetitelji i organizatori besplatnih događanja koristit će aplikaciju besplatno.

Posjetitelji će nakon pokretanja aplikacije moći vidjeti popis događanja u određenom vremenskom razdoblju koje sami odaberu (24h, 7 dana ili 30 dana). Svako događanje imat će definirano mjesto, vrijeme, trajanje i cijenu ulaznice (ukoliko se radi o događanju koje se plaća). Za svako prikazano događanje, korisnik će moći izjasniti svoj interes: “dolazim”, “ne dolazim” ili “možda dolazim”. Bit će ostvarena i funkcionalnost koja posjetiteljima omogućuje da se predomisle i promijene status svog interesa. Korisnik će također moći napisati recenziju za svako prikazano događanje koje se održalo u posljednjih 48 sati. Također, posjetitelji će moći postaviti kriterije po vrsti događanja i lokaciji te primiti notifikacije kad bude objavljen događaj koji ih zadovoljava.

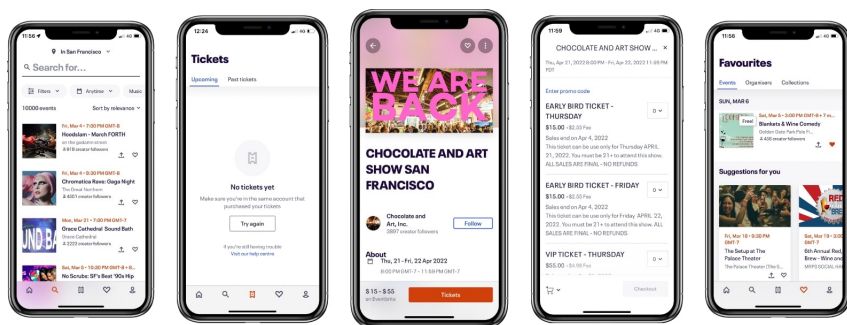
Na javnim profilima organizatora nalazit će se osnovne informacije o njima, poput naziva, adrese i dodatnih linkova. Na njihovim će profilima biti prikazana i sva događanja koja su organizirali u zadnje dvije godine. Za svako događanje dostupne su dodatne informacije poput naziva, vrste, lokacije, vremena početka, trajanja i galerija fotografija i videa. Takve informacije organizator postavlja sam. Za svaki protekli događaj bit će dostupne i korisničke recenzije, a za svaki tek najavljeni bit će moguće vidjeti koliko ljudi planira doći.

U sustavu će postojati i administratori čija je funkcija postavljanje cijene članstva.

Ovakva aplikacija bila bi korisna za oglašavanje događaja koji su ograničeni svojom temom, grupom uzvanika ili lokacijom. Za događaje širih parametara, platforma poput Facebooka ili Instagrama je idealna, ali bi ovakva aplikacija odabirom teme na koju želi suziti svoje događaje mogla postati korisnija korisnicima zainteresiranim za tu temu. Primijenjena na FER-ovce mogla bi služiti oglašavanju natjecanja iz programiranja u Hrvatskoj, a primijenjena na likovne umjetnike mogla bi služiti oglašavanju izložbi. S druge strane, mogla bi biti korisna i odabirom grupe uzvanika umjesto odabirom teme događaja. Npr., mogla bi služiti tome da zaposlenici neke veće firme lakše saznaju za aktivnosti koje su im dostupne kao zaposlenicima te firme. Također bi bila korisna i sužavanjem lokacije samih evenata. Npr., kompleks poput velesajma mogao bi dopuštati organizatorima da oglašavaju svoje događaje koji će se tamo odvijati.

Za slične se svrhe dosad koristio Facebook, ali zbog zakrčenosti platforme i zbog toga što joj organizacija i oglašavanje događanja nije primarna svrha, potrebne su aplikacije koje se bave baš isključivo time. Nekoliko popularnih aplikacija takve vrste su Eventbrite i Bizzabo. Osim samog oglašavanja, obje aplikacije nude i mogućnost

prodaje karata online, a organizator prijavljen u aplikaciju ima i mogućnost skeniranja karata i vođenja evidencije gostiju. Ovakve aplikacije imaju neke dodatne mogućnosti u odnosu na cilj ovog projekta jer se fokusiraju na pružanje usluge samim organizatorima, dok se u ovom projektu stavlja naglasak na samo oglašavanje i promociju događanja.



Slika 2.1: Sučelje aplikacije Eventbrite

Za ovakvu aplikaciju mogle bi biti zainteresirane organizacije koje se bave promocijom događaja unutar neke domene. Npr. organizacije koje se bave organiziranjem tehno-partyja, javnih čitanja poezije ili bilo čega drugog. Time bi privukle publiku koje baš takve određene teme zanimaju i osigurale da ih takva publika nađe lakše nego na već zakrčenom Facebooku. Velike firme ili fakulteti poput FER-a mogli bi ovu aplikaciju koristiti za oglašavanje događanja za svoje zaposlenike ili studente. Mjesta poput Velesajma ili Medike mogla bi koristiti ovakvu aplikaciju kao platformu koju bi organizatori događanja na takvim mjestima koristili za oglašavanje.

U budućnosti bi se ovakva aplikacija mogla nadograditi na više načina.

Moglo bi se omogućiti da se ne ocjenjuje samo događanja nego i njihove organizatore, tako da potencijalni posjetitelji mogu postaviti koliko će kvalitetno događanje biti ne samo prema opisu i riječima organizatora, nego i po riječima onih koji su već imali iskus-

tva s istim organizatorom.

Također bi se mogla dodati funkcionalnost koja omogućuje oglašavanje poslova na oglašenim događanjima. Npr. ako na nekom događanju nedostaje konobar, organizator tog događanja može tu informaciju dodati oglasu događanja.

Osim filtera koji postavljaju kriterije na mjesto i vrstu događaja, mogli bi se dodati filteri koji se odnose na organizatore događaja, njegov datum, dan u tjednu ili vrijeme u danu. Jedan od parametara filtera (i samih događaja) mogao bi biti i jezik na kojem će se događaj održati.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Posjetitelj
2. Organizator
3. Administrator
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
 - (a) pregled događanja
 - (b) odabir događanja i pregled općih informacija o događanju
 - (c) mogućnost registriranja kao posjetitelj, organizator ili administrator
2. Posjetitelj (inicijator) može:
 - (a) pregled i mogućnost izmjene osobnih podataka
 - (b) brisanje vlastitog računa
 - (c) mogućnost ostavljanja recenzije na događaj
 - (d) mogućnost ostavljanja oznake zainteresiranosti za neki događaj
 - (e) mogućnost pretplaćivanja na obavijesti o budućim događanjima
3. Organizator (inicijator) može:
 - (a) dodavanje događanja
 - (b) uređivanje informacija o svom događanju

- (c) brisanje događanja
- (d) obaveza plaćanja članarine ako se događanja naplaćuju

4. Administrator (inicijator) može:

- (a) pregled svih registriranih korisnika i njihovih osobnih podataka
- (b) brisanje i mijenjanje uloga korisnika
- (c) pristup statistici događanja i korisnika
- (d) brisanje recenzija koje krše pravila korištenja
- (e) postavljanje cijene članarina organizatora

5. Baza podataka (sudionik) može:

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje sve podatke o događanjima

6. Banka (sudionik) može:

- (a) pruža uslugu plaćanja članarine

7. Paypal (sudionik) može:

- (a) pruža uslugu plaćanja članarine

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 -Pregled događanja

- **Glavni sudionik:** Korisnik, posjetitelj
- **Cilj:** Pregled ponude događanja
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. prikaz početnog zaslona sa popisom događanja
 2. korisnik bira događanje
 3. prikaz informacija i galerije događanja
 4. mogućnost registracije i prijave

UC2 -Registracija

- **Glavni sudionik:** Korisnik
- **Cilj:** Stvaranje korisničkog računa
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. odabir opcije za registraciju
 2. unos osobnih podataka, postavljanje lozinke i odabir uloge
 3. dobivanje povratne informacije o uspješnoj registraciji
- **Opis mogućih odstupanja:**
 - 2.a Unos već postojećeg emaila, unos podatka u nezadovoljavajućem formatu
 1. obavijest o pogrešnom unosu
 2. ponovni unos potrebnih podataka

UC3 -Prijava u sustav

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Prijava u vlastiti korisnički račun
- **Sudionici:** Baza podataka
- **Preduvjet:** posjetitelj je registriran
- **Opis osnovnog tijeka:**
 1. odabir opcije za prijavu
 2. unos emaila i lozinke
 3. pristup funkcionalnostima korisničkog računa
- **Opis mogućih odstupanja:**
 - 2.a neispravan email ili lozinka
 1. obavijest o pogrešnom unosu emaila ili lozinke
 2. ponovni unos potrebnih podataka

UC4 -Pregled osobnih podataka

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Uvid u vlastite osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** posjetitelj je prijavljen
- **Opis osnovnog tijeka:**
 1. odabir opcije pregleda osobnih podataka
 2. uvid u osobne podatke

UC5 -Promjena osobnih podataka

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Izmjena vlastitih osobnih podataka
- **Sudionici:** Baza podataka
- **Preduvjet:** posjetitelj je prijavljen, otvoren je pregled podataka
- **Opis osnovnog tijeka:**
 1. odabir opcije za izmjenu podataka
 2. izmjena podataka

- 3. spremanje podataka
- 4. ažuriranje baze
- **Opis mogućih odstupanja:**
 - 3.a opcije za spremanje podataka nije odabrana
 - 1. obavijest o opasnosti da se podatci neće spremiti
 - 3.b podaci nisu upisani u dobrom formatu
 - 1. obavijest o krivo napisanom formatu

UC6 -Brisanje korisničkog računa

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Brisanje vlastitog korisničkog računa
- **Sudionici:** Baza podataka
- **Preduvjet:** posjetitelj je prijavljen, otvoren je pregled podataka
- **Opis osnovnog tijeka:**
 - 1. odabir opcije za uređivanje računa
 - 2. odabir opcije za brisanje računa
 - 3. korisnički račun se briše iz baze
 - 4. otvara se početna stranica

UC7 -Reakcija na događanje

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Naznaka o dolaznosti posjetitelja
- **Sudionici:** Baza podataka
- **Preduvjet:** posjetitelj je prijavljen, otvoren je pregled događanja
- **Opis osnovnog tijeka:**
 - 1. odabir opcije za reakciju na određenom događanju
 - 2. mogućnost odabira jedne od opcija: "Dolazim", "Ne dolazim" i "Možda dolazim"

UC8 -Dodavanje recenzije

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Dodavanje recenzije na događaj
- **Sudionici:** Baza podataka
- **Preduvjet:**
 - posjetitelj je prijavljen, otvoren je pregled događanja
 - opcija je odabrana u manje od 48 sati od završetka događanja
 - posjetitelj je prethodno označio događaj s: "Dolazim" ili "Možda dolazim"
- **Opis osnovnog tijeka:**
 1. odabir opcije za dodavanje recenzije
 2. pisanje recenzije
 3. spremanje recenzije na bazu podataka

UC9 -Pregled recenzija

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Uvid u recenzije prijašnjih događanja
- **Sudionici:** Baza podataka
- **Preduvjet:** otvoren je pregled događanja
- **Opis osnovnog tijeka:**
 1. odabir opcije za prikaz recenzija određenog događaja
 2. prikaz svih recenzija poredanih od najnovije

UC10 -Filtriranje ponude

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Primjena filtera pri pregledu početne stranice
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. odabir prikaza početne stranice
 2. odabir opcije za primjenu filtera

3. odabir filtera koji se žele primijeniti
4. odabir opcije primjeni
5. prikaz filtriranog prelgeda

UC11 -Prijava na obavijesti

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Prijava posjetitelja na primanje obavijesti o budućim događanjima
- **Sudionici:** Baza podataka
- **Preduvjet:** posjetitelj je prijavljen
- **Opis osnovnog tijeka:**
 1. odabir opcije za prijavljivanje na obavijesti
 2. odabir parametara prema kojima posjetitelj želi primati obavijesti

UC12 -Uređivanje događanja korisnika

- **Glavni sudionik:** Posjetitelj
- **Cilj:** Uređivanje odabira reakcije ili recenzije na događanjima posjetitelja
- **Sudionici:** Baza podataka
- **Preduvjet:**
 - posjetitelj je prijavljen
 - posjetitelj je prethodno označio događaj s: "Dolazim" ili "Možda dolazim"
- **Opis osnovnog tijeka:**
 1. odabir opcije za uređivanje vlastitih događanja
 2. odabir događanja na kojem se žele izvršiti promjene
 3. izmjena zainteresiranosti ili recenzije za određeno događanje

UC13 -Obavijest o nadolazećem događanju

- **Glavni sudionik:** Baza podataka

- **Cilj:** Slanje obavijesti posjetitelju
- **Sudionici:** Posjetitelj
- **Preduvjet:** Posjetitelj mora zadovoljavati uvjete koji su zadani događanjem
- **Opis osnovnog tijeka:**
 1. podatci o novonastalom događanju zadovoljavaju uvjete koje su određeni posjetitelji označili kao poželjne
 2. aplikacija šalje informacije o događanju svim zainteresiranim posjetiteljima
 3. posjetiteljima na uređaj dolazi obavijest o događanju

UC14 -Stvaranje događanja

- **Glavni sudionik:** Organizator
- **Cilj:** Stvaranje novog događanja
- **Sudionici:** Baza podataka
- **Preduvjet:** organizator je prijavljen
- **Opis osnovnog tijeka:**
 1. odabir opcije za kreiranje novog događanja
 2. unos podataka, fotografija i cijene ulaznica za buduće događanje
 3. spremanje podataka
- **Opis mogućih odstupanja:**
 - 2.a ulaznice se plaćaju
 1. ako organizator nije uplatio članarinu otvara mu se stranica za uplatu
 2. organizator potvrđuje da želi naplaćivati ulaznice i sprema svoj odabir

UC15 -Uređivanje događanja

- **Glavni sudionik:** Organizator
- **Cilj:** Izmjena podataka o vlastitom događanju

- **Sudionici:** Baza podataka
- **Preduvjet:** Organizator je prijavljen i događanje još nije završilo
- **Opis osnovnog tijeka:**
 1. odabir događanja na kojem se žele napraviti izmjene
 2. izmjena podataka
 3. spremanje izmjena na bazu podataka
- **Opis mogućih odstupanja:**
 - 2.a događanje je s besplatnog dobilo cijenu ulaznice
 1. provjerava se jeli članarina uplaćena
 2. ako članarina nije uplaćena korisnik se odводи na stranicu za uplatu članarine

UC16 -Odgovaranje na recenzije

- **Glavni sudionik:** Organizator
- **Cilj:** Odgovaranje na recenzije koje su posjetitelji ostavili na njegovu događanje
- **Sudionici:** Baza podataka
- **Preduvjet:** organizator je prijavljen
- **Opis osnovnog tijeka:**
 1. odabir događanja
 2. odabir popisa recenzija na vlastitom događanju
 3. odabir recenzije na koju želi dati odgovor
 4. pisanje odgovora na recenziju
 5. spremanje u bazu podataka

UC17 -Plaćanje članarine

- **Glavni sudionik:** Organizator
- **Cilj:** Plaćanje članarine organizatora
- **Sudionici:** Baza podataka, Paypal, Banka

- **Preduvjet:** organizator ima barem jedan događaj za koji se na-
plaćuju karte
- **Opis osnovnog tijeka:**
 1. odabir opcije za uplatu članarine
 2. odabir načina plaćanja
 3. popunjavanje podatak vezanih uz način plaćanja
 4. traženje potvrde od pružatelja usluge plaćanja
 5. potvrda o uspješnoj uplati članarine
- **Opis mogućih odstupanja:**
 - 2.a neuspješna uplata
 1. obavijest o neuspješnoj transakciji
 2. opcije za ponovni pokušaj ili za odustajanje od uplate

UC18 -Brisanje događanja

- **Glavni sudionik:** Organizator
- **Cilj:** Brisanje vlastitog događanja
- **Sudionici:** Baza podataka
- **Preduvjet:** Organizator je prijavljen
- **Opis osnovnog tijeka:**
 1. odabir događanja na kojem se žele napraviti izmjene
 2. odabire opciju za brisanje događanja
 3. povratak na glavnu stranicu

UC19 -Pregled korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Uvid u podatke o korisnicima
- **Sudionici:** Baza podataka
- **Preduvjet:** administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. odabir opcije za pregled korisnika

2. prikaz popisa korisnika i njihovih podataka

UC20 -Brisanje korisnika

- **Glavni sudionik:** Administrator,
- **Cilj:** Brisanje određenog korisničkog računa
- **Sudionici:** Baza podataka
- **Preduvjet:** administrator je prijavljen, otvoren je pregled svih korisnika
- **Opis osnovnog tijeka:**
 1. odabir korisnika koji se treba obrisati
 2. odabir opcije za brisanje korisnika
 3. potvrda

UC21 -Postavljanje cijene članarine

- **Glavni sudionik:** Administrator
- **Cilj:** Postavljanje cijene članarine koju plaćaju organizatori
- **Sudionici:** Baza podataka
- **Preduvjet:** administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. odabir opcije za izmjenu cijene članarine
 2. izmjena cijene članarine

UC22 -Uređivanje određenog događanja

- **Glavni sudionik:** Administrator
- **Cilj:** Uređivanje određenog događanja
- **Sudionici:** Baza podataka
- **Preduvjet:** administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. iz prikaza svih događanja odabire se željeni događaj
 2. odabire se opcija uređivanja događanja

3. spremaju se promjene
4. povratak na prikaz svih događanja
- **Opis mogućih odstupanja:**
 - 3.a administrator želi promijeniti cijenu ulaznice s 0 na broj veći od 0
 1. promjena nije moguća

UC23 -Brisanje određenog događanja

- **Glavni sudionik:** Administrator
- **Cilj:** Brisanje određenog događanja
- **Sudionici:** Baza podataka
- **Preduvjet:** administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. iz prikaza svih događanja odabire se željeni događaj
 2. odabire se opcija brisanja događanja
 3. povratak na prikaz svih događanja

UC24 -Brisanje recenzija

- **Glavni sudionik:** Administrator
- **Cilj:** Brisanje recenzija koje ne poštuju pravila korištenja
- **Sudionici:** Baza podataka
- **Preduvjet:** administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. odabir opcije za prikaz recenzija određenog događanja
 2. odabir recenzije
 3. odabir opcije za brisanje recenzije

UC25 -Promjena uloge

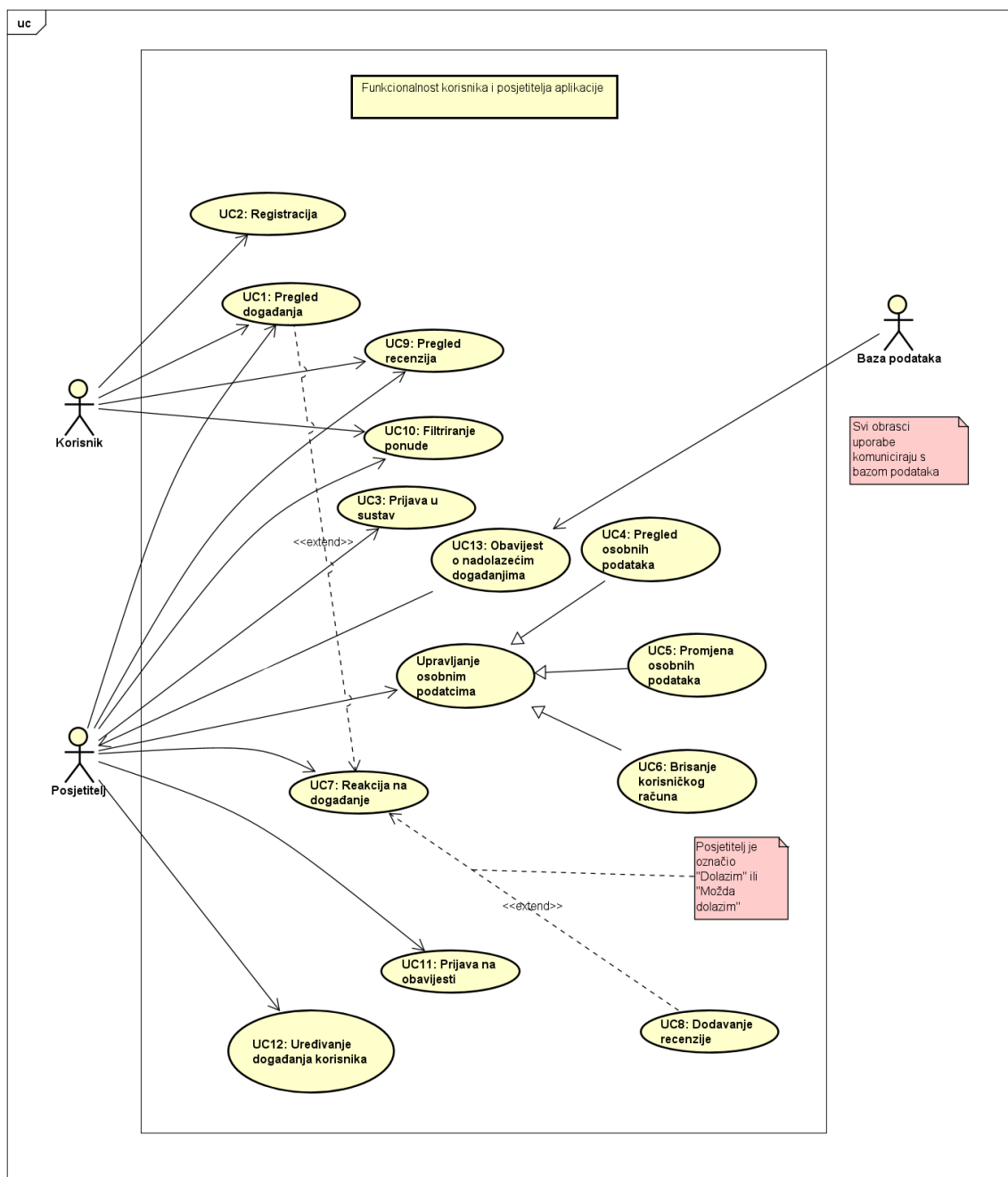
- **Glavni sudionik:** Administrator
- **Cilj:** Izmjena uloge određenog korisnika

- **Sudionici:** Baza podataka
- **Preduvjet:** administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. odabir popisa svih korisnika
 2. odabir korisnika kojemu se želi pormijeniti uloga
 3. odabir opcije za izmjenu uloge
 4. odabir uloge koja mu se dodjeljuje

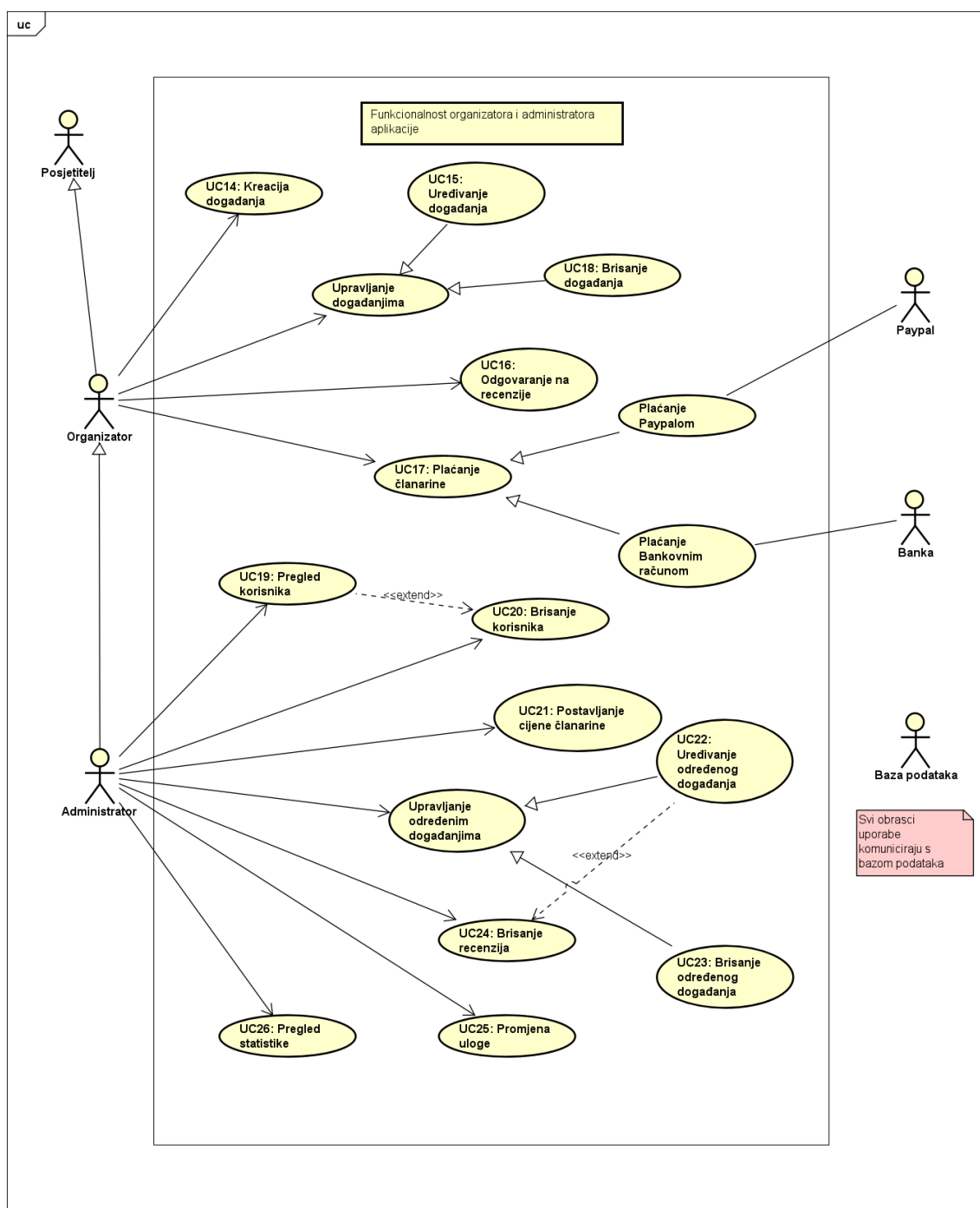
UC26 -Pregled statistike

- **Glavni sudionik:** Administrator
- **Cilj:** Uvid u statistiku svih aktera u aplikaciji
- **Sudionici:** Baza podataka
- **Preduvjet:** administrator je prijavljen
- **Opis osnovnog tijeka:**
 1. odabir opcije za prikaz statistike
 2. prikaz statistike

Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, funkcionalnost korisnika i posjetitelja

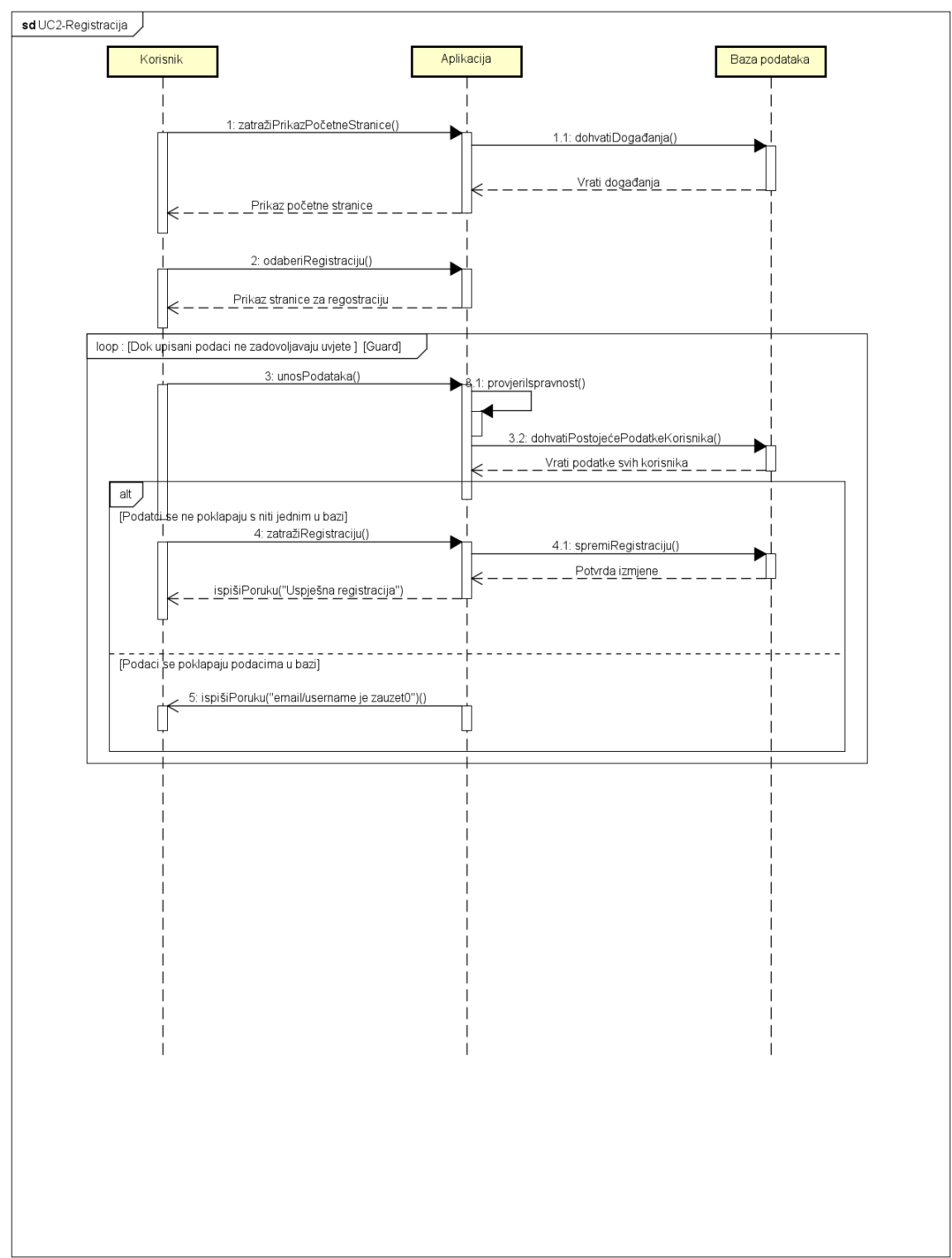


Slika 3.2: Dijagram obrasca uporabe, funkcionalnost organizatora i administratora

3.1.2 Sekvencijski dijagrami

Obrazac uporabe UC2 - Registracija

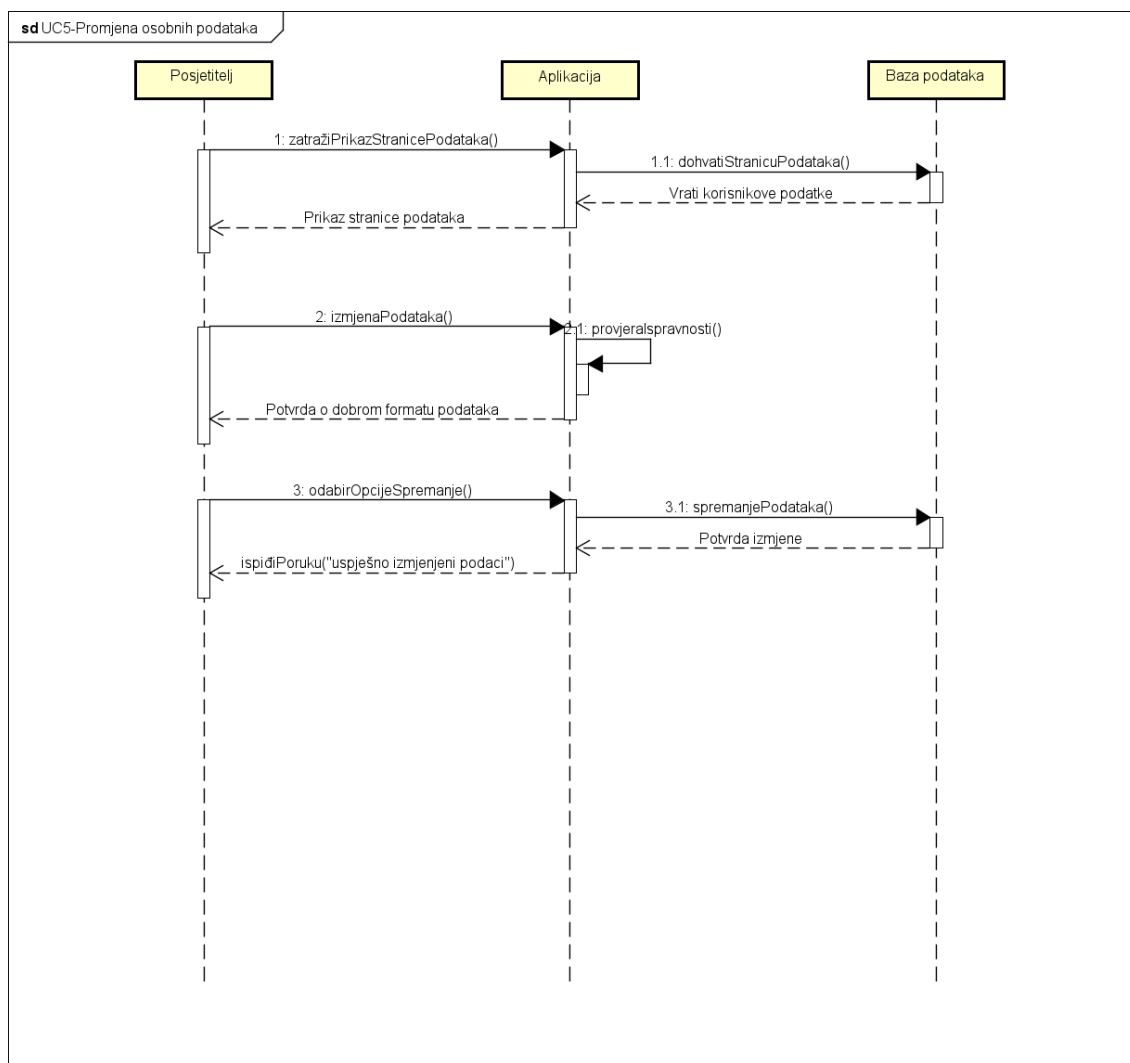
Klijent odabirom opcije za registraciju šalje zahtjev za prikaz odgovarajuće stranice. Poslužitelj dohvaća tu stranicu i prikazuje ju. Klijent unosi osobne podatke, postavlja lozinku i bira ulogu. Aplikacija provjerava jesu li svi podatci u traženom format i postoji li već takav klijent u sustavu. Ako podatci nisu u zadanom formatu, aplikacija šalje obavijest o tome i klijent iznova unosi potrebne podatke. Ako takav klijent već postoji u sustavu, aplikacija klijentu šalje obavijest o tome. Ako su podatci uneseni u traženom format i takav klijent ne postoji u sustavu, aplikacija u bazu podataka prosljeđuje dobivene podatke, a klijentu šalje obavijest o uspješnoj registraciji.



Slika 3.3: Sekvencijski dijagram za UC2

Obrazac uporabe UC5 - Promjena osobnih podataka

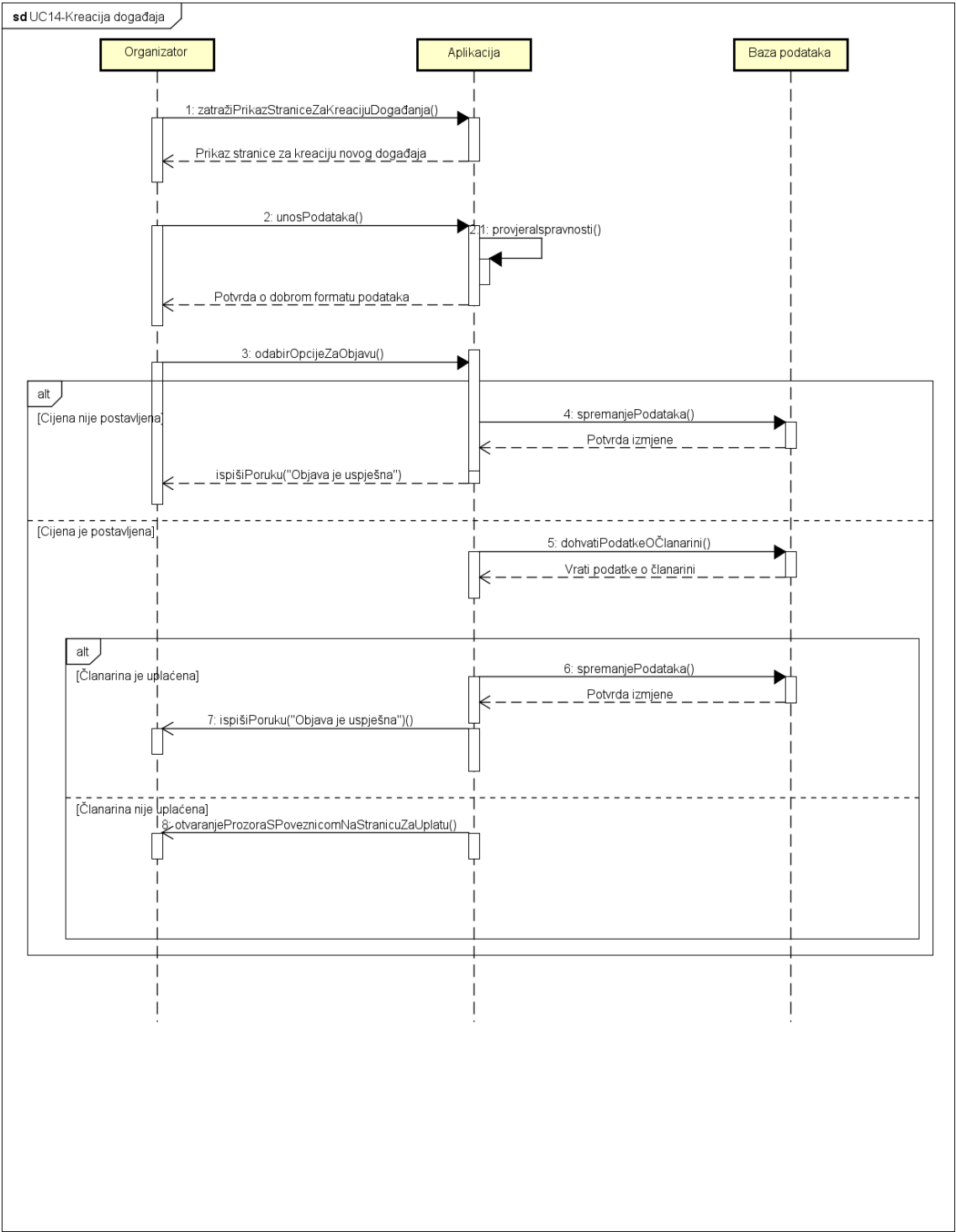
Korisnik šalje zahtjev za prikaz stranice za izmjenu podataka. Poslužitelj dohvaća i prikazuje traženu stranicu. Korisnik izmjenjuje svoje podatke i odabire opciju za spremanje podataka. Ukoliko korisnik ne odabere tu opciju, poslužitelj korisniku notifikacijom skreće pozornost na to. Ukoliko podatci ne zadovoljavaju traženi format, aplikacija notifikacijom skreće korisniku pozornost na to. Ukoliko su podatci u traženom formatu, poslužitelj ih ažurira u bazi podataka.



Slika 3.4: Sekvencijski dijagram za UC5

Obrazac uporabe UC14 - Kreacija događanja

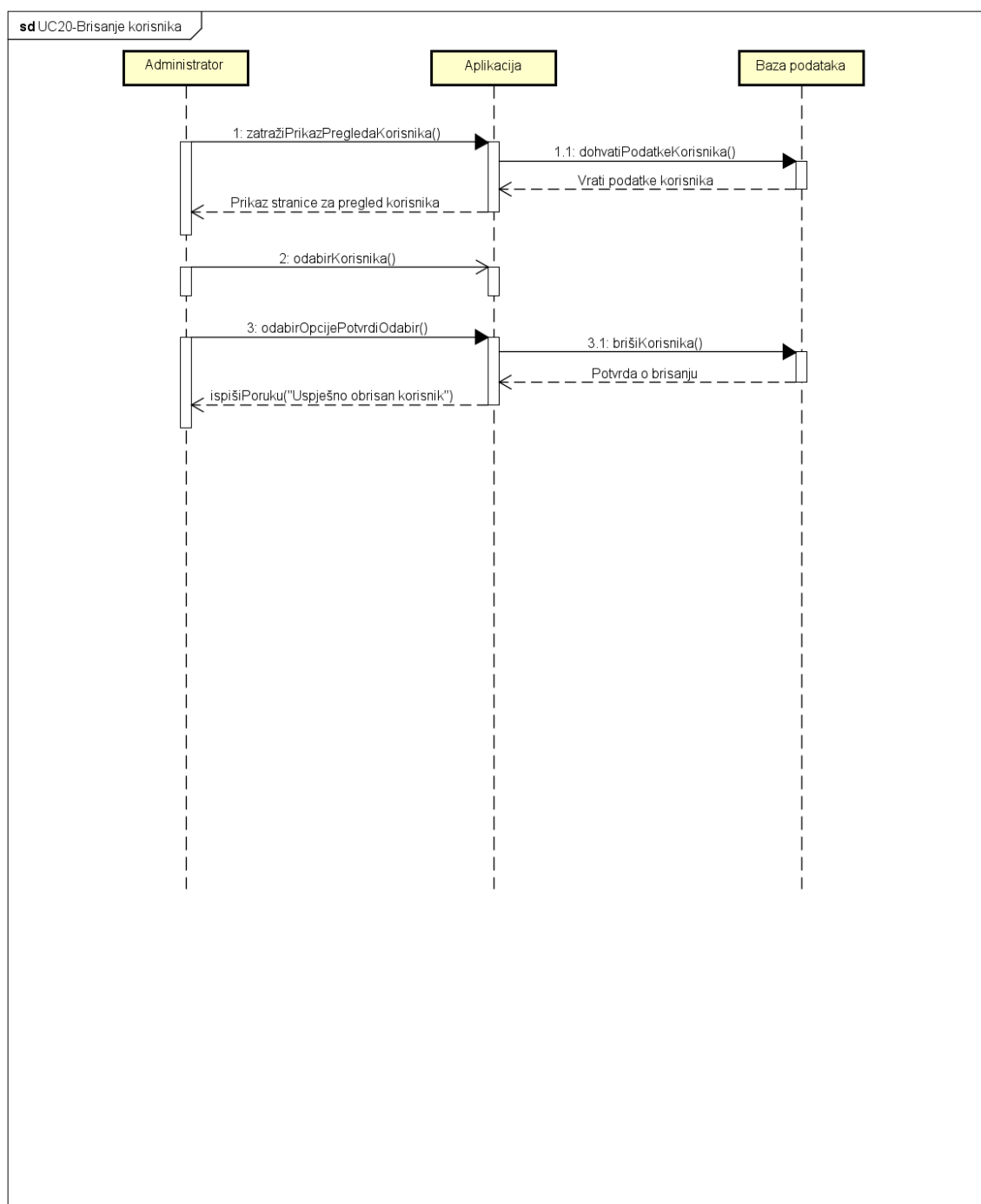
Klijent poslužitelju šalje zahtjev za prikaz stranice za kreiranje novog događanja. Poslužitelj dohvaća traženu stranicu i prikazuje ju. Klijent unosi podatke i fotografije, a po izboru i cijenu ulaznice. Ako je klijent postavio cijenu događanja, poslužitelj dohvaća podatke o klijentu iz baze podataka i provjerava jeli uplatio članarinu. Ako nije, poslužitelj ga o tome obavještava notifikacijom i klijent prihvća ili odbija pretplatiti se. Ako se pretplati, poslužitelj unosi događanje u bazu podataka. Ako klijent ne postavi cijenu događanja, poslužitelj nista ne provjerava i sprema podatke o događanju u bazu podataka.



Slika 3.5: Sekvencijski dijagram za UC14

Obrazac uporabe UC20 - Brisanje korisnika

Administrator odabire opciju za prikaz stranice za pregled korisnika. Poslužitelj dohvaća stranicu i prikazuje ju administratoru. Administrator odabire korisnika kojeg treba izbrisati. Potvrđuje svoj odabir i time šalje zahtjev za brisanjem poslužitelju. Poslužitelj dohvaća bazu podataka i iz nje uklanja podatke o korisniku i samog korisnika. Administratoru šalje obavijest o uspješnom brisanju.



Slika 3.6: Sekvencijski dijagram za UC20

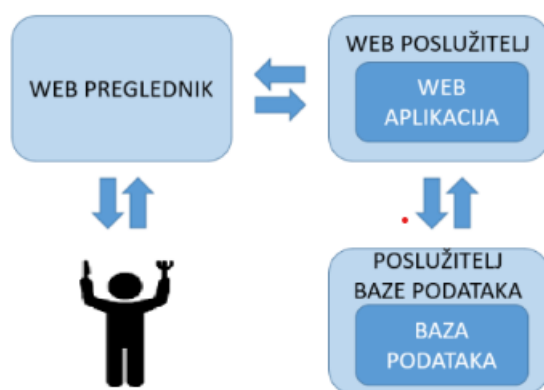
3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika od jednom u stvarnom vremenu
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu (dijakritičke znakove) pri unosu i prikazu tekstualnog sadržaja
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od nekoliko sekundi
- Sustav treba biti implementiran kao web aplikacija koristeći objektno-orijentirane jezike
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
- Sustav treba biti jednostavan za korištenje, korisnici se moraju znati koristiti sučeljem bez opširnih uputa
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava
- Sustav kao valutu koristi EUR
- Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške
- Pristup sustavu mora biti omogućen iz javne mreže pomoću HTTPS
- Sučelje mora biti estetski privlačno

4. Arhitektura i dizajn sustava

Arhitekturu sustava našeg projekta možemo podijeliti na tri glavna podsustava:

1. Baza podataka
2. Web aplikacija
3. Web poslužitelj



Slika 4.1: Skica arhitekture sustava

Internetski preglednik predstavlja programsku platformu koja korisnicima omogućuje pregledavanje web-stranica i konzumiranje raznovrsnih multimedijalnih sadržaja povezanih s njima. Svaki od tih preglednika funkcionira kao svojevrсни tumač, čime interpretira kompleksni kod web stranica i pretvara ga u vizualno prihvatljiv format za korisnike. Bitno je razumjeti da je ključna uloga internetskog preglednika prevesti tehnički zahtjevne elemente weba u nešto pristupačno svakom korisniku.

Kada korisnik putem internetskog preglednika šalje zahtjev, ta komunikacija se uspostavlja s web poslužiteljem. Web poslužitelj pred-

stavlja temelj rada web aplikacija, a njegova osnovna zadaća je posredovati u komunikaciji između korisnika (klijenta) i same aplikacije. Cijeli taj proces komunikacije odvija se putem HTTP (HyperText Transfer Protocol) protokola, koji služi kao standardni način prijenosa informacija na internetu.

Važno je naglasiti da je web poslužitelj taj koji pokreće web aplikaciju i prosljeđuje joj korisnički zahtjev. Kroz korištenje web aplikacije, korisnik šalje različite zahtjeve koje aplikacija obrađuje. Ovisno o konkretnom zahtjevu, web aplikacija može pristupiti bazi podataka kako bi došla do relevantnih informacija. Nakon obrade zahtjeva, aplikacija putem web poslužitelja šalje odgovor korisniku u obliku HTML dokumenta. Taj dokument postaje vizualno doživljajan u internetskom pregledniku, predstavljajući krajnji rezultat korisničkog zahtjeva.

Programski jezik kojeg smo odabrali za izradu naše web aplikacije je Java zajedno s React i Spring Boot stackom. Projekt programiramo u razvojnom okruženju IntelliJ IDEA. Arhitekturu sustava baziramo na MVC konceptu (Model-View-Controller) prilagođenom za React i Spring boot. Glavna karakteristika zbog koje smo se odlučili za MVC je nezavisno razvijanje pojedinih dijelova aplikacije što nam omogućuje jednostavnije testiranje i implementiranje novog koda. Glavna podjela je na tzv. „Backend“ i „Frontend“.

Backend (Spring Boot) MVC koncept:

1. Model: Svi razredi (koje su povezane sa bazom) sa odgovarajućim atributima te servisi u kojima se provodi logika koji kasnije koriste API komponente (Controlleri)
2. View: JSON podatci s kojima upravljaju API komponente(Controlleri)

3. Controller: API komponente koje obrađuju zahtjeve te šalju adekvatne odgovore

Frontend (React) MVC arhitektura:

1. Model: Stanja i podatci koje se koriste i prikazuju u komponentama
2. View: Komponente koje reprezentiraju izgled aplikacije odnosno korisničko sučelje
3. Controller: „Event handleri“ i metode koje obrađuju logiku, unos korisnika te upravljaju podacima i stanjima u komponentama

4.1 Baza podataka

Za potrebe našeg sustava koristit ćemo bazu podataka koja svojom strukturom olakšava modeliranje stvarnog svijeta. Osnovna jedinica baze je dokument, definiran svojim imenom i skupom atributa. Dokument može nadovezivati i klasa koja opisuje attribute sadržane u dokumentu. Korištenjem MongoDB-a kao ne-relacijske baze podataka, informacije će biti pohranjene u obliku fleksibilnih dokumenata umjesto tradicionalnih tablica. Zadaća baze podataka je brza i jednostavna pohrana, izmjena i dohvat podataka za daljnju obradu. Baza podataka ove aplikacije sastoji se od sljedećih dokumenata te klasa koje su sadržane unutar njih:

- Korisnik
- Događanje
 - Mjesto
 - Foto (lista)
 - Video (lista)
 - Recenzije (lista)
- Korisnik-Događanje

Osim dokumenata i klasa u bazi ćemo također imati i vlastite tipove podataka koje će biti prethodno definirani. To su:

- Uloga
- Vrsta
- Interes

4.1.1 Opis tablica

Dokumenti

Korisnik Ovaj dokument sadrži informacije o pojedinačnom korisniku. Sadrži attribute: email, lozinku, ime, prezime, adresu, ulogu, gradove interesa, vrste interesa, web stranicu i facebook. Ovaj dokument je u vezi *One-to-Many* s dokumentom Korisnik-Događanje preko atributa email. Pošto je baza nerelacijska u dokument će se također pisati i liste od: gradova i vrsta koje su potrebne kako bi se izvele sve funkcionalnosti.

Korisnik		
Email	STRING	email korisnika
Lozinka	STRING	hash lozinke
Ime	STRING	ime korisnika
Prezime	STRING	prezime korisnika
Adresa	STRING	kućna adresa korisnika
Uloga	ULOGA	uloga korisnika
Gradovi interesa	LIST<STRING>	gradovi interesa korisnika

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Korisnik		
Vrste interesa	LIST<VRSTA>	vrste interesa korisnika
Web stranica	STRING	link web stranice organizatora
Facebook	STRING	link facebook profila organizatora

Događanje Ovaj dokument sadrži informacije o pojedinačnom događanju. Sadrži attribute: ID, ime, vrsta, mjesto, adresu, datum, vrijeme početka, trajanje, opis, cijenu, foto galeriju, video galeriju i recenzije. Ovaj je dokument u vezi *One-to-Many* s dokumentom Korisnik-Događanje preko attribute ID. Događanje se također nadovezuje na klase mjesto, foto, video i recenziju. Za mjesto bi se moglo reći da je u vezi *One-to-One* jer ima samo jedno mjesto na kojem se događaj može odvijati, a za foto, video i recenzije bi se moglo reći da je u vezi *One-to-Many* jer može imati po više takvih atributa. Zato su ti atributi zapisani u listi.

Događanje		
ID	LONG	jedinstveni identifikator događanja
Ime	STRING	ime događanja
Vrsta	VRSTA	vrsta događanja
Mjesto	MJESTO	grad događanja
Adresa	STRING	adresa događanja
Datum	DATE	datum održavanja

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Događanje		
Vrijeme početka	LOCALTIME	vrijeme početka održavanja
Trajanje	LOCALTIME	trajanje događanja
Opis	STRING	opis događanja
Cijena	DOUBLE	cijena ulaznice
Foto galerija	LIST<FOTO>	fotografije događanja
Video galerija	LIST<VIDEO>	videozapisi događanja
Recenzije	LIST<RECENZIJA>	recenzije događanja

Korisnik-Događanje Ovaj dokument sadrži informacije o odnosu korisnika i događanja. Glavna zadaća mu je da čuva informaciju o tome koji se korisnik izjasni da će doći na koje događanje. Sadrži attribute: email, ID i interes. Sa korisnikom i sa događanjem je povezan *Many-to-One* vezom preko emaila i IDa.

Korisnik-Događanje		
Email	STRING	email posjetitelja
ID	LONG	identifikator događanja
Interes	INTEREST	interes posjetitelja

Klase

Mjesto Ova klasa sadrži informacije o mjestima na kojima se mogu održavati događanja. Atributi mjesta su: ID, ime, poštanski broj, grad i županija. Mjesto je klasa koju koristi događanje u "relaciji" *One-to-One*.

Mjesto		
ID	LONG	jedinstveni identifikator mjesta
Ime	STRING	ime mjesta
Poštanski broj	INTEGER	datum ostavljanje recenzije
Grad	STRING	grad u kojem se nalazi mjesto
Županija	STRING	županija u kojoj se mjesto nalazi

Foto Ova klasa sadrži informacije o fotografijama koje se nalaze u galeriji događanja. Atributi foto-a su: url i datum. Foto je klasa koja je "povezana" s događanjem preko *Many-to-One* veze.

Foto		
URL	STRING	jedinstveni url fotografije
Datum	DATE	datum objave fotografije

Video Ova klasa sadrži informacije o videozapisima koji se nalaze u galeriji događanja. Atributi videa su: url i datum. Video je klasa koja je "povezana" s događanjem preko *Many-to-One* veze.

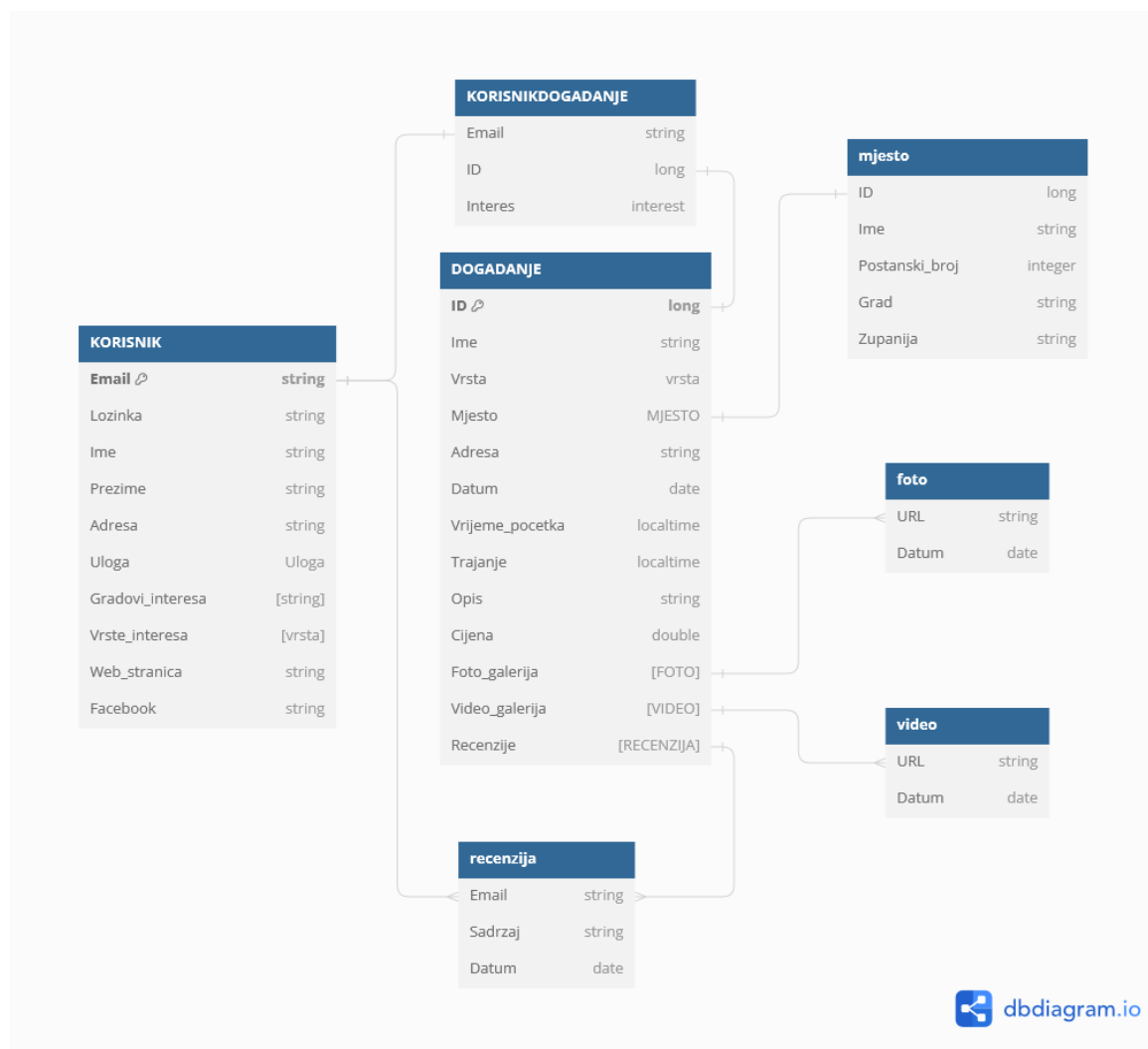
Video		
URL	STRING	jedinstveni url videozapisa
Datum	DATE	datum objave videozapisa

Recenzija Ova klasa sadrži informacije o recenzijama koje su korisnici ostavili na događanja. Ova klasa sadrži attribute: korisničko ime, sadržaj i datum. Recenzija je klasa koja spada pod događanje i s njime je "povezana" *Many-to-One* vezom, a s korisnikom je povezana kao u relacijskoj bazi preko emaila *Many-to-One* vezom.

Recenzija		
Email	STRING	email korisnika koji je napisao recenziju
Sadržaj	STRING	sadržaj recenzije
Datum	DATE	datum objave recenzije

4.1.2 Dijagram baze podataka

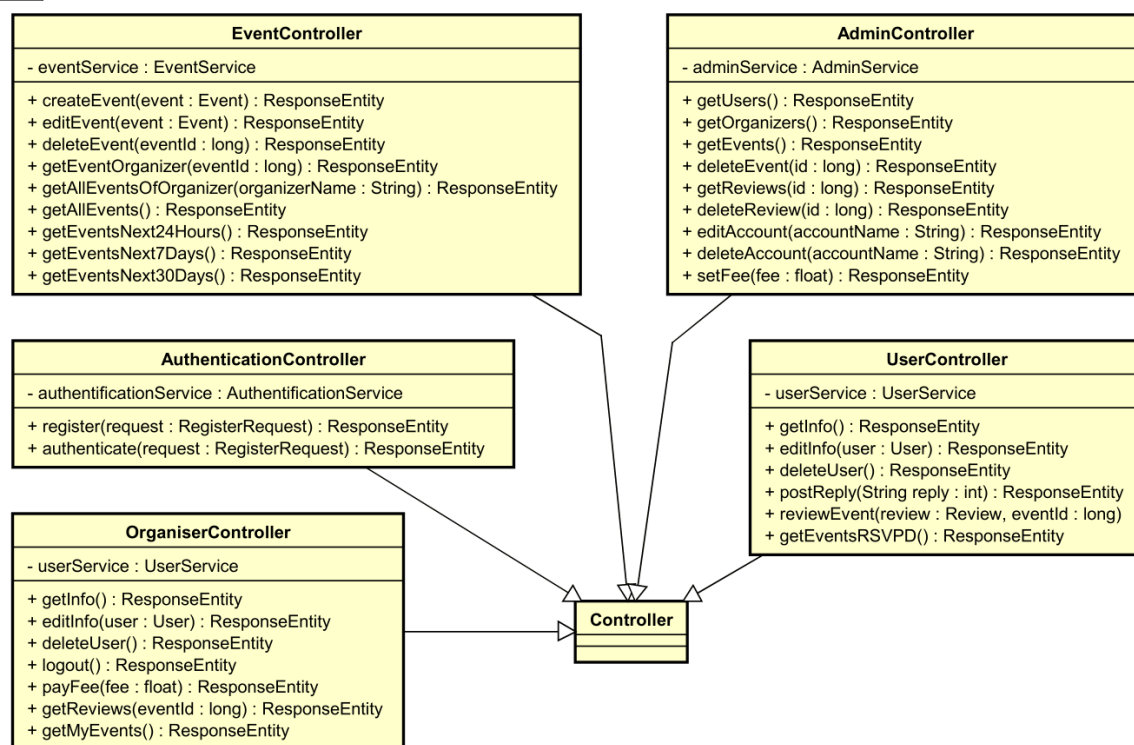
Pošto baza nije relacijska sam dijagram je teže skicirati. Stoga smo radi jednostavnosti odlučili dijagram naše baze skicirati kao da je relacijska. Na dijagramu su dokumenti korišteni u bazi označeni velikim tiskanim slovima, dok su klase koje reprezentiraju attribute u dokumentima označene malim tiskanim slovima i spojene s atributom u kojem se nalaze. U dokumentima su klase pak navedene velikim tiskanim slovima, a one koje se nalaze unutar uglatih zagrada smještene su u listu s više jednakih objekata. Recenzija je jedini rubni slučaj u kojem se korisničko ime zapravo uzima kao foreign key od korisnika, ali je također i u listi atributa u događanju. Program za izradu dijagrama nije imao mogućnost da razdvojimo takve slučajeve pa te veze izgledaju isto.



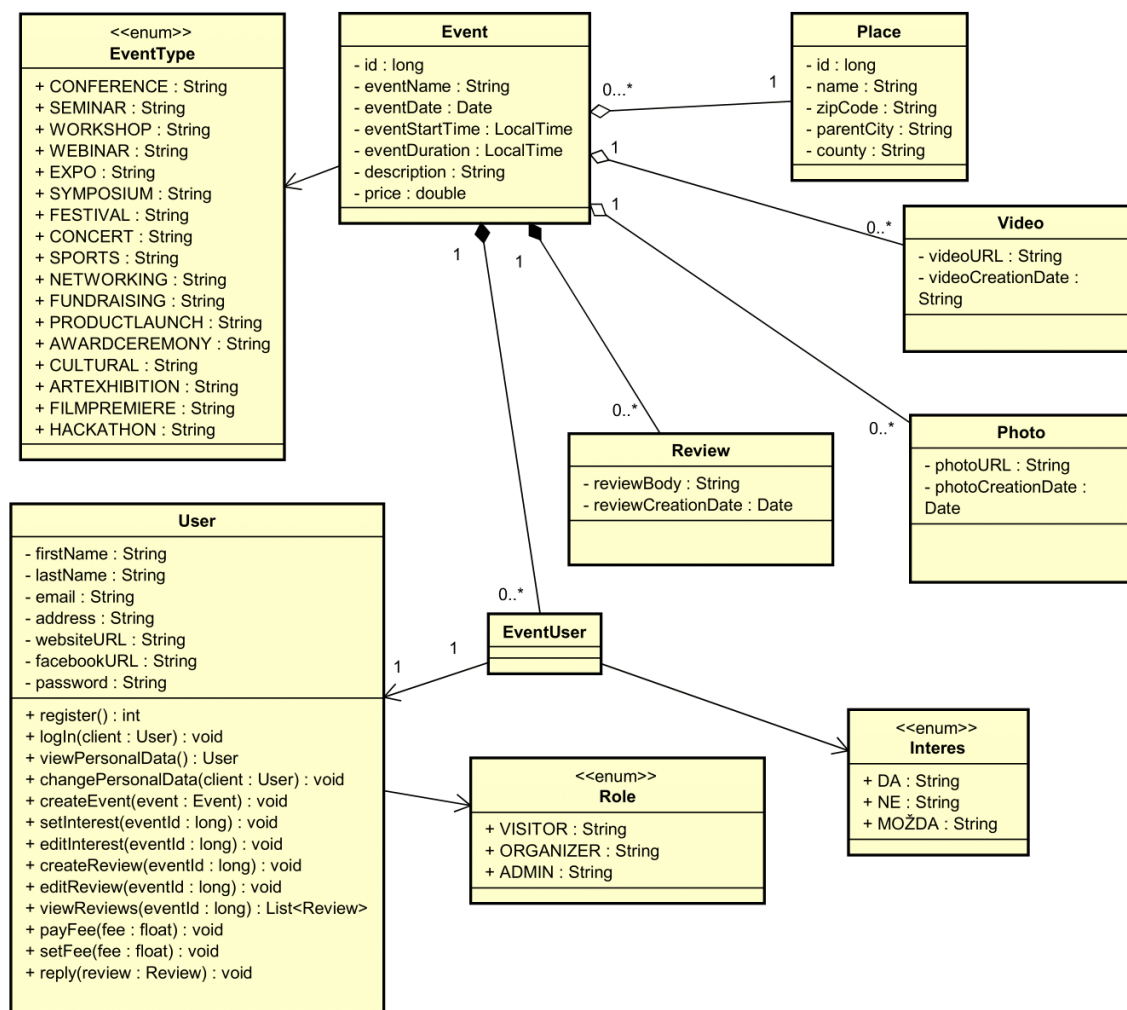
Slika 4.2: Dijagram baze podataka

4.2 Dijagram razreda

Na slikama 4.3 i 4.4 prikazani su razredi koji pripadaju backend dijelu arhitekture aplikacije. Razredi prikazani na slici 4.3 nasljeđuju razred Controller i primarno služe za dohvaćanje i manipulaciju podataka u sustavu. Na slici 4.4 prikazan je dijagram koji preslikava strukturu baze podataka. Metode u tim razredima direktno komuniciraju s bazom podataka. Razred User predstavlja korisnika čiju vrstu određuje enumeracija Role. Ovisno o vrijednosti te enumeracije, može se raditi o posjetitelju, administratoru ili organizatoru. Razred Event predstavlja objavljeno događanje koje je vrste određene vrijednosti enumeracije EventType. Razred Event kao attribute sadrži listu objekata razreda Photo i listu objekata razreda Video. Dva navedena razreda predstavljaju redom fotografije i videozapise događanja. Razred Event također sadrži listu objekata razreda EventUser. Razred EventUser predstavlja oznaku kojom je korisnik označio određeni događaj. Taj razred dakle povezuje korisnika s nekom vrijednosti iz enumeracije Interest. Razred Event također sadrži listu objekata razreda Review koji predstavlja recenzije koje su korisnici ostavili za određeno događanje.



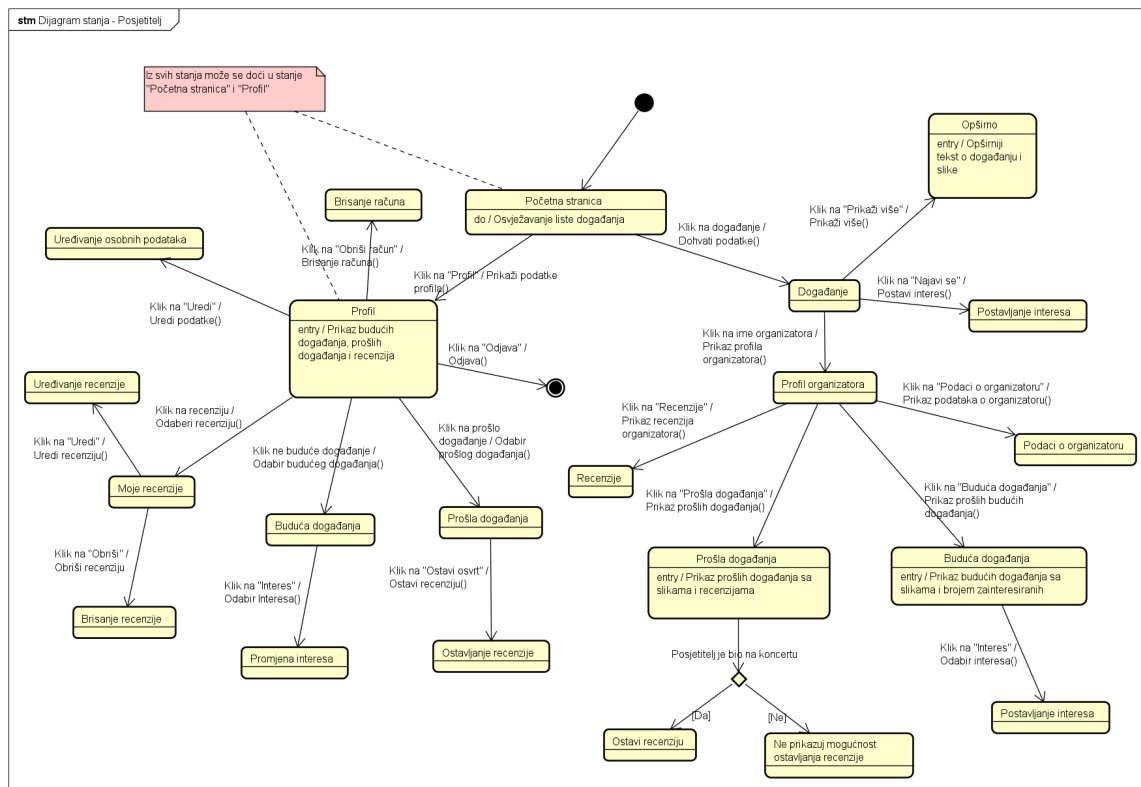
Slika 4.3: Dijagram razreda - Controllers



Slika 4.4: Dijagram razreda - Models

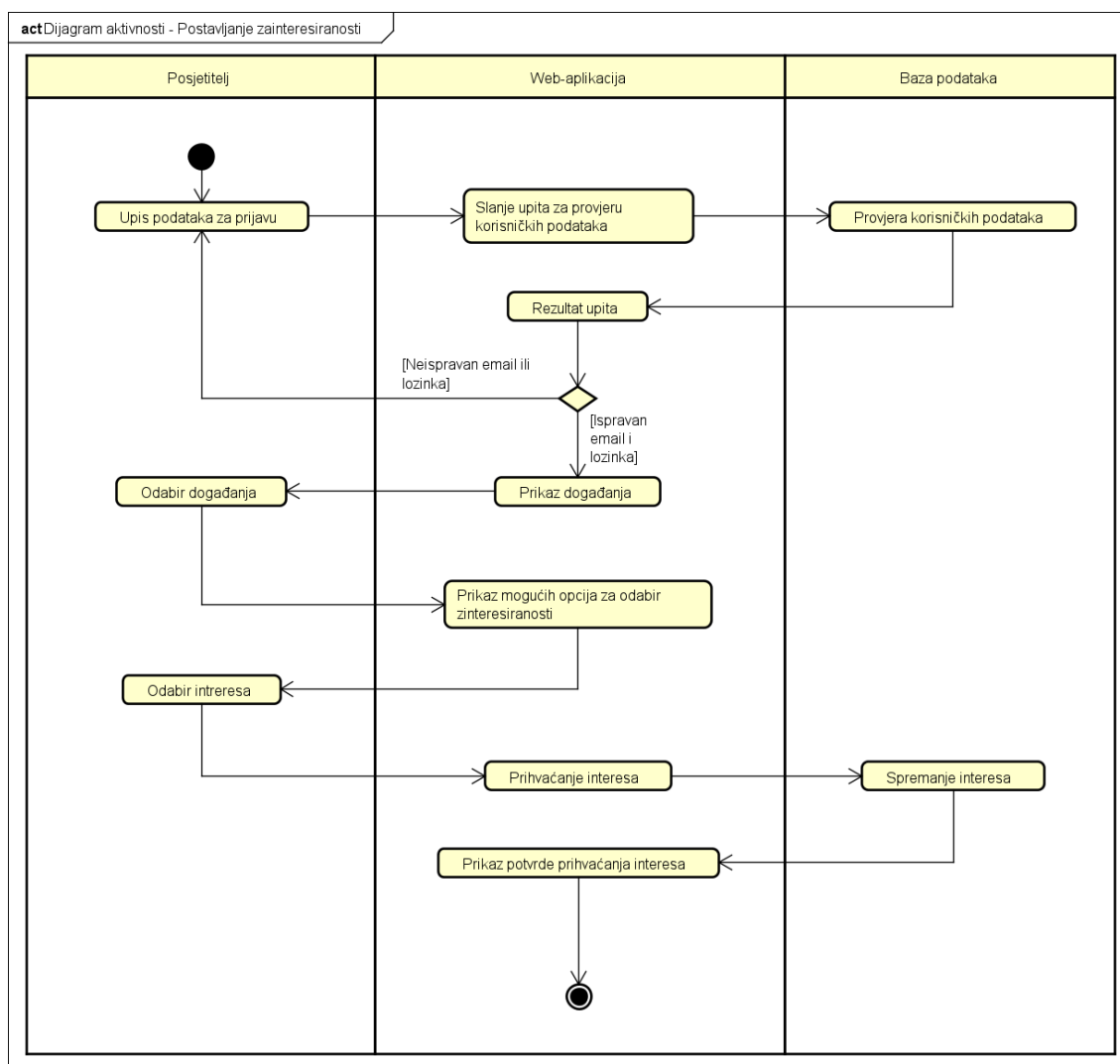
4.3 Dijagram stanja

Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljene na događajima. Slika 4.5 prikazuje dijagram stanja prijavljenog posjetitelja. Nakon prijave posjetitelju se na početnoj stranici prikazuje lista budućih događanja. Odabere li posjetitelj određeno događanje bit će mu ponuđene opcije da pročita više o događanju, označi dolaznost na događanje te da pogleda profil organizatora. Ako posjetitelj odabere pogledati profil organizatora otvoriti će mu se stranica u kojoj će moći vidjeti recenzije ostavljene na neka događanja organizatora, buduća i prošla događanja te podatke o samom organizatoru. Ako posjetitelj želi pregledati svoj profil to će moći napraviti tako da odabere "Profil". Na toj će stranici moći pogledati i urediti svoje podatke, obrisati svoj račun, promijeniti interes za svoja buduća događanja, ostaviti osvrt na događanje na kojem je bio te urediti ili obrisati recenziju koju je već napisao.

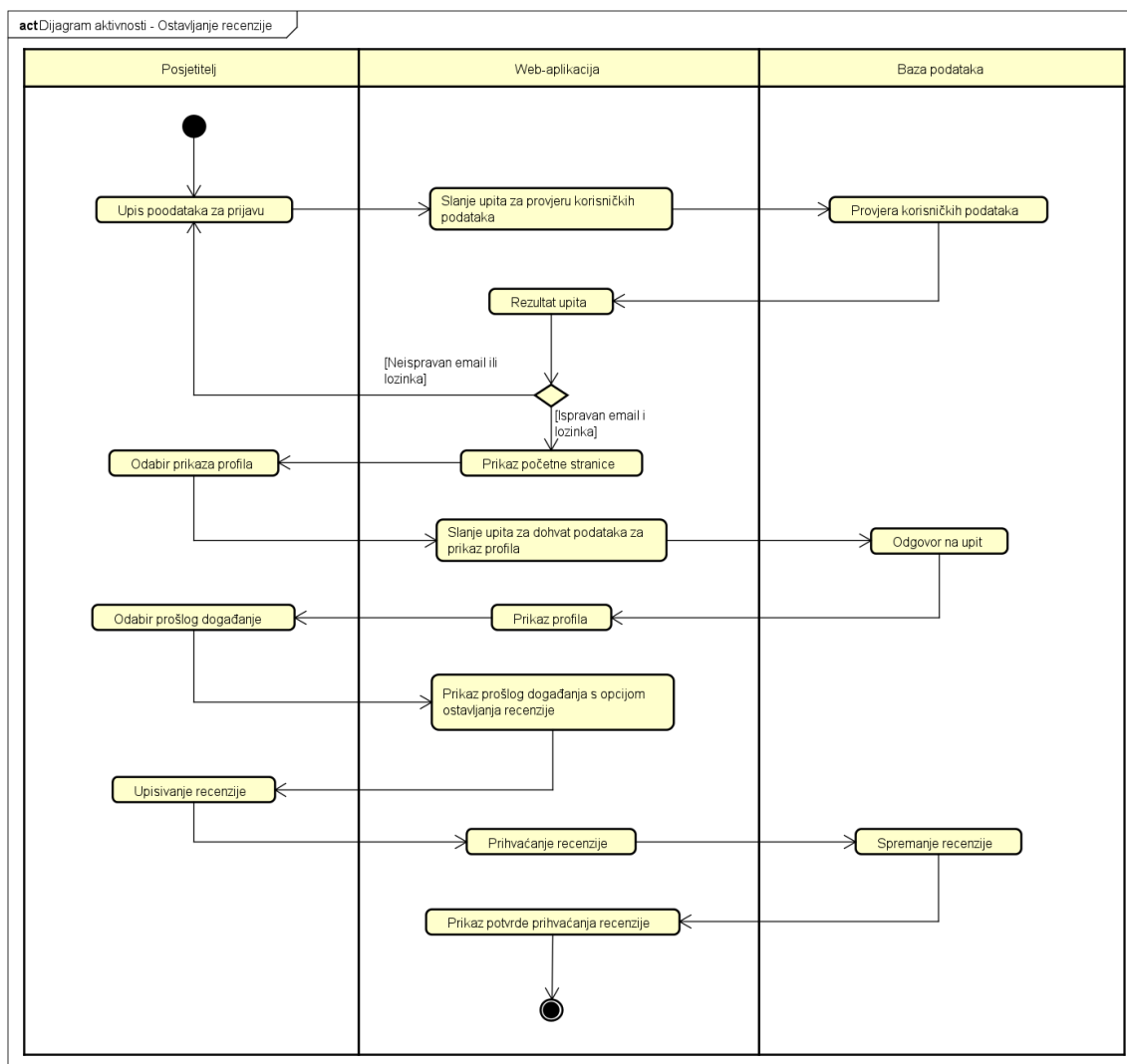


4.4 Dijagram aktivnosti

Dijagram aktivnosti primjenjuje se za opis modela toka upravljanja ili toka podataka. Kao i kod sekvencijskog dijagrama kod dijagrama aktivnosti poseban naglasak ima redoslijed kojim se aktivnosti izvršavaju. Zbog manjka kompleksnosti sustava u nastavku se nalaze dva dijagrama aktivnosti. Prvi dijagram (4.6) prikazuje operacije sustava potrebne za postavljanje interesa na određeni događaj koji se još nije dogodio. Drugi dijagram (4.7) prikazuje što se sve mora dogoditi da bi posjetitelj uspješno ostavio recenziju na događaj koji se već dogodio.



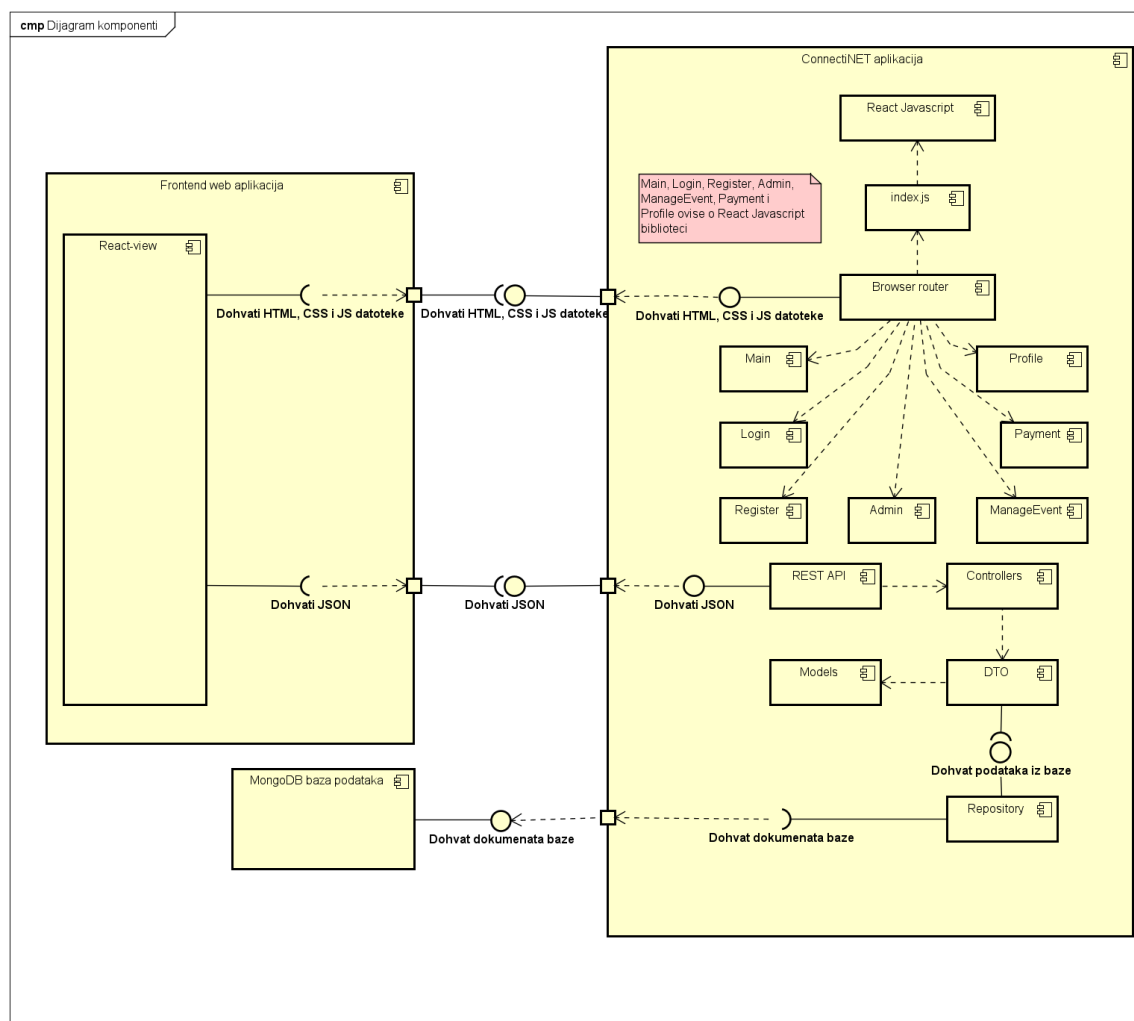
Slika 4.6: Dijagram aktivnosti - Postavljanje zainteresiranosti



Slika 4.7: Dijagram aktivnosti - Ostavljanje recenzije

4.5 Dijagram komponenti

Slika (4.8) prikazuje dijagram komponenti. Dijagram opisuje unutarnju strukturu sustava. Kako bi aplikacija funkcionirala frontend dio aplikacije se povezuje dvama sučeljima sa samom aplikacijom. Prvo mu sučelje služi za dohvaćanje HTML, JS i CSS datoteka. Aplikacija koristi Browser router za preusmjeravanje datoteka koje se trebaju poslati na sučelje. JavaScript datoteke koje Browser router može poslužiti nazvane su po prikazima (*View*) koji se mogu otvoriti u poslužitelju. Iste te datoteke koriste React o kojem ovise u izgradnji izgleda svih komponenata stranice. S druge strane za dohvat JSON podataka koristi se sučelje koje je povezano s REST API komponentom. REST API služi za posluživanje podataka koji pripadaju backend djelu. Repository dohvaća dokumente iz same MongoDB baze i preko MVC arhitekture ih šalje na obradu. Unutar frontend aplikacije nalazi se komponenta Reactview koja pomoću sučelja izmjenjuje podatke s aplikacijom i otvara ili osvježava prikaze u poslužitelju.



Slika 4.8: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Za komunikaciju u timu korištene su aplikacije WhatsApp¹ i Discord². UML dijagrami izrađeni su pomoću alata Astah UML³, a zajedno s ostalim materijalima i informacijama u dokumentaciju su uneseni pomoću alata za uređivanje LaTeX dokumenata koji se zove Texmaker⁴ i beplatne verzije online alata iste vrste Overleaf⁵. Kao sustav za upravljanje izvornim kodom korišten je Git⁶, a udaljeni repozitorij nalazio se na web platformi GitHub⁷.

Kao razvojno okruženje korišteni su Visual Studio Code⁸, uređivač koda koji je razvijen od strane kompanije Microsoft za operacijske sustave Windows, Linux i macOS, i IntelliJ IDEA⁹, integrirano razvojno okruženje (IDE) razvijeno od strane kompanije JetBrains. Backend je realiziran pomoću radnog okvira Node.js¹⁰ koji omogućuje korištenje jezika JavaScript¹¹ na serverskoj strani aplikacije. API je pisan u jeziku Java¹² uz korištenje radnog okvira Java Spring Boot¹³ koji ima ugrađenu podršku za zadatke poput upravljanja konfigura-

¹<https://www.whatsapp.com/>

²<https://discord.com/>

³<https://astah.net/products/astah-uml/>

⁴<https://www.xmlmath.net/texmaker/>

⁵<https://www.overleaf.com/>

⁶<https://git-scm.com/>

⁷<https://github.com/>

⁸<https://code.visualstudio.com/>

⁹<https://www.jetbrains.com/idea/>

¹⁰<https://nodejs.org/>

¹¹<https://www.javascript.com/>

¹²<https://www.java.com/>

¹³<https://spring.io/projects/spring-boot/>

cijom, pristupa podacima u bazi podataka i integracije sigurnosti u aplikaciju. Aplikacija pristupa MongoDB¹⁴ bazi podataka za čiji je lakši pregled korišten alat MongoDB Compass¹⁵. Frontend je realiziran pomoću biblioteke React¹⁶, također poznate kao React.js ili ReactJS, koja omogućuje pisanje HTML koda u JavaScript datotekama, što skupa s modularnom arhitekturom i komponentama koje pruža biblioteka, čini jednostavnim dinamičko generiranje korisničkih sučelja. Za lakše stvaranje nekih komponenti korisničkog sučelja korišten je radni okvir Bootstrap¹⁷ koji pruža HTML, CSS i JavaScript kod često potrebnih dijelova aplikacije.

Za testiranje API-ja korišten je alat Postman¹⁸ koji omogućuje slanje različitih vrsta zahtjeva na server. Aplikacija je testirana pomoću alata Selenium WebDriver¹⁹ i JUnit²⁰, koji omogućuju automatizaciju testiranja web aplikacija pomoću programskog sučelja. Interakcija Selenium WebDrivera s Google Chrome pretraživačem ostavljena alatom ChromeDriver²¹.

¹⁴<https://www.mongodb.com/>

¹⁵<https://www.mongodb.com/products/tools/compass>

¹⁶<https://react.dev/>

¹⁷<https://getbootstrap.com/>

¹⁸<https://www.postman.com/>

¹⁹<https://www.selenium.dev/documentation/webdriver/>

²⁰<https://junit.org/>

²¹<https://chromedriver.chromium.org/>

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Testni slučaj 1: Provjera addEvent funkcije u Event servisu

Ulaz:

1. Inicijalizacija događanja
2. Dodavanje imena "Test Event" događanju
3. Pokretanje funkcije *addEvent*
4. Provjera podudaranja testnog imena s imenom novo nastalog događanja u bazi

Očekivani rezultat:

1. Imena će se podudarati

Rezultat:

1. Imena se podudaraju



```
1  @Test
2  @DisplayName("Test addEvent")
3  void testAddEvent() {
4      Event inputEvent = new Event();
5      inputEvent.setEventName("Test Event");
6
7      Event result = eventService.addEvent(inputEvent);
8
9      assertEquals("Test Event", result.getEventName());
10 }
```

Slika 5.1: Prvi testni slučaj - addEvent

Testni slučaj 2: Provjera editEvent funkcije u Event servisu

Ulaz:

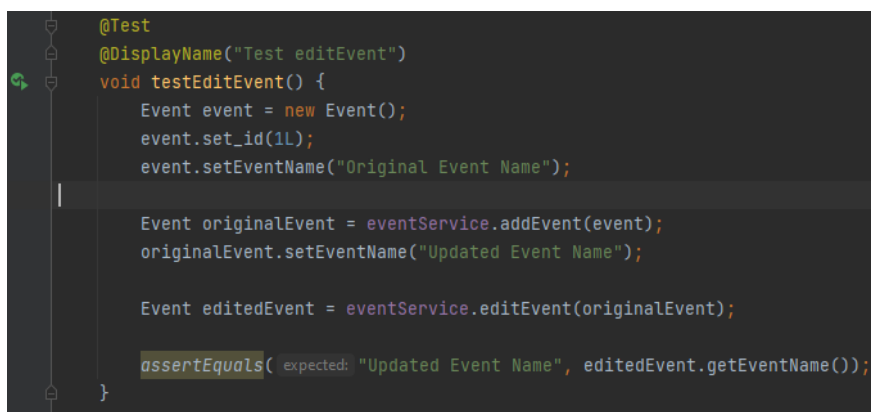
1. Inicijalizacija događanja
2. Dodavanje imena "Original Event Name" i ID-a događanju
3. Spremanje događanja pomoću funkcije *addEvent*
4. Zadavanje novog imena "Updated Event Name" događanju
5. Pokretanje funkcije *editEvent*
6. Usporedba uređenog imena sa imenom događanja u bazi

Očekivani rezultat:

1. Imena će se podudarati

Rezultat:

1. Događanje ima novo ime koje se podudara "Updated Event Name"

The image shows a screenshot of a code editor with a dark background. On the left, there is a vertical toolbar with icons for running, debugging, and other IDE functions. The code is written in Java and is part of a test class. It starts with an annotation `@Test` and `@DisplayName("Test editEvent")`. The method `testEditEvent()` is defined. Inside, it creates a new `Event` object, sets its ID to 1, and sets its name to "Original Event Name". Then, it calls `eventService.addEvent(event)` to save it. Next, it calls `eventService.editEvent(originalEvent)` to update the name. Finally, it uses `assertEquals` to verify that the updated name is "Updated Event Name".

```
@Test
@DisplayName("Test editEvent")
void testEditEvent() {
    Event event = new Event();
    event.set_id(1L);
    event.setEventName("Original Event Name");

    Event originalEvent = eventService.addEvent(event);
    originalEvent.setEventName("Updated Event Name");

    Event editedEvent = eventService.editEvent(originalEvent);

    assertEquals( expected: "Updated Event Name", editedEvent.getEventName());
}
```

Slika 5.2: Drugi testni slučaj - editEvent

Testni slučaj 3: Provjera `getUserById` funkcije u User servisu

Ulaz:

1. Zadavanje ID-a za pronalaženje
2. Dohvaća se korisnik s tim ID-em izravno iz baze
3. Pokretanje funkcije `getUserById` za zadani ID
4. Usporedba dohvaćenog korisnika izravno iz baze i korisnika dohvaćenog preko funkcije `getUserById`

Očekivani rezultat:

1. Korisnici su jednaki

Rezultat:

1. Dohvaćeni korisnici se podudaraju



```
@Test
@DisplayName("Test getUserById")
void testGetUserById() {
    String userId = "capsigmail.com";
    Optional<User> mockUser = userRepository.findByEmail(userId);
    when(userService.getUserById(userId)).thenReturn(mockUser);

    Optional<User> result = userService.getUserById(userId);

    assertEquals(mockUser, result);
}
```

Slika 5.3: Treći testni slučaj - `getUserById`

Testni slučaj 4: Provjera `getUserById` funkcije u User servisu (neuspješno)

Ulaz:

1. Zadavanje ID-a za pronalaženje
2. Dohvaća se korisnik s tim ID-em izravno iz baze
3. Pokretanje funkcije `getUserById` za zadani ID
4. Usporedba dohvaćenog korisnika izravno iz baze i korisnika dohvaćenog preko funkcije `getUserById`

Očekivani rezultat:

1. Uspoređuju se dvije prezne vrijednosti jer takav korisnik ne postoji

Rezultat:

1. Test prolazi jer su obje vraćene vrijednosti prazne



```
@Test
@DisplayName("Test getUserById not found")
void testGetUserByIdNotFound() {
    String userId = "nepostojeci@gmail.com";
    when(userRepository.findById(userId)).thenReturn(Optional.empty());

    Optional<User> result = userService.getUserById(userId);

    assertEquals("expected: Optional.empty(), result);", result);
}
```

Slika 5.4: Četvrti testni slučaj - `getUserById` (nepostojeći)

Testni slučaj 5: Provjera allEvents funkcije u Event servisu

Ulaz:

1. Dohvaćanje svih događanja izravno iz baze
2. Pozivanje funkcije *allEvents*
3. Usporedba dohvaćenih događanja iz baze s dohvaćenim događanjima preko *allEvents* funkcije

Očekivani rezultat:

1. Dobivena događanja se podudaraju

Rezultat:

1. Popisi događanja se podudaraju



```
@Test
@DisplayName("Test AllEvents")
void testAllEvents(){
    List<Event> allEvents = eventRepository.findAll();

    assertEquals(eventService.allEvents(), allEvents);
}
```

Slika 5.5: Peti testni slučaj - allEvents

Testni slučaj 6: Provjera `getEventById` funkcije u Event servisu

Ulaz:

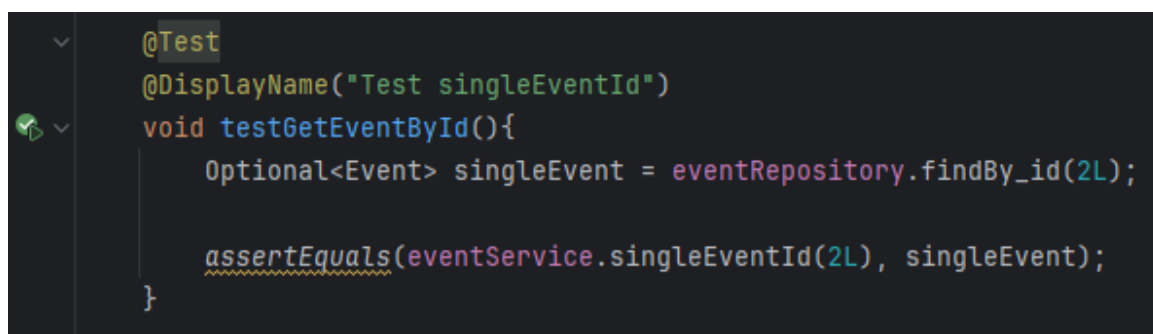
1. Dohvaćanje događanja preko ID-a izravno iz baze
2. Pozivanje funkcije `getEventById` za isti ID
3. Usporedba dohvaćenog događanja iz baze s dohvaćenim događanjem preko `getEventById` funkcije

Očekivani rezultat:

1. Dobivena događanja se podudaraju

Rezultat:

1. Događanja se podudaraju

A screenshot of a code editor showing a Java unit test. The code is as follows:

```
@Test
@DisplayName("Test singleEventId")
void testGetEventById(){
    Optional<Event> singleEvent = eventRepository.findById(2L);

    assertEquals(eventService.singleEventId(2L), singleEvent);
}
```

Slika 5.6: Šesti testni slučaj - `getUserById`

Uspješnost testova - Ispitivanje komponenata

✓ ProjektSpajaliceApplicationTests (col 5 sec 385 ms		✓ Tests passed: 6 of 6 tests – 5 sec 385 ms
✓ Test editEvent	269 ms	"C:\Program Files\Java\jdk-21\bin\j
✓ Test addEvent	2 sec 95 ms	17:27:45.197 [main] INFO org.spring
✓ Test AllEvents	2 sec 893 ms	17:27:45.399 [main] INFO org.spring
✓ Test singleEventId	66 ms	17:27:45.668 [main] INFO org.spring
✓ Test getUserByld	56 ms	
✓ Test getUserByld not found	6 ms	

Slika 5.7: Svi testovi komponenti

5.2.2 Ispitivanje sustava

Testni slučaj 1: Provjera *Login* funkcionalnosti (UC3)

Ulaz:

1. Otvaranje login stranice
2. Upisivanje podataka za prijavu
3. Unos podataka u aplikaciju

Očekivani rezultat:

1. Dobiven novi url kojim se dolazi na prijavljeno sučelje

Rezultat:

1. Test uspješan


```
@org.junit.Test
public void testLoginGoodCredentials(){

    WebDriverManager.chromedriver().setup();
    WebDriver driver = new ChromeDriver();

    driver.manage().timeouts().implicitlyWait(time: 10, TimeUnit.SECONDS);

    driver.get("http://localhost:3000/login");

    WebElement element = driver.findElement(By.name("Email"));
    element.sendKeys(...keysToSend: "admin@gmail.com");

    element = driver.findElement(By.name("password"));
    element.sendKeys(...keysToSend: "Admin123");

    driver.findElement(By.cssSelector("input[type='submit']")).click();

    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(1));
    wait.until(ExpectedConditions.urlToBe("http://localhost:3000/"));
    String redirectUrl = driver.getCurrentUrl();

    boolean success = redirectUrl.equals("http://localhost:3000/");
    assertEquals(expected: true, success);

    driver.quit();
}
```

Slika 5.8: Prvi testni slučaj - Login

Testni slučaj 2: Provjera *Login* funkcionalnosti (pogrešni podatci) (UC3)

Ulaz:

1. Otvaranje login stranice
2. Upisivanje podataka za prijavu
3. Unos podataka u aplikaciju

Očekivani rezultat:

1. Nije dobiven traženi url i javlja se poruka o pogrešno unesenim podacima

Rezultat:

1. Test uspješan

```
@org.junit.Test
public void testLoginBadCredentials(){

    WebDriverManager.chromedriver().setup();
    WebDriver driver = new ChromeDriver();

    driver.manage().timeouts().implicitlyWait(time: 10, TimeUnit.SECONDS);

    driver.get("http://localhost:3000/login");

    WebElement element = driver.findElement(By.name("Email"));
    element.sendKeys(...keysToSend: "random");

    element = driver.findElement(By.name("password"));
    element.sendKeys(...keysToSend: "random");

    driver.findElement(By.cssSelector("input[type='submit']")).click();
    String redirectUrl = driver.getCurrentUrl();

    boolean success = redirectUrl.equals("http://localhost:3000");
    assertEquals(expected: false, success);

    driver.quit();
}
```

Slika 5.9: Drugi testni slučaj - Login (pogrešni podatci)

Testni slučaj 3: Provjera funkcionalnosti kreiranja događanja (UC14)**Ulaz:**

1. Otvaranje login stranice
2. Prijava
3. Otvaranje padajućeg izbornika
4. Odabir opcije stvaranja događanja
5. Upisivanje podataka o događanju
6. Unos podataka u aplikaciju

Očekivani rezultat:

1. Otvaranje *pop up* prozora s potvrdom o stvaranju događanja

Rezultat:

1. Test uspješan

```
public void testCreateEvent() throws InterruptedException {  
  
    WebDriverManager.chromedriver().setup();  
    WebDriver driver = new ChromeDriver();  
    driver.manage().timeouts().implicitlyWait(time: 10, TimeUnit.SECONDS);  
    driver.get("http://localhost:3000/login");  
    WebElement element = driver.findElement(By.name("Email"));  
    element.sendKeys(...keysToSend: "admin@gmail.com");  
    element = driver.findElement(By.name("password"));  
    element.sendKeys(...keysToSend: "Admin123");  
    driver.findElement(By.cssSelector("input[type='submit']")).click();  
    Thread.sleep(millis: 1000);  
    WebElement menuDiv = driver.findElement(By.className("menu"));  
    menuDiv.click();  
    WebElement createEvent=driver.findElement(By.xpath(xpathExpression: "//*[@li[text()='Create Event']]");  
    createEvent.click();  
    WebElement form= driver.findElement(By.className("manageEvent--form"));  
    form.findElement(By.id("eventName")).sendKeys(...keysToSend: "Test Event");  
    WebElement dropdownType= form.findElement(By.id("eventType"));  
    Select dropdownT = new Select(dropdownType);  
    dropdownT.selectByVisibleText("EXP0");  
    WebElement dropdownLocation= form.findElement(By.id("eventLocation"));  
    Select dropdownL = new Select(dropdownLocation);  
    dropdownL.selectByVisibleText("Slavonski Brod");  
    form.findElement(By.id("eventDate")).sendKeys(...keysToSend: "06/25/2024");  
    form.findElement(By.id("eventStartTime")).sendKeys(...keysToSend: "13:00");  
    form.findElement(By.id("eventDuration")).sendKeys(...keysToSend: "17:00");  
    form.findElement(By.id("eventUrl")).sendKeys(...keysToSend: "www.test.com");  
    form.findElement(By.id("eventDescription")).sendKeys(...keysToSend: "Test description");  
    By uploadButtonXPath = By.xpath(xpathExpression: "//*[@div[@class='upload__image-wrapper']]//button[contains(text(), 'Click or Drop here')]");  
    WebElement fileInput=driver.findElement(uploadButtonXPath);  
    fileInput.click();  
    Thread.sleep(millis: 7000);  
    driver.findElement(By.cssSelector("input[type='submit']")).click();  
    Thread.sleep(millis: 2000);  
    Alert alert = driver.switchTo().alert();  
    String alertText = alert.getText();  
    alert.accept();  
  
    boolean success = alertText.equals("Event added!");  
    assertEquals(expected: true, success);  
    driver.quit();  
}
```

Slika 5.10: Treći testni slučaj - Kreiranje događanja

Testni slučaj 4: Provjera funkcionalnosti promjene lozinke (UC5)**Ulaz:**

1. Otvaranje login stranice
2. Prijava
3. Otvaranje padajućeg izbornika
4. Odabir opcije pregleda profila
5. Klik na gumb za promjenu lozinke
6. Upisivanje stare i nove lozinke
7. Unos podataka u aplikaciju

Očekivani rezultat:

1. Otvaranje *pop up* prozora s potvrdom o promjeni lozinke

Rezultat:

1. Test uspješan

```
@Test
public void testChangePassword() throws InterruptedException {

    WebDriverManager.chromedriver().setup();
    WebDriver driver = new ChromeDriver();
    driver.manage().timeouts().implicitlyWait(time: 10, TimeUnit.SECONDS);
    driver.get("http://localhost:3000/login");
    WebElement element = driver.findElement(By.name("Email"));
    element.sendKeys(...keysToSend: "admin@gmail.com");
    element = driver.findElement(By.name("password"));
    element.sendKeys(...keysToSend: "Admin123");
    driver.findElement(By.cssSelector("input[type='submit']")).click();
    Thread.sleep(millis: 1000);
    WebElement menuDiv = driver.findElement(By.className("menu"));
    menuDiv.click();
    WebElement editProfile=driver.findElement(By.xpath(xpathExpression: "//*[@li[text()='Profile']]"));
    editProfile.click();
    driver.findElement(By.className("changePasswordButton")).click();
    driver.findElement(By.id("oldPassword")).sendKeys(...keysToSend: "Admin123");
    driver.findElement(By.id("newPassword")).sendKeys(...keysToSend: "Admin321");
    driver.findElement(By.id("confirmNewPassword")).sendKeys(...keysToSend: "Admin321");
    driver.findElement(By.cssSelector("input[type='submit']")).click();
    Thread.sleep(millis: 2000);
    Alert alert = driver.switchTo().alert();
    String alertText = alert.getText();
    alert.accept();

    boolean success = alertText.equals("Password edited!");
    assertEquals(expected: true, success);
    driver.quit();
}
```

Slika 5.11: Četvrti testni slučaj - Promjena lozinke

Testni slučaj 5: Provjera funkcionalnosti promjene prikaza profila

Ulaz:

1. Otvaranje login stranice
2. Prijava
3. Otvaranje padajućeg izbornika
4. Odabir opcije pregleda profila
5. Klik na gumb za otvaranje prikaza javnog profila

Očekivani rezultat:

1. Dobiven novi url kojim se dolazi na prikaz javnog profila

Rezultat:

1. Test uspješan

```
@Test
public void testChangeProfileView() throws InterruptedException {

    WebDriverManager.chromedriver().setup();
    WebDriver driver = new ChromeDriver();
    driver.manage().timeouts().implicitlyWait(time: 10, TimeUnit.SECONDS);
    driver.get("http://localhost:3000/login");
    WebElement element = driver.findElement(By.name("Email"));
    element.sendKeys(keysToSend: "capsidomi@gmail.com");
    element = driver.findElement(By.name("password"));
    element.sendKeys(keysToSend: "capsidomi");
    driver.findElement(By.cssSelector("input[type='submit']")).click();
    Thread.sleep(millis: 1000);
    WebElement menuDiv = driver.findElement(By.className("menu"));
    menuDiv.click();
    WebElement editProfile = driver.findElement(By.xpath(xpathExpression: "//li[text()='Profile']"));
    editProfile.click();
    driver.findElement(By.className("viewPublicButton")).click();
    String redirectUrl = driver.getCurrentUrl();
    System.out.println(redirectUrl);

    boolean success = redirectUrl.equals("http://localhost:3000/profile/public/capsidomi@gmail.com");
    assertEquals(expected: true, success);

    driver.quit();
}
```

Slika 5.12: Peti testni slučaj - Otvaranje prikaza javnog profila

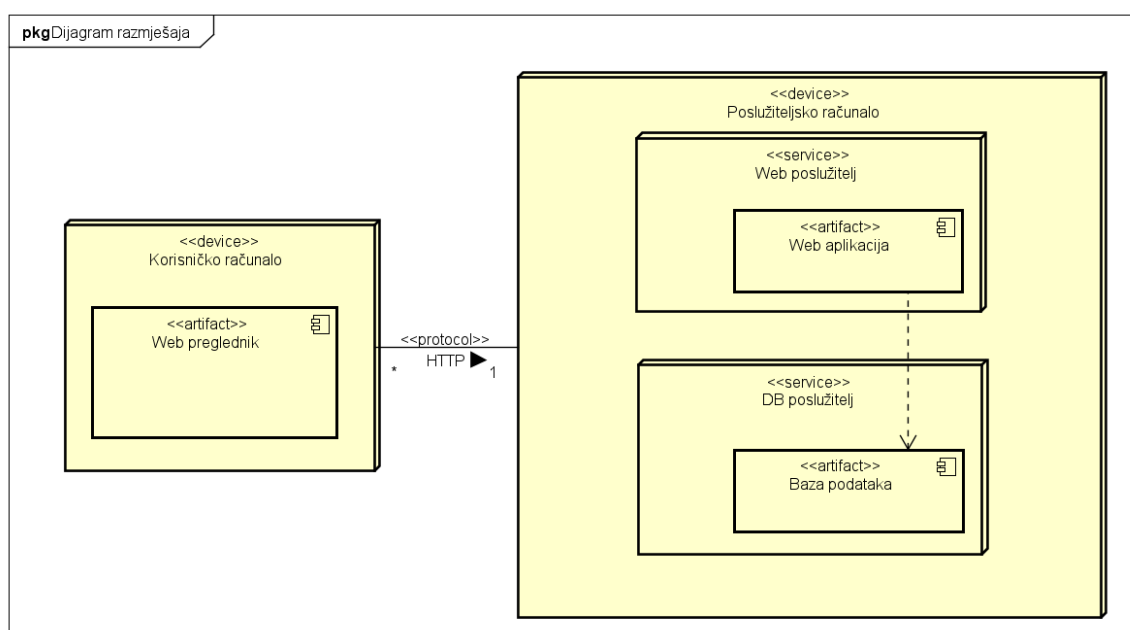
Uspješnost testova - Ispitivanje sustava

✓ Test	7 sec 885 ms	✓ Tests passed: 2 of 2 tests – 7 sec 885 ms
✓ testLoginBadCredentials	4 sec 353 ms	"C:\Program Files\Java\jdk-21\bin\
✓ testLoginGoodCredentials	3 sec 532 ms	18:01:46.776 [main] INFO io.github
✓ Test	15 sec 994 ms	✓ Tests passed: 1 of 1 test – 15 sec 994 ms
✓ testCreateEvent	15 sec 994 ms	"C:\Program Files\Java\jdk-21\bin
✓ ProjektSpajaliceApplicationTests (com.spajalice.ProjektSp	8 sec 660 ms	✓ Tests passed: 1 of 1 test – 8 sec 660 ms
✓ testChangePassword()	8 sec 660 ms	"C:\Program Files\Java\jdk-21\bin
✓ ProjektSpajaliceApplicationTests (com.spajalice.ProjektSp	6 sec 472 ms	✓ Tests passed: 1 of 1 test – 6 sec 472 ms
✓ testChangeProfileView()	6 sec 472 ms	"C:\Program Files\Java\jdk-21\bi

Slika 5.13: Svi testovi sustava

5.3 Dijagram razmještaja

Dijagram razmještaja (5.14) prikazuje fizičku arhitekturu i konfiguraciju razmještaja programskog sustava. Poslužiteljsko računalo sadrži web poslužitelj na kojem je web aplikacija i poslužitelj baze podataka koji omogućava pristup bazi. Klijent se na sustav spaja pomoću vlastitog osobnog računala koje uspostavlja komunikaciju preko HTTP protokola s poslužiteljskim računalom.



Slika 5.14: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Da bi se backend pripremio za deploy na Render, prvo treba provjeriti jesu li postavljene env. varijable u konfiguraciju vašeg IDE-a i po potrebi ih dodati. Potrebno je dodati i Dockerfile koji ima dvije dostupne verzije koje se nalaze u direktoriju docker. Jedna se nalazi u poddirektoriju Maven dok se druga nalazi u poddirektoriju Gradle. Ukoliko se mijenja lokacija Dockerfile-a, treba paziti na putanje COPY naredbi u Dockerfile skripti.

U datoteci application.properties preporuča se postaviti svojstvo `server.servlet.context-path` na `/api` kao prefiks svim zahtjevima na backend. Opcionalno je u application.properties dodati željene env. varijable u formatu `neko.svojstvo=${ENV_VAR_IME:default vrijednost}`. Također je opcionalno u datoteci pom.xml kao dependency dodati `spring-boot-starter-actuator` koji će na putanju `/actuator/health` automatski izložiti informaciju o statusu aplikacije koju može koristiti Render.

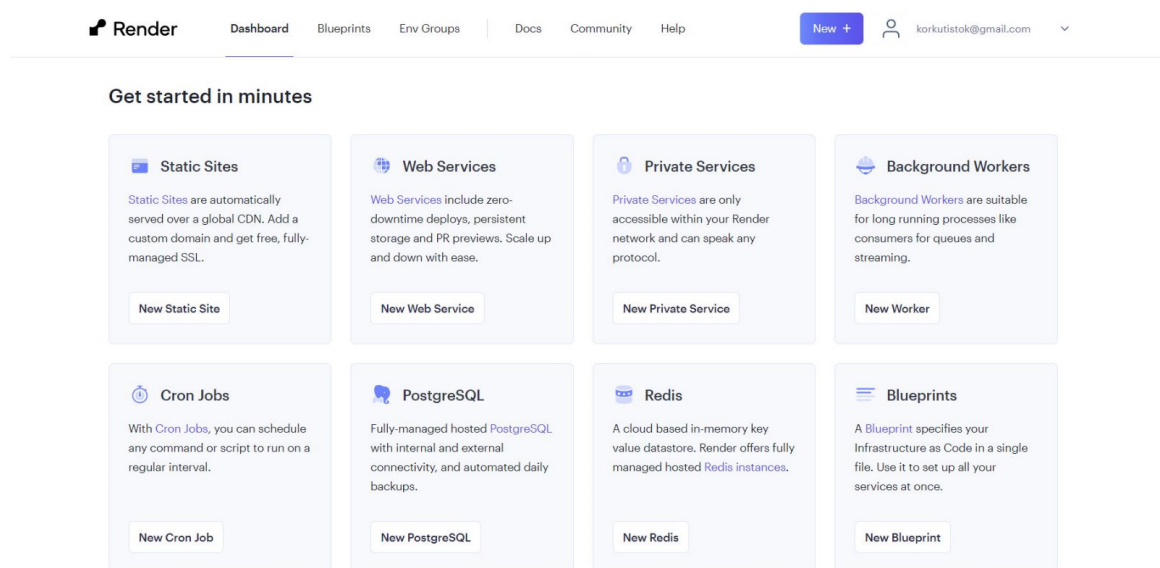
Priprema frontenda za deploy na Render

Potrebno je u package.json dodati svaki dependency potreban za deploy, primarno `http-proxy-middleware`, `dotenv` i `express`.

Također treba dodati `/src/setupProxy.js` koji služi kao proxy server pri lokalnom developmentu (preusmjerava API pozive na `localhost:8080`), odnosno pri korištenju `react-scripts start` skripte. Nakon toga potrebno je dodati app.js datoteku, u kojoj se nalazi Express server za produkcijski proxy i posluživanje frontenda. Potrebno je i u datoteci package.json izmijeniti sljedeću skriptu: `"build": "yarn install && react-scripts build"` i dodati `"start-prod": "node app.js"`.

Kreiranje baze podataka

Nakon kreiranja Render profila, na stranici Render dashboard potrebno je pritisnuti na gumb **New +** i odabrati PostgreSQL. Zatim postaviti ime baze i opcionalno username za korisnika baze (password je automatski generiran). Pod Region odabrati Frankfurt. Odabrati željenu opciju pod Instance type. Besplatna baza podataka ima maksimalnu pohranu od 1GB, te se briše nakon 90 dana. Pritisnuti na gumb **Create Database**.



Slika 5.15: Korisničko sučelje alata Render dashboard

Kreiranje backenda

Na stranici Render dashboard potrebno je pritisnuti na gumb **New +** u gornjem desnom dijelu stranice te odabrati Web Service. U idućem koraku povezati projekt sa GitLab računom i pritisnuti **Next**, nakon čega su dostupni svi projekti na koje imate prava pristupa. Stisnuti **Connect** pored odgovarajućeg projekta. U otvorenoj konfiguraciji postaviti ime za servis koje će postati dio web-adrese.

Nakon toga je potrebno Root directory postaviti na progib-e. Za-

tim pod Environment odabrati Docker, a pod Region Frankfurt. Nakon toga proširiti konfiguraciju pritiskom na Advanced pri dnu stranice.

U toj sekciji dodati potrebne env.varijable (npr. DB username, password, URL) za čije je vrijednosti potrebno kopirati vrijednosti iz postavki baze podataka na Renderu. Pripaziti jer URL koji je prikazan na Renderu nije JDBC URL, za ovaj primjer treba biti u formatu `jdbc:postgresql://hostname:port/database`. Ukoliko je dodan Spring Boot Actuator (prema zadnjoj točki poglavlja pripreme za deploy), potrebno je postaviti `/api/actuator/health` kao Health Check Path (odnosno `/actuator/health`). Nakon toga postaviti putanju za Dockerfile u skladu s tim koji se package manager koristi (u ovom slučaju `./docker/maven/Dockerfile`). Na kraju pritisnuti gumb **Create Web Service**.

Kreiranje frontenda

Na stranici Render dashboard potrebno je pritisnuti na gumb **New** + u gornjem desnom dijelu stranice te odabrati Web Service. U idućem koraku povezati projekt sa GitLab računom i pritisnuti **Next**, nakon čega su dostupni svi projekti na koje imate prava pristupa. Stisnuti **Connect** pored odgovarajućeg projekta. U otvorenoj konfiguraciji postaviti ime za servis koje će postati dio web-adrese. Zatim pod Environment odabrati Node, a pod Region Frankfurt. Build Command postaviti na `yarn build`, a Start Command `yarn start-prod`. Nakon toga je potrebno Root directory postaviti na `progi-be`. Nakon toga proširiti konfiguraciju pritiskom na Advanced pri dnu stranice.

Pod Environment variables postaviti varijablu `API_BASE_URL` na adresu deployanog backenda aplikacije dostupnu na Render dashboardu. Na kraju stisnuti gumb **Create Web Service**.

6. Zaključak i budući rad

Nakon 14 tjedana izrade aplikacije za objavu i traženje događanja u blizini ostvaren je zadani cilj. Većina funkcionalnosti je implementirana te radi ispravno. Tim se pokazao izrazito plodonosan i usklađen. Prema zadanim kontrolnim točkama faze rada se mogu podijeliti u dvije cjeline.

Prva faza poslužila je za dogovor oko podjele poslova i zaključak kako će se u njoj veći fokus staviti na dokumentaciju. Iz tog je razloga tim podjeljen u tri manja podtima: backend, frontend i dokumentaciju. Definirani podtimovi nisu garantirali da će se njihovi članovi baviti samo područjem tog podtima već je donesena odluka kako će si članovi različitih podtimova međusobno pomagati kako bi svi iskusili svaki dio projekta. Tako je u ovoj fazi podtim dokumentacije dobivao konstantne povratne informacije o dogovorima oko arhitektura koje će se koristiti i na koji način. Kako bi se najkvalitetnije dogovorili svi su članovi bili prisutni na gotovo svim sastancima.

U drugoj je fazi stavljen naglasak na programskom rješenju te su se svi članovi aktivnije počeli baviti razvijanjem sustava u praksi. Kako bi svi znali što treba raditi poslužio im je plan iz dokumentacije sastavljen u prvoj fazi. U ovoj je fazi došlo do promjene načina komunikacije, projekt više nije zahtijevao prisutnost svih članova od jednom već bi se članovi međusobno savjetovali kako nebi prelazili preko tuđih rješenja. Dokumentacija se u ovoj fazi radila paralelno s razvojem programskog rješenja jer su određeni dijelovi zahtijevali testiranje samog rješenja.

Komunikacija se pokazala kao ključan faktor u uspješnosti i efikasnosti izrade projekta. Za uspješno uspostavljenu komunikaciju treba dati posebne pohvale voditelju grupe koji je kvalitetno podjelio zadatke i uvijek bio na usluzi kada je situacija to zahtjevala. Svi su članovi bili zainteresirani i odlučni u odluci da se projekt napravi na što bolji način.

Problem na koji su gotovo svi članovi naišli bio je manjak iskustva u korištenju odabranih alata. Manjak iskustva bio je vidljiv i u samom upravljanju projektom gdje je bilo potrebno vrijeme da svaki član ustanovi koja je točno njegova uloga u timu.

Radi manjka vremena neke zahtjevnije funkcionalnosti nisu mogle biti implementirane no zadatak je gotovo u potpunosti obavljen. Među funkcionalnostima koje nisu implementirane nalaze se pretplaćivanje na organizatora i objava videozapisa.

Tim nije pokazivao naznake ne slaganja i loših odnosa međutim odluka o nastavku razvijanja aplikacije nije donesena. Tim je zaključio kako je sudjelovanje na projektu bilo vrijedno iskustvo za daljnji rad no kako sama ideja aplikacije nema primjenjivu namjenu koju bi članovi htijeli poduprijeti. Tim je kolektivno zadovoljan s postignutim rezultatom i ne odbacuje mogućnost buduće suradnje.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. React, <https://react.dev/learn>
3. Spring, Spring Boot, <https://spring.io/projects/spring-boot>
4. MongoDB, <https://www.mongodb.com>
5. Geeks for geeks, MVC Framework, <https://www.geeksforgeeks.org/mvc-framework-introduction/?ref=gcse>
6. Astah Community, <http://astah.net/editions/uml-new>
7. Selenium WebDriver, <https://www.selenium.dev/documentation/webdriver/>

Indeks slika i dijagrama

2.1	Sučelje aplikacije Eventbrite	8
3.1	Dijagram obrasca uporabe, funkcionalnost korisnika i posjetitelja	23
3.2	Dijagram obrasca uporabe, funkcionalnost organizatora i administratora	24
3.3	Sekvencijski dijagram za UC2	26
3.4	Sekvencijski dijagram za UC5	27
3.5	Sekvencijski dijagram za UC14	29
3.6	Sekvencijski dijagram za UC20	31
4.1	Skica arhitekture sustava	33
4.2	Dijagram baze podataka	43
4.3	Dijagram razreda - Controllers	45
4.4	Dijagram razreda - Models	46
4.5	Dijagram stanja - Posjetitelj	48
4.6	Dijagram aktivnosti - Postavljanje zainteresiranosti . .	50
4.7	Dijagram aktivnosti - Ostavljanje recenzije	51
4.8	Dijagram komponenti	53
5.1	Prvi testni slučaj - addEvent	56
5.2	Drugi testni slučaj - editEvent	57
5.3	Treći testni slučaj - getUserId	58
5.4	Četvrti testni slučaj - getUserId (nepostojeći)	59
5.5	Peti testni slučaj - allEvents	60
5.6	Šesti testni slučaj - getUserId	61

5.7	Svi testovi komponenti	62
5.8	Prvi testni slučaj - Login	64
5.9	Drugi testni slučaj - Login (pogrešni podatci)	66
5.10	Treći testni slučaj - Kreiranje događanja	68
5.11	Četvrti testni slučaj - Promjena lozinke	70
5.12	Peti testni slučaj - Otvaranje prikaza javnog profila	71
5.13	Svi testovi sustava	72
5.14	Dijagram razmještaja	73
5.15	Korisničko sučelje alata Render dashboard	75
6.1	Aktivnost u repozitoriju	88

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 19. listopada 2023.
- Prisustvovali: članovi tima, asistent i demos
- Teme sastanka:
 - sastanak s asistentom i demonstratorom
 - raspodjela poslova
 - biranje alata

2. sastanak

- Datum: 23. listopada 2023.
- Prisustvovali: Svi članovi
- Teme sastanka:
 - dodatan dogovor oko raspodjele poslova
 - konačan odabir alata

3. sastanak

- Datum: 2. studenoga 2023.
- Prisustvovali: Mrvčić, Duvančić
- Teme sastanka:
 - dogovor oko funkcionalnosti
 - izrada dijagrama

4. sastanak

- Datum: 3. studenogaa 2023.
- Prisustvovali: Svi članovi

- Teme sastanka:
 - dogovor oko izgleda baze podataka
 - dogovor oko obrazca uporabe

5. sastanak

- Datum: 4. studenoga 2023.
- Prisustvovali: Duvančić, Mrvičić, Radošević
- Teme sastanka:
 - razrada obrazaca uporabe
 - izrada dijagrama

6. sastanak

- Datum: 9. studenoga 2023.
- Prisustvovali: članovi tima, asistent i demos
- Teme sastanka:
 - demonstrature - prva revizija uživo

7. sastanak

- Datum: 13. studenoga 2023.
- Prisustvovali: svi članovi tima
- Teme sastanka:
 - dogovor oko izrade tablica baza
 - dogovor oko povezivanja frontend i backenda

8. sastanak

- Datum: 16. studenoga 2023.
- Prisustvovali: svi članovi tima
- Teme sastanka:
 - zadnje implementacije
 - provjera dokumentacije

9. sastanak

- Datum: 17. studenoga 2023.

- Prisustvovali: svi članovi tima
- Teme sastanka:
 - predaja

10. sastanak

- Datum: 7. prosinca 2023.
- Prisustvovali: svi članovi tima
- Teme sastanka:
 - dogovor za drugi ciklus

11. sastanak

- Datum: 22. prosinca 2023.
- Prisustvovali: svi članovi tima
- Teme sastanka:
 - dogovor za rad preko praznika

12. sastanak

- Datum: 8. siječnja 2024.
- Prisustvovali: svi članovi tima
- Teme sastanka:
 - rješavanje problema u kodu

13. sastanak

- Datum: 15. siječnja 2024.
- Prisustvovali: svi članovi tima
- Teme sastanka:
 - dogovori oko zadnjih zadataka

14. sastanak

- Datum: 17. siječnja 2024.
- Prisustvovali: svi članovi tima
- Teme sastanka:
 - pregled napravljenog i dogovor za završetak

15. sastanak

- Datum: 19. siječnja 2024.
- Prisustvovali: svi članovi tima
- Teme sastanka:
 - dogovor oko predaje

Tablica aktivnosti

	Mario Mrvčić	Toma Žulj	Domagoj Capar	Luka Radošević	Josip Duvančić	Duje Jurić	Istok Korkut
Upravljanje projektom	12						
Opis projektnog zadatka				2		2	4
Funkcionalni zahtjevi	2	2	2	2	3	2	2
Opis pojedinih obrazaca	4	2	2	2	6	2	2
Dijagram obrazaca					4		
Sekvencijski dijagrami					4		3
Opis ostalih zahtjeva					1		
Arhitektura i dizajn sustava	10	15	6	6	6	12	6
Baza podataka	10		12		4		
Dijagram razreda				2			8
Dijagram stanja					3		
Dijagram aktivnosti					3		
Dijagram komponenti					5		
Korištene tehnologije i alati	12	13	14	7	5	7	8

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

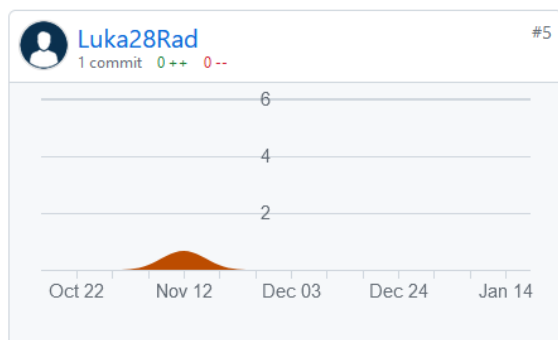
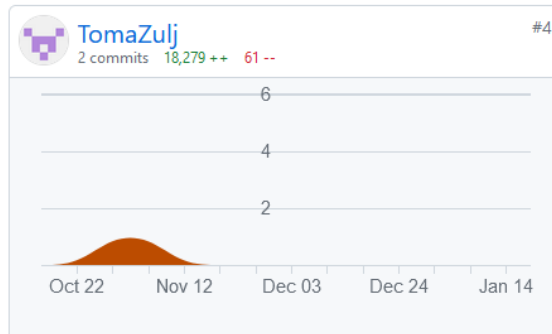
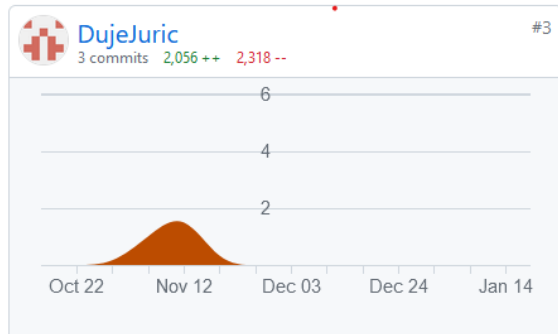
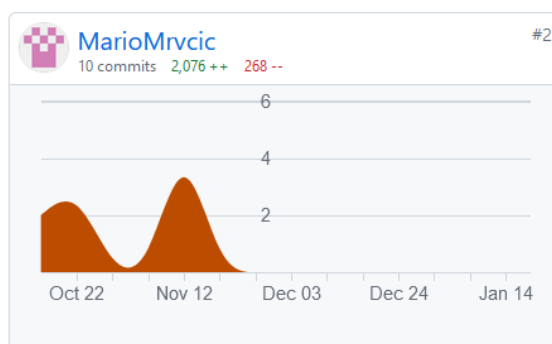
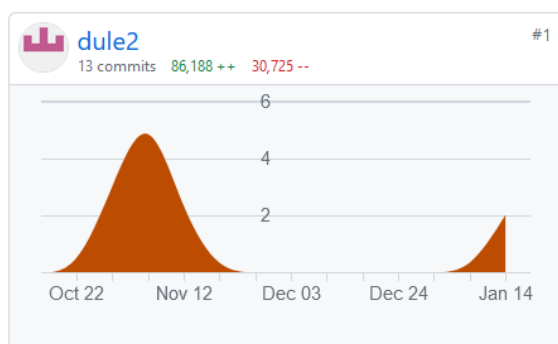
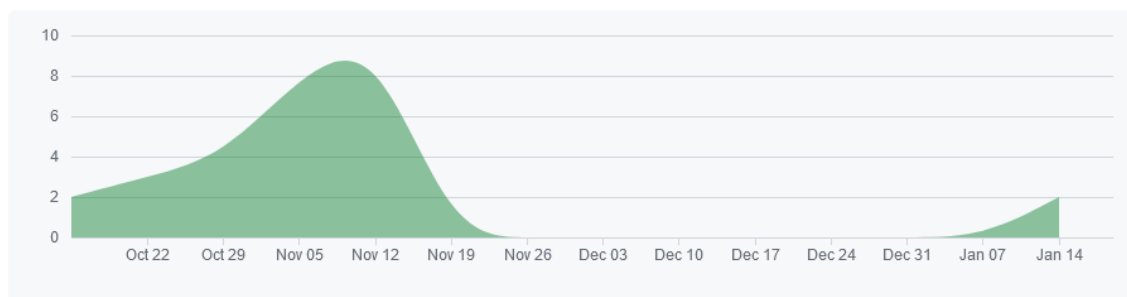
	Mario Mrvčić	Toma Žulj	Domagoj Capar	Luka Radošević	Josip Duvančić	Duje Jurić	Istok Korkut
Ispitivanje programskog rješenja	15	15	20			10	
Dijagram razmještaja					2		
Upute za puštanje u pogon	13		4				6
Backend	30	10	40	5		8	
Dnevnik sastajanja					2		
Zaključak i budući rad					2		
Popis literature					1		
Izrada početne stranice	10	30		20		35	
Izrada baze podataka	20		10				
Spajanje s bazom podataka	30		15				
Upoznavanje s alatima	12	15	18	12	6	10	12

Dijagrami pregleda promjena

Oct 15, 2023 – Jan 19, 2024

Contributions: Commits ▾

Contributions to main, excluding merge commits



Slika 6.1: Aktivnost u repozitoriju (samo main brancha)