

Jednostavna i sigurna razmjena i kupovina karata.

## Uvod

Što je to TickEx? Zamislite sljedeću situaciju: kupite kartu za koncert omiljenog izvođača, kartu za utakmicu omiljenog kluba ili kartu za Vašeg omiljenog komičara, ali zbog nekih okolnosti ne možete iskoristiti tu kartu. Jedina opcija koju imate je objavljivati oglase po Njuškalu, Facebook grupama ili "zivkati" prijatelje da se riješite karte. Sad možete preskočiti sve korake i otići ravno na TickEx!

Možda ste već koristili naizgled slične aplikacije poput Ticketmastera, TicketSwapa, StubHuba ili sličnih. Za razliku od njih naša web aplikacija omogućuje jednostavno sučelje i lako pretraživanje karata koje želite. Omogućuje direktnu komunikaciju s prodavačem bez komplikacija.

## Slične stranice

### TicketSwap

TicketSwap nudi korisnicima sigurnu zamjenu i kupovinu karata. Omogućuje praćenje izvođača te uvid u njihove nadolazeće nastupe. Korisnik može tražiti karte po lokaciji, izvođaču i datumu. Za određeni događaj može stvoriti notifikaciju, vidjeti prodane, tražene i dostupne karte. Korisnik plaća naknade od 9% za kupovinu te 5% za prodaju karte. Sigurnost karte osigurana je SecureSwapom kojim TicketSwap poništi barkod originalne karte i stvori novi u slučaju da imaju partnerstvo s organizatorom ili samim događajem. Spomenuti program partnerstva dozvoljava organizatorima raširenu prodaju karata.

You're seeing recommendations for 📍 Nearby ▼

## Find your next event

Today



Tomorrow



This weekend



Explore all



## Trending near you

Find events right up your alley.



Fri, Nov 15, 7:30 PM

André Rieu

Arena Zagreb, Zagreb

10



Jul 30 - Aug 3

Defected Croatia 2025

The Garden, Tisno

5



Jun 29 - Jul 3

Verknijpt Croatia Holiday Festival 2...

Zrće Beach, Novska

7

## Follow your faves



Tyler, The Creator

21 upcoming events

Follow



Pegassi

5 upcoming events

Follow



Odimel

6 upcoming events

Follow



Steve Redhead

1 upcoming event

Follow



Miamor

1 upcoming event

Follow



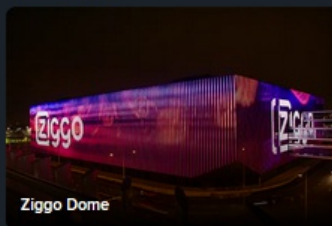
Helena Lauwaert

3 upcoming events

Follow

## Locations

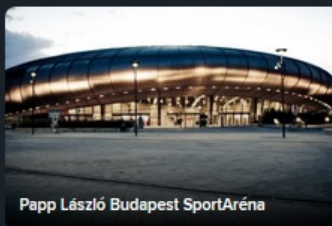
Popular event locations.



Ziggo Dome



Antwerps Sportpaleis



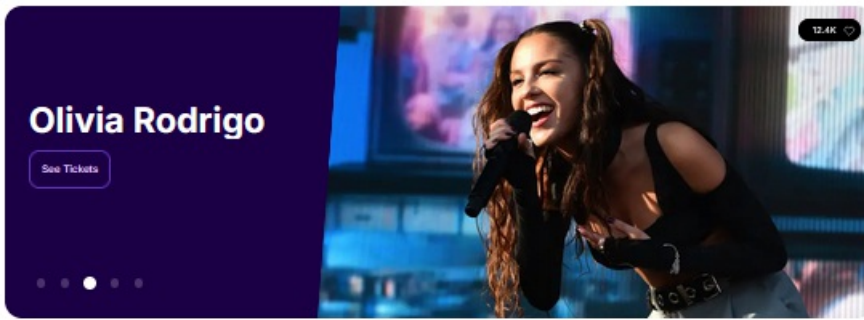
Papp László Budapest SportAréna

Snimka zaslona početne stranice [TicketSwapa](#)

## StubHub

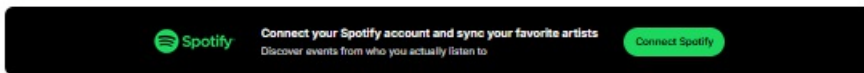
StubHub nudi korisnicima kupnju karata. Nudi karte iz raznih događanja poput bejzbola, nogometa, američkog nogometa, glazbe, kazališta i sličnih. Omogućuju pretraživanje po nazivu, pregled određenih kategorija ovisno o sportskom timu, ligi, žanru, festivalu... Moguće je i "lajkati" izvođače, timove i događaje. Pri kupnji karti vidljiv je prikaz lokacije i sjedala. Prodavač može slobodno odrediti cijenu, naravno StubHub uzima proviziju. Više je centrirana oko fanova koji prodaju karte nego organizatora. Korisnici mogu spojiti svoj Spotify za preporuke.

Search events, artists, teams, and more

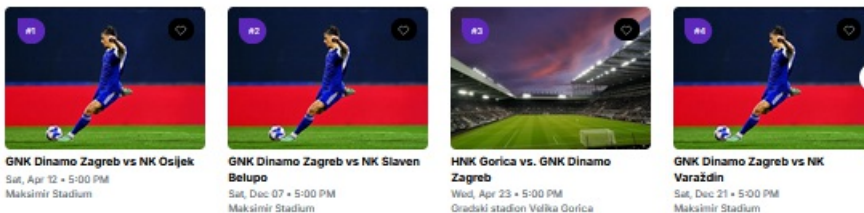


Explore events in

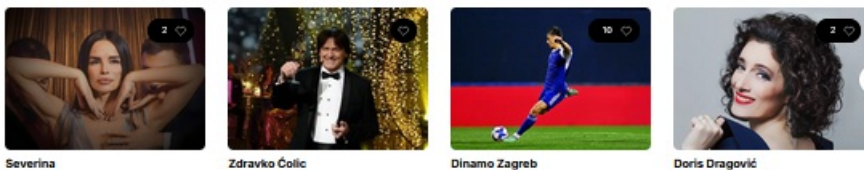
Use my location Zagreb All Dates



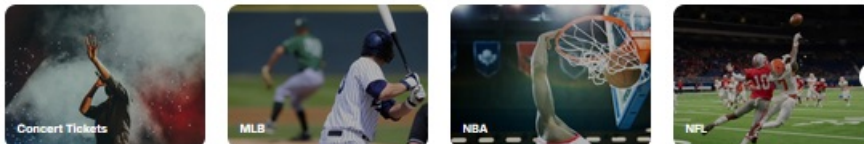
#### Trending Events near Zagreb



#### Recommended for You



#### Popular Categories



#### Last Minute Deals

Snimka zaslona početne stranice [StubHuba](#)

## Korisnici i koristi TickExa

Ovaj projekt pruža značajnu korist svima koji posjećuju razna događanja, olakšavajući proces nabave karata i uklanjajući stres povezan s time. Olakšava proces ljudima koji se slabije snalaze u online trgovinama karata ili pak ne žele davati podatke prodavačima. Svaki korisnik, koji ne želi prolaziti kroz muku prodaje stare karte i kupnje karte koju želi, može jednostavno obaviti zamjenu.

TickEx je web aplikacija pomoću koje korisnici mogu prodavati i mijenjati svoje karte ili kupiti karte koje drugima nisu potrebne. Korisnici mogu gledati karte bez registracije, a uz jednostavnu registraciju koristeći račune s drugih aplikacija (npr. Google) mogu početi mijenjati, kupovati i prodavati karte. Korisnici mogu uređivati svoje profile i podatke o sebi. Imaju pristup svim svojim oglasima i povijesti transakcija. Svoje oglase mogu uređivati i staviti u košaricu ako razmatraju obrisati oglas.

Oglasi sadrži razne detalje koje unosi prodavač, a aplikacija sama vuče podatke o vremenskoj prognozi za sve događaje unutar 15 dana. U slučaju koncerta također se prikazuju dodatne informacije s web servisa za izvođače. Detalji uključuju: naziv, vrstu, mjesto i datum događaja, a dodatno mogu imati broj sjedala i vrstu ulaznice (VIP, obična, stajala...). Oglasi se mogu jednostavno pretraživati pomoću filtriranja. Korisnici lako pronalaze sve što ih zanima, bio to neki specifičan događaj ili generalni tip poput sporta, koncerta, kazališne predstave itd. Ne moraju sve sami filtrirati već imaju mogućnost "lajkanja" tipa karata koje ih zanimaju, a mogu sakriti oglase ukoliko im se ne sviđaju. Ako se objavi oglas koji bi mogao zanimati korisnika ovisno o njegovim "lajkovima", on dobiva obavijest od stranice. Sustav im olakšava zamjenu traženjem lanca zamjene koji im dopušta doći od karte koju imaju do karte koju žele u samo jednom kliku. Osim običnih oglasa, postoje oglasi aukcije. Funkcioniraju na principu „tihe“ aukcije. Korisnik objavi oglas s minimalnom cijenom i više ga ne može povući, a karta ide osobi koja je ponudila najveću svotu novca.

Naravno, svaka zamjena ili prodaja karata bez odgovarajuće zaštite može rezultirati sporovima ili mogućim prevarama. Radi sprječavanja takvih slučajeva, aplikacija ima administratore koji dobivaju prijave sporova ili korisničkih računa, nakon čega mogu reagirati. Oni mogu gledati aktivnosti korisnika tako unaprijed uočavajući sumnjive račune i terminirajući ih pravovremeno, također mogu provjeriti je li neki korisnik pouzdan kada se zaprimaju lažne prijave.

# Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvatanja
F-01	Sustav omogućuje korisnicima login pomoću OAuth	Visok	Zahtjev dionika	Korisnik se može ulogirati Google računom
F-02	Sustav omogućuje prikaz dostupnih oglasa za zamjenu/prodaju karata	Visok	Zahtjev dionika	Korisnik može gledati dostupne karte na web stranici
F-02.1	Sustav omogućuje filtriranje oglasa	Srednji	Zahtjev dionika	Korisnik može filtrirati oglase po vrsti događaja, datumu, izvođačima, žanru...
F-02.2	Sustav omogućuje prikaz vremenske prognoze koju dohvaća s vanjskog servisa za prognozu	Srednji	Zahtjev dionika	Korisnik pri otvaranju oglasa vidi vremensku prognozu, koja se dohvaća s vanjskog servisa, za taj datum
F-03	Korisnici mogu stvoriti profil na stranici za objavu i javljanje na oglase	Visok	Zahtjev dionika	Korisnik se registrira na stranicu pomoću Google-a i upisuje tražene informacije za svoj profil
F-03.1	Sustav omogućuje korisniku uređivanje vlastitog profila	Visok	Zahtjev dionika	Korisnik može urediti svoje korisničko ime ili kontakt podatke
F-04	Sustav omogućuje objavu oglasa za prodaju/zamjenu karte	Visok	Zahtjev dionika	Korisnik može objaviti oglas na kojem mijenja ili prodaje svoju kartu
F-04.1	Sustav dohvaća podatke o glazbeniku s javnog web servisa za izvođače	Srednji	Zahtjev dionika	Sustav dohvaća podatke o izvođaču s javnog web servisa za glazbene koncerte
F-04.2	Sustav notificira korisnike ako su izrazili preferencu za neku vrstu događaja	Visok	Zahtjev dionika	Tražitelj dobije obavijest o novom oglasu unutar kategorija koje preferira
F-05	Sustav omogućuje korisniku javljanje na oglas	Visok	Zahtjev dionika	Korisnik se može javiti na oglas koji ga zanima i dogovoriti zamjenu
F-06	Sustav omogućuje brisanje, uređivanje i pregledavanje svojih oglasa	Srednji	Zahtjev dionika	Korisnik može gledati, urediti i brisati svoje oglase (staviti ih u košaricu)
F-06.1	Sustav omogućuje vraćanje svojih oglasa iz košarice	Srednji	Zahtjev dionika	Korisnik može vratiti neki oglas iz košarice ukoliko se predomislio
F-07	Sustav omogućuje stavljanje karte na aukciju	Srednji	Zahtjev dionika	Korisnik može staviti kartu na aukciju nakon čega ne može maknuti oglas
F-08	Sustav omogućuje uklanjanje oglasa	Srednji	Zahtjev dionika	Administrator može ukloniti neki oglas ako ima valjan razlog
F-09	Sustav omogućuje upravljanje korisničkim računima	Srednji	Zahtjev dionika	Administrator može upravljati korisničkim računima, može ih brisati, pregledavati povijest transakcija i gledati sporove vezane uz račun
F-10	Sustav omogućuje rješavanje sporova između korisnika	Srednji	Zahtjev dionika	Administrator može rješavati sporove između korisnika u slučaju nepoštovanja uvjeta zamjene

## Ostali zahtjevi

### Zahtjevi za održavanje

ID zahtjeva	Opis	Prioritet
M-1	Sustav treba biti oblikovan tako da omogućuje jednostavno održavanje. Commitovi trebaju imati objašnjenje i kod treba imati adekvatne komentare.	Srednji
	Sustav treba imati dokumentirane funkcionalne i nefunkcionalne zahtjeve, arhitekturu pomoću dijagrama	



M-1.1	(dijagrami obrazaca uporabe, sekvencijski dijagrami, dijagrami razreda, komponenti, aktivnosti i razmještaja).	Visok
ID	Opis	Prioritet
zahtjeva	Sustav treba biti popraćen "Wikijem" koji sadrži svu dokumentaciju vezanu za sustav i način uporabe.	Visok
M-1.1.1		

## Nefunkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet
NF-1	Sustav treba koristiti vanjski servis za dohvat prognoze	Visok
NF-2	Sustav treba koristiti vanjski servis za dohvat podataka o glazbeniku	Visok
NF-3	Sustav treba koristiti OAuth	Visok
NF-4	Sustav treba imati responsive design za različite ekrane	Visok
NF-5	Sustav treba biti dostupan 90% vremena	Srednji
NF-6	Sustav treba proći 90% automatiziranih testiranja	Srednji
NF-7	Sustav treba raditi na većim web preglednicima (Chrome, Mozilla, Edge)	Srednji

## Dionici

- Korisnici
- Administratori sustava
- Naručitelji
- Razvojni tim
- Sustavi za prodaju karata

Aktori i njihovi funkcionalni zahtjevi:

### A-1 Korisnik (inicijator) može:

- Ulogirati se F-01
- Vidjeti dostupne oglase F-02
- Filtrirati oglase F-02.1
- Vidjeti vremensku prognozu za događaj F-02.2
- Stvoriti profil F-03
- Urediti profil F-03.1

### A-2 Tražitelj (inicijator) može:

- Javiti se na oglas F-05

### A-3 Tražitelj (sudionik) može:

- Primiti obavijest za relevantne oglase F-04.2

### A-4 Vlasnik (inicijator) može:

- Postaviti oglas F-04
- Dobaviti informacije o izvođaču pri objavi F-04.1
- Brisati, urediti i pregledati svoje oglase F-06
- Vratiti obrisane oglase F-06.1
- Staviti kartu na aukciju F-07

### A-5 Vlasnik (sudionik) može:

- Prihvatiti ili odbiti ponudu F-05

### A-6 Weather servis (sudionik) može:

- Dati vremensku prognozu za događaj F-02.2

### A-7 Servis za login (sudionik) može:

- Dopustiti korisniku da se ulogira F-01

## A-8 Servis za izvođače (sudionik) može:

- Dati detalje o glazbeniku F-04.1

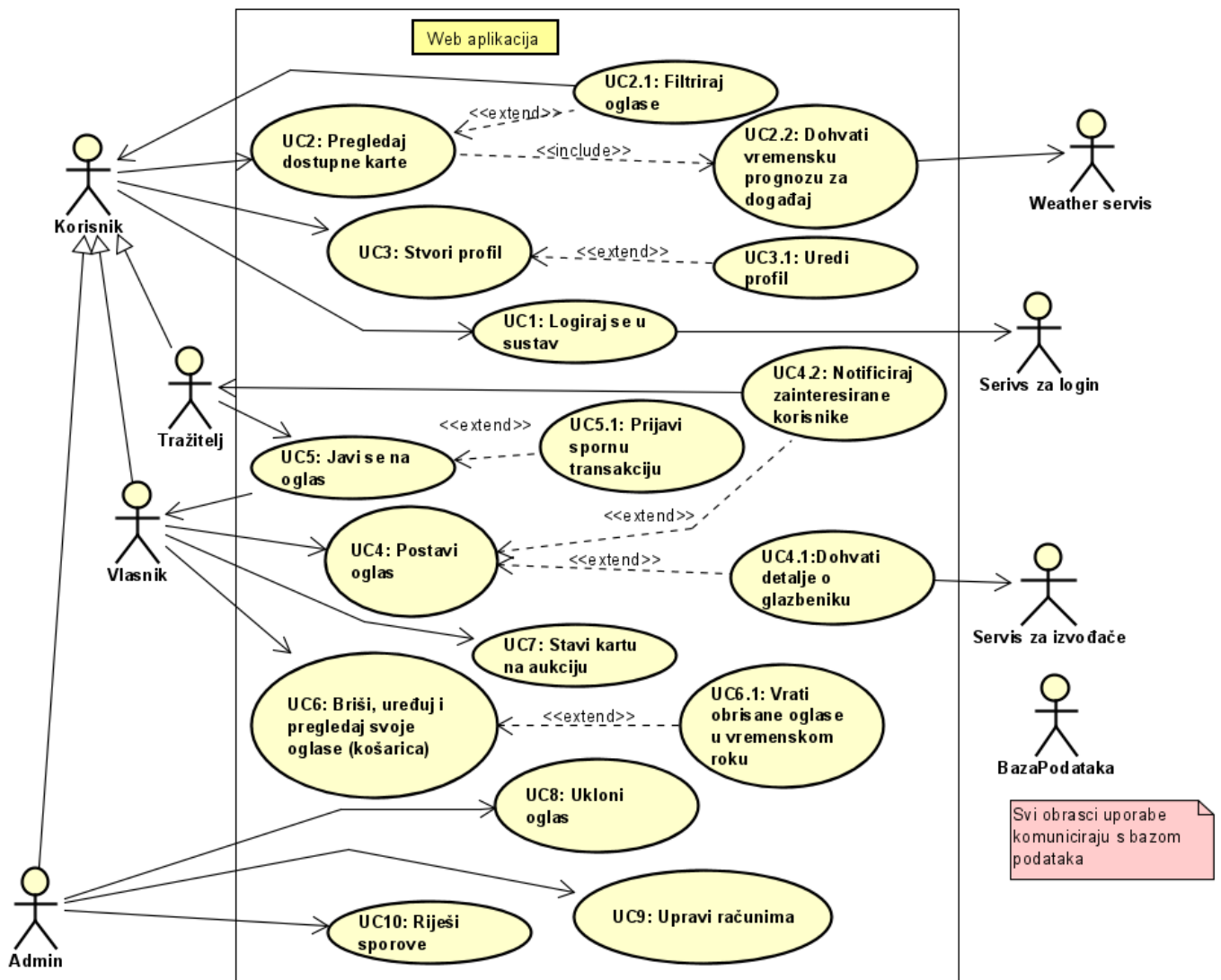
## A-9 Baza podataka (sudionik)

- Sudjeluje u svakom obrascu upotrebe spremanjem ili dohvatom relevantnih podataka iz sustava

## A-10 Administrator (sudionik) može:

- Ukloniti oglas F-08
- Upravlјati računima korisnika F-09
- Rješavati korisničke sporove F-10

# Dijagram obrazaca uporabe



# Obrasci uporabe

## UC1 - Logiraj se u sustav

- Glavni sudionik: Korisnik
- Cilj: Povezati korisnika na njegov profil
- Sudionici: Baza Podataka, OAuth servis
- Preduvjet: Korisnik postoji u bazi podataka (registrirao se)
- Opis osnovnog tijeka:
  1. Korisnik pritisne tipku prijavi se
  2. Prikazuje mu se login stranica gdje bira željeni servis kojim se želi logirati
  3. Upisuje svoje podatke u na stranici odabranog servisa

4. Korisniku se otvara stranica njegovog profila

- **Opis mogućih odstupanja:**
    2. a) OAuth servis nije dostupan
      - Korisnik dobiva obavijest da login nije moguć odabranim servisom i vraća se na početnu stranicu, proces završava u koraku 2
    3. a) Korisnik upiše krive podatke
      - Korisnika se vraća na ponovni upis podataka
      - Upiše točne podatke i proces se nastavlja na 3. koraku osnovnog tijeka
      - Korisnik stisne tipku da odustane od logina i proces završava
- 

## UC2 - Pregledaj dostupne karte

- **Glavni sudionik:** Korisnik
  - **Cilj:** Pregled dostupnih oglasa na stranici
  - **Sudionici:** Baza podataka
  - **Preduvjet:** -
  - **Opis osnovnog tijeka:**
    1. Korisnik lista oglase na početnoj stranici
    2. Korisnik pritiskom gumba odlazi na stranicu za pretraživanje gdje može dodatno pretražiti oglase koristeći filter (F-02.1)
    3. Korisnik ulazi u oglas koji ga zanima
  - **Opis mogućih odstupanja:**
    3. a) Vremenska prognoza (F-02.2)
      - Ako web servis nema prognozu za taj događaj, za prognozu piše: prognoza nije dostupna
      - Proces nastavlja normalno
    - b) Oglas je uklonjen
      - Korisnik dobiva obavijest da oglasa nema
      - Vraća se na početnu stranicu, proces završava
- 

### UC2.1 - Filtriraj oglase

- **Glavni sudionik:** Korisnik
  - **Cilj:** Suziti prikaz oglasa po željenom kriteriju
  - **Sudionici:** Baza podataka
  - **Preduvjet:** -
  - **Opis osnovnog tijeka:**
    1. Korisnik pretražuje oglase u search baru
    2. Stisne tipku za filtriranje i upiše kriterije
    3. Prikazu mu se oglasi koji odgovaraju filtru (ako postoje)
- 

### UC2.2 - Dohvati vremensku prognozu za događaj

- **Glavni sudionik:** Korisnik
  - **Cilj:** Prikaz vremenske prognoze za događaj
  - **Sudionici:** Baza podataka, Weather servis
  - **Preduvjet:** Korisnik izvršio UC2 pregledaj kartu, datum događaja je unutar 15 dana od današnjeg datuma
  - **Opis osnovnog tijeka:**
    1. Korisnik ulazi u oglas koji ga zanima
    2. Sustav dohvaća prognozu s weather servisa
    3. Prikazuje korisniku prognozu na lokaciji događaja
  - **Opis mogućih odstupanja:**
    2. a) Ne postoji prognoza za lokaciju
      - Korisniku piše vremenska prognoza nije dostupna
      - Proces nastavlja u koraku 3
- 

## UC3 - Stvori profil

- **Glavni sudionik:** Korisnik

- **Cilj:** Stvoriti profil za korištenje više funkcionalnosti aplikacije
  - **Sudionici:** Baza podataka
  - **Preduvjet:** Korisnik nema račun
  - **Opis osnovnog tijeka:**
    1. Korisnik obavlja UC1 Logiraj se u sustav, ali nema svoj račun
    2. Sustav ga šalje na obrazac za registraciju
    3. Korisnik upisuje tražene podatke
    4. Korisnik je preusmjeren na svoj profil
  - **Opis mogućih odstupanja:**
    3. a) Korisnik unosi nevaljane podatke
      - Dobiva upozorenje i mora ponovno unositi podatke
      - Proces se nastavlja u koraku 3
- 

### UC3.1 - Uredi profil

- **Glavni sudionik:** Korisnik
  - **Cilj:** Korisnik može urediti svoj profil
  - **Sudionici:** Baza podataka
  - **Preduvjet:** Korisnik izvršio UC3 Stvori profil i ulogiran je
  - **Opis osnovnog tijeka:**
    1. Korisnik ulazi u svoj profil i odabire opciju "Uredi profil"
    2. Dobije obrazac na kojem mijenja podatke koje želi
    3. Korisnik pritisne gumb da potvrdi promjene ili odustane
  - **Opis mogućih odstupanja:**
    2. a) Korisnik unosi nevaljane podatke
      - Stranica daje upozorenje i traži popravak unosa
      - Proces se nastavlja u koraku 3
- 

### UC4 - Postavi oglas

- **Glavni sudionik:** Vlasnik
  - **Cilj:** Dozvoliti objavu karte koju vlasnik želi mijenjati/prodati
  - **Sudionici:** Baza podataka, Servis za izvođače
  - **Preduvjet:** -
  - **Opis osnovnog tijeka:**
    1. Vlasnik odabire opciju objavi oglas
    2. Dobiva obrazac u koji unosi podatke o karti i događaju
    3. Odabire vrstu oglasa
    4. Objavljuje oglas
  - **Opis mogućih odstupanja:**
    2. a) Vlasnik nije unio obavezne podatke o karti
      - Sustav daje upozorenje i traži unos obaveznih podataka
      - Proces se nastavlja u koraku 2
    3. b) Vrsta događaja je koncert
      - Sustav vuče podatke o glazbeniku sa web servisa (F-04.1)
    4. a) Vlasnik odabrao zamjenu
      - Proces se nastavlja u koraku 2 s različitim obrascem
    - b) Vlasnik odabrao aukciju -> F-07
      - Proces se završava i počinje izvođenje obrasca UC7 stavi kartu na aukciju
- 

### UC4.1 - Dohvati detalje o glazbeniku

- **Glavni sudionik:** Sustav
- **Cilj:** Detaljniji detalji o glazbeniku
- **Sudionici:** Baza podataka, Servis za glazbenike



- **Preduvjet:** Korisnik izvršio UC5 postavi oglas i vrsta oglasa je koncert
  - **Opis osnovnog tijeka:**
    1. Korisnik objavljuje oglas tipa koncert
    2. Sustav dohvaća podatke o glazbeniku sa servisa
    3. Sustav prikazuje podatke o glazbeniku pri pregledu oglasa
  - **Opis mogućih odstupanja:**
    2. a) Ne postoje podaci o glazbeniku
      - Sustav ne prikazuje podatke
      - Proces se završava
- 

## UC4.2 - Notificiraj zainteresirane korisnike

- **Glavni sudionik:** Sustav
  - **Cilj:** Obavijestiti korisnike o oglasu koji im je zanimljiv
  - **Sudionici:** Baza podataka
  - **Preduvjet:** Korisnik izvršio UC4 postavi oglas koji je u preferencama drugog korisnika
  - **Opis osnovnog tijeka:**
    1. Korisnik na obrascu registracije u UC3 označio vrstu oglasa kao preferiranu
    2. Korisniku se prikazuje prozorčić: Dostupni novi oglasi!
- 

## UC5 - Javi se na oglas

- **Glavni sudionik:** Tražitelj
  - **Cilj:** Izražavanje interesa za neku kartu
  - **Sudionici:** Baza podataka, Vlasnik
  - **Preduvjet:** -
  - **Opis osnovnog tijeka:**
    1. Tražitelj odabire oglas na početnoj stranici ili nakon pretraživanja
    2. Odabire gumb javi se na oglas
    3. Vlasnik dobiva obavijest o ponudi
  - **Opis mogućih odstupanja:**
    2. a) Oglas je aukcija
      - Tražitelj mora unijeti iznos veći ili jednak minimalnom za javiti se
      - Proces se nastavlja u koraku 3
    - b) Vlasnik obrisao oglas
      - Tražitelj dobiva obavijest da je oglas izbrisan
      - Proces završava i vraća korisnika na početnu stranicu
    3. a) Vlasnik mijenjao oglas
      - Korisnik dobije upozorenje da je oglas promijenjen
      - Proces završava i vraća korisnika na početnu stranicu
- 

## UC5.1 - Prijavi spornu transakciju

- **Glavni sudionik:** Tražitelj
  - **Cilj:** Omogućiti prijavu spornih transakcija
  - **Sudionici:** Baza podataka
  - **Preduvjet:** Korisnik izvršio UC5: javi se na oglas, vlasnik prihvatio zamjenu/prodaju
  - **Opis osnovnog tijeka:**
    1. Tražitelj na profilu odabire kupljene oglase i stisne relevantni oglas
    2. Tražitelj pritisne gumb prijavi korisnika
    3. Sustav zapisuje prijavu i prikazuje ju profilima administratora
- 

## UC6 - Briši, uređuj i pregledaj svoje oglase (košarica)

- **Glavni sudionik:** Vlasnik

- **Cilj:** Jednostavno upravljanje svojim oglasima
  - **Sudionici:** Baza podataka
  - **Preduvjet:** Vlasnik izvršio UC4 postavi oglas
  - **Opis osnovnog tijeka:**
    1. Vlasnik ulazi u svoj profil i stisne tipku moji oglasi
    2. Dobije prikaz svojih karata
    3. Vlasnik odabire opciju uredi
    4. Vlasnik odabire opciju Obriši
  - **Opis mogućih odstupanja:**
    3. Uređivanje
      - Vlasniku se prikaže obrazac na kojem mijenja cijenu ili željenu kartu ovisno o vrsti oglasa
      - Potvrđuje promjenu stiskom gumba
      - Sustav ažurira oglas, proces završava
    4. Brisanje
      - Pritiskom na gumb obriši oglas se prebaci u košaricu
      - Sustav ne prikazuje oglas više u pretraživanju i početnoj stranici
      - Vlasnik pritiskom na obrisani oglasi i odabirom oglasa može trajno obrisati oglas ako se to nije automatski dogodilo (istekao vremenski period), proces završava
- 

## UC6.1 - Vрати obrisane oglase u vremenskom roku

- **Glavni sudionik:** Vlasnik
  - **Cilj:** Oglas se može vratiti ako se vlasnik predomisli
  - **Sudionici:** Baza podataka
  - **Preduvjet:** Vlasnik izvršio UC6 obriši svoj oglas
  - **Opis osnovnog tijeka:**
    1. Vlasnik ulazi u svoj profil
    2. Vlasnik odabire gumb za prikaz obrisanih oglasa
    3. Odabire oglas koji želi vratiti i stisne relevantni gumb da potvrdi
- 

## UC7 - Stavi kartu na aukciju

- **Glavni sudionik:** Vlasnik
  - **Cilj:** Karta se stavlja na aukciju ne bi li se dobila veća cijena
  - **Sudionici:** Baza podataka
  - **Preduvjet:** Vlasnik odabrao opciju aukcija pri postavi oglasa (F-04)
  - **Opis osnovnog tijeka:**
    1. Vlasnik postavlja minimalnu cijenu i trajanje aukcije
    2. Dobiva upozorenje da se ova objava ne može urediti ili obrisati
    3. Vlasnik potvrđuje i stavlja kartu na aukciju
  - **Opis mogućih odstupanja:**
    1. Kraj aukcije je isti datum kao i događaj
      - Iskače upozorenje: Vlasnik mora unijeti trajanje tako da aukcija završi najmanje 24h prije datuma događaja
      - Proces nastavlja u koraku 1
    2. Izabire opciju odustani
      - Vraća se na postavi oglas F-04
      - Proces završava
- 

## UC8 - Ukloni oglas

- **Glavni sudionik:** Administrator
- **Cilj:** Uklanjanje sumnjivih ili nedozvoljenih oglasa
- **Sudionici:** Baza podataka, Vlasnik
- **Preduvjet:** -
- **Opis osnovnog tijeka:**

1. Administrator ulazi u sumnjivi oglas
  2. Odabire opciju obriši i upisuje razlog za brisanje
  3. Sustav prestaje prikazivati oglas
  4. Sustav šalje obavijest vlasniku da mu je oglas uklonjen
- 

## UC9 - Upravi računima

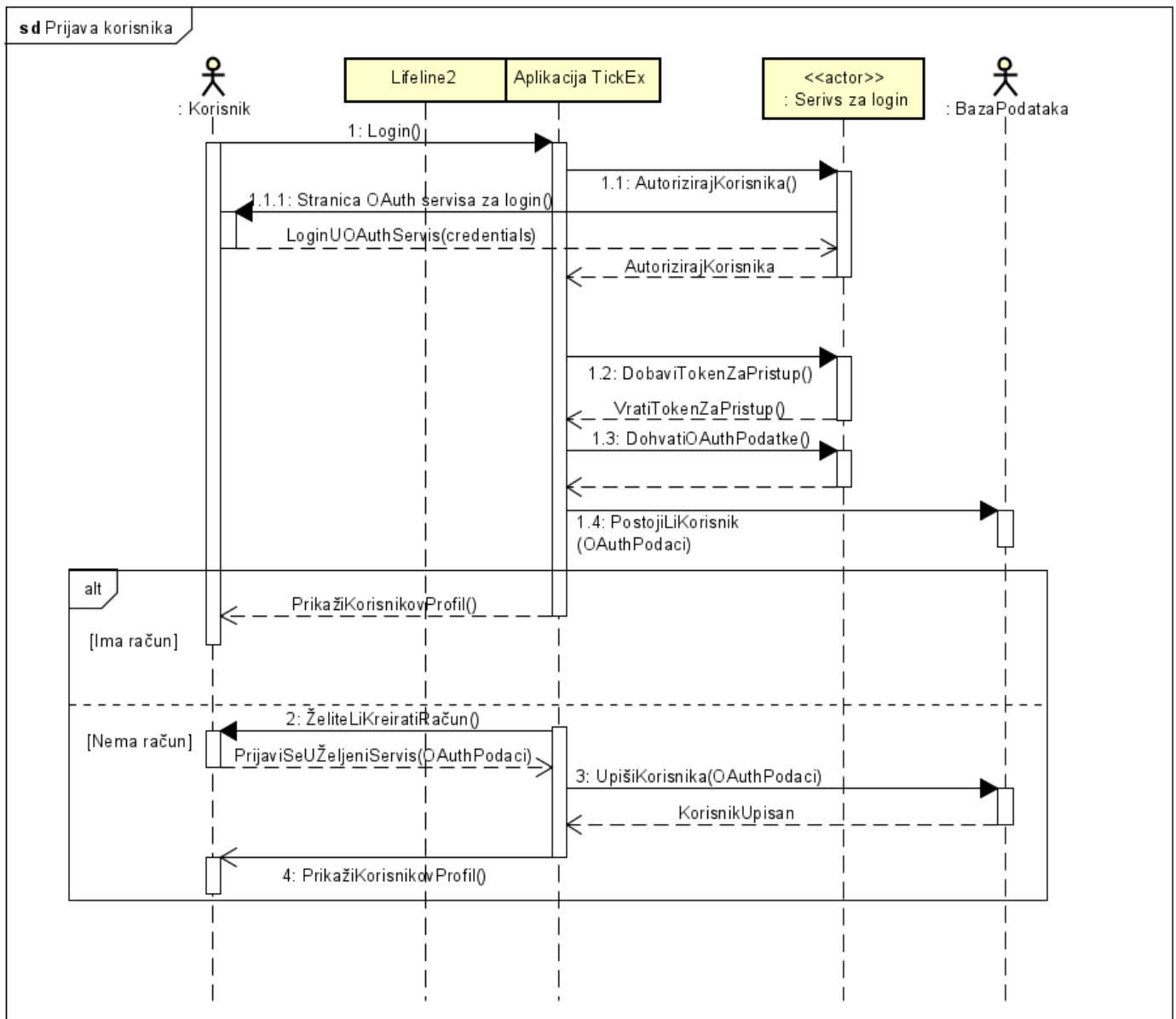
- **Glavni sudionik:** Administrator
  - **Cilj:** Jednostavan pregled računa i upravljanje istima
  - **Sudionici:** Baza podataka
  - **Preduvjet:** -
  - **Opis osnovnog tijeka:**
    1. Administrator ulazi u listu računa
    2. Odabire račun koji želi urediti
    3. Prikazuje mu se obrazac za uređivanje računa na kojem upiše podatke
  - **Opis mogućih odstupanja:**
    3. a) Brisanje računa
      - Administrator stisne gumb ukloni račun
      - Sustav briše račun korisnika i sve njegove oglase
      - Proces završava
    - b) Mijenjanje podataka
      - Administrator mijenja podatke profila i potvrđuje promjenu
      - Proces završava
    - c) Pregled transakcija
      - Administrator stiskom na gumb dobiva pregled svih transakcija računa
      - Proces završava
- 

## UC10 - Riješi sporove

- **Glavni sudionik:** Administrator
- **Cilj:** spornih transakcija
- **Sudionici:** Baza podataka
- **Preduvjet:** Postoje sporne transakcije
- **Opis osnovnog tijeka:**
  1. Administrator ulazi u listu spornih transakcija
  2. Odabire transakciju da vidi detalje
  3. Podnosi primjerenu akciju
  4. Sustav obavještava sudionike o rješenju

## Sekvencijski dijagrami

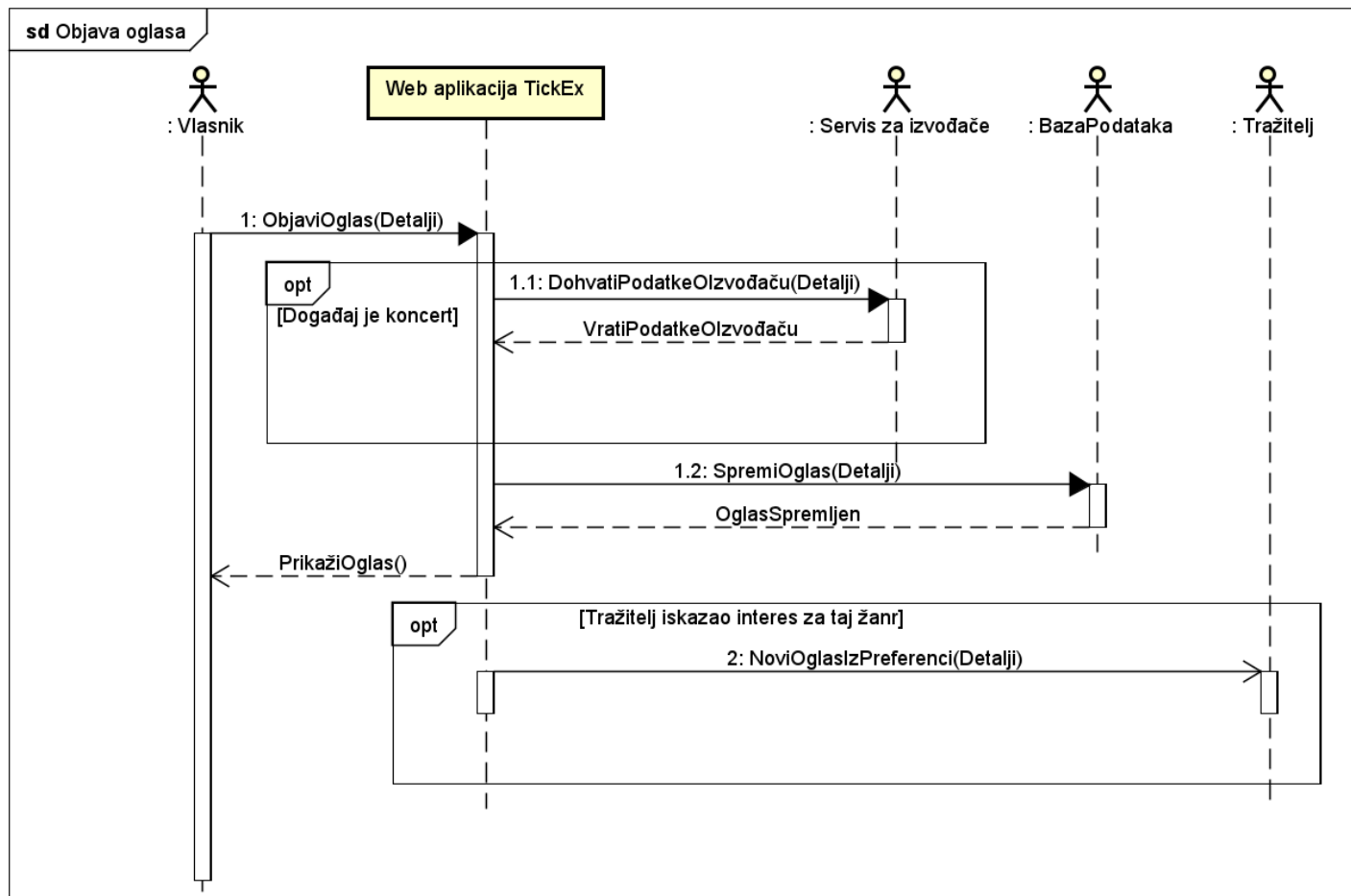
---



1. Korisnik ulazi u login i bira željenu stranicu za ulogirati se
  - 1.1 Sustav traži autorizaciju od te stranice
  - 1.1.1 Korisnik ulazi u login obrazac za tu stranicu gdje upisuje svoje podatke
  - 1.2 Sustav dobavlja token od stranice koja autorizira korisnika
  - 1.3 Sustav dohvaća podatke o korisniku pomoću tokena
  - 1.4 Sustav provjeri postoji li korisnik (treba li ga registrirati)
- Ako korisnik ima račun, sustav mu prikazuje njegov profil

Inače

2. Sustav šalje korisniku obrazac za registrirati račun kojeg on popuni
3. Sustav upisuje podatke o korisniku u bazu podataka
4. Sustav prikazuje korisniku njegov profil



1. Vlasnik karte objavljuje oglas s detaljima o ulaznici

Ako je tip događaja koncert

1.1 Sustav dohvaća podatke o izvođaču s vanjskog servisa za glazbenike

Inače

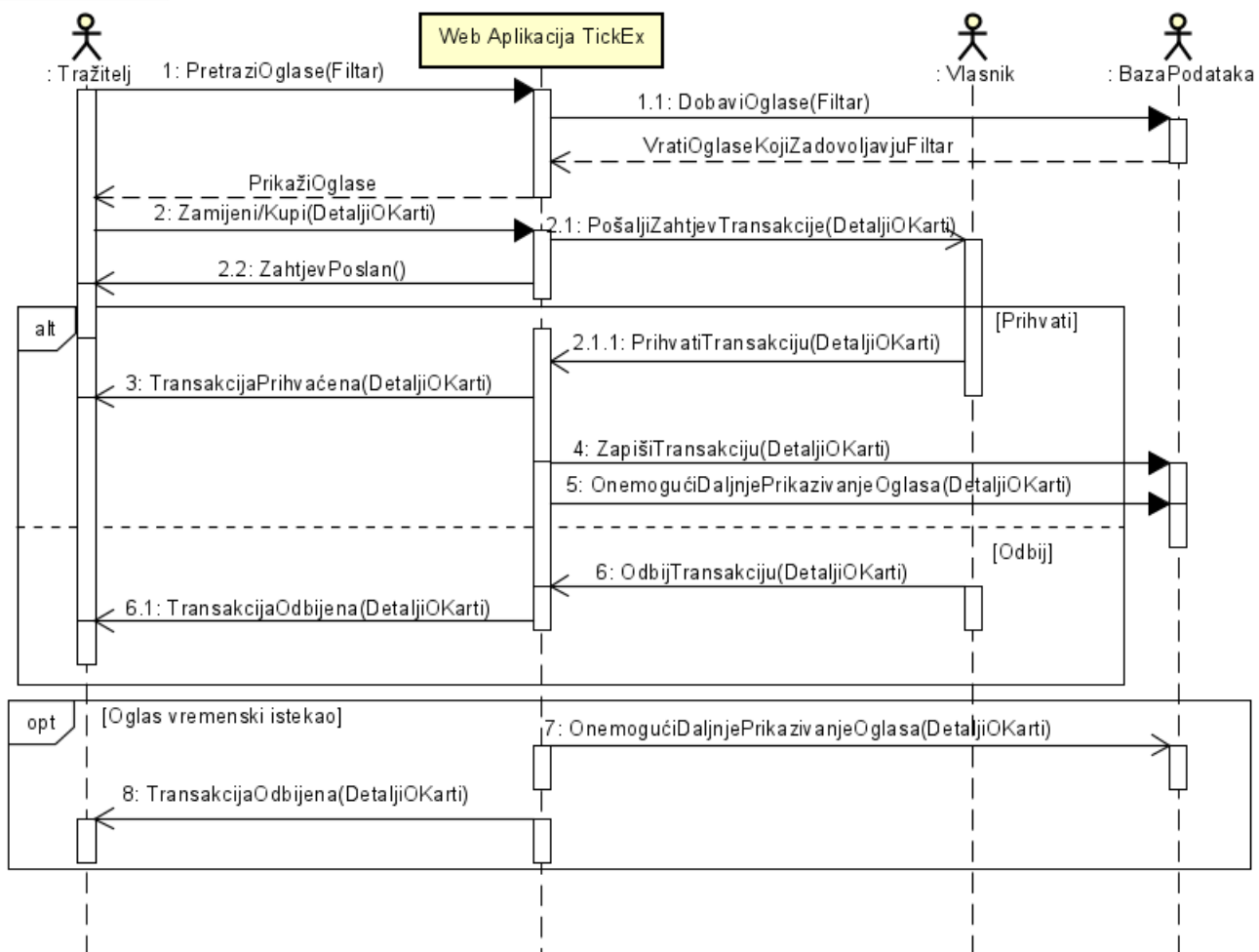
1.2 Sustav sprema oglas u bazu podataka (Sad ga mogu vidjeti drugi korisnici)

Ako neki korisnik u svojim preferencama ima isti tip oglasa kao ovaj objavljen

2. Tražitelj dobije obavijest o novom oglasu koji bi ga mogao interesirati

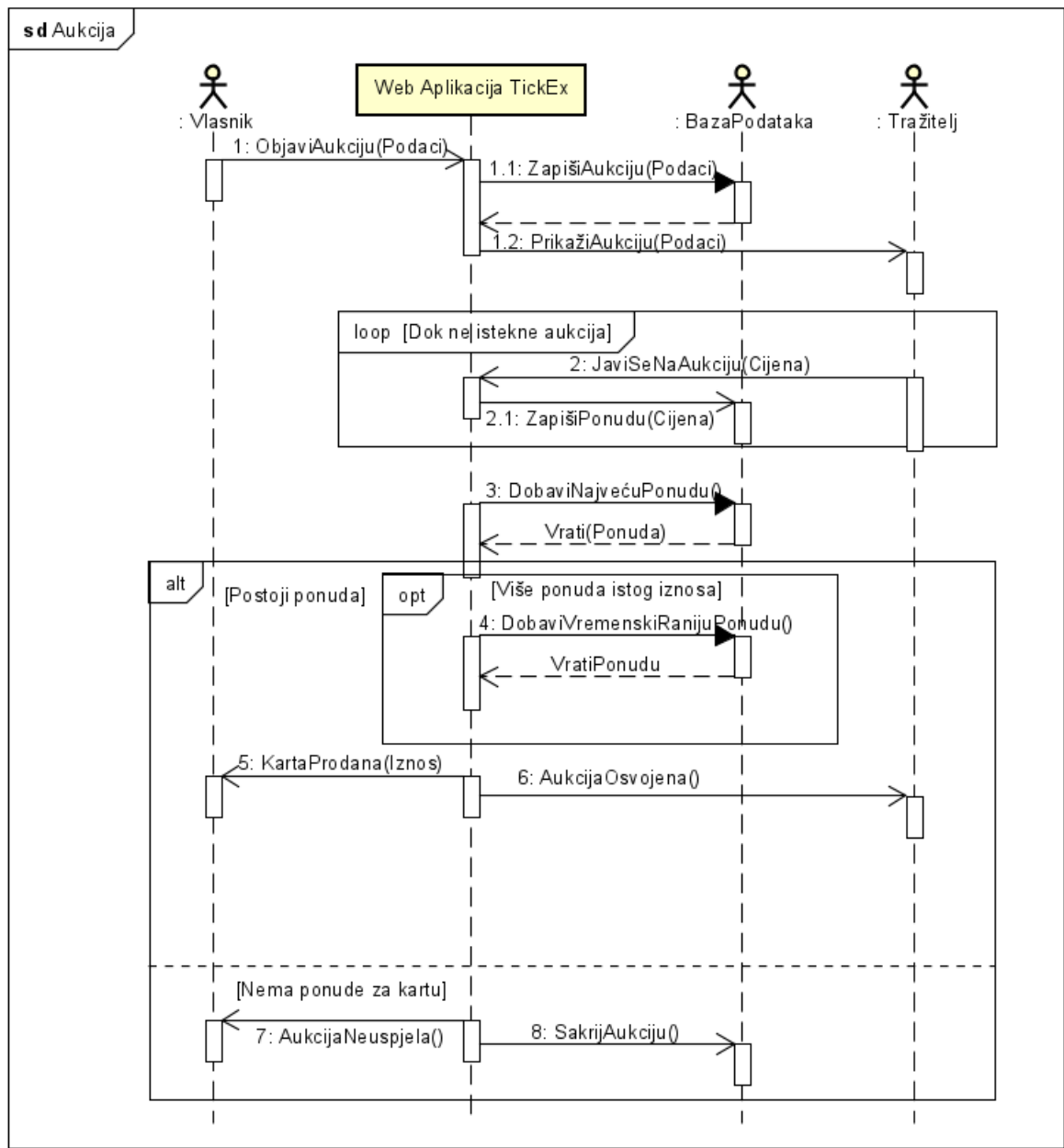


# sd Zamjena karte



1. Tražitelj pretražuje oglase s filtrom ili bez
  - 1.1 Sustav dobavlja oglase iz baze podataka i prikazuje ih korisniku
2. Tražitelj odabire kartu koju želi kupiti/mijenjati se za nju
  - 2.1 Sustav šalje vlasniku zahtjev za zamjenu
  - 2.2 Sustav šalje tražitelju obavijest o poslanom zahtjevu
    - 2.1.1 Vlasnik prihvaća transakciju
3. Tražitelj dobije obavijest o prihvatu i može dogovarati predaju s vlasnikom
4. Sustav evidentira obavljenju transakciju
5. Sustav sakrije oglas od ostalih korisnika
6. Vlasnik odbija transakciju
  - 6.1 Sustav šalje tražitelju obavijest o odbijenom zahtjevu
 

Ako nema odgovora vlasnika i karta vremenski isteče (dogadaj prošao)
7. Sustav onemogućiti daljnji prikaz oglasa
8. Sustav šalje tražitelju obavijest o neuspješnoj transakciji



1. Korisnik objavljuje aukciju 1.1 Sustav zapisuje aukciju u bazu  
1.2 Aukcija se prikazuje ostalim korisnicima
2. Tražitelji šalju ponude dok aukcija traje  
2.1 Sustav zapisuje ponude
3. Sustav dobavlja najveći ponuđeni iznos
4. Ako nađe više iznosa gleda koji je korisnik prije napravio ponudu
5. Vlasniku šalje obavijest o prodanoj karti i najvećoj ponudi
6. Tražitelj dobiva obavijest da je osvojio aukciju
7. Ako nema niti jedna ponuda, vlasnik dobiva obavijest da je aukcija neuspjela

## Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

I za prikazane obrasce uporabe tablično navedite koje funkcionalne zahtjeve obuhvaćaju

UC dijagram	Funkcionalnosti
UC1	F-01
UC2	F-02
UC2.1	F-02.1
UC2.2	F-02.2

UC3 UC dijagram	F-03 Funkcionalnosti
UC3.1	F-03.1
UC4	F-04
UC4.1	F-04.1
UC4.2	F-04.2
UC5	F-05
UC6	F-06
UC6.1	F-06.1
UC7	F-07
UC8	F-08
UC9	F-09
UC10, UC5.1	F-10

# Arhitektura sustava

## Opis arhitekture

### Stil arhitekture

Sustav je napravljen po principu slojevite arhitektura s podjelom: presentation layer (frontend), business logic layer (backend), data access layer (baza podataka). Ovaj pristup temelji se na podjeli sustava na različite slojeve, od kojih svaki ima svoju specifičnu odgovornost i interakciju s drugim slojevima. Ovaj model je odabran zbog odvajanja brige (separation of concerns), bolje održivosti, skalabilnosti, fleksibilnosti te principa rada "podjeli pa vladaj". Kako je ovo naš susret sa ovakvim projektom i alatima koje koristimo, zaključili smo da bi najbolji način rada bio da se tim podjeli po slojevima (frontend, backend i baza) i da svatko temeljno nauči svoj dio umjesto da cijeli tim djelomično nauči sve.

### Podsustavi:

- Usluga autentifikacije i autorizacije: Upravlja prijavom korisnika, autentifikacijom i autorizacijom pomoću usluge OAuth2 koja koristi JSON Web Token (JWT) za provjeru korisničkih zahtjeva.
- Usluga upravljanja korisnicima: Upravlja kreiranjem, ažuriranjem i brisanjem korisničkih profila.
- Usluga za stvaranje i brisanje oglasa: Omogućuje korisnicima dodavanje, uređivanje i brisanje oglasa.
- Usluga za pretraživanje: Omogućuje pretraživanje oglasa

### Preslikavanje na radnu platformu

Za implementaciju arhitekture odabran je cloud pristup putem Render platforme. Ovaj pristup omogućuje jednostavnu implementaciju, skalabilnost i podršku za moderne tehnologije. Render je izabran zbog svojih besplatnih planova za web servise i baze podataka, što omogućuje besplatno testiranje i razvoj aplikacija bez dodatnih troškova.

Proces gradnje i postavljanja aplikacije na Render platformu potpuno je automatiziran korištenjem GitHub Actions. Svaka promjena u kodu automatski se testira, gradi i postavlja u produkcijsko okruženje, čime se smanjuje potreba za ručnim intervencijama. Pipeline skripte, smještene u .github/workflows direktoriju, definiraju korake za build i deploy backend i frontend aplikacije.

Za obavještanje Render platforme o dostupnosti nove verzije aplikacije koriste se webhookovi. Kada pipeline završi uspješno, GitHub Actions automatski aktivira webhook, pokrećući redeploy nove verzije na Renderu.

Ovaj proces omogućuje pouzdano, brzo i efikasno ažuriranje aplikacije u produkciji, uz minimalan ručni rad. Korištenjem kombinacije GitHub Actions i Render webhookova, svaka nova verzija aplikacije može biti postavljena jednostavno i bez zastoja.

### Spremišta podataka

Korištena je PostgreSQL baza podataka jer je open source, besplatna, standardizirana i dovoljna za sve potrebe projekta.

### Mrežni protokoli

Za razgovor između frontenda i backenda se koristi protokol HTTP/HTTPS. Standardiziran je i služi kao osnovna za komunikaciju na webu. HTTP je stateless protokol što znači da nije potrebno pamtiti stanje za koristiti ga, a time se štedi na memoriji i brzini obrade zahtjeva. Svaki zahtjev između klijenta i poslužitelja je neovisan. HTTP podupire širok spektar tipova podataka koji se specificiraju s content-type zaglavljem. Podupire metode: GET, POST, PUT, DELETE koje se koriste u RESTful API-jima.

## Globalni upravljački tok

Korisnik započinje interakciju putem React frontenda, unosi podatke u obrazac, primjerice za prijavu ili kreiranje oglasa i pokreće zahtjev koji se šalje backendu putem REST API poziva. Frontend koristi HTTP/HTTPS protokol za slanje podataka, a u zahtjev može uključiti OAuth token za autentifikaciju. Kada zahtjev stigne u Spring Boot backend, Controller sloj ga prima i šalje Service sloju, koji ga obrađuje, a ako je potrebno, koristi Repository sloj za pristup podacima u PostgreSQL bazi putem JPA (Java Persistence API). Repository sloj preko JPA koristi JDBC za povezivanje s bazom i dohvaća ili pohranjuje podatke. Nakon obrade, backend vraća odgovor s potrebnim podacima (npr. popis oglasa ili potvrdu prijave) ili poruku o grešci. Frontend zatim prikazuje korisniku povratnu informaciju ili potrebne podatke te mu omogućuje daljnje interakcije. Ciklus se ponavlja, osiguravajući korisniku pravovremeni odgovor na sve zahtjeve, dok slojevita arhitektura organizira ovaj protok podataka.

## Sklopovskoprogramski zahtjevi

Za backend i bazu podataka treba Linux ili Windows Server. Potrebni su Java 11, Spring Boot za backend, Maven za upravljanje ovisnostima, PostgreSQL za bazu podataka te OAuth2 za autentifikaciju.

## Obrazloženje odabira arhitekture

### Izbor arhitekture temeljen na principima oblikovanja

- Odvajanje brige (Separation of concerns) - Različiti dijelovi sustava (komponente, moduli, slojevi) trebaju biti odgovorni za različite zadatke ili funkcionalnosti, čime se omogućava bolja organizacija koda, bolja održivost i lakše testiranje.
- Skalabilnost: Svaka komponente sustava (frontend, backend, baza podataka) može se skalirati neovisno, čime se optimiziraju resursi na temelju specifičnih potreba.
- Otpornost na greške: Razdvajanje odgovornosti između frontend-a, backend-a i baze podataka omogućuje sustavu da bude otporan na greške, jer se greške u jednom sloju ne šire na cijeli sustav.
- Fleksibilnost u razvoju: Različiti timovi mogu razvijati i održavati različite komponente sustava, omogućujući brži razvoj i bolju raspodjelu resursa.

### Razmatrane alternative

Nismo razmatrali alternative

## Organizacija sustava na visokoj razini

### Klijent-poslužitelj

Sustav koristi klijent-poslužitelj arhitekturu gdje frontend (React) šalje HTTP zahtjeve (request) backendu (Spring Boot), koji obrađuje podatke i komunicira s bazom podataka, pa onda vraća odgovor (response). Odvajanjem klijenta i poslužitelja dobivamo mogućnost centraliziranja podataka, procesiranja i skladištenja na poslužiteljsku stranu. Bude li potrebno skalirati projekt lako se dodaju dodatni serveri kako bi omogućili obradu većeg workloada. Održavanje servera se događa bez ometanja klijentske strane. Centralni sustav za obradu podataka dozvoljava jednostavnu provjeru integriteta i konzistentnosti podatka svakog klijenta. Ovakva arhitektura je fleksibilna bez obzira na klijentov uređaj. (mobitel, laptop, stolno računalo...) Osigurava konkurentnost više korisnika ukoliko su poslužitelji dovoljno snažni procesorski.

### Baza podataka

Sustav koristi PostgreSQL kao relacijsku bazu podataka za pohranu korisničkih podataka, oglasa i transakcija. Open source i besplatan je što je ključan faktor pri razvoju manjih aplikacija. Podržava visokorazinske tipove podataka poput JSON-a, arraya, XML-a... Najbitniji faktor je stabilnost i konzistentnost podataka koju PostgreSQL ostvaruje s Multi version concurrency control (MVCC) koji dozvoljava da bazu paralelno koristi više korisnika bez da njihove promjene utječu na ostale korisnike. Baza lako može skalirati ovisno o potrebama projekta.

### Datotečni sustav

Sustav pohranjuje sve podatke u PostgreSQL bazu

### Grafičko sučelje

Korisničko sučelje je web aplikacija u Reactu, koja komunicira s backendom putem REST API-ja, prikazujući podatke korisnicima.

## Organizacija aplikacije

### Frontend i Backend slojevi

Frontend (React) upravlja korisničkim sučeljem. Prikazuje korisniku stranicu, dopušta mu mijenjanje prikaza i slanje zahtjeve (npr. ulogiraj se). Frontend tada šalje zahtjeve backendu (Spring Boot) putem REST API-ja koji koristi CORS za omogućavanje sigurne komunikacije između frontend-a i backend-a. Backend je podijeljen u 3 dijela: controller, service i repository. Zahtjev dolazi do kontrolera koji ga tumači kako bi ga service sloj, u kojem je sva poslovna logika, obradio, ako je potrebno service traži od repository-ja da dohvati ili spremi podatke u bazu. Service daje odgovor controlleru koji ga šalje na frontend da bi se korisniku prikazala ažurirana stranica.

### MVC arhitektura

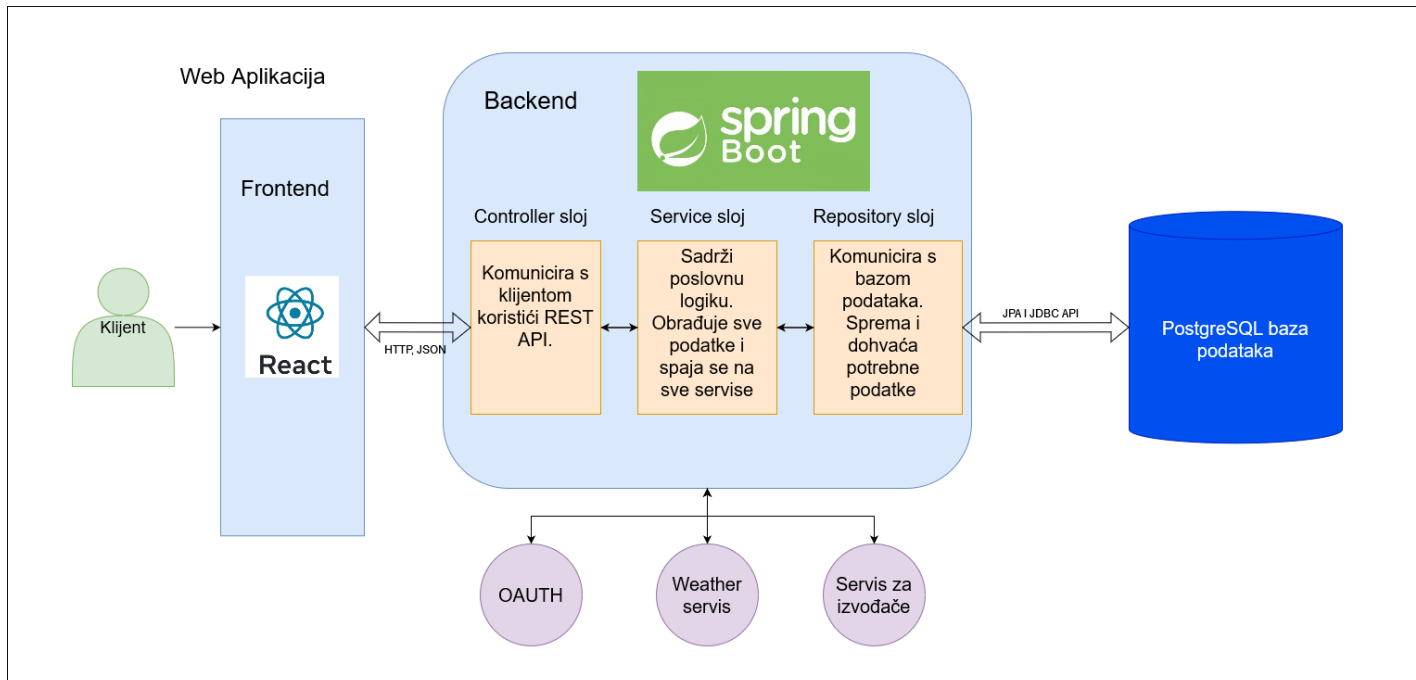
Aplikacija koristi MVC arhitekturu u kojoj:

- Model predstavlja podatke i logiku (npr. entiteti poput User i Ticket).

- View je korisničko sučelje u Reactu.
- Controller upravlja zahtjevima i povezivanjem s poslovnom logikom u Spring Bootu.

Na ovaj način dobivamo odvajanje poslovne logike (Model i controller), podataka (Model) i logike korisničkog sučelja (View). Prednost ovoga je paralelan rad nad komponentama sustava. Dok jedan dio tima programira view drugi dio može raditi na modelu bez ikakvih problema. Kasnije se ta dva dijela međusobno povezuju controllerom. Osim lakšeg paralelnog rada, dobivamo lako održiv kod koji se može ponovno koristiti (reusable). Svaki dio se da odvojeno testirati.

## Dijagram visoke razine



## Baza podataka

Za bazu podataka odabran je PostgreSQL, relacijska baza podataka poznata po svojoj pouzdanosti, fleksibilnosti i mogućnostima. Ova baza podataka omogućuje jednostavno upravljanje velikim količinama podataka i kompleksnim upitima, uz istovremeno osiguravanje visokih performansi. Također, PostgreSQL je open-source rješenje, što dodatno doprinosi smanjenju troškova razvoja i održavanja aplikacije.

## Opis tablica

### KORISNIK

Atribut	Tip podatka	Opis varijable
idKor	INTEGER	Primarni ključ, jedinstven ID korisnika
email	VARCHAR	Jedinstven email korisnika
imeKor	VARCHAR	Ime korisnika
prezKor	VARCHAR	Prezime korisnika
datumUla	DATE	Datum prvog ulaska u stranicu
statusKor	BOOLEAN	Označava je li korisnik suspendiran sa stranice
ocjena	DECIMAL	Prosječna ocjena korisnika
admin	BOOLEAN	Označava je li korisnik admin



VRSTA\_DOGADAJA

Atribut	Tip podatka	Opis varijable
idDog	INTEGER	Primarni ključ, jedinstven ID događaja
nazVrDog	VARCHAR	Naziv vrste događaja

OGLAS

Atribut	Tip podatka	Opis varijable
idOgl	INTEGER	Primarni ključ, jedinstven ID oglasa
nazDog	VARCHAR	Naziv događaja
nazIzv	VARCHAR	Naziv izvođača ako se radi o glazbenom događaju
mjesto	VARCHAR	Naziv mjesta događaja
datum	DATE	Datum održavanja događaja
brSje	INTEGER	Nullable, broj sjedala
vrsUla	VARCHAR	Nullable, vrsta ulaznice
status	VARCHAR	Status ulaznice – razmjena, prodaja, aukcija, obrisana ili istekla
vrijemeObrisano	TIMESTAMP	Ako je ulaznica obrisana, vrijeme kada je obrisana
idKor	INTEGER	Strani ključ, id korisnika koji je prodavač
idDog	INTEGER	Strani ključ, id događaja

PRODAJA

Atribut	Tip podatka	Opis varijable
cijena	INTEGER	Cijena koju je prodavač postavio za oglas
idOgl	INTEGER	Primarni, strani ključ id oglasa kojeg nasljeđuje
idKupac	INTEGER	Strani ključ, id korisnika koji prodaje ulaznicu

RAZMJENA

Atribut	Tip podatka	Opis varijable
zeljeniNazOgl	VARCHAR	Željeni naziv događaja
zeljenoMjesto	VARCHAR	Željeno mjesto održavanja događaja
zeljeniDatum	DATE	Željeni datum događaja
zeljeniBrSje	INTEGER	Nullable, željeni broj sjedala
zeljenaVrsUla	VARCHAR	Nullable, željena vrsta ulaznice
idOgl	INTEGER	Primarni, strani ključ id oglasa kojeg nasljeđuje

AUKCIJA

Atribut	Tip podatka	Opis varijable
pocCijena	INTEGER	Početna cijena stavljena za aukciju
trajanje	DATE	Kojeg datuma završava aukcija
idOgl	INTEGER	Primarni, strani ključ id oglasa kojeg nasljeđuje
konCijena	INTEGER	Cijena koja je dostignuta na aukciji
pobjednik	INTEGER	Strani ključ, označuje id korisnika koji je ponudio najviše na aukciji

sudjeluje

Atribut	Tip podatka	Opis varijable
idSudj	INTEGER	Primarni ključ, jedinstven ID sudjelovanja
idKor	INTEGER[]	Polje id korisnika koji sudjeluju u razmjeni
idOgl	INTEGER[]	Polje id oglasa koji sudjeluju u razmjeni
odgovor	BOOLEAN[]	Polje koje bilježi odgovore (pozitivne ili negativne)
vrijemeNastanka	TIMESTAMP	Trenutak nastanka lanca razmjene

nudi

Atribut	Tip podatka	Opis varijable
ponuda	INTEGER	Ponuda koju korisnik nudi za određenu aukciju
idKor	INTEGER	Primarni i strani ključ
idOgl	INTEGER	Primarni i strani ključ

prijavljuje

Atribut	Tip podatka	Opis varijable
prijavljujeldKor	INTEGER	Primarni i strani ključ
prijavljenIdKor	INTEGER	Primarni i strani ključ
razlog	VARCHAR	Razlog prijave
datumPri	DATE	Datum prijave

ocjenjuje

Atribut	Tip podatka	Opis varijable
ocjenjujeldKor	INTEGER	Primarni i strani ključ
ocijenjenIdKor	INTEGER	Primarni i strani ključ
		Ocjena kojom korisnik ocjenjuje drugog korisnika na temelju obavljene

ocjena Atribut	INTEGER Tip podatka	kupnje Opis varijable
-------------------	------------------------	--------------------------

zainteresiran

Atribut	Tip podatka	Opis varijable
idKor	INTEGER	Primarni i strani ključ
idDog	INTEGER	Primarni i strani ključ, označuje id događaja za koji je korisnik zainteresiran

deaktivira

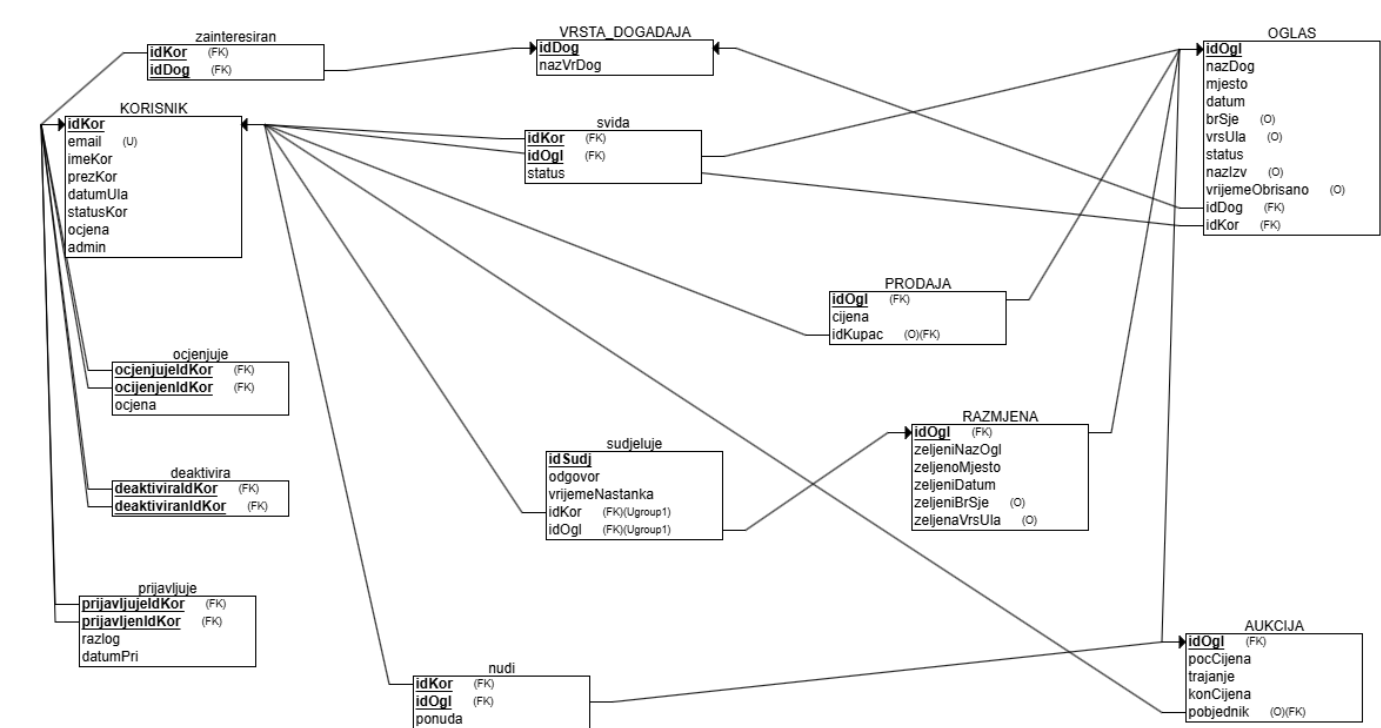
Atribut	Tip podatka	Opis varijable
deaktiviraIdKor	INTEGER	Primarni i strani ključ, označuje id admina koji deaktivira korisnika
deaktiviraIdKor	INTEGER	Primarni i strani ključ, označuje id korisnika koji je deaktiviran

svida

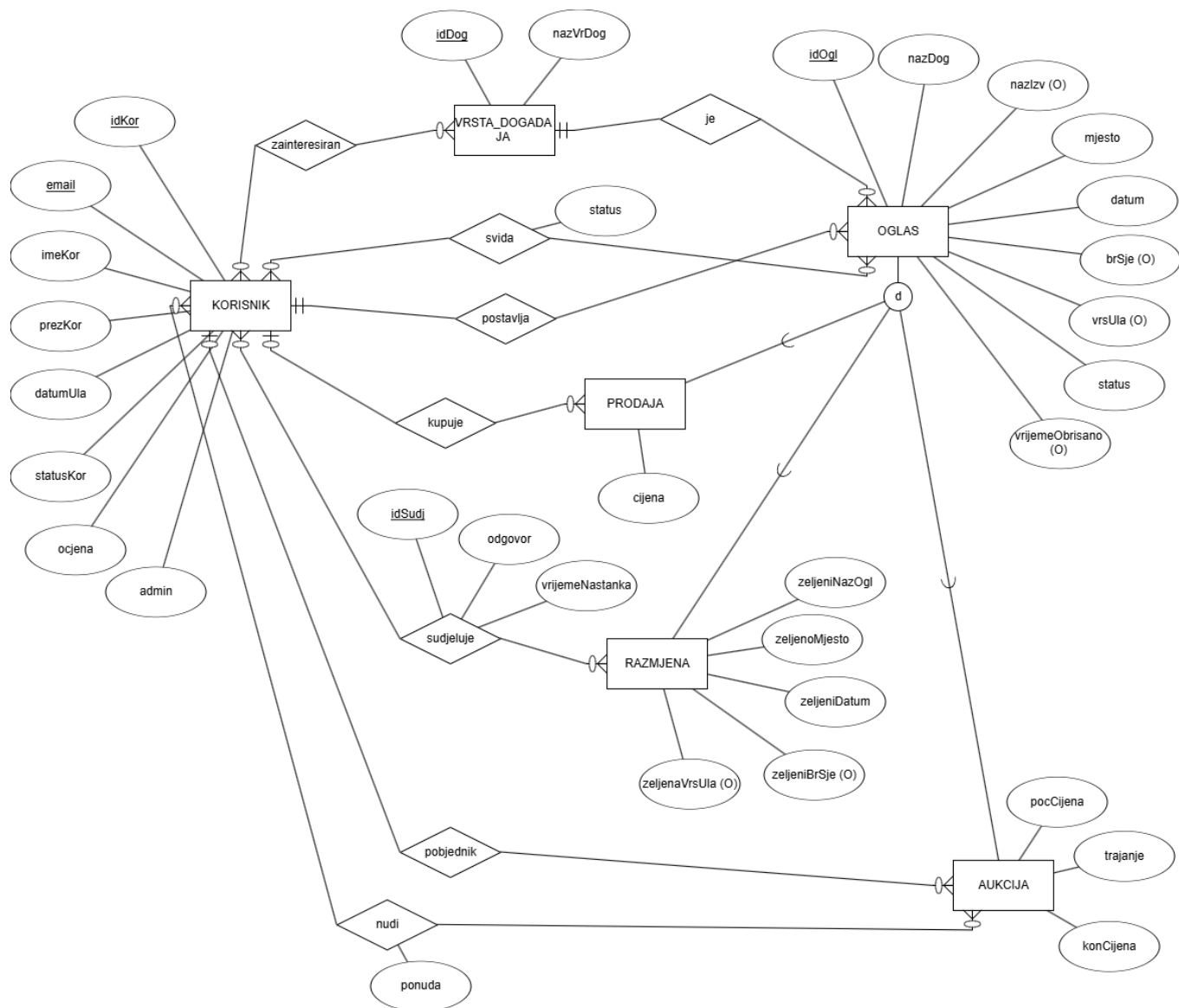
Atribut	Tip podatka	Opis varijable
idKor	INTEGER	Primarni i strani ključ, označuje id korisnika koji označuje oglas
idOgl	INTEGER	Primarni i strani ključ, označuje id oglasa koji je označen
status	BOOLEAN	Korisnik je označio oglas sa sviđa mi se ili ako se radi o false vrijednosti sakrio oglas

## Dijagram baze podataka

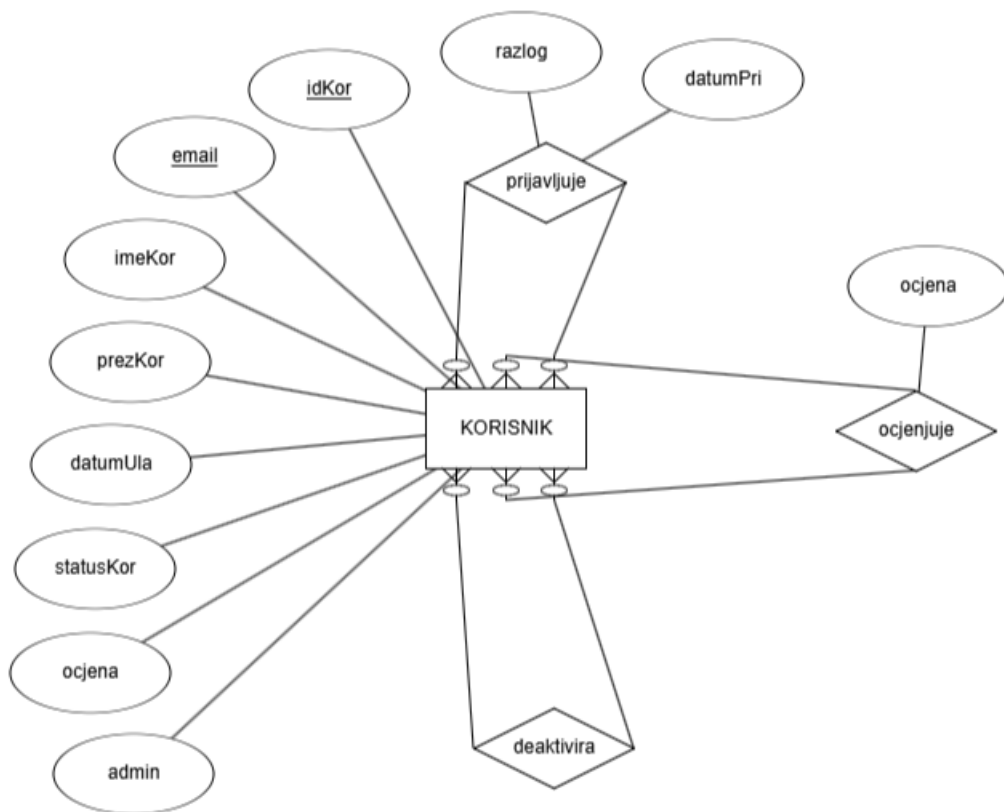
Relacijski dijagram baze podataka



ER dijagram baze podataka



Zbog bolje vidljivosti na zasebni ER dijagram su stavljene sve refleksivne N:N veze vezane za korisnika.



## Dijagram razreda

Prva slika prikazuje sve entitete korištene u aplikaciji, koji su mapirani na odgovarajuće tablice u bazi podataka. Entiteti su označeni pomoću Entity Bean oznake, što ukazuje na to da su u pitanju Java klase koje predstavljaju podatke pohranjene u bazi podataka. ORM Persistable označava da klase komuniciraju s bazom podataka. Omogućava automatsko upravljanje zapisima i omogućava aplikaciji da izvršava upite nad bazom podataka bez potrebe za pisanjem ručnih SQL upita. Property označava postojanje getter i setter metoda za attribute svakog entiteta, čime se omogućava pristup i izmjena tih podataka.



Druga slika prikazuje repozitorije i servise. Repository sloj omogućuje izravnu komunikaciju aplikacije s bazom podataka, i dio je sloja aplikacije odgovornog za podatkovni pristup. U repository sloju nalaze se metode koje omogućuju operacije poput čitanja, pisanja, ažuriranja i brisanja podataka u bazi podataka. Service sloj sadrži poslovnu logiku aplikacije i služi kao posrednik između kontrolera i repository sloja. Zavisnosti (repository objekti) se uvode putem dependency injection-a.



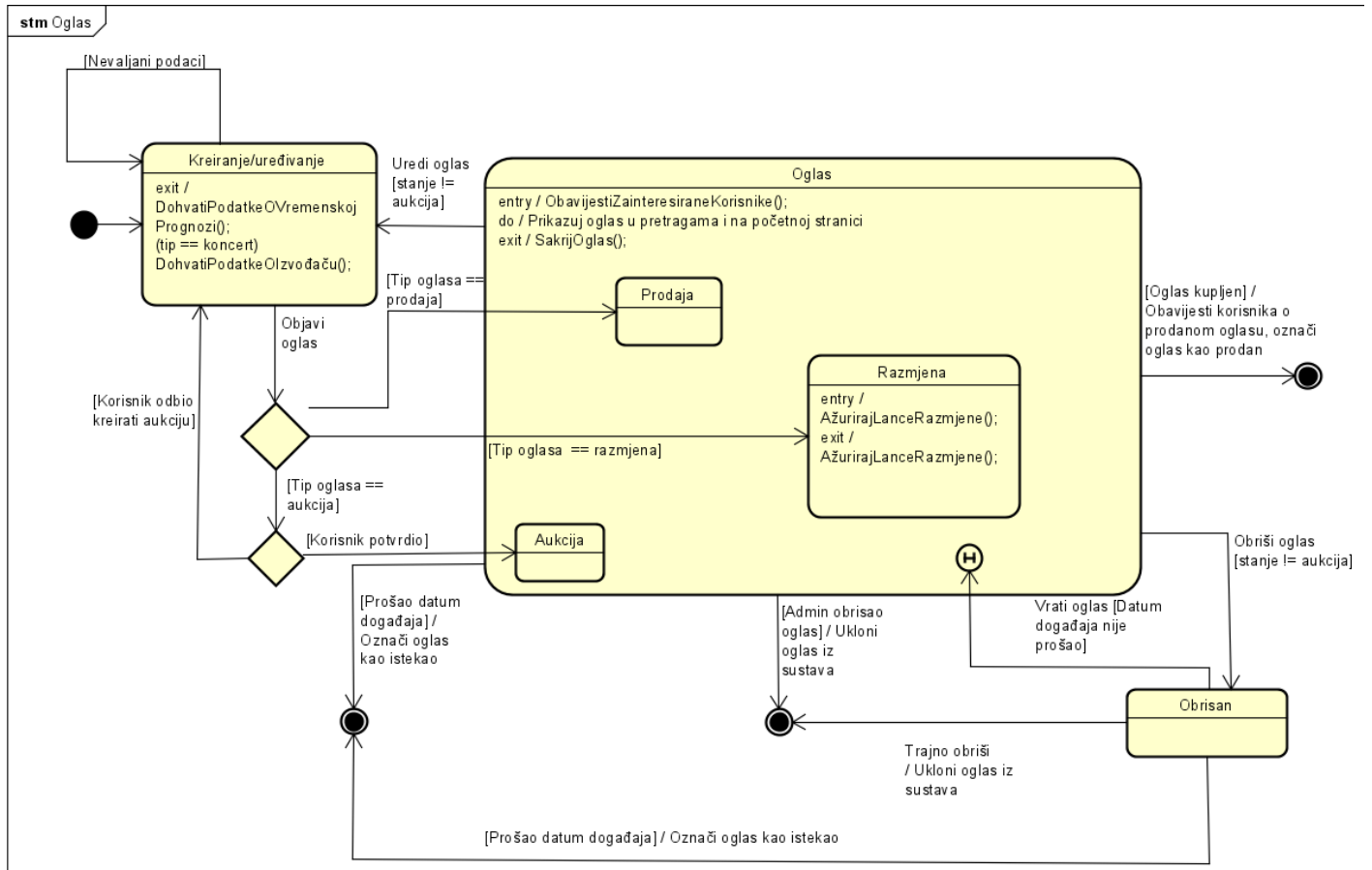
## Dinamičko ponašanje aplikacije

Dinamičko ponašanje aplikacije odnosi se na način na koji objekti u sustavu evoluiraju kroz vrijeme, uključujući prijelaze između različitih stanja. To uključuje aktivnosti, događaje, odluke i interakcije unutar aplikacije. UML dijagrami stanja omogućuju vizualizaciju tih promjena i olakšavaju razumijevanje dinamike sustava.

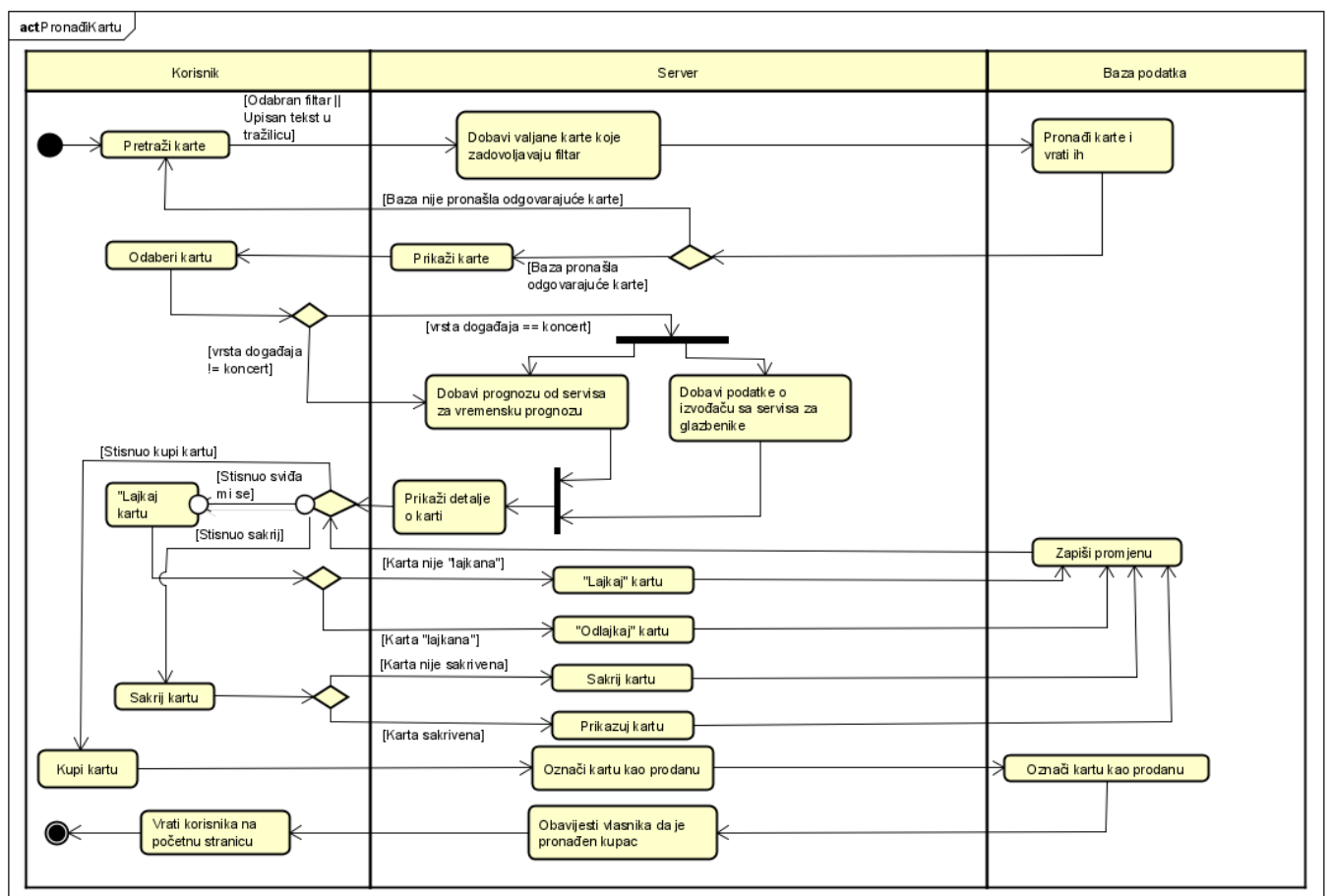
Razumijevanje promjena stanja neophodno je za pravilno funkcioniranje aplikacije jer pruža uvid u interakcije među objektima, komponentama i korisnicima tijekom rada sustava. Korištenjem UML dijagrama stanja i aktivnosti moguće je vizualizirati prijelaze i stanja objekata, identificirati potencijalne probleme, osigurati točnu implementaciju te poboljšati komunikaciju među članovima tima.

## Dijagram stanja za oglas



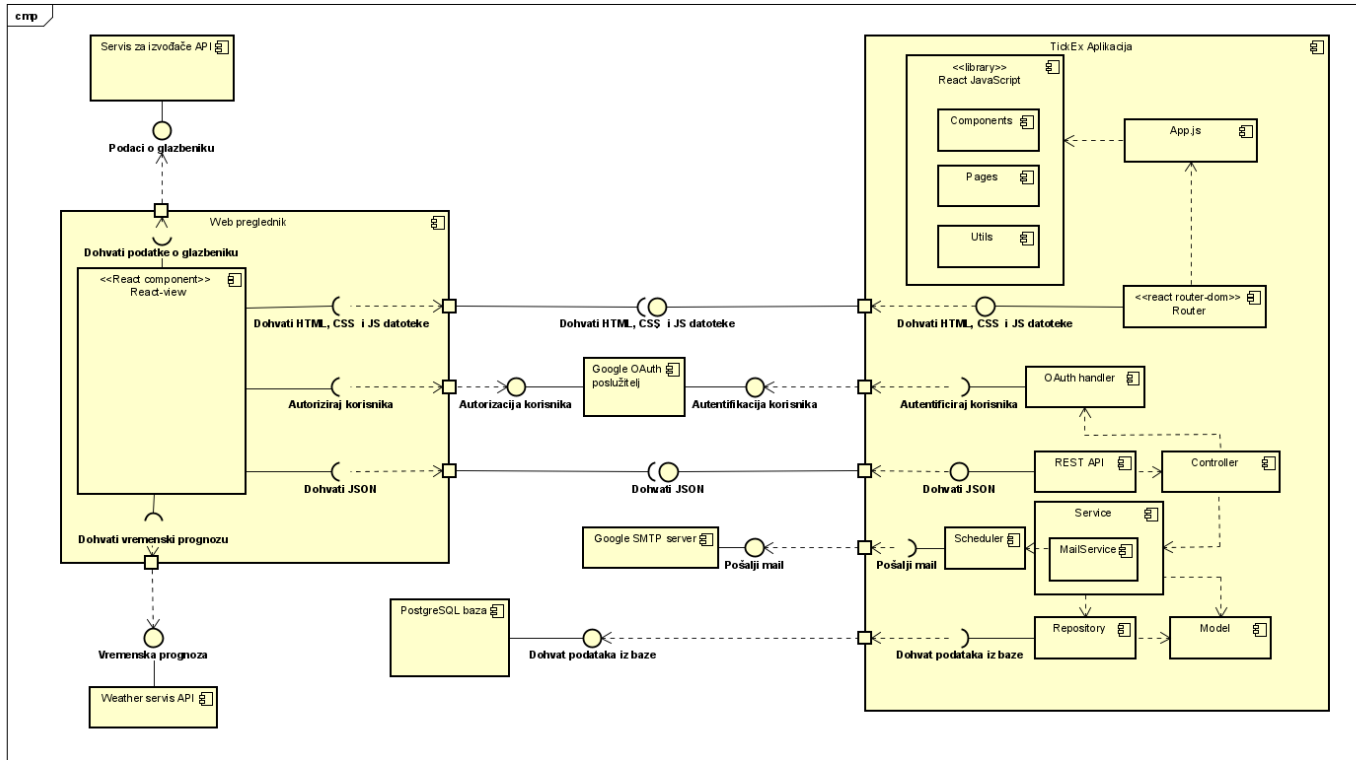


## Dijagram aktivnosti za pronalazak karte



## Dijagram komponentata

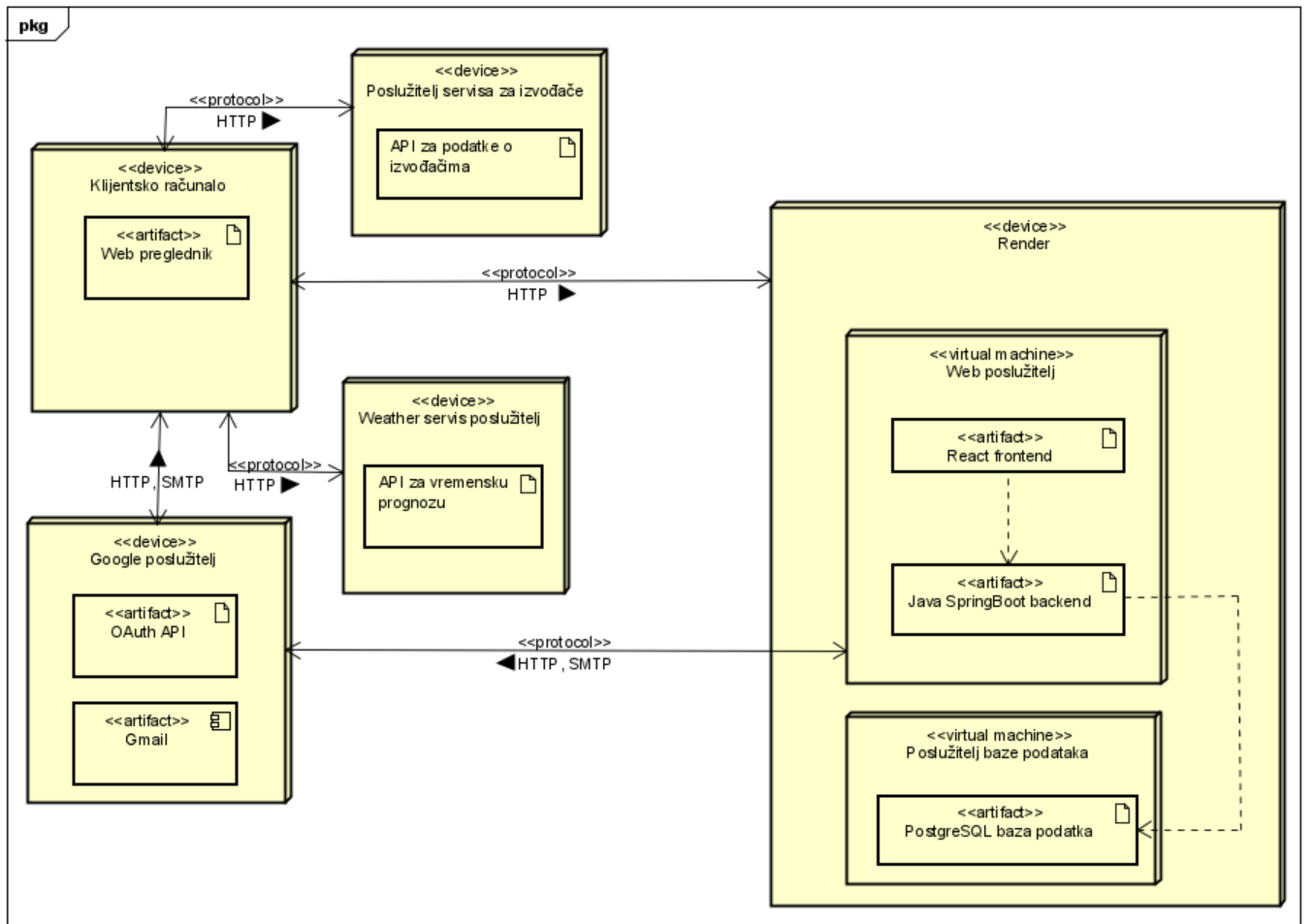
Dijagram komponenata služi za prikaz ključnih dijelova arhitekture naše web aplikacije. Prikazuje ključna sučelja potrebna za ostvarivanje funkcionalnosti kao i komponente koje ostvaruju ta sučelja. Osim našeg koda, neka ključna sučelja pružaju vanjski servisi i API-jevi kao što ovaj dijagram prikazuje.



## Dijagram razmještaja

UML dijagram razmještaja prikazuje fizičku ili virtualnu raspodjelu komponenta sustava unutar infrastrukture. Pokazuje kako su komponente raspoređene na poslužiteljima, u okruženjima oblaka i na uređajima krajnjih korisnika te način komunikacije između navedenih komponenta.

Ovaj dijagram je u implementacijskom obliku koju pruža detaljnu sliku infrastrukture. Reflektira stvarnu situaciju rasporeda komponenti u našoj web aplikaciji.



Ovo poglavlje treba opisivati provedena ispitivanja implementiranih funkcionalnosti na razini komponenti i sustava. Fokus je na odabiru i izvedbi ispitnih slučajeva koji

obuhvaćaju redovne, rubne uvjete i testiranje grešaka, kao i upotrebu odgovarajućih alata za provedbu testiranja.

# Ispitivanje komponenti

Odrađeno je ispitivanje na 6 komponenti sustava. Radi se o različitim testovima koji ispituju funkcionalnosti komponenti pri kontroliranim i nekontroliranim uvjetima. Radi veličine kodova testova oni nisu na ovoj stranici već se nalaze [Ovdje](https://github.com/HJure/TickEx/tree/DEV/IzvorniKod/progi-project/src/test/java/progi_project). (u slučaju da link ne radi ,[https://github.com/HJure/TickEx/tree/DEV/IzvorniKod/progi-project/src/test/java/progi\\_project](https://github.com/HJure/TickEx/tree/DEV/IzvorniKod/progi-project/src/test/java/progi_project))

```
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running progi_project.controller.ReportControllerTest
WARNING: A Java agent has been loaded dynamically (C:\Users\Korisnik\.m2\repository\net\bytebuddy\byte-buddy-agent\1.14.19\byte-buddy-agent-1.14.19.jar)
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.512 s -- in progi_project.controller.ReportControllerTest
[INFO] Running progi_project.controller.TicketControllerTest
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.116 s -- in progi_project.controller.TicketControllerTest
[INFO] Running progi_project.controller.UserControllerTest
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.043 s -- in progi_project.controller.UserControllerTest
[INFO] Running progi_project.service.ReportServiceTest
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.216 s -- in progi_project.service.ReportServiceTest
[INFO] Running progi_project.service.TicketServiceTest
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.116 s -- in progi_project.service.TicketServiceTest
[INFO] Running progi_project.service.UserServiceTest
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.080 s -- in progi_project.service.UserServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 35, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 5.386 s
[INFO] Finished at: 2025-01-24T15:22:12+01:00
[INFO] -----
```

## Izveštaj testiranja: ReportControllerTest

### Ispitni slučaj 1: Dohvaćanje prijava od strane admina

Ulaz:

- 1. Admin pokušava dohvatiti prijave.

Očekivani rezultat:

- 1. Aplikacija vraća status **200 OK**.
- 2. Vraća listu prijava.

Rezultat: Prolazi. Admin uspješno dohvaća listu prijava, a aplikacija vraća status **200 OK** i odgovarajući sadržaj.

### Ispitni slučaj 2: Dohvat prijava od strane običnog korisnika

Ulaz:

- 1. Obični korisnik pokušava dohvatiti listu izvještaja.

Očekivani rezultat:

- 1. Aplikacija vraća status **403 Forbidden**.
- 2. Vraća poruku: "Access denied: You don't have permission to view reports."

Rezultat: Prolazi. Korisniku ne može pristupiti prijavama, a aplikacija vraća status **403 Forbidden** s porukom.

### Ispitni slučaj 3: Validno prijavljivanje korisnika

Ulaz:

- 1. Korisnik pokušava prijaviti drugog korisnika s valjanim podacima (ID prijavitelja, ID prijavljenog i razlog prijave).

Očekivani rezultat:

- 1. Aplikacija provjerava valjanost podataka.
- 2. Kreira prijavu i bilježi ju u sustavu.
- 3. Vraća status uspješnosti.

Rezultat: Prolazi. Korisnik uspješno prijavljuje drugog korisnika.

---

## Ispitni slučaj 4: Pokušaj prijavljivanja samoga sebe

### Ulaz:

1. Korisnik pokušava prijaviti sebe (isti ID prijavitelja i prijavljenog).

### Očekivani rezultat:

1. Aplikacija odbacuje prijavu.
2. Ne šalje zahtjev za kreiranje prijave.

**Rezultat:** Prolazi. Prijava je odbijena, a aplikacija nije poslala zahtjev kreiranje prijave.

---

## Ispitni slučaj 5: Pokušaj ponovne prijave istoga korisnika

### Ulaz:

1. Korisnik pokušava prijaviti drugog korisnika, ali ne ispunjava uvjete (npr. već je prijavio tog korisnika).

### Očekivani rezultat:

1. Aplikacija provjerava uvjete prijave.
2. Odbija prijavu i ne bilježi je u sustavu.

**Rezultat:** Prolazi. Aplikacija odbacuje prijavu, a sustav ne bilježi prijavu.

---

## Izveštaj testiranja: ReportServiceTest

---

### Ispitni slučaj 1: Kreiranje prijave s validnim ID-jevima

#### Ulaz:

1. Reporter ID: 1
2. Reported ID: 2
3. Razlog prijave: "Test"

#### Očekivani rezultat:

1. Aplikacija dohvaća korisnike prema zadanim ID-jevima.
2. Kreira prijavu s odgovarajućim podacima (reporter, reported, razlog, datum).
3. Sprema prijavu u repozitorij.

**Rezultat:** Prolazi. Prijava je uspješno kreirana i spremljena u repozitorij s odgovarajućim podacima.

---

### Ispitni slučaj 2: Kreiranje prijave sa nevažećim ID-jem prijavitelja

#### Ulaz:

1. Reporter ID: 1 (nevažeći)
2. Reported ID: 2
3. Razlog prijave: "Test"

#### Očekivani rezultat:

1. Aplikacija ne pronalazi korisnika s ID-jem prijavitelja.
2. Ne kreira i ne sprema prijavu u repozitorij.

**Rezultat:** Prolazi. Prijava nije kreirana jer ID prijavitelja nije dobar. Nije zabilježeno spremanje u repozitorij.

---

### Ispitni slučaj 3: Kreiranje prijave s nevažećim ID-jem prijavljenog korisnika

#### Ulaz:

1. Reporter ID: 1
2. Reported ID: 2 (nevažeći)
3. Razlog prijave: "Test"

#### Očekivani rezultat:

1. Aplikacija ne pronalazi korisnika s ID-jem prijavljenog korisnika.
2. Ne kreira i ne sprema prijavu u repozitorij.

**Rezultat:** Prolazi. Prijava nije kreirana jer ID prijavljenog korisnika ne postoji. Nije zabilježeno spremanje u repozitorij.

---

## Ispitni slučaj 4: Kreiranje prijave bez navedenog razloga

### Ulaz:

- Reporter ID: 1
- Reported ID: 2
- Razlog prijave: null

### Očekivani rezultat:

- Aplikacija dopušta kreiranje prijave s nepostojećim razlogom.
- Sprema prijavu u repozitorij sa null vrijednošću za razlog.

**Rezultat:** Prolazi. Prijava je kreirana i spremljena s vrijednošću null kao razlog.

---

## Ispitni slučaj 5: Dohvat prijava

### Ulaz:

- Korisnik zahtijeva dohvat prijava.

### Očekivani rezultat:

- Aplikacija dohvaća prijave i vraća ih korisniku kao listu.

**\*\* Rezultat:\*\* Prolazi.** Aplikacija je uspješno dohvatila prijave i vratila ih korisniku.

## Izvještaj testiranja: TicketControllerTest

---

### Ispitni slučaj 1: Kreiranje nove karte

#### Ulaz:

- Nova karta ( Ticket )

#### Očekivani rezultat:

- Aplikacija poziva servis za kreiranje nove karte.
- Karta se uspješno kreira.

**Rezultat:** Prolazi. Karta je uspješno kreirana, a servis je pozvan jednom.

---

### Ispitni slučaj 2: Dohvat karte sa ispravnim ID-jem

#### Ulaz:

- ID karte: 1

#### Očekivani rezultat:

- Aplikacija dohvaća kartu s danim ID-jem.
- Vraća objekt karte.

**Rezultat:** Prolazi. Karta s odgovarajućim ID-jem je uspješno dohvaćena.

---

### Ispitni slučaj 3: Dohvat karte s nevažećim ID-jem

#### Ulaz:

- ID karte: 1 (nevažeći)

#### Očekivani rezultat:

- Aplikacija pokušava dohvatiti kartu s danim ID-jem.
- Vraća null.

**Rezultat:** Prolazi. Nije pronađena karta s ID-jem i vraćena je vrijednost null.

---

### Ispitni slučaj 4: Dohvat preporučenih karata za korisnika

#### Ulaz:

- ID korisnika: 1

#### Očekivani rezultat:



1. Aplikacija dohvaća preporučene karte za korisnika.
2. Vraća listu preporučenih karata.

**Rezultat:** Prolazi. Preporučene karte uspješno dohvaćene.

---

## Ispitni slučaj 5: Dohvaćanje svih karata

**Ulaz:**

1. Bez ulaza.

**Očekivani rezultat:**

1. Aplikacija dohvaća sve karte.
2. Vraća listu svih karata.

**Rezultat:** Prolazi. Sve karte uspješno dohvaćene.

---

## Ispitni slučaj 6: Ažuriranje statusa karte

**Ulaz:**

1. ID karte: 1
2. Mapa sa novim statusom: { "status": "obrisano" }

**Očekivani rezultat:**

1. Aplikacija pronalazi kartu s danim ID-jem.
2. Ažurira se status karte.

**Rezultat:** Prolazi. Status karte uspješno je ažuriran na "obrisano".

---

## Izveštaj testiranja: TicketServiceTest

---

### Ispitni slučaj 1: Kreiranje nove karte

**Ulaz:**

1. Nova karta ( Ticket )

**Očekivani rezultat:** 2. Karta se uspješno sprema.

**Rezultat:** Prolazi. Karta je uspješno spremljena.

---

### Ispitni slučaj 2: Dohvat karte sa ispravnim ID-jem

**Ulaz:**

1. ID karte: 1

**Očekivani rezultat:**

1. Aplikacija dohvaća kartu s danim ID-jem.
2. Vraća odgovarajuću kartu.

**Rezultat:** Prolazi. Karta s odgovarajućim ID-jem je uspješno dohvaćena.

---

### Ispitni slučaj 3: Dohvat karte prema nevažećem ID-u

**Ulaz:**

1. ID karte: 84698384

**Očekivani rezultat:**

1. Aplikacija pokušava dohvatiti kartu s danim ID-jem.
2. Ne pronalazi kartu i vraća null.

**Rezultat:** Prolazi. Nije pronađena karta i vraćena je vrijednost null.

---

### Ispitni slučaj 4: Dohvat svih karata

**Ulaz:**

1. Bez ulaza.

**Očekivani rezultat:**

1. Aplikacija dohvaća sve karte i vraća ih kao listu.

**Rezultat:** Prolazi. Sve karte uspješno su dohvaćene.

---

## Ispitni slučaj 5: Ažuriranje karata

**Ulaz:**

1. ID karte: 1
2. Ažurirana karta ( Ticket )

**Očekivani rezultat:**

1. Aplikacija pronalazi postojeću kartu s danim ID-jem.
2. Ažurira postojeću kartu s podacima iz ažurirane karte.
3. Sprema ažuriranu kartu.

**Rezultat:** Prolazi. Postojeća karta uspješno je ažurirana i spremljena.

---

## Ispitni slučaj 6: Brisanje karte sa ispravnim ID-jem

**Ulaz:**

1. ID karte: 1

**Očekivani rezultat:**

1. Aplikacija provjerava postoji li karta s danim ID-jem.
2. Briše kartu iz baze.

**Rezultat:** Prolazi. Karta uspješno obrisana.

---

## Ispitni slučaj 7: Brisanje karte sa krivim ID-jem

**Ulaz:**

1. ID karte: 1 (nevažeći)

**Očekivani rezultat:**

1. Aplikacija provjerava postojanje karte.
2. Budući da karta ne postoji, baca iznimku `EntityNotFoundException`.

**Rezultat:** Prolazi. Iznimka `EntityNotFoundException` uspješno je bačena, a brisanje karte se nije izvršilo.

---

## Izveštaj testiranja: UserControllerTest

---

### Ispitni slučaj 1: Prijava korisnika

**Ulazi:**

1. ID korisnika koji prijavljuje: 1
2. ID prijavljenog korisnika: 2
3. Razlog prijave: "Test"

**Očekivani rezultati:**

1. Aplikacija kreira prijavu korisnika koristeći zadane podatke.
2. Poziva se metoda `createReport` iz servisa za prijave.

**Rezultat:** Prolazi. Prijava je uspješno kreirana, a metoda `createReport` je pozvana jednom.

---

### Ispitni slučaj 2: Dohvaćanje korisnika prema ispravnom ID-u

**Ulazi:**

1. ID korisnika: 1

**Očekivani rezultati:**

1. Aplikacija dohvaća korisnika s odgovarajućim ID-jem.
2. Vraća ispravan objekt korisnika.

**Rezultat: Prolazi.** Korisnik s odgovarajućim ID-jem uspješno dohvaćen.

---

### Ispitni slučaj 3: Dohvaćanje korisnika sa nevažećim ID-jem

Ulazi:

- ID korisnika: 84698384 (nevažeći)

Očekivani rezultati:

- Aplikacija pokušava dohvatiti korisnika s danim ID-jem.
- Budući da korisnik ne postoji, vraća null.

**Rezultat: Prolazi.** Korisnik nije pronađen i metoda je vratila null.

---

### Ispitni slučaj 4: Uspješna registracija korisnika

Ulazi:

- Email: "peroperic@gmail.com"
- Ime: "Pero"
- Prezime: "Peric"
- Datum registracije: 2024-11-01

Očekivani rezultati:

- Korisnik se uspješno registrira u sustav.
- Podaci o korisniku su ispravno spremljeni:
  - Email: "peroperic@gmail.com"
  - Ime: "Pero"
  - Prezime: "Peric"
  - Status korisnika: true
  - Ocjena: 0.0
- Poziva se metoda registerUser iz servisa za korisnike.

**Rezultat: Prolazi.** Korisnik je uspješno registriran, a metoda registerUser je pozvana jednom.

---

### Ispitni slučaj 5: Dohvati korisnika prema nepostojećem ID-ju

Ulazi:

- ID korisnika: 84698384 (nevažeći)

Očekivani rezultati:

- Aplikacija pokušava dohvatiti korisnika prema zadanom ID-ju.
- Vraća se vrijednost null jer korisnik ne postoji.

**Rezultat: Prolazi.** Metoda je vratila null.

---

### Ispitni slučaj 6: Dohvaćanje korisnikovih ulaznica (korisnik nema ulaznica)

Ulazi:

- ID korisnika: 1

Očekivani rezultati:

- Aplikacija dohvaća listu ulaznica za korisnika.
- Lista ulaznica je prazna jer korisnik nema ulaznica.

**Rezultat: Prolazi.** Lista ulaznica je prazna, a metoda je uspješno pozvana.

---

## Izvještaj testiranja: UserServiceTest

---

### Ispitni slučaj 1: Provjera postojanja email-a (email postoji)

Ulazi:

- Email: anaanic143@gmail.com

Očekivani izlazi:

- Vraća true.
- Metoda existsByEmail pozvana jednom.

**Rezultat:** Prolazi. Sustav je prepoznao da email postoji i vratio `true`.

---

## Ispitni slučaj 2: Provjera postojanja email-a (email ne postoji)

Ulazi:

1. Email: `peroperic@gmail.com`

Očekivani izlazi:

1. Vraća `false`.
2. Metoda `existsByEmail` pozvana jednom.

**Rezultat:** Prolazi. Sustav je ispravno prepoznao da email ne postoji.

---

## Ispitni slučaj 3: Provjera postojanja email-a (email je null)

Ulazi:

1. Email: `null`

Očekivani izlazi:

1. Sustav vraća `false`.
2. Metoda `existsByEmail` pozvana jednom.

**Rezultat:** Prolazi. Sustav je ispravno prepoznao da e-mail nije valjan i vratio `null`.

---

## Ispitni slučaj 4: Provjera postojanja email-a (email je prazan)

Ulazi:

1. Email: `""`

Očekivani izlazi:

1. Vraća `false`.
2. Metoda `existsByEmail` pozvana jednom.

**Rezultat:** Prolazi. Sustav je prepoznao da email nije valjan i vratio `false` vrijednost.

---

## Ispitni slučaj 5: Provjera može li korisnik ocjenjivati (nema ocjene)

Ulazi:

1. Kupac ID: `1`
2. Vlasnik ID: `2`

Očekivani izlazi:

1. Vraća `true`.
2. Metoda `findById` pozvana jednom.

**Rezultat:** Prolazi. Sustav je ispravno prepoznao da korisnik može ocijeniti.

---

## Ispitni slučaj 6: Provjera može li korisnik ocjenjivati (ocjena postoji)

Ulazi:

1. Kupac ID: `1`
2. Vlasnik ID: `2`

Očekivani izlazi:

1. Sustav vraća `false`.
2. Metoda `findById` je pozvana jednom.

**Rezultat:** Prolazi. Sustav je ispravno prepoznao da korisnik ne može ponovno ocijeniti.

---

## \*\*Ispitivanje sustava \*\*

Cilj ispitivanja sustava je testiranje ponašanja cijelog sustava u uvjetima stvarnog korištenja, uz posebnu pažnju na međusobnu povezanost svih komponenti.

Ispitivanje treba obuhvatiti sve aspekte sustava i njegovu interakciju s korisnicima.

## Ispit 1: Stvaranje oglasa (redovni slučaj)

---

### Ulazi:

- Podaci:
  - Namjena: "Prodaja"
  - Kategorija: "Glazba"
  - Naziv oglasa: "Koncert AG"
  - Naziv artista: "Ariana Grande"
  - Cijena: 150 EUR
  - Mjesto: "Zagreb"
  - Datum: 01-06-2025
  - Vrsta oglasa: "VIP"
  - Broj sjedala: 5

### Koraci ispitivanja:

1. Otvoriti aplikaciju.
2. Otići na profil.
3. Kliknuti na opciju za stvaranje novog oglasa.
4. Unijeti podatke u odgovarajuće forme.
5. Kliknuti na gumb za potvrdu stvaranja oglasa.

### Očekivani izlaz:

- Oglas bi trebao biti dodan u sustav i vidljiv na stranici sa svim oglasima.

### Dobiveni izlaz:

- Uspjeh

## Ispit 2: Stvaranje oglasa (redovni slučaj)

---

### Ulazi:

- Podaci:
  - Namjena: "Razmjena"
  - Kategorija: "Kazalište"
  - Naziv oglasa: "Shakespeare Play"
  - Cijena: 5 EUR
  - Mjesto: "London"
  - Datum: 15-07-2025
  - Vrsta oglasa: "Standard"
  - Broj sjedala: 5
  - Željeni naziv oglasa: "Test Exchange Event"
  - Željeno mjesto: "Zagreb"
  - Željeni datum: 01-10-2025
  - Željeni broj mjesta: 5

### Koraci ispitivanja:

1. Otvoriti aplikaciju.
2. Otići na profil.
3. Kliknuti na opciju za stvaranje novog oglasa.
4. Unijeti podatke u odgovarajuće forme.
5. Kliknuti na gumb za potvrdu stvaranja oglasa.

### Očekivani izlaz:

- Oglas bi trebao biti dodan u sustav i vidljiv na stranici sa svim oglasima.

### Dobiveni izlaz:

- Uspjeh

## Ispit 3: Stvaranje oglasa (redovni slučaj)

---

### Ulazi:

- Podaci:
  - Namjena: "Aukcija"
  - Kategorija: "Nogomet"

- Naziv oglasa: "Dinamo vs Hajduk"
- Cijena: 5 EUR
- Mjesto: "Zagreb"
- Datum: 12-04-2025
- Vrsta oglasa: "Standard"
- Broj sjedala: 5
- Trajanje: 09-09-2025

#### Koraci ispitivanja:

1. Otvoriti aplikaciju.
2. Otići na profil.
3. Kliknuti na opciju za stvaranje novog oglasa.
4. Unijeti podatke u odgovarajuće forme.
5. Kliknuti na gumb za potvrdu stvaranja oglasa.

#### Očekivani izlaz:

- Oglas bi trebao biti dodan u sustav i vidljiv na stranici sa svim oglasima.

#### Dobiveni izlaz:

- Uspijeh

### Ispit 4: Stvaranje oglasa sa rubnim uvjetima (nevažeci podaci)

---

#### Ulazi:

- Podaci:
  - Naziv oglasa: ""

#### Koraci ispitivanja:

1. Otvoriti aplikaciju.
2. Kliknuti na opciju za stvaranje novog oglasa.
3. Unijeti nevažeci podatke u formu.
4. Kliknuti na gumb za potvrdu stvaranja oglasa.

#### Očekivani izlaz:

- Sustav treba prikazati poruku o pogrešci koja označava nevažeci podatke i ne treba dodati oglas u sustav.

#### Dobiveni izlaz:

- Sustav vraća upozorenje da polje treba biti ispunjeno

### Ispit 5: Stvaranje oglasa sa rubnim uvjetima (nevažeci podaci)

---

#### Ulazi:

- Podaci:
  - Cijena: "da"

#### Koraci ispitivanja:

1. Otvoriti aplikaciju.
2. Pokušati pristupiti funkcionalnosti koja ne postoji (npr. API endpoint koji nije implementiran).
3. Unijeti podatke za stvaranje oglasa.
4. Kliknuti na gumb za stvaranje oglasa.

#### Očekivani izlaz:

- Sustav treba prikazati odgovarajuću grešku (npr. 404, 500) ili poruku o grešci koja označava da funkcionalnost nije dostupna.

#### Dobiveni izlaz:

- Sustav vraća upozorenje da polje treba biti ispunjeno cijelim brojem

### Ispit 6: Uređivanje imena i prezimena (redovni slučaj)

---

#### Ulazi:

- **Podaci:**
  - Ime: "Novolme"
  - Prezime: "NovoPrezime"

#### Koraci ispitivanja:

1. Otvoriti aplikaciju.
2. Otići na profil.
3. Kliknuti na gumb "Uredi podatke!".
4. Unijeti novo ime i prezime u odgovarajuća polja.
5. Kliknuti na gumb za potvrdu promjena.

#### Očekivani izlaz:

- Ime i prezime korisnika trebaju biti ažurirani na "Novolme" i "NovoPrezime".

#### Dobiveni izlaz:

- Uspijeh

### Ispit 7: Promjena preferencija (redovni slučaj)

---

#### Ulazi:

- **Podaci:**
  - Promjena preferencija: Označiti checkbox

#### Koraci ispitivanja:

1. Otvoriti aplikaciju.
2. Otići na profil.
3. Kliknuti na gumb "Uredi podatke!".
4. Unijeti novo ime i prezime u odgovarajuća polja.
5. Kliknuti na gumb za potvrdu promjena.

#### Očekivani izlaz:

- Preferencija treba biti označena, što znači da je korisnik uspješno promijenio preferenciju.

#### Dobiveni izlaz:

- Uspijeh

### Ispit 8: Prazno ime (rubni uvjet)

---

#### Ulazi:

- **Podaci:**
  - Ime: ""
  - Prezime: "NovoPrezime"

#### Koraci ispitivanja:

1. Otvoriti aplikaciju.
2. Otići na profil.
3. Kliknuti na gumb "Uredi podatke!".
4. Unijeti novo ime i prezime u odgovarajuća polja.
5. Kliknuti na gumb za potvrdu promjena.

#### Očekivani izlaz:

- Sustav treba prikazati poruku o grešci koja označava da ime nije unijeto i da promjena nije prihvaćena.

#### Dobiveni izlaz:

- Uspijeh, očekivana greška nije prikazana

### Ispit 9: Lajkanje oglasa (redovni slučaj)

---

#### Ulazi:

#### Koraci ispitivanja:

1. Otvoriti aplikaciju.
2. Otvoriti profil.
3. Navigirati na početnu stranicu.
4. Kliknuti na oglas.
5. Kliknuti na gumb za lajkanje oglasa.
6. Vratiti se na profil.
7. Otvoriti Oglase koji mi se sviđaju

#### Očekivani izlaz:

- Oglas se označava kao lajkani i to bi trebalo biti vidljivo na profilu ili na stranici s oglasima.

#### Dobiveni izlaz:

- Uspijeh

### Ispit 10: Neuspješan pokušaj lajkanja zbog nedostajuće "oglas-preview" stavke

---

#### Ulazi:

#### Koraci ispitivanja:

1. Otvoriti aplikaciju.
2. Otvoriti profil.
3. Navigirati na početnu stranicu.
4. Kliknuti na oglas.
5. Kliknuti na gumb za lajkanje oglasa.
6. Vratiti se na profil.
7. Otvoriti Oglase koji mi se sviđaju

#### Očekivani izlaz:

- Ako "oglas-preview" div nije pronađen, sustav treba obavijestiti o grešci ili izvesti alternativnu akciju, poput nastavka navigacije bez promjena.

#### Dobiveni izlaz:

- Ništa se ne dogodi jer ne postoji oglas za lajkanje

### Ispit 11: Brisanje oglasa (redovni slučaj)

---

#### Ulazi:

#### Koraci ispitivanja:

1. Pokrenuti aplikaciju.
2. Otvoriti profil.
3. Kliknuti na kategoriju moje ponude.
4. Odabrati prvi oglas sa liste.
5. Kliknuti na gumb za brisanje oglasa.

#### Očekivani izlaz:

- Oglas se briše i više nije vidljiv na stranici s oglasima.

#### Dobiveni izlaz:

- Uspijeh

### Ispit 12: Brisanje oglasa kada ne postoji 'ticket-preview' div (rubni uvjet)

---

#### Koraci ispitivanja:

1. Pokrenuti aplikaciju.
2. Otvoriti profil.
3. Kliknuti na kategoriju moje ponude.
4. Odabrati prvi oglas sa liste.
5. Kliknuti na gumb za brisanje oglasa.

#### Očekivani izlaz:



- Ako 'ticket-preview' div nije pronađen, brisanje oglasa ne bi trebalo biti moguće.

#### Dobiveni izlaz:

- Ništa, s obzirom da ne postoji naš ticket, ne možemo ga obrisati

## Korištene tehnologije i alati

---

### 1. Programski jezici

---

#### Java (verzija 22)

Java je odabrana za razvoj backend-a aplikacije radi toga što je:

- Jako raširena, stabilna, efikasna i nudi visoke performanse.
- Ima široku podršku za razvoj velikih aplikacija, omogućujući laku i jednostavnu integraciju s različitim tehnologijama i alatima.
- Pruža mnoštvo biblioteka i alata koji ubrzavaju razvoj i olakšavaju održavanje koda aplikacije.
- Objektno orijentirana priroda Java-e omogućuje modularnost i izradu lako proširivog koda.

#### JavaScript (verzija 20.14)

JavaScript je korišten za razvoj frontend-a aplikacije:

- Omogućuje dinamičko upravljanje korisničkim sučeljem i brzu interakciju s korisnikom.
- Široka podrška za različite radne okvire, poput React-a, povećava primjenjivost i olakšava rad nad kompleksnim aplikacijama.
- Fleksibilnost JavaScript-a čini ga idealnim za implementaciju interaktivnih funkcionalnosti, poput animacija, validacija obrazaca i manipulacije DOM elementima.

---

### 2. Radni okviri i biblioteke

---

#### Spring Boot (verzija 3.3.5)

- Moderan radni okvir za razvoj backend aplikacija, pojednostavljuje kreiranje RESTful web servisa.
- Prednost: automatizacija konfiguracija, što omogućuje brzi početak rada i smanjuje kompleksnost tokom razvoja.
- Ugrađena podrška za integraciju s bazama podataka, sigurnost, autentifikaciju i upravljanje sesijama.
- Fleksibilna prilagodba aplikacije specifičnim potrebama projekta.

#### React (verzija 18.3.1)

- Široko korištena JavaScript biblioteka za izgradnju interaktivnog korisničkog sučelja s višekratno upotrebljivim komponentama.
- Modularnost olakšava upravljanje kodom i doprinosi bržem testiranju pojedinačnih funkcionalnosti komponenti.
- Jednostavna sintaksa pogoduje razvoju i održavanju većih i kompleksnijih aplikacija.

#### JPA (verzija 6.5.3)

- Omogućuje jednostavno mapiranje podataka između Java objekata i relacijskih baza podataka.
- Smanjuje potrebu za pisanjem složenih SQL upita, što ubrzava razvoj i smanjuje mogućnost grešaka.

---

### 3. Baza podataka

---

#### PostgreSQL (verzija 16)

- Relacijska baza podataka poznata po pouzdanosti, skalabilnosti i podršci za napredne značajke.
- Tim je najbolje upoznat s PostgreSQL-om od svih drugih baza podataka.
- Pruža visoku razinu kompatibilnosti sa SQL standardima, što pojednostavljuje integraciju sa Spring Boot aplikacijom.
- Otpornost na kvarove i podrška za složene transakcije čine je idealnom za aplikacije koje obrađuju velike količine podataka.

---

### 4. Razvojni alati

---

#### Visual Studio Code (verzija 1.96)

- Lagan za korištenje i brz IDE s podrškom za veliki broj proširenja i prilagodbi.
- Ključne značajke:
  - Integrirani terminal.
  - Podrška za Git.

- Veliki izbor dodataka za prilagodbu radnog okruženja.
- Besplatan alat koji uvelike ubrzava i pojednostavljuje razvoj aplikacija.

## Git (verzija 2.39)

- Alat za upravljanje i dokumentiranje koda:
  - Omogućuje učinkovitu suradnju tima i praćenje povijesti razvoja projekta.
  - Podrška za paralelni rad na različitim značajkama bez preklapanja koda.
  - Upravljanje granama omogućuje jednostavno spajanje i razdvajanje verzija aplikacije.

---

## 5. Alati za ispitivanje

### Selenium (verzija 4.23.0)

- Alat za automatizirano testiranje web aplikacija pomoću simulacije korisničkog ponašanja.
- Ključne prednosti:
  - Omogućuje pisanje testova za različite preglednike.
  - Podržava razne programske jezike, uključujući Javu i JavaScript.
  - Automatizacija rutinskih testova smanjuje vrijeme potrebno za ručno testiranje i povećava efikasnost.
  - Integracija s alatima poput TestNG i Jenkins omogućuje kontinuiranu integraciju i isporuku.

### JUnit 5 (verzija 5.10)

- JUnit 5 je popularni framework za testiranje u Javi.
- Omogućuje lakše upravljanje i kreiranje testova te pruža napredne mogućnosti za testiranja.
- Ključne prednosti:
  - Podrška za lambda izraze i stream-ove za efikasnije i kraće testove.
  - Testovi se mogu generirati tijekom izvođenja.
  - Omogućava korištenje dodatnih anotacija i ekstenzija za lakše integriranje sa raznim alatima i bibliotekama.

---

## 6. Cloud platforma (Render)

- Render je moderna cloud platforma koja omogućuje jednostavno hostanje aplikacija, statičnih stranica, API-ja, i baza podataka.
- Ključne prednosti:
  - Automatski deploy: Jednostavna integracija s GitHub-om ili GitLab-om za automatsko pokretanje deploy-a nakon svakog push-a.
  - Podrška za više jezika i okruženja: Podržava Node.js, Python, Ruby, Java, Go, Docker i druge tehnologije.
  - Pristupačna cijena: Besplatan sloj dostupan uz smanjene performanse.
  - Podrška za baze podataka: Omogućuje jednostavno postavljanje i upravljanje bazama podataka, poput PostgreSQL-a, uz automatske sigurnosne kopije.

---

## 7. Obrazloženje izbora tehnologija

- **Java i Spring Boot:** Stabilnost, pouzdanost, široka upotreba u industriji i veliki broj resursa i razvijene literature. Idealni za razvoj enterprise aplikacija s velikim brojem korisnika.
- **React:** Modularni razvoj i optimizacija pružaju izvrsno korisničko iskustvo. Široka zajednica programera osigurava stalnu podršku i nove značajke.
- **PostgreSQL:** Pouzdanost, napredne SQL mogućnosti i mogućnost rukovanja velikim količinama podataka čine je idealnom za velike aplikacije koje zahtijevaju integritet podataka.

---

## 8. Specifične postavke konfiguracije

### Backend

- Servis je konfiguriran za rad na portu **8080** (standardni port za web aplikacije).
- Omogućuje jednostavno povezivanje s drugim servisima i alatima za ispitivanje.

### Frontend

- Aplikacija radi na portu **3000** (standardni port za frontend aplikacije).
- Konfiguracija uključuje poveznice na REST API endpoint-ove backend servisa, omogućujući besprijekornu komunikaciju između korisničkog sučelja i serverske logike.

## Upute za puštanje u pogon (Render)

Render je moderna platforma za hosting koja omogućava brzo i jednostavno postavljanje aplikacija bez potrebe za kompleksnom infrastrukturom. Idealna je za manje projekte jer nudi besplatni plan koji uključuje automatski deploy putem GitHub-a te podršku za baze podataka poput PostgreSQL-a.

- **Priprema repozitorija:**

- Osigurajte da vaš projekt ima Dockerfile za konfiguraciju deploya
- Na backendu u application.properties postaviti server.servlet.context-path na /api, tako da svi zahtjevi na backendu imaju /api prefiks
- U datoteku pom.xml dodajte dependency za spring-boot-starter-actuator, koji će automatski omogućiti endpoint /actuator/health za prikaz informacija o statusu aplikacije, što Render koristi za health check

- **Dodavanje pipelineova:**

- Pipeline skripte nalaze se u direktoriju .github/workflows unutar repozitorija
- Skriptama je konfigurirano da se build automatski pokreće nakon pusha na main, ali se može pokrenuti i ručno putem taba Actions – odaberite željeni workflow i kliknite Run workflow

- **Postavljanje na Render:**


- Prijavite se na Render
- Kreirajte novi **Web Service** i povežite ga s vašim GitHub repozitorijem (Source Code postaviti na existing image, image URL zalijepiti putanju image-a u vašem repozitoriju)

## You are deploying a Web Service

### Source Code

Git Provider	Public Git Repository	Existing Image
--------------	-----------------------	----------------

Image URL  
Deploy an image from a Docker registry

 ghcr.io/hjure/ticex/backend:latest

Credential (Optional)

No credential

Connect →

- **Webhook**

- Da bismo Renderu omogućili automatsku obavijest o novom buildu i iniciranje redeploya, potrebno je postaviti webhook koji će pozivati GitHub pipeline:
1. U postavkama backend servisa na Renderu otvorite Settings i kopirajte vrijednost Deploy hook.

My Workspace

Aplikacija

Aplikacija

Events

Settings

MONITOR

Logs

Metrics

MANAGE

Environment

Shell

Scaling

Docs

Channels

Docker Command

Optionally override your Dockerfile's CMD and ENTRYPOINT instructions with a different command to start your service.

Edit

Pre-Deploy Command Optional

Render runs this command before the start command. Useful for database migrations and static asset uploads.

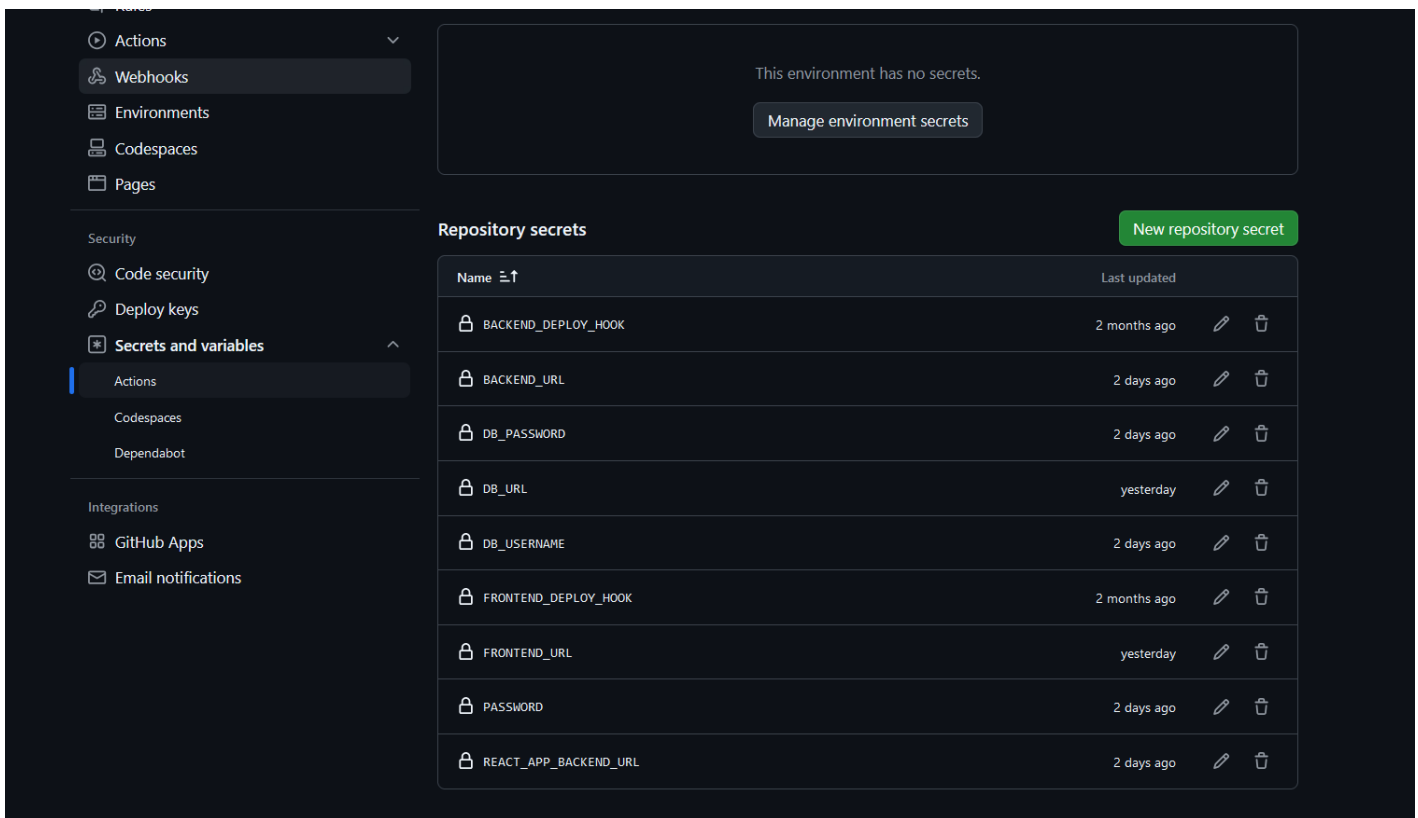
Edit

Deploy hook

Your private URL to trigger a deploy for this server. Remember to keep this a secret.

Regenerate hook

2. U GitHub repozitoriju otvorite Settings -> Secrets and variables -> Actions -> New repository secret.



3. Postavite naziv secreta na BACKEND\_DEPLOY\_HOOK / FRONTEND\_DEPLOY\_HOOK, zalijepite kopirani hook URL u polje vrijednosti i kliknite Add secret.

- **Dodavanje environment varijabli**

- Budući da se build proces izvodi na GitHubu, varijable okruženja postavljene na Renderu nisu dostupne tijekom izrade frontenda i backenda
- Vrijednosti tih varijabli se obrađuju u trenutku builda i trajno ugrađuju u statičke datoteke koje se kasnije posluživaju, pa promjene na Renderu ne utječu na te već generirane datoteke
- U GitHub Actions workflowu dodajte environment varijable tijekom koraka za izradu frontenda / backenda
- Postavite environment varijable u GitHub Secrets

- **Pokretanje aplikacije:**

- Render će automatski preuzeti repozitorij, instalirati ovisnosti i pokrenuti aplikaciju
- Nakon deploja, aplikaciji možete pristupiti putem generiranog URL-a

## Opis pristupa aplikaciji na javnom poslužitelju

### Pristup aplikaciji

Korisnici mogu otvoriti URL aplikacije u svom internetskom pregledniku. Aplikacija će biti dostupna nakon završetka build i deploy procesa.

### Ograničenja

- Web servisi na free planu mogu imati ograničenja u performansama i resursima.
- PostgreSQL baza podataka na free planu automatski se briše nakon 30 dana.
- Nakon 15 minuta neaktivnosti aplikacije, Render automatski gasi servise. Ponovno podizanje može trajati nekoliko minuta.
- GitHub Actions free plan pruža 2000 minuta mjesečno za build procese

Za pristup administracijskoj strani aplikacije putem Google logina:

Username: [anaanic143@gmail.com](mailto:anaanic143@gmail.com) Password: prog123

Izrada ovog projekta bila je zanimljivo, izazovno i poučno iskustvo za naš tim, iako je zahtijevala dosta vremena i truda. Na samom početku razmišljali smo kako osmisliti aplikaciju za razmjenu ulaznica na način koji će biti intuitivan, jednostavan za korištenje i funkcionalan. Uz osnovne zahtjeve iz zadatka, odlučili smo uvesti i dodatne funkcionalnosti kako bismo proširili mogućnosti aplikacije. Tako smo osmislili opcije poput aukcija za ulaznice i ocjenjivanja korisnika. Nakon što smo definirali glavne ciljeve, raspodijelili smo se u manje timove i svatko je preuzeo ulogu prema uputama s kolegija. Morali smo savladati nove tehnologije i alate koje do tada nismo koristili, što je bio poseban izazov, ali nam je ujedno pomoglo u širenju znanja i vještina.

Tijekom rada na projektu, od početka do kraja semestra, suočili smo se s tehničkim izazovima poput integracije frontenda, backenda i baze podataka, što je zahtijevalo pažljivo usklađivanje kako bi prijenos podataka funkcionirao bez poteškoća. Bazu podataka smo često nadograđivali, što je tražilo brojne prilagodbe i dosta strpljenja. Zbog njene veličine i složenosti, morali smo osigurati da se prema njoj šalju optimalni upiti kako bismo održali performanse aplikacije na razini. Deployanje aplikacije na server bilo je još jedan izazovan korak, posebno zbog konfiguracija i neočekivanih problema koji su se pojavljivali tijekom procesa. Ipak, unatoč svim poteškoćama, ovakvi zadaci pomogli su nam da bolje razumijemo kako različiti dijelovi aplikacije funkcioniraju i međusobno komuniciraju. Također, važan korak u izradi aplikacije bilo je dizajnirati i izraditi responzivno i korisniku intuitivno korisničko sučelje. Na samom kraju uslijedilo je i testiranje aplikacije, ispravljanje uočenih pogrešaka te sama dokumentacija.

Kroz ovaj projekt naučili smo izuzetno mnogo, počevši od rada preko Gita, zatim rada sa Spring Bootom za backend i Reactom za frontend, pa sve do upravljanja bazama podataka i bolje organizacije timskog rada. Ovo je bila naša prva prilika da sudjelujemo u razvoju stvarnog projekta, gdje smo mogli primijeniti sve dosad stečeno znanje, ali i naučiti mnoge nove vještine kroz rad na ovom semestru.

Shvatili smo koliko je važno unaprijed dobro isplanirati aplikaciju, odrediti jasne ciljeve i znati prioritizirati zadatke kako bismo izbjegli gubljenje vremena na manje bitne detalje. Također smo naučili koliko je bitno poštivati tuđe vrijeme i trud, redovito komunicirati s članovima tima te na vrijeme izvršavati ono što je svatko od nas preuzeo. Ovakav način rada omogućava bolje rezultate i poboljšava našu sposobnost rada u timu, što je ključno za uspješan završetak svakog projekta.

Zaključno smatramo da naša aplikacija, zahvaljujući svojim zanimljivim i korisnim funkcionalnostima, ima potencijal za stvarnu primjenu u svakodnevnom životu korisnika te predstavlja temelj za buduće nadogradnje. Ipak, svjesni smo da uvijek postoji prostor za poboljšanja i daljnji razvoj. Stoga predlažemo nekoliko rješenja koja bismo mogli implementirati u budućim fazama rada:

1. Pretraživanje ulaznica prema stvarnoj lokaciji korisnika: Kako bismo korisnicima omogućili personaliziranije iskustvo i olakšali

2. Povezivanje s javnom uslugom PayPal radi provedbe plaćanja unutar aplikacije: U cilju omogućavanja sigurnih i jednostavnih transa

3. Mogućnost chattanja preko aplikacije s korisnicima: Kako bismo poboljšali interakciju među korisnicima, predlažemo uvođenje opcij

4. Mogućnost uploadanja realnog PDF-a ulaznice koja je na prodaju, sudjeluje u aukciji ili razmjeni: Kako bi proces kupovine i prod

5. Mogućnost pregleda kontejnera sakrivenih ulaznica i opcija njihovog otkrivanja: Trenutno je implementirana opcija sakrivanja ulaz

6. Napredna analitika korisničkih aktivnosti: Ideja je dodati vizualne analitičke alate koji bi administratorima omogućili detaljan

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>

2. Baze podataka, FER ZPR, <https://www.fer.unizg.hr/predmet/bazpod>

3. Razvoj programske potpore za web, FER ZPR/ZARI, <https://www.fer.unizg.hr/predmet/web1>

4. Visual Paradigm, <https://www.visual-paradigm.com/>

5. The Unified Modeling Language, <https://www.uml-diagrams.org/>

6. Astah Community, <http://astah.net/editions/uml-new>

7. Layered Application Guidelines, [https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ee658109\(v=pandp.10\)](https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ee658109(v=pandp.10))

8. Microsoft Ignite, <https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/architectural-principles>

9. React, <https://react.dev/learn>

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak	Karlo Bašić	27.10.2024
0.2	Opisana baza podataka s prikladnim slikama	Karlo Bašić i Jure Herceg	31.10.2024.
0.3	Napisao sve funkcionalne zahtjeve	Jure Herceg	5.11.2024.
0.4	Dodao UC i sekvencijske dijagrame	Jure Herceg	5.11.2024
0.5	Dovršio stranicu 2. Analiza zahtjeva	Jure Herceg	9.11.2024
0.6	Dijagram razreda	Karlo Bašić	10.11.2024
0.7	Opis projekta i arhitektura sustava	Jure Herceg i David Kovčo	11.11.2024
0.8	Dodao licencu, readme, code of conduct	Jure Herceg	12.11.2024
0.9	Dovršio stranicu 4. Specifikacija zahtjeva sustava, pregledao, uredio i stilizirao ostale stranice	Jure Herceg	14.11.2024
1.0	Dovršena arhitektura , napravljena revizija	Karlo Bašić	15.11.2024.
1.1	Dovršena 7. stranica wiki-ja (Tehnologije za implementaciju aplikacije)	Ivan Đurić	21.1.2025.
1.2	Dovršen dijagram aktivnosti i dijagram stanja	Jure Herceg	22.1.2025.
1.3	Dovršen dijagram baze podataka	Karlo Bašić	23.1.2025.
1.4	Dovršen dijagram komponenti i razmještaja	Jure Herceg	23.1.2025.
1.5	Dovršeno ispitivanje komponenti	Ivan Đurić	24.1.2025.
1.6	Dovršen dijagram razreda	Karlo Bašić	24.1.2025.
1.7	Dovršeno ispitivanje sustava	David Kovčo	24.1.2025.
1.8	Napisan završetak	Laura Barišić	24.1.2025.
1.9	Zadnja revizija wikija, promjene i stiliziranje	Cijeli tim	24.1.2025.

### 1. sastanak

- Datum: 15. listopada 2024.
- Prisustvovali: K.Bašić, L.Barišić, L.Lučin, D.Kovčo, I.Đurić, J.Herceg
- Teme sastanka:

Podjela uloga na projektu

Frontend: L.Lučin, I.Zečević. Backend: D.Kovčo, I.Đurić, L.Barišić  
Baze: K.Bašić, J.Herceg  
Dokumentacija: J.Herceg

Uspostavljanje komunikacije

Napravljen Discord kanal  
Napravljena WhatsApp grupa

### 2. sastanak

- Datum: 21. listopada 2024.
- Prisustvovali: K.Bašić, L.Barišić, I.Zečević., D.Kovčo, I.Đurić, J.Herceg
- Teme sastanka:

Uspostavljanje funkcionalnih i nefunkcionalnih zahtjeva

Analizirani zahtjevi i zapisani u README.md  
Podjela zadataka i odabir prve funkcionalnosti  
Prodaja karata odabrana kao prva funkcionalnost  
Frontend: login i početna stranica  
Backend: spajanje na bazu  
Baza: kreacija baze

### 3. sastanak

- Datum: 28. listopada 2024
- Prisustvovali: K.Bašić, L.Barišić, D.Kovčo, I.Đurić, J.Herceg
- Teme sastanka:

Revizija dosad napisanog koda

Podjela zadataka za daljnje ostvarivanje funkcionalnosti prodaje karata

Frontend: stranica za stavljanje karata na prodaju, stranica za kupnju  
Backend: ostvarivanje opcije kupi, spajanje s frontendom i do kraja spajanje s bazom  
Baza: izrada rel, er dijagrama, punjenje baze  
Dokumentacija: 2. i 4. točka wikija

### 4. sastanak

- Datum: 5. studenog 2024.
- Prisustvovali: K.Bašić, L.Barišić, D.Kovčo, I.Đurić, J.Herceg, I.Zečević
- Teme sastanka:

Revizija dosad napisanog koda

Podjela zadataka za ostvarivanje funkcionalnosti objave karata

Frontend: sign up stranica  
Backend: spajanje s frontendom do kraja, funkcionalnost objave karata  
Dokumentacija: 1. i 5. točka wikija  
Deployment

### 5. sastanak

- Datum: 11. studenog 2024.
- Prisustvovali: K.Bašić, L.Barišić, D.Kovčo, I.Đurić, J.Herceg, I.Zečević
- Teme sastanka:

Revizija dosad napisanog koda

Potrebno napraviti deployment i dijelove wikija

### 6. sastanak

- Datum: 9. prosinca 2024.
- Prisustvovali: K.Bašić, L.Barišić, D.Kovčo, I.Đurić, J.Herceg
- Teme sastanka:

Revizija dosad napisanog koda

Dodjela zadataka:

K. Bašić: Brisanje karata s mogućnošću vraćanja  
L. Barišić: Intuitivnije sučelje  
D. Kovčo: Spajanje s vremenskom prognozom  
I. Đurić: Kupnja ulaznica, ocjenjivanje korisnika  
J. Herceg: Označavanje karata sa sviđa mi se  
I. Zečević: Uređivanje izgleda ulaznica

### 7. sastanak

- Datum: 16. prosinca 2024.
- Prisustvovali: K.Bašić, L.Barišić, I.Đurić, J.Herceg, I.Zečević
- Teme sastanka:

Revizija dosad napisanog koda

Dodjela zadataka:

- K. Bašić: Funkcionalnost razmjena
- L. Barišić: Stranica za razmjenu, prikaz lanaca razmjene
- D. Kovčo: Spajanje sa servisom za izvođače
- I. Đurić: Funkcionalnost aukcije
- J. Herceg: Istekle ulaznice, notifikacije korisniku za zainteresirane događaje
- I. Zečević: Stranica za aukciju

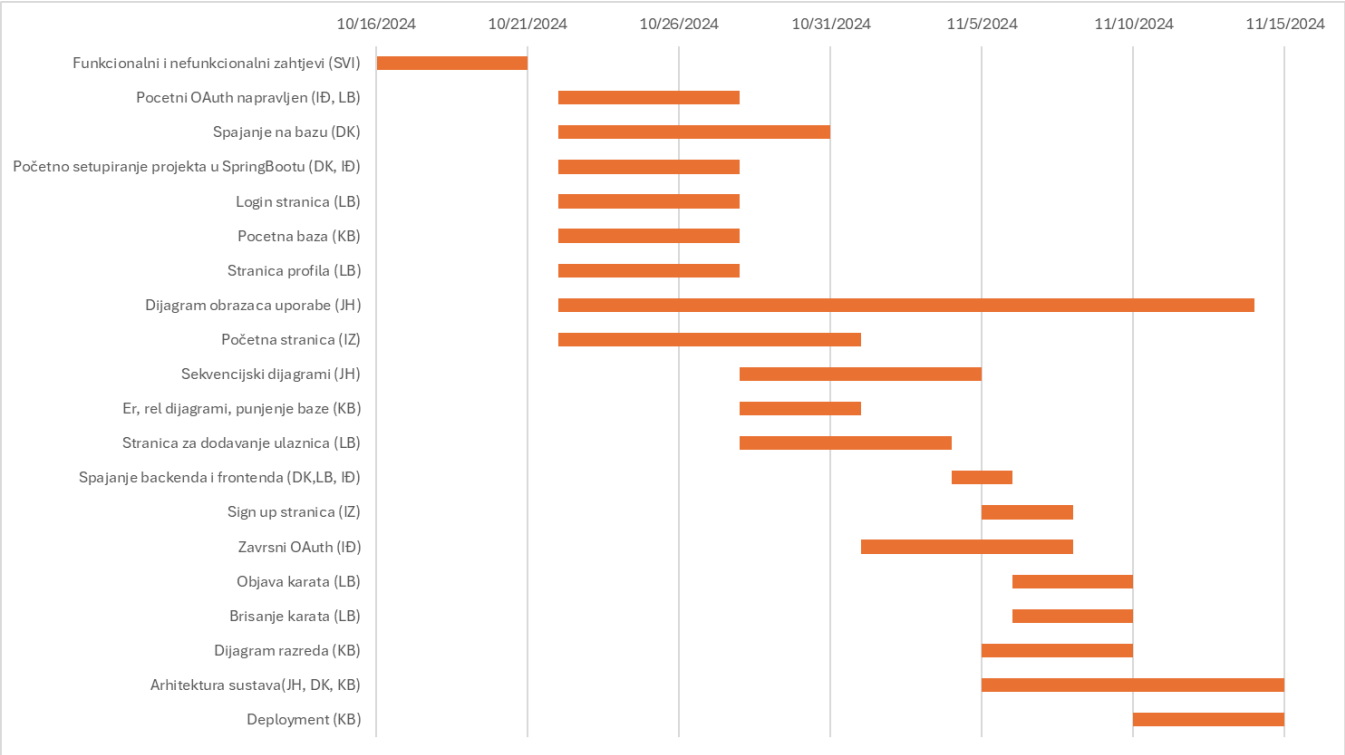
8. sastanak

- Datum: 7. siječnja 2025.
- Prisustvovali: K.Bašić, L.Barišić, I.Đurić, D.Kovčo, I.Zečević
- Teme sastanka:

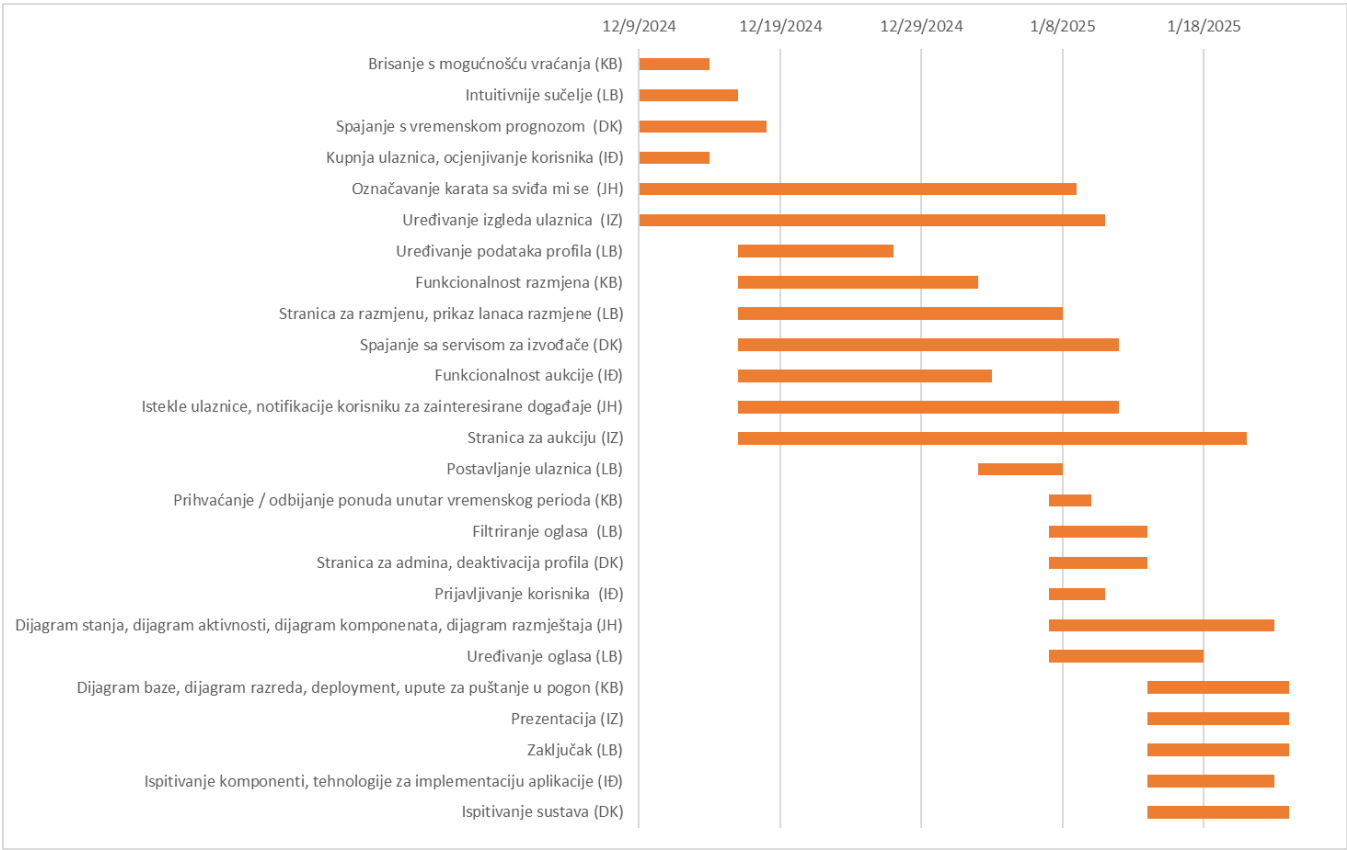
- Revizija dosad napisanog koda
- Dodjela zadataka:
- K. Bašić: Prihvaćanje / odbijanje ponuda unutar vremenskog perioda
- L. Barišić: Filtriranje oglasa
- D. Kovčo: Stranica za admina, deaktivacija profila
- I. Đurić: Prijavljivanje korisnika
- J. Herceg: Dijagram stanja, dijagram aktivnosti, dijagram komponenata, dijagram razmještaja
- I. Zečević: Uređivanje oglasa

Plan rada

1. ciklus:



2. ciklus:



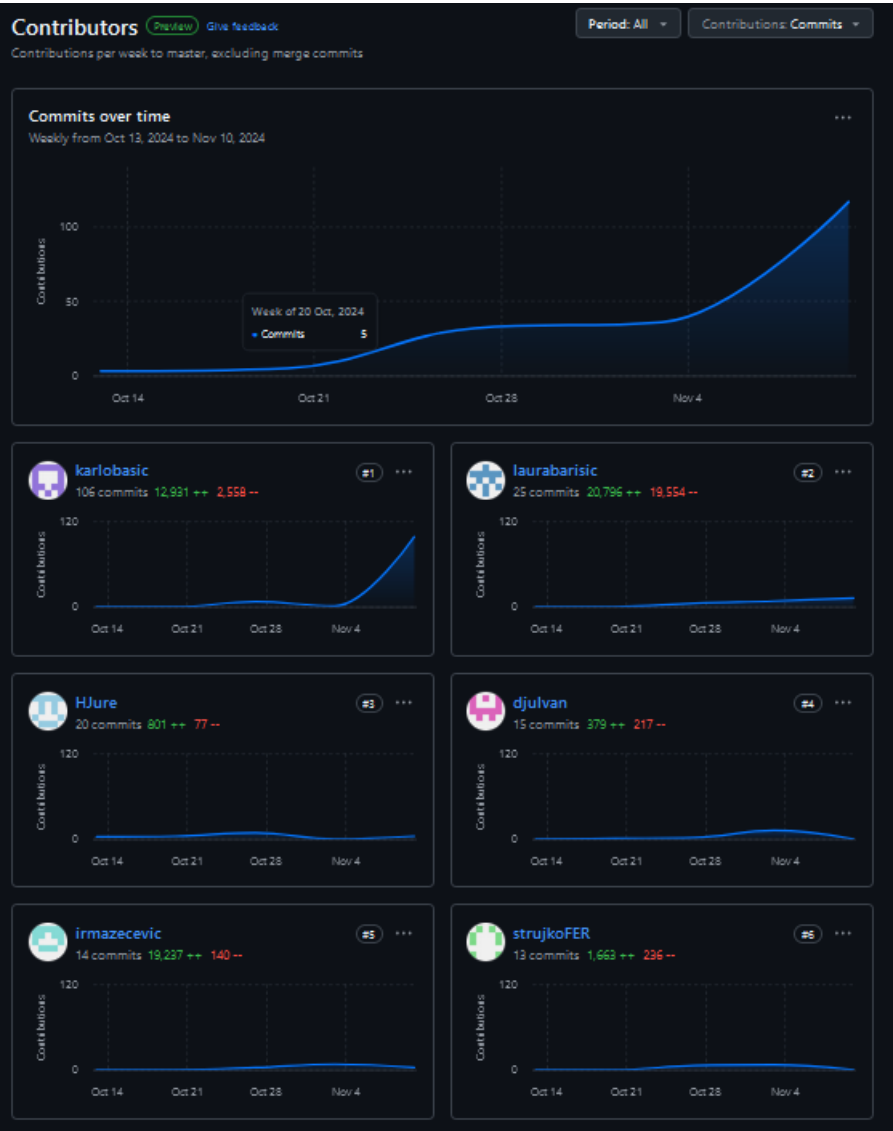
Tablica aktivnosti

Aktivnost	Karlo Bašić	Laura Barišić	Ivan Đurić	Jure Herceg	David Kovčo	Irma Zečević
Upravljanje projektom	15	8	8	7	7	5
Opis projektnog zadatka				2		
Funkcionalni zahtjevi				3		
Opis pojedinih obrazaca				4		
Dijagram obrazaca				3		
Sekvencijski dijagrami				2		
Opis ostalih zahtjeva				1		
Arhitektura i dizajn sustava	1			1	5	
Baza podataka	8			1		
Dijagram razreda	4					
Dijagram stanja				2		
Dijagram aktivnosti				2		
Dijagram komponenti				2		
Korištene tehnologije i alati			5	2		
Ispitivanje programskog rješenja			22		7	
Upute za puštanje u pogon	1					



Prikaz aktivnosti grupe	2			1		
Aktivnost	Karlo Bašić	Laura Barišić	Ivan Đurić	Jure Herceg	David Kovčo	Irma Zečević
Zaključak i budući rad		2				
Popis literature						
Spajanje s bazom podataka					20	
Izrada prezentacije						4
Frontend		70	15	3	40	65
Backend	30	20	55	15	30	1
Puštanje aplikacija u pogon	12			3		

## Dijagram pregleda promjena



## Ključni izazovi i rješenja

Glavni izazovi i rješenja:

- Problem s autentifikacijom putem GitHuba, autentifikacija putem GitHuba nije pružala korisnički email -> prelazak na autentifikaciju putem Googlea kako bi proces bio jednostavniji
- Problemi u implementaciju zbog nerazumijevanja koda drugih članova tima -> na timskim sastancima svaki član tima detaljno predstavlja svoj kod
- Problemi rukovanja s Githubom -> proučavanje dodatnih Github tutoriala

## Programsko inženjerstvo ak.god 2024./2025

# Razmijeni Ulaznice

---

Tim: <TG06.1>

Ime tima: TickEx

Nastavnik: Vlado Sruk

Asistent: Ivan Lovrić