

Programsko inženjerstvo

Ak. god. 2022./2023.

## Pub kvizovi

Dokumentacija, Rev. 2

Grupa: *RoyalStandard*

Voditelj: *Branimir Stanković*

Datum predaje: 13. 01. 2023.

Nastavnik: *Manuela Lukić*

# Sadržaj

<b>1 Dnevnik promjena dokumentacije</b>	<b>3</b>
<b>2 Opis projektnog zadatka</b>	<b>5</b>
<b>3 Specifikacija programske potpore</b>	<b>10</b>
3.1 Funkcionalni zahtjevi . . . . .	10
3.1.1 Obrasci uporabe . . . . .	12
3.1.2 Sekvencijski dijagrami . . . . .	25
3.2 Ostali zahtjevi . . . . .	28
<b>4 Arhitektura i dizajn sustava</b>	<b>29</b>
4.1 Baza podataka . . . . .	30
4.1.1 Opis tablica . . . . .	31
4.1.2 Dijagram baze podataka . . . . .	34
4.2 Dijagram razreda . . . . .	36
4.3 Dijagram stanja . . . . .	40
4.4 Dijagram aktivnosti . . . . .	42
4.5 Dijagram komponenti . . . . .	44
<b>5 Implementacija i korisničko sučelje</b>	<b>46</b>
5.1 Korištene tehnologije i alati . . . . .	46
5.2 Ispitivanje programskog rješenja . . . . .	48
5.2.1 Ispitivanje komponenti . . . . .	48
5.2.2 Ispitivanje sustava . . . . .	55
5.3 Dijagram razmještaja . . . . .	60
5.4 Upute za puštanje u pogon . . . . .	61
<b>6 Zaključak i budući rad</b>	<b>70</b>
<b>Popis literature</b>	<b>72</b>
<b>Indeks slika i dijagonama</b>	<b>74</b>

**Dodatak: Prikaz aktivnosti grupe**

**75**

# 1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Šetka	28.10.2022.
0.2	Dodani opis projektnog zadatka i funkcionalni zahtjevi. Ažuriran dnevnik sastajanja.	Šetka	31.10.2022.
0.3	Dodana arhitektura	Panić	03.11.2022.
0.5	Dodani svi obrasci uporabe	Pejić, Samardžić, Šarić, Šetka	05.11.2022.
0.6	Dodani svi <i>Use Case</i> dijagrami i svi sekvencijski dijagrami	Pejić, Samardžić, Šarić, Šetka	10.11.2022.
0.7	Dodan opis baze podataka, opis tablica i dijagram	Kompar	10.11.2022.
0.8	Dodani dijagrami razreda	Šetka	17.11.2022.
<b>1.0</b>	Verzija samo s bitnim dijelovima za 1. ciklus	Šetka	18.11.2022.
1.1	Dodan opis korištenih tehnologija i alata	Šarić	07.01.2023.
1.2	Dodani dijagram stanja, aktivnosti, komponenti i razmještaja Ažuriran dnevnik sastajanja.	Šetka	07.01.2023.
1.3	Dodane upute za puštanje u pogon	Pejić	10.01.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

<b>Rev.</b>	<b>Opis promjene/dodataka</b>	<b>Autori</b>	<b>Datum</b>
1.4	Dodano ispitivanje programskog rješenja Ispitivanje sustava	Panić, Samardžić	10.01.2023.
1.5	Dodani dijagrami razreda	Šetka	10.01.2023.
1.6	Dodan zaključak i budući rad	Kompar	10.01.2023.
1.7	Dodano ispitivanje programskog rješenja Ispitivanje komponenti	Stanković, Šarić	13.01.2023.
2.0	Konačni tekst predloška dokumentacije	Šetka	13.01.2023.

## 2. Opis projektnog zadatka

U velikoj konkurenciji različitih pub kvizova kojoj i sami svjedočimo, želja je svakog sastavljača privući što veći broj ekipa i učiniti kviz što je moguće zanimljivijim. Svaki pojedini igrač ili tim želi pronaći pub kviz na kojem mu najbolje odgovara koncept pitanja i gdje je atmosfera jednostavno priča za sebe. Kako bi se povezali sastavljači, kvizaši, ali i oni koji to tek žele postati ili jednostavno probati nešto novo, cilj je razviti aplikaciju koja će znatno olakšati taj proces i ponuditi dodatne funkcionalnosti.

Glavni je cilj ove aplikacije omogućiti sastavljačima kvizova objavu nadolazećih pub kvizova koje organiziraju i osigurati igračima da vide koliko se raznolikih kvizova održava u njihovoј blizini, kao i kada se održavaju oni za koje su najviše zainteresirani. S obzirom na to da često postoje osobe koje žele igrati neki pub kviz, ali nažalost nemaju ekipu, aplikacija će riješiti i taj problem. Igrač koji nema ekipu, a ipak želi sudjelovati na pub kvizu, moći će u aplikaciji pronaći tim kojemu bi najviše odgovarao kao suigrač i tako zaigrati kviz.

Neregistrirani korisnik imat će mogućnost isključivo pregleda pub kvizova, ali s ograničenim uvidom u detalje objavljenih događaja. Moći će vidjeti naziv kviza, ime kafića, vrijeme održavanja te vrstu kviza, ali za sve ostale pojedinosti prvo se mora registrirati u sustav.

Registrirani korisnik može biti ili sastavljač ili igrač kviza. Svaka od ovih uloga ima drugačiji skup mogućnosti u aplikaciji. Iznimno, ako je jedan korisnik sastavljač pub kvizova, ali isto tako i sudjeluje na nekim drugima kao igrač, on može imati obje uloge u aplikaciji. Prilikom registracije korisnik najprije odabire ulogu na temelju koje ispunjava prilagođenu formu. Ako se registrira kao igrač ili istovremeno kao igrač i sastavljač onda unosi sljedeće podatke:

- *Ime*
- *Prezime*
- *Nadimak*

- *Email adresa*
- *Lozinka*
- *Područja znanja za kviz*
- *Imaš ekipu?*
  - [DA] **Naziv ekipe**
  - [NE]
- *Slika*
- *Broj telefona*

Pri čemu su podebljani podaci obvezni za unijeti. Ako ipak odabere samo ulogu sastavljača, onda unosi ove podatke:

- *Ime*
- *Prezime*
- *Nadimak*
- *Email adresa*
- *Lozinka*
- *Slika*
- *Broj telefona*

U slučaju da korisnik ima svoju pub kviz ekipu, pri registraciji će odabrati vrijednost DA za polje Imaš ekipu? te će morati obvezno unijeti naziv svoje ekipe. Inače, ako nema ekipu odabrat će opciju NE.

Sastavljač pub kvizova, ujedno je i organizator događaja (pub kviza) te kroz aplikaciju ima mogućnost objaviti nadolazeći događaj tako da prilikom objave unese sve potrebne podatke bitne za kviz:

- *Naziv kviza*
- *Kratki opis*
- *Ime kafića*
- *Vrijeme održavanja*
- *Lokacija*
- *Maksimalan broj ekipa*
- *Iznos kotizacije*
- *Nazivi nagrada*
- *Vrsta kviza*
- *Informacije o sastavljaču*

Pri kreiranju nove objave događaja potrebno je paziti da sastavljač ne smije objaviti više događaja koji su u isto vrijeme (s preklapanjem termina). Ako korisnik sastavljač ima i ulogu igrača, treba dodatno paziti da u isto vrijeme događaja koji je korisnik objavio, isti korisnik nije prijavljen kao igrač na neki od kvizova. Sastavljač može vidjeti sve objave pub kvizova u aplikaciji na pregledu "Svi pub kvizovi", a svoje objave vidi na pregledu "Moji pub kvizovi". Uz to, sastavljač može pregledati i uređivati svoj profil na pregledu "Moj profil" s podacima unesenim pri registraciji.

Igrač pub kvizova ima dvije glavne mogućnosti u aplikaciji, a to su pronalazak ekipe (samo za one igrače koji ju nemaju) i prijava svoje ekipe na neki od objavljenih kvizova. Ako pri registraciji korisnik navede da nema vlastitu ekipu, onda u aplikaciji ima mogućnost pronaći ju tako da odabere opciju „Pronađi tim“ i aplikacija će ga spojiti s ekipom kojoj najviše odgovara (većina područja znanja u kojima je dobar nedostaju ekipi s kojom će ga aplikacija spojiti). Kada je korisniku pronađen tim, dolazi mu obavijest s nazivom ekipe i brojem članova. Također, svakom igraču te ekipe dolazi obavijest da su dobili novog igrača i popis njegovih boljih područja znanja s kojima će doprinijeti timu. Igrači koji imaju svoju ekipu mogu je napustiti odabirom opcije "Napusti tim". Igrač može vidjeti sve objave pub kvizova u aplikaciji na pregledu "Svi pub kvizovi", a događaje na koje je prijavljen može vidjeti na pregledu "Moji pub kvizovi". Također, igrači mogu prijaviti svoju ekipu na kviz, tako da 1 igrač iz ekipe prijavljuje cijelu ekipu na određeni kviz. Pri tome treba paziti da jedna ekipa ne bude prijavljena na više različitih kvizova u isto vrijeme. Kada igrač uspješno prijavi svoju ekipu na kviz, njemu i njegovim suigračima dolazi obavijest da su prijavljeni na novi kviz i na objavi za taj događaj broj slobodnih mjesta se ažurira (jedno slobodno mjesto manje). Uz to, igrač može pregledati i uređivati svoj profil na pregledu "Moj profil" s podacima unesenim pri registraciji.

Administrator sustava uz ovlasti svih korisnika ima i neke dodatne mogućnosti u aplikaciji, a to su: blokiranje korisnika koji krše pravila sustava (na primjer imaju nepristojan nadimak ili naziv ekipe), odobravanje ili zabranjivanje objava za pub kvizove koje prethodno kreiraju sastavljači te mogućnost naknadnog brisanja istih događaja. Također, administrator može dodati administratorska prava drugim korisnicima.

Aplikacija za sve prijavljene korisnike treba omogućiti pretraživanje objavljenih događaja te filtriranje prema određenim parametrima. Funkcionalnost pretraživanja ostvarena je okvirom za pretraživanje (eng. search bar), pomoću kojeg prijavljeni korisnik može pronaći određeni događaj unoseći njegov naziv ili ime kafića u kojem se kviz održava. Filtriranje se dodatno ostvaraju na način da korisnik odabire željene vrijednosti ili raspon vrijednosti za svoju udaljenost od kafića gdje se kviz održava, iznos kotizacije, vrstu kviza ili slično.

Također, svi igrači pub kvizova imaju mogućnost uvida u statističke detalje na pregledu "Statistika", u kojem mogu vidjeti i usporediti prikazane podatke u obliku grafa ili tablice, a bilježi se prosječan broj ekipa po kvizu za nekog sastavljača, broj održanih kvizova u posljednjih tjedan dana te prosječna popunjenošć kviza za pojedinu vrstu kviza.

Pretpostavka je da je 1 igrač fiksno u 1 ekipi te da ekipa ima maksimalno 5 članova.

Ovaj projekt potencijalno bi mogao koristiti za organizaciju i prijave na sve postojeće pub kvizove u Hrvatskoj, a posebna prednost je što bi sve objave kvizova bile razvrstane, pregledne i skupljene na jednom mjestu kako bi korisnici mogli brzo i jednostavno dobiti bitne informacije o pojedinom događaju. Zainteresirani skup korisnika činili bi svi sastavljači i organizatori kvizova, članovi ekipa te igrači koji želeći ići na neki kviz, ali još nemaju suigrače. Opseg projektnog zadatka ponajprije obuhvaća funkcionalnosti objave kvizova, prijave i pronalazak ekipe, a moguće je i dobiti dodatne informacije o pojedinom kvizu ili sastavljaču. Također, projektni zadatak mogao bi se nadograditi tako da sastavljači mogu dodati slike, poredak ekipa i nekoliko pitanja sa svojih bivših događaja te se na taj način predstaviti svim igračima koji pregledavaju njihove profile. Dodatno proširenje bilo bi uvođenje recenzija za kvizove i sastavljače te komentari i osvrti o pojedinim događajima. Iako ne postoje programska rješenja koja nude sve mogućnosti kao ova aplikacija, postoje različite web stranice gdje se može pronaći popis kvizova za određeno mjesto i datum te osnovne informacije o kreiranim događajima. Kao jednu od najpopularnijih navodimo <https://www.pubquizzers.com/index.php>.

The screenshot shows the homepage of the Pub Quizzers website. At the top, there's a navigation bar with days of the week: Monday (red), Tuesday (orange), Wednesday (yellow), Thursday (green), Friday (light blue), Saturday (purple), and Sunday (dark blue). Below the navigation is the website's logo, "PUB QUIZZERS", and a search bar asking "Find a pub quiz near you, now." A search icon is located to the right of the search bar.

The main content area has several sections:

- Welcome To Pub Quizzers**: A sub-section of the search bar with the same search query.
- > Add A Quiz To The Most Visited Pub Quiz Website In The World**: A button with a yellow background and black text.
- 27th October 2022**: A news item stating there are now 876 pub quizzes live in the database.
- Select a city or place from this list to view quizzes in that area:** A dropdown menu labeled "Select Place...".
- Latest Pub Quizzes**: A section showing ten recent quizzes with small thumbnail images and details.

Pub Name	Day	Time	Address	Action
The Foxley Hatch	Wed	19:30	C2.00 Entry Purley, CR8 2LE	More Info >>
The Stonhouse	Mon	20:00	C2.50 Entry London, SW14 6BQ	More Info >>
The Harringay Arms	Sun	19:30	C2.50 Entry London, N15 9JH	More Info >>
Gorringe Park	Tue	19:30	C2.50 Entry London, SW17 9JR	More Info >>
Clifton Arms	Tue	20:00	C2.50 Entry London, SE25 6NU	More Info >>
- Add A Pub Quiz**: A section with a "Select Place..." dropdown and a "Search" button.
- Quiz Of The Moment**: A section featuring a thumbnail of a pub interior and the text "Every Tuesday, 8.00pm £2 Entry The Brewery Tap Westgate, Peterborough".
- Quiz Question #8**: A section asking "What is the capital of Estonia?" with five multiple-choice options: (1) Tallinn, (2) Tartu, (3) Narva, (4) Viljandi, (5) Rakvere.

Slika 2.1: Primjer sličnog rješenja

# 3. Specifikacija programske potpore

## 3.1 Funkcionalni zahtjevi

Dionici:

1. Naručitelj
2. Korisnik aplikacije
  - (a) Igrač kviza
  - (b) Sastavljač kviza
3. Administratori
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
  - (a) registrirati se u sustav kao igrač, sastavljač ili oboje istovremeno
  - (b) pregledati osnovne informacije o događaju (pub kvizu)
2. Igrač kviza (inicijator) može:
  - (a) prijaviti se u sustav
  - (b) vidjeti sve objavljene događaje na pregledu "Svi pub kvizovi"
  - (c) vidjeti događaje na koje je prijavljen na pregledu "Moji pub kvizovi"
  - (d) pronaći ekipu za kviz
  - (e) napustiti svoju ekipu
  - (f) prijaviti svoju ekipu na kviz
  - (g) pregledati svoj profil
  - (h) uređivati svoj profil
  - (i) pregledati profil sastavljača
  - (j) pretraživati i filtrirati događaje
  - (k) vidjeti statističke podatke
3. Sastavljač kviza (inicijator) može:

- (a) prijaviti se u sustav
- (b) objaviti novi događaj (pub kviz)
- (c) vidjeti sve objavljene događaje na pregledu "Svi pub kvizovi"
- (d) vidjeti svoje objavljene događaje na pregledu "Moji pub kvizovi"
- (e) pregledati svoj profil
- (f) uređivati svoj profil
- (g) pretraživati i filtrirati događaje
- (h) vidjeti statističke podatke

4. Administrator (inicijator) može:

- (a) sve isto što mogu i ostali korisnici sustava
- (b) blokirati korisnike koji krše pravila sustava
- (c) odobriti ili zabraniti objavu koju je kreirao sastavljač
- (d) brisati objavljene događaje
- (e) davati administratorska prava drugim korisnicima

5. Baza podataka (sudionik) može:

- (a) pohranjivati sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjivati sve podatke o događajima (kvizovima)

### 3.1.1 Obrasci uporabe

#### Opis obrazaca uporabe

##### UC01 - Pregled naslovnice

- **Glavni sudionik:** Neregistrirani korisnik, registrirani korisnik
- **Cilj:** Pregledati naslovnicu aplikacije
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik nije prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik pregledava opis i sve objavljene događaje na naslovnici
- **Opis mogućih odstupanja:**
  - 1.a Ne postoji ni jedan objavljeni događaj
    1. Korisniku se prikazuje samo opis aplikacije

##### UC02 - Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za registraciju
  2. Korisnik unosi tražene podatke i potvrđuje unos
  3. Podaci se spremaju u bazu podataka
  4. Korisnika se preusmjerava na početnu stranicu
- **Opis mogućih odstupanja:**
  - 2.a Korisnik unosi neispravne podatke (zauzeti ili neispravni e-mail ili nadimak, unos podataka u nedopuštenom formatu)
    1. Sustav upozorava korisnika na neispravnost unesenih podataka i vraća ga na stranicu za registraciju.
    2. Korisnik mijenja podatke ili odustaje od registracije

##### UC03 - Prijava u sustav

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pristupiti korisničkom sučelju
- **Sudionici:** Baza podataka

- **Preduvjet:** Korisnik je registriran u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik unosi nadimak i lozinku te potvrđuje unos
  2. Sustav provjerava ispravnost unesenih podataka
  3. Korisnik dobiva pristup korisničkom sučelju
- **Opis mogućih odstupanja:**
  - 2.a Uneseni su neispravni nadimak i/ili lozinka
    1. Sustav obaveštava korisnika da su uneseni pogrešni podaci i vraća ga na stranicu za prijavu

#### UC04 - Pregled podataka korisničkog profila

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledati podatke korisničkog profila
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju „Moj profil“
  2. Korisnik pregledava podatke profila

#### UC05 - Uređivanje podataka korisničkog profila

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Urediti podatke korisničkog profila
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju „Moj profil“
  2. Korisnik odabire opciju za promjenu podataka
  3. Korisnik mijenja odabrane podatke
  4. Korisnik sprema promjene
  5. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
  - 4.a Korisnik ne spremi promjene
    1. Promjene se odbacuju

#### UC06 - Kreiranje nadolazećih događaja

- **Glavni sudionik:** Registrirani korisnik (Sastavljač kviza)

- **Cilj:** Kreirati nove događaje
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za kreiranje novog događaja
  2. Korisnik popunjava potrebne podatke
  3. Korisnik spremi promjene
  4. Podaci se spremaju u bazu podataka
  5. Novi kviz čeka odobrenje administratora
- **Opis mogućih odstupanja:**
  - 3.a Korisnik je već kreirao događaj u isto vrijeme
    1. Sustav vraća poruku o neispravno unesenom podatku
    2. Korisnik unosi ispravno vrijeme ili odustaje od kreiranja događaja

#### UC07 - Odobravanje kreiranih događaja

- **Glavni sudionik:** Administrator
- **Cilj:** Odobriti nove događaje
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Administratoru je dostupan popis novih događaja koji čekaju odobrenje
  2. Administrator odabire događaj i odobrava ga
  3. Baza podataka se ažurira
  4. Odobreni događaj se objavljuje

#### UC08 - Zabranja kreiranih događaja

- **Glavni sudionik:** Administrator
- **Cilj:** Zabraniti nove događaje
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Administratoru je dostupan popis novih događaja koji čekaju odobrenje
  2. Administrator odabire događaj i zabranjuje ga
  3. Događaj se briše iz baze podataka

#### UC09 - Pregled svih objavljenih događaja

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledati sve događaje koji su objavljeni
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju „Svi pub kvizovi“
  2. Korisniku se prikazuju svi objavljeni događaji
- **Opis mogućih odstupanja:**
  - 2.a Ne postoji ni jedan objavljeni događaj
    1. Korisniku se prikazuje odgovarajuća poruka

#### **UC10 - Pregled svojih objavljenih događaja**

- **Glavni sudionik:** Registrirani korisnik (Sastavljač kviza)
- **Cilj:** Pregledati svoje objavljene događaje
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju „Moji pub kvizovi“
  2. Korisniku se prikazuju njegovi objavljeni događaji
- **Opis mogućih odstupanja:**
  - 2.a Ne postoji ni jedan objavljeni događaj koji je korisnik kreirao
    1. Korisniku se prikazuje odgovarajuća poruka

#### **UC11 - Pregled prijava za objavljene događaje**

- **Glavni sudionik:** Registrirani korisnik (Igrač kviza)
- **Cilj:** Pregledati događaje na koje je korisnik prijavljen
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju "Moji pub kvizovi"
  2. Korisniku se prikazuju svi događaji na koje je prijavljena njegova ekipa
- **Opis mogućih odstupanja:**
  - 2.a Korisnikova ekipa nije prijavljena ni na jedan događaj
    1. Korisniku se prikazuje odgovarajuća poruka

#### **UC12 - Pregled detalja objavljenog događaja**

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledati detalje o objavljenom događaju
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju "Svi pub kvizovi"
  2. Korisnik odabire pregled detalja određenog događaja
  3. Korisniku se prikazuju detalji odabranog događaja
- **Opis mogućih odstupanja:**
  - 1.a Ne postoji ni jedan objavljeni događaj
    1. Korisniku se prikazuje odgovarajuća poruka

#### UC13 - Pretraživanje objavljenih događaja

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Na lakši način pronaći željeni događaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik na pregledu objavljenih događaja unosi željene podatke za pretraživanje te potvrđuje unos
  2. Korisniku se prikazuju događaji koji odgovaraju podacima unesenim za pretragu
- **Opis mogućih odstupanja:**
  - 2.a Ni jedan događaj ne odgovara pretraživanim podacima
    1. Korisniku se prikazuje odgovarajuća poruka

#### UC14 - Filtriranje objavljenih događaja

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pronaći događaj koji najviše odgovara korisniku
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik na pregledu objavljenih događaja odabire željene filtre za pretraživanje specifičnih događaja te potvrđuje unos
  2. Korisniku se prikazuju događaji koji odgovaraju odabranim filtrima
- **Opis mogućih odstupanja:**

- 2.a Ni jedan događaj ne odgovara filtriranim podacima
1. Korisniku se prikazuje odgovarajuća poruka

### UC15 - Pronalazak ekipe za kviz

- **Glavni sudionik:** Registrirani korisnik (Igrač kviza)
- **Cilj:** Pronaći ekipu za sudjelovanje na događajima
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za pronalazak ekipe
  2. Sustav pronađe ekipu najbolje odgovara korisniku
  3. Sustav šalje obavijest korisniku o ekipi u koju je dodan
  4. Sustav šalje obavijest o korisniku članovima ekipe u koju je dodan
- **Opis mogućih odstupanja:**
  - 2.a U sustavu ne postoji ni jedna ekipa
    1. Korisniku se prikazuje odgovarajuća poruka
  - 2.b Sve ekipe u sustavu imaju maksimalan broj članova
    1. Korisniku se prikazuje odgovarajuća poruka

### UC16 - Pregled obavijesti o pronađenoj ekipi

- **Glavni sudionik:** Registrirani korisnik (Igrač kviza)
- **Cilj:** Pregledati obavijest o pronađasku ekipe
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav, prethodno je odabrao opciju za pronađak ekipe i dodan je u neku ekipu
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za pregled obavijesti
  2. Korisnik pregledava obavijest o pronađenoj ekipi

### UC17 - Pregled obavijesti o novom članu ekipe

- **Glavni sudionik:** Registrirani korisnik (Igrač kviza)
- **Cilj:** Pregledati obavijest o dodavanju novog člana u ekipu
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i pripada ekipi u koju je dodan neki novi član
- **Opis osnovnog tijeka:**

1. Korisnik odabire opciju za pregled obavijesti
2. Korisnik pregledava obavijest o novom članu ekipe

### **UC18 - Napuštanje ekipe**

- **Glavni sudionik:** Registrirani korisnik (Igrač kviza)
- **Cilj:** Napustiti trenutnu ekipu
- **Sudionici:** Baza podataka
- **Preduvjet:** Igrač je prijavljen u sustav i trenutno se nalazi u nekoj ekipi
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju "Moj profil"
  2. Korisnik odabire opciju napuštanja trenutne ekipe
  3. Sustav izbacuje korisnika iz trenutne ekipe i šalje obavijest ostalim članovima da je korisnik napustio ekipu
  4. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
  - 3.a Korisnik je prije odabira opcije napuštanja ekipe bio jedini njezin član
    1. Sustav nikome ne šalje obavijest o napuštanju

### **UC19 - Pregled obavijesti o napuštanju člana ekipe**

- **Glavni sudionik:** Registrirani korisnik (Igrač kviza)
- **Cilj:** Pregledati obavijest o napuštanju člana ekipe
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za pregled obavijesti
  2. Korisnik pregledava obavijest o napuštanju člana ekipe

### **UC20 - Prijava ekipe na događaj**

- **Glavni sudionik:** Registrirani korisnik (Igrač kviza)
- **Cilj:** Prijaviti ekipu na događaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik pregledava sve objavljene događaje
  2. Korisnik odabire događaj na koji želi prijaviti ekipu
  3. Korisnik prijavljuje ekipu na događaj

4. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
    - 2.a Ekipa je već prijavljena na neki događaj u isto vrijeme
      1. Korisniku se prikazuje odgovarajuća poruka i odbija se prijava ekipe za taj događaj

### UC21 - Pregled obavijesti o prijavi na događaj

- **Glavni sudionik:** Registrirani korisnik (Igrač kviza)
- **Cilj:** Pregledati obavijest o prijavi na događaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za pregled obavijesti
  2. Korisnik pregledava obavijest o prijavi na događaj zajedno s informacijama događaja

### UC22 - Blokiranje korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Blokirati korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Administrator pregledava listu svih korisnika
  2. Administrator odabire opciju blokiranja korisnika
  3. Administrator pronađe željenog korisnika
  4. Administrator blokira korisnika
  5. Baza podataka se ažurira

### UC23 - Brisanje objavljenih događaja

- **Glavni sudionik:** Administrator
- **Cilj:** Obrisati objavljeni događaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Administrator pregledava sve objavljene događaje
  2. Administrator odabire opciju brisanja događaja

3. Administrator odabire događaj koji želi obrisati
  4. Administrator briše događaj
  5. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
    - 1.a Ne postoji ni jedan objavljeni događaj
      1. Administratoru se prikazuje odgovarajuća poruka

#### UC24 - Dodjela administratorskih prava korisniku

- **Glavni sudionik:** Administrator
- **Cilj:** Dodijeliti administratorska prava korisniku
- **Sudionici:** Baza podataka
- **Preduvjet:** Administrator je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Administrator pregledava listu svih korisnika
  2. Administrator bira korisnika kojem želi dodijeliti administratorska prava
  3. Administrator dodjeljuje administratorska prava korisniku
  4. Baza podataka se ažurira

#### UC25 - Pregled profila sastavljača

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Pregledati podatke profila sastavljača
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik pregledava sve objavljene događaje
  2. Korisnik odabire željeni događaj
  3. Korisnik pregledava detalje o odabranom događaju
  4. Korisnik odabire poveznicu na profil sastavljača
  5. Korisnik pregledava podatke profila sastavljača
- **Opis mogućih odstupanja:**
  - 1.a Ne postoji ni jedan objavljeni događaj
    1. Administratoru se prikazuje odgovarajuća poruka

#### UC26 - Pregled statistike

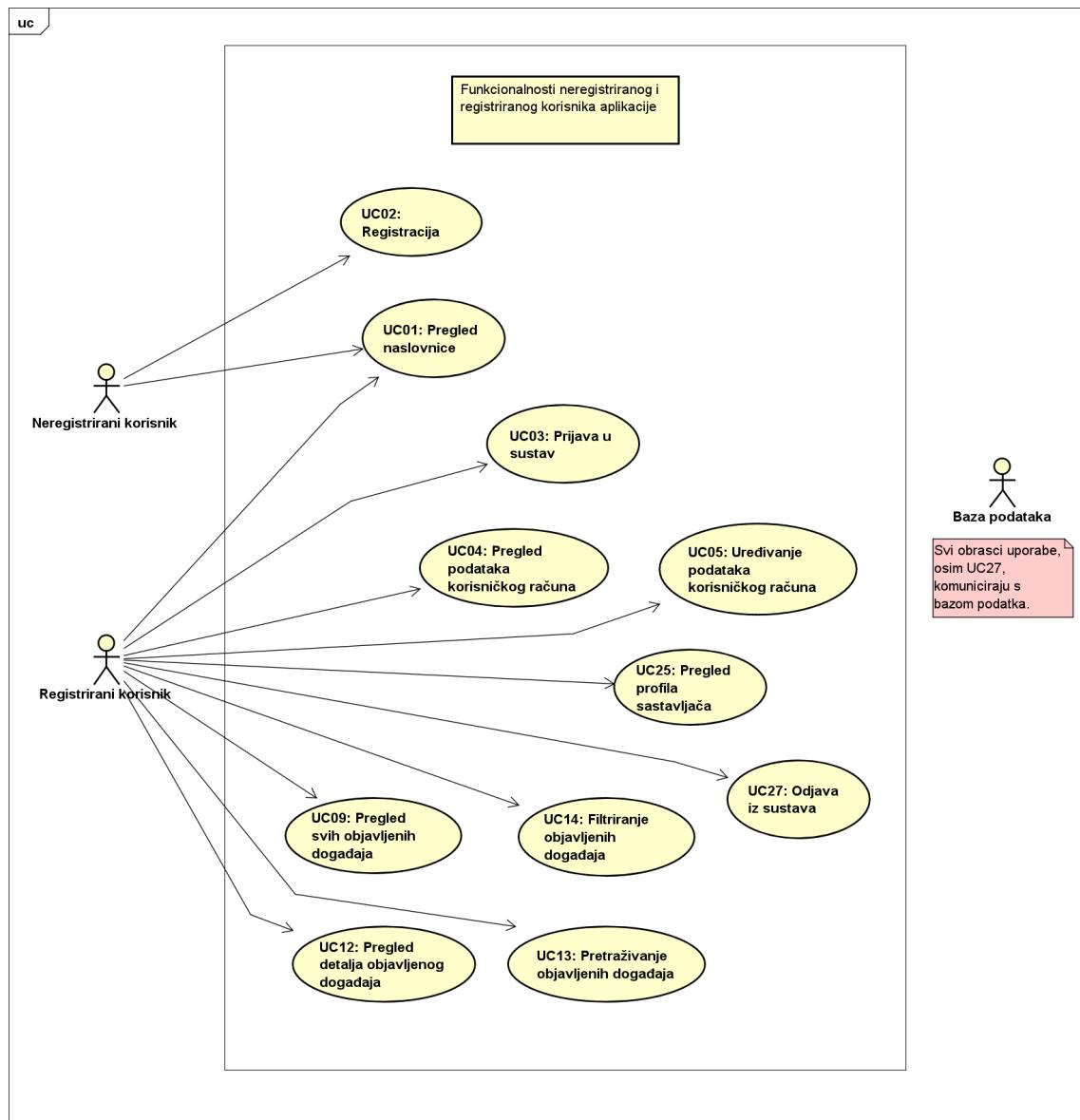
- **Glavni sudionik:** Registrirani korisnik (Igrač kviza)
- **Cilj:** Pregledati statistiku

- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire pregled „Statistika“
  2. Korisnik pregledava prikazane statističke podatke
- **Opis mogućih odstupanja:**
  - 2.a Ne postoje odgovarajući podaci za prikaz
    1. Korisniku se prikazuje odgovarajuća poruka

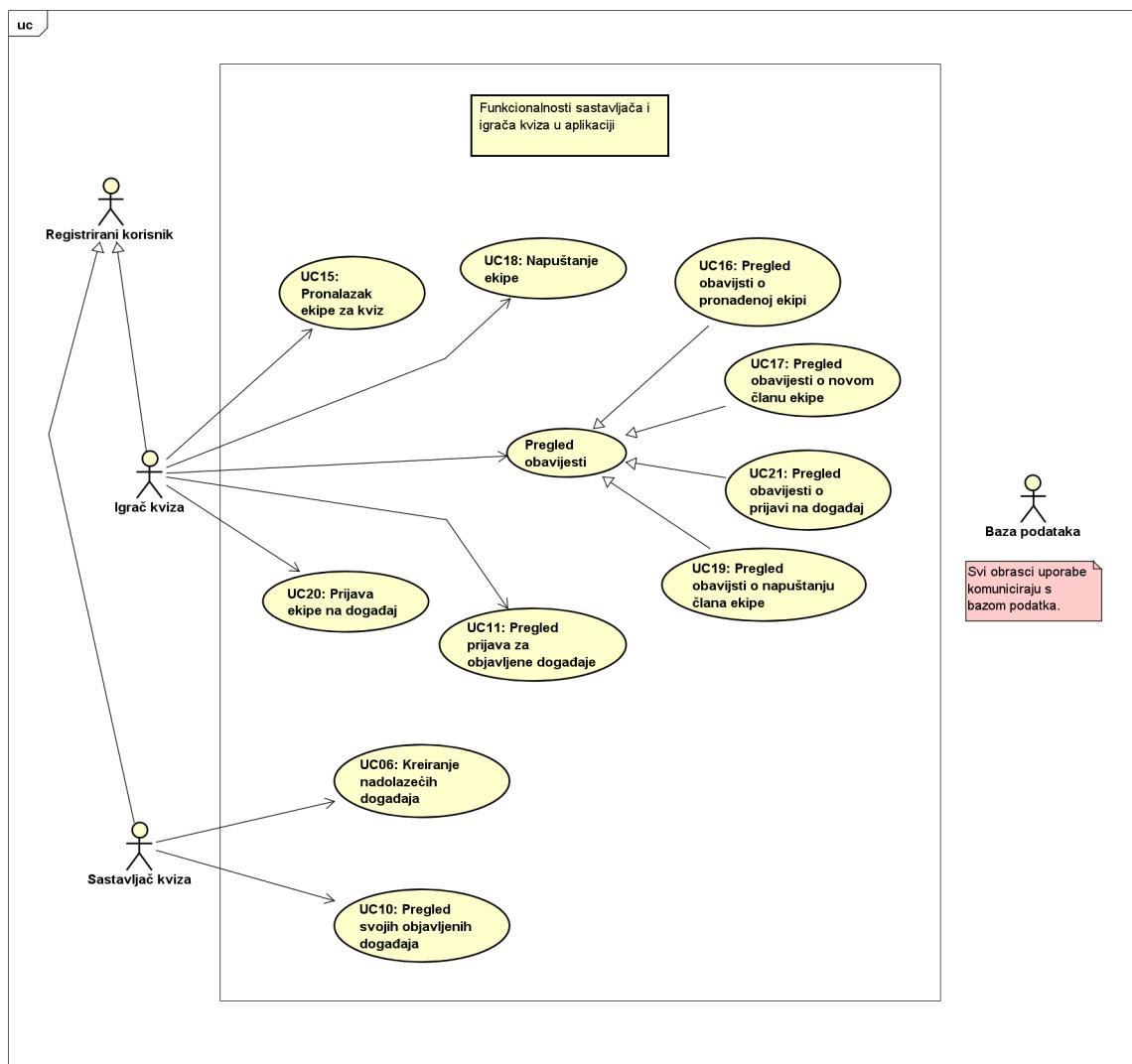
#### UC27 - Odjava iz sustava

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Odjaviti se iz sustava
- **Sudionici:** -
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik odabire opciju za odjavu iz sustava
  2. Korisnika se preusmjerava na stranicu za prijavu

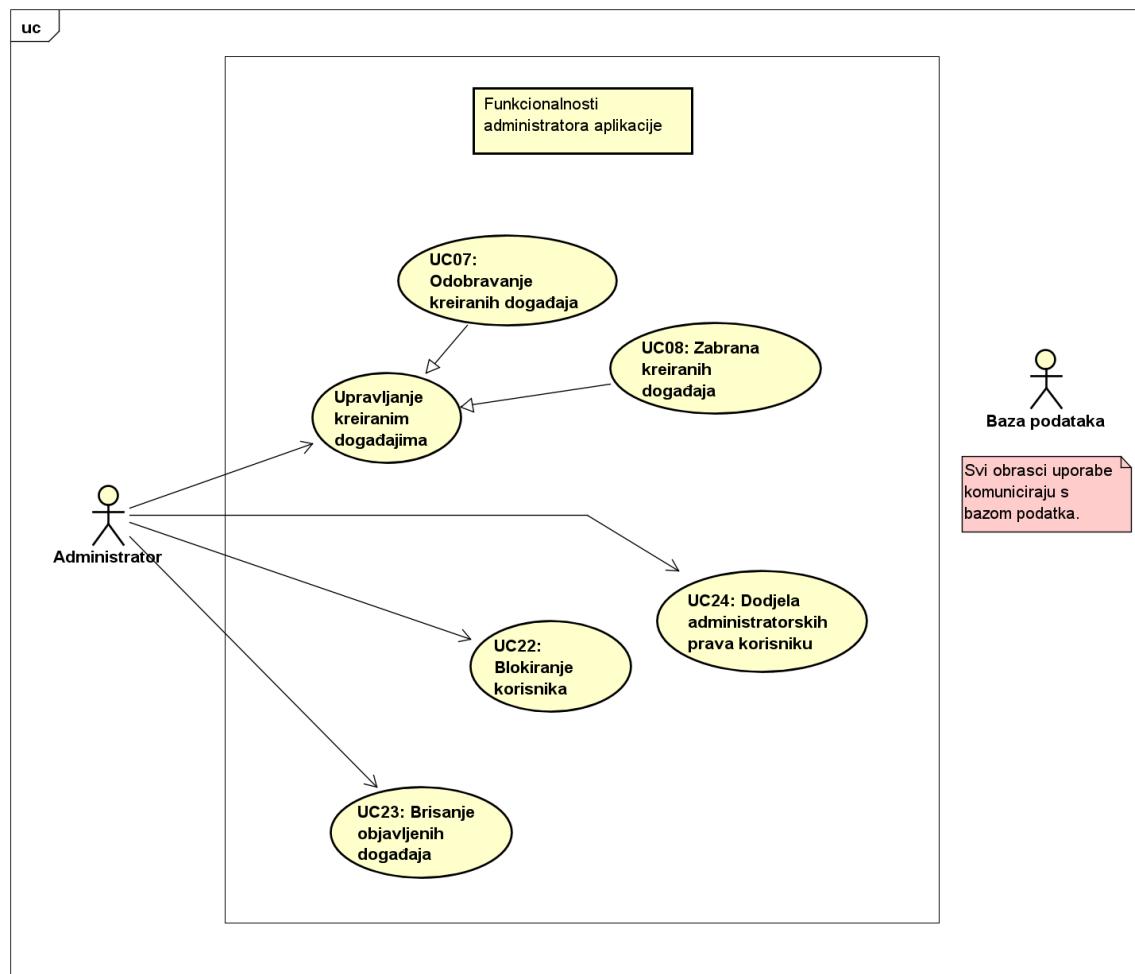
## Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrazaca uporabe, funkcionalnosti neregistriranog i registriranog korisnika



Slika 3.2: Dijagram obrazaca uporabe, funkcionalnosti igrača i sastavljača kviza

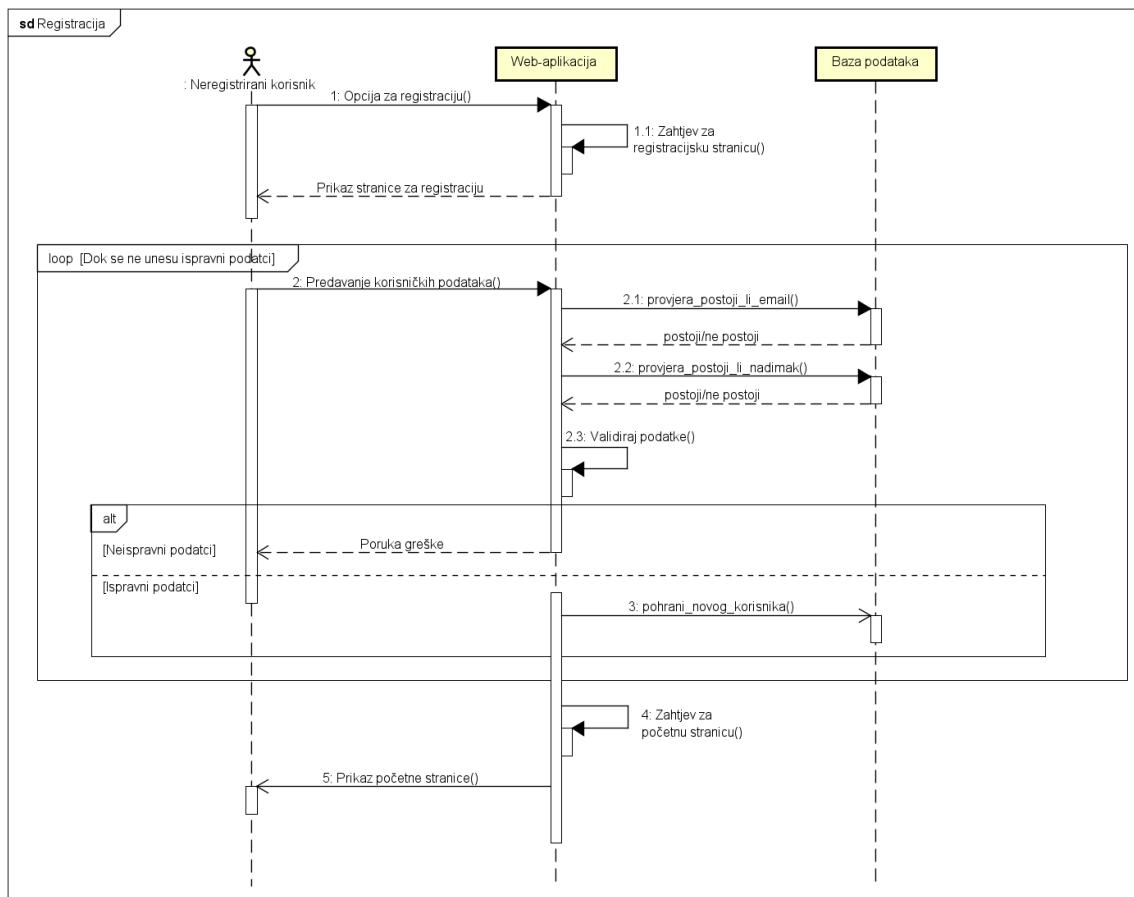


Slika 3.3: Dijagram obrazaca uporabe, funkcionalnosti administratora

### 3.1.2 Sekvencijski dijagrami

#### Obrazac uporabe UC02 - Registracija

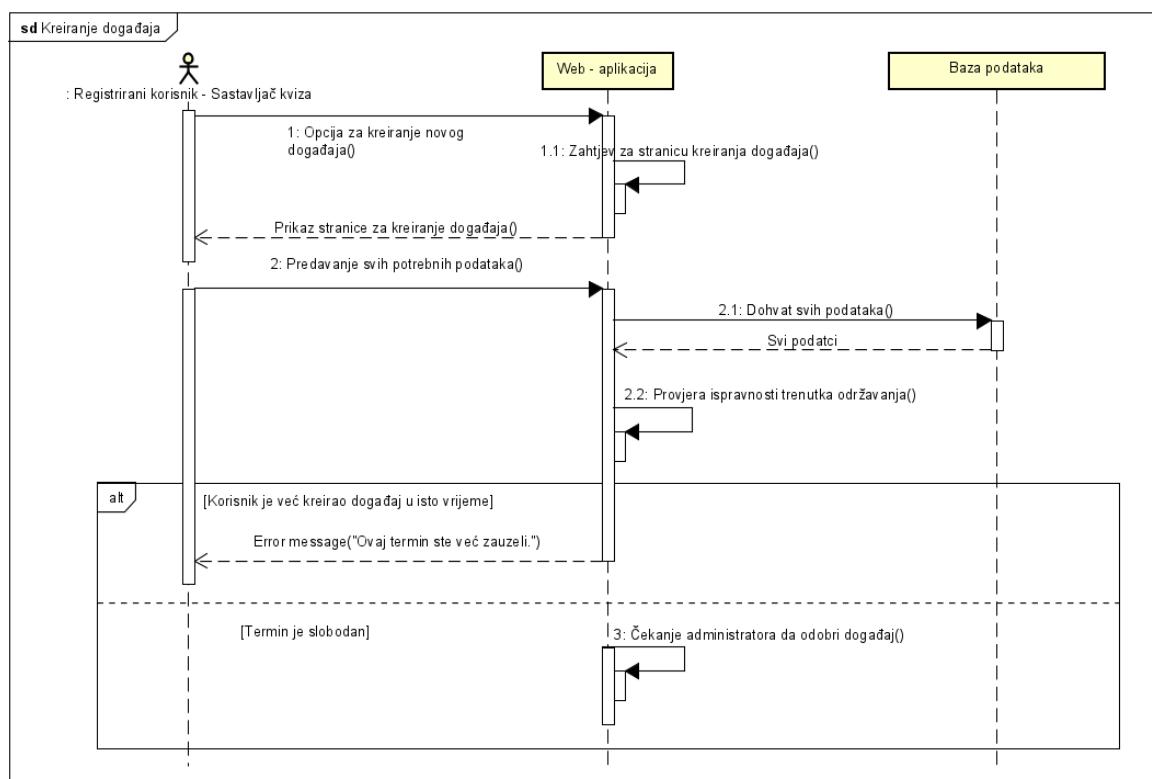
Neregistrirani korisnik šalje zahtjev za registracijskom stranicom. Web aplikacija korisniku dohvaća stranicu i prikazuje na ekranu sva registracijska polja. Nakon što neregistrirani korisnik popuni sva polja i preda ih, server provjerava ima li korisnika s istim e-mailom i nadimkom u bazi podataka, kako bi provjerio validnost predanih podataka. Ukoliko neko od polja nije ispunjeno, lozinka ne zadovoljava uvjete i korisničko ime je već zauzeto, mail nije jedinstven, podatci nisu validni i onda server obavještava neregistriranog korisnika o grešci. Ako su podatci validni, server pohranjuje novog korisnika u bazu podataka. Nakon toga web aplikacija dohvaća početnu stranicu i prikazuje ju korisniku.



Slika 3.4: Sekvencijski dijagram, registracija korisnika

### Obrazac uporabe UC06 - Kreiranje nadolazećih događaja

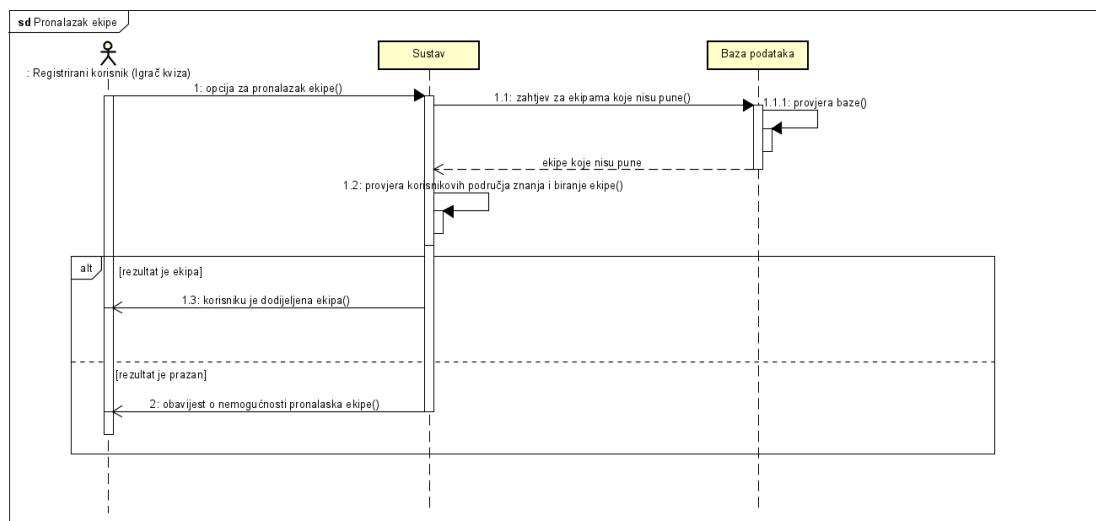
Registrirani korisnik(sastavljač kviza) šalje zahtjeva za stranicom za kreiranje novih događaja. Web aplikacija dohvaća stranicu i prikazuje tu stranicu na ekranu sa svim poljima koja trebaju biti popunjena. Nakon popunjavanja polja, korisnik predaje sve podatke. Nakon toga, server dohvaća sve podatke kako bi provjerio njihovu ispravnost. Ukoliko je sastavljač kviza stvorio novi događaj i održavanje novog događaja se poklapa s nekim drugim događajem koji je stvorio isti sastavljač, server obavještava sastavljača o pogrešci. Ako su podatci ipak ispravni, server čeka administratora kako bi on odobrio taj događaj.



Slika 3.5: Sekvencijski dijagram, kreiranje događaja

### Obrazac uporabe UC15 - Pronalazak ekipe za kviz

Registrirani korisnik (igrač kviza) bira opciju za pronalazak ekipe. Aplikacija su već dostupni podaci o igračevim najjačim područjima znanja. Aplikacija šalje upit bazi za ekipama koje još nisu pune. Baza nakon pretrage vraća što se od nje tražilo. Aplikacija zatim filtrira dobivene ekipe prema podacima koje ima o igračevim područjima znanja. U slučaju da takvih ekipa ima više, dodjeljuje mu jednu, a ako se dogodi da nema dostupnih ekipa, aplikacija o tome obavijesti igrača.

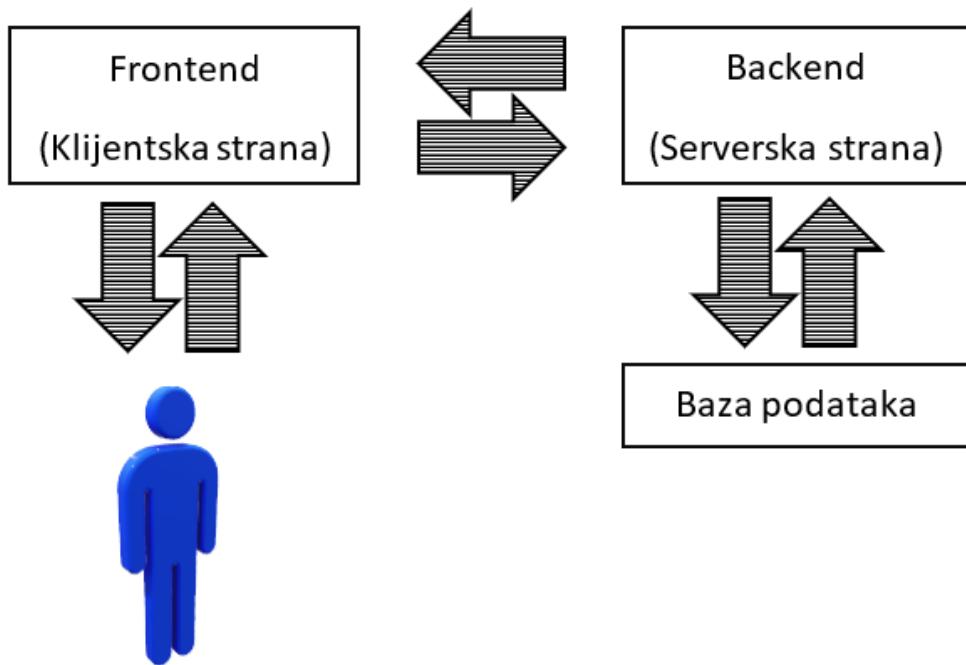


Slika 3.6: Sekvencijski dijagram, pronalazak ekipe

### 3.2 Ostali zahtjevi

- Aplikacija treba biti izvedena kao web aplikacija kojoj će korisnici pristupati uz pomoć jedinstvenog nadimka i lozinke.
- Aplikacija treba biti implementirana u arhitekturi klijent-poslužitelj, a potrebna funkcionalnost izložena kroz REST Web servise.
- Aplikacija treba biti prilagođena (engl. responsive) za mobilne uređaje i tablete.
- Programske dijelove koji uključuju rad s bazom podataka ne smiju se izvršavati duže od nekoliko sekundi.
- Sustav treba podržavati rad više korisnika u stvarnom vremenu.
- Sustav treba biti intuitivan i jednostavan za korištenje, a korisnici se moraju znati koristiti sučeljem bez korištenja opširnijih uputa.
- Korisničko sučelje pri unosu i prikazu tekstualnog sadržaja mora podržavati hrvatsku abecedu, odnosno dijakritičke znakove.
- Prijelaz sustavu treba biti omogućeno iz javne mreže pomoću protokola HTTPS.
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava.

## 4. Arhitektura i dizajn sustava



Slika 4.1: Arhitektura sustava

Korisnik aplikaciji pristupa putem web preglednika. Interakciju s aplikacijom ostvaruje preko korisničkog sučelja pomoću kojeg šalje zahtjeve web poslužitelju i prima odgovore.

Programski jezik pomoću kojeg je ostvaren backend web aplikacije je Java, a korišteni radni okvir je Spring Boot. Frontend aplikacije ostvaren je programskim jezikom JavaScript i bibliotekom React.js. Za razvojno okruženje odabran je Intellij IDEA. Spring Boot je radni okvir namijenjen stvaranju mikroservisa. Mikroservis je arhitektura koja omogućuje neovisan razvoj više različitih servisa od kojih svaki ima svoj proces.

Web aplikaciju čine tri osnovna dijela:

- **frontend**
- **backend**
- **baza podataka**

Frontend se sastoji od komponenata i logike. Istu komponentu je moguće koristiti za različite namjene (engl. reusability). React.js koristi virtualni DOM (engl. Document Object Model) čiji se sadržaj uspoređuje sa stvarnim DOM-om i na osnovu toga se provode promjene što za posljedicu ima poboljšanje performansi. Struktura ostvarena međusobnim povezivanjem različitih komponenti je stablo.

Backend se sastoji od:

- programskog sučelja za reprezentacijski prijenos stanja (REST API), odnosno Controller-a
- sloja poslovne logike (Service)
- sloja za pristup bazi podataka (Repository)

**Controller** izlaže funkcionalnost web aplikacije kao RESTful web usluge, tj. prima zahtjeve čiji su glavni dijelovi URI, metoda i HTTP zaglavlj, a korisniku šalje odgovor koji se sastoji od statusnog koda, tijela poruke i zaglavlj. U tijelu poruke se nalazi sadržaj kojeg korisnik konzumira nakon što je prikazan u web pregledniku. Komunikaciju sa slojem poslovne logike Controller ostvaruje pomoću umetanja ovisnosti (engl. dependency injection). Dependency injection je obrazac prema kojemu se u određeni objekt/funkciju umeće neki drugi objekt/funkcija na koji se prvobitno spomenuti objekt/funkcija oslanja.

**Service** omogućuje komunikaciju između slojeva Controller i Repository, zadužen je za provjeru ispravnosti podataka. Osim na sloju Service, provjera ispravnosti se obavlja na frontend-u i u bazi podataka. Komunikaciju sa slojem za pristup bazi podataka ostvaruje umetanjem ovisnosti.

**Repository** omogućuje komunikaciju s bazom podataka pomoću SQL-a. Objekti iz relacijske baze podataka pretvaraju se objekte programskog jezika Java korištenjem tehnike ORM.

## 4.1 Baza podataka

Baza podataka za aplikaciju implementirana je u obliku relacijske baze podataka, koja podatke sprema u obliku redaka ili n-torki i stupaca ili atributa koji zajedno tvore tablicu.

Glavna komponenta baze je tablica Korisnik, koja se puni osobnim podacima unesenim pri registraciji u sustav. Svakom novom korisniku dodjeljuje se primarni ključ, unikatan identifikacijski broj pomoću kojeg se korisnici u bazi podataka međusobno razlikuju. Budući da korisnik ima jednu ili više dodijeljenih uloga, koje su definirane u tablici Uloga, izrađena je veza između dvije tablice koja sadrži identifikacijske brojeve korisnika i odgovarajuće uloge.

Kako bi korisnik zaigrao na kvizu, potrebno je prvo pronaći ekipu kojoj odgovara svojim područjima znanja, zbog čega je izrađena tablica Ekipa koja sadrži minimalno jednoga, a maksimalno petero članova.

Ako korisnik ima ulogu sastavljača, on kreira kviz s podacima i informacijama koji se spremaju u tablicu Kviz. Svaki kviz ima svoj identifikacijski broj i unikatne attribute, vrijeme održavanja i ID sastavljača, jer jedan sastavljač ne može objaviti dva različita kviza koja se održavaju u isto vrijeme.

Tablica Obavijest sadrži svoj identifikacijski broj, tekst obavijesti koji se prikazuje korisniku i ID korisnika kao strani ključ.

#### 4.1.1 Opis tablica

<b>Korisnik</b>		
korisnik_id	BIG INT	Identifikacijski broj korisnika, primarni ključ.
ime	VARCHAR(100)	Ime korisnika.
prezime	VARCHAR(100)	Prezime korisnika.
email_adresa(U)	VARCHAR(100)	E-mail adresa korisnika.
lozinka	VARCHAR(50)	Lozinka koju korisnik osmisli pri registraciji.
slika	VARBINARY	Profilna slika korisnika u aplikaciji, optionalno.
broj_telefona	VARCHAR(50)	Broj telefona korisnika, optionalno.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Korisnik		
područja_znanja	VARCHAR(150)	Područja znanja igrača, koriste se pri odabiru ekipe.
nadimak(U)	VARCHAR(50)	Nadimak koji si korisnik osmisli pri registraciji.
prosj_broj_ekipa	INT	Prosječan broj ekipa koji sudjeluju u kvizovima nekog sastavljača, optionalno.
blokiran	BOOLEAN	Atribut koji daje informaciju o tome je li korisnik blokiran ili ne.
ekipa_id	BIG INT	Identifikacijski broj ekipe kojoj korisnik pripada, optionalno.

Uloga		
uloga_id	BIG INT	Identifikacijski broj uloge.
uloga_ime	VARCHAR(50)	Ime uloge.

korisnik_uloga		
uloga_id	BIG INT	Identifikacijski broj uloge, strani ključ s referencom na ulogu.
korisnik_id	BIG INT	Identifikacijski broj korisnika, strani ključ s referencom na korisnika.

Ekipa		
ekipa_id	BIG INT	Identifikacijski broj ekipe, primarni ključ.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

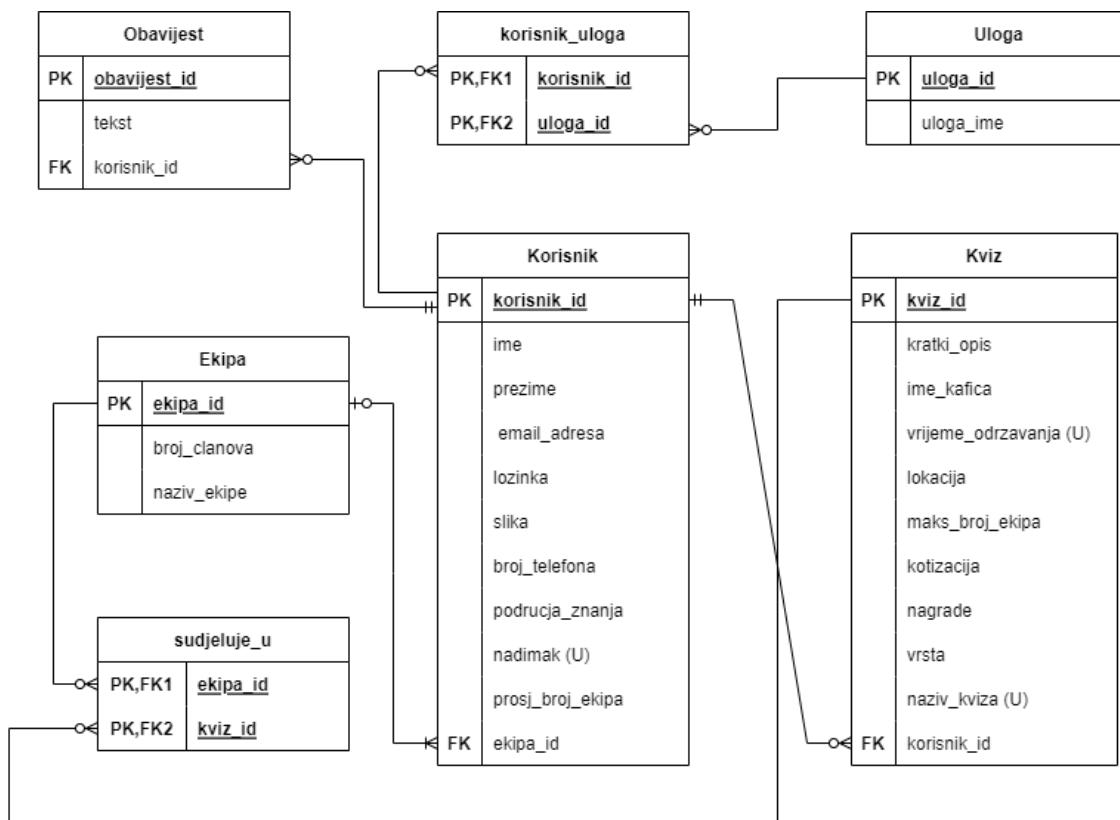
Ekipa		
broj_clanova	INT	Broj članova ekipe, ne smije biti manji od 1 ni veći od 5.
naziv_ekipe(U)	VARCHAR(50)	Ime ekipe.

Kviz		
kviz_id	BIG INT	Identifikacijski broj kviza, primarni ključ.
kratki_opis	VARCHAR(200)	Sažeti tekst o sadržaju kviza.
ime_kafica	VARCHAR(50)	Ime kafića u kojem se kviz održava.
vrijeme_odrzavanja (U)	TIMESTAMP	Vrijeme održavanja kviza, unikatan atribut zajedno s identifikacijskim brojem sastavljača.
lokacija	GEOGRAPHY	Lokacija mesta održavanja.
maks_broj_ekipa	INT	Maksimalni broj ekipa koje sudjeluju na kvizu.
kotizacija	FLOAT	Kotizacija za sudjelovanje na kvizu.
nagrade	VARCHAR(100)	Popis nagrada na visoko plasirane ekipe.
vrsta	VARCHAR(50)	Vrsta kviza (povijest, sport, ...).
naziv_kviza (U)	VARCHAR(50)	Unikatan naziv kviza.
aktivran	BOOLEAN	Predstavlja je li kviz trenutno aktivran ili ne.
korisnik_id	BIG INT	Identifikacijski broj sastavljača, strani ključ s referencom na korisnika.

sudjeluje_u		
ekipa_id	BIG INT	Identifikacijski broj ekipe, strani ključ s referencom na ekipu.
kviz_id	BIG INT	Identifikacijski broj kviza na kojem sudjeluje ekipa, strani ključ.

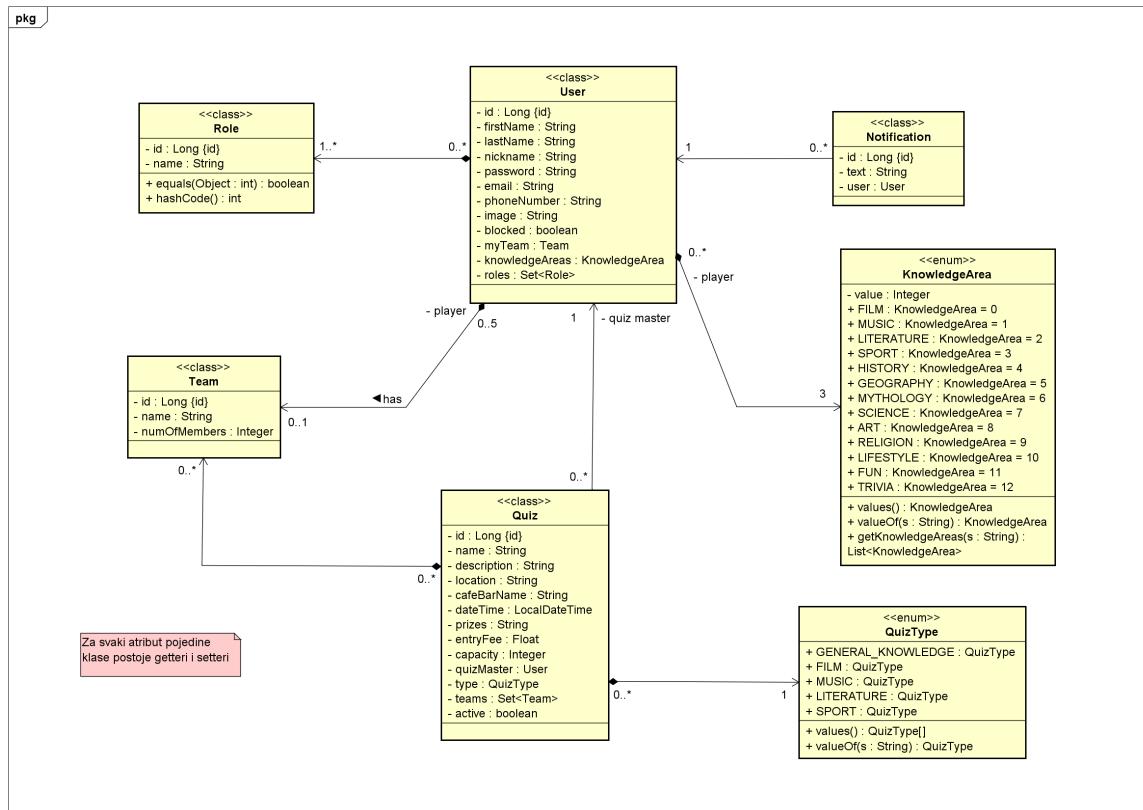
Obavijest		
obavijest_id	BIG INT	Identifikacijski broj obavijesti, primarni ključ.
tekst	VARCHAR(300)	Tekst obavijesti.
korisnik_id	BIG INT	Identifikacijski broj korisnika koji je primio obavijest.

#### 4.1.2 Dijagram baze podataka



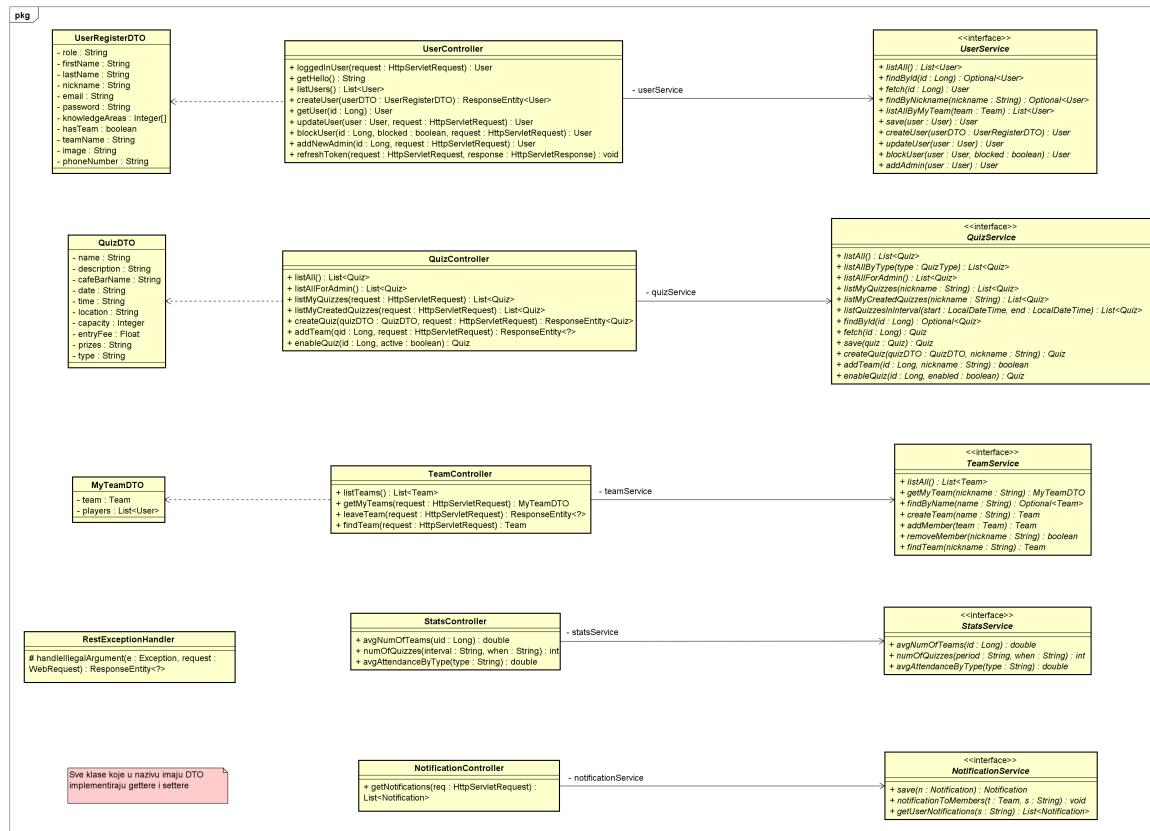
Slika 4.2: ER dijagram

## 4.2 Dijagram razreda



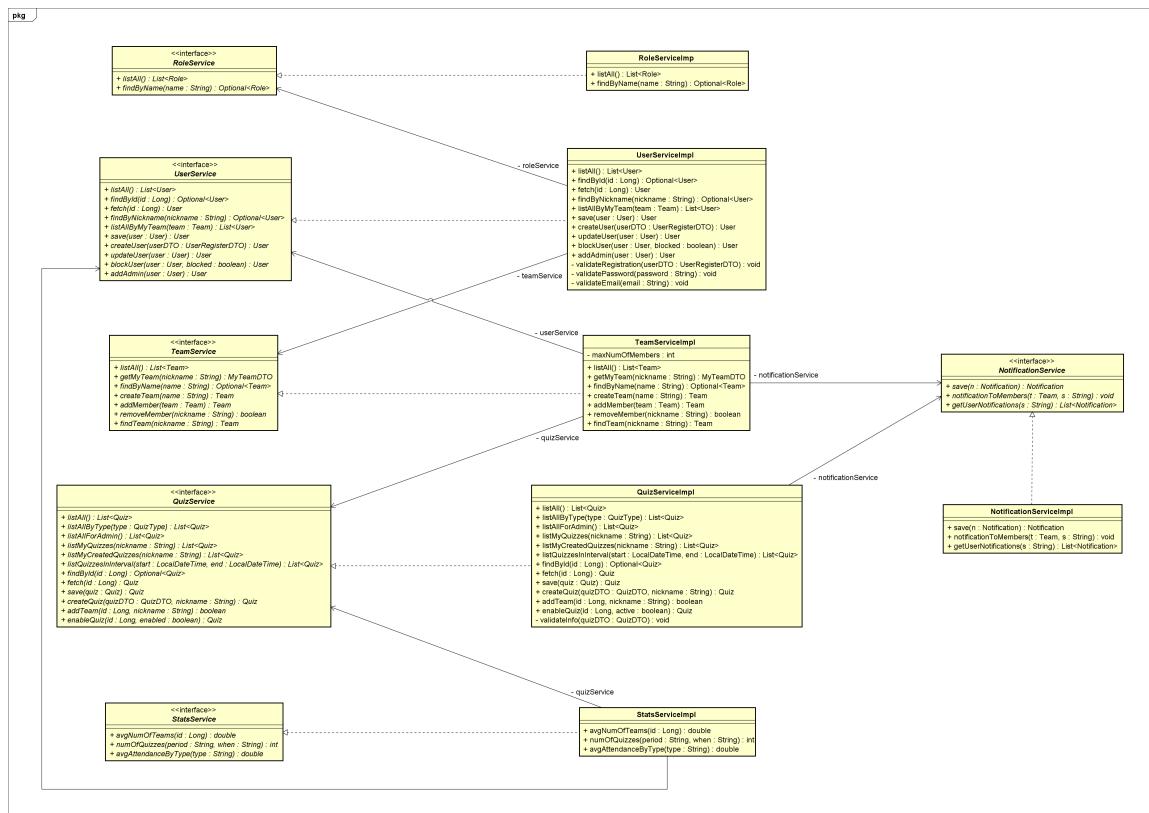
Slika 4.3: Dijagram razreda, dio Domain

Na slici 4.3 prikazan je dijagram razreda koji prikazuje razrede u objektno orijentiranom sustavu, njihove atribute i metode te međusobnu povezanost. Model razreda preslikava se u bazu podataka prema načelu ORM-a. Razred User predstavlja registriranog korisnika aplikacije koji može imati jednu ili više uloga opisane razredom Role. Korisnik prima obavijesti predstavljene razredom Notification, a kao igrač može pripadati najviše jednoj ekipi koja je opisana razredom Team. Također, korisnik koji je igrač izabire točno tri područja znanja koja su u aplikaciji ostvarena kao enumeracija KnowledgeArea. Korisnik koji je sastavljač može kreirati neograničen broj kvizova opisanih razredom Quiz, a svaki je kviz točno jedne vrste predstavljene enumeracijom QuizType te na jednom kvizu sudjeluje više ekipa, čiji je maksimalni broj određen atributom capacity razreda Quiz.



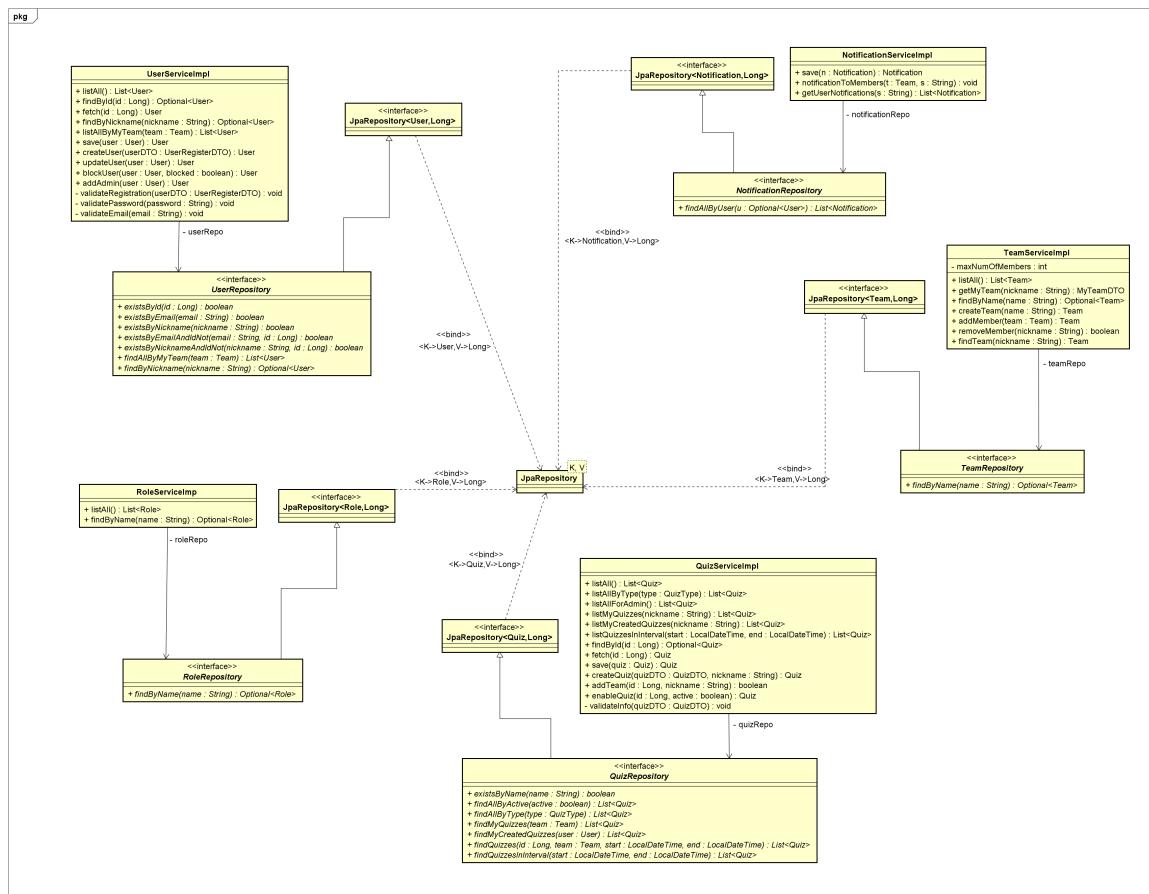
Slika 4.4: Dijagram razreda, Controllers, Services i DTO

Na slici 4.4 prikazan je dijagram razreda koji opisuje razrede i sučelja u aplikaciji koji pripadaju slojevima kontrolera i poslovne logike, njihove atribute i metode te međusobnu povezanost. Klase na sloju kontrolera prihvataju HTTP zahtjeve, pozivaju odgovarajuće usluge s kojima su na dijagramu povezane asocijacijom te vraćaju odgovor klijentu (najčešće u obliku JSON ili HTML datoteke). Za svoj rad koriste i razrede iz paketa dto, što je na dijagramu prikazano vezom ovisnosti pojedinog kontrolera i klase koja u sebi sadrži naziv DTO. Svi razredi iz spomenutog paketa implementiraju gettere i setttere.



Slika 4.5: Dijagram razreda, Services i ServicesImpl

Na slici 4.5 prikazan je dijagram razreda koji opisuje servisni sloj, na kojem se općenito ostvaruje temeljna funkcionalnost web aplikacije. Svaka klasa koja obavlja određeni zadatak na ovome sloju predstavlja implementaciju sučelja, čime se ostvaruje koncept smanjenja međuvisnosti i omogućava inverzija upravljanja radnog okvira. Osim realizacija sučelja, prikazane su i veze asocijacije pojedinih klasa koje za svoj rad koriste i druga sučelja sa servisnog sloja, a za čije se ubacivanje ovisnosti specifičnih implementacija pobrine sam Spring.

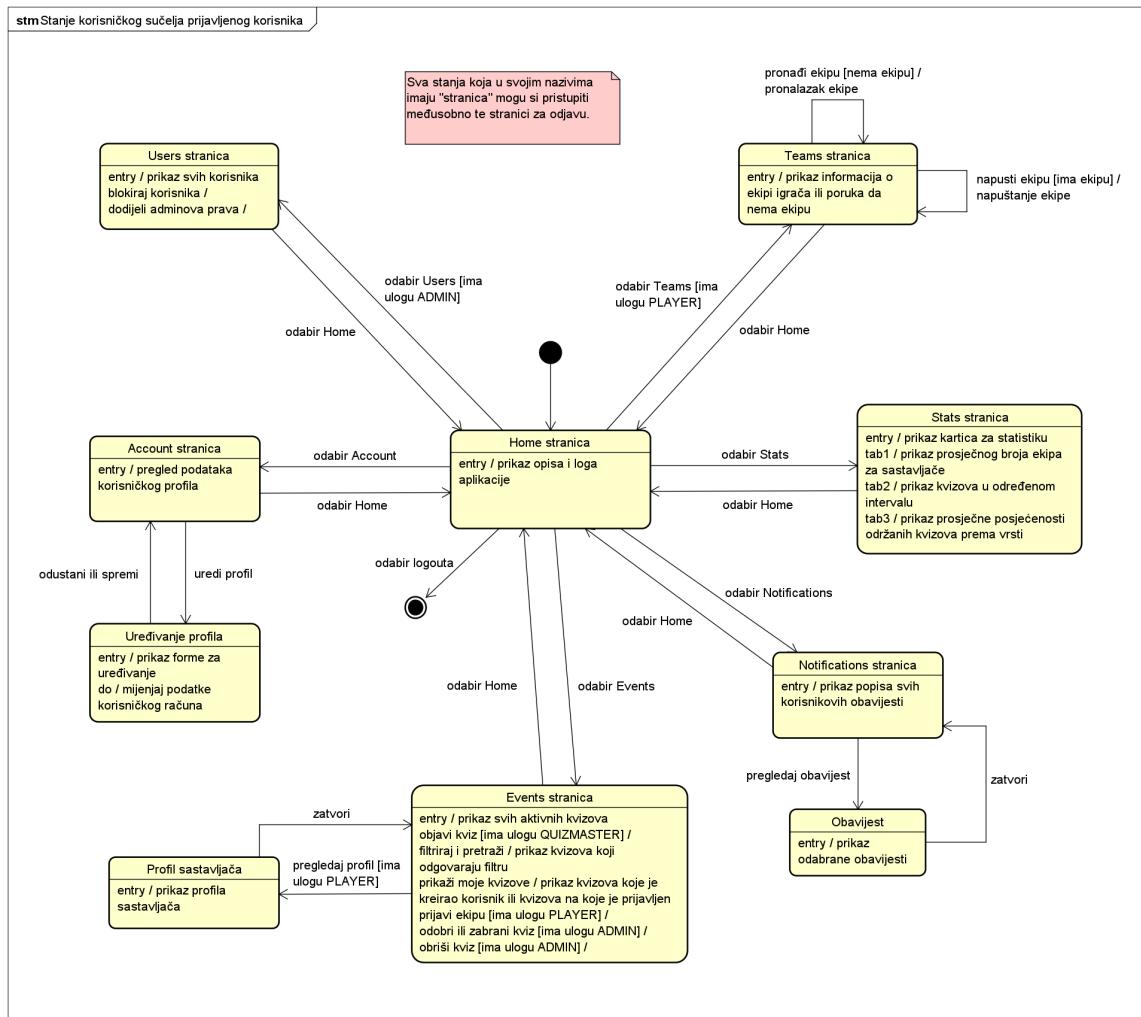


Slika 4.6: Dijagram razreda, Repositories i ServicesImpl

Na slici 4.6 prikazan je dijagram razreda koji opisuje sloj repozitorija te prikazuje pripadajuće implementacije sučelja servisnog sloja aplikacije. Svaki od navedenih razreda implementacije ostvaruje vezu sa svojim repozitorijem, a svaki repozitoriji predstavljen je sučeljem koje onda dodatno nasljeđuje sučelje JpaRepository<T, ID> radnog okvira Spring. Također su prikazane veze između spomenutih sučelja i JpaRepository<K, V> koje se nazivaju vezujući odnos (engl. binding relationship), a predstavljaju dodjeljivanje vrijednosti parametrima predloška i generiranje novih element modela iz postojećeg predloška.

## 4.3 Dijagram stanja

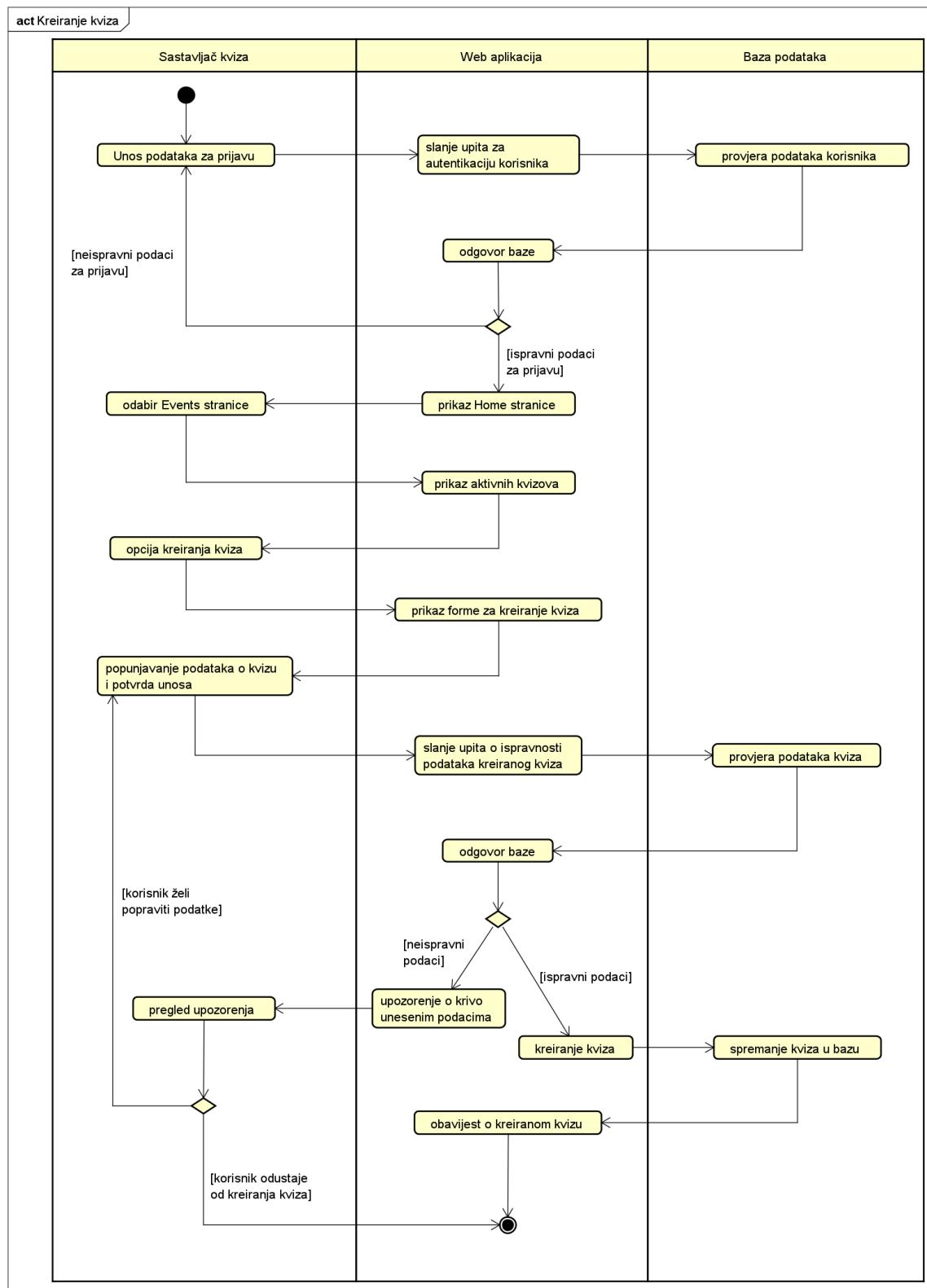
Dijagram stanja opisuje dinamičko ponašanje dijela sustava u vremenu i modelira situaciju tijekom koje vrijedi određeni skup uvjeta. Na slici je prikazan dijagram stanja koji prikazuje korisničko sučelje prijavljenog korisnika. Nakon prijave, korisnik je preusmjeren na Home stranicu na kojoj može pregledati opis i logo aplikacije. Koristeći navigacijsku traku može odabirati bilo koju od ponuđenih stranica aplikacije i tako mijenjati prikaz, odnosno stanja. Na stranici Account pregledava svoj korisnički profil te ga može ažurirati odabirom uređivanja profila. Odabirom stranice Stats korisnik može pregledati statistiku, a odabere li stranicu Notifications može vidjeti svoje obavijesti i bilo koju od njih dodatno pregledati. Na stranici Events prikazuju se svi aktivni kvizovi, a korisnik ovisno o ulozi koju ima u aplikaciji može izvoditi opisane akcije. Također, ako korisnik ima ulogu administratora može odabrati stranicu Users te blokirati ili dodijeliti administratorska prava ostalim korisnicima aplikacije. Igrač može pregledati stranicu Teams na kojoj su prikazane informacije o njegovoj ekipi, pri čemu može izvršiti akcije pronašlaska ili napuštanja tima. Iz svih stanja moguće je odjaviti se iz sustava.



Slika 4.7: Dijagram stanja

## 4.4 Dijagram aktivnosti

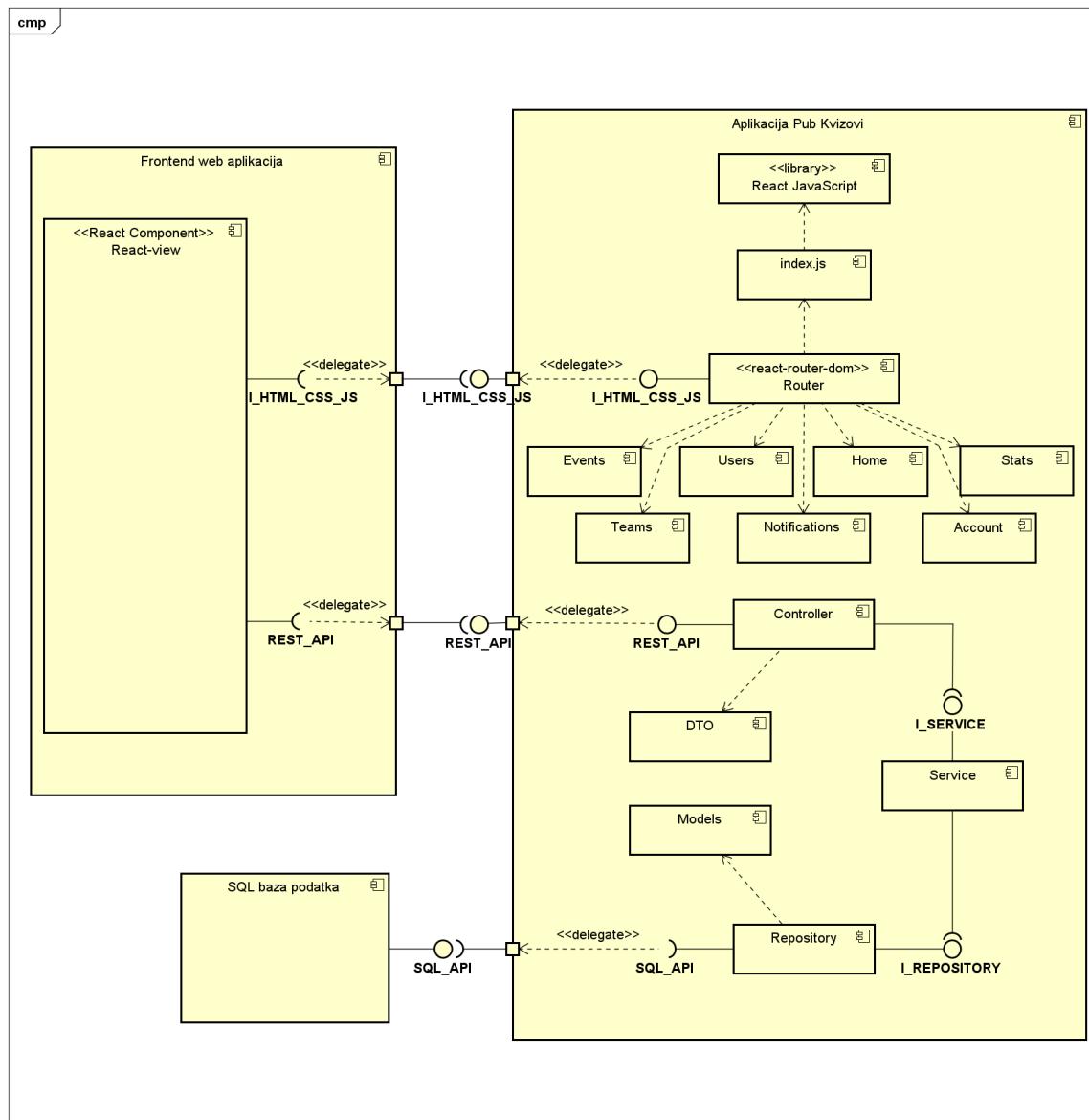
Dijagram aktivnosti je ponašajni UML dijagram koji modelira ponašanja nizom akcija, pri čemu se mogu definirati uvjeti prije i nakon njihova izvođenja. Primjenjuje se za modeliranje poslovnih procesa i upravljačkog toka, na primjer toka analize obrazaca uporabe kao što je to i ovdje slučaj. Dijagram na slici prikazuje aktivnost kreiranja pub kviza, a podijeljen je na tri vertikalne particije prema aktorima sustava. Sastavljač kviza prijavljuje se u aplikaciju, nakon čega se provjerava ispravnost unesenih podataka. Ako su podaci neispravni korisnik se vraća na unos podataka, a ako su ispravni preusmjerava se na početnu stranicu. Nakon odabira stranice Events i unosa podataka za kreiranje kviza slijedi evaluacija istih podataka i vraćanje odgovora o njihovoj ispravnosti. Ako su podaci neispravni sastavljač može odustati od kreiranja kviza ili ponovno unijeti podatke. Međutim, ako su uneseni podaci ispravni kviz se kreirao i sprema se u bazu podataka, nakon čega slijedi obavijest o kreiranom kvizu i završavanje s radom.



Slika 4.8: Dijagram aktivnosti

## 4.5 Dijagram komponenti

Dijagram komponenti je strukturni statički UML dijagram koji služi za vizualizaciju organizacije i međuvisnosti između implementacijskih komponenata i čini dio specifikacije arhitekture programske potpore. Za pristup sustavu koriste se dva sučelja. Preko sučelja I\_HTML\_CSS\_JS dohvaćaju se HTML, CSS i JS datoteke koje pripadaju frontend dijelu aplikacije. Komponenta Router na upit preko URL-a odlučuje koja će se datoteka poslužiti na sučelje. Frontend se sastoji od JavaScript datoteka koje su raspoređene po komponentama nazvanim prema stranicama aplikacije. Sve te datoteke ovise o React biblioteci iz koje dohvaćaju već gotove komponente poput formi i slično. Preko REST API sučelja pristupa se komponenti Controller koja poslužuje podatke iz backend dijela aplikacije, a za svoj radi koristi klase komponente DTO koje služe za prijenos podataka između frontenda i backenda. Service ima dva sučelja - implementirano I\_SERVICE preko kojeg ostvaruje vezu s Controllerom te zahtijevano I\_REPOSITORY preko kojeg poziva metode klase sa sloja Repository. Komponenta Repository povezana je vezom ovisnosti s komponentom Models koja predstavlja domenske klase aplikacije, a Repository je zadužen za dohvaćanje podataka iz tablica baze podataka pomoću SQL upita, koristeći pritom SQL API sučelje. React-view komponenta preko dostupnih sučelja komunicira s aplikacijom te ovisno o korisnikovim akcijama osvježava prikaz i dohvaća nove podatke ili datoteke.



Slika 4.9: Dijagram komponenti

# 5. Implementacija i korisničko sučelje

## 5.1 Korištene tehnologije i alati

Komunikacija u timu realizirana je korištenjem aplikacija WhatsApp<sup>1</sup> i Discord<sup>2</sup>. Za izradu UML dijagrama korišten je alat Astah Professional<sup>3</sup>, a kao sustav za upravljanje izvornim kodom Git<sup>4</sup>. Udaljeni rezervorij projekta je dostupan na web platformi GitLab<sup>5</sup>.

Kao razvojno okruženje korišten je IntelliJ IDEA<sup>6</sup> - integrirano razvojno okruženje (IDE) napisano u Javi za razvoj računalnih softvera napisanih u Javi, Kotlinu, Scali, Groovyu i drugim jezicima koji se temelje na JVM-u. Razvio ga je JetBrains (ranije poznat kao IntelliJ) i dostupan je kao Apache 2 licencirano izdanje zajednice (Community Edition) te u vlasničkom komercijalnom izdanju (Ultimate). Oba se mogu koristiti za komercijalni razvoj, a dostupni su za operacijske sisteme Windows, macOS i Linux.

Aplikacija je napisana koristeći radni okvir Spring Boot<sup>7</sup> i jezik Java<sup>8</sup> za izradu backenda te React<sup>9</sup> i jezik JavaScript<sup>10</sup> za izradu frontenda. React, također poznat kao React.js ili ReactJS, je biblioteka u JavaScriptu za izgradnju korisničkih sučelja. Održava je Facebook. React se najčešće koristi kao osnova u razvoju web ili mobilnih aplikacija. Složene aplikacije u Reactu obično zahtijevaju korištenje dodatnih biblioteka za interakciju s API-jem. Spring je radni okvir korišten za olakšan razvoj aplikacija, a omogućava konfiguraciju objekata i lakše upravljanje objektima poslovne logike te općenito brine o kreiranju svih objekata, njihovom povezivanju i čitavom životnom ciklusu. Spring Boot je jedna od komponenti unutar Springa, koja dodatno olakšava njegovo korištenje i omogućava izradu čitavih samostalnih

<sup>1</sup><https://www.whatsapp.com/>

<sup>2</sup><https://discord.com/>

<sup>3</sup><https://astah.net/editions/professional>

<sup>4</sup><https://git-scm.com/>

<sup>5</sup><https://gitlab.com/>

<sup>6</sup><https://www.jetbrains.com/idea/>

<sup>7</sup><https://start.spring.io/>

<sup>8</sup><https://www.oracle.com/java>

<sup>9</sup><https://reactjs.org/>

<sup>10</sup><https://www.javascript.com/>

aplikacija. Cilj je ubrzanje i pojednostavljenje razvoja programske potpore.

Baza podataka se nalazi na hosting aplikaciji u oblaku, Render<sup>11</sup>. Renderova PostgreSQL ponuda olakšava korištenje PostgreSQL-a na siguran, pouzdan i potpuno nesmetan način.

---

<sup>11</sup><https://render.com/>

## 5.2 Ispitivanje programskog rješenja

### 5.2.1 Ispitivanje komponenti

#### 1. Testovi za QuizServiceImpl

Izvorni kod:

```
└── brancokas
    └── @ExtendWith(MockitoExtension.class)
        └── class QuizServiceImplTest {
            ...
            7 usages
            @InjectMocks
            private QuizServiceImpl quizService;

            @Mock
            private NotificationService notificationService;

            4 usages
            @Mock
            private QuizRepository quizRepo;

            6 usages
            @Mock
            private UserService userService;

            7 usages
            @Mock
            private User user;

            3 usages
            @Mock
            private Team team;
            9 usages
            @Mock
            private Quiz quiz1, quiz2;
```

Slika 5.1: Klasa QuizServiceImplTest

```

9 usages
@Mock
private Quiz quiz1, quiz2;

9 usages
@Mock
private QuizDTO quizDTO;

▲ brancokas
@Test
void listMyQuizzes_NoUser_ThrowException() {
    given(userService.findByNickname("nickname")).willReturn(Optional.empty());

    var error : EntityMissingException = assertThrows(EntityMissingException.class, () -> quizService.listMyQuizzes( nickname: "nickname"));

    assertThat(error.getMessage()).isEqualTo(
        expected: "Entity with reference nickname of class hr.fer.progi.pubkvizovi.domain.User not found.");
}

▲ brancokas
@Test
void listMyQuizzes_NoTeam_ThrowException() {
    given(userService.findByNickname("nickname")).willReturn(Optional.of(user));

    assertThrows(RequestDeniedException.class, () -> quizService.listMyQuizzes( nickname: "nickname"));
}

```

Slika 5.2: Testovi za listu mojih kvizova, iznimke „nema korisnika” i „nema tima”

```

▲ brancokas
@Test
void listMyQuizzes_TeamsQuizzes() {
    given(userService.findByNickname("nickname")).willReturn(Optional.of(user));
    given(user.getMyTeam()).willReturn(team);
    given(quizRepo.findMyQuizzes(team)).willReturn(List.of(quiz1, quiz2));

    var result : List<Quiz> = quizService.listMyQuizzes( nickname: "nickname");

    assertThat(result).containsExactly(quiz1, quiz2);
}

▲ brancokas
@Test
void listMyCreatedQuizzes_NoUser_ThrowException() {
    given(userService.findByNickname("quizmaster")).willReturn(Optional.empty());

    assertThrows(EntityMissingException.class, () -> quizService.listMyCreatedQuizzes( nickname: "quizmaster"));
}

▲ brancokas
@Test
void listMyCreatedQuizzes_QuizmasterCreatedQuizzes() {
    given(userService.findByNickname("quizmaster")).willReturn(Optional.of(user));
    given(quizRepo.findMyCreatedQuizzes(user)).willReturn(List.of(quiz1, quiz2));

    var result : List<Quiz> = quizService.listMyCreatedQuizzes( nickname: "quizmaster");

    assertThat(result).containsExactly(quiz1, quiz2);
}

```

Slika 5.3: Lista kvizova, iznimka u mojim kreiranim kvizovima „nema korisnika”, lista mojih kreiranih kvizova

```

1 usage  ✘ brancokas
private void init() {
    when(quizDTO.getName()).thenReturn("Name");
    when(quizDTO.getDescription()).thenReturn("Description");
    when(quizDTO.getCafeBarName()).thenReturn("CaffeBar");
    when(quizDTO.getDate()).thenReturn("2024-10-10");
    when(quizDTO.getTime()).thenReturn("18:00:00");
    when(quizDTO.getPrizes()).thenReturn("Prize");
    when(quizDTO.getLocation()).thenReturn("Location");
}

✘ brancokas
@Test
void createQuiz_QuizWithSameName_ThrowException() {
    init();
    given(quizRepo.existsByName(quizDTO.getName())).willReturn(true);

    assertThrows(RequestDeniedException.class, () -> quizService.createQuiz(quizDTO, nickname: "master"));
}

```

Slika 5.4: Iznimka, „kreiran kviz sa istim imenom”

```

✘ brancokas
@Test
void addTeamToQuiz_FullCapacity_ThrowException() {
    long id = 1L;
    String nickname = "nickname";
    given(quizRepo.findById(id)).willReturn(Optional.of(quiz1));
    given(userService.findByNickname(nickname)).willReturn(Optional.of(user));
    given(user.getMyTeam()).willReturn(team);
    when(quiz1.getCapacity()).thenReturn(10);
    when(quiz1.getNumOfTeams()).thenReturn(10);

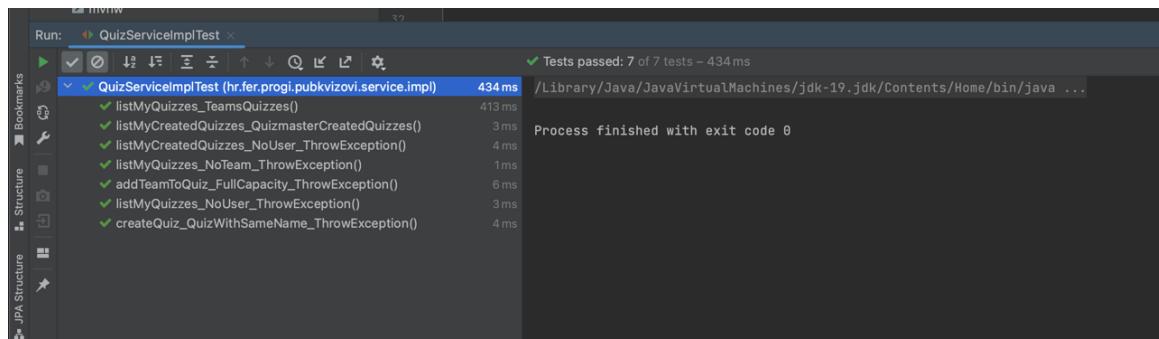
    var error : IllegalArgumentException = assertThrows(IllegalArgumentException.class, () -> quizService.addTeam(id, nickname));

    assertThat(error.getMessage().equals("Quiz is full. " + quiz1.getNumOfTeams() + "/" + quiz1.getCapacity()
        + " teams"));
}

}

```

Slika 5.5: Iznimka za dodavanje tima kvizu, „pun kapacitet”



Slika 5.6: Rezultati testova

## 2. Testovi za TeamServiceImpl

Izvorni kod:

```
brancokas
@ExtendWith(MockitoExtension.class)
class TeamServiceImplTest {

    2 usages
    @InjectMocks
    TeamServiceImpl teamService;

    @Mock
    NotificationService notificationService;

    @Mock
    UserRepository userRepository;

    1 usage
    @Mock
    UserService userService;

    @Mock
    TeamRepository teamRepo;

    2 usages
    @Mock
    Team team, team1;
```

Slika 5.7: Klasa TeamServiceImplTest

```
3 usages
@Mock
User user;

▲ brancokas
@Test
void addMember_FullTeam_ThrowException() {
    when(team.getNumberOfMembers()).thenReturn( t 5);

    var error : IllegalArgumentException = assertThrows(IllegalArgumentException.class, () -> teamService.addMember(team, user));

    assertThat(error.getMessage().equals("This team is full."));
}

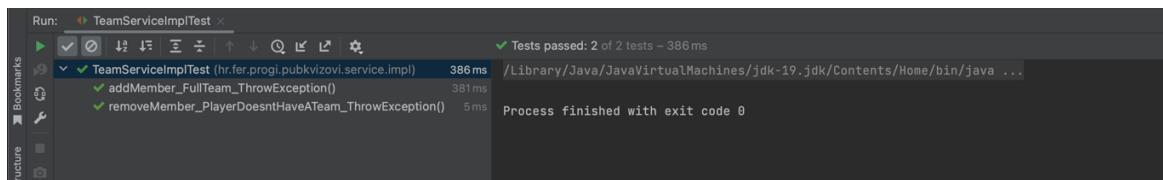
▲ brancokas
@Test
void removeMember_PlayerDoesntHaveATeam_ThrowException() {
    given(userService.findByNickname("nickname")).willReturn(Optional.of(user));
    when(user.getMyTeam()).thenReturn( t null);

    var error : RequestDeniedException = assertThrows(RequestDeniedException.class,
        () -> teamService.removeMember( nickname: "nickname"));

    assertThat(error.getMessage().equals("Player doesn't have a team."));
}

}
```

Slika 5.8: Iznimke „pun tim“ i „igrac nema tim“



Slika 5.9: Rezultati testova

### 3. Testovi za UserServiceImpl

Izvorni kod:

```
└── brancokas
    └── ExtendWith(MockitoExtension.class)
        class UserServiceImplTest {

            3 usages
            @InjectMocks
            UserServiceImpl userService;

            3 usages
            @Mock
            UserRepository userRepo;

            4 usages
            @Mock
            User user;

            └── brancokas
                └── Test
                    void fetch_NoId_ThrowException() {
                        given(userRepo.findById(1L)).willReturn( t: Optional.empty());
                        assertThrows(EntityMissingException.class, () -> userService.fetch( id: 1L));
                    }
                }
            }
        }
    }
}
```

Slika 5.10: Klasa UserServiceImplTest, iznimka „nema id-a”

```
brancokas
@Test
void fetchId() {
    given(userRepo.findById(1L)).willReturn(Optional.of(user));

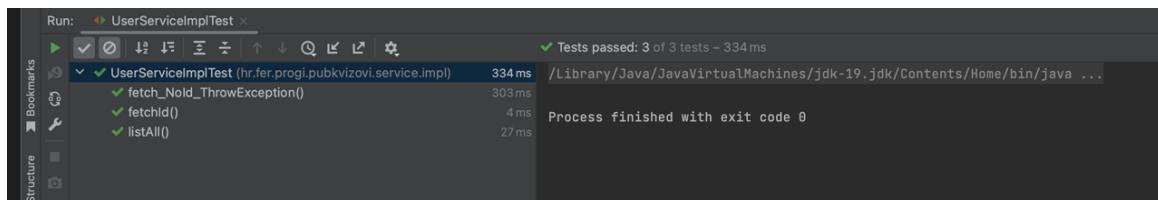
    assertEquals(user, userService.findById(1L).get());
}

brancokas
@Test
void listAll() {
    given(userRepo.findAll()).willReturn(List.of(user));

    var result : List<User> = userService.listAll();

    assertThat(result).containsExactly(user);
}
}
```

Slika 5.11: Testovi za „dohvati id” i dohvati svih korisnika



Slika 5.12: Rezultati testova

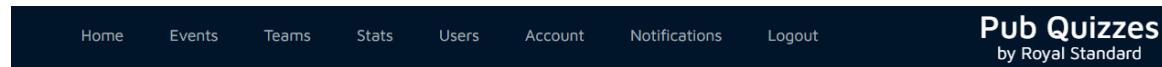
## 5.2.2 Ispitivanje sustava

### 1. Prijava

Već registrirani korisnik se nalazi na stranici za prijavu. U polja za unos(Nickname i Password) unosi svoje podatke koje je odredio pri registraciji. Ako je prijava uspješna kao u ovom slučaju, prijavljeni korisnik se usmjerava na Home stranicu, odnosno početnu stranicu aplikacije.

http://localhost:3000/login		
	Command	Target
		Value
1	✓ open	http://localhost:3000/login
2	✓ set window size	1042x662
3	✓ click	id=basic_nickname
4	✓ type	id=basic_nickname
5	✓ click	id=basic_password
6	✓ type	id=basic_password
7	✓ click	css=.ant-btn > span
8	✓ execute script	return window.location.href
9	✓ assert	pageurl
		http://localhost:3000/home

Slika 5.13: Postavke testa



### Home

Aplikacija nudi mogućnost lakšeg oglašavanja kvizova kvizaškoj zajednici, ali isto tako i lakše pronalaženje istih. Igrači koji bi se htjeli okušati u igranju kvizova, ali nemaju ekipu, takoder će pronaći prednosti aplikacije jer će ih ona sama međusobno povezati ovisno o njihovim područjima znanja. Aplikacija će upravo tim međusobnim povezivanjem igrača, ali i sastavljača kvizova pridonijeti stvaranju veće i jače kvizaške zajednice te samim time jačanju i širenju kvizaške scene.

Slika 5.14: Rezultat testa

## 2. Neuspješna prijava

Korisnik se nalazi na stranici za prijavu te u polja za unos(Nickname i Password) unosi podatke. Nakon unosa i pokušaja prijave, korisniku je javljeno da se nije moguće prijaviti. Sustav mu javlja "You are blocked or your credentials are wrong!". Dvije su mogućnosti zašto se korisnik ne može prijaviti, jedna od njih je ako je korisnik blokiran od strane admina, a druga mogućnost je da je unio krive podatke za prijavu.

http://localhost:3000/login		
Command	Target	Value
1 ✓ open	http://localhost:3000/login	
2 ✓ set window size	1046x666	
3 ✓ click	id=basic_nickname	
4 ✓ type	id=basic_nickname	a
5 ✓ click	id=basic_password	
6 ✓ type	id=basic_password	Sifra123
7 ✓ click	css= ant-btn > span	
8 ✓ execute script	return window.location.href	pageurl
9 ✓ assert	pageurl	http://localhost:3000/login
10 ✓ verify text	css=.centered-flex > div	You are blocked or your credentials are wrong!

Slika 5.15: Postavke testa

The screenshot shows a login interface. At the top, there's a dark navigation bar with 'Home', 'Register', and 'Login' buttons. The main content area is titled 'Login'. It features two input fields: one for 'Nickname' containing 'a' and another for 'Password' containing 'Sifra123'. Below these is a blue 'Submit' button. A red error message 'You are blocked or your credentials are wrong!' is centered at the bottom of the form.

Slika 5.16: Rezultat testa

### 3. Uspješna registracija

Korisnik se nalazi na stranici za registraciju i ispunjava sve potrebne podatke(ime, prezime, nadimak, email, ulogu, područja znanja i šifru). Nakon klika gumba Submit, ako je registracija uspješna kao u ovom slučaju, novoregistrirani korisnik se usmjerava na stranicu za prijavu gdje se s tim istim podatcima mora prijaviti.

	Command	Target	Value
1	✓ open	http://localhost:3000/register	
2	✓ set window size	1042x662	
3	✓ click	id=basic(firstName	
4	✓ type	id=basic(firstName	Petar
5	✓ click	id=basic(lastName	
6	✓ type	id=basic(lastName	Petrović
7	✓ click	id=basic(nickname	
8	✓ type	id=basic(nickname	Petar123
9	✓ click	id=basic(email	
10	✓ type	id=basic_email	petar@gmail.com
11	✓ click	id=basic_role	
12	✓ click	css=div[title="quiz master"]	
13	✓ click	id=basic_password	
14	✓ type	id=basic_password	Petar123
15	✓ click	css= ant-btn nth-child(1) > span	
16	✓ pause	1000	
17	✓ execute script	return window.location.href	pageurl
18	✓ assert	pageurl	http://localhost:3000/login
19	✓ close		

Slika 5.17: Postavke testa

The screenshot shows a browser window with the following details:

- URL Bar:** http://localhost:3000/login
- Page Title:** Pub Quizzes by Royal Standard
- Form Fields:**
  - \* Nickname: (empty input field)
  - \* Password: (empty input field with a clear icon)
- Buttons:** A blue "Submit" button at the bottom of the form.

Slika 5.18: Rezultat testa

#### 4. Neuspješna registracija

Korisnik se nalazi na stranici za registraciju i ispunjava polja, no nije ispunio polja "E-mail" i "Knowledge areas:" koja su obavezna. Nakon klika na gumb Submit, neće doći do uspješne registracije jer korisnik nije unio sve obavezne podatke, pa sukladno tome neće biti registriran.

http://localhost:3000/register		
	Command	Target
1	✓ open	http://localhost:3000/register
2	✓ set window size	1042x662
3	✓ click	id=basic(firstName
4	✓ type	id=basic(firstName
5	✓ click	id=basic(lastName
6	✓ type	id=basic(lastName
7	✓ click	id=basic(nickname
8	✓ type	id=basic(nickname
9	✓ click	id=basic(role
10	✓ click	css=div[title="player"]
11	✓ click	id=basic(password
12	✓ type	id=basic(password
13	✓ click	css=ant-btn:nth-child(1) > span
14	✓ pause	1000
15	✓ execute script	return window.location.href
16	✓ assert	pageurl
17	✓ verify text	css=div#basic_email_help > div
18	✓ verify text	css=div#basic_hasTeam_help > div
19	✓ verify text	css=div#basic_knowledgeAreas_help > div

Slika 5.19: Postavke testa

\* E-mail :

Please input your e-mail!

Phone number :

\* Role :

player



\* I have a team :  Yes

No

Please choose this option!

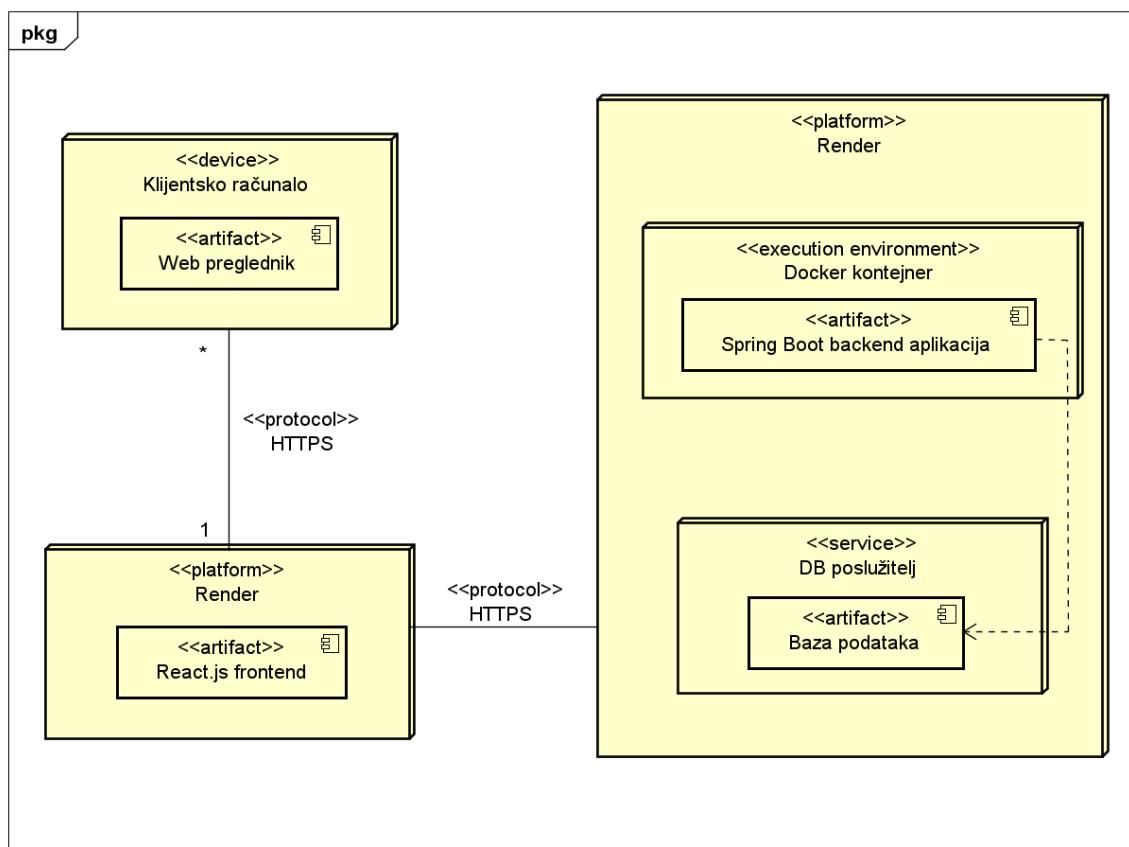
\* Knowledge areas :

Please select three areas!

Slika 5.20: Rezultat testa

## 5.3 Dijagram razmještaja

Dijagram razmještaja je strukturalni staticki UML dijagram koji opisuje topologiju sustava i općenito je usredotočen na odnos sklopovskih i programske dijelova. U ovom slučaju klijent se pomoću web preglednika spaja na korisničko sučelje (frontend) preko kojeg komunicira s poslužiteljskom aplikacijom (backend), a za komunikaciju se koristi protokol HTTPS. Korisničko sučelje implementirano je koristeći radni okvir React.js i pokrenuto je na platformi Render, dok je poslužiteljska aplikacija također pokrenuta na platformi Render, ali ostvarena je pomoću radnog okvira Spring Boot te pokrenuta unutar kontejnera Docker. Baza podataka, u koju backend aplikacija sprema sve podatke, izvodi se na svome poslužitelju, a nalazi se isto na platformi Render.

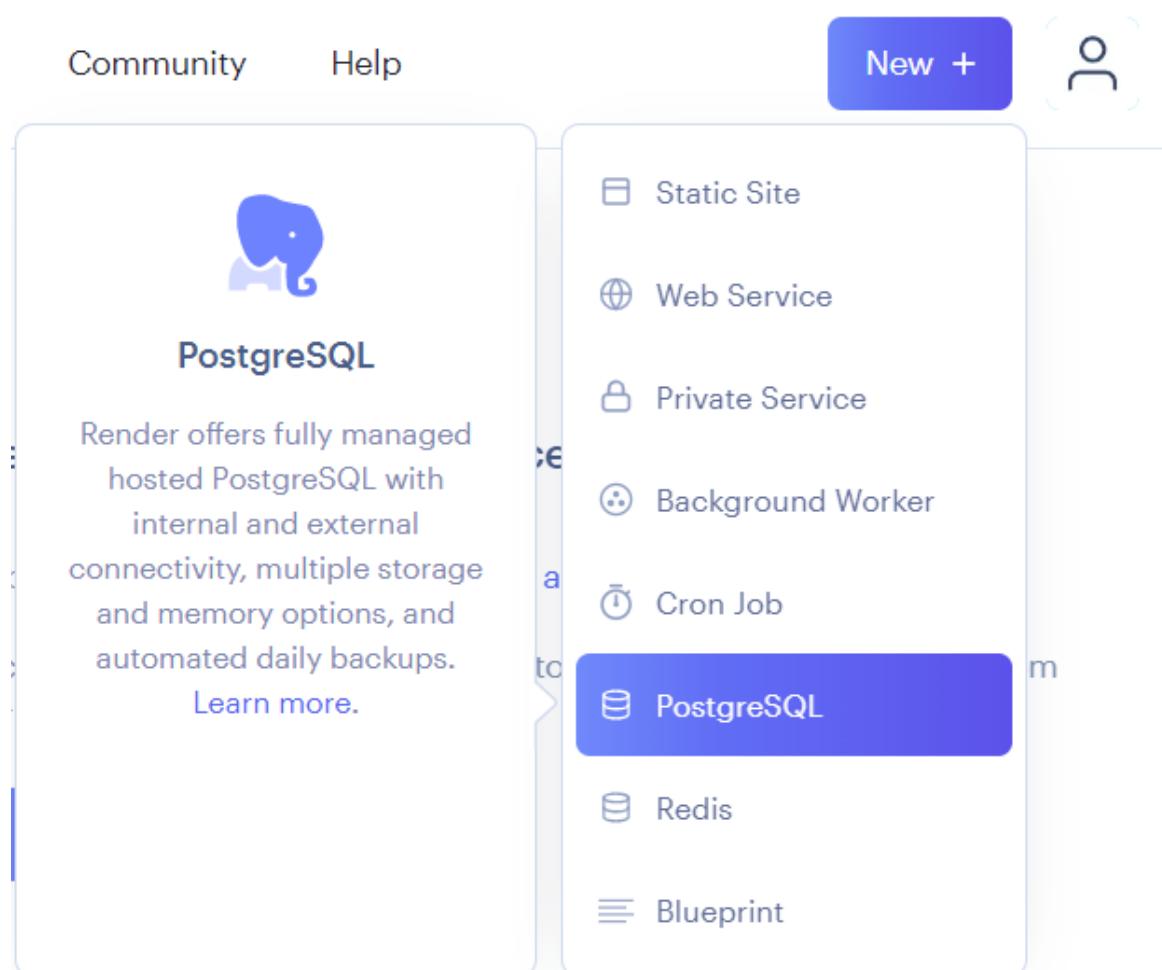


Slika 5.21: Dijagram razmještaja

## 5.4 Upute za puštanje u pogon

Za puštanje aplikacije u pogon bila su potrebna tri koraka, odnosno puštanje u pogon svake od tri važne komponente aplikacije, baze podataka, poslužiteljske strane i klijentske strane. To smo radili na platformi Render.

Izbornik New pokraj korisničkog imena nudi opcije za puštanje u pogon željene komponente aplikacije. U prvom koraku odabrana je opcija „PostgreSQL“ za pohranu baze podataka na platformi. Nakon toga se unesu osnovni podaci o bazi, a platforma Render onda sama generira ostale potrebne podatke (HostName, Password,...). Nakon što je baza postavljena i na platformi, u željenom alatu (odabran PgAdmin) se baza treba registrirati pomoću podataka koje je generirala platforma Render i time je puštanje baze podataka u pogon završeno.



Slika 5.22: Izbornik za kreiranje novih komponenata

## New PostgreSQL

Name Required RoyalStandard

Database Required royalstandard

User Required username

Region  
The region where your Database runs. Services must be in the same region to communicate privately and you currently have services running in Frankfurt.

PostgreSQL Version Required 15

Datadog API Key Optional

Please enter your payment information to select an instance type with higher limits.

Instance Type	RAM	CPU	Storage	Price
<input checked="" type="radio"/> Free	256 MB	Shared	1 GB	\$0 / month
<input type="radio"/> Starter	256 MB	Shared	1 GB	\$7 / month
<input type="radio"/> Standard	1 GB	1 CPU	16 GB	\$20 / month
<input type="radio"/> Pro	4 GB	2 CPU	96 GB	\$95 / month
<input type="radio"/> Pro Plus	8 GB	4 CPU	256 GB	\$185 / month

Need a [custom plan](#)? We support up to 512 GB RAM, 64 CPUs, and 5 TB storage.

Slika 5.23: Upisivanje početnih podataka za bazu podataka

### General

Name RoyalStandard [Edit](#)

Created 2 months ago

Expiration February 13, 2023 [Edit](#)

Status Available

PostgreSQL Version 15

Region Frankfurt (EU Central)

Read Replica [Add Read Replica](#) [Edit](#)

Storage 4.81% used out of 1.0 GiB

### Instance Type

Free RAM 256 MB CPU shared Storage 1 GB [Please enter your payment information to upgrade your plan.](#)

Slika 5.24: Generirani podaci za bazu podataka

Connections

Hostname ⓘ dpg-cdprm4en6mppiadfp1dg-a

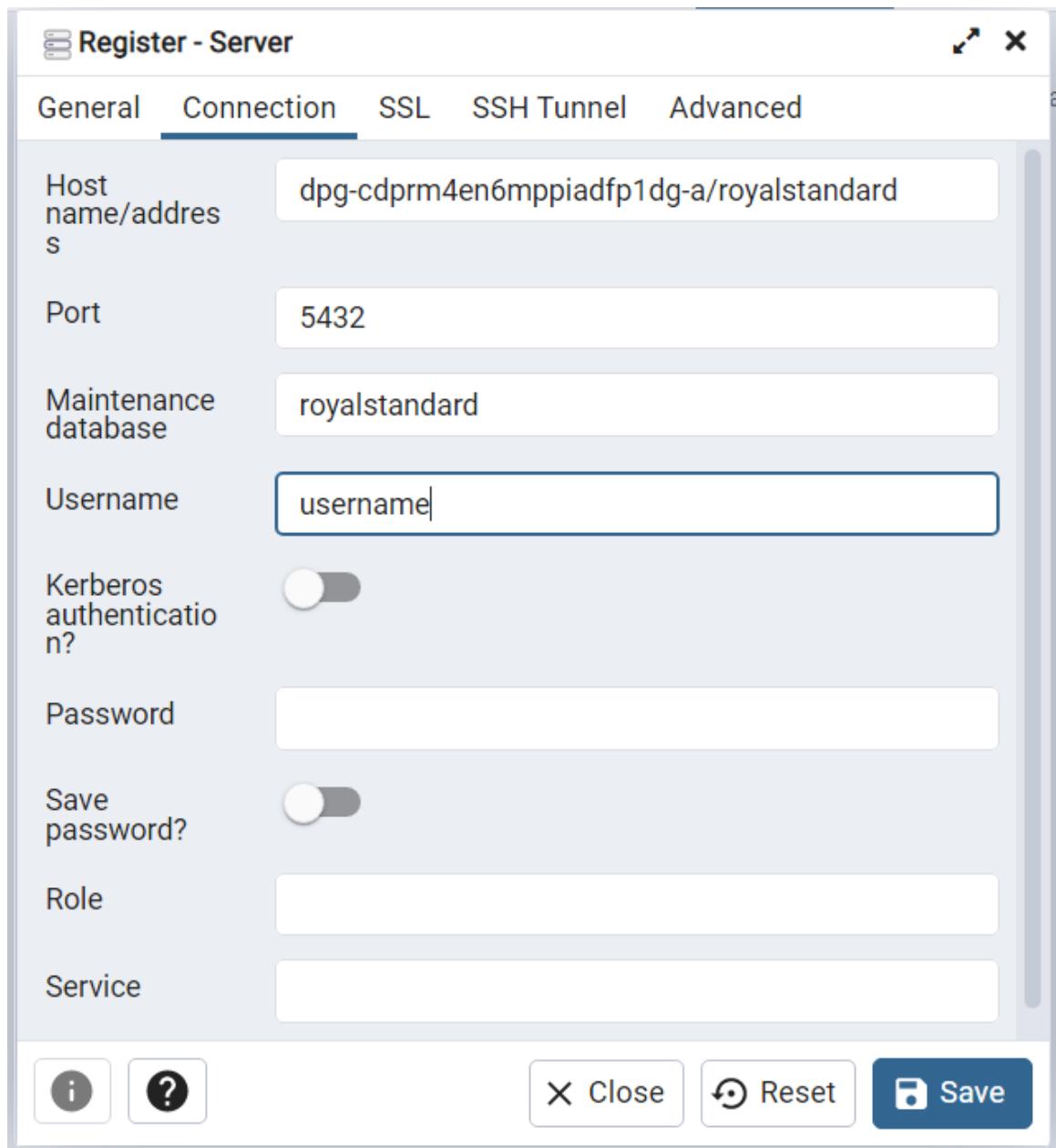
Port 5432

Database royalstandard

Username username

Password  ⌂ ⌂ .....  
Internal Database URL  ⌂ ⌂ .....  
External Database URL  ⌂ ⌂ .....  
PSQL Command  ⌂ ⌂ .....

Slika 5.25: Generirani podaci za bazu podataka



Slika 5.26: Registracija servera u sučelju PgAdmin alata

Za puštanje u pogon poslužiteljske strane u izborniku New platforme Render odabire se opcija WebService te se ostvaruje veza s GitLab repozitorijem iz kojeg se dohvaca kod. Svaki put kad dođe do promjene u repozitoriju, puštanje u pogon će se obaviti automatski ponovno. Osim ispunjavanja osnovnih podataka o poslužiteljskoj strani na Renderu, potrebno je i u kodu unijeti određene promjene. Osim Dockerfile-a, treba ažurirati i datoteku application.properties s podacima za povezivanje s bazom podataka.

Name  
A unique name for your web service.

Region  
The **region** where your web service runs. Services must be in the same region to communicate privately and you currently have services running in **Frankfurt**.

Branch  
The repository branch used for your web service.

Root Directory Optional  
Defaults to repository root. When you specify a **root directory** that is different from your repository root, Render runs all your commands in the **specified directory** and ignores changes outside the directory.

Environment  
The runtime environment for your web service.

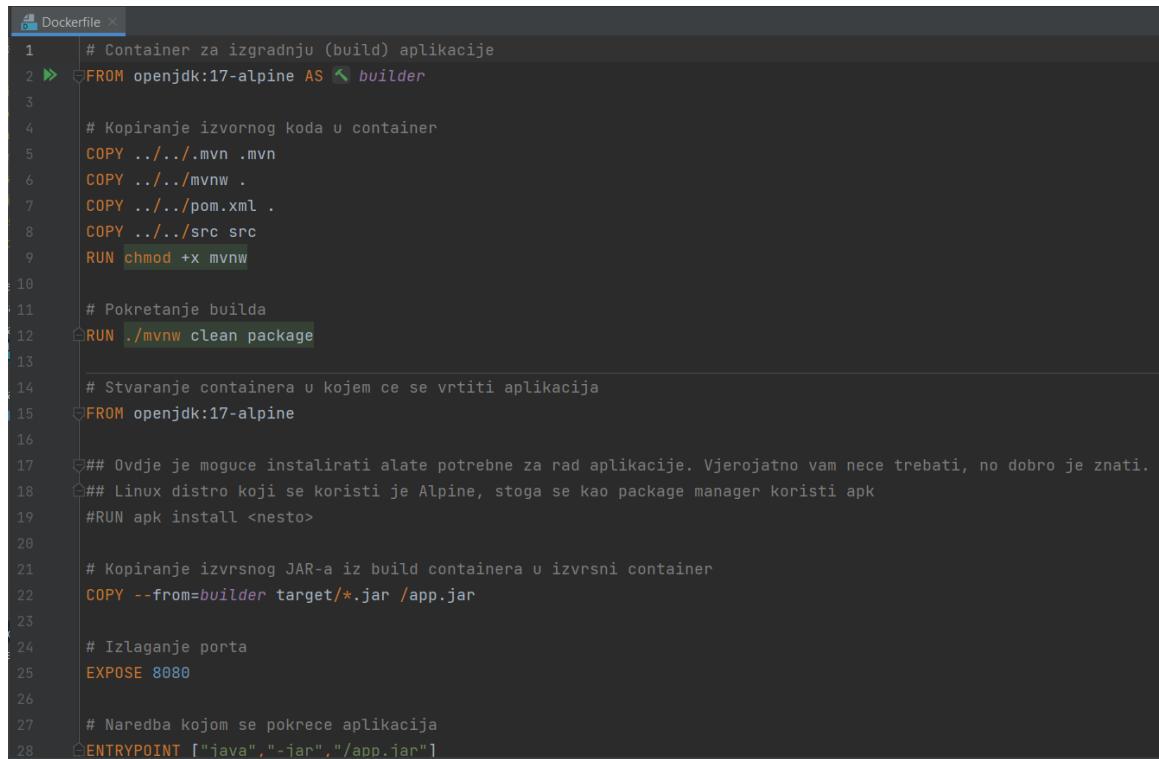
Please enter your payment information to select an instance type with higher limits.

Instance Type	RAM	CPU	Price
<input checked="" type="radio"/> Free	512 MB	Shared	\$0 / month
<input type="radio"/> Starter	512 MB	0.5 CPU	\$7 / month

Slika 5.27: Podaci za poslužiteljsku stranu

```
application.properties
1 #spring.h2.console.enabled: true
2 #spring.datasource.url: jdbc:h2:mem:testdb
3
4 server.port=8080
5 server.servlet.context-path=/api
6 spring.datasource.url: jdbc:postgresql://dpq-cdprm4en0mppiadfp1dg-a:5432/royalstandard
7 spring.datasource.username: username
8 spring.datasource.password: ****
9 spring.jpa.hibernate.ddl-auto: update
10 spring.jpa.show-sql: true
11 spring.jpa.properties.hibernate.dialect: org.hibernate.dialect.PostgreSQLDialect
12 spring.jpa.properties.hibernate.format_sql: true
13
14 pubkvizovi.team.maxNumberOfMembers: 5
15
16 ##### h2 init #####
17 #spring.sql.init.mode=always
```

Slika 5.28: Datoteka "application.properties"



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

Slika 5.29: Datoteka "Dockerfile"

Za puštanje u pogon klijentske strane odabire se opcija „WebService“ kao i u prethodnom koraku, ali se upisuju drugačiji podaci (npr. Environment je sada Node i postavljaju se „build command“ kao i „start command“). U environment variable dodaje se varijabla API-BASE-URL koja se postavlja na adresu poslužiteljske strane. U kodu se kreira datoteka „setupProxy.js“ u kojoj se zahtjevi preusmjeravaju na poslužiteljsku stranu.

**Name**  
A unique name for your web service.

**Region**  
The **region** where your web service runs. Services must be in the same region to communicate privately and you currently have services running in Frankfurt.

**Branch**  
The repository branch used for your web service.

**Root Directory** Optional  
Defaults to repository root. When you specify a **root directory** that is different from your repository root, Render runs all your commands in the specified directory and ignores changes outside the directory.

**Environment**  
The runtime environment for your web service.

**Build Command**  
This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.

**Start Command**  
This command runs in the root directory of your app and is responsible for starting its processes. It is typically used to start a webserver for your app. It can access environment variables defined by you in Render.

Slika 5.30: Podaci za klijentsku stranu

**Environment Variables**

Use environment variables to store API keys and other configuration values and secrets. You can access them in your code like regular environment variables, for example with `os.getenv()` in Python or `process.env` in Node.

Key	Value
API_BASE_URL	<input type="text" value="https://royalstandard-backend.onrender.com"/> <span style="border: 1px solid #ccc; padding: 2px;">Delete</span>

[Create Environment Group](#) Add Environment Variable Save Changes

Slika 5.31: Postavljanje varijable okruženja API-BASE-URL



```
setupProxy.js
1  const { createProxyMiddleware } = require("http-proxy-middleware");
2
3  module.exports = function (app) {
4      app.use(
5          "/api",
6          createProxyMiddleware({
7              target: "https://royalstandard-backend.onrender.com",
8              changeOrigin: true,
9          })
10     );
11 }
12 }
```

Slika 5.32: Datoteka "setupProxy.js"

## 6. Zaključak i budući rad

Aplikaciju smo osmislili kao pomoć kvizašima da pronađu događaje u svojoj blizini koji ih zanimaju i da im olakšamo prijavljivanje vlastitih ekipa na iste. Također, onima koji nemaju ekipu, a žele se okušati u nadolazećim kvizovima, omogućen je pronalazak nove ekipe kojoj korisnik odgovara svojim definiranim područjima znanja. Korisnik aplikacije može se identificirati kao igrač, sastavljač ili oboje, a sastavljačima aplikacija olakšava objavljivanje i skupljanje pozornosti na kvizove koje kreiraju.

Nakon upoznavanja tima i definiranja uloga u timu, počeli smo s radom koji se provodio u dvije faze. U prvoj fazi oblikovali smo potrebnu bazu podataka, dokumentirali funkcionalne i nefunkcionalne zahtjeve koje smo prikazali obrascima uporabe, događaje u aplikaciji smo dokumentirali sekvencijskim dijagramima, a modele objekata koji sudjeluju u aplikaciji dijagramima razreda. Opisali smo arhitekturu sustava i detaljno obrazložili naš cilj i ideju za projekt. Složili smo potreban *frontend* i *backend* koji su zadovoljavali naše zahtjeve za prvu predaju projekta.

U drugoj fazi rada svaki je član *frontend* tima radio na svojim dodijeljenim komponentama aplikacije, dok su članovi *backend* tima razvili potporu za spremanje, dohvaćanje i izmjenu podataka. Svi su članovi obogatili svoje iskustvo rada u razvojnom timu i naučili osnove programskog inženjerstva. Vremenski smo bili dobro organizirani, za što je zaslužna česta međusobna komunikacija putem društvenih mreža. Za drugu predaju smo također dokumentirali ispitivanje sustava, *deploy* i dijagrame.

Jedno od mogućih budućih funkcionalnih proširenja aplikacije, koje bi bilo dobro implementirati za što bolje korisničko iskustvo, uključuje povećanje maksimalnog broja igrača u ekipi, budući da u stvarnom životu različiti pub kvizovi mogu imati različito definiran maksimalan broj članova. Drugo moguće proširenje je implementacija *inboxa* među članovima ekipe, ali i među svim korisnicima aplikacije, jer ponekad igrače zanimaju dodatni detalji o događaju i *inbox* bi olakšao komunikaciju između igrača i sastavljača, kao i ostalo dogovaranje unutar ekipe, što bi korisnicima bilo jako povoljno jer sve vezano za kviz mogu obaviti u jednoj aplikaciji. Također, igračima koji se žele okušati u mnogo kvizova s većim spektrom

suigrača, bilo bi dobro omogućiti članstvo u više različitih ekipa.

# Popis literature

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

# Popis slika

2.1 Primjer sličnog rješenja . . . . .	9
3.1 Dijagram obrazaca uporabe, funkcionalnosti neregistriranog i registriranog korisnika . . . . .	22
3.2 Dijagram obrazaca uporabe, funkcionalnosti igrača i sastavljača kviza	23
3.3 Dijagram obrazaca uporabe, funkcionalnosti administratora . . . . .	24
3.4 Sekvencijski dijagram, registracija korisnika . . . . .	25
3.5 Sekvencijski dijagram, kreiranje događaja . . . . .	26
3.6 Sekvencijski dijagram, pronalazak ekipe . . . . .	27
4.1 Arhitektura sustava . . . . .	29
4.2 ER dijagram . . . . .	35
4.3 Dijagram razreda, dio Domain . . . . .	36
4.4 Dijagram razreda, Controllers, Services i DTO . . . . .	37
4.5 Dijagram razreda, Services i ServicesImpl . . . . .	38
4.6 Dijagram razreda, Repositories i ServicesImpl . . . . .	39
4.7 Dijagram stanja . . . . .	41
4.8 Dijagram aktivnosti . . . . .	43
4.9 Dijagram komponenti . . . . .	45
5.1 Klasa QuizServiceImplTest . . . . .	48
5.2 Testovi za listu mojih kvizova, iznimke „nema korisnika” i „nema tima” . . . . .	49
5.3 Lista kvizova, iznimka u mojim kreiranim kvizovima „nema korisnika”, lista mojih kreiranih kvizova . . . . .	49
5.4 Iznimka, „kreiran kviz sa istim imenom” . . . . .	50
5.5 Iznimka za dodavanje tima kvizu, „pun kapacitet” . . . . .	50
5.6 Rezultati testova . . . . .	51
5.7 Klasa TeamServiceImplTest . . . . .	51
5.8 Iznimke, „pun tim” i „igrač nema tim” . . . . .	52
5.9 Rezultati testova . . . . .	52

5.10 Klasa UserServiceImplTest, iznimka „nema id-a” . . . . .	53
5.11 Testovi za „dohvati id” i dohvati svih korisnika . . . . .	54
5.12 Rezultati testova . . . . .	54
5.13 Postavke testa . . . . .	55
5.14 Rezultat testa . . . . .	55
5.15 Postavke testa . . . . .	56
5.16 Rezultat testa . . . . .	56
5.17 Postavke testa . . . . .	57
5.18 Rezultat testa . . . . .	57
5.19 Postavke testa . . . . .	58
5.20 Rezultat testa . . . . .	59
5.21 Dijagram razmještaja . . . . .	60
5.22 Izbornik za kreiranje novih komponenata . . . . .	61
5.23 Upisivanje početnih podataka za bazu podataka . . . . .	62
5.24 Generirani podaci za bazu podataka . . . . .	63
5.25 Generirani podaci za bazu podataka . . . . .	64
5.26 Registracija servera u sučelju PgAdmin alata . . . . .	65
5.27 Podaci za poslužiteljsku stranu . . . . .	66
5.28 Datoteka "application.properties" . . . . .	66
5.29 Datoteka "Dockerfile" . . . . .	67
5.30 Podaci za klijentsku stranu . . . . .	68
5.31 Postavljanje varijable okruženja API-BASE-URL . . . . .	68
5.32 Datoteka "setupProxy.js" . . . . .	69
6.1 Prikaz aktivnosti za dokumentaciju . . . . .	79
6.2 Prikaz aktivnosti za frontend . . . . .	79
6.3 Prikaz aktivnosti za backend . . . . .	79

## Popis tablica

1.0 . . . . .	3
6.0 . . . . .	77

# Dodatak: Prikaz aktivnosti grupe

## Dnevnik sastajanja

### 1. sastanak

- Datum: 12. listopada 2022.
- Prisustvovali: svi
- Teme sastanka:
  - upoznavanje
  - dogovor oko prijedloga nove teme

### 2. sastanak

- Datum: 20. listopada 2022.
- Prisustvovali: svi
- Teme sastanka:
  - sastanak s asistenticom i demonstratorom
  - konačan odabir alata i tehnologija
  - upute o radu

### 3. sastanak

- Datum: 31. listopada 2022.
- Prisustvovali: svi
- Teme sastanka:
  - opis teme
  - definiranje funkcionalnih zahtjeva
  - podjela zadataka (obrasci uporabe, opis arhitekture, model baze podataka)

### 4. sastanak

- Datum: 7. studenog 2022.
- Prisustvovali: svi
- Teme sastanka:
  - sastanak s asistenticom i demonstratorom
  - provjera obrazaca uporabe

- upute o programiranju

#### 5. sastanak

- Datum: 14. studenog 2022.
- Prisustvovali: svi
- Teme sastanka:
  - sastanak s asistenticom i demonstratorom
  - pokazivanje generičkih funkcionalnosti
  - dodatne upute za dovršavanje 1. revizije

#### 6. sastanak

- Datum: 19. prosinca 2022.
- Prisustvovali: svi
- Teme sastanka:
  - prezentiranje alfa inačice
  - dogovor o dalnjem radu

#### 7. sastanak

- Datum: 5. siječnja 2023.
- Prisustvovali: svi
- Teme sastanka:
  - plan o završetku projekta
  - podjela preostalih zadataka

#### 8. sastanak

- Datum: 9. siječnja 2023.
- Prisustvovali: svi
- Teme sastanka:
  - završne promjene
  - dogovor o preostalim poslovima

## Tablica aktivnosti

	Branimir Stanković	Ivan Šetka	Luka Panić	Ivan Pejić	Marija Kompar	Karlo Šarić	Ivan Samardžić
Upravljanje projektom	5	3	2				
Opis projektnog zadatka		3					
Funkcionalni zahtjevi		1					
Opis pojedinih obrazaca		3		2		2	4
Dijagram obrazaca		2					
Sekvencijski dijagrami				1		2	4
Opis ostalih zahtjeva		0,5					
Arhitektura i dizajn sustava			4				
Baza podataka					12		
Dijagram razreda		6					
Dijagram stanja		2					
Dijagram aktivnosti		1					
Dijagram komponenti		1					
Korištene tehnologije i alati						2	
Ispitivanje programskog rješenja	8		2			1	1
Dijagram razmještaja		1					
Upute za puštanje u pogon				20			
Dnevnik sastajanja		0,3					
Zaključak i budući rad					0,4		

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Branimir Stanković		Ivan Šetka	Luka Panić	Ivan Pejić	Marija Kompar	Karlo Šarić	Ivan Samardžić
Izrada početne stranice			2,5					
Back end	30	60	17					
Front end			63	10	15	12	12	
Popis UC-ova						3		

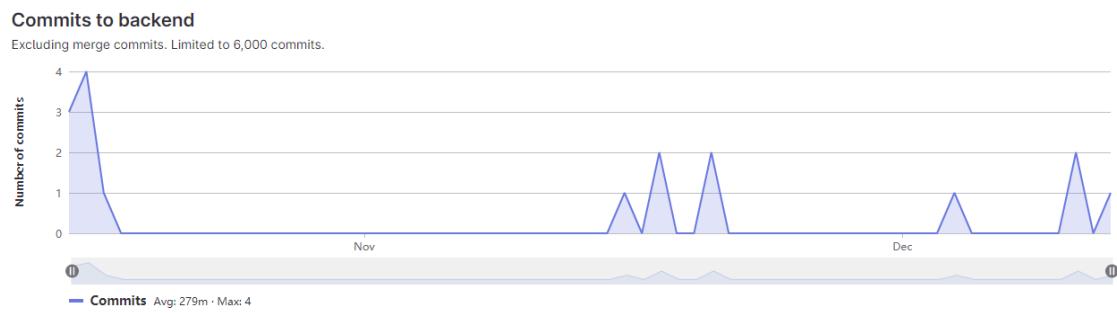
## Dijagrami pregleda promjena



Slika 6.1: Prikaz aktivnosti za dokumentaciju



Slika 6.2: Prikaz aktivnosti za frontend



Slika 6.3: Prikaz aktivnosti za backend