

Programsko inženjerstvo

Ak. god. 2023./2024.

BytePit

Dokumentacija, Rev. 1

Grupa: *Looney Codes*

Voditelj: *Vedran Ćutić*

Datum predaje: 17.11.2023.

Nastavnik: *Hrvoje Nuić*

Sadržaj

1 Dnevnik promjena dokumentacije	2
2 Opis projektnog zadatka	5
3 Specifikacija programske potpore	11
3.1 Funkcionalni zahtjevi	11
3.1.1 Obrasci uporabe	13
3.1.2 Sekvencijski dijagrami	24
3.2 Ostali zahtjevi	30
4 Arhitektura i dizajn sustava	31
4.1 Baza podataka	32
4.1.1 Opis tablica	33
4.1.2 Dijagram baze podataka	38
4.2 Dijagram razreda	39
4.3 Dijagram aktivnosti	41
4.4 Dijagram komponenti	43
5 Implementacija i korisničko sučelje	44
5.1 Korištene tehnologije i alati	44
5.2 Ispitivanje programskog rješenja	45
5.2.1 Ispitivanje komponenti	45
5.2.2 Ispitivanje sustava	49
5.3 Dijagram razmještaja	52
5.4 Upute za puštanje u pogon	53
Popis literature	57
Indeks slika i dijagrama	59
Dodatak: Prikaz aktivnosti grupe	60

1. Dnevnik promjena dokumentacije

Kontinuirano osvježavanje

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Preuzet i postavljen predložak.	Nikola Vlahović	20.10.2023.
0.2	Dodan opis projektnog zadatke. Upisani članovi.	Marko Varga	21.10.2023.
0.3	Nadopisan opis projektnog zadatka.	Marina Hrbud	22.10.2023.
0.4	Dodani funkcionalni zahtjevi	Lara Marčec	23.10.2023.
0.5	Dodani opisi obrazaca uporabe	Nikola Vlahović, Lara Marčec	29.10.2023.
0.6	Dodani dijagrami obrazaca uporabe	Jakov Novak, Marko Varga	29.10.2023.
0.7	Dodani opisi ostalih zahtjeva	Lara Marčec	30.10.2023.
0.8	Dodani sekvencijski dijagrami	Lara Marčec	2.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
0.9	Dodan opis arhitekture i dizajna sustava	Antonio Glavaš	4.11.2023.
0.10	Dodani opis i dijagram baze podataka	Marina Hrbud, Marko Varga	5.11.2023.
0.11	Ispravljen sekvencijski dijagram	Lara Marčec	10.11.2023.
0.12	Dodani dijagrami razreda	Antonio Glavaš	15.11.2023.
0.13	Upisan dnevnik sastajanja	Marina Hrbud	15.11.2023
0.14	Promjene u arhitekturi i dizajnu sustava	Antonio Glavaš	17.11.2023.
0.15	Revizija dijagrama i opisa baze podataka	Vedran Ćutić, Nikola Vlahović	17.11.2023.
1.0	Dodan dijagram razmještaja	Antonio Glavaš	10.12.2023.
1.1	Dodan dijagram komponenti	Vedran Ćutić	10.12.2023.
1.2	Dodan dijagram aktivnosti i ispravljen dio grešaka iz prve predaje	Lara Marčec	10.12.2023.
1.3	Ažuriran dijagram razmještaja	Antonio Glavaš	13.1.2024.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodataka	Autori	Datum
1.4	Dodano potpoglavlje "Korištene tehnologije i alati"	Marina Hrbud	15.1.2024.
1.5	Ažuriran dijagram razreda	Antonio Glavaš	15.1.2024.
1.6	Dodano potpoglavlje "Upute za puštanje u pogon"	Jakov Novak, Marko Varga	15.1.2024.

2. Opis projektnog zadatka

Projektni zadatak jest razvoj programske podrške za ostvarenje mrežne aplikacije BytePit. Aplikacija predstavlja platformu za rješavanje programerskih zadataka i provođenje natjecanja u programiranju. Aplikacija BytePit pruža ključne mogućnosti kao što su:

- *registracija natjecatelja i voditelja natjecanja*
- *dijeljenje osmišljenih zadataka iz natjecateljskog programiranja*
- *rješavanje zadataka i evaluacija rješenja na temelju testnih primjera*
- *kreiranje i provođenje natjecanja.*

Zbog ovih je mogućnosti aplikacija BytePit od posebnog interesa za odgojno-obrazovne ustanove koje alat mogu koristiti kao dodatak u nastavi predmeta povezanih uz programiranje (provodjenje ispita ili domaćih zadaća), sudionike natjecanja u programiranju koji na platformi mogu u slobodno vrijeme vježbati i usavršavati svoje vještine te udruge i druga tijela koja se bave organizacijom natjecanja iz programiranja ili pak promocijom računalne znanosti. IT kompanije bi ovu aplikaciju mogle upotrebljavati kako bi ispitale relevantna znanja i programerske sposobnosti potencijalnih novih zaposlenika.

Aplikacija na početnoj stranici mora nuditi mogućnost registracije novim korisnicima. Proces registracije, uz mogućnost odabira uloge (voditelj natjecanja ili natjecatelj), zahtijeva unos korisničkog imena, fotografije, lozinke, imena, prezimena i adrese e-pošte. Za završetak registracije potrebno je otvoriti poveznicu dobivenu putem e-pošte. Voditeljevu registraciju mora dodatno odobriti administrator, koji pored toga ima ovlasti za upravljanje registriranim korisnicima i njihovim podacima, uključujući dodjelu prava i promjene osobnih podataka.

Tri su osnovna tipa korisnika aplikacije:

- **natjecatelji** — Natjecatelji su osnovni korisnici aplikacije i njezina glavna ciljna skupina. Osnovne usluge koje im se pružaju su sudjelovanje u programerskim natjecanjima, rješavanje zadataka i praćenje svog napretka. Na zahjev mogu pristupiti zadacima (tijekom ili izvan termina natjecanja) . Mogu

slati programska rješenja na evaluaciju, a nakon natjecanja pratiti i rangiranje s obzirom na ostale natjecatelje. Na profilima natjecatelja, vidljive su različite statistike, uključujući broj točno riješenih zadataka, broj pokušaja rješavanja zadataka te prikaz osvojenih pehara za natjecanja. Natjecatelji mogu organizirati i virtualna natjecanja, koja se temelje na prošlim natjecanjima u kalendaru. Ta natjecanja su aktivna samo za njih, a rangiraju se prema službenim rezultatima originalnih natjecanja. Također, mogu kreirati virtualna natjecanja s nasumično odabranim zadacima, ravnomjerno raspoređenim prema težini zadataka.

- **voditelji** — Voditelji natjecanja organiziraju i upravljaju programerskim natjecanjima na platformi. To uključuje postavljanje natjecanja, dodavanje zadataka i praćenje rezultata sudionika. Prilikom kreiranja novih natjecanja moraju postaviti parametre, kao što su datum objave i trajanje natjecanja, dodati zadatke i po želji slike pehara. Dodavanje programerskih zadataka u natjecanja uključuje i postavljanje njihovih karakteristika – bodova i vremenskog ograničenja, ali i testnih primjera. Profili voditelja sadrže popis zadataka koje su učitali, s mogućnošću sortiranja.
- **administratori** — Administratori imaju ovlasti za upravljanje cijelom platformom, uključujući i korisničkim računima. Upravljanje registriranim korisnicima i njihovim podacima uključuje dodjelu prava (primjerice potvrdu registracije voditelja) i promjene osobnih podataka korisnika. Administrator ima ovlasti za uređivanje svih zadataka i natjecanja pri čemu se ne mijenjaju prethodno ostvareni rezultati.

Središnji dijelovi aplikacije su *zadaci* i *natjecanja*. Programerski zadaci su ključna komponenta aplikacije i pružaju izazove korisnicima koji žele razvijati svoje programerske sposobnosti. Svaki zadatak dolazi s nazivom, jedinstvenim opisom, brojem bodova koji se mogu osvojiti i vremenskim ograničenjem za rješavanje. Zadatak može biti postavljen kao privatni ili javan, a svaki privatni zadatak dodan u natjecanje, po završetku istog, postaje javan. Natjecatelji biraju zadatke koji ih zanimaju (ili koji su dio natjecanja) i šalju svoje rješenje u obliku izvornog programskog koda. Evaluacija rješenja (omogućena samo za jedan programski jezik) temelji se na usporedbi generiranih izlaza programa s očekivanim izlazima iz testnih primjera. Dodijeljeni broj bodova proporcionalan je postotku točnih primjera. Za svaki zadatak dostupan je popis svih natjecatelja koji su poslali rješenje za taj zadatak.

tak. Ovaj popis sadrži informacije o broju točno riješenih primjera, prosječnom vremenom izvršavanja po primjeru te omogućava pristup učitanim rješenjima. Važno je napomenuti da opcija za pristup učitanim rješenjima aktivira samo za one natjecatelje koji su uspješno i potpuno riješili zadatak.

Natjecanja predstavljaju srce platforme, potičući korisnike da pokažu svoje programerske vještine i natječe se s drugima. Voditelji natjecanja imaju ključnu ulogu u organizaciji ovih događaja. Oni kreiraju i definiraju ključne elemente natjecanja. Natjecatelji se prijavljuju za sudjelovanje u natjecanjima, a tijekom trajanja rješavaju zadatke i šalju svoja rješenja. Početkom natjecanja prijavljeni natjecatelji mogu pristupiti svim zadacima koji se nalaze u njemu. Nakon završetka natjecanja, rezultati se prikazuju na profilima natjecatelja, a osvajači pehara nagrađuju se za svoje uspjehe. Također, natjecatelji imaju mogućnost pregledavanja svih rješenja koja su drugi natjecatelji poslali za isti zadatak.

Rješenje za ovu aplikaciju biti će ostvareno pomoću razvojnog okvira Springboota i JavaScript biblioteka React kojom će biti modelirano korisničko sučelje.

Moguća poboljšanja za aplikaciju BytePit uključuju unaprjeđenje korisničkog sučelja radi olakšane navigacije, uvođenje ocjena i komentara za zadatke i natjecanja, implementaciju sustava za korisničku komunikaciju putem chatova ili foruma, bolje statističke podatke i analitiku na korisničkim profilima, integraciju s razvojnim alatima, raznolikost vrsta zadataka, napredniju evaluaciju rješenja s podrškom za različite programske jezike, te dodatak edukativnih materijala kako bi se potaknulo učenje i razvoj programerskih vještina među korisnicima.

Primjeri sličnih rješenja kroz koje ćemo proći su:

- *Edgar*
- *Sphere online judge*
- *Codeforces*

Kao i u sustavima *Edgar*, *SPOJ* i *Codeforces*, korisnici će moći pregledavati zadatke na stranici, te će moći na svom profilu vidjeti statistiku u odnosu na preostale natjecatelje, te svoj rank. Također mogu na kalendaru vidjeti vrijeme odvijanja budućih natjecanja.

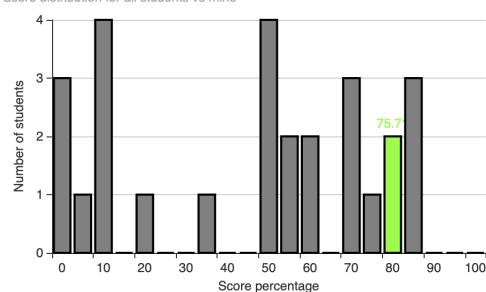
list of classical problems

ID	NAME	QUALITY	USERS	ACC %	DIFFICULTY C I
1	Life, the Universe, and Everything	↳ 294	207076	32.89	1 15 15
2	Prime Generator	↳ 616	75846	15.94	15 15
3	Substring Check (Bug Funny)	↳ 9	1057	7.99	41 15
4	Transform the Expression	↳ 257	41908	49.17	15 15
5	The Next Palindrome	↳ 450	19715	10.82	25 18
6	Simple Arithmetics	↳ 10	2978	12.49	30 9
7	The Bulk!	↳ 6	476	16.32	30 24
8	Complete the Sequence!	↳ 30	4249	38.15	16 19
9	Direct Visibility	↳ 4	247	12.48	30 21
10	Complicated Expressions	↳ 11	1515	32.43	16 17
11	Factorial	↳ 171	56448	44.98	15 15
12	The Game of Master-Mind	↳ 1	717	49.89	17 17

Slika 2.1: Sustav SPOJ - pregled zadataka na stranici

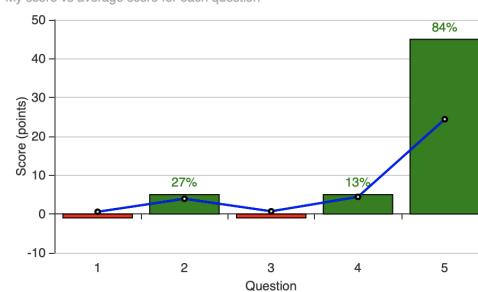
Exam score percentage distribution (n = 31)

Score distribution for all students vs mine



Per Question

My score vs average score for each question



Rank

#6 /31

83%

Percentile rank

Score

53 /70

76%

Percentage

Slika 2.2: Sustav Edgar - pregled statistike ispita

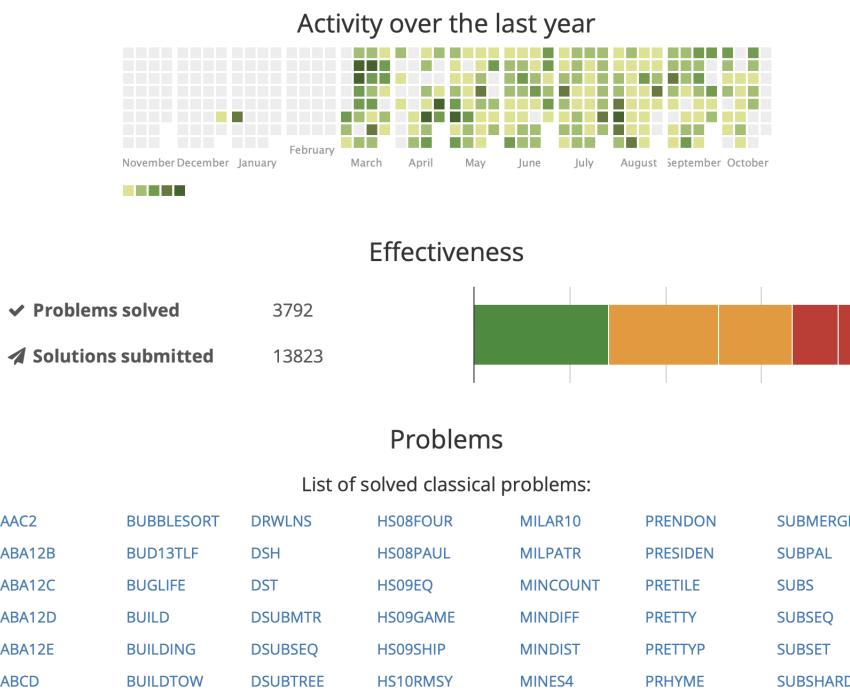
MAIN	ACMSGURU	PROBLEMS	SUBMIT	STATUS	STANDINGS	CUSTOM TEST
Contest status						
#	When	Who	Problem	Lang	Verdict	Time
229301890	Oct/22/2023 16:35UTC+2	-JAAT-	1883A - Morning	GNU C++17	Accepted	31 ms 0 KB
229301889	Oct/22/2023 16:35UTC+2	Part_time_Ray	1850C - Word on the Paper	GNU C++17 (64)	Accepted	15 ms 0 KB
229301887	Oct/22/2023 16:35UTC+2	bkifhr9	1873B - Good Kid	GNU C++17	Time limit exceeded on test 1	1000 ms 0 KB
229301886	Oct/22/2023 16:35UTC+2	manish36	486A - Calculating Function	GNU C++17	Accepted	15 ms 0 KB
229301885	Oct/22/2023 16:35UTC+2	regain0002	1888D2 - Dances (Hard Version)	GNU C++17	Wrong answer on test 2	46 ms 37600 KB
229301883	Oct/22/2023 16:35UTC+2	P.N.V.Sumanasree	50A - Domino piling	GNU C++20 (64)	Accepted	30 ms 0 KB
229301882	Oct/22/2023 16:35UTC+2	_SAK_	1883E - Look Back	GNU C++17	Accepted	93 ms 1200 KB
229301881	Oct/22/2023 16:35UTC+2	ValeriaDubinets	1808A - Lucky Numbers	GNU C++20 (64)	Time limit exceeded on test 2	1000 ms 0 KB
229301880	Oct/22/2023 16:35UTC+2	Huihuilucky	1791C - Prepend and Append	GNU C++17	Accepted	31 ms 0 KB
229301879	Oct/22/2023 16:35UTC+2	xgyxgy	803B - Distances to Zero	GNU C++20 (64)	Wrong answer on test 13	15 ms 800 KB
229301878	Oct/22/2023 16:35UTC+2	vjudge3	1538C - Number of Pairs	Clang++17 Diagnostics	Wrong answer on test 2	514 ms 2100 KB
229301876	Oct/22/2023 16:35UTC+2	imaginesabab	520A - Pangram	GNU C++17	Accepted	15 ms 0 KB
229301875	Oct/22/2023 16:35UTC+2	gdaaay	1883C - Raspberries	GNU C++20 (64)	Accepted	15 ms 0 KB
229301874	Oct/22/2023 16:35UTC+2	rohitrojo	1863A - Channel	GNU C++20 (64)	Wrong answer on test 1	15 ms 0 KB
229301873	Oct/22/2023 16:35UTC+2	3inf	1884C - Medium Design	GNU C++17	Accepted	499 ms 7300 KB

Slika 2.3: Sustav Codeforces - pregled statusa na natjecanju

Calendar								
Programming Contests Calendar								
Today	◀	▶	October 2023	▼	Print	Week	Month	Agenda ▾
Sun	Mon	Tue	Wed	Thu	Fri	Sat		
Oct 1	2	3	4	5	6	7		
8 11:05am CF R (Div. 1, based o 11:05am CF R 902	9 4:35pm Educational CF R 156	10 8:15pm Stream: Teaching Cha		11 4:35pm CF R 903 (Div. 3)	12	13		
						14		
15 9:45pm Stream: No code CF	16	17	18	19	20	21		
22 9:05am CF R 904 (Div. 2) 1:05pm CF R 905	23	24	25	26	27 4:35pm CF R	28		
29 3:35pm CF R (Div. 2)	30	31	Nov 1	2	3	4		
Events shown in time zone: Central European Time - Belgrade								
+ Google Calendar								

Slika 2.4: Sustav Codeforces - kalendar nadolazećih natjecanja

Ukoliko natjecatelja zanima profil nekog drugog natjecatelja, klikom na ime profila će se otvoriti stranica na kojoj je moguće vidjeti statistiku natjecatelja: broj točno rješenih zadataka, pehari, broj svih pokrenutih zadataka. *Codeforces* dodatno daje prikaz aktivnosti natjecatelja u vremenskom periodu od jedne godine.



Slika 2.5: Sustav SPOJ - profil natjecatelja

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Vlasnik/naručitelj
2. Voditelji
3. Natjecatelji
4. Administrator
5. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani/neprijavljeni korisnik (inicijator) može:
 - (a) vidjeti kalendar s budućim natjecanjima
 - (b) pregledati dostupne zadatke na stranici
 - (c) pregledati profile natjecatelja i voditelja
 - (d) registrirati se u sustav stvaranjem novog korisničkog računa pri čemu odabire jednu od uloga (natjecatelj ili voditelj), a potrebni su mu: korisničko ime, fotografija, lozinka, ime, prezime te email adresa
2. Natjecatelj (inicijator) može:
 - (a) sudjelovati na natjecanju
 - (b) vidjeti rang listu natjecatelja na natjecanju kojeg je i sam bio sudionik
 - (c) vidjeti popis svih učitanih rješenja od ostalih sudionika za prethodno završena natjecanja
 - (d) pristupiti rješavanju već objavljenih zadataka
 - (e) izraditi virtualno natjecanje odabirom nekog prošlog natjecanja ili nsumičnim generiranjem zadataka
 - (f) vidjeti vlastiti profil s osobnim podacima, statistikom o broju točno riješenih zadataka, broju isprobanih zadataka te prikaz pehara za osvojena natjecanja

3. Aktivni natjecatelj (inicijator) može:

- (a) rješavati zadatke i slati datoteke s programskim kodom tijekom natjecanja na kojem se natječe
- (b) osvojiti bodove na natjecanju s obzirom na potrošeno vrijeme za rješavanje zadatka i postotak točnih primjera
- (c) na temelju postignuća za prva tri mesta osvojiti pehar koji je vidljiv na vlastitom profilu

4. Voditelj (inicijator) može:

- (a) pregledati dostupne zadatke na stranici
- (b) izraditi novi zadatak pri čemu treba definirati: naziv zadatka, broj boda, vremensko ograničenje, tekst zadatka i primjere za evaluaciju
- (c) organizirati novo natjecanje pri čemu treba definirati: vrijeme početka i završetka, broj zadataka, koji zadaci će biti aktivni te po želji sličicu pehara
- (d) uređivati vlastite prethodno objavljene zadatke te natjecanja
- (e) vidjeti vlastiti profil s osobnim podacima, popisom učitanih zadataka s mogućnošću sortiranja te kalendar s popisom objavljenih natjecanja

5. Administrator (inicijator) može:

- (a) uređivati sve zadatke i natjecanja
- (b) potvrditi voditelja prilikom registracije
- (c) vidjeti popis svih registriranih korisnika i njihovih osobnih podataka
- (d) mijenjati dodijeljena prava i osobne podatke

6. Baza podataka (sudionik):

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima te zadacima i natjecanjima
- (b) pohranjuje rezultate natjecanja, rješenja zadataka i statistiku natjecatelja

3.1.1 Obrasci uporabe

UC1 - Pregled kalendarja

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati kalendar s nadolazećim natjecanjima
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Klijent otvara početnu stranicu web aplikacije
 2. Prikazuje se kalendar
 3. Klijent odabire određeni datum
 4. Aplikacija prikazuje popis nadolazećih natjecanja za odabrani datum
 5. Klijent odabire natjecanje
 6. Prikazuju se detalji i informacije o odabranom natjecanju

UC2 - Pregled zadataka

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati zadatke završenih natjecanja
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Klijent odabire završeno natjecanje
 2. Prikazuje se lista zadatka povezana s odabranim natjecanjem
 3. Klijent odabire specifičan zadatak
 4. Aplikacija prikazuje detalje zadatka

UC3 - Pregled profila natjecatelja i voditelja

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati profile natjecatelja i voditelja
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Klijent otvara profil određenog korisnika klikom na njihovo korisničko ime
 2. Ako je otvoren profil natjecatelja, aplikacija prikazuje informacije o broju točno riješenih zadataka, broju isprobanih zadataka te osvojene pehare

3. Ako je otvoren profil voditelja, aplikacija prikazuje popis učitanih zadataka i kalendar s popisom objavljenih natjecanja

UC4 - Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvoriti korisnički račun
- **Sudionici:** baza podataka, administrator
- **Preduvjet:** korisnik nije prethodno registriran ili prijavljen u sustav
- **Opis osnovnog tijeka:**
 1. Neregistrirani korisnik odabire opciju za registraciju
 2. Neregistrirani korisnik popunjava obrazac za registraciju s potrebnim podacima
 3. Neregistrirani korisnik na svoju e-mail adresu prima obavijest i zahtjev za potvrdu registracije
 4. Ako je odabrana uloga "voditelj", sustav dodatno šalje obavijest administratoru
 5. Administrator potvrđuje registraciju voditelja
- **Opis mogućih odstupanja:**
 - 2.a Već zauzeto korisničko ime ili e-mail adresa, uneseni podaci u nedozvoljenom formatu ili neispravna e-mail adresa
 1. Sustav obavještava korisnika o neuspjelom unosu i prikazuje relevantne poruke o greškama
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od registracije
 - 5.a Administrator odbija zahtjev za registraciju voditelja natjecanja:
 1. Sustav obavještava korisnika putem e-maila

UC5 - Prijava

- **Glavni sudionik:** Neprijavljeni korisnik
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** baza podataka
- **Preduvjet:** korisnik je registriran u sustav, ali nije prijavljen
- **Opis osnovnog tijeka:**
 1. Korisnik unosi korisničko ime i lozinku
 2. Sustav potvrđuje ispravnosti unesenih podataka
 3. Korisniku je omogućen pristup korisničkim funkcijama

- **Opis mogućih odstupanja:**

2.a Neispravno korisničko ime/lozinka

1. Sustav obavještava korisnika o neuspjeloj prijavi uz informaciju o pogrešci i omogućuje korisniku ponovan pokušaj prijave

UC6 - Pregled profila

- **Glavni sudionik:** Prijavljeni korisnik

- **Cilj:** Pregledati vlastiti profil

- **Sudionici:** baza podataka

- **Preduvjet:** korisnik je prijavljen

- **Opis osnovnog tijeka:**

1. Prijavljeni korisnik pristupa opciji "Moj profil" klikom na ikonu profila
2. Aplikacija prikazuje podatke o prijavljenom korisniku (ime, fotografija, lozinka, ime, prezime i e-mail adresa)

UC7 - Pregled vlastite statistike

- **Glavni sudionik:** Natjecatelj

- **Cilj:** Pregledati vlastitu statistiku unutar web aplikacije

- **Sudionici:** baza podataka

- **Preduvjet:** korisnik je prijavljen

- **Opis osnovnog tijeka:**

1. Korisnik pristupa svojem profilu i odabire opciju "Moje statistike"
2. Aplikacija prikazuje statistike korisnika(broj točno riješenih zadataka, broj isprobanih zadataka, pehare za osvojena natjecanja)

UC8 - Pregled vlastitih zadataka

- **Glavni sudionik:** Voditelj

- **Cilj:** Pregledati popis svih vlastito objavljenih zadataka

- **Sudionici:** baza podataka

- **Preduvjet:** korisnik je prijavljen

- **Opis osnovnog tijeka:**

1. Voditelj odabire opciju za prikaz vlastitih zadataka
2. Prikazuju se svi objavljeni zadaci tog voditelja

UC9 – Sudjelovanje na natjecanju

- **Glavni sudionik:** Natjecatelj

- **Cilj:** Pristupiti natjecanju
- **Sudionici:** baza podataka
- **Preduvjet:** korisnik je prijavljen i postoji natjecanje u tijeku
- **Opis osnovnog tijeka:**
 1. Prikazuju se informacije o natjecanju u tijeku i natjecatelj potvrđuje da želi sudjelovati na natjecanju
 2. Natjecatelju se prikazuju zadaci koji su dio natjecanja te preostalo vrijeme do kraja natjecanja

UC10 – Prijenos datoteke rješenja

- **Glavni sudionik:** Natjecatelj, Aktivni natjecatelj
- **Cilj:** Prenijeti datoteku kao rješenje nekog zadatka na natjecanju
- **Sudionici:** baza podataka
- **Preduvjet:** natjecatelj je pristupio natjecanju ili rješavanju zadatka
- **Opis osnovnog tijeka:**
 1. Korisnik nakon napisanog rješenja zadatka odabire opciju za predaju rješenja
 2. Korisnik odabire datoteku s rješenjem koju želi predati i potvrđuje odabir
 3. Datoteka se prenosi, predaje kao rješenje i šalje na evaluaciju
- **Opis mogućih odstupanja:**
 - 2.a Odabir pogrešne datoteke
 1. Aktivni natjecatelj odabire opciju za brisanje datoteke
 2. Aktivni natjecatelj ponovno kreće s 1. korakom iz osnovnog tijeka

UC11 – Pregled rang liste

- **Glavni sudionik:** Natjecatelj
- **Cilj:** Pregledati rang listu nekog natjecanja
- **Sudionici:** baza podataka
- **Preduvjet:** korisnik je prijavljen kao natjecatelj
- **Opis osnovnog tijeka:**
 1. Natjecatelj odabire opciju za prikaz prošlih natjecanja na kojima je sudjelovao
 2. Odabire natjecanje za koje želi vidjeti rang listu ostalih sudionika i odabire prikaz rang liste

3. Otvara se popis s prikazom postignuća ostalih sudionika po broju ostvarenih bodova

UC12 – Pregled rješenja zadataka

- **Glavni sudionik:** Natjecatelj
- **Cilj:** Vidjeti tuđa rješenja zadataka s nekog natjecanja
- **Sudionici:** Baza podataka
- **Preduvjet:** Natjecatelj je sudjelovao na natjecanju koje je završilo
- **Opis osnovnog tijeka:**
 1. Natjecatelj odabire opciju za prikaz prošlih natjecanja na kojima je sudjelovao
 2. Natjecatelj odabire natjecanje za koje želi vidjeti predana rješenja
 3. Prikazuju se informacije o natjecanju s popisom svih predanih rješenja
 4. Odabirom opcije za prikaz natjecanja po zadacima prikazuje se pregled zadataka s tog natjecanja
 5. Odabirom zadatka otvara se tekst zadatka s prikazom svih predanih rješenja drugih sudionika uz informacije o uspješnosti predanog rješenja

UC13 – Rješavanje zadataka

- **Glavni sudionik:** Natjecatelj
- **Cilj:** Vježbanje već objavljenih zadataka
- **Sudionici:** baza podataka
- **Preduvjet:** korisnik je prijavljen kao natjecatelj
- **Opis osnovnog tijeka:**
 1. Natjecatelj odabire opciju za prikaz svih objavljenih zadataka
 2. Prikazuje se popis svih zadataka
 3. Odabirom zadatka koji želi rješavati otvara se sučelje slično onom na natjecanju koje natjecatelju omogućuje učitavanje datoteke kao rješenja zadataka i evaluiranje istog

UC14 – Izrada virtualnog natjecanja

- **Glavni sudionik:** Natjecatelj
- **Cilj:** Vježbanje simuliranjem pravog natjecanja
- **Sudionici:** baza podataka
- **Preduvjet:** postoji barem jedno završeno natjecanje i/ili barem jedan javno vidljiv zadatak u bazi

- **Opis osnovnog tijeka:**
 1. Natjecatelj odabire opciju "virtualno natjecanje"
 2. Otvara se prikaz s dvije mogućnosti odabira
 3. Natjecatelj odabire jednu od dvije opcije - stvaranje natjecanja pokretanjem nekog prošlog natjecanja iz kalendara ili stvaranje natjecanja nsumičnim odabirom zadatka iz baze zadatka
 4. Natjecanje se stvara i natjecatelj može krenuti s rješavanjem zadatka

UC15 – Unos novog zadatka

- **Glavni sudionik:** Voditelj
- **Cilj:** Stvoriti novi zadatak u bazi postojećih zadataka
- **Sudionici:** baza podataka
- **Preduvjet:** korisnik je prijavljen kao voditelj
- **Opis osnovnog tijeka:**
 1. Voditelj odabire opciju za unos novog zadatka
 2. Otvara se forma gdje voditelj upisuje potrebne podatke
 3. Voditelj odabire želi li zadatak stvoriti kao privatni ili javan
 4. Voditelj potvrđuje unos zadatka
 5. Zadatak se pohranjuje u bazu ostalih zadataka s informacijom o autoru zadatka
- **Opis mogućih odstupanja:**
 - 2.a Voditelj ostavlja prazno neko polje u formi i pokušava predati takav zadatak
 1. Sustav ga obavještava o neispravnom pokušaju predaje zadatka
 2. Sustav omogućuje ponovno ispunjavanje forme u svrhu ispravne predaje

UC16 – Uređivanje zadatka

- **Glavni sudionik:** Voditelj
- **Cilj:** Uređivanje postojećeg zadatka
- **Sudionici:** baza podataka
- **Preduvjet:** postoji zadatak koji je prijavljeni voditelj unio u bazu
- **Opis osnovnog tijeka:**
 1. Voditelj odabire opciju "moji zadaci"
 2. Otvara se pregled svih zadataka koje je voditelj unio u sustav
 3. Voditelj odabire opciju uređivanja zadataka

4. Otvara se forma slična onoj kod unosa novog zadatka koja voditelju omogućuje izmjenu podataka te spremanje istih

UC17 – Organiziranje natjecanja

- **Glavni sudionik:** Voditelj
- **Cilj:** Organizirati novo natjecanje
- **Sudionici:** baza podataka
- **Preduvjet:** korisnik je prijavljen
- **Opis osnovnog tijeka:**
 1. Voditelj odabire opciju za organiziranje novog natjecanja
 2. Otvara se forma gdje voditelj odabire vrijeme početka i završetka natjecanja, broj zadataka, koji zadaci će biti aktivni te po želji učitava sličicu pehara
 3. Voditelj potvrđuje podatke o novom natjecanju
 4. Natjecanje se dodaje u kalendar natjecanja
- **Opis mogućih odstupanja:**
 - 2.a Voditelj ne ispunjava neko polje u formi i pokušava potvrditi natjecanje
 1. Sustav ga obavještava o neispravnosti ispunjene forme
 2. Sustav omogućuje ponovno ispunjavanje forme u svrhu ispravne predaje

UC18 – Uređivanje natjecanja

- **Glavni sudionik:** Voditelj
- **Cilj:** Uređivanje postojećeg natjecanja
- **Sudionici:** baza podataka
- **Preduvjet:** postoji natjecanje koje je prijavljeni voditelj organizirao, a ono nije u tijeku ili završeno
- **Opis osnovnog tijeka:**
 1. Voditelj odabire opciju "moja natjecanja"
 2. Otvara se pregled svih natjecanja koje je voditelj organizirao
 3. Voditelj odabire opciju uređivanja natjecanja kojemu želi izmijeniti podatke
 4. Otvara se pregled sličan onom prilikom organiziranja novog natjecanja koji voditelju omogućuje izmjenu i spremanje novih postavki natjecanja

UC19 – Pregled svih korisnika u bazi

- **Glavni sudionik:** Administrator
- **Cilj:** Pregled svih registriranih korisnika u bazi
- **Sudionici:** baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju prikaza korisnika u bazi
 2. Otvara se popis svih registriranih BytePit korisnika

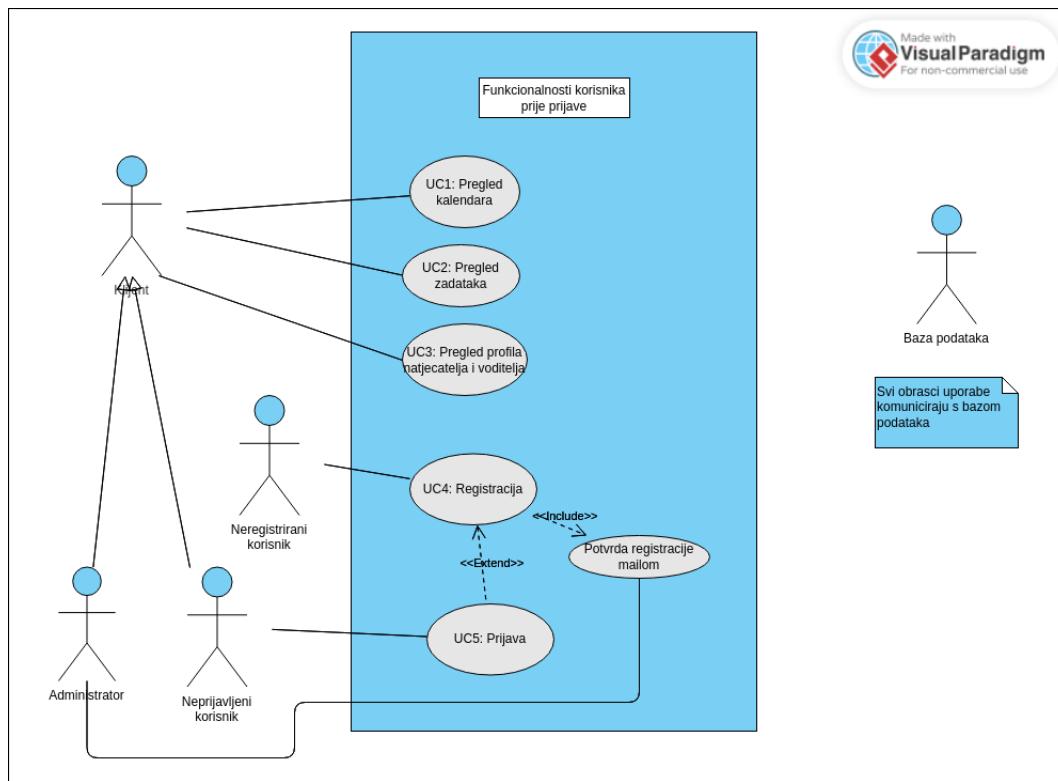
UC20 – Izmjena osobnih podataka

- **Glavni sudionik:** Administrator
- **Cilj:** Uređivanje podataka nekog korisnika
- **Sudionici:** baza podataka
- **Preduvjet:** postoji barem jedan registrirani korisnik u bazi
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju prikaza korisnika u bazi
 2. Otvara se popis svih registriranih BytePit korisnika
 3. Odabirom korisnika prikazuju se njegovi osobni podaci s mogućnošću izmjene istih uključujući i izmjenu dodijeljene uloge korisniku

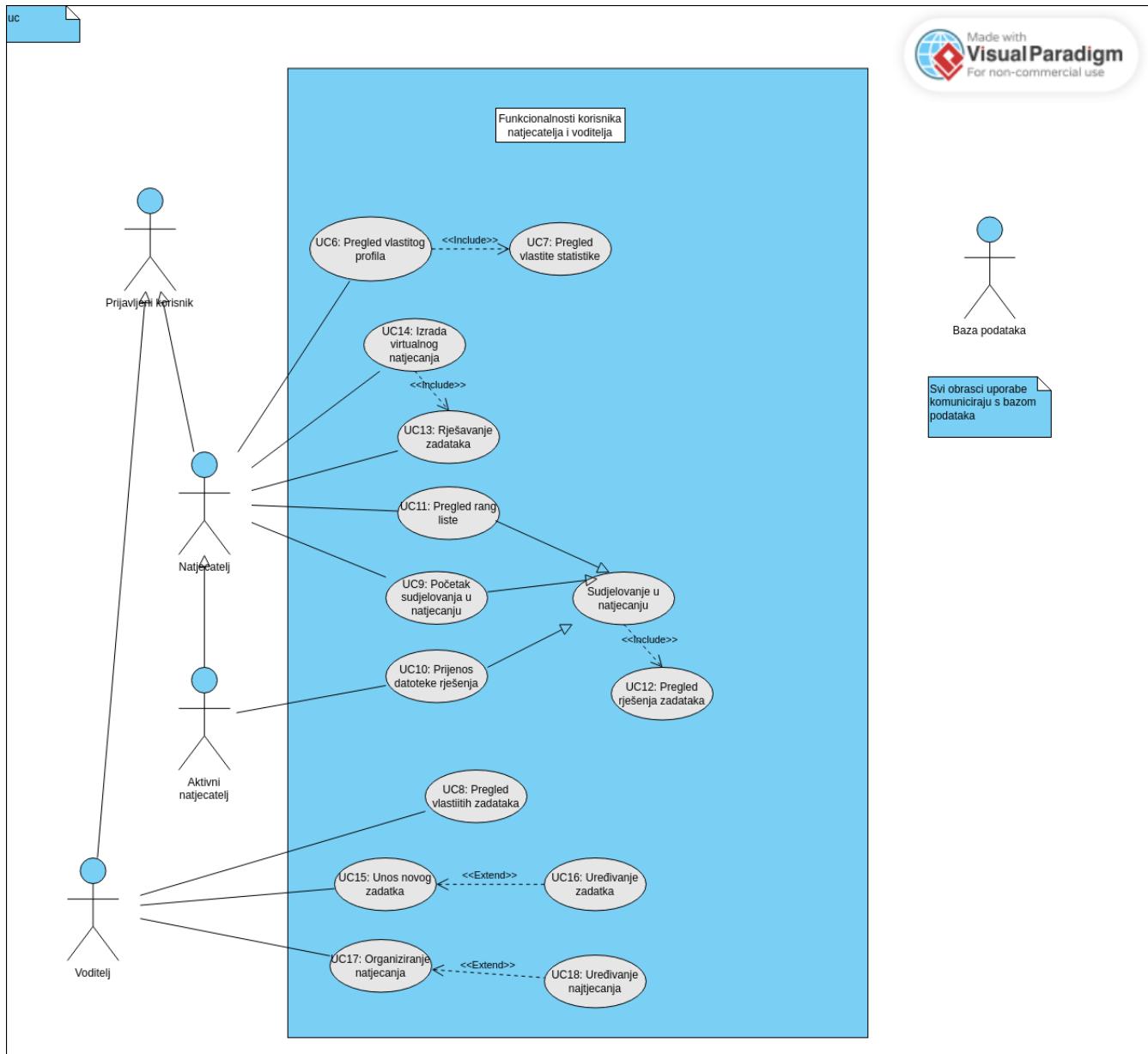
UC21 – Brisanje korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Brisanje postojećeg korisnika iz baze podataka
- **Sudionici:** baza podataka
- **Preduvjet:** postoji barem jedan registrirani korisnik u bazi
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju prikaza korisnika u bazi
 2. Otvara se popis svih registriranih BytePit korisnika
 3. Administrator odabire i potvrđuje opciju brisanja korisnika
 4. Korisnik se briše iz baze podataka

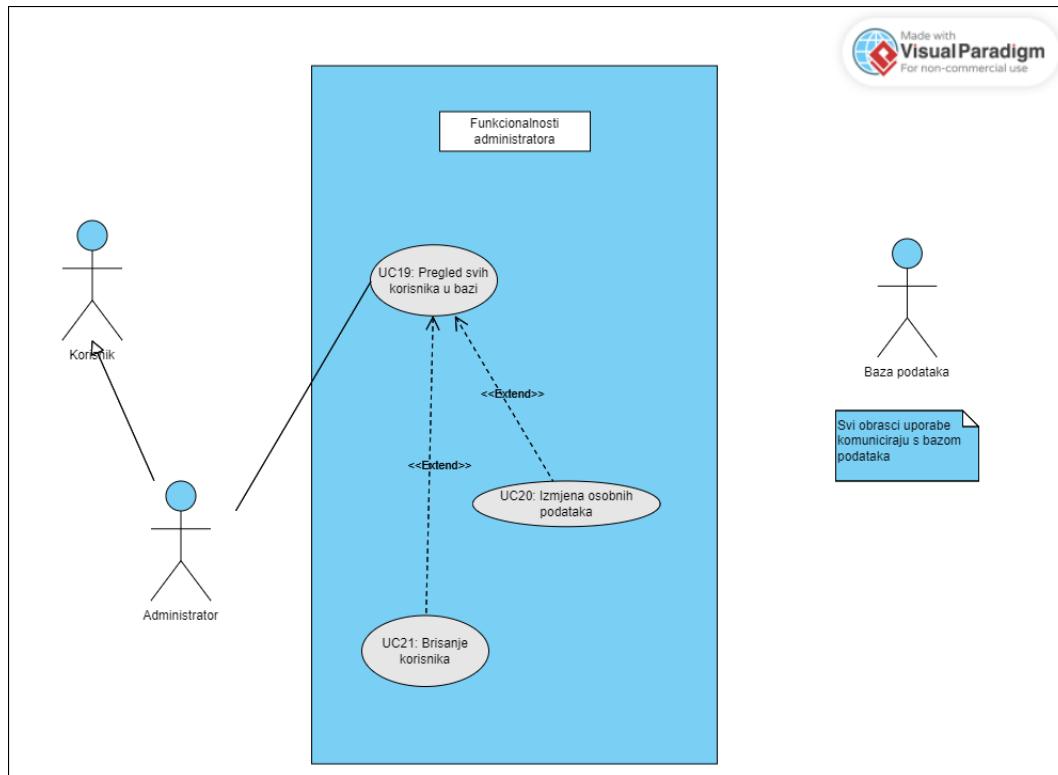
Dijagrami obrazaca uporabe



Slika 3.1: Obrasci uporabe - funkcionalnosti za neprijavljene korisnike



Slika 3.2: Obrasci uporabe - funkcionalnosti za natjecatelje i voditelje

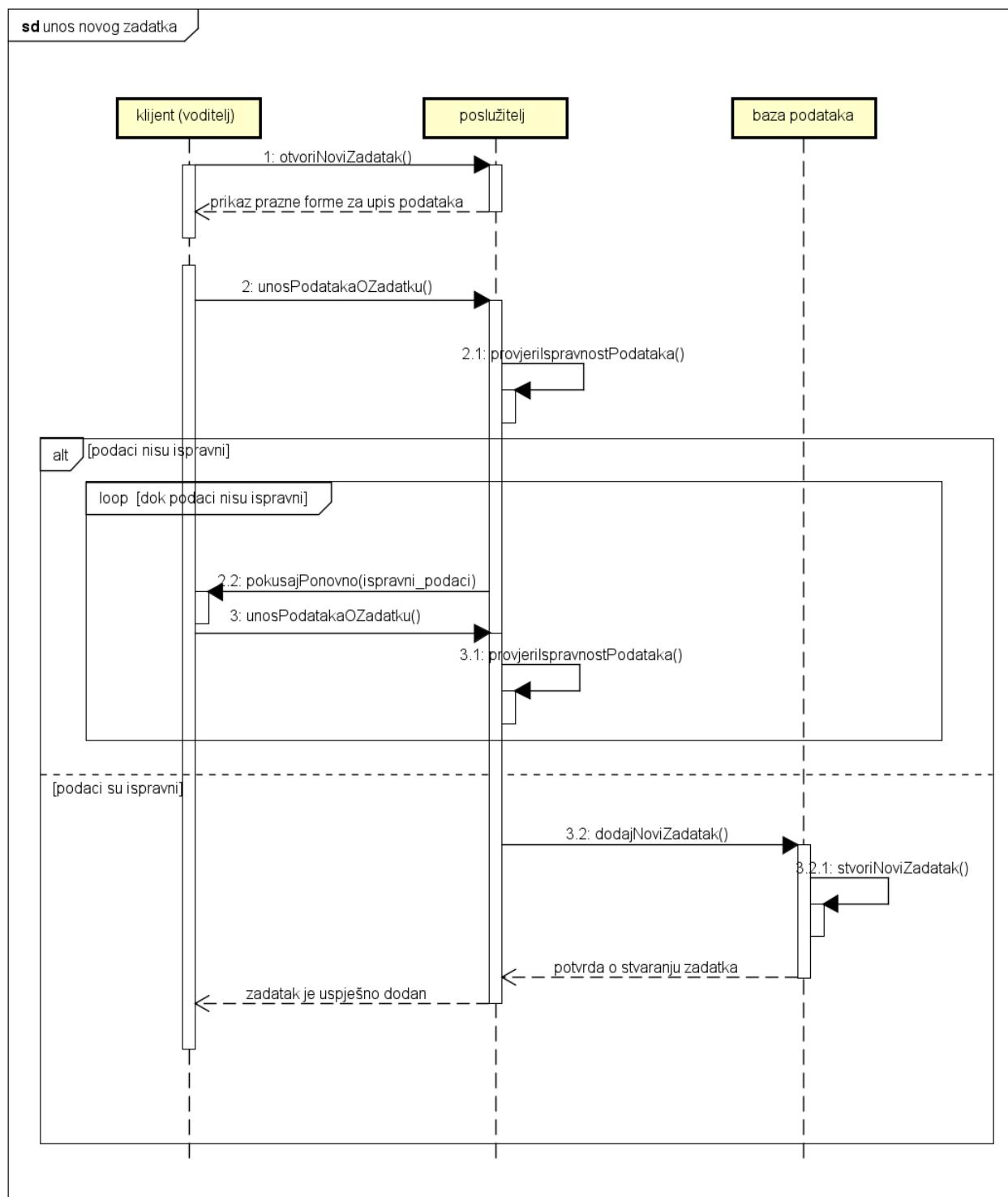


Slika 3.3: Obrasci uporabe - funkcionalnosti za administratore

3.1.2 Sekvencijski dijagrami

UC15 – unos novog zadatka

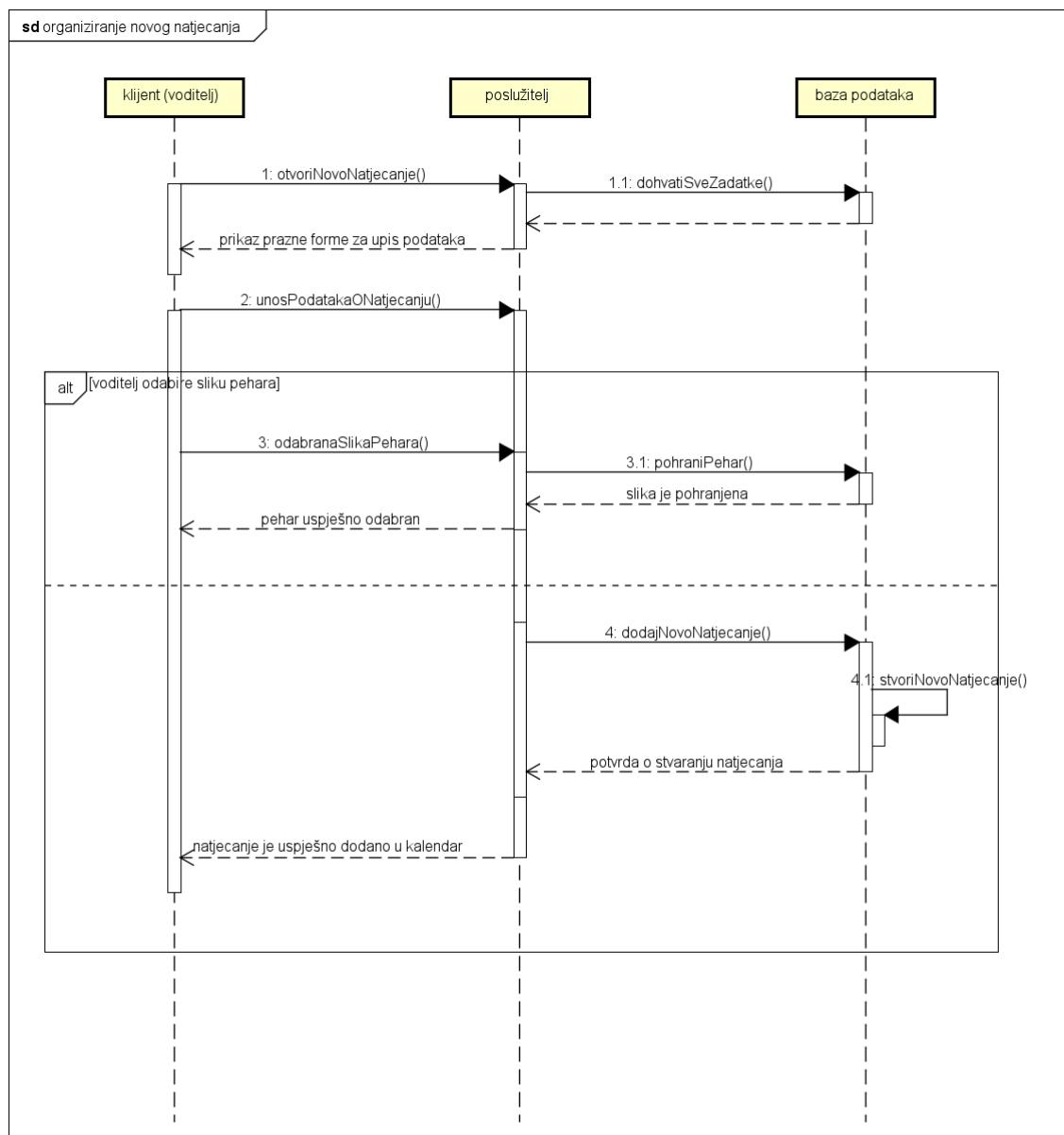
Korisnik prijavljen u sustav kao voditelj odabire opciju za stvaranje novog zadatka. Poslužitelj prikazuje formu s praznim poljima koje voditelj treba ispuniti podacima o zadatku. Točnije, potrebno je navesti naziv zadatka, broj bodova koje nosi, vremensko ograničenje izvršavanja, tekst zadatka i primjere za evaluaciju, a moguće je i odabrati opciju da zadatak bude stvoren kao privatni. Voditelj upisuje navedene podatke te ih šalje poslužitelju koji prvo provjerava da su svi podaci ispravno uneseni te da nema polja koja su ostala prazna. U slučaju neispravnosti podataka, poslužitelj prikazuje relevantnu poruku o problemu i ponovno omogućuje ispunjavanje forme. Nakon uspješne provjere, podaci se šalju bazi podataka koja ih sprema i time stvara novi zadatak. Ako sve prođe bez problema, baza podataka šalje potvrdu poslužitelju o uspješnom stvaranju novog zadatka, a poslužitelj zatim javlja korisniku da je njegov zahtjev uspješno proveden.



Slika 3.4: Sekvencijski dijagram za obrazac uporabe - unos novog zadatka

UC17 – organiziranje novog natjecanja

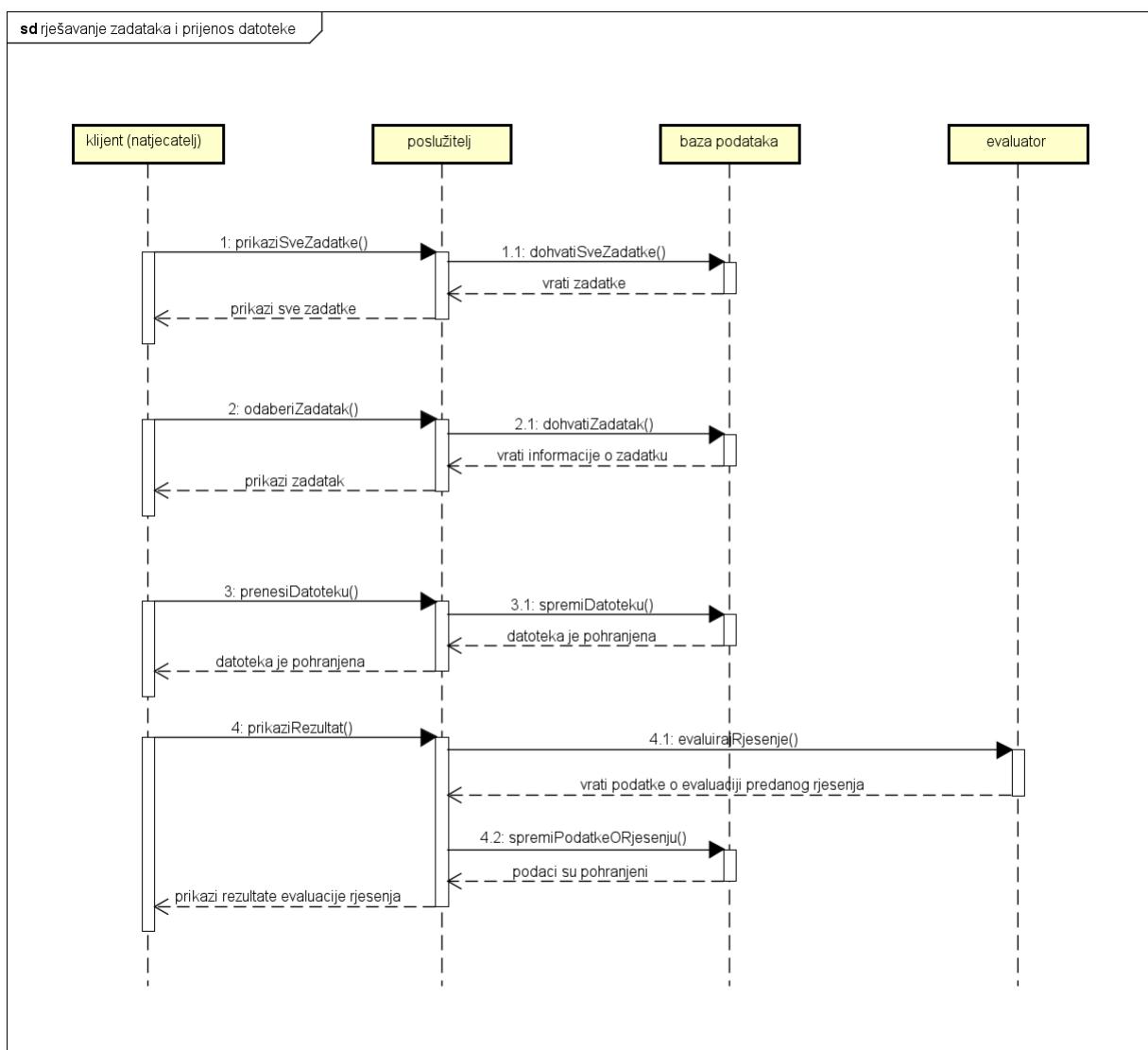
Korisnik prijavljen u sustav kao voditelj odabire opciju za stvaranjem novog natjecanja. Poslužitelj prikazuje formu s praznim poljima koje je voditelj dužan ispuniti informacijama o natjecanju. Potrebno je unijeti vrijeme početka i završetka natjecanja, broj zadataka, odabrati zadatke koji će biti aktivni te po želji učitati sličicu pehara, koja se pohranjuje u bazu podataka, kojom se nagrađuju najbolji natjecatelji. U slučaju da je voditelj izostavio nešto i ostavio polje praznim, sustav će ga obavijestiti i zatražiti ponovni unos podataka. Nakon uspješno ispunjene forme i uspješno primljenih podataka, poslužitelj će ih proslijediti bazi podataka i zatražiti stvaranje novog natjecanja. Nakon pohrane, ako sve prođe bez problema, baza podataka bi trebala poslati povratnu informaciju poslužitelju o uspješnom stvaranju novog natjecanja, a poslužitelj bi zatim trebao javiti korisniku da je njegov zahtjev uspješno proveden.



Slika 3.5: Sekvencijski dijagram za obrazac uporabe - organiziranje novog natjecanja

UC10 i UC13 – rješavanje zadatka i prijenos datoteke

Korisnik prijavljen u sustav kao natjecatelj odabire opciju za prikaz svih objavljenih zadataka. Poslužitelj prikazuje popis svih javnih zadataka. Natjecatelj zatim bira zadatak koji želi pokušati riješiti pri čemu se otvara sučelje s tekstrom zadatka i opcijom za prijenos datoteke rješenja. Natjecatelj odabire datoteku koju želi prenijeti kao rješenje i potvrđuje svoj odabir, a poslužitelj proslijeđuje datoteku do baze podataka koja ju sprema za prikaz prilikom nekih drugih funkcionalnosti. Na klijentov zahtjev predana datoteka se predaje evaluatoru koji pomoću definiranih primjera ulaza i izlaza određuje točnost rješenja zadatka i vraća ih poslužitelju. Poslužitelj sve rezultate sprema u bazu podataka i prikazuje klijentu u aplikaciji.

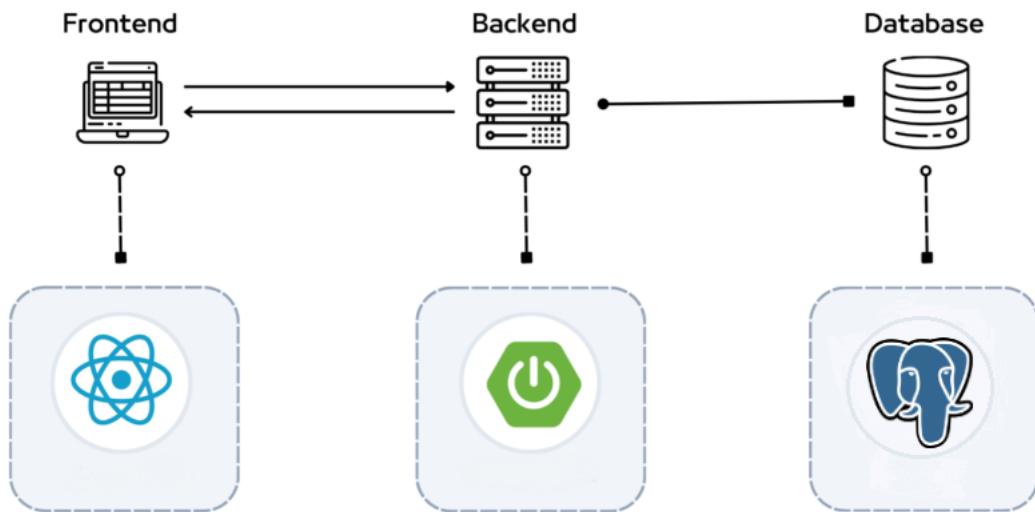


Slika 3.6: Sekvencijski dijagram za obrasc uporabe - rješavanje zadatka i prijenos datoteke

3.2 Ostali zahtjevi

- Sustav treba dopustiti višekorisnički rad u stvarnom vremenu.
- Sustav treba biti jednostavan i intuitivan za korištenje tako da djeca nemaju problema sa razumijevanjem i snalaženjem na aplikaciji.
- Sustav se treba izgraditi kao mrežna aplikacija pomoću objektno-orientiranih jezika.
- Sustav treba biti prilagođen za hrvatski jezik i abecedu prilikom prikaza i unosa tekstualnog sadržaja, uključujući i dijakritičke znakove.
- Pristup sustavu treba biti omogućen putem javne mreže.
- Sustav treba omogućiti evaluaciju rješenja zadataka za minimalno jedan programski jezik.
- Sustav treba biti takav da se sve funkcije izvršavaju brzo, ne duže od nekoliko sekundi, uključujući i evaluaciju rješenja zadataka.
- Baza podataka sustava mora biti kvalitetno i ispravno povezana sa sučeljem aplikacije.

4. Arhitektura i dizajn sustava



Slika 4.1: Prikaz arhitekture sustava

Arhitektura sustava može se podijeliti na tri glavna podsustava, a to su **Frontend Web aplikacija**, **Backend Web aplikacija** i **baza podataka**.

- **Web poslužitelj** prima zahtjeve od klijenata putem interneta, obrađuje ih i pruža resurse poput web stranice, slike, videa i datoteke kao odgovor. Za distribuciju resursa najčešće se koriste protokoli kao što su HTTP (Hypertext Transfer Protocol) ili HTTPS (HTTP Secure).
- **Web aplikacija** je program koji se izvršava na web pregledniku (Google Chrome, Mozilla Firefox, Safari itd.) i pruža korisnicima mogućnost izvršavanja željenih zahtjeva, odnosno interakciju s određenim uslugama i funkcionalnostima web aplikacije. Prilikom obrade zahtjeva pristupa se bazi podataka i korisniku se odgovor vraća kao HTML dokument.
- **Baza podataka** je organizirani skup podataka namijenjen za efikasno upravljanje, ažuriranje, pretraživanje i dohvatanje podataka. Uloge baze podataka u web aplikaciji su pohrana podataka, očuvanje integriteta podataka i osiguranje dosljednosti, upravljanje transakcijama itd.

Za izradu naše web aplikacije odabrali smo **Spring Boot** (open-source Java framework) i **React** (open-source JavaScript library). Odabrana razvojna okruženja su IntelliJ IDEA i Eclipse IDE za Spring Boot, odnosno Visual Studio Code i WebStorm za React. Za izradu baze podatka koristimo PostgreSQL. Arhitektura, koja je podržana Spring Boot-om, temelji se na **MVC (Model-View-Controller)** konceptu koji strogo odvaja model, akcije i prezentaciju, olakšava razvoj i održavanje aplikacije te čini aplikaciju prilagodljivom i jednostavnom za proširenje.

- **Model** - predstavlja poslovnu logiku, odnosno dinamičke strukture podataka, mijenja pogled na zahtjev kontrolera. Modeli u pravilu predstavljaju podatke(objekte) koje aplikacija obrađuje.
- **View** - ono što klijent vidi, odnosno korisničko sučelje potrebno za interakciju s aplikacijom kao što su dijagrami, linkovi, slike, tablice itd.
- **Controller** - presreće zahtjeve klijenata i prilagođava model, odnosno obavještava model o promjeni zahtjeva korisnika i u skladu sa zahtjevima daje prikladan View.

4.1 Baza podataka

Za potrebe našeg sustava koristit ćemo relacijsku bazu podataka koja svojom strukturom olakšava modeliranje stvarnog svijeta. Gradivna jedinka baze je relacija, odnosno tablica koja je definirana svojim imenom i skupom atributa. Zadaća baze podataka je brza i jednostavna pohrana, izmjena i dohvata podataka za daljnju obradu. Baza podataka ove aplikacije sastoји se od sljedećih entiteta:

- Natjecanje
- VirtualnoNatjecanje
- Korisnik
- Pehar
- Zadatak
- Rješenje
- TestniPrimjer

4.1.1 Opis tablica

Natjecanje Ovaj entitet sadržava informacije o natjecanju koje trenutno rješava korisnik. Atributi koje sadržava su: NatjecanjeID, KorisnikID, NazivNatjecanja, PocetakNatjecanja, KrajNatjecanja. Ovaj entitet ima *One-to-Many* vezu s entitetom Pehar preko atributa NatjecanjeID, te *One-to-Many* vezu sa slabim entitetom VirtualnoNatjecanje preko atributa OriginalnoNajtecanjeID. Ima *One-to-Many* vezu s entitetom Korisnik preko atributa KorisnikID.

Natjecanje		
NatjecanjeID	INT	Jedinstveni identifikator natjecanja
KorisnikID	INT	Jedinstveni identifikator korisnika
NazivNatjecanja	VARCHAR	Naziv natjecanja
PocetakNatjecanja	TIMESTAMP	Vrijeme početka natjecanja
KrajNatjecanja	TIMESTAMP	Vrijeme završetka natjecanja

VirtualnoNatjecanje Ovaj entitet sadržava informacije o virtualnom natjecanju koje je pokrenuo korisnik na temelju nekog natjecanja. Atributi koje sadržava su: virtualnoNatjecanjeID, KorisnikID, OriginalnoNatjecanjeID i PocetakNatjecanja. Ovaj entitet ima *Many-To-One* vezu s entitetom Korisnik preko atributa i vanjskog ključa KorisnikID. Atribut originalnoNatjecanjeID predstavlja vanjski ključ koji se referencira na natjecanjeID u relaciji natjecanje pa se time tvori *Many-To-One* veza.

VirtualnoNatjecanje		
VirtualnoNatjecanjeID	INT	Jedinstveni identifikator virtualnog natjecanja
KorisnikID	INT	ID korisnika koji je stvorio virtualno natjecanje
originalnoNatjecanjeID	INT	ID natjecanja na kojem se temelji virtualno natjecanje
PocetakNatjecanja	TIMESTAMP	Vrijeme početka natjecanja

Korisnik Ovaj entitet sadržava sve bitne informacije o korisniku aplikacije.

Sadrži atribute: KorisnickoIme, lozinka, ime, prezime, email, fotografija, vrijemeRegistracije, ulogaID, requestedUloga i confirmedEmail. Ovaj entitet je u vezi *One-to-Many* s entitetom Natjecanje preko atributa KorisnikID, u vezi *One-to-Many* s entitetom VirtualnoNatjecanje preko atributa KorisnikID, u vezi *One-to-Many* s entitetom Zadatak preko atributa VoditeljID, u vezi *One-to-Many* s entitetom Rješenje preko atributa NatjecateljID, te je u *One-to-Many* s entitetom Pehar preko atributa NatjecateljID.

Korisnik		
KorisnikID	INT	Jedinstveni identifikator korisnika
KorisničkoIme	VARCHAR	Jedinstveno ime korisnika
Lozinka	VARCHAR	Korisnikova lozinka
Ime	VARCHAR	Ime korisnika
Prezime	VARCHAR	Prezime korisnika
Email	VARCHAR	elektronička pošta korisnika
Fotografija	PATH	fotografija korisnika
VrijemeRegistracije	TIMESTAMP	Vrijeme kada se korisnik registrirao u sustav
UlogaID	VARCHAR	Jedinstveni identifikator uloge
requestedUloga	VARCHAR	Uloga koju je korisnik zatražio
confirmedEmail	BOOLEAN	Zastavica koja određuje je li korisnik potvrdio Email za registraciju

Pehar Ovaj entitet predstavlja pehar kojeg natjecatelji (korisnici) mogu osvojiti u natjecanju. Sadrži atribute: PeharID, NatjecateljID, NatjecanjeID, Mjesto te SlikaPehara. Ovaj entitet ima *Many-to-One* vezu s entitetom Natjecanje preko atributa NatjecanjeID, te ima vezu *Many-to-One* s entitetom Korisnik preko atributa NatjecateljID.

Pehar		
PeharID	INT	Jedinstveni identifikator pehara

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Pehar		
NatjecateljID	INT	Jedinstveni identifikator natjecatelja
NatjecanjeID	INT	Jedinstveni identifikator natjecanja
Mjesto	INT	Mjesto koje je dobiveno peharom (1, 2 ili 3)
SlikaPehara	VARCHAR	Slika dobivenog pehara

Zadatak Ovaj entitet sadržava sve bitne značajke za definiciju jednog zadatka u aplikaciji. Atributi koje sadržava su: zadatakID, voditeljID, nazivZadatka, brojBodova, vremenskoOgranicenje, tekstZadatka, tezinaZadatka te privatniZadatak. Ovaj entitet ima *One-To-Many* vezu sa slabim entitetom testniPrimjer preko atributa zadatakID. Vanjskim ključem natjecanjeID stvorena je opcionalna *Many-To-One* veza sa entitetom natjecanje. Postoji i *One-To-Many* veza s entitetom rješenje preko atributa zadatakID. *Many-To-One* veza postoji i sa entitetom korisnik preko vanjskog ključa voditeljID (označava identifikator korisnika s ulogom voditelja).

Zadatak		
ZadatakID	INT	Jedinstveni privatni identifikator zadatka
nazivZadatka	VARCHAR	Naziv zadatka
tekstZadatka	VARCHAR	tekst kojim je zadan zadatak
tezinaZadatka	VARCHAR	Predstavlja tezinu zadatka
brojBodova	INT	broj bodova koliko nosi zadatak
vremenskoOgranicenje	INT	vremensko ograničenje za izvođenje predanog rješenja
privatniZadatak	BOOLEAN	zastavica koja određuje ako je zadatak privatn
VoditeljID	INT	Jedinstveni identifikator voditelja koji je stvorio zadatak

Rješenje Ovaj slabi entitet sadržava informacije o predenim rješenjima pojedinog korisnika za određeni zadatak. Atributi koje sadržava su: rješenjeRb, natje-

cateljID, zadatakID, vrijemeOdgovora, brojTočnihPrimjera, brojBodova i programskiKod. Ovaj entitet ima *Many-To-One* vezu s entitetom zadatak preko atributa i vanjskog ključa zadatakID. *Many-To-One* veza postoji i sa entitetom korisnik preko vanjskog ključa korisnikID.

Rješenje		
rjesenjeRb	INT	Redni broj predanog rješenja određenog korisnika za predani zadatak
ZadatakID	INT	Jedinstveni privatni identifikator zadatka
NatjecateljID	INT	Jedinstveni identifikator natjecatelja koji je predao rješenje
vrijemeOdgovora	TIMESTAMP	Vrijeme predaje rješenja
brojTočnihOdgovora	DOUBLE	Broj primjera koji prolaze evaluaciju
brojBodova	INT	Ostvareni broj bodova na zadatu
programsckiKod	TEXT	Programski kod predanog rješenja

TesniPrimjer Ovaj slabi entitet sadržava informacije o testnim primjerima za određeni zadatak. Atributi koje sadržava su: testniPrimjerRb, zadatakID, ulazniPodaci i izlazniPodaci. Ovaj entitet ima *Many-To-One* vezu s entitetom zadatak preko atributa i vanjskog ključa zadatakID.

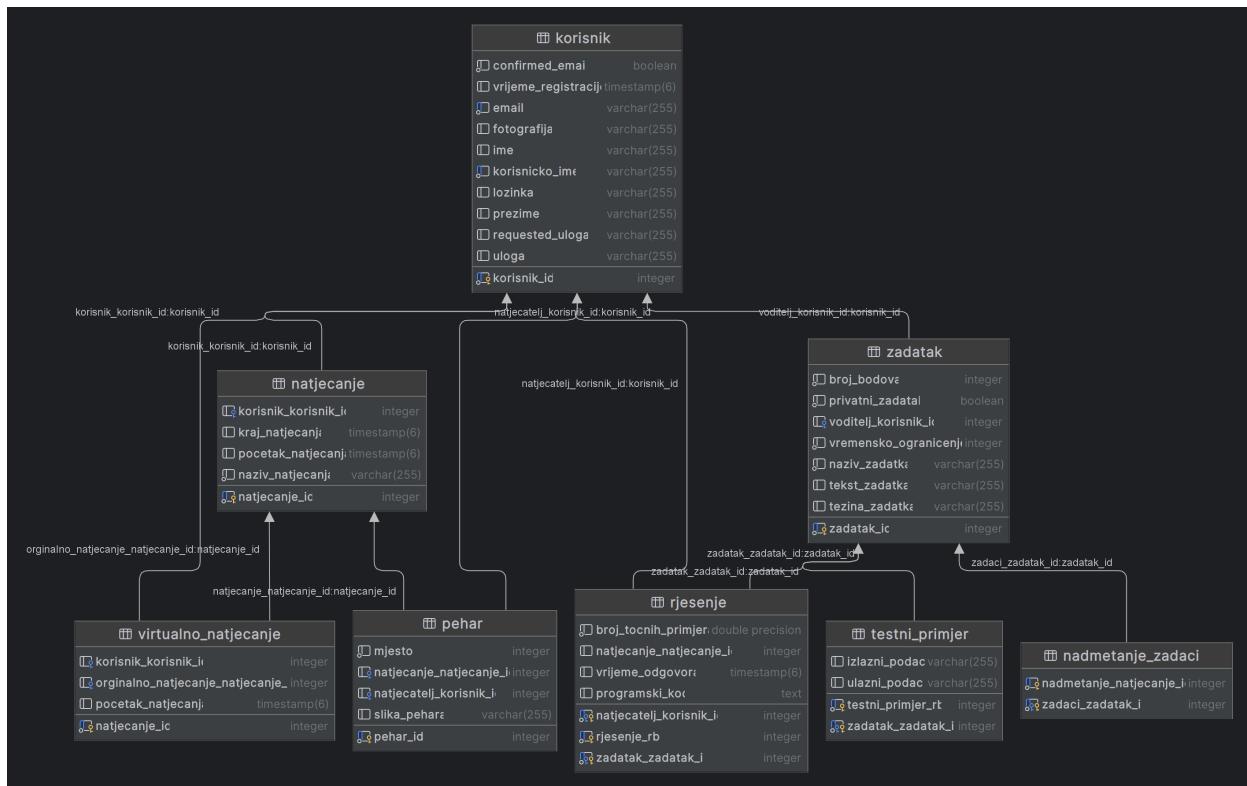
TestniPrimjer		
TestniPrimjerRB	INT	Redni broj testnog primjera za pojedini zadatak
zadatakID	INT	Jedinstveni privatni identifikator zadatka
ulazniPodaci	VARCHAR	ulazni podaci za testiranje programskog rješenja
izlazniPodaci	VARCHAR	očekivani ispis programskog rješenja

NadmetanjeZadaci Ovaj slabi entitet sadržava informacije o zadacima veza-

nim uz natjecanje. Atributi koje sadržava su: NadmetanjeID i ZadatakID. Ovaj entitet ima *Many-To-One* vezu s entitetom zadatak preko atributa i vanjskog ključa ZadatakID.

TestniPrimjer		
NadmetanjeID	INT	Jedinstveni privatni identifikator zadatka nadmetanja
ZadatakID	INT	Jedinstveni privatni identifikator zadatka

4.1.2 Dijagram baze podataka

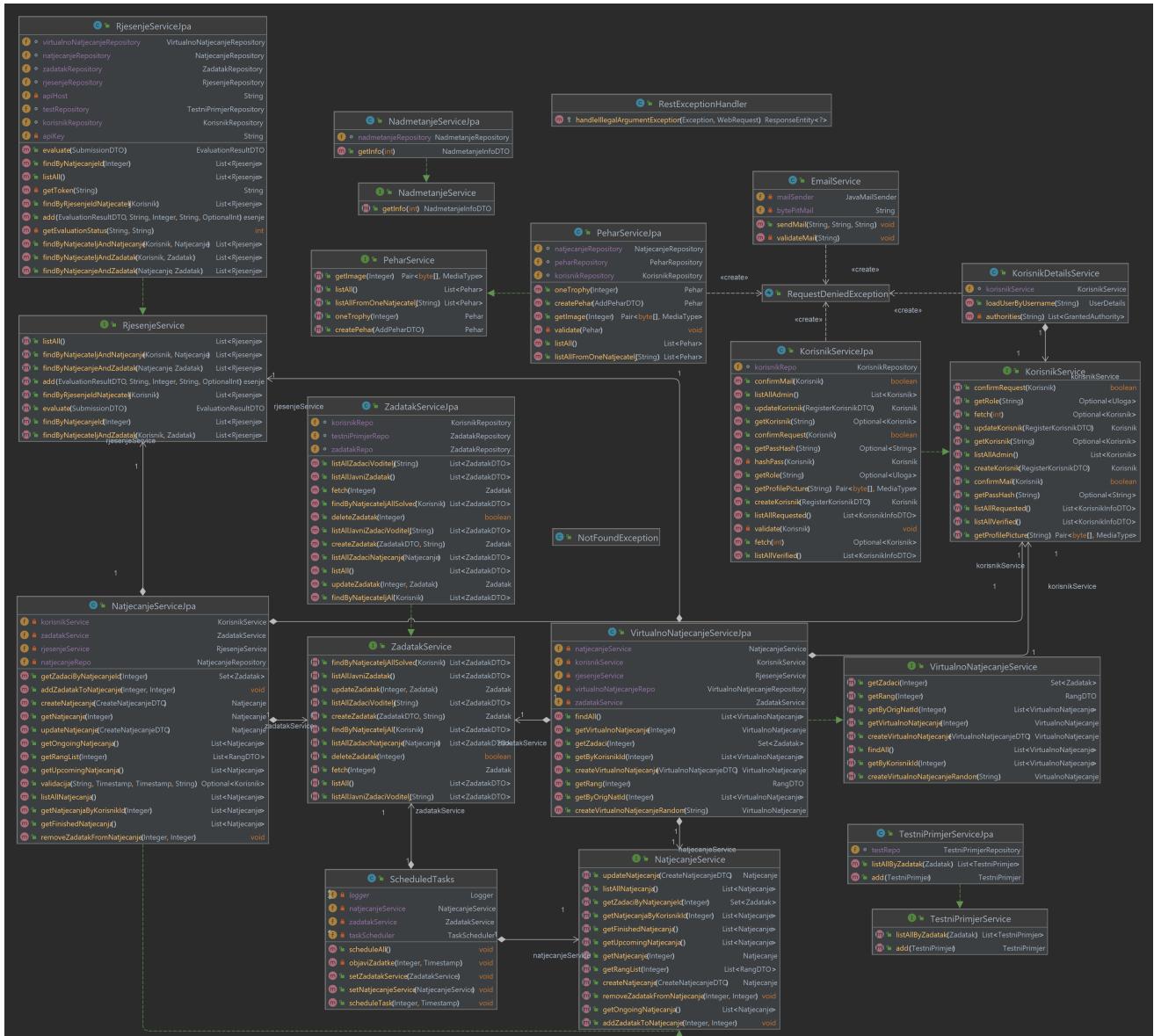


Slika 4.2: Dijagram baze podataka

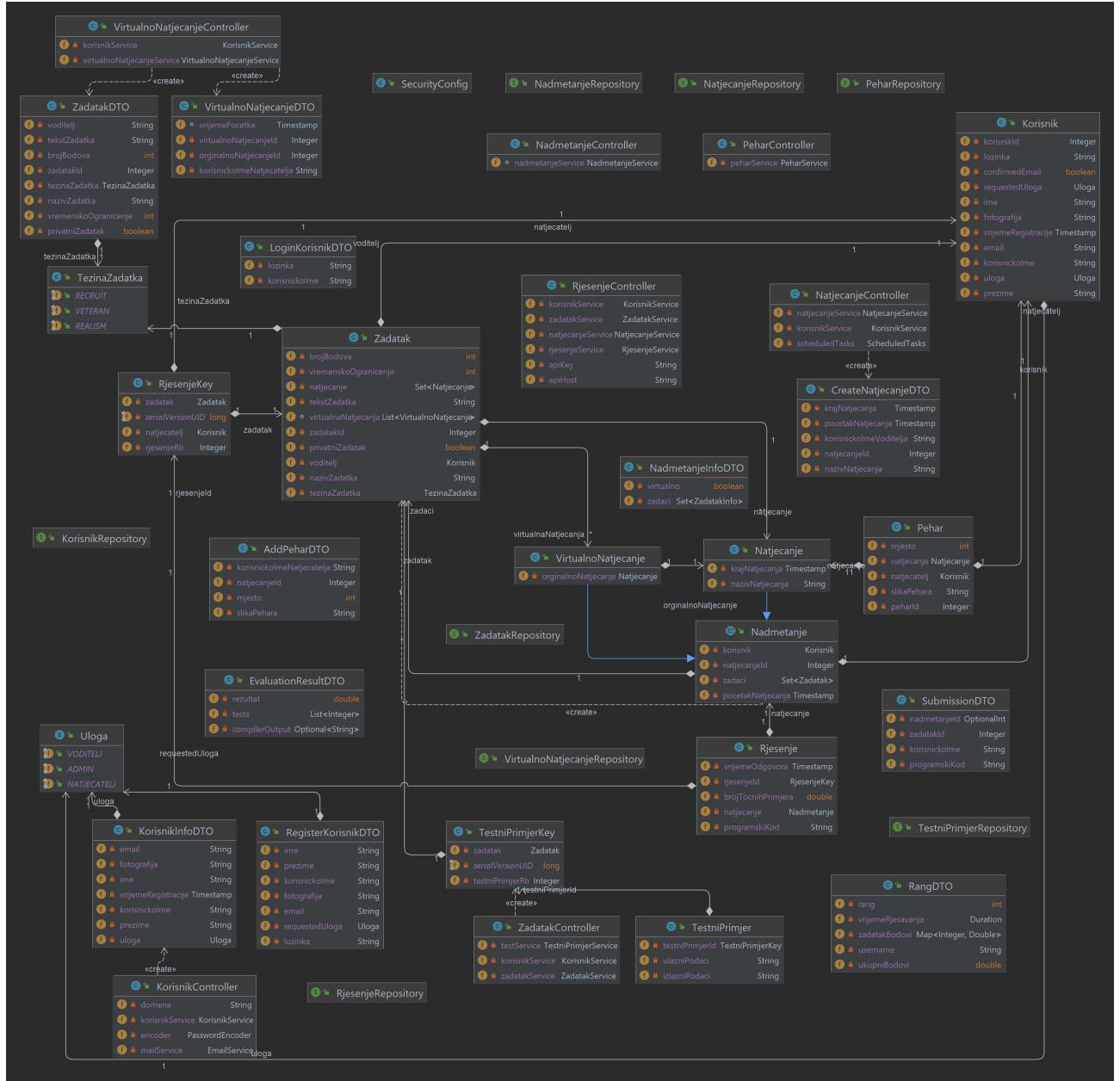
4.2 Dijagram razreda

Na slikama 4.3 i 4.4 prikazani su dijagrami razreda koji predstavljaju backend dio arhitekture. Prva slika opisuje servise, odnosno njihovu implementaciju i prikazani su atributi i metode u pojedinim razredima. Servisi predstavljaju glavnu logiku i oni su uglavnom u interakciji s repozitorijima koji su između ostalog prikazani na slici 4.4. Osim repozitorija, na slici 4.4 prikazani su i modeli (predstavljaju strukturu baze podataka) te kontroleri. Kontrolери su zaduženi za upravljanje HTTP zahtjevima i pružanje prikladnog odgovora. Na drugoj slici možemo uočiti attribute i međusobne odnose prethodno navedenih razreda.

Razred Korisnik predstavlja registriranog korisnika koji se može registrirati kao voditelj ili natjecatelj unoseći svoje podatke u sustav. Nadmetanje je bazni razred kojeg nasljeđuju razredi Natjecanje i VirtualnoNatjecanje. Razred Natjecanje definira naziv, početak i kraj natjecanja te voditelja, a VirtualnoNatjecanje predstavlja natjecanje koje se može naknadno pokrenuti i rješavati. Razred Pehar je entitet u kojem spremamo koje mjesto je osvojio natjecatelj na nekom natjecanju i sliku pehara. Razred uloga je enumeracija koja definira ulogu korisnika (admin, natjecatelj ili voditelj). Razred zadatak je entitet koji predstavlja određeni problem na natjecanju i sadrži referencu na testne primjere. Testni primjer sadrži informacije o tesnim primjerima za pojedinačni zadatak. Razred Rjesenje predstavlja predano rješenje pojedinog korisnika za zadatak.



Slika 4.3: Dijagram razreda - Servisi

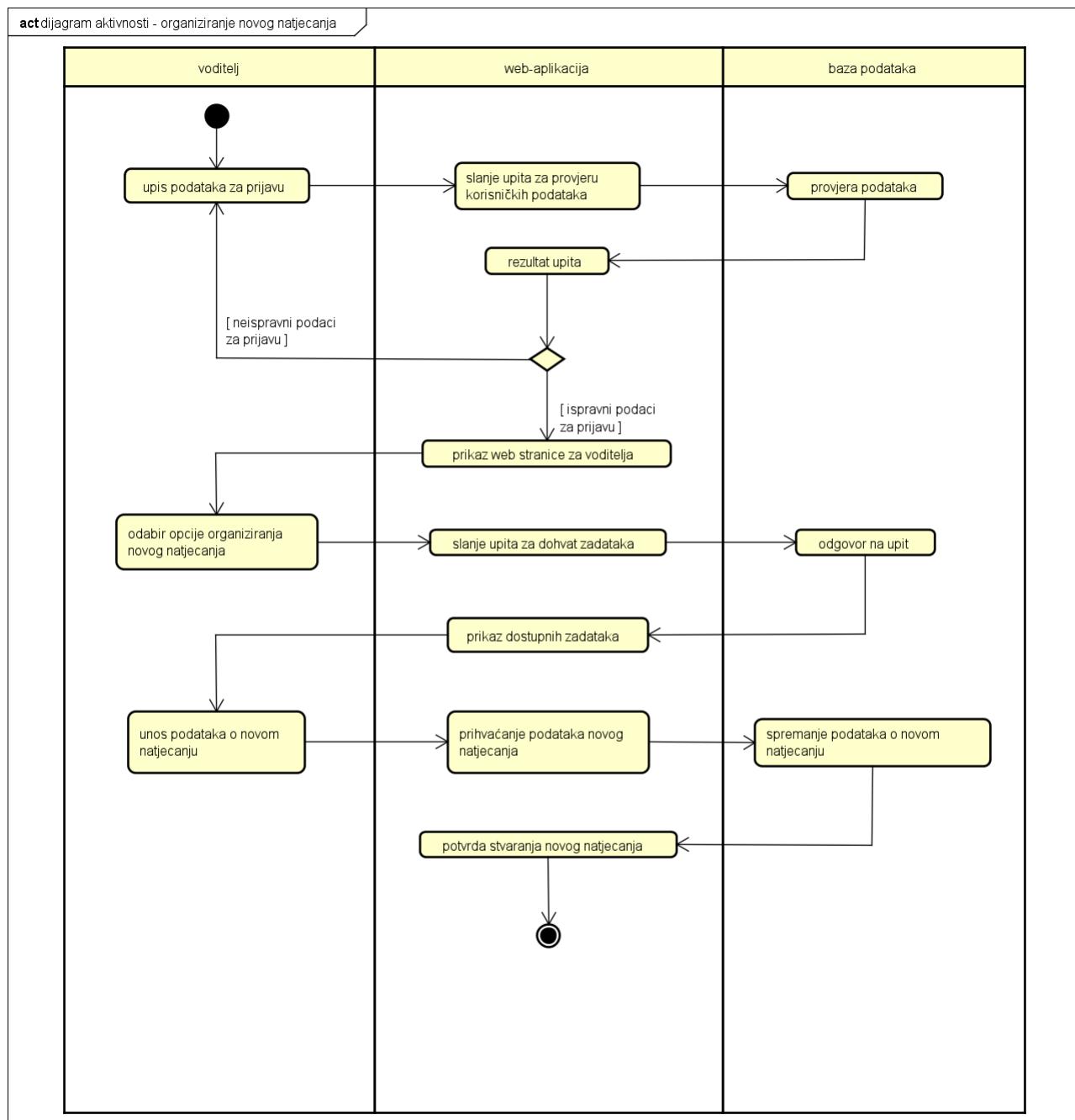


Slika 4.4: Dijagram razreda - Kontroleri, Repozitoriji i Modeli

4.3 Dijagram aktivnosti

Dijagram aktivnosti na slici 4.5 prikazuje proces kreiranja novog natjecanja. Voditelj se prijavljuje u sustav te nakon uspješne prijave odabire opciju za organiziranje novog natjecanja. Web stranica preko baze podataka dohvata dostupne zadatke te ih prikazuje kao dio forme u koju voditelj unosi podatke te odabire zadatke koji će se ispitivati u sklopu natjecanja. Nakon slanja unesenih podataka web aplikacija

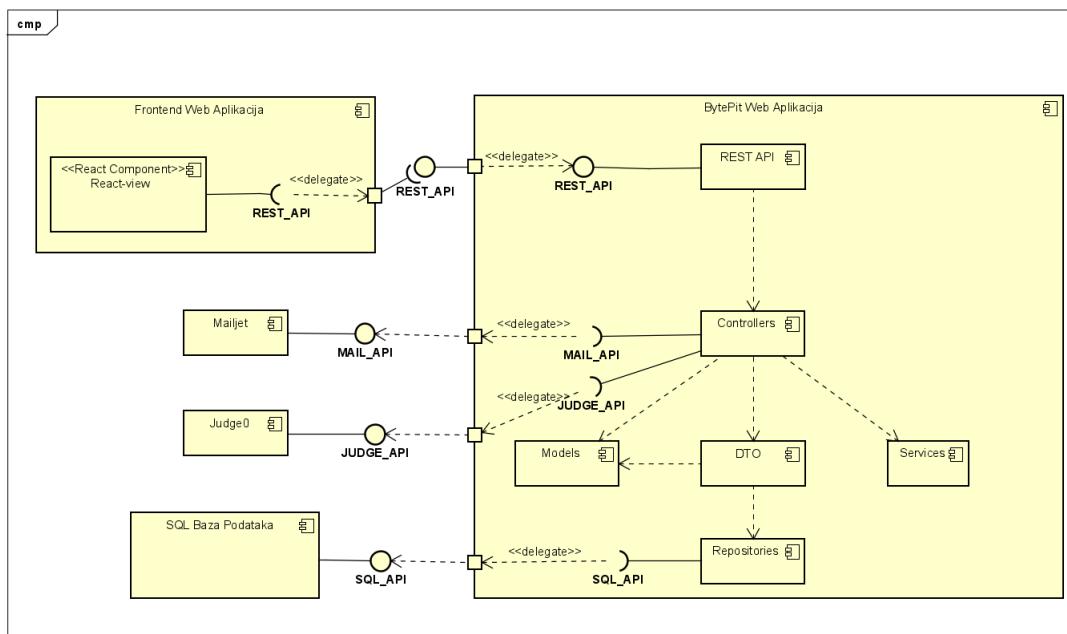
ciji, ona ih prosljeđuje bazi podataka koja ih zatim pohranjuje i šalje potvrdu o uspješnom stvaranju novog natjecanja.



Slika 4.5: Dijagram aktivnosti - organiziranje novog natjecanja

4.4 Dijagram komponenti

Dijagram komponenti prikazan na slici 4.6 opisuje organizaciju i međuvisnost komponenti, interne strukture i odnose prema okolini. Web aplikacija sastoji se od komponenti: Controllers, Services, Repositories, DTO i Models. Controllers pruža REST API sučelje na koje vanjski web preglednik može slati zahtjeve i primati odgovore pomoću JSON datoteka. Repositories pristupa SQL bazi podataka koja ostvaruje sučelje SQL API. Podaci koji su pristigli iz baze se šalju dalje MVC arhitekturi u obliku DTO(Data transfer objcet). Repositories upisuje podatke iz baze podataka u komponentu Services. Models oblikuje korištene entitete i njihov međuodnos. Controllers šalje i prima podatke od komponente Services. Controllers pruža MAIL API sučelje Mailjet preko kojeg šalje mailove i Judge0 API za evaluaciju programskog rješenja.



Slika 4.6: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Za komunikaciju tima korištene su aplikacije WhatsApp¹ i Discord². Za izradu UML dijagrama korišten je Astah Professional³.

Za izradu dokumentacije korišten je LaTeX⁴, sustav za izradu dokumenata koji koristi markup jezik. Za upravljanje izvornim kodom korišten je Git⁵. Repozitorij projekta dostupan je na platformi GitHub⁶.

Za razvojna okruženja korišteni su Visual Studio Code⁷. Visual Studio Code je integrirano razvojno okruženje (IDE) kompanije Microsoft. Koristi se za razvoj wen-stranica, web-aplikacija, web-usluga i mobilnih aplikacija. Za razvoj softvera koristi Windows API, Windows Forms, Windows Presentation Foundation, Windows Store i Microsoft Silverlight.

Za pisanje aplikacije korišten je Spring Boot⁸, framework za razvoj Java aplikacija, za razvoj backenda. Za razvoj frontenda korišten je React⁹, biblioteka za izgradnju sučelja u jeziku JavaScript¹⁰. Također je korišten TypeScript¹¹ kao nadogradnja nad JavaScriptom. On poboljšava kvalitetu i održivost koda napisanog u JavaScriptu.

Baza podataka nalazi se na poslužitelju

¹<https://www.whatsapp.com/>

²<https://discord.com>

³<http://astah.net/editions/professional>

⁴<https://www.latex-project.org>

⁵<https://git-scm.com/>

⁶<https://github.com>

⁷<https://visualstudio.microsoft.com/>

⁸<https://spring.io/projects/spring-boot/>

⁹<https://react.dev>

¹⁰<https://www.javascript.com/>

¹¹<https://www.typescriptlang.org>

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

U procesu testiranja funkcionalnosti komponenti, koristili smo popularni radni okvir *JUnit*. Kako bi testiranje bilo izolirano, potrebno je simulirati komponente o kojima ovisi komponenta koju testiramo. U tu svrhu smo koristili programski okvir *Mockito*. Ukupno smo implementirali sedam testirajućih metoda unutar tri testirajuće klase.

U svrhu testiranja funkcionalnosti klase *KorisnikController*, implementirali smo tri metode. Specifično, testirajuća metoda 5.1 ima za cilj provjeriti ispravnost metode za registraciju novog korisnika kada su joj proslijedeni valjni podaci za registraciju (ime, prezime, korisničko ime, lozinka, e-mail i uloga) te slika profila. Očekujemo da će rezultat testirane metode biti objekt tipa *Korisnik*, koji će sadržavati točne podatke (prethodno proslijedene) o novostvorenom korisniku.

```
@Test
public void test_register_new_user_with_valid_data_and_image() {
    // Arrange
    MultipartFile file = new MockMultipartFile(
        "image", "test.jpg", "image/jpeg", "test".getBytes());
    RegisterKorisnikDTO dto = new RegisterKorisnikDTO();
    dto.setKorisnickoIme("DeanKotiga");
    dto.setIme("Dean");
    dto.setPrezime("Kotiga");
    dto.setLozinka("MladenVukorepaJeKul");
    dto.setEmail("dean@kotiga.com");
    dto.setRequestedUloga(Uloga.NATJECATELJ);
    when(korisnikRepository.save(any(Korisnik.class))).thenAnswer(i -> (Korisnik) i.getArguments()[0]);

    // Act
    Korisnik result = korisnikController.addKorisnik(file, dto);

    // Assert
    assertNotNull(result);
    assertEquals(dto.getKorisnickoIme(), result.getKorisnickoIme());
    assertEquals(dto.getIme(), result.getIme());
    assertEquals(dto.getPrezime(), result.getPrezime());
    assertEquals(dto.getEmail(), result.getEmail());
    assertEquals(dto.getRequestedUloga(), result.getRequestedUloga());
    assertNotNull(result.getLozinka());
    assertNotNull(result.getFotografija());
}
```

Slika 5.1: Testirajuća metoda - registracija korisnika

Za razliku od prethodne, testirajuća metoda 5.2 provjerava ispravnost iste metode u situaciji kada se pokuša dodati korisnik s korisničkim imenom koje je već zauzeto. U ovome ispitnom slučaju se očekuje da će testirana metoda prepoznati

ovu situaciju i shodno tome baciti očekivanu iznimku.

```
@Test
public void test_throw_exception_if_username_already_exists() {
    // Arrange
    String existingUsername = "MoranaZibar";
    when(korisnikRepository.findByKorisnickoIme(existingUsername)).thenReturn(Optional.of(mock(Korisnik.class)));
    MultipartFile file = new MockMultipartFile(
        name: "image", originalFilename: "test.jpg", contentType: "image/jpeg", "test".getBytes());
    RegisterKorisnikDTO dto = new RegisterKorisnikDTO();
    dto.setKorisnickoIme(existingUsername);
    dto.setIme("Morana");
    dto.setPrezime("Zibar");
    dto.setLozinka("kresojezivcenjak");
    dto.setEmail("morana@zibar.hr");
    dto.setRequestedUloga(Uloga.NATJECATELJ);

    // Act & Assert
    assertThrows(RequestDeniedException.class, () -> {
        korisnikController.addKorisnik(file, dto);
    });
}
```

Slika 5.2: Testirajuća metoda - neuspješna registracija korisnika

Posljednjim ispitnim slučajem 5.3 za klasu *KorisnikController* želimo verificirati ispravnost metode za potvrdu zahtijevanih uloga korisnika od strane administratora. Očekuje se da će nakon uspješne potvrde testirana metoda, koja kao argument prima korisničko ime, vratiti HTTP odgovor sa statusom OK.

```
@Test
public void test_confirmRequest_success() {
    // Arrange
    Authentication auth = new UsernamePasswordAuthenticationToken(
        principal: "admin",
        credentials: "adminSifra",
        Collections.singleton(new SimpleGrantedAuthority(role: "ADMIN")));
    SecurityContext sc = SecurityContextHolder.getContext();
    sc.setAuthentication(auth);

    Korisnik korisnik = new Korisnik();
    korisnik.setRequestedUloga(Uloga.VODITELJ);
    korisnik.setKorisnickoIme("MladenVukorepa");
    korisnik.setEmail("ScarlettOHara@gmail.com");
    when(korisnikRepository.findByKorisnickoIme("MladenVukorepa")).thenReturn(Optional.of(korisnik));

    // Act
    ResponseEntity<?> response = korisnikController.confirmRequest(korisnickoIme: "MladenVukorepa");

    // Assert
    assertEquals(HttpStatus.OK, response.getStatusCode());
}
```

Slika 5.3: Testirajuća metoda - potvrda zatražene uloge

S ciljem testiranja klase *NatjecanjeController* napisana su dva testa. Testirajućom metodom 5.4 želimo provjeriti ispravnost metode za stvaranje novog natjecanja

kada su pruženi ispravni ulazni podaci (naziv, voditelj, početak i kraj natjecanja). Očekuje se da će nakon uspješnog stvaranja natjecanja, testirana metoda vratiti odgovarajući objekt *CreateNatjecanjeDTO* koji sadrži proslijedene ulazne podatke o novostvorenom natjecanju.

```
@Test
public void testCreatesNewNatjecanjeWithValidInputData() {
    // Arrange
    CreateNatjecanjeDTO natjecanjeDTO = new CreateNatjecanjeDTO();
    natjecanjeDTO.setNazivNatjecanja("testno natjecanje");
    natjecanjeDTO.setPocetakNatjecanja(Timestamp.valueOf("2022-01-01 00:00:00"));
    natjecanjeDTO.setKrajNatjecanja(Timestamp.valueOf("2022-01-02 00:00:00"));
    natjecanjeDTO.setKorisnickoImeVoditelja("buttler");
    when(natjecanjeRepository.save(any(Natjecanje.class))).thenAnswer(i -> (Natjecanje) i.getArguments()[0]);

    // Act
    CreateNatjecanjeDTO result = natjecanjeController.createNatjecanje(natjecanjeDTO, userDetails);

    // Assert
    assertNotNull(result);
    assertEquals(natjecanjeDTO.getNazivNatjecanja(), result.getNazivNatjecanja());
    assertEquals(natjecanjeDTO.getPocetakNatjecanja(), result.getPocetakNatjecanja());
    assertEquals(natjecanjeDTO.getKrajNatjecanja(), result.getKrajNatjecanja());
    assertEquals(natjecanjeDTO.getKorisnickoImeVoditelja(), result.getKorisnickoImeVoditelja());
}
```

Slika 5.4: Testirajuća metoda - stvaranje natjecanja

Druga i posljednja testirajuća metoda 5.5 za klasu *NatjecanjeController* ima za cilj provjeriti ispravnost iste metode u situaciji kada je odabrani početak natjecanja nakon završetka natjecanja. Očekuje se da će u takvim slučajevima metoda baciti iznimku.

```
@Test
public void testThrowsExceptionIfPocetakNatjecanjaIsAfterKrajNatjecanja() {
    // Arrange
    CreateNatjecanjeDTO natjecanjeDTO = new CreateNatjecanjeDTO();
    natjecanjeDTO.setNazivNatjecanja("testno natjecanje");
    natjecanjeDTO.setKorisnickoImeVoditelja("buttler");
    natjecanjeDTO.setPocetakNatjecanja(Timestamp.valueOf("2025-01-02 00:00:00"));
    natjecanjeDTO.setKrajNatjecanja(Timestamp.valueOf("2024-01-01 00:00:00"));

    //Act & Assert
    assertThrows(IllegalArgumentException.class, () -> {
        natjecanjeController.createNatjecanje(natjecanjeDTO, userDetails);
    });
}
```

Slika 5.5: Testirajuća metoda - neuspješno stvaranje natjecanja

Na kraju, implementirana su i dva testa za klasu *ZadatakController*. U metodi 5.6, fokusirali smo se na testiranje metode koja dohvata sve javne zadatke, s po-

sebnim naglaskom na rubni slučaj kada nema dostupnih javnih zadataka. U toj situaciji očekujemo praznu listu kao povratnu vrijednost.

```
@Test
public void test_returns_empty_list_when_no_public_tasks() {
    // Arrange
    List<Zadatak> problems = new ArrayList<>();
    when(zadatakRepository.findAllJavniZadatak()).thenReturn(problems);
    // Act
    List<ZadatakDTO> result = zadatakController.listAll();
    // Assert
    assertTrue(result.isEmpty());
}
```

Slika 5.6: Testirajuća metoda - dohvati javnih zadataka

Posljednjom testirajućom metodom 5.7 ispitujemo funkcionalnost metode za ažuriranje zadatka u scenariju kada administrator želi izvršiti ažuriranje. Testiranoj metodi *updateKorisnik* proslijedujemo identifikator zadatka, podatke koje želimo ažurirati (naziv i težina) te informacije o prijavljenom korisniku (administratoru). Očekujemo da će kao izlaz metoda vratiti ažurirani objekt tipa *Zadatak* te provjeravamo je su li naziv zadatka i težina zadatka ispravno ažurirani prema zadanim promjenama.

```
@Test
public void test_admin_can_update_any_problem() {
    // Arrange
    Authentication auth = new UsernamePasswordAuthenticationToken(
        principal: "admin", credentials: "adminSifra",
        Collections.singleton(new SimpleGrantedAuthority( role: "ADMIN")));
    SecurityContext sc = SecurityContextHolder.getContext();
    sc.setAuthentication(auth);
    UserDetailsService userDetailsService = new User( username: "admin", password: "adminSifra",
        Collections.singleton(new SimpleGrantedAuthority( role: "ADMIN")));

    Zadatak zadatak = new Zadatak();
    zadatak.setNazivZadatka("Naziv");
    zadatak.setTezinaZadatka(TezinaZadatka.RECRUIT);
    zadatak.setVoditelj(mock(Korisnik.class));
    when(zadatakRepository.save(any(Zadatak.class))).thenAnswer(i -> (Zadatak) i.getArguments()[0]);
    when(zadatakRepository.findById(any(Integer.class))).thenReturn(Optional.of(zadatak));

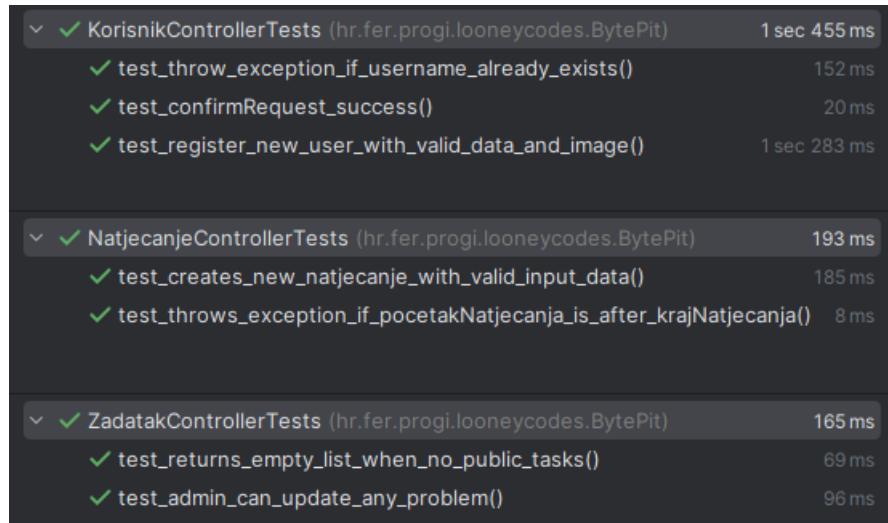
    Zadatak updates = new Zadatak();
    updates.setNazivZadatka("Novi naziv");
    updates.setTezinaZadatka(TezinaZadatka.VETERAN);

    // Act
    Zadatak updatedZadatak = zadatakController.updateKorisnik( id: 1, updates, userDetailsService);

    // Assert
    assertNotNull(updatedZadatak);
    assertEquals(updates.getNazivZadatka(), updatedZadatak.getNazivZadatka());
    assertEquals(updates.getTezinaZadatka(), updatedZadatak.getTezinaZadatka());
}
```

Slika 5.7: Testirajuća metoda - ažuriranje zadatka od strane administratora

Rezultati testiranja se mogu vidjeti na slici 5.8.



Slika 5.8: Rezultati testiranja

5.2.2 Ispitivanje sustava

Ispitivanje sustava proveli smo pomoću *Selenium Web Drivera*, implementirajući ispitne slučajeve unutar *JUnit* testova. Ukupno smo napisali tri testna slučaja.

Ispitnim slučajem 5.9 ispitali smo uspješnost prijave korisnika s točnim korisničkim imenom i lozinkom te potvrdili očekivano preusmjeravanje na početnu stranicu i prisutnost gumba za odjavu.

```

@Test
public void test_login_with_valid_credentials() {
    WebDriver driver = new FirefoxDriver();
    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
    driver.get("http://localhost:8080/");

    WebElement loginButton = wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector(".loginDiv button")));
    loginButton.click();
    WebElement usernameInput = wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("korisnickoIme")));
    usernameInput.sendKeys("spiderman");
    WebElement passwordInput = wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("lozinka")));
    passwordInput.sendKeys("iLoveMJ");
    WebElement submitButton = wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector("button[type='submit']")));
    submitButton.click();
    Boolean logoutPrisutan = wait.until(ExpectedConditions.textToBe(By.cssSelector(".loginDiv button"), value: "odjavi se!"));

    assertEquals(expected: "http://localhost:8080/", driver.getCurrentUrl());
    assertTrue(logoutPrisutan);
    driver.quit();
}

```

Slika 5.9: Selenium test - prijava korisnika s ispravnim podacima

Testom 5.10 ispitali smo ponašanje sustava prilikom pokušaja prijave s nepostojećim korisničkim imenom i lozinkom. U takvim slučajevima očekujemo zadržavanje na stranici za prijavu.

```
@Test
public void test_login_with_invalid_credentials() {
    WebDriver driver = new FirefoxDriver();
    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
    driver.get("http://localhost:8080/");

    WebElement loginButton = wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector(".loginDiv button")));
    loginButton.click();
    WebElement usernameInput = wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("korisnickoIme")));
    usernameInput.sendKeys("mojstarifrendimarokenrolbend");
    WebElement passwordInput = wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("lozinka")));
    passwordInput.sendKeys("netkotooodgorevidisve");
    WebElement submitButton = wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector("button[type='submit']")));
    submitButton.click();
    Boolean loginPrisutan = wait.until(ExpectedConditions.textToBe(By.cssSelector(".loginDiv button"), value: "prijava se!"));

    assertEquals(expected: "http://localhost:8080/login", driver.getCurrentUrl());
    assertTrue(loginPrisutan);
    driver.quit();
}
```

Slika 5.10: Selenium test - pokušaj prijave nepostojećeg korisnika

Ispitni slučaj 5.11 provjerava ispravnost stvaranja novog zadatka od strane prijavljenog voditelja. Kao rezultat očekujemo pojavljivanje novostvorenog zadatka na korisničkom profilu voditelja.

```

@Test
public void test_create_zadatak() {
    WebDriver driver = new FirefoxDriver();
    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
    driver.get("http://localhost:8080/");

    wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector(".loginDiv button"))).click();
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("korisnickoIme"))).sendKeys(...keysToSend: "buttler");
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("lozinka"))).sendKeys(...keysToSend: "robin");
    wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector("button[type='submit']"))).submit();
    wait.until(ExpectedConditions.urlToBe("http://localhost:8080/"));
    wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//button[text()='zadaci']"))).click();
    wait.until(ExpectedConditions.urlContains("problems/all"));
    wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//button[text()='novi zadatak']"))).click();
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("nazivZadatka"))).sendKeys(...keysToSend: "zivotjemo");
    wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("input[name='tekstZadatka']"))).sendKeys(
        ...keysToSend: "Da se ja pitam, ja bih proterao autobus ovuda."
        " Nije to tako loše kao što izgleda.");
    WebElement dropdown = wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("span.ant-select-selection-item")));
    dropdown.click();
    List<WebElement> bodoviOpcije = wait.until(ExpectedConditions.visibilityOfAllElementsLocatedBy(By.cssSelector(".ant-select-item-option-content")));
    bodoviOpcije.get(1).click();
    wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector("button[type='submit']"))).click();
    wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//button[text()='spremi promjene']"))).click();

    assertTrue(driver.getCurrentUrl().contains("/user/profile/buttler"));
    assertTrue(wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//td[text()='zivotjemo']"))).isDisplayed());
    driver.quit();
}

```

Slika 5.11: Selenium test - dodavanje novog zadatka

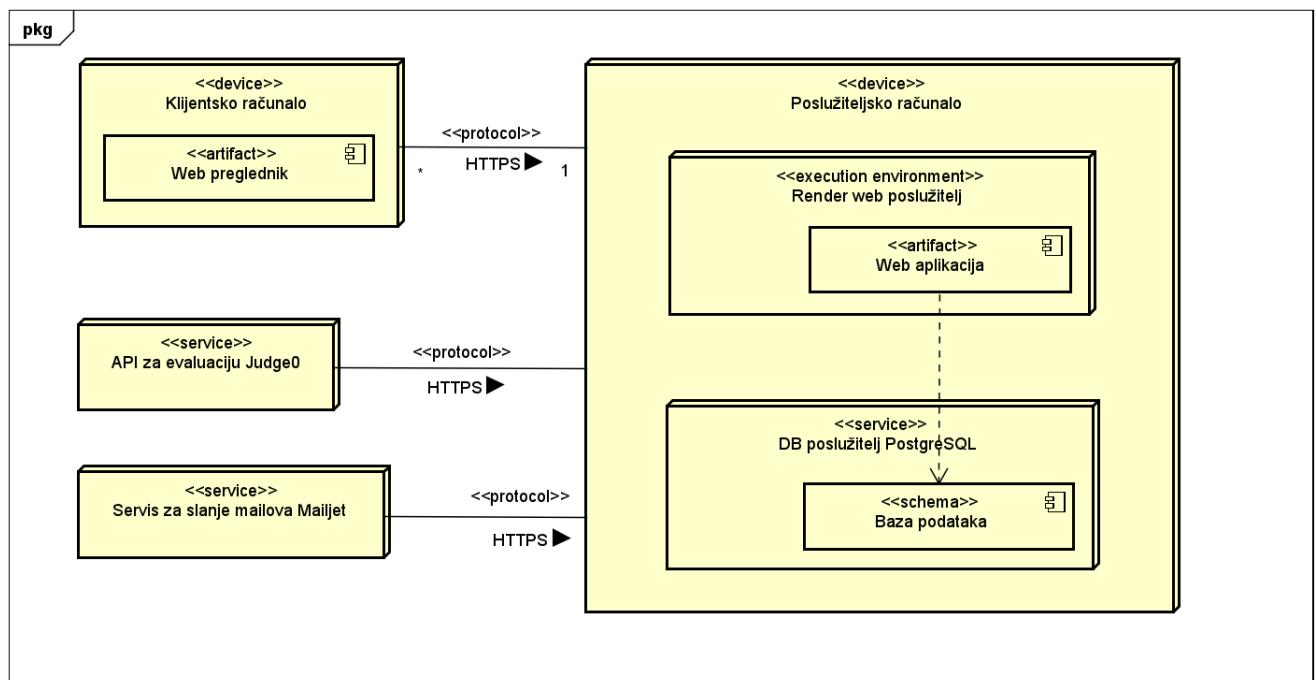
Rezultati ispitivanja sustava mogu se vidjeti na slici 5.12.

BytePitApplicationTests (hr.fer.progi.looneycodes.BytePit)		1 min 1 sec
✓	test_login_with_invalid_credentials()	27 sec 670 ms
✓	test_create_zadatak()	20 sec 833 ms
✓	test_login_with_valid_credentials()	12 sec 734 ms

Slika 5.12: Rezultati testiranja

5.3 Dijagram razmještaja

UML-dijagrami razmještaja prikazuju fizičku arhitekturu i konfiguraciju razmještaja programskog sustava, a pomažu u planiranju održavanja, nadogradnji sustava, identifikaciji potencijalnih uskih grla i pojedinačnih točaka kvara. U našem primjeru, na poslužiteljskom računalu nalaze se Render web poslužitelj i poslužitelj baze podataka PostgreSQL. Poslužiteljsko računalo komunicira s klijentskim računalom putem protokola HTTPS, a na klijentskom računalu pristupamo aplikaciji putem odabranog web preglednika. Osim s klijentskim računalom, poslužiteljsko računalo komunicira i sa servisom za slanje emailova Mailjet, a koristi se i API za evaluaciju programskog rješenja kojeg predaju natjecatelji Judge0.



Slika 5.13: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Za puštanje našeg projekta u pogon može se koristiti Render.com. Ovu smo platformu odabrali zbog njezine jednostavnosti i pristupačnosti. U nastavku ćemo opisati korake koje smo poduzeli kako bismo uspješno postavili našu aplikaciju na Render.com.

Na početku je pripremiti otvoriti GitHub repozitorij i korisnički račun na Renderu. Dovoljno je za kratkoročnu uporabu koristiti besplatni plan na obje platforme. Aplikacija je prilagođena za korištenje izgrađenih statičkih resursa korisničke strane koji se potom dostavljaju s poslužitelja. Prije samog pokretanja u pogon potrebno je generirati te datoteke naredbom *npm run build* iz vršnog foldera implementacije klijentske strane kao što je prikazano na slici dolje.



```
marko@marko-ZenBook-UX434FLC:~/Documents/progi/Looney-Codes/IzvorniKod/frontend/BytePitFE$ npm run build
> bytepitfe@0.0.0 build
> tsc && vite build --emptyOutDir

vite v4.5.1 building for production...
✓ 91 modules transformed.
./../backend/src/main/resources/static/index.html      0.44 kB | gzip: 0.29 kB
./../backend/src/main/resources/static/assets/index-282145cb.css  6.64 kB | gzip: 1.75 kB
./../backend/src/main/resources/static/assets/index-393abdfc.js 220.04 kB | gzip: 68.19 kB
✓ built in 1.57s
```

Slika 5.14: Generiranje statičkih resursa korisničke strane

Prije postavljanja na GitHub potrebno je napraviti Maven Wrapper naredbom *mvn wrapper:wrapper* u vršnom direktoriju izvornog koda poslužiteljske strane te dodati u isti direktorij direktorij naziva *Docker* koji sadrži datoteku *Dockerfile* sadržaja kao na donjoj slici. Sve spomenute datoteke valja spremiti u GitHub repozitorij zajedno s ostatkom izvornog koda.

```
# Container za izgradnju (build) aplikacije
FROM openjdk:17-alpine AS builder

# Kopiranje izvornog koda u container
COPY ./mvnw .
COPY ./mvnw .
COPY ./pom.xml .
COPY ./src src
RUN chmod +x mvnw

# Pokretanje builda
RUN ./mvnw clean package -Dmaven.test.skip

# Stvaranje containera u kojem ce se vrtiti aplikacija
FROM openjdk:17-alpine

## Ovdje je moguce instalirati alate potrebne za rad aplikacije. Vjerojatno vam nece trebati, no dobro je znati.
## Linux distro koji se koristi je Alpine, stoga se kao package manager koristi apk
#RUN apk install <nesto>

# Kopiranje izvrsnog JAR-a iz build containera u izvrsni container
COPY --from=builder target/*.jar /app.jar
# kopiraj mapu s profilnim slikama u docker
RUN mkdir -p src/main/resources
COPY --from=builder src/main/resources/profilneSlike src/main/resources/profilneSlike
# kopiraj mapu s slikama pehara u docker
COPY --from=builder src/main/resources/slikePehara src/main/resources/slikePehara

# Izlaganje porta
EXPOSE 8080

# Naredba kojom se pokrece aplikacija
ENTRYPOINT ["java","-jar","/app.jar"]
```

Slika 5.15: Dockerfile korišten za puštanje u pogon

Sada prelazimo na pokretanje aplikacije na Render platformi. Koraci koje treba obaviti su:

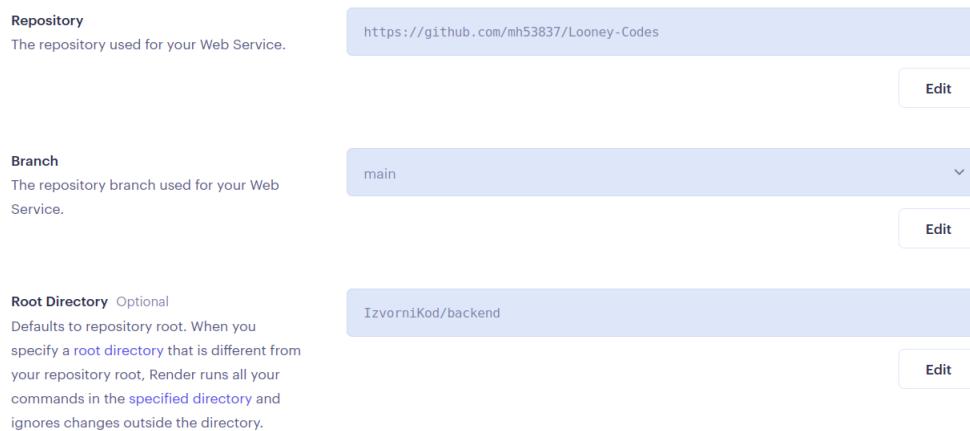
- **Stvaranje baze podataka**

Po izradi računa treba otvoriti novu PostgreSQL bazu podataka pritiskom na *new* gumb u izborniku početne stranice i odabirom prikladne opcije. Postaviti željeno ime baze i korisnika te regiju na Frankfurt (EU Central). U *application.properties* datoteci treba postaviti *spring.datasource.url* na dobivenu adresu baze i *spring.datasource.username* te *spring.datasource.password* na dobivene korisničke podatke, a promjene pohraniti na GitHub repozitorij.

- **Stvaranje novog web servisa**

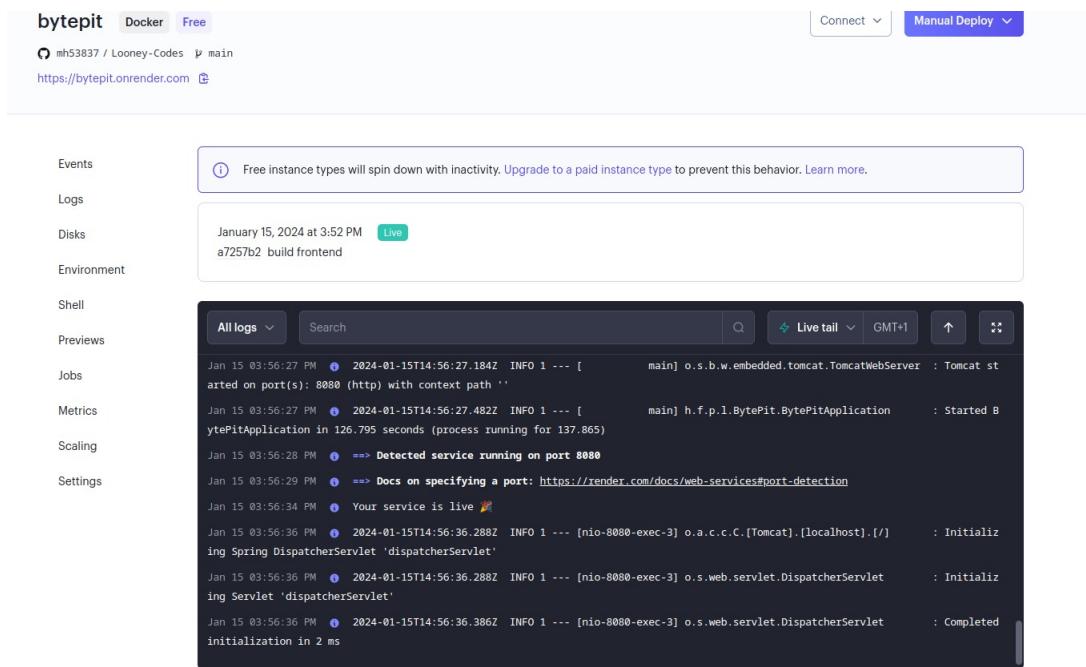
Slijedi inicijalizacija novog web servisa za poslužitelj naše stranice. Odabir početka stvaranja takvog servisa vrši se na sličan način kao i stvaranje nove baze podataka - pritiskom *new* gumb u izborniku početne stranice i odabirom prikladne opcije.

Pri početnom odabiru odabrati *Build and deploy from a Git repository* te zatim povezati s željenim GitHub repozitorijem. Na slici 5.16 prikazan je odabir vršnog direktorija i grane repozitorija. Kao *Environment* odabrati Docker, a vrijednost putanje do *Dockerfilea* treba biti postavljena na *./Docker/Dockerfile*. Nije potrebno postavljati varijable okruženja jer su postavljene u aplikacijskim specifikacijama. Završiti kreiranje novog web servisa pritiskom na gumb *Create Web Service*.



Slika 5.16: Povezivanje s GitHub repozitorijem

- **Pokretanje web servisa** Po stvaranju web servisa počet će proces inicijalizacije poslužitelja. Prilikom toga na karticama *Events* i *Logs* potrebno je pratiti uspjehost puštanja u pogon kako bi se uvidjeli eventualni problemi i ispravili isti (ako pažljivo nije proveden neki od prethodnih koraka treba se vratiti i ponoviti ga). U logovima moguća je vidjeti kako se prvo vrši alociranje samog poslužitelja, pa *dokerizacija* i na kraju se pokreće naša aplikacija. Kada je završeno puštanje u pogon moguće je vidjeti obavijest kao na slici ispod.



Slika 5.17: Završetak pokretanja web servisa

Nakon završetka puštanja u pogon na Renderu, moguće se spojiti na bazu s prije-dobivenim korisničkim podacima. Ovo je moguće napraviti kroz grafičko sučelje (primjerice PGAdmin) ili kroz naredbenu liniju. Po spajanju se dolje prikazanom naredbom u bazu podataka mogu dodati inicijalni korisnici i primjeri ostalih entiteta.

```

→ psql postgres:
DELETE 0
DELETE 0
DELETE 0
DELETE 0
CREATE SEQUENCE
DROP SEQUENCE
DROP SEQUENCE
DROP SEQUENCE
CREATE SEQUENCE
CREATE SEQUENCE
CREATE SEQUENCE
CREATE SEQUENCE
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1

```

postgres.render.com/bytepit -f backend/src/main/resources/data.sql

Slika 5.18: Dodavanje podataka u bazu

Primjer naše aplikacije moguće je pronaći na poveznici bytepit.onrender.com.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinzh>
2. Visual Paradigm, <https://www.visual-paradigm.com/>
3. Spring Framework Documentation, <https://docs.spring.io/spring-framework-reference/index.html>
4. ReactJS Legacy Documentation, <https://legacy.reactjs.org/docs/getting-started.html>
5. ViteJS, <https://vitejs.dev>
6. Astah, <https://astah.net/>
7. Ant Design, <https://ant.design/>

Indeks slika i dijagrama

2.1	Sustav SPOJ - pregled zadataka na stranici	8
2.2	Sustav Edgar - pregled statistike ispita	8
2.3	Sustav Codeforces - pregled statusa na natjecanju	9
2.4	Sustav Codeforces - kalendar nadolazećih natjecanja	9
2.5	Sustav SPOJ - profil natjecatelja	10
3.1	Obrasci uporabe - funkcionalnosti za neprijavljenе korisnike	21
3.2	Obrasci uporabe - funkcionalnosti za natjecatelje i voditelje	22
3.3	Obrasci uporabe - funkcionalnosti za administratore	23
3.4	Sekvencijski dijagram za obrazac uporabe - unos novog zadatka . .	25
3.5	Sekvencijski dijagram za obrazac uporabe - organiziranje novog natjecanja	27
3.6	Sekvencijski dijagram za obrasce uporabe - rješavanje zadataka i prijenos datoteke	29
4.1	Prikaz arhitekture sustava	31
4.2	Dijagram baze podataka	38
4.3	Dijagram razreda - Servisi	40
4.4	Dijagram razreda - Kontroleri, Repozitoriji i Modeli	41
4.5	Dijagram aktivnosti - organiziranje novog natjecanja	42
4.6	Dijagram komponenti	43
5.1	Testirajuća metoda - registracija korisnika	45
5.2	Testirajuća metoda - neuspješna registracija korisnika	46
5.3	Testirajuća metoda - potvrda zatražene uloge	46
5.4	Testirajuća metoda - stvaranje natjecanja	47
5.5	Testirajuća metoda - neuspješno stvaranje natjecanja	47
5.6	Testirajuća metoda - dohvati javnih zadataka	48
5.7	Testirajuća metoda - ažuriranje zadatka od strane administratora . .	48
5.8	Rezultati testiranja	49
5.9	Selenium test - prijava korisnika s ispravnim podacima	49

5.10 Selenium test - pokušaj prijave nepostojećeg korisnika	50
5.11 Selenium test - dodavanje novog zadatka	51
5.12 Rezultati testiranja	51
5.13 Dijagram razmještaja	52
5.14 Generiranje statičkih resursa korisničke strane	53
5.15 Dockerfile korišten za puštanje u pogon	54
5.16 Povezivanje s GitHub repozitorijem	55
5.17 Završetak pokretanja web servisa	56
5.18 Dodavanje podataka u bazu	56

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 18. listopada 2023.
- Prisustvovali: Vedran Ćutić, Antonio Glavaš, Marina Hrbud, Lara Marčec, Jakov Novak, Marko Varga, Nikola Vlahović
- Teme sastanka:
 - upoznavanje
 - analiza zadatka
 - postavljanje GitHub-a

2. sastanak

- Datum: 20. listopada 2023.
- Prisustvovali: Vedran Ćutić, Antonio Glavaš, Marina Hrbud, Lara Marčec, Jakov Novak, Marko Varga, Nikola Vlahović
- Teme sastanka:
 - uvodni sastanak s asistentom i demonstratoricom
 - razrješavanje upita o osnovnim funkcionalnostima aplikacije
 - dogovor oko alata i tehnologija

3. sastanak

- Datum: 31. listopada 2023.
- Prisustvovali: Vedran Ćutić, Antonio Glavaš, Marina Hrbud, Lara Marčec, Jakov Novak, Marko Varga, Nikola Vlahović
- Teme sastanka:
 - pregled odrđenih zadataka
 - dogovor o vizualnom izgledu aplikacije
 - raspodjela dalnjih poslova i dogovor internih rokova za iste

4. sastanak

- Datum: 7. studenog 2023.
- Prisustvovali: Vedran Ćutić, Antonio Glavaš, Marina Hrbud, Lara Marčec,

Jakov Novak, Marko Varga, Nikola Vlahović

- Teme sastanka:

- zajednička diskusija o dosad odrađenim zadacima
 - konkretna raspodjela dalnjih poslova i uloga

5. sastanak

- Datum: 15. studenog 2023.
- Prisustvovali: Vedran Ćutić, Antonio Glavaš, Marina Hrbud, Lara Marčec, Jakov Novak, Marko Varga, Nikola Vlahović
- Teme sastanka:
 - pregled svih odrađenih zadataka
 - diskusija o predaji projekta 17.11.
 - deployment aplikacije

6. sastanak

- Datum: 11. prosinca 2023.
- Prisustvovali: Vedran Ćutić, Antonio Glavaš, Marina Hrbud, Lara Marčec, Jakov Novak, Marko Varga, Nikola Vlahović
- Teme sastanka:
 - diskusija o ostvarenom uspjehu nakon prve predaje
 - detaljni pregled preostalih zadataka
 - raspodjela poslova

7. sastanak

- Datum: 10. siječnja 2024.
- Prisustvovali: Vedran Ćutić, Antonio Glavaš, Marina Hrbud, Lara Marčec, Jakov Novak, Marko Varga, Nikola Vlahović
- Teme sastanka:
 - pregled odrađenih zadataka
 - dogovor o dalnjem nastavku rada

Tablica aktivnosti

Kontinuirano osvježavanje

Napomena: Doprinose u aktivnostima treba navesti u satima po članovima grupe po aktivnosti.

	Vedran Ćutić	Antonio Glavaš	Marina Hrbud	Lara Marčec	Jakov Novak	Marko Varga	Nikola Vlahović
Upravljanje projektom	11	11	11	11	11	11	11
Opis projektnog zadatka			2			2	
Funkcionalni zahtjevi				1			2
Opis pojedinih obrazaca				2			1
Dijagram obrazaca					2	1	
Sekvencijski dijagrami				4			
Opis ostalih zahtjeva				1			
Arhitektura i dizajn sustava		3					
Baza podataka	1		4			2	
Dijagram razreda		3					
Dijagram stanja							
Dijagram aktivnosti				2			
Dijagram komponenti	2						
Korištene tehnologije i alati			1				
Ispitivanje programskog rješenja	2						
Dijagram razmještaja		2					
Upute za puštanje u pogon						1	

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Vedran Ćutić	Antonio Glavaš	Marina Hrbud	Lara Marčec	Jakov Novak	Marko Varga	Nikola Vlahović
Dnevnik sastajanja		1	2				
Zaključak i budući rad							
Popis literature							
<i>Dodatne stavke kako ste podijelili izradu aplikacije</i>							
izrada front-end stranice		20	30		15		
izrada baze podataka		4			4		
spajanje s bazom podataka	1				5	3	
back-end - servisi i controlleri	7	8			15	20	13
deployment					5	6	