Programsko inženjerstvo

Ak. god. 2023./2024.

BytePit

Dokumentacija, Rev. *1*

Grupa: *Rade*

Voditelj: *Marko Bolt*

Datum predaje: *17. 11. 2023.*

Nastavnik: *Hrvoje Nuić*

Sadržaj

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Marko Bolt	04.11.2023.
0.2	Napravljen opis projekta	Marko Bolt	05.11.2023.
0.3	Napravljeni funkcionalni zahtjevi	Jakov Vinožganić Teo Radolović	09.11.2023.
0.4	Napravljeni obrasci uporabe	Filip Bernt	10.11.2023.
0.5	Napravljeni sekvencijski dijagrami	Fran Sipić	10.11.2023.
0.6	Napravljeni ostali zahtjevi	Jure Franjković	12.11.2023.
0.7	Napravljena arhitektura i dizajn sustava	Jure Franjković	13.11.2023.
0.8	Napravljen opis baze podataka	Filip Bernt Fran Sipić	14.11.2023.
0.9	Napravljen dijagram razreda	Jure Franjković	15.11.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.0	Napravljena finalna inačica za prvi ciklus	Filip Bernt	17.11.2023.
1.1	Napravljen dijagram komponenti	Filip Bernt	19.12.2023.
1.2	Napravljen dijagram aktivnosti i implementacijski dijagram razreda	Jure Franjković	23.12.2023.
1.3	Napravljen dijagram stanja i dijagram razmještaja	Fran Sipić	23.12.2023.
1.4	Napravljene upute za puštanje u pogon i popis literature	Filip Bernt	12.1.2024.
1.5	Prepravljeni dijagrami razreda i napisane korištene tehnologije	Jure Franjković	12.1.2024.
1.6	Napisan zaključak	Marko Bolt	16.1.2024.
1.7	Ispitivanje programskog sučelja	Fran Sipić Filip Bernt	18.1.2024.

2. Opis projektnog zadatka

Aplikacija ByteBit inovativno je rješenje namijenjeno provjeri programerskih vještina, omogućavajući korisnicima sudjelovanje u programskim natjecanjima i vježbanje rješavanja zadataka. Aplikacija se može implementirati kao web ili desktop platforma koristeći objektno-orijentirane programerske jezike, sa podrškom za kompilaciju i evaluaciju rješenja u najmanje jednom programskom jeziku.

Glavne funkcionalnosti aplikacije uključuju:

- **Pregledavanje zadataka i kalendar natjecanja:** Korisnici mogu pregledavati dostupne zadatke i kalendar natjecanja, što omogućava planiranje sudjelovanja i pripremu.
- **Profili korisnika:** Natjecatelji imaju profile s prikazom statistika njihovih rezultata, uključujući broj točno riješenih i isprobanih zadataka, kao i pehare za osvojena natjecanja. Profili voditelja sadrže informacije o učitanim zadacima i organiziranim natjecanjima.
- **Registracija korisnika:** Registracija zahtijeva osnovne podatke kao što su korisničko ime, fotografija, lozinka, ime, prezime i email adresa. Administrator potvrđuje registraciju, a u slučaju voditelja natjecanja potrebna je i dodatna verifikacija.
- **Natjecanja:** Početkom natjecanja, zadaci postaju vidljivi aktivnim natjecateljima. Nakon završetka, zadaci su dostupni svima. Postoji mogućnost slanja datoteka s programskim kodom i objavljivanje rang lista po bodovima, uzimajući u obzir vrijeme i točnost rješenja.
- **Prikaz rješenja:** Natjecatelji mogu vidjeti rješenja drugih natjecatelja nakon natjecanja, dok su tijekom natjecanja vidljiva samo ako su sami točno riješili zadatak.
- **Vježbanje:** Natjecatelji mogu vježbati zadatake i učitavati rješenja, s automatskom provjerom točnosti i vremenskih ograničenja.
- **Virtualna natjecanja:** Natjecatelji mogu stvoriti vlastita virtualna natjecanja odabirom prošlih natjecanja ili nasumičnim odabirom zadataka, a rezultati se uspoređuju s originalnim.

- **Uloga voditelja:** Voditelji imaju mogućnost učitavanja novih zadataka, organiziranja natjecanja, i odabira zadataka koji će biti aktivni tijekom natjecanja, uz mogućnost uređivanja vlastitih zadataka.
- **Administrativne funkcije:** Administratori imaju ovlasti za upravljanje korisničkim računima i uređivanjem svih zadataka i natjecanja, bez mijenjanja prethodno ostvarenih rezultata.

Trenutno postoje brojne platforme poput Codeforces, LeetCode, HackerRank i drugih koje nude slične funkcionalnosti. Razlike u odnosu na BytePit bi se mogle ogledati u specifičnostima kao što su lokalizacija (jezik i regionalni fokus), specijalizacija za određene programerske jezike ili tehnologije, ili u razini personalizacije korisničkog iskustva. Slična rješenja i razlike:

- **Codeforces**: Fokusira se na konkurentno programiranje s redovitim natjecanjima. BytePit bi mogao uključivati više edukativnih resursa i prilagodljiva virtualna natjecanja.
- **LeetCode**: Specijaliziran za pripremu tehničkih intervjua. BytePit može nuditi širi raspon zadatka, uključujući one van konteksta intervjua.
- **HackerRank**: Osim izazova u programiranju, nudi i podršku za tvrtke u regrutiranju. BytePit bi mogao dodati vrednost kroz specifične profile natjecanja i detaljnu statistiku.Sličan koncept, ali s fokusom na poslovne korisnike. BytePit bi mogao biti više usmjeren na edukaciju i natjecanja.

BytePit bi mogao privući širok spektar korisnika, od početnika u programiranju koji traže mjesto za učenje i prakticiranje, do iskusnih programera koji žele oštriti svoje vještine ili sudjelovati u natjecanjima, te edukativnih institucija koje traže platformu za organiziranje natjecanja ili kao alat u nastavi. Ova raznolikost korisnika stvara dinamično okruženje u kojem se znanje i iskustvo neprestano razmjenjuju, pružajući beskrajne mogućnosti za učenje i napredak. S jedne strane, početnici imaju priliku učiti kroz praksu, rješavajući stvarne probleme i dobivajući povratne informacije od iskusnijih kolega. S druge strane, iskusni programeri mogu se suočavati s izazovnijim problemima i sudjelovati u zahtjevnijim natjecanjima, što ih stavlja u poziciju mentora, potičući ih na daljnji razvoj i dijeljenje znanja. Edukativne institucije mogu koristiti BytePit za stvaranje prilagođenih tečajeva i kurikuluma, pružajući studentima praktično iskustvo koje je ključno za

njihov razvoj. Poslodavci koji traže talentirane programere mogli bi koristiti platformu za identifikaciju potencijalnih kandidata, provodeći natjecanja ili izazove koji služe kao dio procesa zapošljavanja ili kroz analizu rješenja i pristupa problemima koje programeri koriste na platformi.

BytePit bi trebao biti projektiran s mogućnostima prilagodbe, što bi uključivalo konfigurabilne module za natjecanja, personalizirane profile korisnika, i fleksibilan sustav ocjenjivanja. Prilagodba omogućava korisnicima da kreiraju jedinstveno iskustvo koje odgovara njihovim specifičnim potrebama i stilu učenja ili rada. Personalizirani profili omogućavaju korisnicima da predstave svoje vještine i postignuća, kao i da prate svoj napredak kroz različite metrike i statistike. Fleksibilan sustav ocjenjivanja omogućava različite forme evaluacije, od standardnog bodovanja do složenih algoritama koji mogu vrednovati efikasnost koda ili kreativno rješavanje problema. Ovo omogućava platformi da raste i evoluira zajedno sa svojim korisničkim bazama i tehnološkim trendovima, ostajući relevantna i privlačna za široki spektar korisnika, od individualaca i obrazovnih institucija do korporacija i tehničkih zajednica.

Opseg projekta uključuje dizajniranje i implementaciju korisničkih sučelja, koje moraju biti intuitivna i pristupačna, osiguravajući ugodno iskustvo za sve korisnike, neovisno o njihovom tehničkom znanju ili iskustvu u programiranju. Backend servisi za upravljanje bazom podataka moraju biti robustni i optimizirani za brzu obradu upita, što je ključno za održavanje visoke performanse platforme čak i pod velikim opterećenjem. Sustav za evaluaciju koda mora biti precizan i pouzdan, sposoban za brzo i efikasno procjenjivanje raznovrsnih programskih rješenja koja korisnici predaju. Sustav za autentifikaciju i autorizaciju korisnika treba osigurati da su svi podaci sigurni i da pristup platformi

Nadogradnje projekta mogu uključivati:

- Integraciju s vanjskim servisima poput GitHub-a za uvoz postojećeg koda.
- Implementaciju naprednijih algoritama za detekciju plagijata.
- Proširenje podrške za više programskih jezika i tehnoloških okvira.
- Razvoj mobilne aplikacije za pristup platformi.
- Uvođenje umjetne inteligencije za personalizirane preporuke zadataka na temelju korisnikovih prethodnih aktivnosti i uspjeha.

- ”Gamification” sustav za nagrađivanje korisnika za postignuća i napredak.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Administrator
2. Voditelj natjecanja
3. Natjecatelj
4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Administrator (inicijator) može:
   (a) pregledati korisnike
   (b) pregledati natjecanja
   (c) pregledati zadatke
   (d) mijenjati prava korisnicima
2. Neregistrirani korisnik (inicijator) može:
   (a) se registrirati u sustav
   (b) stvoriti novi korisnički račun
3. Voditelj natjecanja (inicijator) može:
   (a) kreirati nove zadatke
   (b) kreirati nova natjecanja
   (c) pregledati svoje osobne podatke
4. Natjecatelj (inicijator) može:
   (a) pristupiti natjecanjima
   (b) pregledati svoje osobne podatke
   (c) pregledati zadatke
   (d) predati rješenje zadatka

5. Baza podataka (sudionik):
   (a) pohranjuje sve podatke o korisnicima
   (b) pohranjuje sve podatke o natjecanjima i zadacima

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Pregled početne stranice

- **Glavni sudionik:** Korisnik
- **Cilj:** Odlučiti se za opciju registracije ili prijave
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Prikazuje se početna stranica
  2. Odabir prijave ili registracije
  3. Prosljeđivanje na odabranu opciju

UC2 - Registracija

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Odlučiti se za opciju registracije kao voditelj ili natjecatelj
- **Sudionici:** Baza podataka
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
  1. Prikazuje se stranica za registraciju
  2. Neregistrirani korisnik ispunjava zadane podatke
  3. Odabire opciju registracije za voditelja ili natjecatelja
  4. Potvrđivanje registracije putem email-a
  5. Dodatna potvrda administratora za novog voditelja
- **Opis mogućih odstupanja:**

  2.a Odabir već zauzetog korisničkog imena i/ili e-maila, unos korisničkog podatka u nedozvoljenom formatu ili pružanje neispravnoga e-maila
  1. Sustav odbija odobravanje registracije
  2. Korisnik mijenja podatke u ispravan unos ili odustaje od registracije

UC3 - Prijava u sustav

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Prijava u sustav i pristup korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
  1. Prikazuje se stranica za prijavu
  2. Unos emaila i lozinke
  3. Pristup korisničkoj stranici
- **Opis mogućih odstupanja:**

  2.a Neispravan unos emaila ili lozinke
  1. Sustav odbija pristup te nudi ponovni pokušaj za prijavu

UC4 - Pregled osobnih podataka

- **Glavni sudionik:** Natjecatelj, voditelj
- **Cilj:** Pregledati osobne podatke
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Pritisak korisnika na svoj profil
  2. Prikaz osobnih podataka

UC5 - Pregled zadataka

- **Glavni sudionik:** Natjecatelj, voditelj
- **Cilj:** Pregledati zadatke na stranici
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik pritiskom na dostupne zadatke dobiva pristup
  2. Korisnik može pregledavati već objavljene zadatke

UC6 - Pregled kalendara

- **Glavni sudionik:** Natjecatelj, voditelj
- **Cilj:** Pregledati kalendar
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik na svojoj stranici ima pristup kalendaru
  2. Korisnik unutar kalendara može pregledati datume s dostupnim natjecanjima

UC7 - Pregled profila drugih korisnika

- **Glavni sudionik:** Natjecatelj, voditelj
- **Cilj:** Pregledati profile drugih korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Korisnik na svojoj stranici odabire profil koji želi vidjeti
  2. Ako je pritisnuti profil natjecatelj prikazuje se statistika o zadatcima i osvojeni pehari
  3. Ako je pritisnuti profil voditelj prikazuje se popis učitanih zadataka i kalendar s popisom natjecanja

UC8 - Sudjelovanje u natjecanju

- **Glavni sudionik:** Natjecatelj
- **Cilj:** Prikazati natjecatelju zadatke za odabrano natjecanje
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i postoji natjecanje u kojem može sudjelovati
- **Opis osnovnog tijeka:**
  1. Natjecatelj sudjeluje u natjecanju
  2. Završetkom natjecanja zadaci postaju dostupni svim korisnicima za vježbanje
  3. Objava rang liste i dodjeljivanje pehara za prva tri mjesta
- **Opis mogućih odstupanja:**

  2.a Pokušaj sudjelovanja u natjecanju korisnika koji nije natjecatelj
    1. Sustav odbija sudjelovanje voditelja u natjecanju

UC9 - Kreiranje natjecanja

- **Glavni sudionik:** Voditelj
- **Cilj:** Voditelj stvara natjecanje
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i odobren kao voditelj
- **Opis osnovnog tijeka:**
  1. Voditelj ispunjava formu za izradu natjecanja
  2. Voditelj odabire željene zadatke za natjecanje
  3. Natjecanje se kreira i sprema u bazu podataka
- **Opis mogućih odstupanja:**

  2.a Pokušaj kreiranja natjecanja korisnika koji nije voditelj
  1. Sustav odbija kreiranje natjecanja i šalje poruku da korisnik nema ovlasti za taj postupak

UC10 - Kreiranje zadatka

- **Glavni sudionik:** Voditelj
- **Cilj:** Voditelj stvara zadatak
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i odobren kao voditelj
- **Opis osnovnog tijeka:**
  1. Voditelj ispunjava formu za izradu zadatka
  2. Voditelj unosi naziv, opis zadatka, testne primjere, broj bodova koji nosi zadatak te ih prosljeđuje sustavu
  3. Voditelj odabire opciju privatnog zadatka, on će postati javan tek nakon završetka natjecanja
  4. Sustav pohranjuje zadatak u bazu podataka
- **Opis mogućih odstupanja:**

  2.a Pokušaj kreiranja zadatka korisnika koji nije voditelj
  1. Sustav odbija kreiranje zadatka i šalje poruku da korisnik nema ovlasti za taj postupak

UC11 - Predaja rješenja zadatka

- **Glavni sudionik:** Natjecatelj
- **Cilj:** Natjecatelj predaje svoje rješenje zadatka
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Natjecatelj šalje zahtjev za evaulaciju rješenja
  2. Evaulator prosljeđuje compileru zahtjev
  3. Compiler pokreće testne primjere nad programom
  4. Evaulator računa broj bodova nakon dobivenih rezultata od compilera
  5. Sustav ažurira broj bodova za natjecatelja i pohranjuje ih u bazu
- **Opis mogućih odstupanja:**

  2.a Natjecatelj ne predaje rješenje zadatka
    1. Natjecatelj ne predaje rješenje zadatka te se ne pokreće proces ocjenjivanja već dodjeljuje 0

UC12 - Pregled učitanih rješenja

- **Glavni sudionik:** Natjecatelj
- **Cilj:** Natjecatelj pregledava rješenja drugih natjecatelja
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i natjecanje u kojem je sudjelovao je završilo
- **Opis osnovnog tijeka:**
  1. Natjecatelj odabire zadatak unutar natjecanja
  2. Prikazuje mu se popis svih natjecatelja koji su učitali neko rješenje
  3. Rješenja su prikazana sortirano
  4. Ako je natjecatelj potpuno točno riješio zadatak pojavljuje se opcija za preuzimanje rješenja

UC13 - Vježbanje zadataka

- **Glavni sudionik:** Natjecatelj
- **Cilj:** Natjecatelj ima mogućnost vježbanja već objavljenih zadataka
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Natjecatelj odabire popis objavljenih zadataka
  2. Natjecatelj pritiskom otvara zadatak koji želi vježbati
  3. Nakon rješavanja zadatka učitava programski kod u aplikaciju
  4. Natjecatelju se dodjeljuju bodovi proporcionalno broju točno riješenih primjera
- **Opis mogućih odstupanja:**

  2.a Natjecatelj ne predaje rješenje zadatka
  1. Natjecatelj ne predaje rješenje zadatka te se ne pokreće proces ocjenjivanja već dodjeljuje 0

UC14 - Izrada virtualnog natjecanja od prošlih natjecanja

- **Glavni sudionik:** Natjecatelj
- **Cilj:** Natjecatelj ima mogućnost vježbanja na već objavljenim natjecanjima
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Natjecatelj otvara popis prošlih natjecanja
  2. Natjecatelj odabire neko natjecanje
  3. Natjecanje je aktivno samo za njega
  4. Nakon završetka natjecatelja se rangira po službenim rješenjima

UC15 - Izrada virtualnog natjecanja od nasumičnih zadataka

- **Glavni sudionik:** Natjecatelj
- **Cilj:** Natjecatelj ima mogućnost vježbanja natjecanja uz pomoć objavljenih zadataka simuliranih u natjecanje
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav
- **Opis osnovnog tijeka:**
  1. Natjecatelj odabire opciju natjecanja napravljenog od nasumičnih objavljenih zadataka
  2. Zadatci se spremaju u natjecanje ravnomjerno po količini bodova
  3. Natjecanje je aktivno samo za njega
  4. Završetkom natjecanja ima pristup rješenjima zadataka

UC16 - Pregled korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Administrator ima mogućnost pregleda svih korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je registriran i ima prava administratora
- **Opis osnovnog tijeka:**
  1. Administrator odabire opciju pregleda korisnika
  2. Prikazuje se lista ispravno registriranih korisnika

UC17 - Uređivanje vlastitih natjecanja i zadataka

- **Glavni sudionik:** Voditelj
- **Cilj:** Mogućnost izmjene zadataka i natjecanja koje je napravio voditelj
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i odobren je kao voditelj
- **Opis osnovnog tijeka:**
  1. Voditelj odabire natjecanje ili zadatak
  2. Ako je autor pruža mu se opcija za uređivanje
  3. Nakon uređivanja sprema promjene, koje ne utječu na prethodne rezultate

UC18 - Uređivanje svih natjecanja i zadataka

- **Glavni sudionik:** Administrator
- **Cilj:** Mogućnost izmjene svih zadataka i natjecanja
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i ima prava administratora
- **Opis osnovnog tijeka:**
  1. Administrator odabire natjecanje ili zadatak
  2. Administrator mijenja zadatke ili natjecanja
- **Opis mogućih odstupanja:**

  2.a Administrator nakon izmjene ne pohrani promjene
  1. Sustav obavještava administratora da nije pohranio promjene

UC19 - Promjena prava pristupa

- **Glavni sudionik:** Administrator
- **Cilj:** Promijeniti razinu pristupa korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i ima prava administratora
- **Opis osnovnog tijeka:**
  1. Administrator pronalazi željenog korisnika
  2. Administrator mijenja razinu prava korisnika
- **Opis mogućih odstupanja:**

  2.a Administrator nakon izmjene ne pohrani promjene
  1. Sustav obavještava administratora da nije pohranio promjene

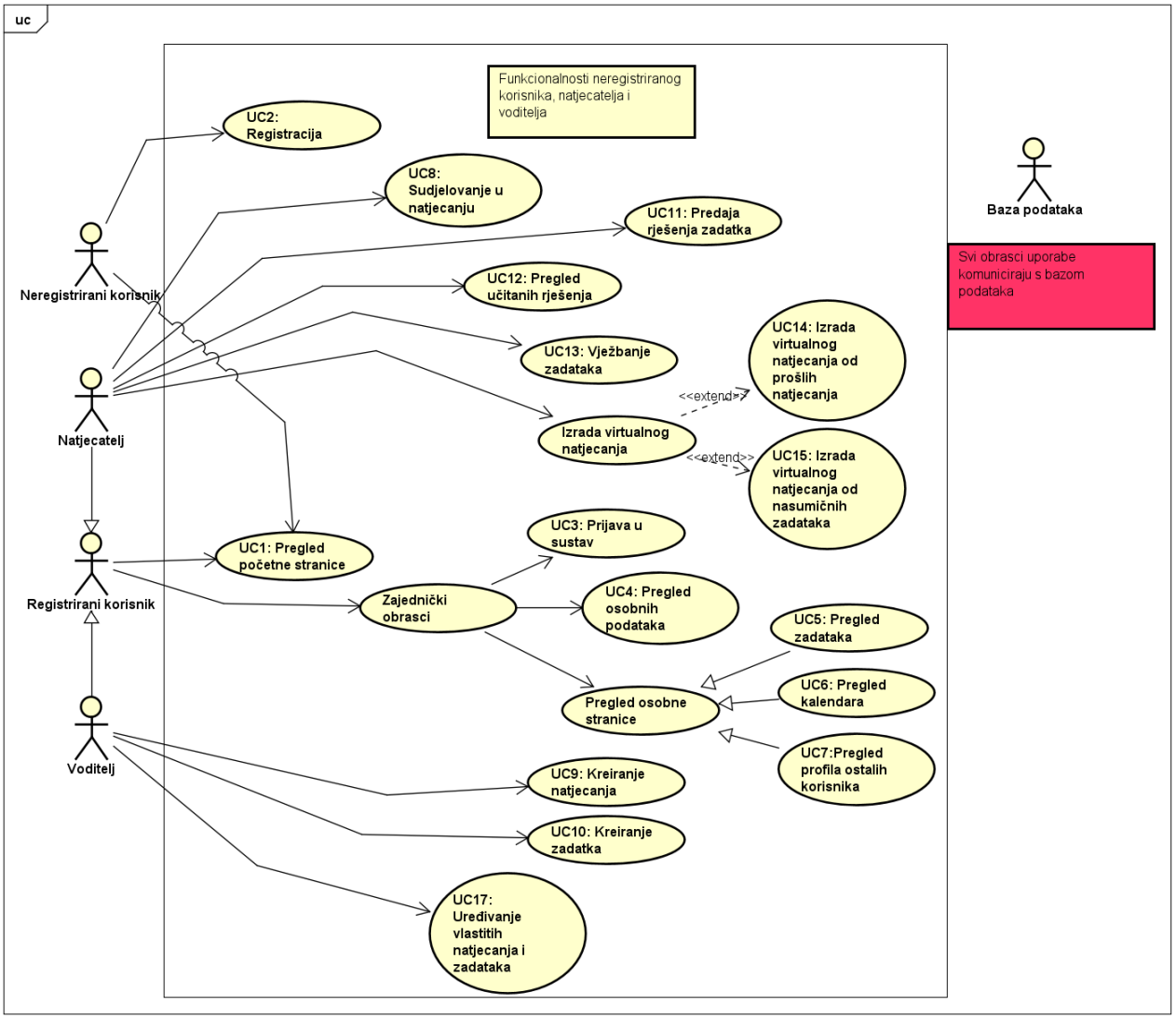
UC20 - Promjena osobnih podataka

- **Glavni sudionik:** Administrator
- **Cilj:** Mogućnost izmjene osobnih podataka korisnika
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i ima prava administratora
- **Opis osnovnog tijeka:**
  1. Administrator pronalazi željenog korisnika
  2. Administrator mijenja osobne podatke korisnika
  3. Administrator sprema promjene
  4. Baza podataka se ažurira
- **Opis mogućih odstupanja:**
  - 2.a Administrator nakon izmjene ne pohrani promjene
    1. Sustav obavještava administratora da nije pohranio promjene

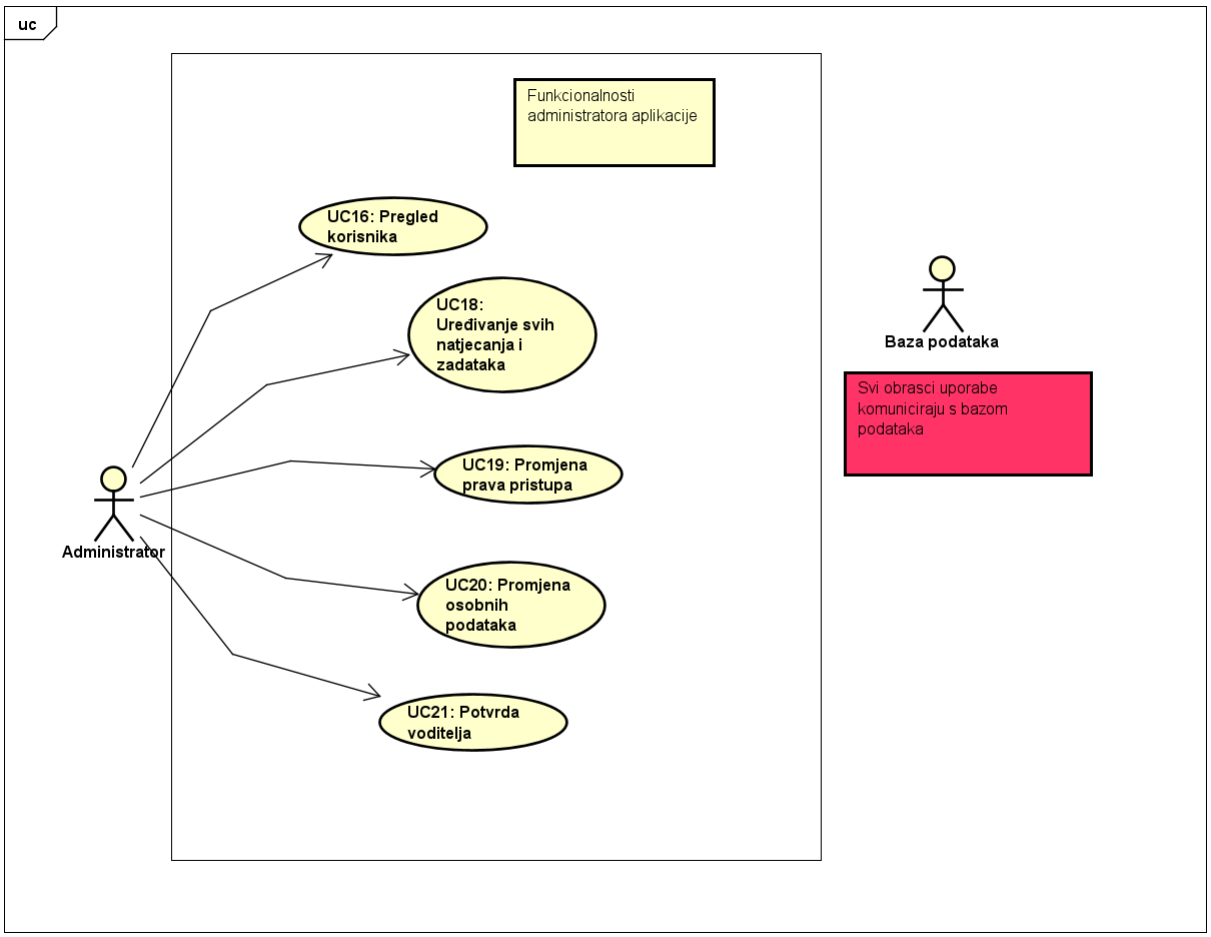
UC21 - Potvrda voditelja

- **Glavni sudionik:** Administrator
- **Cilj:** Administrator potvrđuje voditelja
- **Sudionici:** Baza podataka
- **Preduvjet:** Korisnik je prijavljen u sustav i ima prava administratora
- **Opis osnovnog tijeka:**
  1. Administrator pronalazi željenog korisnika
  2. Administrator mu odobrava pravo voditelja
  3. Baza podataka se ažurira

Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, fukcionalnost natjecatelja, voditelja i neregistriranog korisnika

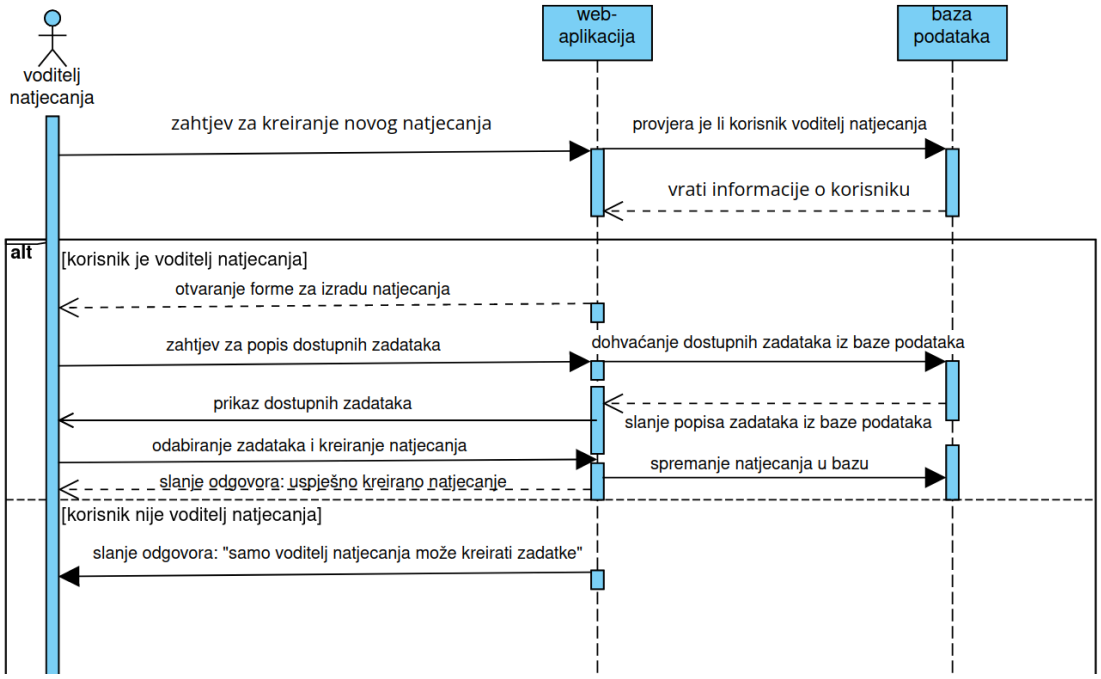


Slika 3.2: Dijagram obrasca uporabe, fukcionalnost administratora

3.1.2 Sekvencijski dijagrami

UC9 Kreiranje natjecanja

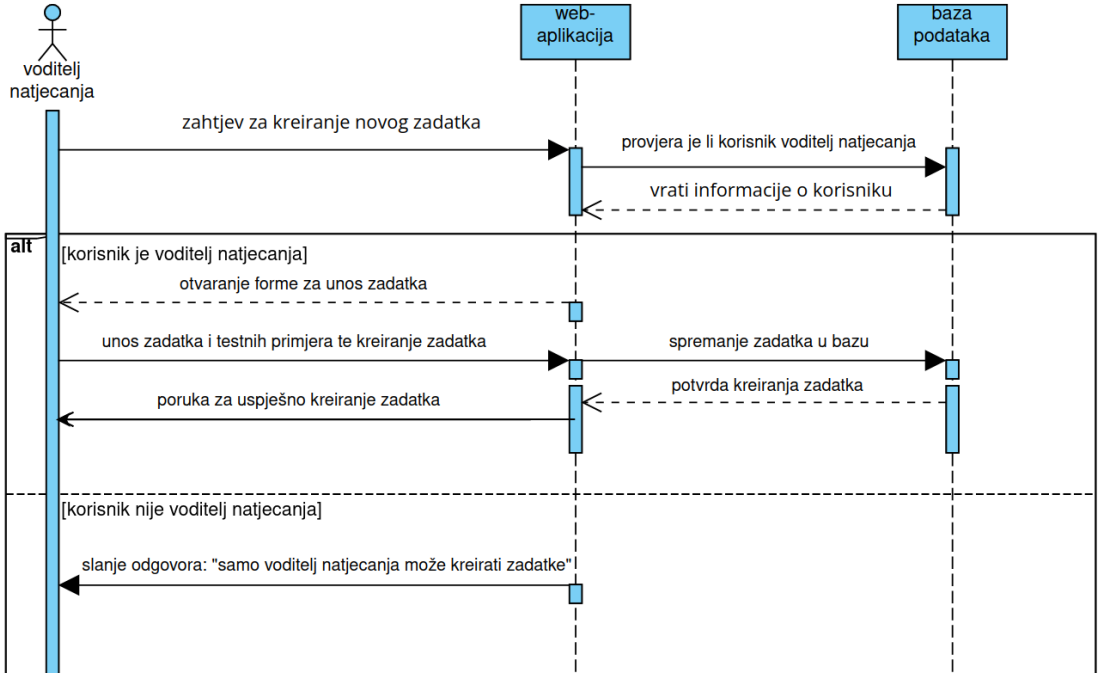
Voditelj natjecanja započinje proces kreiranja natjecanja slanjem zahtjeva poslužitelju. Poslužitelj provjerava informacije o korisniku u bazi podataka kako bi utvrdila je li korisnik ovlašten za kreiranje natjecanja. Ako je korisnik voditelj natjecanja na web-aplikaciji otvara se forma za izradu natjecanja, voditelj šalje zahtjev za popis dostupnih zadataka, poslužitelj ih dohvaća iz baze podataka i prikazuje ih voditelju. Zatim voditelj odabire željene zadatke i kreira natjecanje. Poslužitelj sprema informacije o natjecanju u bazu podataka i šalje povratnu informaciju voditelju o uspješnom kreiranju natjecanja. Ako korisnik nije voditelj natjecanja poslužitelj šalje odgovor voditelju da samo voditelji natjecanja mogu kreirati zadatke.



Slika 3.3: Sekvencijski dijagram kreiranja novog natjecanja

UC10 Kreiranje zadatka

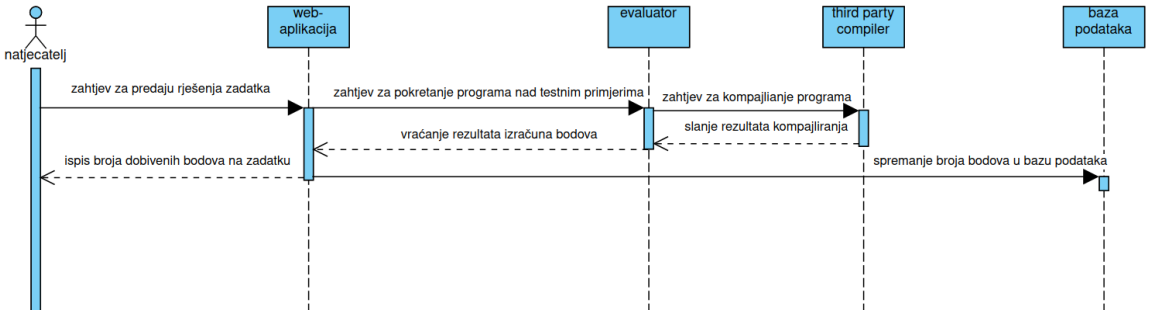
Voditelj natjecanja počinje proces kreiranja novog zadatka slanjem zahtjeva poslužitelju koji provjerava je li korisnik voditelj natjecanja. Ako je korisnik voditelj natjecanja poslužitelj omogućuje pristup formi za unos zadatka. Voditelj unosi detalje zadatka i testne primjere te šalje podatke. Poslužitelj prosljeđuje spremanje zadatka u bazu podataka te šalje potvrdu voditelju o uspješnom kreiranju zadatka. Ako korisnik nije voditelj natjecanja poslužitelj šalje poruku korisniku da samo voditelj natjecanja može kreirati zadatke.



Slika 3.4: Sekvencijski dijagram kreiranja novog zadatka

UC11 Predaja rješenja

Natjecatelj započinje proces predaje rješenja zadatka slanjem zahtjeva poslužitelju koji proslijedi zahtjev evaluatoru za pokretanje programa nad testnim primjerima. Evaluator proslijedi zahtjev third party compileru. Compiler pokreće program nad testnim primjerima i šalje rezultate evaluatoru. Evaluator izračunava broj dobivenih bodova i šalje ih Poslužitelju koji ažurira natjecatelja s brojem dobivenih bodova te šalje rezultate i broj bodova bazi podataka za spremanje.



Slika 3.5: Sekvencijski dijagram predaje rješenja zadatka

3.2 Ostali zahtjevi

- Cijeli sustav namijenjen je upotrebi na engleskom jeziku
- Sustav mora biti u mogućnosti podržati veći broj korisnika zbog svoje namjene
- Upotreba gotove aplikacije zamišljena je preko web preglednika
- Aplikacija mora implementirati autentikaciju i autorizaciju svakog korisnika
- Osjetljivi podaci od korisnika se moraju pretvarati u hash prije spremanja u bazu
- Greška u jednom dijelu sustava ne smije onemogućiti korištenje drugih usluga
- Odgovori sustava prema korisniku moraju biti brzi i ne dulji od nekoliko sekundi

4. Arhitektura i dizajn sustava

Arhitektura cijelog sustava može se podijeliti na četiri ključna dijela:

- Web preglednik
- Web aplikacija
- Baza podataka
- Code runner

Web preglednik omogućuje interakciju između korisnika i aplikacije. Svaki se korisnikov zahtjev događa na web pregledniku i prosljeđuje aplikaciji na obradu. Osim slanja zahtjeva, korisniku se omogućuje bolji pregled aplikacije pomoću apstraktnog koda kojeg web preglednik pretvara u lako shvatljive strukture.

Aplikacija se sastoji od tri servisa, frontenda, API servisa i evaluatora. Tehnologije koje se koriste na backendu uključuju programski jezik **Python** i **FastApi** web framework koji omogućuje stvaranje aplikacije temeljene na **REST** arhitekturi i rutera na kojem su definirane rute za slanje zahtjeva i komunikaciju korisnika i aplikacije. Frontend aplikacije izrađen je korištenjem libraryja **React**, a kod je pisan u **TypeScriptu**. On omogućuje korisniku jednostavan pregled aplikacije i jednostavno korištenje aplikacije, odnosno slanje zahtjeva.

Kontejneri se koriste unutar aplikacije kako bi poboljšali korisničko iskustvo, optimizirali performanse sustava te osigurali skalabilnost za povećani broj korisnika. Glavni se dio aplikacije s ruterima i logikom za komunikaciju s bazom nalazi u vlastitom kontejneru kao i glavna baza, evaluator i blob storage.

Za primarnu bazu podataka koristi se **PostgreSQL**. Primarna baza se nalazi u vlastitom kontejneru kojemu backend aplikacije pristupa kako bi spremio informacije u nju. Na backendu se koristi **psycopg** PostgreSQL adapter koji omogućuje komunikaciju između baze i backenda.

Evaluator komunicira s vanjskim Code runnerom za kompajliranje kodova koje korisnik šalje na backend. Ovaj ključni dio sustava odgovoran je za izvršavanje kôda unutar sigurnog okruženja, čime se osigurava izolacija korisničkih skripti od ostatka aplikacije. Kroz ovu komunikaciju, Evaluator prima kôd od korisnika, proslijeđuje ga Code runneru na izvršavanje te zatim interpretira rezultate kako bi ih pravilno integrirao natrag u aplikaciju.

Dodatno, evaluator koristi **blob storage** (Azure blob storage) pomoću kojeg dohvaća podatke koji su mu potrebni za evaluaciju određenog zadatka. Evaluator ima pristup informacijama poput ulaznih podataka za testiranje, očekivanih rezultata te drugih specifičnih parametara vezanih uz pojedini zadatak.

4.1 Baza podataka

Za implementaciju našeg sustava koristit ćemo PostgreSQL, relacijsku bazu podataka. Ovaj tip baze podataka posebno je pogodan za modeliranje kompleksnih struktura koje odražavaju realni svijet, zahvaljujući svojoj sposobnosti da efikasno upravlja relacijama, odnosno tablicama. Svaka tablica u bazi unikatno je definirana svojim nazivom i skupom atributa, čime se omogućava precizno strukturiranje podataka. Jedna od ključnih prednosti PostgreSQL baze je njena brzina i efikasnost u obradi podataka, uključujući pohranu, ažuriranje i dohvaćanje informacija. Naša baza podataka obuhvatit će niz entiteta koji su specifično prilagođeni zahtjevima naše aplikacije.

Popis entiteta: users, competitions, problems, competition_participations, problems_on_competitions, problem_results, trophies i verification_tokens.

4.1.1 Opis tablica

Users

Entitet **users** sadržava sve važne informacije o korisniku aplikacije. Sadrži atribute: *id, role, username, image_url, password_hash, email, name, surname, is_verified, created_on*. Ovaj entitet je u *many-to-many* vezi s entitetom *competitions* preko tablice *competition_participations*. Ovaj entitet je u *one-to-many* vezi s entitetom *verification_tokens* preko atributa *email*.

users		
id	UUID	jedinstveni brojčani identifikator
role	ENUM	uloga korisnika (admin, voditelj natjecanja ili natjecatelj)
username	VARCHAR(32)	korisničko ime
image	BYTEA	slika
password_hash	VARCHAR(255)	hash šifre korisnika
email	VARCHAR(128)	adresa e-pošte korisnika
name	VARCHAR(64)	ime korisnika
surname	VARCHAR(64)	prezime korisnika
is_verified	BOOLEAN	bool vrijednost koja govori je li korisnik verificirao adresu e-pošte
created_on	TIMESTAMP	timestamp trenutka kreiranja korisnika

Slika 4.1: Entitet **users**

Problems

Entitet **problems** sadržava sve važne informacije o zadacima. Sadržava atribute: *id, name, description, example_input, example_output, is_hidden, num_of_points, runtime_limit, tests_dir, is_private, created_on*. Ovaj entitet je u *many-to-many* vezi s entitetom *competitions* preko tablice *problems_on_competitions*. Ovaj entitet je u *one-to-many* vezi s entitetom *problem_results* preko atributa *id*.

problems		
id	UUID	jedinstveni brojčani identifikator
name	TEXT	naziv zadatka
example_input	TEXT	primjer unosa
example_output	TEXT	primjer izlaza
is_hidden	BOOLEAN	bool vrijednost koja govori je li skriven zadatak ili ne
num_of_points	REAL	broj bodova na zadatku
runtime_limit	TIME	ograničenje vremena izvođenja rješenja zadatka
description	TEXT	opis zadatka
tests_dir	TEXT	direktorij s testovima za ispravljanje rješenja
is_private	BOOLEAN	bool vrijednos koja govori je li privatan zadatak
created_on	TIMESTAMP	timestamp trenutka kreiranja zadatka

Slika 4.2: Entitet **problems**

Competitions

Entitet **competitions** sadržava sve važne informacije o natjecanjima. Sadržava atribute: *id, name, description, start_time, end_time, parent_id, problems*. Natjecanja mogu imati svoja nadređena natjecanja. Ovaj entitet je u *many-to-many* vezi s entitetom *users* preko *competition_participations* tablice. Ovaj entitet je u *many-to-many* vezi s entitetom *problems* preko *problems_on_competitions* tablice.

competitions		
id	UUID	jedinstveni brojčani identifikator
name	TEXT	naziv natjecanja
description	TEXT	opis natjecanja
start_time	TIMESTAMP	vrijeme početka natjecanja
end_time	TIMESTAMP	vrijeme završetka natejcanja
parent_id	UUID	identifikator natjecanja-roditelja
problems	UUID[ ]	svi zadaci u natjecanju

Slika 4.3: Entitet **competitions**

Competition_participations

Entitet **competition_participations** sadržava sve važne informacije o sudjelovanju korisnika na natjecanjima. Sadržava atribute: *id, user_id, competition_id, num_of_points*. Ovaj entitet je u *many-to-many* vezi s entitetom *users*. Ovaj entitet je u *many-to-many* vezi s entitetom *competitions* preko atributa *competition_id*.

competition_participations		
id	UUID	jedinstveni brojčani identifikator
user_id	UUID	jedinstveni brojčani identifikator korisnika, users.id
competition_id	UUID	jedinstveni brojčani identifikator natjecanja, competitions.id
num_of_points	REAL	broj bodova korisnika na natjecanju

Slika 4.4: Entitet **competition_participations**

Problems_on_competitions

Entitet **problems_on_competitions** sadržava informacije o zadacima koji pripadaju pojedinim natjecanjima. Sadržava atribute: *id, problem_id, competition_id, num_of_points*.

problems_on_competitions		
id	UUID	jedinstveni brojčani identifikator
problem_id	UUID	jedinstveni brojčani identifikator zadatka, problems.id
competition_id	UUID	jedinstveni brojčani identifikator natjecanja, competitions.id
num_of_points	REAL	broj bodova na zadatku u natjecanju

Slika 4.5: Entitet **problems_on_competitions**

Problem_results

Entitet **problem_results** sadrži rezultate pojedinih korisnika na pojedinim zadacima. Sadržava atribute: *id*, *user_id*, *problem_id*, *competition_id*, *num_of_points*, *average_runtime*, *is_correct*, *source_code*. Ovaj entitet je u *one-to-many* vezi s entitetom *users* preko *user_id* te *one-to-many* vezi s entitetom *problems* preko atributa *problem_id*. Ovaj entitet je u *one-to-many* vezi s entitetom *competitions* preko atributa *competition_id*.

problem_results		
id	UUID	jedinstveni brojčani identifikator
problem_id	UUID	jedinstveni brojčani identifikator zadatka, problems.id
competition_id	UUID	jedinstveni brojčani identifikator natjecanja, competitions.id
user_id	UUID	jedinstveni brojčani identifikator korisnika, users.id
average_runtime	TIME	prosječno vrijeme izvođenja programa
is_correct	BOOLEAN	bool vrijednost točnosti zadatka
num_of_points	REAL	korisnikov broj bodova na zadatku
source_code	TEXT	izvorni kod korisnika koji je predan kao rješenje

Slika 4.6: Entitet **problem_results**

Trophies

Entitet **trophies** sadržava informacije o trofejima koje korisnici mogu osvojiti. Sadržava atribute: *id*, *competition_id*, *user_id*, *position*, *icon*. Ovaj entitet je u *one-to-many* vezi s entitetom *users* preko atributa *user_id* te *one-to-many* vezi s entitetom *competitions* preko atributa *competition_id*.

trophies		
id	UUID	jedinstveni brojčani identifikator
competition_id	UUID	jedinstveni brojčani identifikator natjecanja, competitions.id
user_id	UUID	jedinstveni brojčani identifikator korisnika, users.id
position	SMALLINT	pozicija korisnika na ljestvici
icon	TEXT	ikona pehara

Slika 4.7: Entitet **trophies**

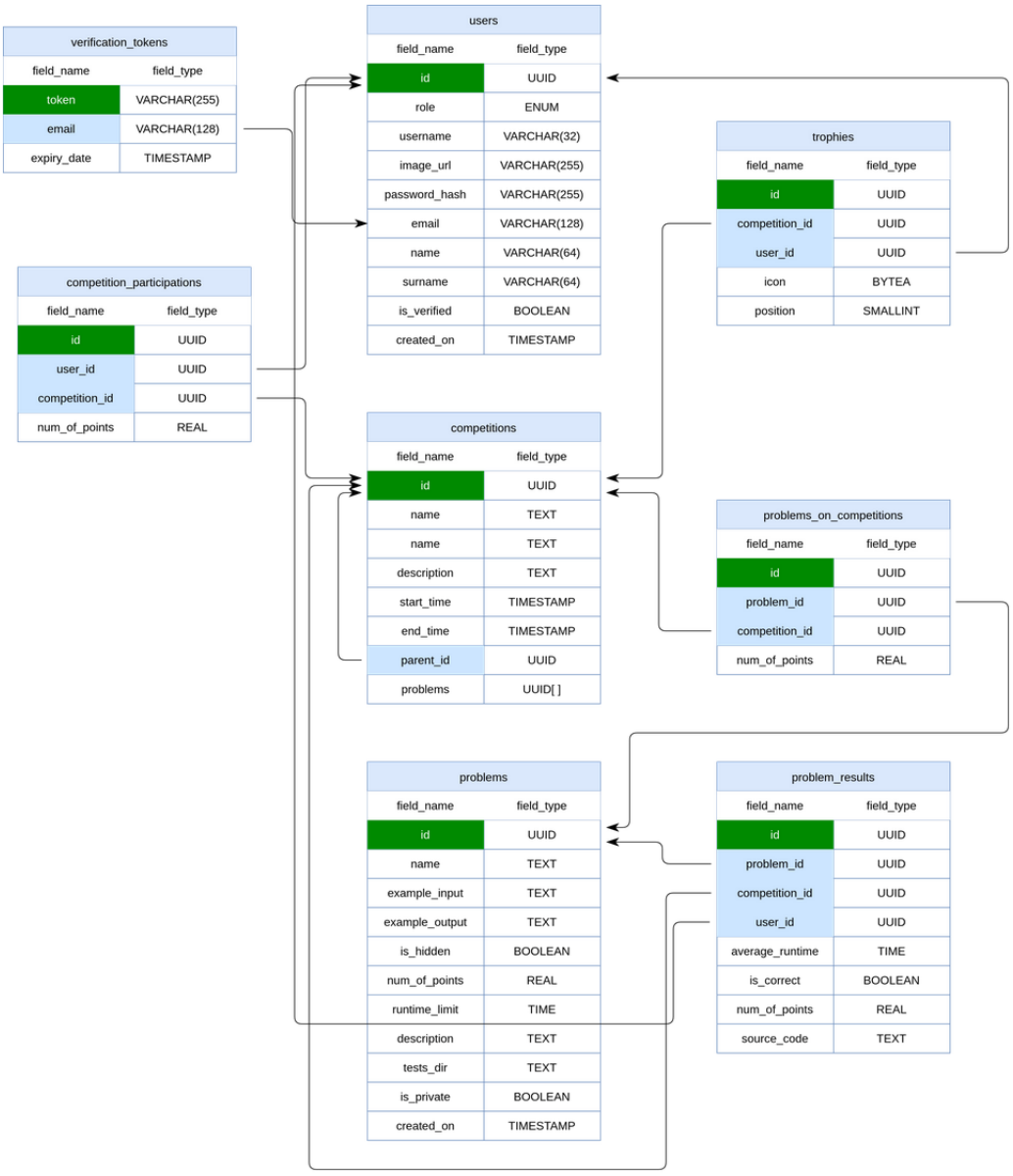
Verification_tokens

Entitet **verification_tokens** sadržava informacije o tokenima za verifikaciju korisnika. Sadržava atribute: *token*, *email*, *expiry_date*. Ovaj entitet je u *one-to-many* vezi s entitetom *users* preko atributa *email*.

verification_tokens		
token	VARCHAR(255)	verifikacijski token
email	VARCHAR(128)	adresa e-pošte korisnika
expiry_date	TIMESTAMP	datum isteka verifikacijskog tokena

Slika 4.8: Entitet **verification_tokens**

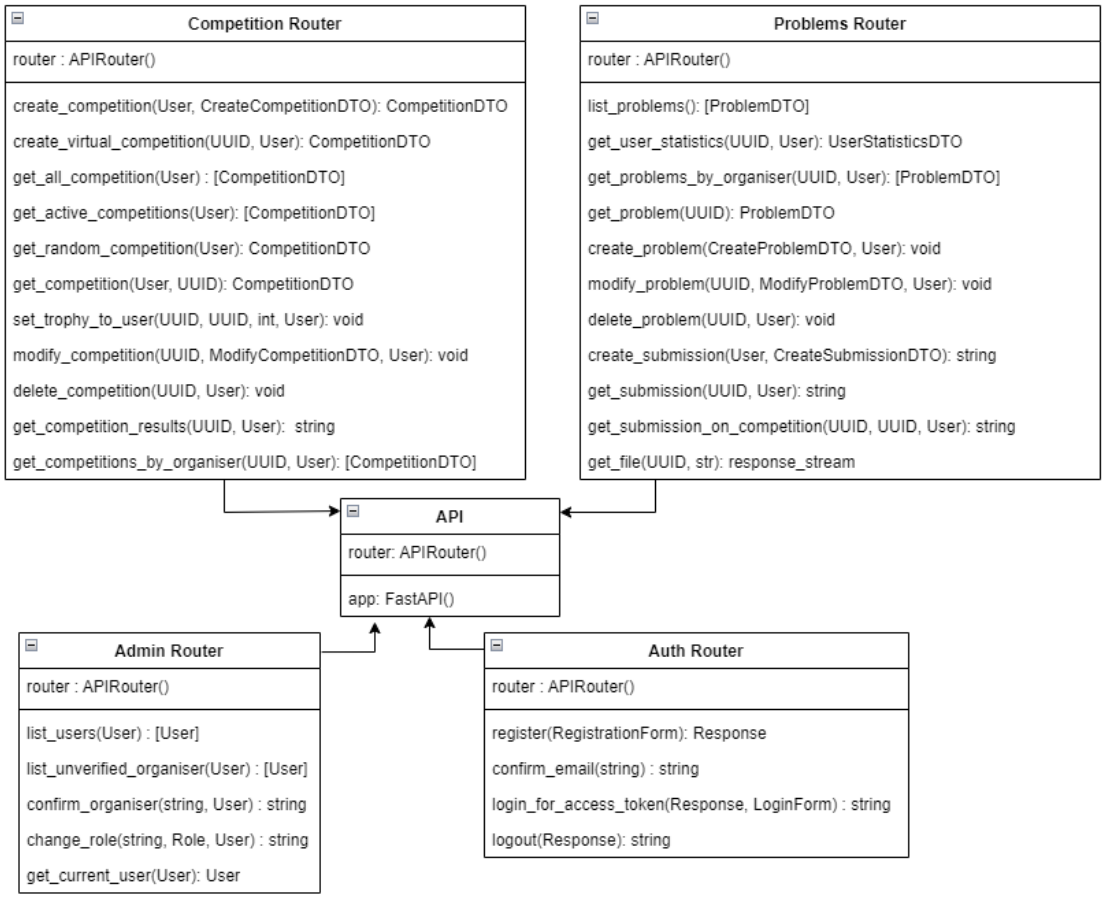
4.1.2 Dijagram baze podataka



Slika 4.9: Dijagram baze podataka

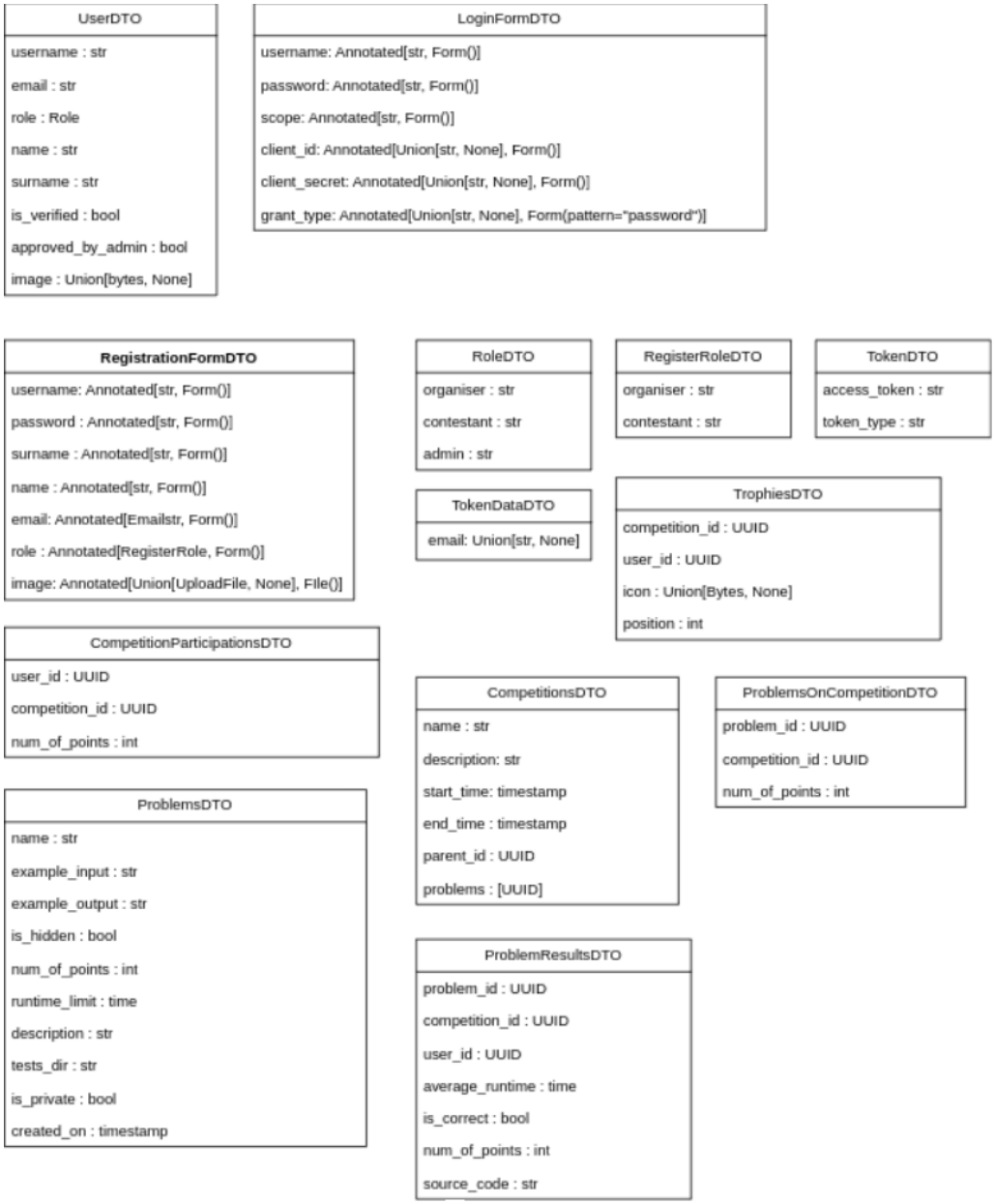
4.2 Dijagram razreda

Na sljedećim slikama, 4.10, 4.11, 4.12 i 4.13 prikazani su dijagrami razreda koji pripadaju backend dijelu aplikacije. Na slici 4.10 prikazani su ruteri logički podijeljeni metodama određenih aktora. Tako imamo "Users Router", "Admin Router" i "Auth Router" koje sve sadrži API (glavni dio backenda aplikacije). Metode implementirane u tim klasama manipuliraju podacima iz baze i podacima o trenutnom korisniku (web token i ostali podaci u kolačiću).

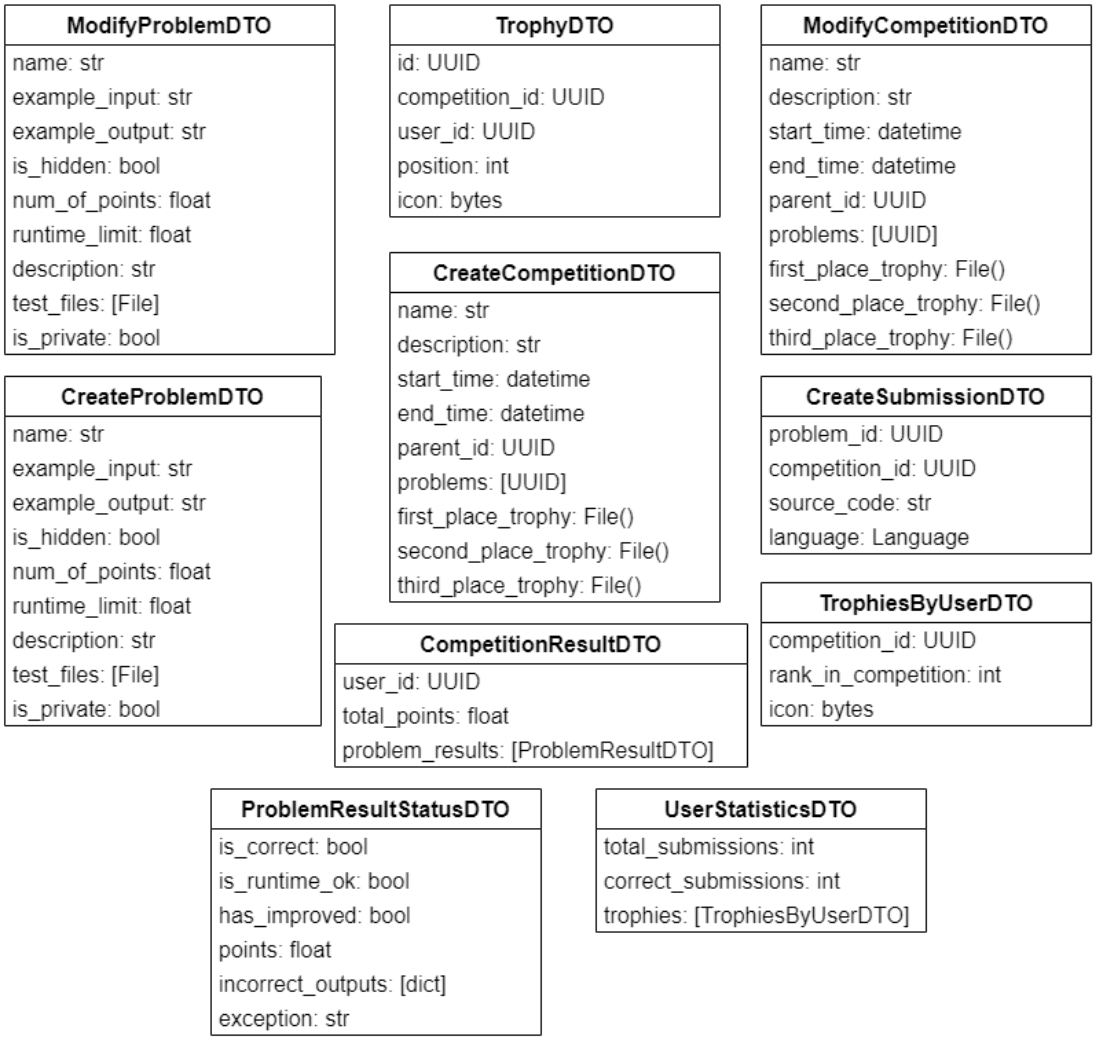


Slika 4.10: Dijagram razreda - Dio Controllers

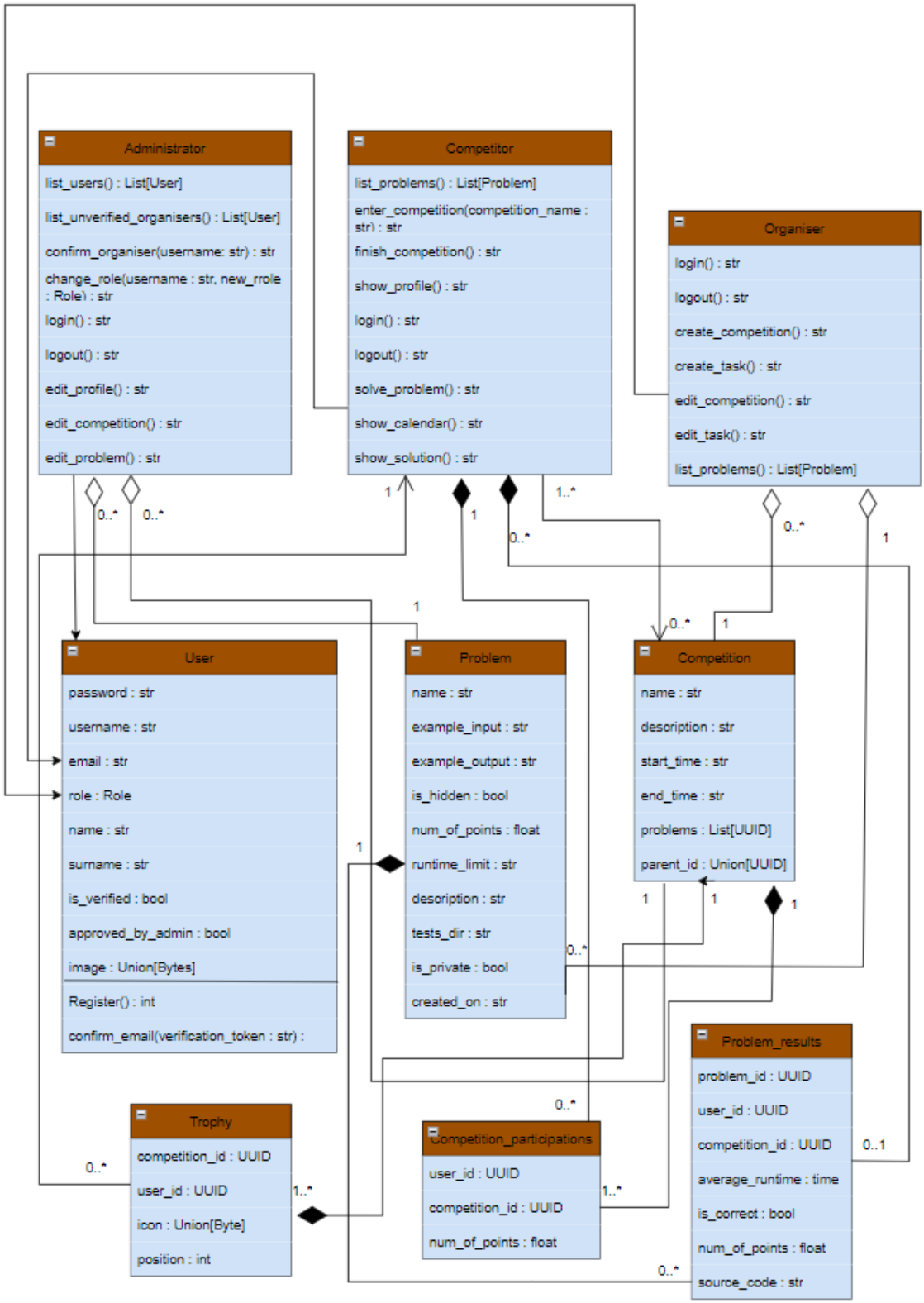
Modeli razreda, odnosno DTO (Data transfer object) služe za komunikaciju i prijenos podataka i baze u aplikaciju i obratno. Kada se korisnik bez korisničkog računa registrira, aplikacija koristi **RegistrationFormDTO** dok se pri prijavi korisnika s postojećim računom koristi **LoginFormDTO**. Kada se kreira korisnički račun, aplikacija za njegov prikaz i spremanje u bazu koristi **UserDTO**. Pri izradi natjecanja se koristi **ProblemsDTO**, a za kreiranje natjecanja **CreateCompetitionDTO**. Kada korisnik ide predati svoje rješenje zadatka na natjecanju, koristi se **CreateSubmissionDTO**. Za prikaz rezultata natjecanja koristimo **CompetitionResultDTO**, za izmjenu postojećih natjecanja, odnosno problema koriste se **ModifyCompetitionDTO** i **ModifyProblemDTO**. Trofeji se prikazuju pomoću **TrophyDTO**, korisnikova statistika sa **UserStatisticsDTO**.



Slika 4.11: Dijagram razreda - DTO dio 1



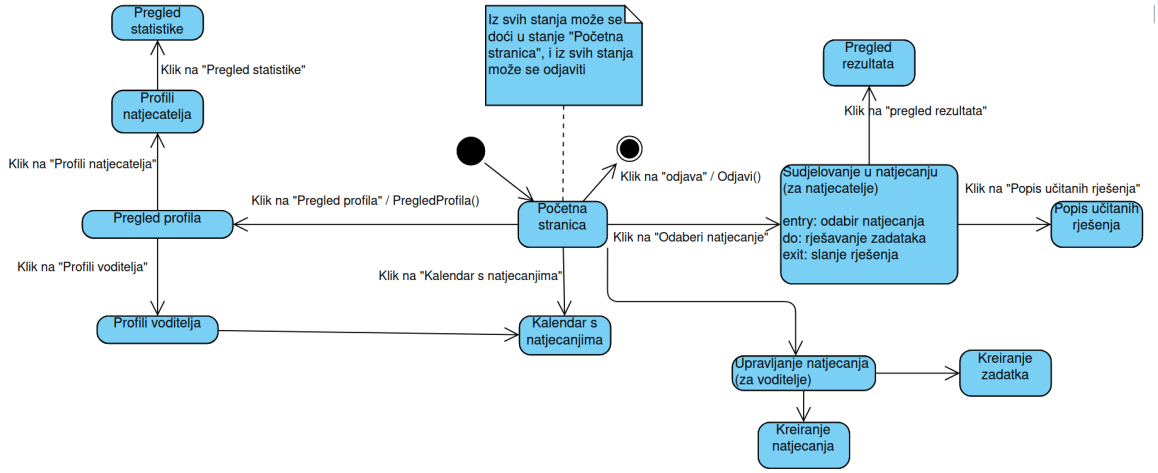
Slika 4.12: Dijagram razreda - DTO dio 2



Slika 4.13: Dijagram razreda - implementacijski dijagram

4.3 Dijagram stanja

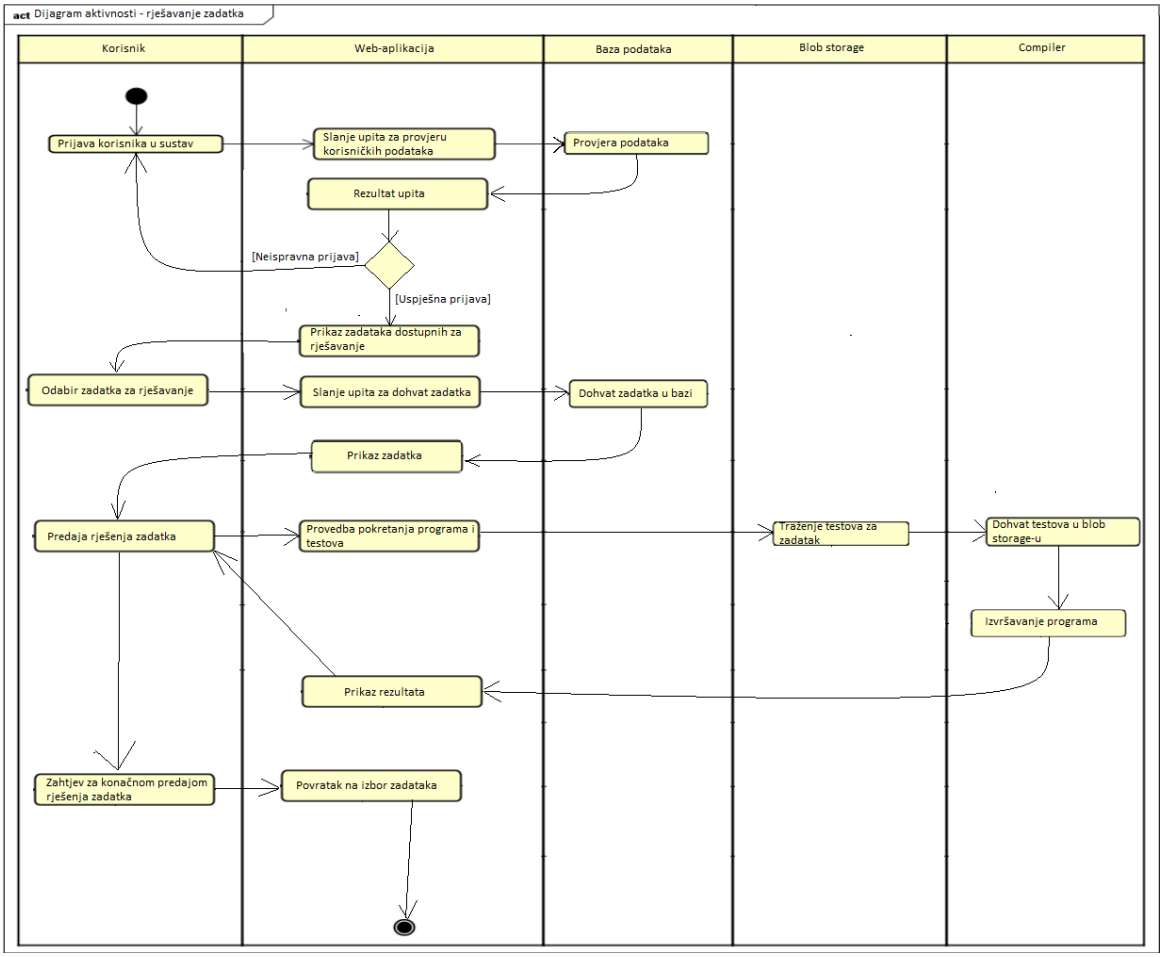
Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici je prikazan dijagram stanja registriranog korisnika. Nakon prijave, klijentu se prikazuje početna stranica na kojoj ima opciju pregleda profila natjecatelja i voditelja te opciju pregleda kalendara s natjecanjima. Nakon klika na profile natjecatelja, korisnik može pregledat njihovu statistiku sa svih natjecanja. Natjecatelj može sudjelovati u natjecanju i rješavati zadatke te predati rješenje zadatka. Nakon predaje svog rješenja zadataka, ima opciju pregleda popisa svih učitanih rješenja za pojedine zadatke. Voditelji imaju opciju upravljanja natjecanja u kojemu mogu stvarati nova natjecanja ili kreirati nove zadatke.



Slika 4.14: Dijagram stanja

4.4 Dijagram aktivnosti

Dijagram aktivnosti prikazuje proces rješavanja jednog zadatka od prijave u sustav pa sve do predaje rješenja. Korisnik otvara web aplikaciju i unosi svoje pristupne podatke. Web aplikacija šalje upit bazi podataka radi provjere pristupnih podataka korisnika. Ako su pristupni podaci ispravni, korisnik je uspješno prijavljen, inače se vraća na stranicu za prijavu. Web aplikacija prikazuje dostupne zadatke korisniku. Korisnik pregledava dostupne zadatke i odabire zadatak koji želi riješiti. Web aplikacija šalje upit bazi podataka za dohvat podataka odabranog zadatka. Podaci zadatka se prikazuju korisniku. Korisnik rješava zadani zadatak i predaje svoje rješenje putem web aplikacije. Web aplikacija šalje rješenje korisnika compileru. Aplikacija preuzima testne podatke iz blob storagea, šalje ih compileru koji izvršava program s korisnikovim rješenjem. Rezultati izvršavanja programa vraćaju se web aplikaciji. Rezultati izvršavanja programa prikazuju se korisniku, uključujući informacije o ispravnosti rješenja i ostale relevantne informacije. Ako je korisnik zadovoljan rezultatima, može ponovno predati drugo rješenje ili poslati zahtjev za konačnom predajom zadatka. Ako korisnik želi konačno predati rješenje, proces završava.



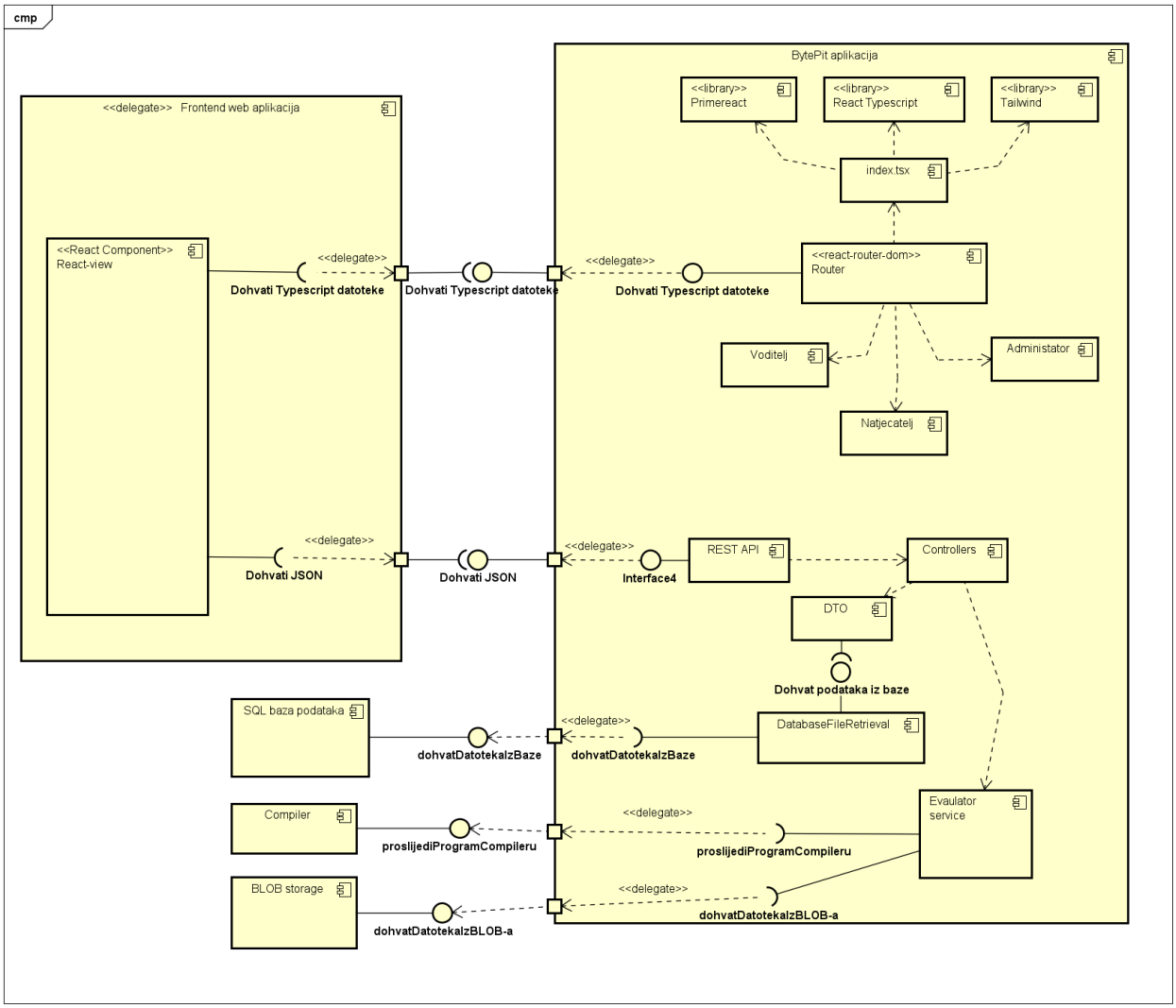
Slika 4.15: Dijagram aktivnosti

4.5 Dijagram komponenti

Slika 4.10 prikazuje dijagram komponenti koji ilustrira kako su komponente organizirane međusobno, unutar sustava te kako su povezane s okolinom. Sustavu se pristupa kroz dva sučelja.

Prvo sučelje ima funkciju prikupljanja Typescript datoteka koje pripadaju frontend dijelu aplikacije. Ključnu ulogu ima Router, određujući koje će datoteke biti poslužene na sučelju. Frontend se sastoji od niza cjelina organiziranih prema tipovima korisnika, a Typescript datoteke koriste React, Tailwind i Primereact biblioteke za integraciju već pripremljenih komponenata.

Drugo sučelje ima ulogu u preuzimanju JSON podataka i pristupanju REST API komponentama koje poslužuju podatke iz backend dijela aplikacije. DatabaseFileRetrieval zadatak ima dohvaćanje datoteka iz baze podataka korištenjem SQL upita, a zatim ti podaci putuju prema DTO-u. Reactview komponenta komunicira s ostatkom aplikacije putem dostupnih sučelja. Sustav uključuje Evaluator komponentu koja komunicira s BLOB storage-om i odgovorna je za evaluaciju rješenja zadataka. Evaulator koristi Compiler kako bi doznao ispis korisnikova koda, te ga uspoređuje sa ispravnim koji je na BLOB storage-u. Ovaj dio sustava doprinosi skalabilnosti, efikasnosti i fleksibilnosti u upravljanju podacima.



Slika 4.16: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu ostvarena je kombinacijom **Atlassin Jira**, **Atlassin Confluence** i **WhatsApp**. Za upravljanje kodom korišten je sustav **Git** i web platforma **GitHub** za pregled udaljenog repozitorija i bolje snalaženje u projektu. Kao razvojno okruženje korišteni su **Microsoft Visual Studio Code** i **JetBrains PyCharm**. Microsoft Visual Studio Code pruža integrirano razvojno okruženje (IDE) koje podržava različite programske jezike, uključujući C++, Python i mnoge druge. Ovo okruženje dolazi s alatima za uređivanje koda, debugiranje i upravljanje datotekama i kontejnerima. S druge strane, **JetBrains PyCharm** je posebno usmjeren na podršku Python razvoju. Ovaj IDE pruža bogat skup značajki, uključujući pametno završavanje koda, analizu koda, integrirano upravljanje verzijama, i podršku za virtualno okruženje. Korištenjem ova dva razvojna okruženja, tim je imao pristup snažnim alatima za učinkovit i produktivan razvoj softvera. Aplikacija je napisana koristeći radni okvir **FastApi** koji se koristi za izradu REST API-ja u pythonu. Za izradu *frontenda* koristili smo **React** i **TypeScript**. Prednost **TypeScripta** nad običnim **Javascriptom** je u njegovoj mogućnosti definiranja tipa varijabli i posljedično tome, lakšem rukovanju greškama i boljom kontrolom varijabli. Za izradu kontejnera je korišten **Docker**, a sve je stavljeno na poslužitelja u oblaku na **Microsoft Azure**.

5.2 Ispitivanje programskog rješenja

dio 2. revizije

U ovom poglavlju je potrebno opisati provedbu ispitivanja implementiranih funkcionalnosti na razini komponenti i na razini cijelog sustava s prikazom odabranih ispitnih slučajeva. Studenti trebaju ispitati temeljnu funkcionalnost i rubne uvjete.

5.2.1 Ispitivanje komponenti

Potrebno je provesti ispitivanje jedinica (engl. unit testing) nad razredima koji implementiraju temeljne funkcionalnosti. Razraditi ***minimalno 6 ispitnih slučajeva*** *u kojima će se ispitati redovni slučajevi, rubni uvjeti te izazivanje pogreške (engl. exception throwing). Poželjno je stvoriti i ispitni slučaj koji koristi funkcionalnosti koje nisu implementirane. Potrebno je priložiti izvorni kôd svih ispitnih slučajeva te prikaz rezultata izvođenja ispita u razvojnom okruženju (prolaz/pad ispita).*

5.2.2 Ispitivanje sustava

Ispitni slučaj 1: Prijava u sustav

Ulaz:

1. Otvaranje početne stranice u web pregledniku.
2. Pritisak na gumb **Login** u gornjem desnom kutu.
3. Ispunjavanje forme za prijavu.
4. Pritisak na gumb **Submit**

Očekivani rezultat:

1. Otvaranje početne stranice korisnika.

Rezultat: Očekivani rezultat [1.] nije zadovoljen jer je korisnik unio pogrešnu zaporku. **Aplikacija nije prošla test.**

Potrebno je provesti i opisati ispitivanje sustava koristeći radni okvir Selenium[1]*. Razraditi* ***minimalno 4 ispitna slučaja*** *u kojima će se ispitati redovni slučajevi, rubni uvjeti te poziv funkcionalnosti koja nije implementirana/izaziva pogrešku kako bi se vidjelo na koji način sustav reagira kada nešto nije u potpunosti ostvareno. Ispitni slučaj*

[1] `https://www.seleniumhq.org/`

se treba sastojati od ulaza (npr. korisničko ime i lozinka), očekivanog izlaza ili rezultata, koraka ispitivanja i dobivenog izlaza ili rezultata.

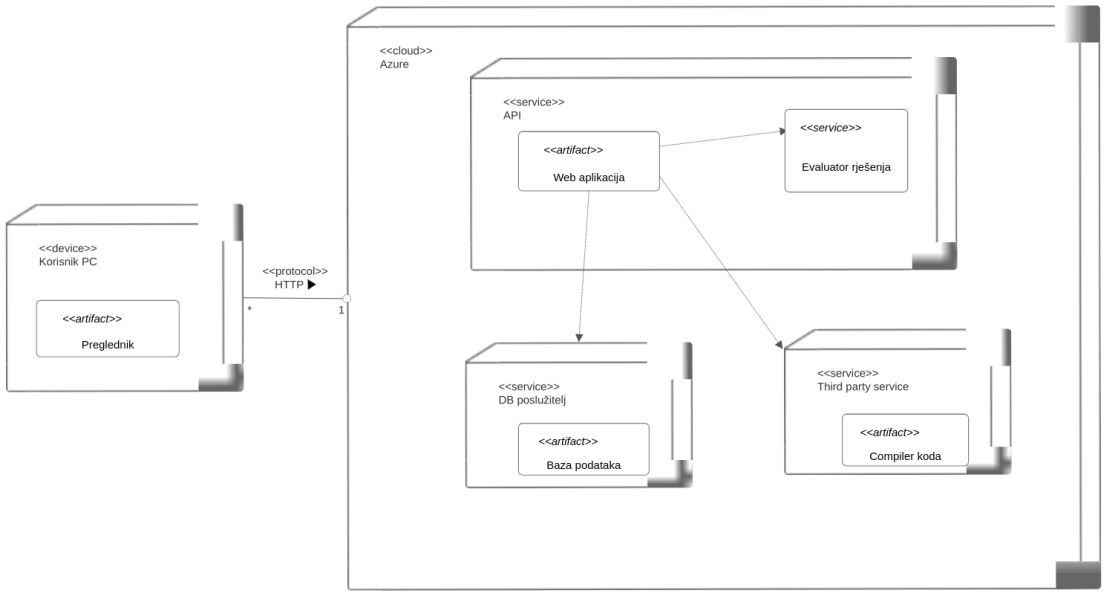
Izradu ispitnih slučajeva pomoću radnog okvira Selenium moguće je provesti pomoću jednog od sljedeća dva alata:

- *dodatak za preglednik **Selenium IDE** - snimanje korisnikovih akcija radi automatskog ponavljanja ispita*
- ***Selenium WebDriver** - podrška za pisanje ispita u jezicima Java, C#, PHP koristeći posebno programsko sučelje.*

Detalji o korištenju alata Selenium bit će prikazani na posebnom predavanju tijekom semestra.

5.3 Dijagram razmještaja

Dijagrami razmještaja opisuju topologiju sklopovlja i programsku potporu koja se koristi u implementaciji sustava u njegovom radnom okruženju. Na azure app service-u nalaze se API sa evaluatorom rješenja, poslužitelj baze podataka (azure sql database) te third party servis za kompajliranje koda. Klijenti koriste web preglednik za pristup web aplikaciji. Sustav je baziran na REST API arhitekturi, a komunikacija između klijenta i poslužitelja odvija se preko HTTP veze.

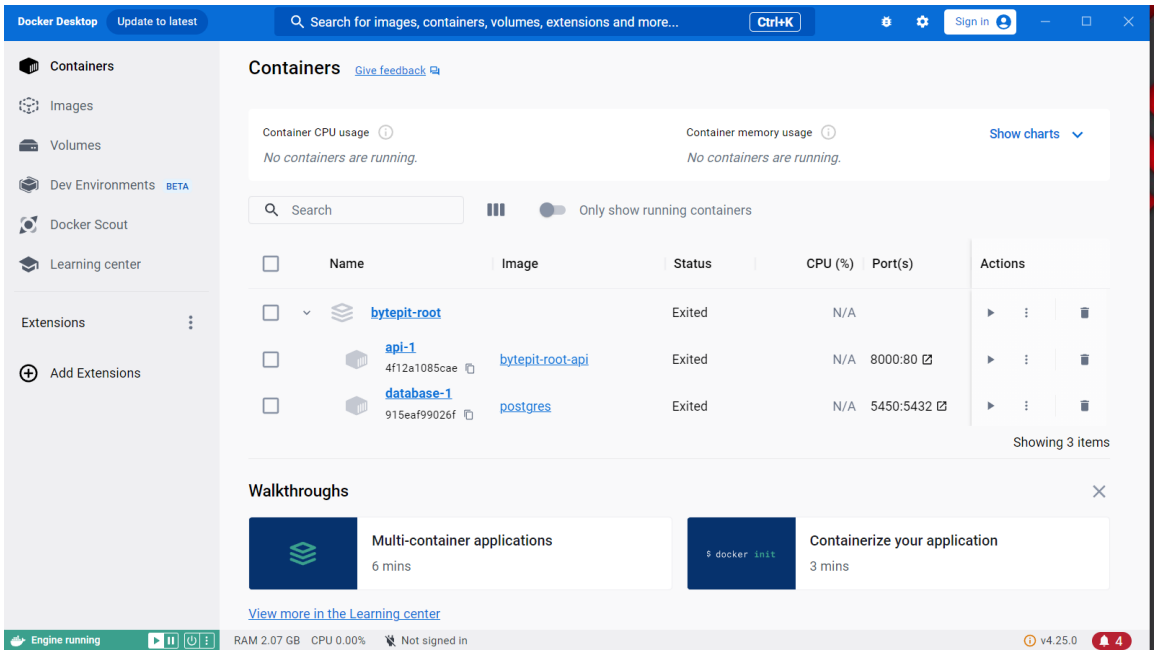


Slika 5.1: Dijagram razmještaja

5.4 Upute za puštanje u pogon

Pokretanje backend-a i baze podataka

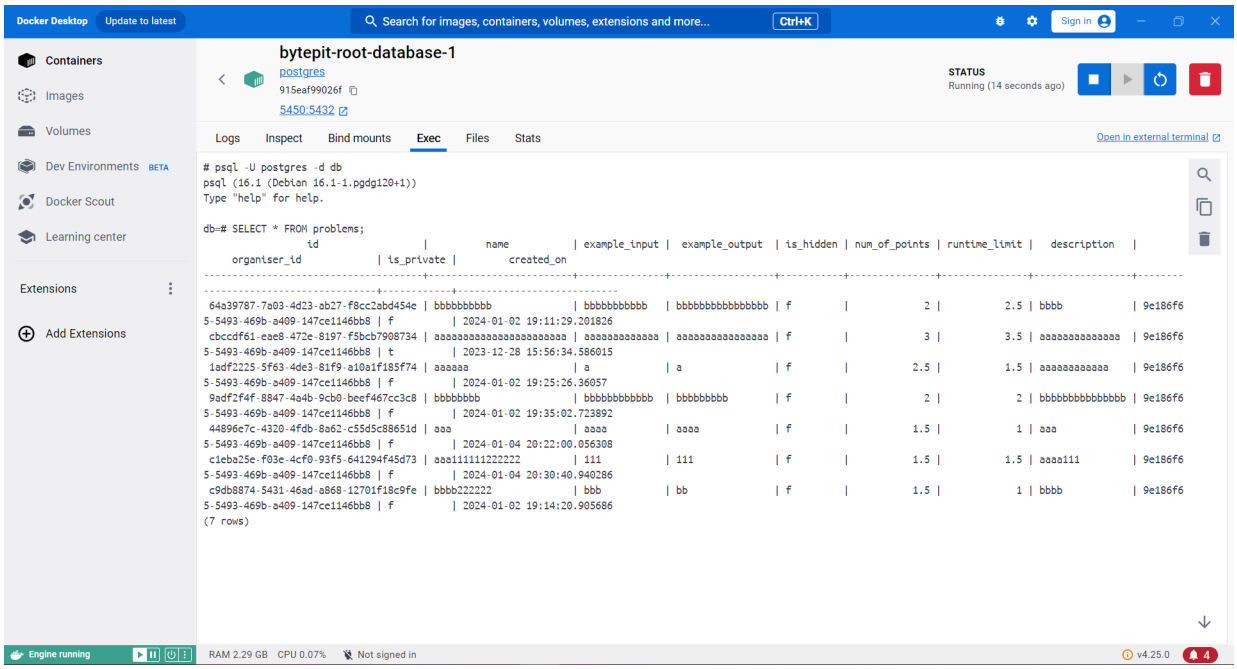
Potrebno je preuzeti i instalirati aplikaciju Docker Desktop uz pomoć koje se pokreću container-i za backend i bazu podataka. Nakon preuzimanja aplikacije moramo se u terminalu navigirati do direktorija *bytepit-root* unutar kojeg se nalazi docker-compose datoteka koja sadrži informacije za pokretanje. Pri prvom pokretanju potrebno je nekoliko minuta za podizanje.
Unutar terminala koristimo naredbu: **docker-compose up –build**



Slika 5.2: Aplikacija Docker Desktop sa kreiranim containerima

Pri prvom pokretanju unutar baze neće biti kreirane potrebne tablice, ali ih možemo kreirati uz pomoć datoteke **ddl.sql** koja se nalazi unutar direktorija *bytepit-root*. Unutar aplikacije Docker Desktop možemo kliknuti na opcije od baze (tri uspravne točkice ispod zaglavlja Actions), te odabrati opciju **Open in terminal** uz pomoć koje možemo upisati naredbe za kreaciju tablica unutar baze. Unutar terminala odabiremo opciju **Exec** te u njoj napišemo naredbu: **psql -U postgres -d db**

Tada možemo pisati naredbe za našu bazu u SQL-u unutar terminala. Sljedeća naredba koju unosimo je cijela **ddl.sql** datoteka uz pomoć koje se kreiraju tablice. Nakon toga su backend i baza podataka uspješno postavljeni i spremni za korištenje. Kao provjeru za ispravno kreiranje tablica možemo napisati query poput ovoga: **SELECT * FROM problems;**, u slučaju ispravnog postavljanja ispisuje se tablica *problems*.

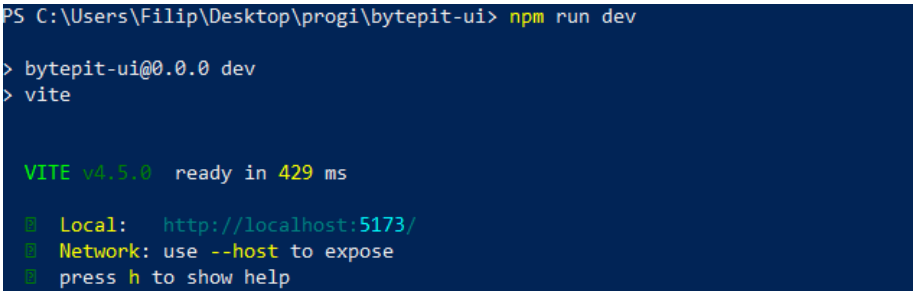


Slika 5.3: Ispravno ispunjena baza i provjera

Za zaustavljanje Docker container-a koristimo naredbu: **docker-compose down**

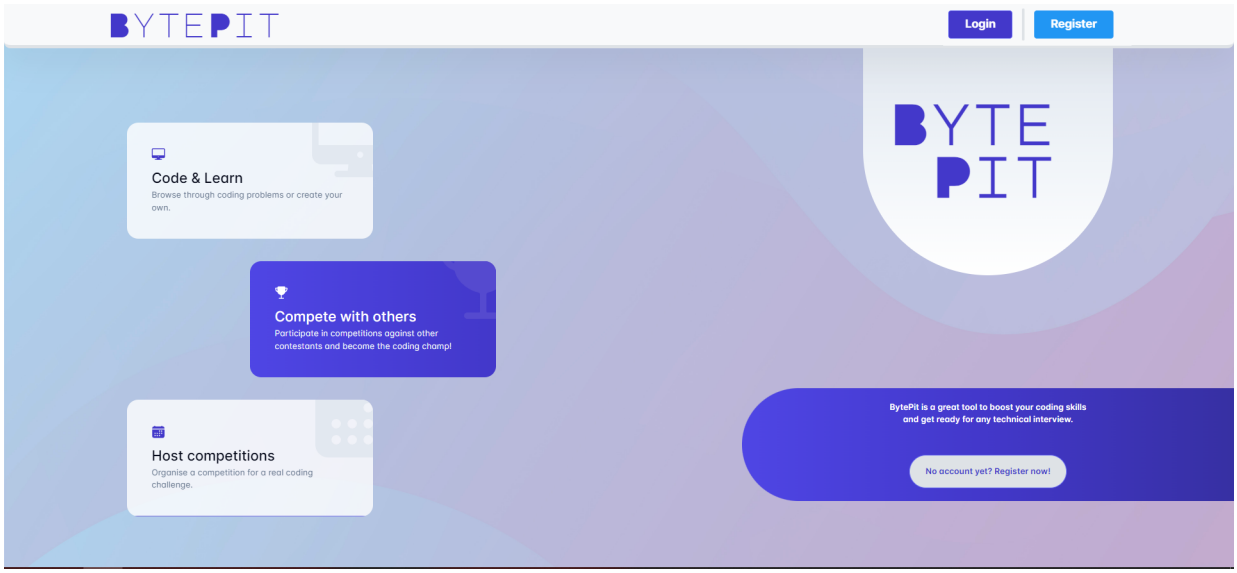
Pokretanje frontend-a

Unutar terminala se navigiramo do direktorija *bytepit-ui* te pokrećemo naredbu: **npm install**
Pomoću te naredbe se pokreće instalacija Vite-a i ostalih dependency-a koji su potrebni za pokretanje frontend-a. Za instalaciju je potrebno nekoliko minuta.
Nakon instalacije pokrećemo naredbu: **npm run dev** koja pokreće frontend.



Slika 5.4: Terminal nakon uspješno izvedene funkcije za pokretanje

Tada možemo otvoriti web-preglednik i upisati **http://localhost:5173/** i početi koristiti aplikaciju.



Slika 5.5: Pokrenuta aplikacija

Za zaustavljanje koristimo naredbu: **npm run stop**

6. Zaključak i budući rad

Razumijevajući postavljene zahtjeve ovog projekta, pristupili smo zadatku s dubokim poštovanjem prema tradicionalnim metodama, što je uključivalo stvaranje opsežne dokumentacije i brojnih UML dijagrama. Međutim, kroz ovaj proces, naš tim je stekao dublje razumijevanje o ograničenjima i nedostacima ovakvog pristupa, posebno kada se uspoređuje s agilnijim i fleksibilnijim metodama koje su danas na snazi u industriji softverskog inženjerstva.

Naša iskustva s ovim projektom jasno su istaknula zašto je u posljednjih deset godina industrija napustila praksu pisanja opsežne dokumentacije. Iako smo pažljivo i precizno kreirali UML dijagrame i detaljnu dokumentaciju, često smo se suočavali s teškoćama koje su proizašle iz njihove inherentne rigidnosti. Ova vrsta dokumentacije stvara okvir koji može ograničiti sposobnost projekta da se brzo prilagodi promjenama, što je od ključne važnosti u brzom i dinamičnom tehnološkom okruženju.

UML dijagrami, iako korisni za vizualizaciju i planiranje, mogu biti prekomjerni i često ne odražavaju stvarnu kompleksnost ili fluidnost razvojnog procesa. Naša iskustva su pokazala da ovakva vrsta dokumentacije može biti kontraproduktivna, jer zahtijeva značajne količine vremena i resursa koji bi se mogli bolje iskoristiti u stvarnom razvoju proizvoda.

S druge strane, Agile pristup, koji smo kritički promatrali kroz prizmu našeg projekta, nudi alternativu koja se fokusira na adaptabilnost, iterativni razvoj i stalnu suradnju. Agile priznaje da se pravi uspjeh u razvoju softvera ne postiže kroz statične planove i dokumente, već kroz neprekidno prilagođavanje, učenje i suradnju. Agile metodologija potiče timove na brzu reakciju na promjene i na fokusiranje na stvaranje proizvoda koji odgovara trenutnim potrebama tržišta i korisnika.

Kroz naš projekt, usmjerili smo fokus prema stvaranju zapravo korisnih elemenata, koji su u suštini temelj modernog softverskog razvoja. Ovaj pristup je uključivao razvoj i implementaciju Continuous Integration/Continuous Deployment (CI/CD) pipelineova, korištenje alata kao što su Jira i Confluence za upravljanje projektima i dokumentacijom, te usvajanje automatizacije u različitim aspektima

našeg rada.

CI/CD pipelineovi su bili ključni za našu sposobnost brzog i efikasnog razvoja softvera. Kroz automatizaciju procesa builda, testiranja i deploymenta, uspjeli smo znatno ubrzati razvojni ciklus i povećati kvalitetu našeg koda. Ovo je omogućilo timu da se brzo prilagodi promjenama, smanjujući vrijeme potrebno za implementaciju novih funkcionalnosti i popravak bugova.

Korištenje Jire i Confluencea je dodatno pojačalo našu sposobnost efikasnog upravljanja projektom. Jira nam je omogućila da učinkovito pratimo napredak i upravljamo zadacima, dok je Confluence poslužio kao centralno mjesto za pohranu i organizaciju naše projektne dokumentacije. Ovaj integrirani pristup upravljanju projektima i dokumentacijom omogućio nam je da ostanemo organizirani, transparentni i usmjereni na ciljeve projekta.

Ovi elementi projekta, iako manje vidljivi u tradicionalnoj dokumentaciji, bili su vitalni za naš uspjeh i efikasnost. Dok smo se pridržavali zahtjeva za opsežnom dokumentacijom i UML dijagramima, naša stvarna vrijednost ležala je u primjeni ovih naprednih alata i tehnika. Naš rad demonstrira kako, unatoč poštivanju tradicionalnih zahtjeva, možemo uspješno integrirati i koristiti moderne alate i prakse koje su sada standard u industriji.

U retrospektivi, naše iskustvo s ovim projektom je bilo dragocjeno učenje o važnosti balansa između planiranja i fleksibilnosti. Dok smo uspješno ispunili zahtjeve za detaljnom dokumentacijom i UML dijagramima, također smo stekli uvid u njihova ograničenja i razloge zašto suvremena industrija softverskog inženjerstva preferira agilnije pristupe. Kritički osvrt na ovaj projekt ne umanjuje vrijednost rada koji smo obavili, već osvjetljava put ka efikasnijim i prilagodljivijim metodama koje su danas relevantnije u dinamičkom tehnološkom svijetu.

Popis literature

1. Programsko inženjerstvo, FER ZEMRIS, http://www.fer.hr/predmet/proinz
2. The Unified Modeling Language, https://www.uml-diagrams.org/
3. Astah Community, http://astah.net/editions/uml-new
4. Docker Desktop, https://www.docker.com/products/docker-desktop/
5. Primereact, https://primereact.org/
6. JSON Web tokens, https://auth0.com/docs/secure/tokens/json-web-tokens
7. CodeX-API, https://github.com/Jaagrav/CodeX-API/tree/master
8. FastAPI, https://fastapi.tiangolo.com/tutorial/first-steps/
9. SQLAlchemy, https://www.sqlalchemy.org/
10. Pydantic, https://docs.pydantic.dev/latest/concepts/models/
11. PostgreSQL, https://www.postgresql.org/
12. Vite, https://vitejs.dev/

Indeks slika i dijagrama

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

Kontinuirano osvježavanje

U ovom dijelu potrebno je redovito osvježavati dnevnik sastajanja prema predlošku.

1. sastanak
    - Datum: u ovom formatu: 18. siječnja 2024.
    - Prisustvovali: I.Prezime, I.Prezime
    - Teme sastanka:
        - opis prve teme
        - opis druge teme
2. sastanak
    - Datum: u ovom formatu: 18. siječnja 2024.
    - Prisustvovali: I.Prezime, I.Prezime
    - Teme sastanka:
        - opis prve teme
        - opis druge teme

Tablica aktivnosti

Kontinuirano osvježavanje

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

	Marko Bolt	**Filip Bernt**	**Teo Matošević**	**Teo Radolović**	**Jure Franjković**	**Fran Sipić**	**Jakov Vinožganić**
Upravljanje projektom	10		4				3
Opis projektnog zadatka	3						
Funkcionalni zahtjevi				3			4
Opis pojedinih obrazaca		2					
Dijagram obrazaca		5					
Sekvencijski dijagrami			1			4	
Opis ostalih zahtjeva					3		
Arhitektura i dizajn sustava					3		
Baza podataka		1				4	
Dijagram razreda					6		
Dijagram stanja						3	
Dijagram aktivnosti					3		
Dijagram komponenti		3					
Korištene tehnologije i alati					1		
Ispitivanje programskog rješenja							
Dijagram razmještaja						3	
Upute za puštanje u pogon		2					

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

	Marko Bolt	**Filip Bernt**	**Teo Matošević**	**Teo Radolović**	**Jure Franjković**	**Fran Sipić**	**Jakov Vinožganić**
Dnevnik sastajanja							
Zaključak i budući rad							
Popis literature		1					
Dodatne stavke kako ste podijelili izradu aplikacije							
front end	10	4	15	20			30
izrada baze podataka	2		3				4
spajanje s bazom podataka	2						2
back end	4				6	7	10
deploy	8	1	2	6			

Dijagrami pregleda promjena

dio 2. revizije

Prenijeti dijagram pregleda promjena nad datotekama projekta. Potrebno je na kraju projekta generirane grafove s gitlaba prenijeti u ovo poglavlje dokumentacije. Dijagrami za vlastiti projekt se mogu preuzeti s gitlab.com stranice, u izborniku Repository, pritiskom na stavku Contributors.