

Programsko inženjerstvo

Ak. god. 2022./2023.

Ples

Dokumentacija, Rev. 2

Grupa: *ProgramTvogKompjutera*

Voditelj: *Mateja Golec*

Datum predaje: 13. 1. 2023.

Nastavnik: *Hrvoje Nuić*

Sadržaj

1	Dnevnik promjena dokumentacije	3
2	Opis projektnog zadatka	6
2.1	Opis projektnog zadatka 'Ples'	6
3	Specifikacija programske potpore	12
3.1	Funkcionalni zahtjevi	12
3.1.1	Obrasci uporabe	14
3.1.2	Sekvencijski dijagrami	26
3.2	Ostali zahtjevi	31
4	Arhitektura i dizajn sustava	32
4.1	Baza podataka	35
4.1.1	Opis tablica	35
4.1.2	Dijagram baze podataka	41
4.2	Dijagram razreda	42
4.3	Dijagram stanja	46
4.4	Dijagram aktivnosti	47
4.5	Dijagram komponenti	47
5	Implementacija i korisničko sučelje	50
5.1	Korištene tehnologije i alati	50
5.2	Ispitivanje programskog rješenja	52
5.2.1	Ispitivanje komponenti	52
5.2.2	Ispitivanje sustava	52
5.2.3	Ispitivanje sustava	57
5.3	Dijagram razmještaja	61
5.4	Setup projekta	62
5.5	Upute za puštanje u pogon	64
6	Zaključak i budući rad	69

Popis literature	71
Indeks slika i dijagrama	73
Dodatak: Prikaz aktivnosti grupe	74

1. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljen predložak.	Lara Đaković	30.10.2022.
0.2	Dodana 2 <i>Use Case</i> dijagrama - admin i vlasnik kluba te opisi za dio obrazaca uporabe	Lara Đaković	1.11.2022.
0.3	Dodan opis projektnog zadatka	Nina Đurić	1.11.2022.
0.4	Dodani funkcionalni zahtjevi	Mateja Golec	2.11.2022.
0.5	Dodan <i>Use Case</i> dijagram - neregistrirani korisnik, klijent i trener te opisi za dio obrazaca uporabe	Ana Vrabec	3.11.2022.
0.6	Dodan UC za neregistriranog korisnika i admina	Lucija Domić	3.11.2022.
0.7	Ispravak obrazaca uporabe i dijagrama obrazaca uporabe za neregistriranog korisnika, klijenta i trenera	Ana Vrabec	4.11.2022.
0.8	Dodani opisi obrazaca uporabe za klub	Karlo Boroš	5.11.2022.
0.9	Dodan sekvencijski dijagram Prijava trenera i funkcionalnosti	Mateja Golec	7.11.2022.
0.10	Dodan opis baze podataka i opis tablica Tečaj, Ples, Lokacija, PlesnjakPles i Trening	Lara Đaković	7.11.2022.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
0.11.	Dodani sekvencijski dijagrami Upravljanje tečajem i Upravljanje plesovima	Nina Đurić	9.11.2022.
0.12	Dodani ostali zahtjevi	Ana Vrabec	9.11.2022.
0.13	Dodani opisi tablica za Klub, Korisnik, TrenerPrijava, Plesnjak, KorisnikTecaj	Karlo Boroš	9.11.2022.
0.14	Napravljena autorizacija i autentikacija na backendu	Luka Nola i Karlo Boroš	11.11.2022.
0.15	Promjena sekvencijskih dijagrama Prijava trenera i Upravljanje tečajem	Mateja Golec	16.11.2022.
0.16	Promijenjen opis, popravljani opisi i tablice baze	Nina Đurić	16.11.2022.
0.17	Dodana arhitektura projekta i promjene opisa baze podataka	Mateja Golec	17.11.2022.
0.18	Dodani dijagrami razreda	Lara Đaković	17.11.2022.
1.0	Verzija samo s bitnim dijelovima za 1. ciklus	*	18.11.2022.
1.1	Dodan dijagram stanja - registrirani korisnik (klijent)	Mateja Golec	11.1.2023.
1.2	Dodan opis korištenih alata i tehnologija	Mateja Golec	11.1.2023.
1.3	Dodan dijagram aktivnosti	Karlo Boroš	12.1.2023.
1.4	Dodan dijagram razmještaja	Nina Đurić	13.1.2023.

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Rev.	Opis promjene/dodatka	Autori	Datum
1.5	Dodan dijagram komponenti	Lucija Domić	13.1.2023.
1.6	Dodae upute za puštanje aplikacije u pogon	Mateja Golec	13.1.2023.
1.7	Dodan zaključak	Lara Đaković	13.1.2023.
1.8	Dovršen klasni dijagram	Lara Đaković	13.1.2023.
1.9	Dodana aktivnost na gitlabu	Lara Đaković	13.1.2023.
1.91	Dodani testovi	Lara Đaković, Nina Đurić, Karlo Boroš, Mateja Golec	13.1.2023.
1.92	Dodana lokalno postavljanje aplikacije	Lucija Domić	13.1.2023.
2.0	Verzija samo s bitnim dijelovima za 2. ciklus	*	13.1.2023.

2. Opis projektnog zadatka

2.1 Opis projektnog zadatka 'Ples'

Cilj ovog projekta je razviti programsku podršku za stvaranje web aplikacije "Ples", koja korisniku olakšava pronalaženje i organizaciju plesnih tečajeva i plesnjaka, time uvelike olakšavajući život ljubiteljima plesa jer će imati sve što im je potrebno u jednoj web aplikaciji

Neregistriranom korisniku se, otvaranjem stranice, prikazuje karta sa dostupnim plesnjacima i lokacijama klubova. Neregistrirani korisnik također može i pregledati profile klubova i tipove plesova koji nudi, a kako bi si olakšao pretragu, plesnjake može filtrirati po tipu plesa, klubu koji ga organizira i tipovima plesa. Isto tako, klubove je moguće filtrirati po tipovima plesa za koje organiziraju tečajeve. Neregistrirani korisnik može kreirati novi račun za klijenta navođenjem sljedećih obaveznih informacija:

- *korisničko ime*
- *lozinka*
- *ime*
- *prezime*
- *spol*
- *datum rođenja*
- *broj mobitela*
- *email adresa*

Korisnik također može, ali i ne mora unijeti sljedeće informacije:

- *opis plesnog iskustva*
- *fotografija*

Neregistrirani korisnik također može kreirati novi račun za klub, te će mu za isto biti potrebno navođenje sljedećih informacija:

- *korisničko ime*

- *lozinka*
- *ime kluba*
- *adresa sjedišta*
- *broj telefona*
- *email*

Nakon stvaranja novog računa za klub, korisnik mora čekati na potvrdu od strane administratora. Nakon provedene registracije, korisnik može mijenjati sve informacije potrebne za registraciju po volji. Također, korisnik uvijek može izbrisati svoj račun.

Registrirani korisnik može obnašati ulogu klijenta, ulogu kluba, uloge trenera i klijenta ili ulogu administratora.

Klijentu se na karti prikazu i tečajevi slobodni za upis. Rezultate može filtrirati po vremenu i tipu plesa. Sličnu mogućnost nalazimo na web stranici <https://danceprogram.duke.edu/courses>, gdje korisnici filtriraju tečajeve po vremenu i vrsti tečaja.

COURSE TYPE		TYPICALLY OFFERED	
- Any -		- Any -	Apply
Number	Title	Codes	
DANCE 89S	First-Year Seminar		
DANCE 101	Introduction to Dance	CCI, ALP	
DANCE 105S	Dance Composition	R, ALP	
DANCE 110	Elementary Modern Dance		
DANCE 116	Alexander Technique for Musicians, Dancers, and Actors	ALP	
DANCE 120	Beginning Ballet		
DANCE 122	Intermediate Ballet		
DANCE 130	African Dance Technique I		
DANCE 131	Capoeira: Brazilian Dance/Martial Art		

Slika 2.1: Primjer prikaza tečajeve

Nakon što odabere željeni tečaj, klijentu se prikazuju informacije o odabranom tečaju, primjerice, vrsta plesa, kalendar te ime, prezime i slika trenera. Ponovno sličnu mogućnost nalazimo na <https://danceprogram.duke.edu/courses/dance-composition>, gdje, kad odaberemo tečaj, prikazu nam se neke informacije o tečaju, u ovom slučaju, vrsta plesa i razdoblje godine u kojem se taj tečaj izvodi, međutim, na ra-

nije spomenutoj web stranici, pri odabiru tečaja, ne nalazimo informacije o treneru i kalendar, što je jedna od odlika projektne aplikacije.

Beginning Ballet

DANCE 120

Basic classical ballet technique, body alignment, vocabulary, and musicality for the absolute beginner. Barre and center exercises included.

Typically Offered

Fall Only



Slika 2.2: Primjer prikaza informacija o tečaju

U kalendaru korisnik može pregledati sve zapisane termine tečaja i njihovu lokaciju s dvoranom. Također, klijent može pregledati i aktivne prijave za tečaj od određenog kluba i prijaviti se na njih.

Klubovi organiziraju plesnjake koji se mogu izvoditi i na lokacijama van kluba, a sadrže naziv, opis i sliku. Plesnjaci nisu ograničeni na samo jedan tip plesa, već je moguće plesati više različitih vrsta plesova.

Na profilu kluba se nalaze informacije o imenu kluba, broj telefona, kratki opis, poveznica na stranicu s grupama za upis, popis plesova koje njihovi treneri nude, prikaz lokacija na karti te popis dvorana po lokacijama.

Plesni Klub Tina, na poveznici <https://www.plesniklubtina.hr/>, korisnicima također nudi informacije o imenu kluba, broj telefona, kratki opis i lokaciju, koja, za razliku od projektne aplikacije, nije prikazana na karti. Također, za razliku od projektne aplikacije, 'Tina' korisnicima prikazuje svoj mail i sponzora, dok s druge strane ne nudi poveznicu na stranicu s grupama za upis, popis plesova u svom repertoaru i popis dvorana.



Slika 2.3: Primjer opisa kluba

Klub može objaviti upise za tečaj s krajnjim rokom za prijavu, te može ograničiti upis u grupu postavljanjem dobne granice ili ograničavanjem upisa na samo jedan spol. Zatim se grupi dodjeljuje trener, skup treninga kroz neko vrijeme, gornja granica broja sudionika, te mnoge informacije koje klijentu mogu pomoći pri odabiru grupe, primjerice težina treninga, posebni uvjeti treniranja i pravila ponašanja. Nakon isteka roka, klub radi selekciju prijavljenih klijenata za grupu. Klub može naknadno mijenjati popis klijenata te uređivati i brisati grupe.

Jedna od zadaća kluba je također izbor trenera.

Ulogu trenera može dobiti i obnašati bilo koji klijent koji određenom klubu pošalje prijavu, motivacijsko pismo i potvrdu u obliku pdf dokumenta da je sposoban držati tečaj plesa, te ga zatim isti klub i potvrdi. Svaki trener ima podstranicu na kojoj se nalazi popis svih grupa koje trenira, a termine može pronaći u kalendaru. Na stranici <https://danceprogram.duke.edu/courses> također nailazimo na podstranice trenera, čiji je sadržaj potpuno drugačiji od sadržaja podstranice trenera projektne aplikacije, primjerice, na projektnoj web aplikaciji ne nailazimo na detaljan opis trenera, njegovog školovanja i kompetencija, te broj za kontakt. Projektna web aplikacija također ne sadrži sliku trenera na njegovoj podstranici.

Iyun Ashani Harrison
Associate Professor of the Practice of Dance

Iyun Ashani Harrison is a dance maker, educator, and executive director of Ballet Ashani. Born in Saint Andrew, Jamaica, Harrison first trained in acting, classical ballet, modern technique, and Jamaican folk dance. He is a graduate of both the Juilliard School (BFA) and Hollins University (MFA).

Harrison danced with the Dance Theatre of Harlem under the artistic direction of Arthur Mitchell. It was at Dance Theatre of Harlem that he developed a love for neo-classical ballet, dancing chor (... [more](#))

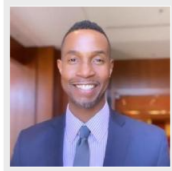
Current Appointments & Affiliations

- Associate Professor of the Practice of Dance, [Dance Program, Trinity College of Arts & Sciences](#) 2021
- Director of Undergraduate Studies of Dance, [Dance Program, Trinity College of Arts & Sciences](#) 2021

Contact Information

2020 Campus Drive, Durham, NC 27705
2020 Campus Drive, Durham, NC 27705
✉ iyun.harrison@duke.edu ☎ (919) 660-3355
[Ballet Ashani](#)

[Manage This Profile](#)
[Add Data to my Website](#)



Slika 2.4: Primjer podstranice o treneru

Trener, nakon otvaranja tečaja, uz opće informacije, dobije i popis klijenata koji su potvrdili svoj dolazak na tečaj.

Administrator ima najveće ovlasti. On ima mogućnost mijenjanja, dodavanja i brisanja plesova. Plesovi sadrže naziv, kratki opis, sliku i link na video primjer dotičnog plesa. Također, administrator može pregledati popis svih klijenata i klubova te istima uređivati korisničke račune.

Uz manje promjene, ova web aplikacija može imati širok spektar uporabe. Uz ovakvu implementaciju, neće ju biti teško izmijeniti da pokrije neka druga područja. Primjerice, ukoliko stavimo nogometni klub, trening, nogometni tečaj i vrstu treninga u ulogu plesnog kluba, plesnjaka, plesnog tečaja i plesa, uz neznatne adaptacije ova aplikacija će biti transformirana u web aplikaciju koja nogometašima olakšava pronalazak nogometnog kluba. Osim u svijetu sporta, ova web aplikacija može se adaptirati da postane pogodna za razne umjetnosti. Zamjenimo li plesni klub sa školom slikanja, plesnjak sa satom slikanja, tečaj plesa s tečajem slikanja a ples sa tehnikom slikanja, dobit ćemo web aplikaciju koja uživateljima slikanja olakšava pronalaženje pogodnih tečajeva i škola. Također, ova web aplikacija može prilagodbom postati aplikacija za pronalaženje škola i tečajeva za strane jezike, aplikacija za pronalaženje raznih volonterskih udruga i slično.

Zbog svoje jednostavnosti, ovu web aplikaciju je vrlo lako nadograditi da korisniku nudi brojne druge funkcionalnosti. Primjerice, uz određene adaptacije na klijentskoj i poslužiteljskoj strani, klijentu se može omogućiti davanje ocjene klubu, koja se uračuna u dosadašnje ocjene tog kluba i prikazuje korisnicima na stranici kluba,

čime bi korisnici mogli jasno vidjeti koliko su prijašnji polaznici tečajeva tog kluba bili zadovoljni s njim. Također, web aplikacija bi se mogla nadograditi tako da sadrži forum u kojem korisnici raspravljaju o plesovima i tečajevima, čime bi se korisnicima omogućilo sklapanje novih prijateljstava i diskutiranje o svom hobiju.

3. Specifikacija programske potpore

3.1 Funkcionalni zahtjevi

Dionici:

1. Neregistrirani korisnik
2. Registrirani korisnik
 - (a) klijent
 - (b) vlasnik kluba
 - (c) trener
 - (d) administrator
3. Razvojni tim
4. Asistenti

Aktori i njihovi funkcionalni zahtjevi:

1. Neregistrirani korisnik (inicijator) može:
 - (a) pregledati na karti dostupne plesnjake i lokacije klubova
 - (b) odabrati profile klubova (ime kluba, kontakt telefon, email adresu, kratki opis) i pregledati tipove plesa (naziv, kratki opis, slika i link na video primjer plesa) koje ti klubovi nude
 - (c) registrirati se, kreirati novi korisnički račun za klijenta za koji su mu potrebni korisničko ime, lozinka, ime, prezime, spol, datum rođenja, broj mobitela, email adresa i opcionalno opis plesnog iskustva i fotografija
2. Klijent (inicijator) može:
 - (a) pregledavati i mijenjati osobne podatke
 - (b) izbrisati svoj korisnički račun
 - (c) pregledati na karti tečajeve slobodne za upis
 - (d) odabrati tečaj/grupu i dobiti prikaz relevantnih informacija (vrsta plesa, kalendar s terminima tečaja i lokacijama s dvoranama, ime i prezime te sliku trenera)

- (e) kreirati novi korisnički račun za klub za koji su mu potrebni korisničko ime, lozinka, ime kluba, adresa sjedišta, telefon i email čime postaje vlasnik kluba
- (f) poslati prijavu klubu da postane trener koja mora sadržavati motivacijsko pismo i potvrdu u pdf formatu da je osposobljen držati tečaj plesa
- (g) pregledati aktivne prijave na tečaj/grupu kluba (informacije o treneru, skupu treninga kroz neko vrijeme, maksimalnom broju sudionika, opis s dodatnim informacijama o težini treninga, uvjetima treniranja i pravilima ponašanja) i prijaviti se

3. Vlasnik kluba (inicijator) može:

- (a) organizirati plesnjak uz odgovarajući opis (lokacija, naziv, opis i slika) – lokacija plesnjaka može biti na lokaciji kluba ili izvan nje
- (b) potvrditi klijenta kao trenera
- (c) objaviti upise za tečaj/grupu s krajnjim rokom prijave i ograničenjem na dob i spol
- (d) odabrati klijente koji se primaju na tečaj/grupu
- (e) naknadno mijenjati popis klijenata
- (f) uređivati i brisati tečajeve/grupe

4. Trener (inicijator) može:

- (a) imati podstranicu na kojoj vidi popis tečajeve/grupa koje trenira
- (b) vidjeti na kalendaru sve termine tečajeve/grupa koje vodi
- (c) vidjeti opće informacije tečaja/grupe za kojeg je zadužen i popis klijenata koji bi trebali biti nazočni na tečaju/grupi

5. Administrator (inicijator) može:

- (a) može dodati, mijenjati i brisati plesove
- (b) odobriti prijavu kluba za korištenje aplikacije
- (c) pregledati popis svih klijenata i klubova
- (d) uređivati korisničke račune

6. Baza podataka(sudionik) može:

- (a) pohranjuje sve podatke o korisnicima i njihovim ovlastima
- (b) pohranjuje sve podatke o plesnim tečajevima i plesnjacima, njihovim lokacijama i opisima

3.1.1 Obrasci uporabe

Opis obrazaca uporabe

UC1 - Registracija korisnika

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Stvoriti korisnički račun za pristup sustavu
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za registraciju
 2. Korisnik unosi potrebne korisničke podatke
 3. Korisnik prima obavijest o uspješnoj registraciji
- **Opis mogućih odstupanja:**
 - 4.a Odabir već zauzetog korisničkog imena i/ili e-maila, unos korisničkog podatka u nedozvoljenom formatu ili pružanje neispravnog e-maila
 1. Sustav obavještava korisnika o neuspjelom upisu i vraća ga na stranicu za registraciju
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od registracije

UC2 - Registracija kluba

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Stvoriti korisnički račun za klub
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Korisnik odabire opciju za registraciju kluba
 2. Korisnik unosi podatke vezano za klub
 3. Korisnik prima obavijest o uspješnoj registraciji
- **Opis mogućih odstupanja:**
 - 4.a . Odabir već zauzetog korisničkog imena i/ili e-maila, unos korisničkog podatka u nedozvoljenom formatu ili pružanje neispravnog e-mail
 1. Sustav obavještava korisnika o neuspjelom upisu i vraća ga na stranicu za registraciju
 2. Korisnik mijenja potrebne podatke te završava unos ili odustaje od registracije

UC3 - Pregled klubova

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Pregledati sve dostupne klubove
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Karta je prikazana prilikom učitavanja aplikacije
 2. Korisnik na karti odabire klub
 3. Prikazuju se tipovi plesa koje pojedini klub nudi
 4. Klubovi se mogu filtrirati po tipovima plesa za koje organizira tečaj
 5. Odabir filtra
 6. Prikazuju se filtrirani plesnjaci

UC4 -Pregled plesnjaka

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Pregledati dostupne plesnjake
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Korisniku se na karti prikazuju dostupni plesnjaci
 2. Dostupni plesnjaci se mogu filtrirati po:
 - (a) Tipu plesa
 - (b) Klubu koji ga organizira
 3. Odabir filtra
 4. Prikazuju se filtrirani plesnjaci

UC5 -Odabir kluba

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Pretraga određenog kluba
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Pretraživanje kluba odvija se upisom osnovnih podataka o klubu
 2. Prikazuju se podatci za traženi klub

UC6 -Dodavanje plesa

- **Glavni sudionik:** Administrator
- **Cilj:** Dodati novi ples
- **Preduvjet:** : Korisnik je registriran i dodijeljena su mu prava administratora
- **Sudionici:** Baza podataka

- **Opis osnovnog tijeka:**

1. Prikazuju se dostupni klubovi
2. Odabire se klub kojem želi dodati određeni ples
3. Unose se osnovni podatci o plesu
4. Dodaje se novi ples u bazu podataka
5. Ples postaje vidljiv unutar mogućih plesova za određeni klub

UC7 - Izmjena plesa

- **Glavni sudionik:** Administrator
- **Cilj:** Izmjena postojećeg plesa
- **Preduvjet:** : Korisnik je registriran i dodijeljena su mu prava administratora
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Prikazuju se dostupni klubovi
 2. Administrator odabire klub
 3. Administrator odabire ples kojem želi izmijeniti određene podatke
 4. Promjene se upisuju u bazu podataka
 5. Izmijenjene postaju vidljive za odabrani ples

UC8 - Brisanje plesa

- **Glavni sudionik:** Administrator
- **Cilj:** Brisanje postojećeg plesa
- **Preduvjet:** : Korisnik je registriran i dodijeljena su mu prava administratora
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Prikazuju se dostupni klubovi
 2. Odabir kluba
 3. Odabir plesa koji se želi obrisati
 4. Odabrani ples se uklanja iz baze podataka
 5. Obrisani ples nije više vidljiv u aplikaciji

UC9 - Odobrenje prijave kluba

- **Glavni sudionik:** Administrator
- **Cilj:** Odobrenje prijave kluba za korištenje aplikacije
- **Preduvjet:** : Korisnik je registriran i dodijeljena su mu prava administratora
- **Sudionici:** Baza podataka

- **Opis osnovnog tijeka:**

1. Prikaz svih klubova koji traže pristup sustavu
2. Prijava se može odbiti ili prihvatiti
3. U slučaju prihvatanja, podatci za klub se spremaju u bazu podataka
4. Klub je nakon toga vidljiv u aplikaciji

UC10 - Pregled korisnika

- **Glavni sudionik:** Administrator
- **Cilj:** Pregledati registrirane korisnike
- **Preduvjet:** : Korisnik je registriran i dodijeljena su mu prava administratora
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Administrator odabire opciju pregledavanja korisnika
 2. Prikaže se lista svih ispravno registriranih korisnika s osobnim podacima

UC11 - Uređivanje korisničkih računa

- **Glavni sudionik:** Administrator
- **Cilj:** Uređivanje korisničkih računa
- **Preduvjet:** : Korisnik je registriran i dodijeljena su mu prava administratora
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Odabir hoće li se urediti korisnički račun kluba ili klijenta
 2. Odabir određenog korisničkog računa
 3. Prikazuju se podatci koje je moguće izmijeniti za određeni korisnički račun
 4. Promjene se spremaju u bazu podataka

UC12 -Prijava u sustav

- **Glavni sudionik:** Klijent
- **Cilj:** Dobiti pristup korisničkom sučelju
- **Sudionici:** Baza podataka
- **Preduvjet:** Registracija
- **Opis osnovnog tijeka:**
 1. Unos korisničkog imena i lozinke
 2. Potvrda o ispravnosti unesenih podataka

3. Pristup korisničkim funkcijama
- **Opis mogućih odstupanja:**
 - 2.a Neispravno korisničko ime/lozinka
 1. Sustav obavještava korisnika o neuspješnoj prijavi i vraća ga na stranicu kao neregistriranog korisnika

UC13 -Pregled osobnih podataka

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati osobne podatke
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju prikaza osobnih podataka
 2. Aplikacija prikazuje osobne podatke

UC14 -Promjena osobnih podataka

- **Glavni sudionik:** Klijent
- **Cilj:** Promijeniti osobne podatke
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju promjene osobnih podataka
 2. Klijent mijenja svoje osobne podatke
 3. Klijent sprema promjene
 4. Ažuriraju se podatci u bazi podataka
- **Opis mogućih odstupanja:**
 - 2.a Klijent promjeni osobne podatke, ali ne odabere opciju spremanja promjena
 1. Sustav obavještava klijenta da nije spremio podatke prije izlaska iz prozora

UC15 -Brisanje korisničkog računa

- **Glavni sudionik:** Klijent
- **Cilj:** Obrisati korisnički račun
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju prikaza osobnih podataka
 2. Otvara se stranica s osobnim podacima klijenta

3. Klijent briše račun
4. Korisnički račun briše se iz baze podataka
5. Otvara se stranica vidljiva neregistriranom korisniku

UC16 -Pregled tečajeva na karti slobodnih za upis

- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati na karti tečajeve slobodne za upis
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju prikaza slobodnih tečajeva na karti
 2. Otvara se stranica s kartom i prikazom tečajeva
 3. Ukoliko želi, klijent filtrira tečajeve prema željenom vremenu i vrsti plesa
- **Opis mogućih odstupanja:**
 - 2.a Nema tečajeva slobodnih za upis
 1. Sustav obavještava korisnika da trenutno nema tečajeva slobodnih za upis (općenito ili za određeno mjesto i/ili vrstu plesa)

UC17 -Odabir tečaja s karte

- **Glavni sudionik:** Klijent
- **Cilj:** Odabrati jedan od slobodnih tečajeva
- **Sudionici:** Baza podataka
- **Preduvjet:** Postoji slobodno mjesto za upis na tečaj
- **Opis osnovnog tijeka:**
 1. Klijent odabire željeni tečaj na karti
 2. Otvara se stranica s prikazom relevantnih informacija o tečaju

UC18 - Organizacija plesnjaka

- **Glavni sudionik:** Vlasnik kluba
- **Cilj:** Organizacija plesnjaka uz opis
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Vlasnik kluba odabirom pretraživanja dolazi na stranicu za unos novog plesnjaka
 2. Vlasnik kluba putem forme upisuje potrebne podatke o plesnjaku
 3. Vlasnik kluba potvrđuje izradu plesnjaka

4. Potvrdom odabira podaci se spremaju u bazu podataka

UC19 - Objava upisa za tečaj

- **Glavni sudionik:** Vlasnik kluba
- **Cilj:** Upis klijenata na tečaj
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Vlasnik kluba na izborniku odabire opciju za objavu novog tečaja
 2. Vlasnik kluba treba unijeti potrebne podatke o tečaju
 3. Potvrdnim odabirom podaci o tečaju se spremaju u bazu podataka i postaju dostupni svima na izbor
- **Opis mogućih odstupanja:**
 - 2.a Vlasnik kluba nije ispunio sve podatke
 1. Stranica vraća vlasnika kluba natrag te traži od njega unos svih potrebnih podataka

UC20 - Odabir klijenata za tečaj

- **Glavni sudionik:** Vlasnik kluba
- **Cilj:** Popuna kvote za tečaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Postoje prijave za tečaj
- **Opis osnovnog tijeka:**
 1. Vlasnik kluba dobiva popis klijenata koji su se prijavili na tečaj
 2. Vlasnik kluba bira klijente prema ograničenjima koja su postavili
 3. Klikom na potvrdu odabira lista klijenata za tečaj je završena i sprema se u bazu podataka

UC21 - Mijenjanje popisa klijenata

- **Glavni sudionik:** Vlasnik kluba
- **Cilj:** Promjena popisa klijenata primljenih na tečaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Postoji popis odabranih klijenata za tečaj
- **Opis osnovnog tijeka:**
 1. Vlasnik kluba iz baze podataka uzima popis trenutnih klijenata na tečaju
 2. Vlasnik kluba miče klijente sa popisa ili dodaje nove, ovisno o potrebi
 3. Promijenjeni popis sprema se natrag u bazu podataka

UC22 - Uređivanje tečaja

- **Glavni sudionik:** Vlasnik kluba
- **Cilj:** Promjena uvjeta nekog tečaja
- **Sudionici:** Baza podataka
- **Preduvjet:** Tečaj postoji i objavljen je
- **Opis osnovnog tijeka:**
 1. Vlasnik kluba na izborniku odabire promjenu tečaja
 2. Iz baze podataka se dopremaju podaci o tečaju
 3. Vlasnik kluba mijenja prethodno određene karakteristike
 4. Vlasnik kluba sprema podatke nakon što je gotov sa izmjenom
 5. Promijenjene karakteristike se spremaju u bazu podataka
- **Opis mogućih odstupanja:**
 - 2.a Vlasnik kluba unio je neispravne podatke
 1. Stranica od njega zahtjeva da unese ispravne podatke

UC23 - Brisanje tečaja

- **Glavni sudionik:** Vlasnik kluba
- **Cilj:** Obrisati tečaj
- **Sudionici:** Baza podataka
- **Preduvjet:** Tečaj postoji
- **Opis osnovnog tijeka:**
 1. Vlasnik kluba odabire opciju brisanja tečaja
 2. Tečaj se briše iz baze podataka i nije više dostupan

UC24 -Slanje prijave za trenera

- **Glavni sudionik:** Klijent
- **Cilj:** Klijent šalje prijavu određenom klubu kako bi postao njihov trener
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Klijent odabire određeni klub
 2. Klijent odabire opciju slanja prijave za trenera
 3. Otvara se prozor sa formom za unos osobnih podataka i datoteka – motivacijsko pismo i potvrda o sposobnosti vođenja tečaja
 4. Klijent unosi podatke i potvrđuje slanje
 5. Podaci se pohranjuju u bazu podataka

6. Klubu se aplikaciji prikazuje nova prijava
- **Opis mogućih odstupanja:**
 - 4.a Podaci u formi imaju krivi format ili nisu ispunjena obavezna polja
 1. Aplikacija obavještava klijenta da unese ispravne i obavezne podatke

UC25 – Potvrda prijave za trenera

- **Glavni sudionik:** Vlasnik kluba
- **Cilj:** Klijent postaje trener u određenom klubu
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent poslao prijavu za trenera
- **Opis osnovnog tijeka:**
 1. Klub odabire prijavu od određenog klijenta
 2. Klub potvrđuje prijavu
 3. Klijentu se dodjeljuju ovlasti trenera
 4. Prijava se označava kao obrađena

UC26 - Odbijanje prijave za trenera

- **Glavni sudionik:** Vlasnik kluba
- **Cilj:** Klijentu se odbija poslana prijava za trenera
- **Sudionici:** Baza podataka
- **Preduvjet:** Klijent poslao prijavu za trenera
- **Opis osnovnog tijeka:**
 1. Klub odabire prijavu od određenog klijenta
 2. Klub odbija određenog klijenta
 3. Prijava se označava kao obrađena

UC27 – Pregled vođenih tečajeva

- **Glavni sudionik:** Trener
- **Cilj:** Vidjeti popis tečajeva koje vodi
- **Sudionici:** Baza podataka
- **Preduvjet:** Trener vodi tečaj za određeni klub
- **Opis osnovnog tijeka:**
 1. Trener odabire opciju za prikaz svojih tečajeva
 2. Aplikacije prikazuje popis tečajeva koje trener vodi
 3. Trener odabire određeni tečaj

4. Prikazuje se naziv tečaja sa opisom i nužnim informacijama o tečaju

UC28 - Pregled kalendara s vođenim terminima

- **Glavni sudionik:** Trener
- **Cilj:** Vidjeti kalendar s popisom treninga koje vodi određenog dana i sata
- **Sudionici:** Baza podataka
- **Opis osnovnog tijeka:**
 1. Trener odabire opciju za prikaz svojih vođenih termina
 2. Prikazuje se kalendar s označenim terminima kada vodi tečaj

UC29 - Pregled polaznika

- **Glavni sudionik:** Trener
- **Cilj:** Vidjeti popis nazočnih sudionika na tečaju
- **Sudionici:** Baza podataka
- **Preduvjet:** Trener vodi odabrani tečaj
- **Opis osnovnog tijeka:**
 1. Trener odabire određeni tečaja iz popisa tečajeva
 2. Prikazuje se popis sudionika na tečaju s osnovnim osobnim podacima

UC30 - Pregled aktivnih prijava na tečaj kluba

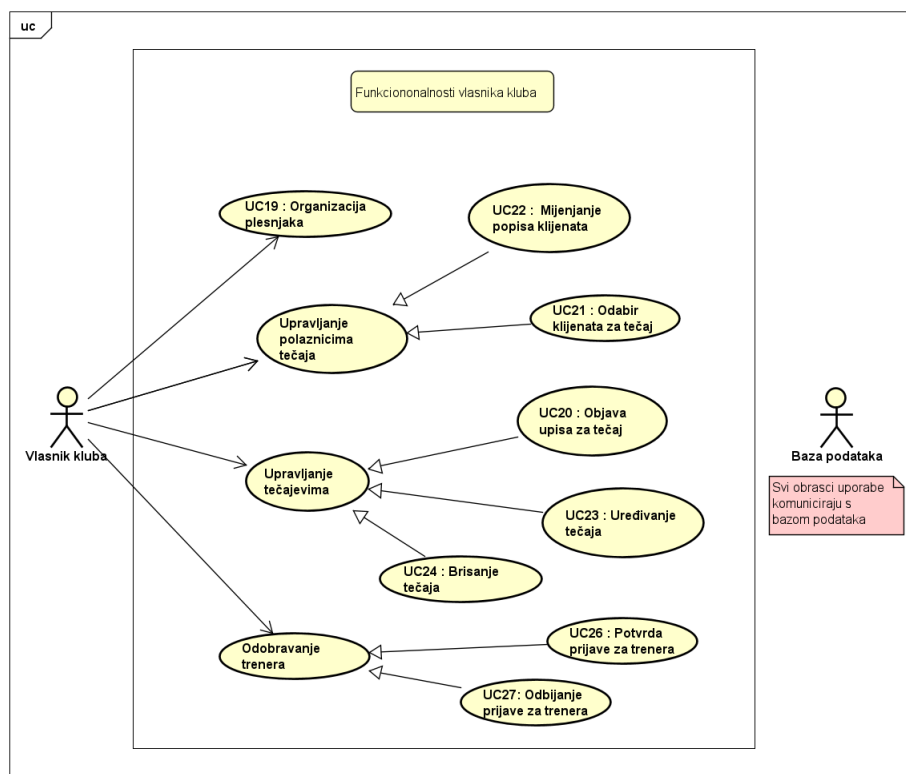
- **Glavni sudionik:** Klijent
- **Cilj:** Pregledati aktivne prijave na tečajeve od klubova
- **Sudionici:** Baza podataka
- **Preduvjet:** Postoje aktivne prijave na tečajeve
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju prikaza aktivnih prijava na tečajeve
 2. Prikazuje se stranica koja sadrži relevantne informacije o treninzima

UC31 -Prijava na tečaj kluba

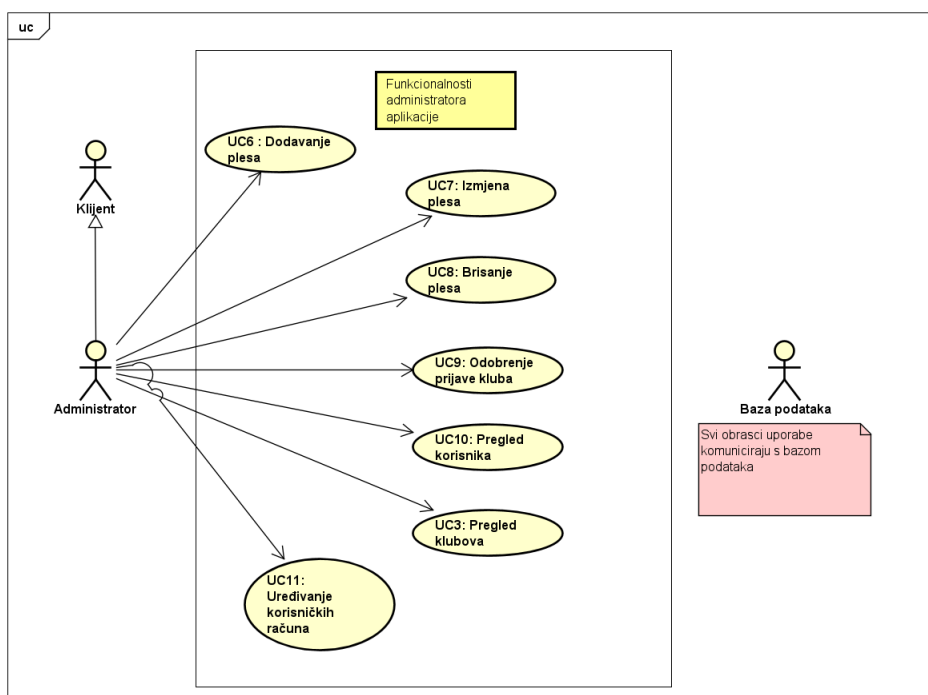
- **Glavni sudionik:** Klijent
- **Cilj:** Prijaviti se na tečaj kluba
- **Sudionici:** Baza podataka
- **Preduvjet:** Postoje aktivne prijave na tečaj
- **Opis osnovnog tijeka:**
 1. Klijent odabire opciju prikaza aktivnih prijava na tečajeve
 2. Prikazuje se stranica koja sadrži relevantne informacije o treninzima

3. Klijent odabire opciju prijave na tečaj
4. Klijent potvrđuje svoj odabir
5. Prijava klijenta sprema se u bazu podataka te na listu prijavljenih kluba

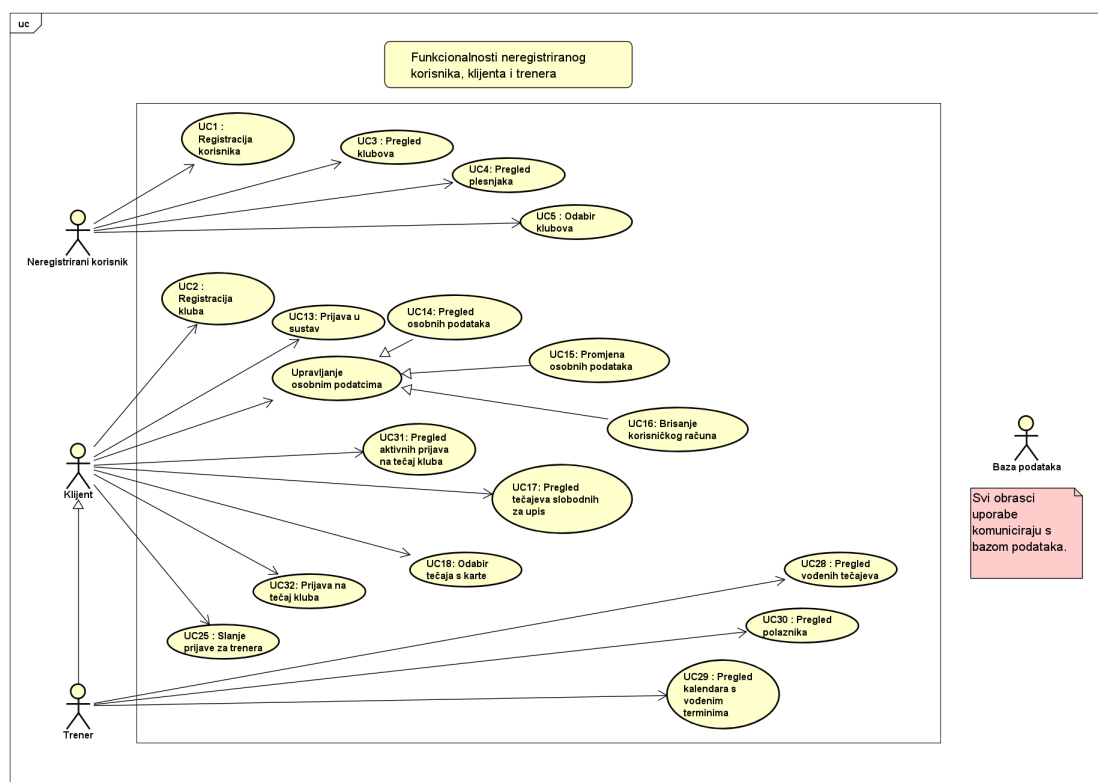
Dijagrami obrazaca uporabe



Slika 3.1: Dijagram obrasca uporabe, funkcionalnost vlasnika kluba.



Slika 3.2: Dijagram obrasca uporabe, funkcionalnost administratora.

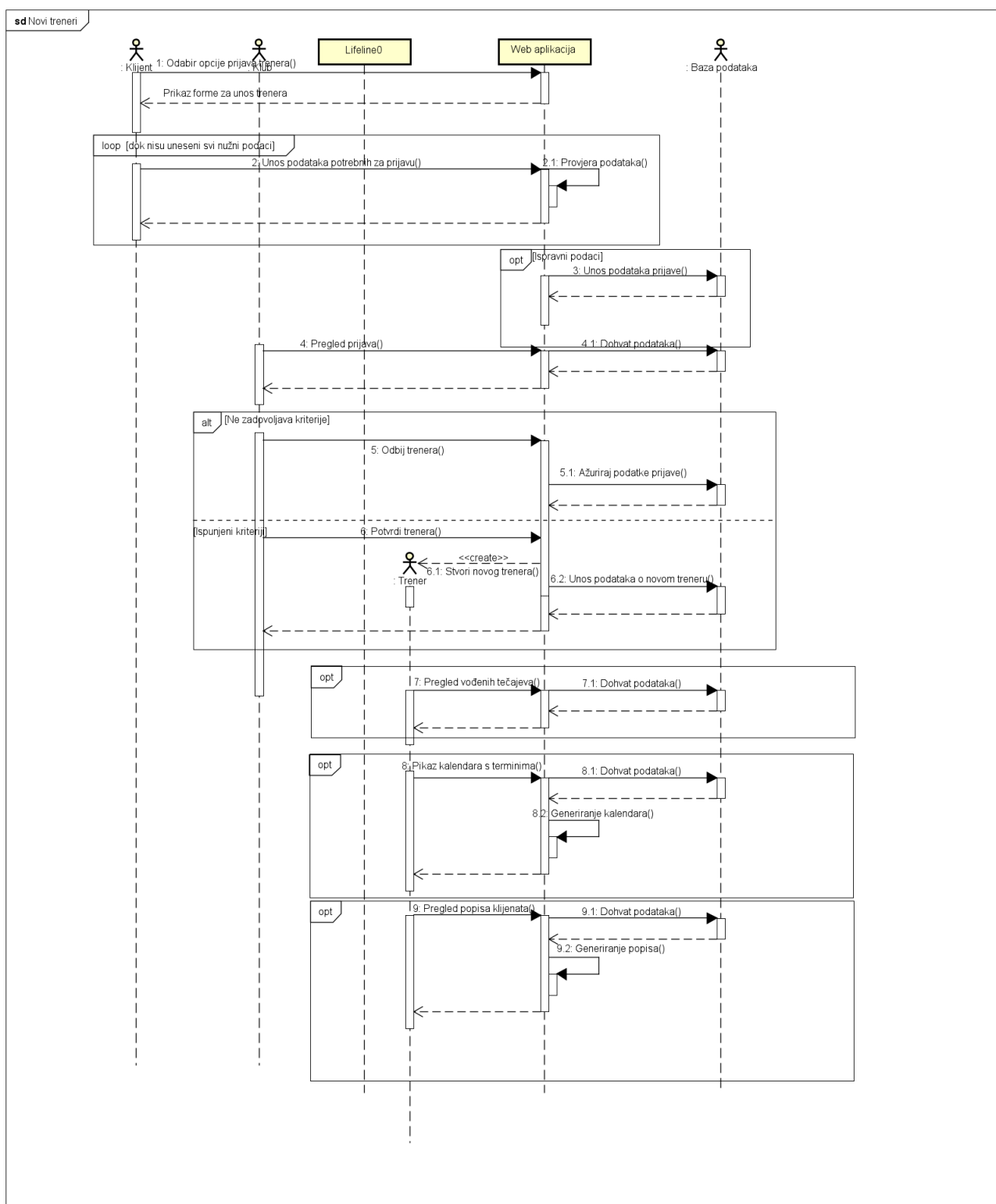


Slika 3.3: Dijagram obrasca uporabe, funkcionalnost neregistriranog korisnika, klijenta i trenera.

3.1.2 Sekvencijski dijagrami

Obrasci uporabe – UC25, UC26, UC27, UC28, UC29, UC30 - Slanje prijave za trenera, Potvrda prijave za trenera, Odbijanje prijave za trenera, Pregled vođenih tečajeva, Pregled kalendara s vođenim terminima, Pregled polaznika

Klijent odabire opciju prijave trenera na poslužitelju, a poslužitelj mu vraća formu koju je potrebno ispuniti. Klijent unosi podatke potrebne za prijavu kako bi postao trener, poslužitelj provjerava ispravnost podataka te ih, ako su ispravni, pohranjuje u bazu podataka kao novu prijavu. Vlasnik kluba ima mogućnost pregleda svih prijava za trenere, za što odabire pregled prijava na poslužitelju, poslužitelj dohvaća podatke iz baze podataka te ih prikazuje vlasniku kluba. Novu prijavu za trenera vlasnik kluba može potvrditi ili odbiti. U slučaju ispunjenih kriterija vlasnik kluba odabire opciju potvrde trenera na poslužitelju, poslužitelj generira objekt novog trenera i unosi podatke o novom treneru u bazu podataka te vraća potvrdu o obavljenim akcijama. Ukoliko kriteriji nisu zadovoljeni, vlasnik kluba odabire opciju odbijanja nove prijave na poslužitelju, poslužitelj mijenja podatke vezane uz tu prijavu u bazi podataka te šalje potvrdu o obavljanju navedenih promjena. Klijent koji ima potvrđenu ulogu trenera može na poslužitelju zatražiti pregled vođenih tečajeva, poslužitelj dohvaća podatke o njegovim tečajevima iz baze podataka te mu ih prikazuje. Također, može vidjeti prikaz kalendara s terminima tečajeva koje vodi, šalje zahtjev na poslužitelj, poslužitelj dohvaća podatke iz baze podataka, generira kalendar s dohvaćeni podacima te ih prikazuje treneru. Treneru je na poslužitelju omogućen i odabir pregleda popisa klijenata nazočnih na pojedinom terminu, poslužitelj u tom slučaju dohvaća podatke iz baze podataka, generira odgovarajući popis i prikazuje ih treneru.

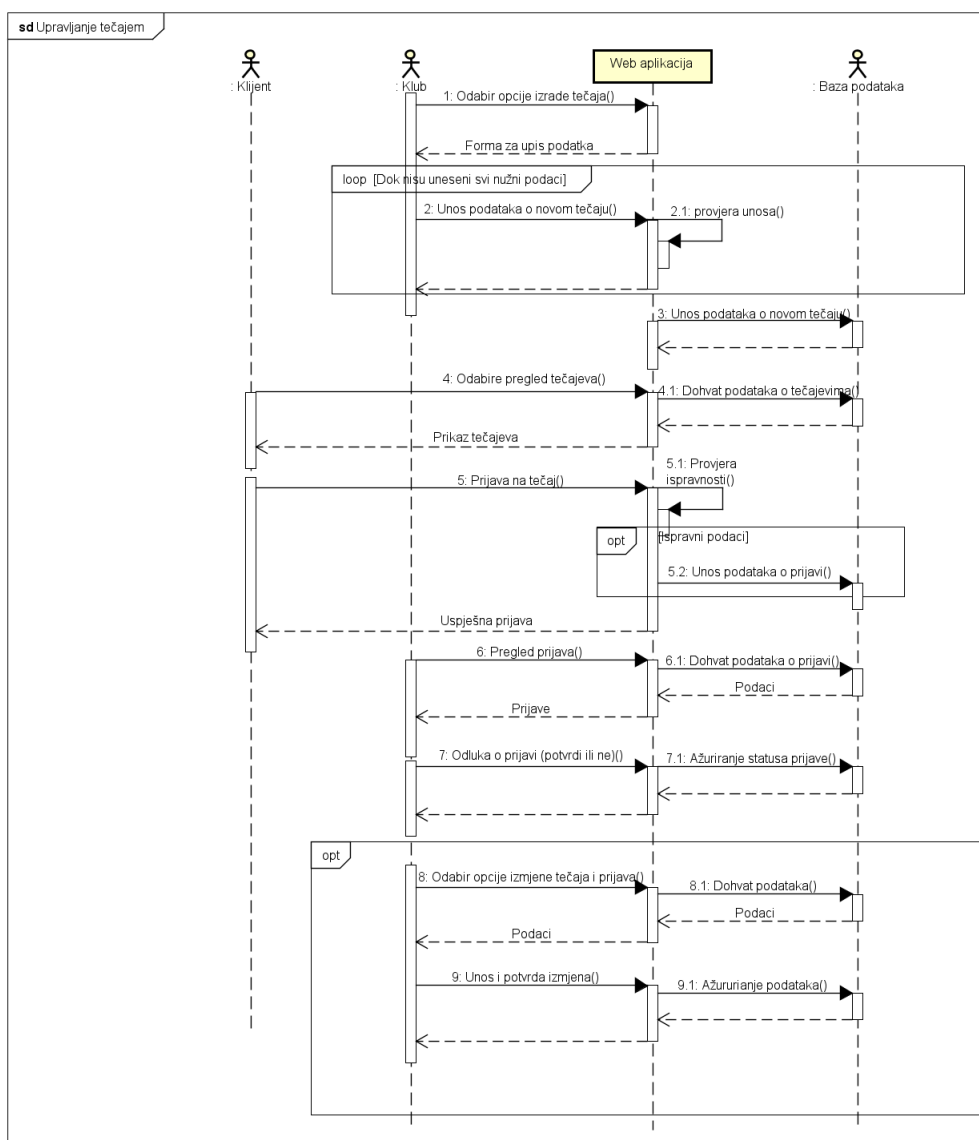


Slika 3.4: Sekvencijski dijagram - Prijava trenera i njegove funkcionalnosti

Obrasci uporabe - UC17, UC20, UC21, UC22, UC32 - Objava upisa na tečaj, pre-

gled tečajeva, prijava na tečaj, odabir klijenata za tečaj i mijenjanje popisa klijenata

Vlasnik kluba šalje zahtjev za izradu tečaja, nakon čega mu poslužitelj vraća formu za upis podataka o tečaju. Poslužitelj zatim podatke o tečaju upisuje u bazu. Klijent šalje zahtjev za pregledom tečajeva. Poslužitelj iz baze dohvaća podatke o tečajevima i prikazuje ih klijentu. Klijent odabire jedan tečaj i prijavi se na njega, nakon čega poslužitelj vrši provjeru ispravnosti. Ako je klijent unio ispravne podatke, poslužitelj u bazu unosi podatke o prijavi i vraća korisniku povratnu informaciju o uspješnosti prijave. Vlasnik kluba šalje zahtjev poslužitelju za pregled prijave. Poslužitelj dohvaća prijave iz baze i šalje ih vlasniku kluba na pregled. Vlasnik kluba zatim odlučuje o potencijalnoj potvrdi klijenta na tečaj, nakon čega poslužitelj ažurira status prijave u bazi podataka. Vlasnik kluba po volji može izmijeniti popis klijenata za tečaj, za što prvo mora poslati zahtjev poslužitelju za dohvat podataka o tečaju. Poslužitelj dohvaća podatke iz baze i šalje ih vlasniku kluba. Vlasnik kluba zatim vrši izmjene i potvrđuje ih, nakon čega poslužitelj ažurira podatke o tečaju u bazi.

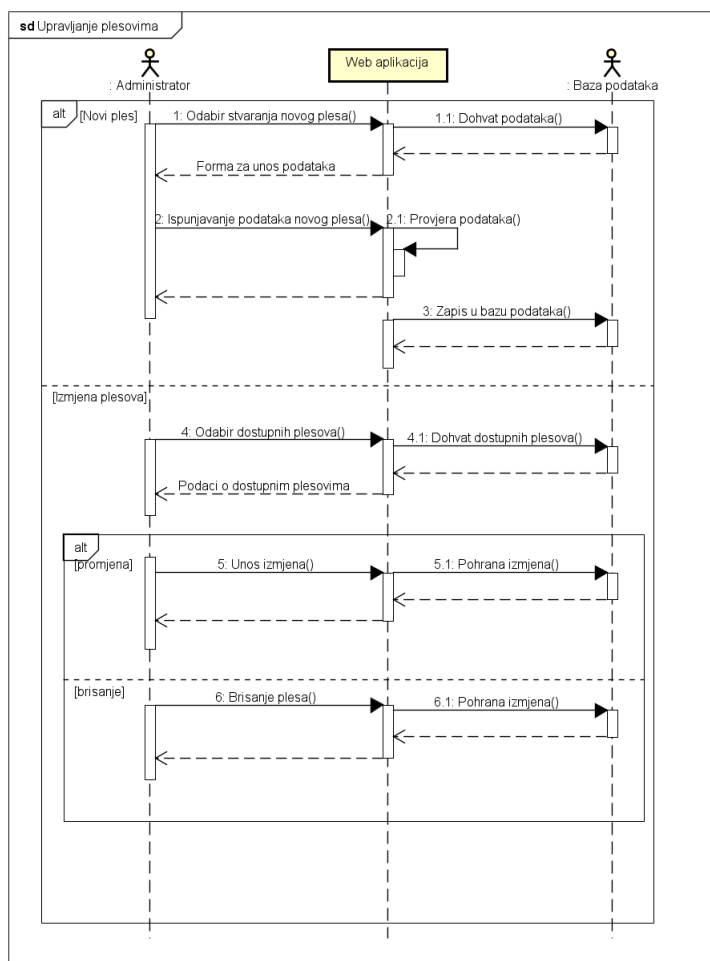


Slika 3.5: Sekvencijski dijagram - Dodavanje tečajeva i prijava na tečajeve

Obrasci uporabe UC6, UC7 i UC8 – dodavanje plesova, izmjena plesova i brisanje plesova

Administrator može dodavati nove plesove i vršiti promjenu nad već postojećim plesovima. Ukoliko se administrator odluči za dodavanje plesa, on šalje poslužitelju zahtjev za stvaranje novog plesa, nakon čega web aplikacija vrši dohvat podataka iz baze te vraća administratoru formu za unos podataka. Administrator ispunjava podatke o novom plesu, a zatim poslužitelj provjerava ispravnost unesenih podataka i šalje ih u bazu podataka. Ukoliko se administrator odluči za izmjenu plesova, on šalje zahtjev poslužitelju za odabir dostupnih plesova. Poslužitelj dohvaća dos-

tupne plesove iz baze te ih vraća poslužitelju. Poslužitelj zatim nad dostupnim plesovima vrši promjene ili briše ples, nakon čega poslužitelj pohranjuje izmjene u bazu podataka



Slika 3.6: Sekvencijski dijagram - Dodavanje, izmjena i brisanje plesova

3.2 Ostali zahtjevi

- Sustav treba omogućiti rad više korisnika u stvarnom vremenu
- Sustav treba ispravno funkcionirati u svim web preglednicima
- Korisničko sučelje i sustav moraju podržavati hrvatsku abecedu (dijakritičke znakove) pri unosu i prikazu tekstualnog sadržaja
- Potpuno učitavanje početne stranice aplikacije ne smije trajati duže od 3 sekunde
- Izvršavanje dijela programa u kojem se pristupa bazi podataka ne smije trajati duže od 4 sekunde
- Sustav treba biti implementiran kao web aplikacija koristeći objektno-orijentirane jezike
- Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava
- Sustav treba biti jednostavan za korištenje, jasan korisnicima s manje informatičkog iskustva bez opširnih uputa
- Nadogradnja sustava ne smije narušavati postojeće funkcionalnosti sustava
- Veza s bazom podataka mora biti kvalitetno zaštićena, brza i otporna na vanjske greške
- Sustav ne smije omogućiti registraciju korisnika dok nije unesena jaka lozinka (barem jedno veliko slovo, znamenka i specijalni znak)
- Sustav koristi srednjoeuropsko standardno vrijeme, GMT+1
- Pristup sustavu mora biti omogućen iz javne mreže pomoću HTTPS

4. Arhitektura i dizajn sustava

- **Proces odabira arhitekture**

Razmatrajući cilj projektnog zadatka, koji je izrada web aplikacije koja olakšava pronalaženje i organizaciju plesnih tečajeva i plesnjaka, specifikacijom svih dionika i njihovih mogućnosti prilikom korištenja aplikacije te definiranjem funkcionalnih zahtjeva zaključili smo da naša aplikacija mora imati tri osnovne razine – razinu klijenta, razinu web-aplikacije te razinu baze podataka. Također, prateća tri sloja koja odgovaraju navedenoj podjeli – sloj korisničkog sučelja, sloj aplikacijske logike te sloj pristupa podacima.

Nadalje, slijedeći principe dobrog oblikovanja, kako bi povećali koheziju i smanjili nepotrebne međuovisnosti, kod koji obavlja pojedinu operaciju vezanu uz određeni entitet je grupiran, dok se sve ostalo što nije izravno vezano uz tu funkcionalnost stavlja izvan te operacije.

Razine također trebaju formirati hijerarhiju, tako da su svi resursi za pristup skupu povezanih usluga na jednom mjestu, čime se omogućuje da viša razina može putem programskog sučelja pristupiti nižoj.

Također, kako bi si olakšali snalaženje cilj je grupirati sve dijelove programa koji pristupaju ili mijenjaju određene podatke te grupirati procedure koje se izvode slijedno i izmjenjuju podatke. Pomoćne programe primjenjive na više različitih grupa, koji se logički ne mogu smjestiti u ostale grupe odvajamo u grupu pomoćnih programa (utils/shared/common).

Ovisnosti između pojedinih modula želimo smanjiti kako nam neke jednostavnije izmjene ne bi zahtijevale promjene na velikom broju ostalih modula te želimo da nam se jasno vidi povezanost određenih entiteta i njihovih odgovarajućih sučelja.

U aplikaciji također želimo ostvariti povećanje ponovne uporabivosti što ćemo postići korištenjem programskih knjižnica i radnih okvira.

Nakon svih razmatranja odabrali smo višeslojnu arhitekturu temeljenu na odnosu klijent-poslužitelj, najbližnju MVC stilu arhitekture.

- **Višeslojna arhitektura**

Karakteristika klijent-poslužitelj opisuje odnos programa koji surađuju u aplikaciji. Ovaj model arhitekture uključuje razmatranje funkcioniranja klijenta i poslužitelja prije, tijekom i po završetku zajedničke komunikacije.

Aplikacija ima tri osnovna sloja:

1. *Klijentsku komponentu* – sloj s korisničkim sučeljem
2. *Poslužiteljsku komponentu* – sloj aplikacijske logike s implementacijom poslovnih procesa i izračuna
3. *Bazu podataka* – podatkovni sloj za pohranu podataka

Klijentska komponenta predstavljena je web preglednikom, programom koji klijentu omogućuje pregled web-stranica i multimedijских sadržaja vezanih uz te stranice odnosno ima mogućnost pristupa informacijama i zahtijevanja usluga poslužitelja. Web preglednik je prevoditelj koji web stranicu pisanu u kodu interpretira i prikazuje u klijentu razumljivom obliku. Koristeći web preglednik klijent šalje zahtjeve web poslužitelju.

Poslužiteljska komponenta predstavlja web poslužitelj, program čija je osnovni zadatak pohrana, obrada i dostava web stranica klijentu te nam ona predstavlja centar za razmjenu informacija i pružanje usluga. Komunikacija se odvija na aplikacijskoj razini OSI modela koristeći HTTP protokol prijenosa informacija na webu temeljen na izmjeni poruka zahtjev-odgovor. Sam poslužitelj pokreće web aplikaciju i prosljeđuje joj klijentske zahtjeve.

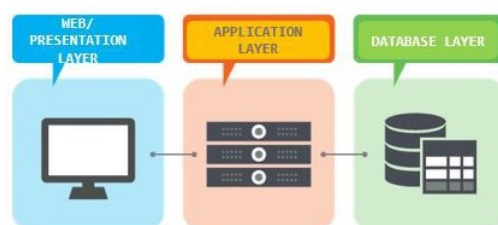
Web aplikacija obrađuje klijentske zahtjeve pri čemu prema potrebi pristupa bazi podataka te preko poslužitelja, klijentu vraća odgovor vidljiv u web pregledniku.

Prednosti odabrane arhitekture su te što se slojevi mogu oblikovati odvojeno te sve komponente mogu biti jednostavnije i razumljivije, ostvaruje se podjela brige (separation of concerns) time što svaki sloj brine o svojoj funkcionalnosti i ne miješa se u brige nekog drugog sloja, a njihova međuovisnost ostvaruje se komunikacijom putem sučelja čija se implementacija može prilagoditi u određenom sloju.

Za izradu aplikacije kao radni okvir na klijentskoj strani odabrali smo Vue.js zasnovan na JavaScriptu koji pruža učinkoviti razvoj klijentskog sučelja koristeći već dostupne web komponente.

Za radni okvir na poslužiteljskoj strani odabrali smo radni okvir Node.js Express koji pruža funkcionalnosti posredniče arhitekture (middleware) i usmjeravanja te Sequelize koji apstrahira pristup bazi podataka i omogućuje jednostavnije kreiranje upita bez obzira koja se baza koristi.

Članovi tima kod pišu u razvojnom okruženju Visual Studio Code, no to razvojno okruženje nije nužno i može se koristiti bilo kojim drugim uređivačem teksta. Jednostavno pokretanje, konfiguraciju i puštanje u pogon te neovisnost o računalu na kojem se kod izvršava omogućava korištenje platforme Docker. Kroz 3 Docker kontejnera pokriveni su svi ključni dijelovi/slojevi aplikacije - klijent, poslužitelj i baza podataka te su osigurane točne verzije svih vanjskih biblioteka i alata.



Slika 4.1: Slojevi arhitekture

- **MVC stil arhitekture**

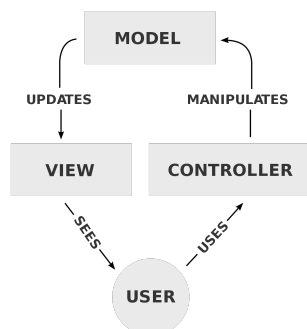
Arhitektura je detaljnije razrađena najsličnija stilu arhitekture MVC (Model-View-Controller).

Osnovna karakteristika ovog stila je nezavisan razvoj pojedinih dijelova aplikacije što omogućuje jednostavnije testiranje i razvijanje dijelova sustava te njihove dorade. Korisničko sučelje je odvojeno od ostatka sustava, a kohezija elemenata se postiže kroz tri sloja, jednog sloja na klijentskoj strani - pogled (View) te dva na poslužiteljskoj strani – nadglednik (Controller) i model (Model).

1. *Model* – predstavlja glavnu komponentu sustava koja sadrži dinamičke strukture podataka odnosno razrede koji opisuju domenu primjene te sadrže pravila i aplikacijsku logiku. Blisko je povezan s bazom podataka aplikacije.

2. *Pogled (View)* – komponenta koja sadrži niz drugih komponente koje služe za prikaz podataka modela i interakciju s korisnikom kroz grafičko sučelje.

3. *Nadglednik (Controller)* – komponenta koja upravlja korisničkim zahtjevima prema modelu i odgovorima modela natrag prema pogledima. Omogućuje poveznicu korisničke strane s poslužiteljskom.



Slika 4.2: MVC stil arhitekture

4.1 Baza podataka

Za potrebe našeg sustava koristit ćemo relacijsku bazu podataka koja je jednostavna za modeliranje naših potreba - upravljanje podacima o klubovima koji organiziraju određene događaje i korisnicima koji ih pohađaju. Sastavni elementi baze su relacije, odnosno tablice koje sadrže svoje atribute. Zadaća baze podataka je jednostavna pohrana, umetanje, izmjena i dohvat podataka za daljnju obradu. Baza podataka ove aplikacije sastoji se od sljedećih tablica:

- User
- Club
- Course
- UserCourse
- TrainerApplication
- Event
- Dance
- Location
- EventDance
- Lesson

4.1.1 Opis tablica

User Ovaj entitet opisuje korisnike i sadrži sve informacije o njima. Povezan je sa vezom *One-to-Many* sa *TrainerApplication* preko *trainerId*, *One-to-Many* sa *UserCo-*

urse preko *userId* i *One-to-Many* sa Club preko *ownerId*.

User		
id	int	jedinstveni identifikator korisnika
username	varchar	jedinstveno korisničko ime korisnika
firstName	varchar	ime korisnika
lastName	varchar	prezime korisnika
password	varchar	zaporka za prijavu korisnika
gender	enum	spol korisnika
dateOfBirth	date	datum rođenja korisnika
phone	varchar	jedinstveni kontakt broj korisnika
email	varchar	jedinstvena email adresa korisnika
role	enum	uloga korisnika
experienceDescription	varchar	opis korisnikovog iskustva u različitim plesovima
image	varchar	fotografija korisnika
refreshToken	varchar	korisnikov refresh token

Club Ovaj entitet opisuje klubove i sadrži sve informacije o njima. Povezan je sa vezom *One-to-Many* sa TrainerApplocation preko *id*, *One-to-Many* sa Event preko *id*, *One-to-Many* sa Course preko *id*, *Many-to-One* sa User preko *ownerId*, *Many-to-One* sa Location preko *locationId*.

Club		
id	int	jedinstveni identifikator treninga
ownerId	int	identifikator korisnika [ref: > korisnik.id]
name	varchar	ime kluba
phone	varchar	kontakt broj kluba
email	varchar	kontakt mail kluba
description	varchar	opis kluba

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Club		
approvalStatus	enum	status odobrenja kluba
locationId	int	identifikator lokacije

Course Ovaj entitet opisuje tečajeve i sadrži sve informacije o njima. Povezan je sa vezom *Many-to-One* sa User preko *trainerId*, *Many-to-One* sa Club preko *clubId*, *Many-to-One* sa Location preko *locationId*, *Many-to-One* sa Dance preko *danceId*, *One-to-Many* sa UserCourse preko *courseId* i *One-to-Many* sa Lesson preko *id*.

Course		
id	int	jedinstveni identifikator tečaja
locationId	int	identifikator lokacije [ref: > Lokacija.id]
trainerId	int	identifikator klijenta(trenera) [ref: > Korisnik.id]
name	varchar	ime tečaja
description	varchar	opis tečaja
image	varchar	url slike za opis tečaja
capacity	int	kapacitet polaznika
minAge	int	ograničenje za minimalni broj godina polaznika
maxAge	int	ograničenje za maksimalni broj godina
gender	enum	ograničenje za spol polaznika
applicationDeadline	datetime	datum za krajnji rok prijave
maxApplicants	int	maksimalan broj polaznika
additionalRules	varchar	dodatne informacije i pravila za tečaj
clubId	int	identifikator kluba koji organizira tečaj [ref: > Klub.id]
danceId	int	identifikator plesa koji će se plesati na tečaju [ref: > Ples.id]

UserCourse Ovaj entitet opisuje korisnike tečaja i sadrži sve potrebne informacije i reference o njima. Povezan je sa vezom *Many-to-One* sa User preko *userId* i *Many-to-One* sa Course preko *courseId*.

UserCourse		
id	int	jedinstveni identifikator
userId	int	identifikator korisnika [ref: > Korisnik.id]
courseId	int	identifikator tečaja [ref: > Tecaj.id]
status	enum	status je li korisnik primljen na tečaj

TrainerApplication Ovaj entitet opisuje korisnike koji su se prijavili za trenera i sadrži potrebne informacije o njima. Povezan je sa vezom *Many-to-One* sa User preko *trainerId* i *Many-to-One* sa Club preko *clubId*.

TrainerApplication		
id	int	jedinstveni identifikator trenera
trainerId	int	identifikator korisnika [ref: > Korisnik.id]
clubId	int	identifikator kluba [ref: > Klub.id]
motivationLetter	varchar	motivacijsko pismo
certificate	varchar	položeni certifikat za trenera
status	enum	status odobrenja trenera

Event Ovaj entitet opisuje plesnjake i sadrži sve informacije o njima. Povezan je sa vezom *Many-to-One* sa Location preko *locationId*, *Many-to-One* sa Club preko *clubId* i *One-to-Many* sa EventDance preko *id*.

Event		
id	int	jedinstveni identifikator plesnjaka
locationId	int	identifikator lokacije [ref: > Lokacija.id]
clubId	int	identifikator kluba [ref: > Klub.id]
name	varchar	naziv plesnjaka

Nastavljeno na idućoj stranici

Nastavljeno od prethodne stranice

Event		
description	varchar	opis plesnjaka
image	varchar	slika plesnjaka

Location Ovaj entitet opisuje lokaciju sa imenom i koordinatama pomoću kojih se prikazuje na karti. Povezan je sa vezom *One-to-Many* sa Event preko *locationId*, *One-to-Many* sa Course preko *courseId* i *One-to-Many* sa Club preko *locationId*.

Location		
id	int	jedinstveni identifikator lokacije
name	varchar	ime lokacije
coordinates	varchar	geografska širina i dužina lokacije na karti

EventDance Ovaj entitet opisuje identifikatore plesnjaka i identifikatore plesova koji se na njima plešu. Povezan je sa vezom *Many-to-One* sa Event preko *eventId* i *Many-to-One* sa Dance preko *danceId*.

EventDance		
id	int	jedinstveni identifikator koji povezuje plesnjake i plesove
eventId	int	identifikator plesnjaka [ref: > Plesnjak.id]
danceId	int	identifikator plesa [ref: > Ples.id]

Lesson Ovaj entitet definira treninge za pojedini tečaj i informacije o vremenu održavanja. Povezan je sa vezom *Many-to-One* sa Course preko *courseId*.

Lesson		
id	int	jedinstveni identifikator treninga
courseId	int	identifikator tečaja [ref: > Tecaj.id]
startTime	datetime	vrijeme početka treninga

Nastavljeno na idućoj stranici

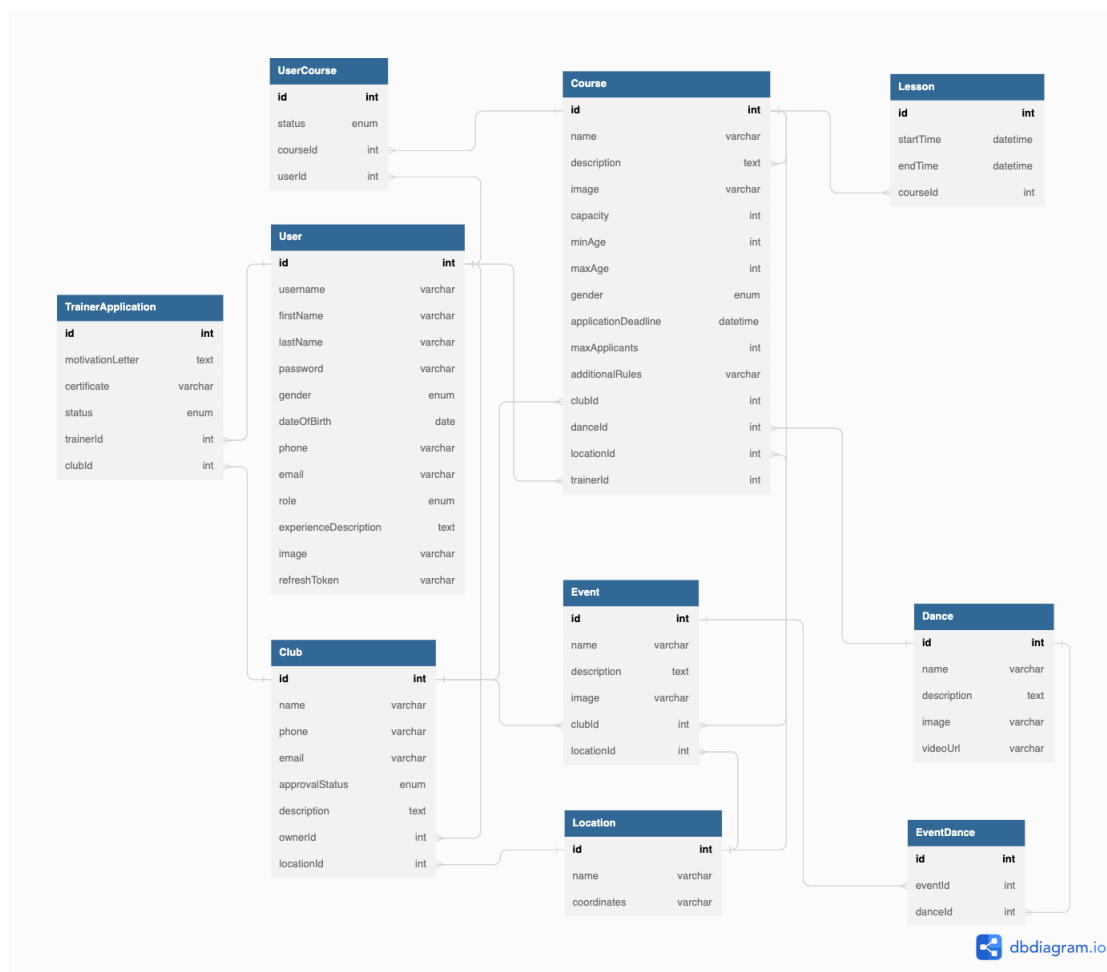
Nastavljeno od prethodne stranice

Lesson		
endTime	datetime	vrijeme kraja treninga

Dance Ovaj entitet opisuje plesove i sadrži informacije o plesovima. Povezan je sa vezom *One-to-Many* sa EventDance preko *danceId* i *One-to-Many* sa Course preko *danceId*.

Dance		
id	int	jedinstveni identifikator plesa
name	varchar	ime plesa
description	varchar	opis plesa
image	varchar	url slike koja opisuje ples
videoLink	varchar	url na video koji opisuje ples

4.1.2 Dijagram baze podataka

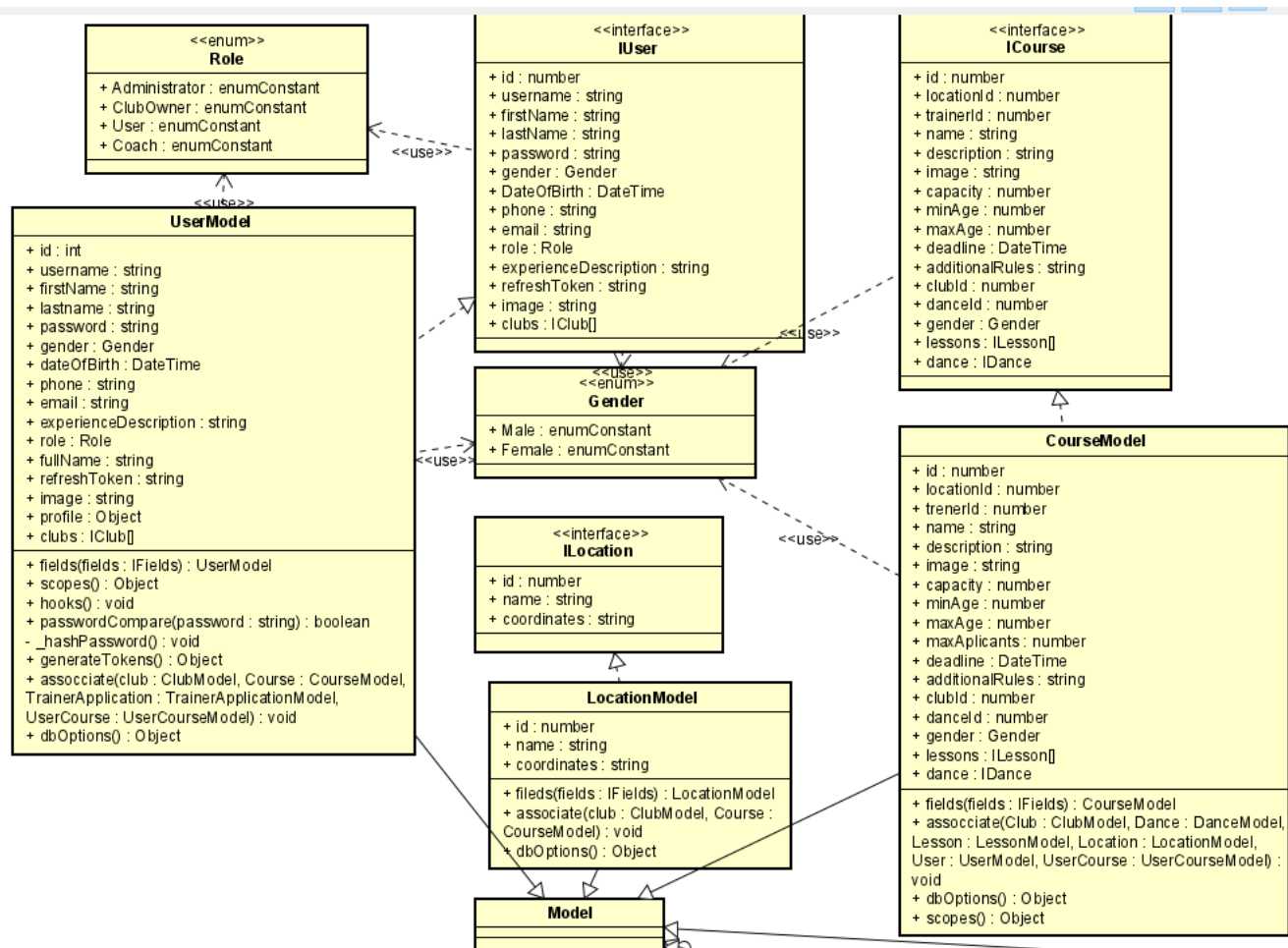


Slika 4.3: Relacijski dijagram baze podataka.

4.2 Dijagram razreda

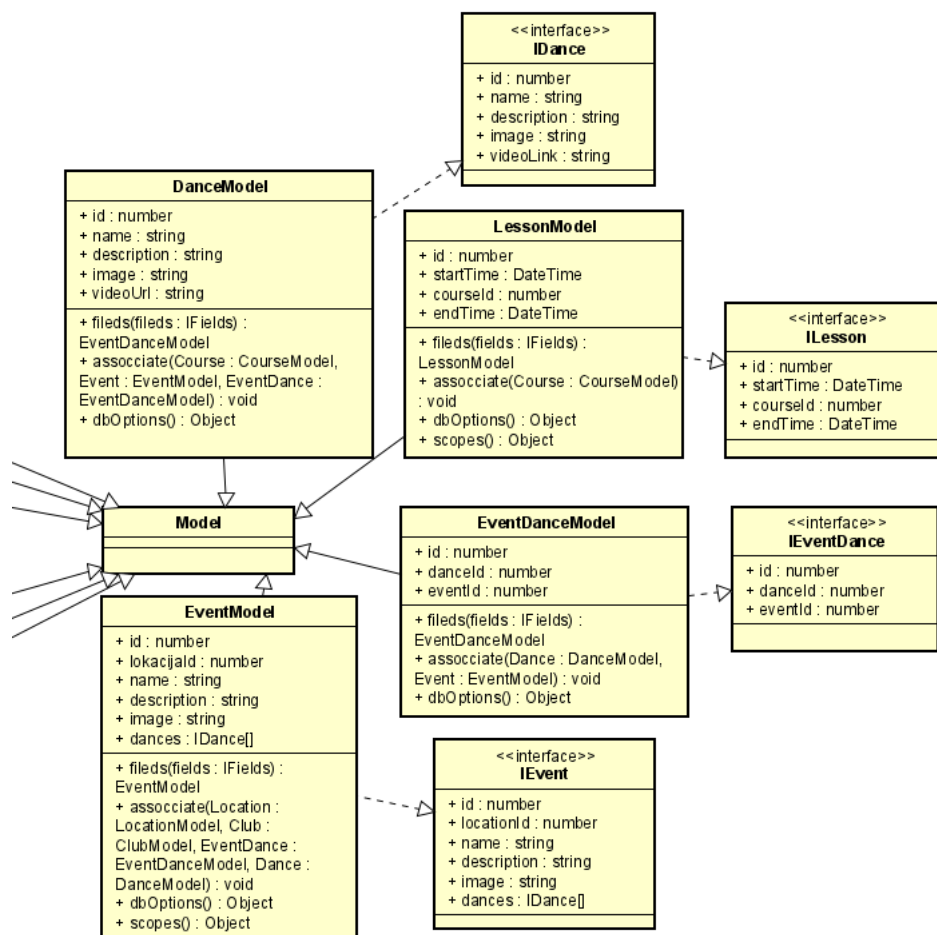
Na slikama su prikazani razredi koji pripadaju serverskoj strani aplikacije - konkretno modelima i sučeljima koje oni nasljeđuju. Controlleri nisu definirani kao klase nego samo sadrže funkcije za upravljanje, stoga nisu prikazani na klasnom dijagramu. Svi modeli nasljeđuju klasu Model. Model razredi preslikavaju strukturu baze podataka u aplikaciji. Zbog lakše organizacije, dijagram je podijeljen u tri skupine, razredi su podijeljeni logički po dijelovima koji međusobno čine logičku cjelinu ili nasljeđuju i implementiraju zajedničke enumeracije ili sučelja. Iz naziva i tipova atributa u razredima može se zaključiti vrsta ovisnosti među različitim razredima.

Dijagram koji slijedi prikazuje modele User i Location te model CourseModel. Model User implementira sučelje IUser, koje definira njegove attribute, i koristi enumeraciju Role i Gender. Role definira ulogu korisnika, a Gender spol korisnika. Model Course implementira sučelje ICourse i korisni enumeraciju Gender za definiranje spola kojem je tečaj namijenjen, ako je za tečaj definirano ograničenje po spolu.



Slika 4.4: Dio klasnog dijagrama vezan uz modele User, Course i Location.

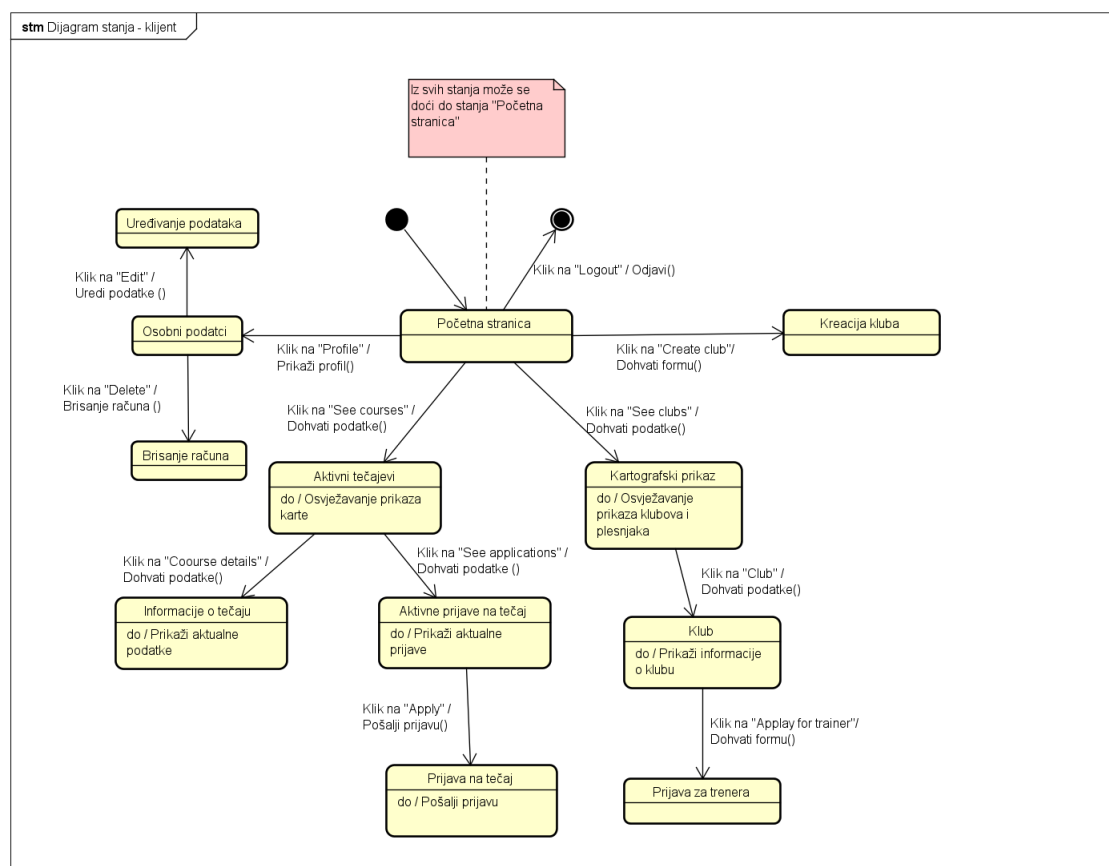
Dijagram koji slijedi prikazuje modele Club, TrainerApplication i UserCourse. Model Club implemetnira sučelje IClub koji definira njegove attribute te korisni enumeraciju ApprovalStatus koja opisuje je li administrator prihvatio tog kluba u sustavu ili ne. Klasa HttpError implementira sučelje IHttpError kojom lovimo moguće iznimke nastale u kodu.



Slika 4.6: Dio klasnog dijagrama vezan uz modele Dance, Event, EventDance i Lesson.

4.3 Dijagram stanja

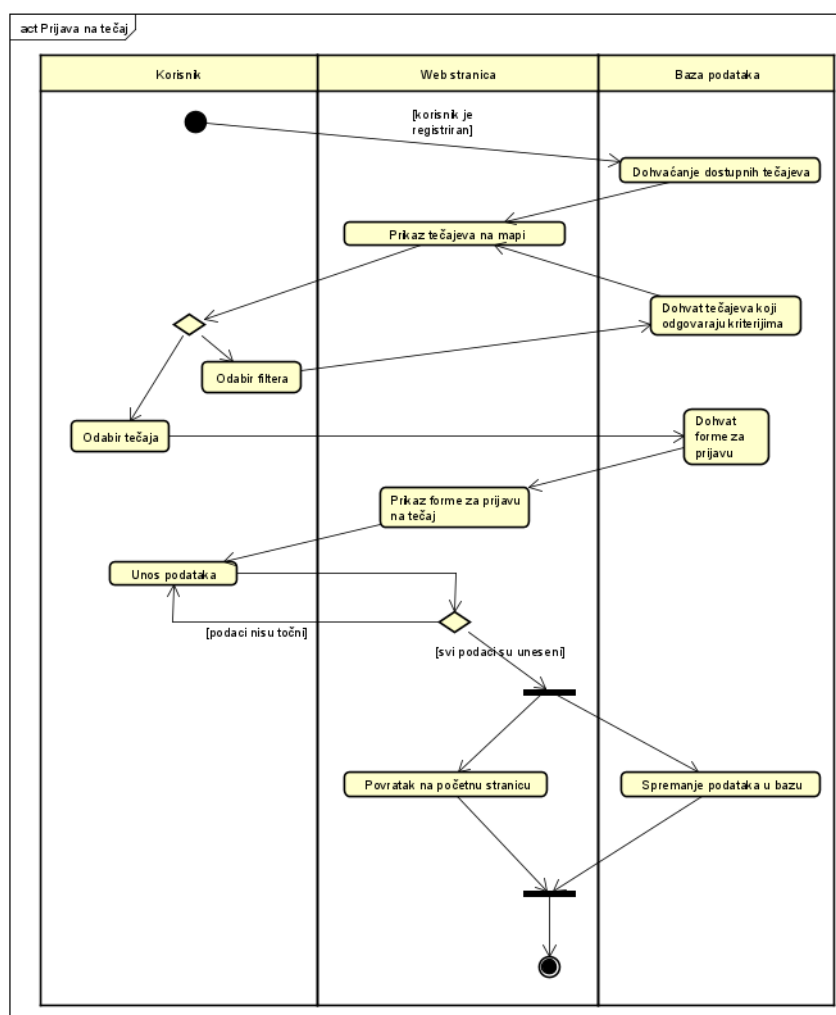
Dijagram stanja prikazuje stanja objekta te prijelaze iz jednog stanja u drugo temeljene na događajima. Na slici je prikazan dijagram stanja za registriranog korisnika. Nakon prijave, klijentu se prikazuje početna stranica na kojoj može odabrati prikaz karte sa tečajevima slobodnima za upis. Za odabrani tečaj ima mogućnost pregleda relevantnih informacija te aktivnih prijava na tečaj. Kod pregleda aktivnih prijava ima opciju osobne prijave na tečaj. Također, klijent na početnoj stranici može odabrati opciju prikaza klubova i plesnjaka. Ima mogućnost pregledati informacije o postojećim klubovima te poslati prijavu za trenera u odabranom klubu. Klijent može odabrati kreaciju novog kluba. Odabirom „Osobni podaci“ klijent može vidjeti svoje osobne podatke te ih može urediti ili odabrati brisanje svojeg računa.



Slika 4.7: Dijagram stanja - klijent

4.4 Dijagram aktivnosti

Dijagram aktivnosti pokazuje proces prijave registriranog korisnika na tečaj. Prijavom u sustav njemu se prikazuje mapa sa listom tečajeva koji su slobodni za upis. Korištenjem filtera on može prilagoditi rezultate svojim preferencijama i pronaći tečaj koji njemu odgovara. Odabirom tečaja stranica ga odvodi na prikaz detalja istoga. Klikom na prijavu korisnik mora upisati tražene podatke i tada se njegova prijava šalje u bazu podataka, a stranica ga vraća na kartu sa prikazom svih tečajeva.

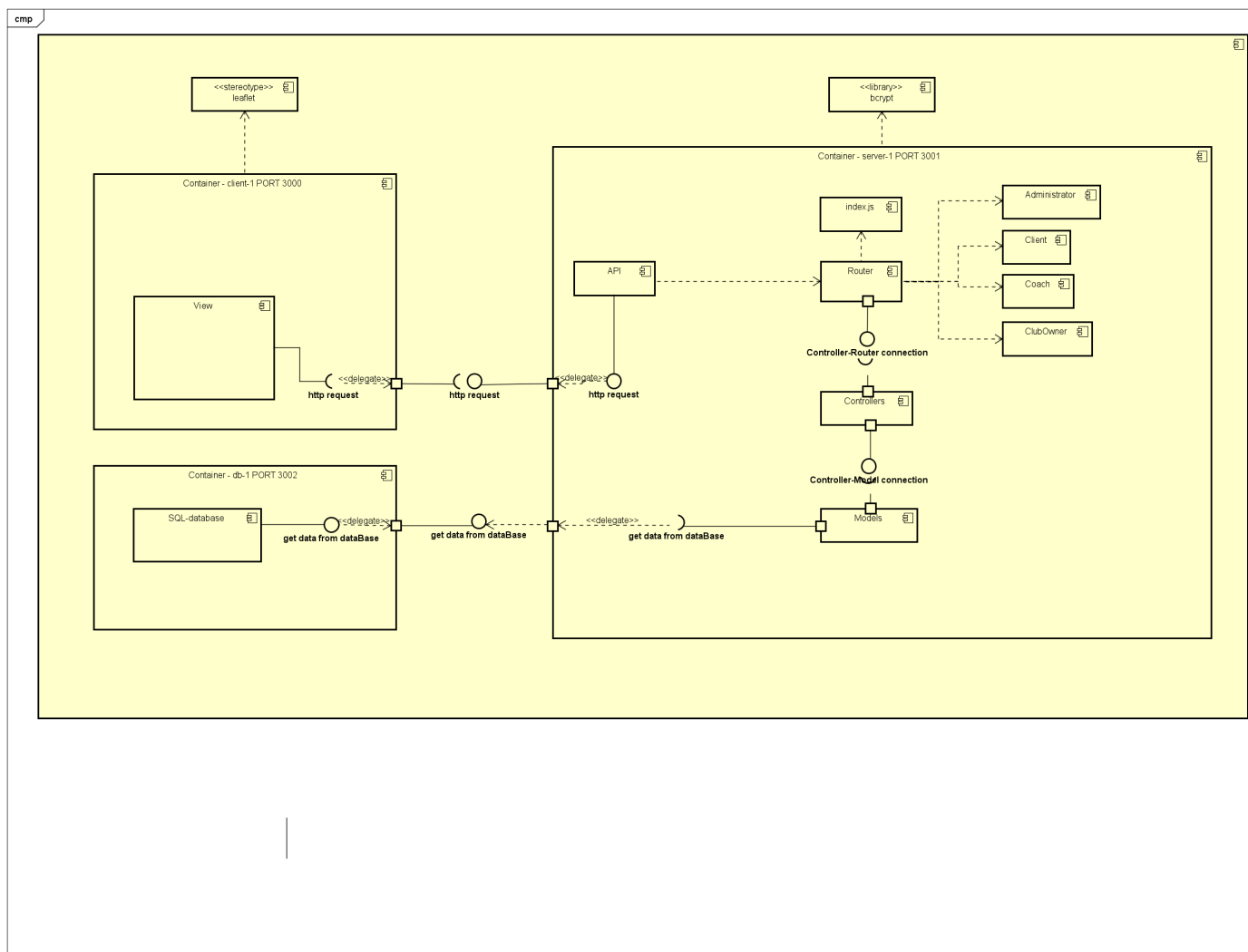


Slika 4.8: Dijagram aktivnosti - prijava na tečaj

4.5 Dijagram komponenti

Dijagram komponenti opisuje organizaciju i međuovisnost komponenti, interne strukture i odnose prema okolini. Postoje 3 Docker kontejnera: jedan za bazu po-

dataka (port 3002), jedan za poslužiteljsku stranu (port 3001) i jedan za klijentsku stranu (port 3000). Klijentska strana podigne jedan html dokument i podigne virtualni DOM. View napravi virtualni DOM, provjerava podatke i pošalje na fizički DOM. View automatski puni fizički DOM s .json file-om. Nema slanja html dokumenta između klijentske i poslužiteljske strane. Klijentska strana preko http requestova traži od poslužiteljske strane informacije iz baze podataka. Postoji 10 „Controllera“ (za klub, tecaj, ples, događaj, događajPles, lekciju, lokaciju, trenerAplikaciju, korisnika i korisnikaTecaja) i njihovih 10 „Modela“. Svaki entitet je model. Modeli su zadužni za dobavljanje podataka iz baze podataka. „Controller“ se nalazi na poslužiteljskoj strani i prima informaciju od „View“ komponente. On validira primljenu informaciju iz zahtjeva s klijentske strane, obavi sve što je potrebno za taj zadatak napraviti te vrati povratnu informaciju klijentu je li akcija uspješno obavljena. JavaScript „leaflet“ biblioteka služi za prikaz interaktivnih mapa „Bycrypt“ je biblioteka koja služi za kriptiranje lozinke.



Slika 4.9: Dijagram komponenti

5. Implementacija i korisničko sučelje

5.1 Korištene tehnologije i alati

Komunikacija u timu realizirana je korištenjem aplikacije Slack¹ koja omogućuje jednostavnu i veoma organiziranu komunikaciju kroz „channel“ i „thread“ mogućnosti komuniciranja. Za izradu UML dijagrama korišten je alat Astah UML², a kao sustav za upravljanje izvornim kodom Git³. Udaljeni repozitorij projekta dostupan je na web platformi GitLab⁴.

Kao razvojno okruženje korišten je Visual Studio Code⁵ – integrirano razvojno okruženje (IDE) tvrtke Microsoft⁶ koje dolazi sa ugrađenom podrškom za JavaScript⁷, TypeScript⁸ i Node.js⁹ te pruža IntelliSense¹⁰ (inteligentno dovršavanje koda obzirom na kontekst), podršku za otklanjanje pogrešaka, refaktoriranje koda, ugrađene naredbe za Git i mnogobrojna proširenja za ostale programske jezike i alate.

Klijentska strana aplikacije (*frontend*) napisana je koristeći radni okvir Vue.js¹¹ zasnovan na JavaScriptu odnosno TypeScriptu (JavaScript s podrškom za tipove) koji pruža učinkoviti razvoj klijentskog sučelja koristeći već dostupne web komponente i predloške.

Poslužiteljska strana aplikacije (*backend*) napisana je u programskom jeziku TypeScript koristeći radni okvir Express¹² koji pojednostavljuje proces izgradnje i razvoja web aplikacija koristeći Node.js. Express pruža funkcionalnosti posredničke arhitekture (*middleware*) i usmjeravanja, jedan je od najpopularnijih okvira za Node.js te se široko koristi za izgradnju malih i velikih web aplikacija. Također, korištena

¹<https://slack.com/>

²<https://astah.net/products/astah-uml/>

³<https://git-scm.com/>

⁴<https://about.gitlab.com/>

⁵<https://code.visualstudio.com/>

⁶<https://www.microsoft.com/hr-hr/>

⁷<https://www.javascript.com/>

⁸<https://www.microsoft.com/hr-hr/>

⁹<https://nodejs.org/en/>

¹⁰<https://code.visualstudio.com/docs/editor/intellisense>

¹¹<https://vuejs.org/>

¹²<https://expressjs.com/>

je i ORM (*Object-Relational Mapping*) biblioteka Sequelize¹³ koja omogućuje interakciju s relacijskim bazama podataka koristeći objektno orijentiranu sintaksu, umjesto izravnih SQL upita.

Jednostavno pokretanje, konfiguraciju i puštanje u pogon te neovisnost o računalu na kojem se kod izvršava omogućava korištenje platforme Docker¹⁴. Kroz tri Docker kontejnera pokriveni su svi ključni dijelovi/slojevi aplikacije – klijent, poslužitelj i PostgreSQL¹⁵ baza podataka te su osigurane točne verzije svih vanjskih biblioteka i alata.

¹³<https://sequelize.org/>

¹⁴<https://www.docker.com/>

¹⁵<https://www.postgresql.org/>

5.2 Ispitivanje programskog rješenja

5.2.1 Ispitivanje komponenti

Unit testovima provjeravamo ispravnost funkcionalnosti „utils“ funkcija koje su korištene kao pomoćne funkcije na više različitih dijelova koda. Proveli smo sedam unit testova te u nastavku slijede njihovi kratki opisi i odgovarajući kodovi, na kraju slijedi slika izvođenja rezultata testova.

Ispitni slučaj 1. : Unos e-mail adrese

Ulaz:

1. Unos ispravne e-mail adrese.
2. Unos adrese bez domene.
3. Unos domene bez username-a.
4. U upisanome e-mailu nedostaje '@'.
5. Nedostaje točka koja razdvaja domen i ekstenziju.
6. Nedostaje ekstenzija.
7. Ekstenzija je predugačka.

Očekivani rezultat:

1. Provjera je prošla i e-mail adresa je validna.
- 2.-7. Sustav javlja grešku kako je i predviđeno kao cilj ovog testa.

Rezultat:

Svi testovi su zadovoljeni, aplikacija je prošla test!

```
test('Should validate email address', t => {
  const validInput = 'some_username@gmail.com';
  const onlyName = 'some_username';
  const missingName = '@gmail.com';
  const missingAt = 'some_username@gmail.com';
  const missingDot = 'some_username@gmailcom';
  const missingEnd = 'some_username@gmail.';
  const longEnd = 'some_username@gmail.comercial';

  const error = Error(WRONG_EMAIL_FORMAT_MESSAGE);

  t.true(emailValidator(null, validInput));
  t.deepEqual(emailValidator(null, onlyName), error);
  t.deepEqual(emailValidator(null, missingName), error);
  t.deepEqual(emailValidator(null, missingAt), error);
  t.deepEqual(emailValidator(null, missingDot), error);
  t.deepEqual(emailValidator(null, missingEnd), error);
  t.deepEqual(emailValidator(null, longEnd), error);
});
```

Slika 5.1: Test unosa e-mail adrese

Ispitni slučaj 2. : Funkcija „subtractYears“ – funkcije koja oduzima zadan broj godina od nekog određenog datuma

U svrhu tog testiranja definiramo novi datum od kojeg ćemo oduzeti zadan broj godina, definiramo očekivani datum te u varijablu spremamo rezultat funkcije. Naredbom „deepEqual“ provjeravamo jesu li rezultat funkcije i očekivana vrijednost jednaki.

Ulaz:

1. Slobodno izabran datum, očekivani rezultat funkcije, broj godina koje oduzimamo

Očekivani rezultat:

1. Slobodno izabran datum umanjen za zadani broj godina, provjera je prošla

Rezultat:

Svi testovi su zadovoljeni, aplikacija je prošla test!

Ispitni slučaj: Funkcija „toDatePicker“ koja zadano vrijeme vraća u milisekundama

U svrhu tog testiranja definiramo dva datuma čija je vremenska razlika u milisekundama jednaka poznatoj vrijednosti, te provjeravamo rezultat vremenske razlike dvaju datuma pretvorenih funkcijom „toDatePicker“ u milisekunde, ako rezultat funkcije „is“ bude potvrđan, funkcija „toDatePicker“ radi kako je očekivano.

Ulaz:

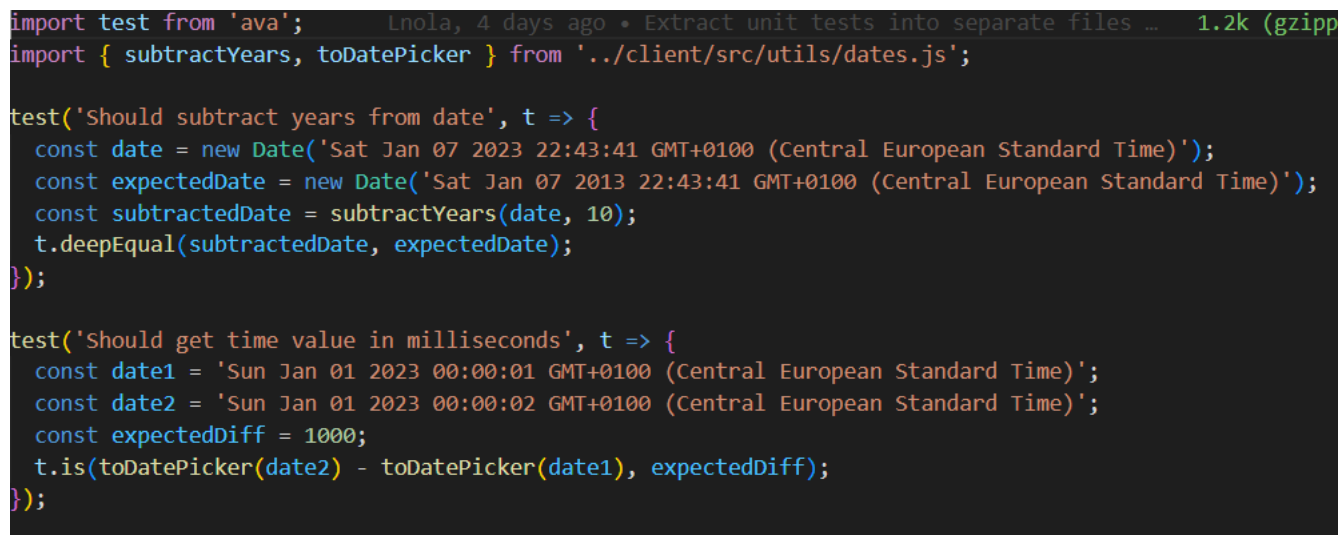
1. Dva datuma čija je razlika poznat broj milisekundi, poznat broj milisekundi

Očekivani rezultat:

1. Rezultat razlike poziva funkcije koja datume pretvara u milisekunde, provjera je prošla

Rezultat:

Svi testovi su zadovoljeni, aplikacija je prošla test!

A screenshot of a code editor with a dark background. It shows JavaScript code for unit testing. At the top, there's a header bar with a search icon, the text 'Lnola, 4 days ago • Extract unit tests into separate files ...', and a file size '1.2k (gzip)'. The code starts with two import statements: 'import test from 'ava';' and 'import { subtractYears, toDatePicker } from '../client/src/utils/dates.js';'. There are two test functions. The first is 'test('Should subtract years from date', t => {' followed by a block of code that creates a date 'Sat Jan 07 2023 22:43:41 GMT+0100 (Central European Standard Time)', subtracts 10 years using 'subtractYears', and checks if the result is equal to 'Sat Jan 07 2013 22:43:41 GMT+0100 (Central European Standard Time)' using 't.deepEqual'. The second test is 'test('Should get time value in milliseconds', t => {' followed by code that creates two dates one second apart ('Sun Jan 01 2023 00:00:01' and 'Sun Jan 01 2023 00:00:02'), calculates the difference using 'toDatePicker', and checks if it's equal to 1000 using 't.is'.

Slika 5.2: Test unosa e-mail adrese

Ispitni slučaj: provjera zadovoljenosti dobne granice***Ulaz:***

1. Korisnik unosi današnji datum u mjesto za unos datuma rođenja
2. Korisnik unosi datum točno 15 godina stariji od današnjeg
3. Korisnik unosi datum točno 12 godina stariji od današnjeg

Očekivani rezultat:

1. Sustav javlja error - uneseni datum rođenja nije dozvoljen!
2. i 3. Sustav ne javlja error - uneseni datum rođenja je dozvoljen!

Rezultat:

Svi testovi su zadovoljeni, aplikacija je prošla test!

```
test('Should check if user is old enough to register', t => {
  const now = new Date().getTime();
  const validDate = subtractYears(new Date(), 15).getTime();
  const edgeCase = subtractYears(new Date(), 12).getTime();

  const error = Error(UNDER_AGE_MINIMUM_MESSAGE);

  t.deepEqual(dateOfBirthValidator(null, now), error);
  t.true(dateOfBirthValidator(null, validDate));
  t.true(dateOfBirthValidator(null, edgeCase));
});
```

Slika 5.3: Test zadovoljenosti dobne granice

Ispitni slučaj: provjera ispravnosti broja mobitela***Ulaz:***

1. Korisnik u odgovarajuće mjesto upisuje broj mobitela sa 9 znamenki
2. Korisnik u odgovarajuće mjesto upisuje broj mobitela sa 10 znamenki
3. Korisnik u odgovarajuće mjesto upisuje broj mobitela sa 9 znamenki čije su znamenke odvojene u grupe po 3 koje su razdvojene razmakom
4. Korisnik u odgovarajuće mjesto upisuje broj mobitela sa 10 znamenki čije su znamenke odvojene u dvije grupe po 3 i jednu grupu po 4 znamenke te su grupe razdvojene znakom
5. Korisnik u odgovarajuće mjesto upisuje broj mobitela sa više od 10 znamenki
6. Korisnik u odgovarajuće mjesto upisuje broj mobitela sa manje od 9 znamenki
7. Korisnik u odgovarajuće mjesto upisuje broj mobitela koji sadrži slovo
8. Korisnik u odgovarajuće mjesto upisuje broj mobitela koji sadrži znak

Očekivani rezultat:

1. - 4. Sustav ne javlja error - broj mobitela je validan!
5. - 8. Sustav javlja error - broj mobitela nije validan!
6. Sustav javlja error - broj mobitela nije validan!

Rezultat:

Svi testovi su zadovoljeni, aplikacija je prošla test!


```
test('Should validate phone number', t => {
  const nineLength = '123456789';
  const tenLength = '1234567890';
  const spaces = '123 123 123';
  const dashes = '123-123-1234';
  const tooLong = '12345678910';
  const tooShort = '1234';
  const alpha = '1234b1234';
  const nonAlphaNumeric = '123#123#12';

  const error = Error(WRONG_PHONE_FORMAT_MESSAGE);

  t.true(phoneNumberValidator(null, nineLength));
  t.true(phoneNumberValidator(null, tenLength));
  t.true(phoneNumberValidator(null, spaces));
  t.true(phoneNumberValidator(null, dashes));
  t.deepEqual(phoneNumberValidator(null, tooLong), error);
  t.deepEqual(phoneNumberValidator(null, tooShort), error);
  t.deepEqual(phoneNumberValidator(null, alpha), error);
  t.deepEqual(phoneNumberValidator(null, nonAlphaNumeric), error);
});
```

Slika 5.4: Test ispravnosti broja mobitela

Ispitni slučaj: provjera valjanosti poslanih url adresa

Testiramo kako će aplikacija reagirati kada joj šaljemo različite url destinacije, odnosno testovima definiramo ispravne destinacije i pogrešne destinacije u kojima je učinjena pogreška ili fali neki logički dio url adrese. U prva dva slučaja kada je ulaz ispravan test je ispravan, a u ostalim slučajevima testni slučaj ne prolazi.

Ulaz:

1. i 2. Unos ispravne url adrese
3. Unos url adrese bez protokola
4. unos url adrese bez dvotočke
5. unos url adrese bez /
6. unos rla adrese bez //
7. unos url adrese bez .com domene

Očekivani rezultat:

1. i 2. Sustav ne javlja error - url adresa je ispravna
- 3.-7. Sustav javlja error - url adresa je nesipravna

Rezultat:

Svi testovi su zadovoljeni, aplikacija je prošla test!

```
test('Should validate url', t => {
  const https = 'https://gitlab.com/programtvogkompjuter1/ples/';
  const http = 'http://gitlab.com/programtvogkompjuter1/ples/';
  const noProtocol = '://gitlab.com/programtvogkompjuter1/ples/';
  const missingDot = 'https://gitlab.com/programtvogkompjuter1/ples/';
  const missingSlash = 'https://gitlab.com/programtvogkompjuter1/ples/';
  const missingSlashes = 'https:gitlab.com/programtvogkompjuter1/ples/';
  const missingTopLevelDomain = 'https://gitlab/programtvogkompjuter1/ples/';

  const error = Error(WRONG_URL_FORMAT_MESSAGE);

  t.true(urlValidator(null, https));
  t.true(urlValidator(null, http));
  t.deepEqual(urlValidator(null, noProtocol), error);
  t.deepEqual(urlValidator(null, missingDot), error);
  t.deepEqual(urlValidator(null, missingSlash), error);
  t.deepEqual(urlValidator(null, missingSlashes), error);
  t.deepEqual(urlValidator(null, missingTopLevelDomain), error);
});
```

Slika 5.5: Test valjanosti url adresa

Pokretanje testova se vrši otvaranjem terminala unutar *source code* mape u repozitoriju te pokretanjem naredbe *npm run test* te je rezultat izvođenja kako je prikazano na slici:

```
> source_code@1.0.0 test
> ava

✓ emailSuggestions › Should return email suggestions
✓ dates › Should subtract years from date (127ms)
✓ dates › Should get time value in milliseconds
✓ rules › Should validate email address (131ms)
✓ rules › Should check if user is old enough to register
✓ rules › Should validate phone number
✓ rules › Should validate url
-

7 tests passed
```

Slika 5.6: Izvođenje testa valjanosti

5.2.2 Ispitivanje sustava

Ispitni slučaj 1: provjera ispravnosti logina

Ulaz:

1. Korisnik se ulogira u sustav kao user
2. Korisnik se ulogira u sustav kao trainer

3. Korisnik se ulogira u sustav kao club owner
4. Korisnik se ulogira u sustav kao admin
5. Korisnik, pri pokušaju logina, ostavi mjesto za username prazno

Očekivani rezultat:

1. Sustav korisnika prebacuje na početnu stranicu - login je prošao uspješno
2. Sustav korisnika prebacuje na početnu stranicu - login je prošao uspješno
3. Sustav korisnika prebacuje na početnu stranicu - login je prošao uspješno
4. Sustav korisnika prebacuje na početnu stranicu - login je prošao uspješno
5. Sustav javlja error i vraća korisnika na stranicu /auth

```
const userUsername = 'user';
const trainerUsername = 'trainer';
const clubOwnerUsername = 'owner';
const adminUsername = 'admin';

describe('Login Test', () => {
  before(() => {
    cy.task('seed');
  });

  beforeEach(() => {
    cy.visit('/');
    cy.contains('Login').click();
  });

  it('User should be able to log in', () => {
    cy.login(userUsername);
    cy.url().should('be.equal', Cypress.config('baseUrl'));
  });

  it('Trainer should be able to log in', () => {
    cy.login(trainerUsername);
    cy.url().should('be.equal', Cypress.config('baseUrl'));
  });

  it('Club Owner should be able to log in', () => {
    cy.login(clubOwnerUsername);
    cy.url().should('be.equal', Cypress.config('baseUrl'));
  });

  it('Admin should be able to log in', () => {
    cy.login(adminUsername);
    cy.url().should('be.equal', Cypress.config('baseUrl'));
  });

  it('User with username field missing should not be able to log in', () => {
    cy.findByAriaLabel('password').type('123');
    cy.contains('Login').click();
    const caught = {
      message: null,
    };
    Cypress.on('uncaught:exception', (err, runnable, promise) => {
      caught.message = err.message;
      if (promise) return false;
    });
    cy.wrap(caught).should(c => {
      expect(c.message).to.include('unhandled promise rejection');
    });
    cy.url().should('be.equal', 'http://localhost:3000/auth');
  });
});
```

Slika 5.7: Login sitemski test

Rezultat:

Svi testovi su zadovoljeni, aplikacija je prošla test!

Ispitni slučaj 2: provjera ispravnosti registracije

Ulaz:

1. Korisnik se pokušava registrirati u sustav sa ispravnim podacima
2. Korisnik se pokušava registrirati u sustav sa pogrešno unesenim mailom

Očekivani rezultat:

1. Sustav korisnika prebacuje na početnu stranicu - registracija je prošla uspješno!
2. Sustav javlja error i vraća korisnika na stranicu /auth

Rezultat:

Svi testovi su zadovoljeni, aplikacija je prošla test!

```
const newUser = {
  username: 'Username',
  firstName: 'Username',
  lastName: 'LastName',
  email: 'username@gmail.com',
  password: '12345',
  dateOfBirth: '2003-01-06',
  phoneNumber: '0911111111',
  experienceDescription: 'Some experience',
};

const invalidMail = {
  username: 'invalidMail',
  firstName: 'Username',
  lastName: 'LastName',
  email: 'username@gmail.com',
  password: '12345',
  dateOfBirth: '2003-01-06',
  phoneNumber: '0911111111',
  experienceDescription: 'Some experience',
};

describe('Register Test', () => {
  before(() => {
    cy.task('seed');
  });

  beforeEach(() => {
    cy.visit('/auth');
    cy.contains('Click here to register').click();
  });

  it('Unregistered user should be able to register', () => {
    cy.register(newUser);
    cy.url().should('be.equal', Cypress.config('baseUrl'));
  });

  it('Unregistered user with invalid email should not be able to register', () => {
    cy.register(invalidMail);

    const caught = {
      message: null,
    };

    Cypress.on('uncaught:exception', (err, runnable, promise) => {
      caught.message = err.message;
      if (promise) return false;
    });

    cy.wrap(caught).should(c => {
      expect(c.message).to.include('unhandled promise rejection');
    });

    cy.url().should('be.equal', 'http://localhost:3000/auth');
  });
});
```

Slika 5.8: Registracija - sitemski test

Ispitni slučaj 3: provjera ispravnosti kreacije tečaja**Ulaz:**

1. Vlasnik kluba pokušava kreirati tečaj sa ispravnim podacima

2. Vlasnik kluba pokušava kreirati tečaj sa krajnjim datumom krivog fromata

Očekivani rezultat:

1. Sustav korisnika prebacuje na početnu stranicu - registracija je prošla uspješno!
2. Sustav javlja error i vraća korisnika na stranicu za unos podataka

Rezultat:

Svi testovi su zadovoljeni, aplikacija je prošla test!

```
const username = 'mateja.golec';
const name = 'Test course';
const description = 'Test course description';
const capacity = 24;
const applicationDeadline = '2023-12-12';
const locationName = 'NSB';
const coordinates = '45.815399, 15.966568';
const dance = 'Tango';
const trainer = 'Lucija Domić';
const invalidDate = '12-12';

describe('Create Course Test', () => {
  before(() => {
    cy.task('seed');
  });

  beforeEach(() => {
    cy.visit('/auth');
    cy.login(username);
    cy.contains('Courses').click();
    cy.contains('Create course').click();
  });

  it('Club owner should create a course', () => {
    cy.createCourse({
      name,
      description,
      capacity,
      applicationDeadline,
      locationName,
      coordinates,
      dance,
      trainer,
    });
  });

  it('Club owner should fail creating a course', () => {
    cy.createCourse({
      name,
      description,
      capacity,
      applicationDeadline: invalidDate,
      locationName,
      coordinates,
      dance,
      trainer,
    });

    const caught = {
      message: null,
    };

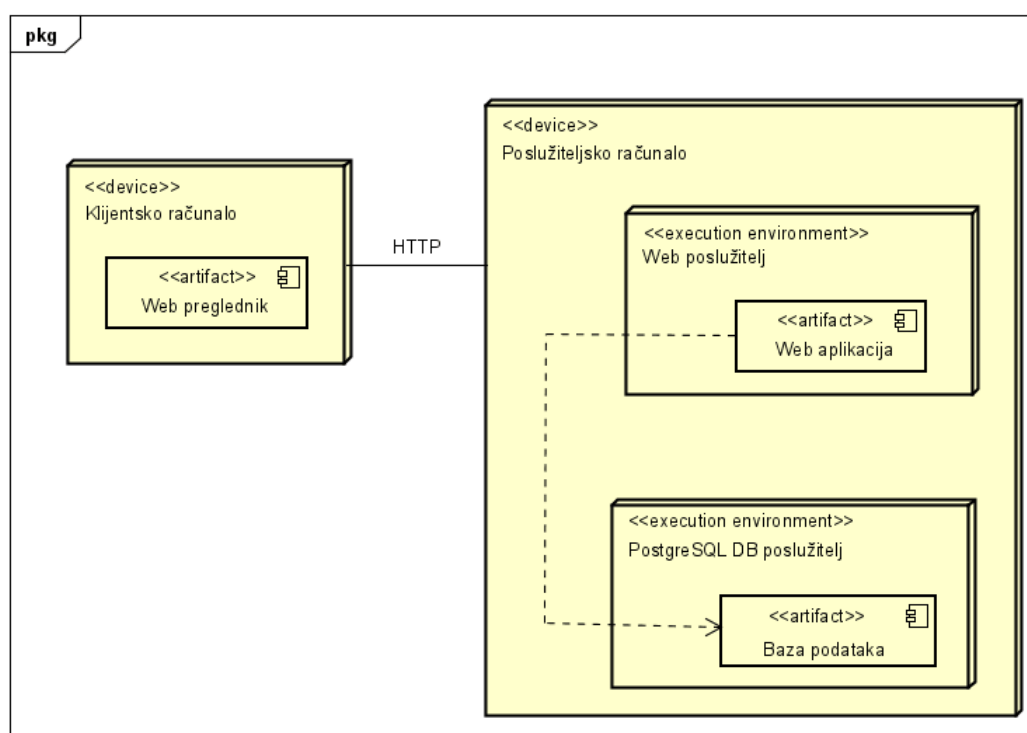
    Cypress.on('uncaught:exception', (err, runnable, promise) => {
      caught.message = err.message;
      if (promise) return false;
    });

    cy.wrap(caught).should(c => {
      expect(c.message).to.include('unhandled promise rejection');
    });
  });
});
```

Slika 5.9: Kreacija tečaja - sitemski test

5.3 Dijagram razmještaja

Na poslužiteljskom računalu nalaze se web poslužitelj i poslužitelj baze podataka. Web poslužitelj ovisi o poslužitelju baze podataka, što je na dijagramu jasno prikazano odgovarajućom strelicom. Klijenti pristupaju web aplikaciji preko web preglednika. Komunikacija između poslužitelja i klijenata se odvija preko HTTP veze.



Slika 5.10: Dijagram razmještaja

5.4 Setup projekta

Koraci za setup cijelog projekta su:

1. Otvoriti terminal u folderu u kojem želimo klonirati projekt. U terminalu pokrenuti naredbu „git config –global core.autocrlf input”
2. Klonirati projekt u navedeni folder
3. Pozicionirati se u source_code u našem projektu. Tamo otvoriti terminal i pokrenuti naredbu docker compose up –build
4. Daljne detaljne upute se nalaze u README.md datoteci

Potrebno je instalirati docker i dodati lokalne varijable (.env dokument u „root” projekta slijedeći primjer u .env.example)

Pokrenuti docker compose up u terminalu iz „source code”-a.

Upute za razvojno okruženje:

Preduvjeti – instalirana zadnja verzija Node.js-a

- instalirana zadnja verzija Node.js-a
- instaliran Docker

Pozicionirati se u:

- source code i pokrenuti npm install
- server i pokrenuti npm install
- client i pokrenuti npm install
- source code i docker compose up

Za instalirat/deinstalirati paket:

- pozicionirati se u server/client
- pokrenuti npm install package-name ili npm uninstall package-name

Upute za pokretanje migracija i sidova:

- pozicionirati se u server
- pokrenuti npm run db:migration:up
- potom npm run db:seed

Ekstenzije na projektu su slijedeće:

- Prettier
- Editor config
- Vue Language Feature (Volar)

Pokretanje projekta bez instaliranog Dockera nije moguće!

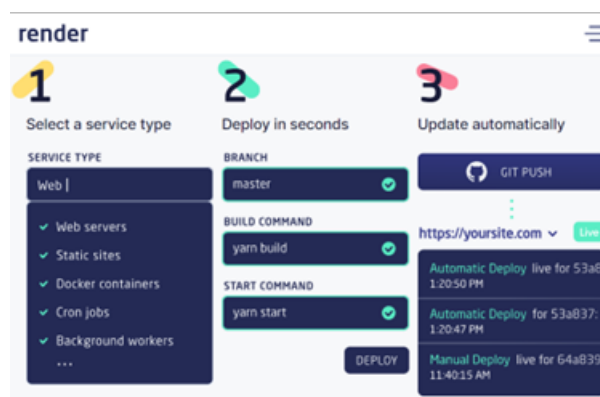
Kako bi projekt radio u datoteku server/src/shared/database/config/config.json treba zamijeniti postojeće i dodati slijedeće :

```
"development":  
"username": "ples",  
"password": "ples",  
"database": "ples",  
"host": "localhost",  
"port": 3002,  
"dialect": "postgres"
```


5.5 Upute za puštanje u pogon

Izvorni kod svih komponenti aplikacije potrebno je postaviti na „GitLab“ platformu jer će se koristiti pri izgradnji kontejnera potrebnih za puštanje aplikacije u pogon. Izvorni kod naše aplikacije dostupan je na adresi <https://gitlab.com/programtvogkomputera1/ples>.

Deployment izvodimo koristeći „Render“ platformu u oblaku (platform-as-a-service provider) koja nudi infrastrukturu puštanja aplikacije u pogon u kodu.



Slika 5.11: Render platforma

Prije prijave na „Render“ platformu i kreiranja poveznice prema našem repozitoriju, u korijenskoj mapi izvornog koda potrebno je kreirati „render.yaml“ datoteku. Ta datoteka predstavlja „Blueprint spec“ - konfiguraciju potrebnu za izvođenje i puštanje aplikacije u pogon na „Render“ platformi.

Unutar datoteke potrebno je definirati tri servisa koji predstavljaju komponente aplikacije – „server“, „client“ i „db“. Svaki servis, osim servisa vezanog u „PostgreSQL“ bazu, mora sadržavati svojstva „type“, „name“ i „env“.

Servis vezan uz „PostgreSQL“ bazu mora sadržavati samo svojstvo „name“, a informacije za njegovo izvođenje dostupne su kroz već gotovi i dostupan servis „Render“ platforme.

```
databases:  
  - name: ples-prod-db1
```

Slika 5.12: Servis baze podataka

Kod preostalih dvaju servisa - jednog za „frontend“, jednog za „backend“ navedena svojstva postavljamo kako je prikazano na slikama 5.13 i 5.14:

```
services:  
- type: web  
  env: docker  
  name: ples-prod-server1
```

Slika 5.13: Servis servera

```
- type: web  
  env: docker  
  name: ples-prod-client1
```

Slika 5.14: Servis klijenta

Svojstvo „env“, koje predstavlja okruženje izvođenja, potrebno je postaviti na „Docker“ čime definiramo izgradnju naše aplikacije kroz „Docker“ kontejnere. Svaki definirani servis zapravo predstavlja informacije o „Docker image“ datotekama. Sadržaj tih datoteka uključuje kod komponenti, alate sustava, biblioteke i sve ostale postavke potrebne za izgradnju kontejnera potrebnih za izvođenje aplikacije.

Svaka „Docker image“ datoteka kreirana je koristeći skup naredbi definiran u datotekama naziva „Dockerfile“ koje se nalaze u izvornom kodu aplikacije (jedna za „server“ - slika 5.15, jedna za „client“- slika 5.16). „Render“ iz tih datoteka dohvaća „startCommand“ svojstvo kojim se pokreće pojedini kontejner.

```
source_code > server > Dockerfile > FROM  
1 FROM node:18  
2  
3 WORKDIR /app  
4  
5 COPY package.json ./  
6 COPY package-lock.json ./  
7  
8 RUN npm i  
9  
10 COPY . .  
11  
12 EXPOSE 3001  
13  
14 COPY docker-entrypoint.sh /  
15 RUN chmod +x /docker-entrypoint.sh  
16 ENTRYPOINT ["/docker-entrypoint.sh"]  
17
```

Slika 5.15: Dockerfile server

```
source_code > client > Dockerfile > FROM
1 FROM node:18
2
3 WORKDIR /app
4
5 COPY package.json ./
6 COPY package-lock.json ./
7
8 RUN npm i
9
10 COPY . .
11
12 EXPOSE 3000
13
14 COPY docker-entrypoint.sh /
15 RUN chmod +x /docker-entrypoint.sh
16 ENTRYPOINT ["/docker-entrypoint.sh"]
17
```

Slika 5.16: Dockerfile klijent

Svojstvo „repo” kod oba servisa postavljamo na putanju na kojoj je dostupan repozitorij sa izvornim kodom aplikacije. Ovo svojstvo govori „Render” platformi gdje se nalazi izvorni kod aplikacije koja će biti puštena u pogon.

Svojstvo „branch” postavljamo na „master” granu repozitorija definiranog pod „repo”, dok svojstvom „rootDir” svakom servisu definiramo korijensku mapu unutar repozitorija gdje se nalazi sav kod potreban za izgradnju i izvođenje tog pojedinog servisa aplikacije.

Svojstvo „buildFilter” potrebno je za svaki servis postaviti na uzorak putanje koji predstavlja lokacije datoteka unutar repozitorija čije bi izmjene na „master” grani trebale uzrokovati ponovnu izgradnju pojedinog servisa (redployment).

```
repo: https://gitlab.com/programtvogkompjuterai/ples
branch: master
rootDir: source_code/server
buildFilter:
  paths:
    - source_code/server/**
```

Slika 5.17: Direktoriji servera

```
repo: https://gitlab.com/programtvogkompjuterai/ples
branch: master
rootDir: source_code/client
buildFilter:
  paths:
    - source_code/client/**
```

Slika 5.18: Direktoriji klijenta

Ova svojstva zapravo omogućuju automatski redeployment samo onog dijela aplikacije kod kojeg su na „master” grani uvedene nove promjene u odnosu na zadnji deploy.

Uz dosad navedena svojstva, još je potrebno definirati varijable okruženja.

„Render“ nudi mogućnost ovisnosti varijabli okruženja jednog servisa o svojstvima već gotovih „Render“ servisa. Varijable okruženja koje koristimo za spajanje „backend“ servisa na bazu podataka dohvaćamo preko definiranih svojstava „Render“ gotovog servisa za „PostgreSQL“ bazu podataka. Ostale varijable okruženja, koje ne bi smjele biti dostupne svima, stavljamo direktno na „Render“ platformu prije spremanja promjena.

```
envVars:
  - key: POSTGRES_HOST
    fromDatabase:
      name: ples-prod-db1
      property: host
  - key: POSTGRES_DB
    fromDatabase:
      name: ples-prod-db1
      property: database
  - key: POSTGRES_USER
    fromDatabase:
      name: ples-prod-db1
      property: user
  - key: POSTGRES_PASSWORD
    fromDatabase:
      name: ples-prod-db1
      property: password
  - key: POSTGRES_PORT
    fromDatabase:
      name: ples-prod-db1
      property: port
  - key: PORT
    value: 3001
```

Slika 5.19: Varijable okruženja za spajanje na bazu podataka

Za korištenje već navedene platforme, prvo je potrebno napraviti račun te se prijaviti na adresi <https://dashboard.render.com/>. Nakon izrade računa u navigaciji „Render“ web aplikacije potrebno je odabrati „Blueprints“ te „New Blueprint Instance“ - u ovom koraku potrebno je izabrati repozitorij i granu na kojoj se nalazi naš izvorni kod aplikacije u čijem se korijenskom direktoriju nalazi gore opisana „render.yaml“ datoteka. („Render“ platforma nudi unošenje podataka o vlastitom računu na „GitLab“ platformi i povezivanje s dostupnim repozitorijima).

Nakon spremanja promjena, „Render“ web aplikacija povezana je s repozitorijem u kojem se nalazi izvorni kod aplikacije sa „render.yaml“ - slika 5.20 datotekom. „Render“ dohvaća i čita sadržaj te datoteke te na temelju definirane konfiguracije pušta aplikaciju u pogon.

```
render.yaml
services:
  - type: web
    env: docker
    name: ples-prod-server1
    repo: https://gitlab.com/programtvogkompjuterai/ples
    branch: master
    rootDir: source_code/server
    buildFilter:
      paths:
        - source_code/server/**
    envVars:
      - key: POSTGRES_HOST
        fromDatabase:
          name: ples-prod-db1
          property: host
      - key: POSTGRES_DB
        fromDatabase:
          name: ples-prod-db1
          property: database
      - key: POSTGRES_USER
        fromDatabase:
          name: ples-prod-db1
          property: user
      - key: POSTGRES_PASSWORD
        fromDatabase:
          name: ples-prod-db1
          property: password
      - key: POSTGRES_PORT
        fromDatabase:
          name: ples-prod-db1
          property: port
      - key: PORT
        value: 3001

  - type: web
    env: docker
    name: ples-prod-client1
    repo: https://gitlab.com/programtvogkompjuterai/ples
    branch: master
    rootDir: source_code/client
    buildFilter:
      paths:
        - source_code/client/**
    envVars:
      - key: PORT
        value: 3000

databases:
  - name: ples-prod-db1
```

Slika 5.20: Prikaz cijele "render.yaml" datoteke

Aplikacija puštena u pogon se može pokrenuti na adresi <https://ples-prod-client1-go.j4.onrender.com/>.

6. Zaključak i budući rad

Zadatak naše grupe Program Tvoj Komputera iz predmeta programsko inženjerstvo bio je napraviti web aplikaciju koja će olakšati pronalazak plesnih klubova, tečajeva i plesnjaka u blizini korisnika.

Izrada aplikacije trajala je otprilike 16 tjedana, a provedena je u dvije glavne faze koje su odvojene unaprijed određenim rokovima projekata.

Prva faza izrade aplikacije više je naglašavala izradu dokumentacije i dobru pripremu za izradu programske potpore. Na samom početku okupljanja tima uslijedilo je upoznavanje i generalni dogovor i ideje oko mogućih tehnologija koje će biti korištene, načina komunikacije i sl. Nakon toga je uslijedio intenzivan rad na dokumentaciji, izradi modela baze podataka i napokon, implementacija određenog dijela obrazaca uporabe iz dokumentacije. U početku je većini bilo nejasno zašto je tako veliki naglasak na dokumentaciji jer nam je, naravno, to bio dosadniji dio posla koji nam je oduzimao dosta vremena jer se prije s njim nismo susreli. Kad smo krenuli s prvim implementacijskim detaljima shvatili smo zbog čega se toliko zahtjevalo na dokumentaciji jer nam je ona poslužila kao odlična podloga za realizaciju programske potpore. Budući da smo postavili dobre temelje sa obrascima uporabe, UML dijagramima i dijagramom baze podataka sve implementacijske zadatke je bilo lako i brzo podijeliti. Pomoću dijagrama smo uočili ključne i kritične dijelove koje je potrebno realizirati prije, a koje kasnije kako nebi došlo do problema.

U drugoj fazi uslijedilo je intenzivno programiranje i implementacija programske potpore za ostatak obrazaca uporabe kako bi aplikacija obuhvatila sve zadane funkcionalnosti. Generalno smo podijelili poslove po parovima kako bi zajedno radili, istraživali i međusobno se provjeravali. U nekim slučajevima smo naišli na problem zato što smo posao u parovima podijelili tako da izrada dijela programa jednog para ovisi o izradi dijela drugog para što nije bilo najidealnije rješenje zato što bi se u nekim trenucima morali čekati i parovi bi jako ovisili jedan o drugome. Jasno je da rad u timu sa sobom implicira takve probleme, no kad smo to primjetili, pokušali smo dogovoriti da jedna osoba radi zadatak u cjelini što se pokazalo kao puno brža opcija i jednostavnija za testiranje rješenja. Osim toga, nakon realizacije

je bilo potrebno izraditi ostatak dokumentacije kako bi aplikacija imala dobre temelje za održavanje.

Ulaskom u ovaj projekt imali smo različita iskustva i znanja s novim tehnologijama što nas je primoralo na dodatno istraživanje. Jednostavni zadaci su u početku trajali puno duže i bili su vrlo iscrpni tako dugo dok se nismo uhodali i samim time postali brži. Iskusniji članovi koji su se već susreli s potencijalnim problemima drugih uvijek su bili spremni ukazati na moguća rješenja i pomoći.

Svaki zadatak i merge request bio je popraćen detaljnim pregledom drugog člana tima koji je ukazao na moguće greške i bolja rješenja što nam je jako pomoglo da cijeli projekt bude konzistentan te da se potencijalne greške uoče na vrijeme.

Komunikacija grupe bila je na Slack-u gdje smo obavještavali ostale članove o zadacima na kojima radimo, stanju zadataka, problemima i sl.

Obzirom na ostale fakultetske obaveze vrlo smo zadovoljni ostvarenim ciljem. Raznolikost u iskustvima nam je puno pomogla u ostvarivanju ciljeva i možemo reći da se nakon ovog projekta osjećamo zrelije u izradi web aplikacija. Svatko od nas se susreo s nečim novim i našao način za rješenje problema. Osjetili smo što znači raditi u timu i kakve organizacijske odgovornosti nosi projekt na kojem radiš od definiranja zahtjeva do produkcije.

Popis literature

Kontinuirano osvježavanje

Popisati sve reference i literaturu koja je pomogla pri ostvarivanju projekta.

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

Indeks slika i dijagrama

2.1	Primjer prikaza tečajeva	7
2.2	Primjer prikaza informacija o tečaju	8
2.3	Primjer opisa kluba	9
2.4	Primjer podstranice o treneru	10
3.1	Dijagram obrasca uporabe, funkcionalnost vlasnika kluba.	24
3.2	Dijagram obrasca uporabe, funkcionalnost administratora.	25
3.3	Dijagram obrasca uporabe, funkcionalnost neregistriranog korisnika, klijenta i trenera.	25
3.4	Sekvencijski dijagram - Prijava trenera i njegove funkcionalnosti . .	27
3.5	Sekvencijski dijagram - Dodavanje tečajeva i prijava na tečajeve . . .	29
3.6	Sekvencijski dijagram - Dodavanje, izmjena i brisanje plesova	30
4.1	Slojevi arhitekture	34
4.2	MVC stil arhitekture	35
4.3	Relacijski dijagram baze podataka.	41
4.4	Dio klasnog dijagrama vezan uz modele User, Course i Location. . .	43
4.5	Dio klasnog dijagrama vezan uz modele Club, TrainerApplication i UserCourse.	44
4.6	Dio klasnog dijagrama vezan uz modele Dance, Event, EventDance i Lesson.	45
4.7	Dijagram stanja - klijent	46
4.8	Dijagram aktivnosti - prijava na tečaj	47
4.9	Dijagram komponenti	49
5.1	Test unosa e-mail adrese	53
5.2	Test unosa e-mail adrese	54
5.3	Test zadovoljenosti dobne granice	55
5.4	Test ispravnosti broja mobitela	56
5.5	Test valjanosti url adresa	57
5.6	Izvođenje testa valjanosti	57

5.7	Login sitemski test	58
5.8	Registracija - sitemski test	59
5.9	Kreacija tečaja - sitemski test	60
5.10	Dijagram razmjesta	61
5.11	Render platfroma	64
5.12	Servis baze podataka	64
5.13	Servis servera	65
5.14	Servis klijenta	65
5.15	Dockerfile server	65
5.16	Dockerfile klijent	66
5.17	Direktoriji servera	66
5.18	Direktoriji klijenta	66
5.19	Varijable okruženja za spajanje na bazu podataka	67
5.20	Prikaz cijele "render.yaml" datoteke	68
6.1	Aktivnost develop grane 1	80
6.2	Aktivnost develop grane 2	80
6.3	Aktivnost develop grane 3	81
6.4	Aktivnost develop grane 4	81
6.5	Aktivnostde develop grane 5	82
6.6	Aktivnost devdoc grane	83

Dodatak: Prikaz aktivnosti grupe

Dnevnik sastajanja

1. sastanak

- Datum: 14.10.2022
- Prisustvovali: Mateja Golec, Luka Nola, Nina Đurić, Lara Đaković, Lucija Domić, Ana Vrabec, Karlo Boroš
- Teme sastanka:
 - upoznavanje
 - dogovor oko korištenih tehnologija
 - dogovor načina komunikacije

2. sastanak

- Datum: 20.10.2022
- Prisustvovali: Mateja Golec, Luka Nola, Nina Đurić, Lara Đaković, Lucija Domić, demonstrator, asistent
- Teme sastanka:
 - sastanak s asistentom i demonstratorom
 - dogovor funkcionalnosti i rješavanje osnovnih dilema
 - dogovor oko korištenih tehnologija
 - analiza zadatka

3. sastanak

- Datum: 24.10.2022.
- Prisustvovali: Mateja Golec, Luka Nola, Nina Đurić, Lara Đaković, Lucija Domić, Ana Vrabec, Karlo Boroš
- Teme sastanka:
 - dogovor oko pisanja dokumentacije i podjele poslova u početnoj fazi

4. sastanak

- Datum: 25.10.2022.
- Prisustvovali: Luka Nola, Lara Đaković
- Teme sastanka:
 - izrada ER dijagrama baze podataka

5. sastanak

- Datum: 25.10.2022.
- Prisustvovali: Mateja Golec, Luka Nola, Nina Đurić, Lara Đaković, Lucija Domić, Ana Vrabec, Karlo Boroš
- Teme sastanka:
 - definiranje obrazaca uporabe
 - definiranje funkcionalnih zahtjeva
 - raspodjela poslova oko izrade obrazaca uporabe, funkcionalnih zahtjeva i opisa zadatka

6. sastanak

- Datum: 27.10.2022.
- Prisustvovali: Mateja Golec, Nina Đurić, Lara Đaković, Luka Nola, asistent
- Teme sastanka:
 - pregled funkcionalnih zahtjeva
 - pregled obrazaca uporabe
 - pregled opisa
 - razješavanje dilema za ER dijagram baze podataka

7. sastanak

- Datum: 31.10.2022.
- Prisustvovali: Mateja Golec, Nina Đurić, Lara Đaković, Lucija Domić, Ana Vrabec, Karlo Boroš
- Teme sastanka:
 - pregled obrazaca uporabe
 - upoznavanje sa latexom
 - dogovor za crtanje dijagrama obrazaca uporabe

8. sastanak

- Datum: 4.11.2022.

- Prisustvovali: Mateja Golec, Luka Nola, Nina Đurić, Lara Đaković, Lucija Domić, Ana Vrabec, Karlo Boroš
- Teme sastanka:
 - objašnjavanje radnog okruženja i pokretatanje stranice
 - definiranje sekvencijskih dijagrama
 - raspodjela poslova oko baze podataka i sekvencijskih dijagrama

9. sastanak

- Datum: 9.11.2022.
- Prisustvovali: Mateja Golec, Luka Nola, Nina Đurić, Lara Đaković, Lucija Domić, Ana Vrabec, Karlo Boroš
- Teme sastanka:
 - dogovor oko opisa arhitekture
 - raspodjela posla i dogovor oko dijagrama razreda
 - raspodjela posla i dogovor za autentikaciju i autorizaciju

10. sastanak

- Datum: 10.11.2022.
- Prisustvovali: Mateja Golec, Luka Nola, Nina Đurić, Lucija Domić, Ana Vrabec, asistent
- Teme sastanka:
 - pregled dokumentacije
 - razrješavanje pitanja u vezi baze, autorizacije i klasnih dijagrama

11. sastanak

- Datum: 14.11.2022.
- Prisustvovali: Mateja Golec, Luka Nola, Nina Đurić, Lara Đaković, Lucija Domić, Ana Vrabec, Karlo Boroš
- Teme sastanka:
 - pregled rada aplikacije
 - dovršavanje klasnih dijagrama

12. sastanak

- Datum: 16.11.2022.
- Prisustvovali: Mateja Golec, Luka Nola, Nina Đurić, Lara Đaković, Lucija Domić, Ana Vrabec, Karlo Boroš

- Teme sastanka:
 - pregled rada aplikacije
 - pregled dokumentacije

13. sastanak

- Datum: 15.12.2022.
- Prisustvovali: Mateja Golec, Luka Nola, Nina Đurić, Lara Đaković, Lucija Domić, Ana Vrabec, Karlo Boroš
- Teme sastanka:
 - podjela askova

14. sastanak

- Datum: 4.1.2023.
- Prisustvovali: Mateja Golec, Luka Nola, Nina Đurić, Lara Đaković, Lucija Domić, Ana Vrabec, Karlo Boroš
- Teme sastanka:
 - pregled odrađenog posla
 - podjela taskova

15. sastanak

- Datum: 9.1.2023.
- Prisustvovali: Mateja Golec, Luka Nola, Nina Đurić, Lara Đaković, Lucija Domić, Ana Vrabec, Karlo Boroš
- Teme sastanka:
 - pregled odrađenog posla
 - podjela taskova
 - dogovor posla za dokumentaciju

16. sastanak

- Datum: u ovom formatu: 16.12.2022.
- Prisustvovali: Mateja Golec, Nina Đurić, Lara Đaković, Lucija Domić, Ana Vrabec, Karlo Boroš
- Teme sastanka:
 - analiza use case-ova
 - podjela u parove za izradu controllera
 - dogovor oko sljedećeg checkpoint-a

Tablica aktivnosti

	Mateja Golec	Karlo Boroš	Lucija Domić	Lara Đaković	Nina Đurić	Luka Nola	Ana Vrabec
Upravljanje projektom	15h	13h	13h	14h	13h	20h	16h
Opis projektnog zadatka					6h		
Funkcionalni zahtjevi	3h						
Opis pojedinih obrazaca		5h	6h	7h			7h
Dijagram obrazaca		3h	4h	4h			4h
Sekvencijski dijagrami	4h				2h		
Opis ostalih zahtjeva							2h
Arhitektura i dizajn sustava	3h					2h	
Baza podataka	1h	5h		5h		2h	5h
Dijagram razreda			3h	11h	2h	2h	5h
Dijagram stanja	2h						
Dijagram aktivnosti		3h					
Dijagram komponenti			5h				
Korištene tehnologije i alati	2h						
Ispitivanje programskog rješenja							
Dijagram razmještaja					1.5h		
Upute za puštanje u pogon	5h						
Dnevnik sastajanja				2h			3h
Zaključak i budući rad				1.5h			
Popis literature							

Nastavljeno na idućoj stranici

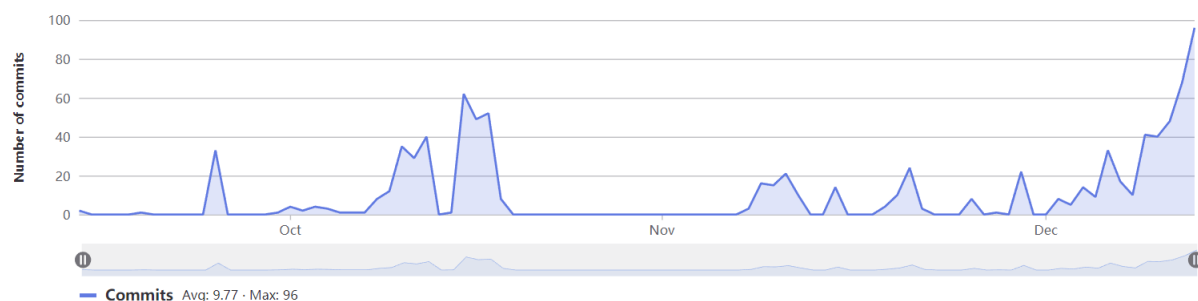
Nastavljeno od prethodne stranice

	Mateja Golec	Karlo Boroš	Lucija Domić	Lara Đaković	Nina Đurić	Luka Nola	Ana Vrabec
<i>Postavljanje klijentske strane aplikacije</i>		9h	20h		5h	2h	
<i>Postavljanje serverske strane aplikacije</i>		3h	20h		4h	2h	
<i>Konfiguracija infrastrukture</i>						1h	
<i>Izrada modela baze podataka</i>			10h	4h		3h	4h
<i>Autorizacija i autentikacija - backend</i>		5h				5h	
<i>Kreacija modela, migracija i punjenje baze</i>			3h			2h	
<i>Kreacija kluba</i>						2h	
<i>Kreacija, uređivanje i pregled korisnika</i>						5h	
<i>Pustanje u pogon (deployment)</i>						4h	
<i>Kreacija modela Course i Lesson, migracija, punjenje baze</i>				10h			
<i>Inicijalizacija mape</i>				13h			
<i>Prikaz courseva na karti</i>				17h			
<i>Prikaz Lessona na kalendaru</i>				10h			
<i>Seed adekvatnih podataka u bazu</i>				3h			

Dijagrami pregleda promjena

Commits to develop

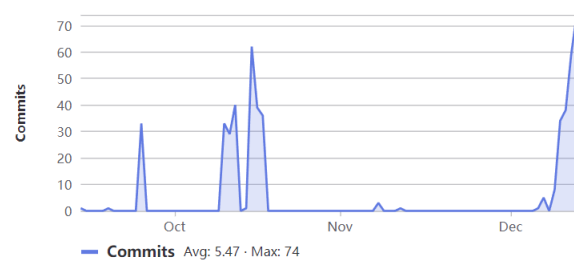
Excluding merge commits. Limited to 6,000 commits.



Slika 6.1: Aktivnost develop grane 1

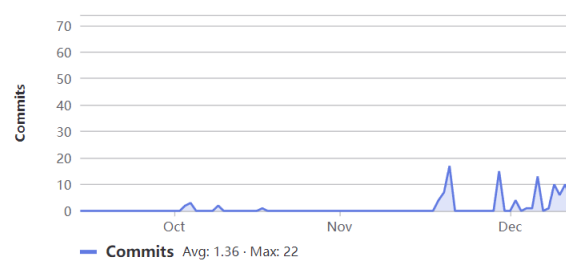
Lnola

498 commits (nola.luka@gmail.com)



Ana Vrabec

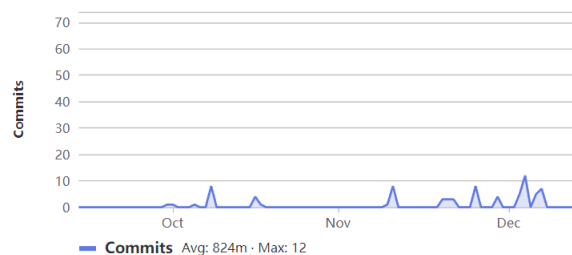
124 commits (ana.vrabec@student.hr)



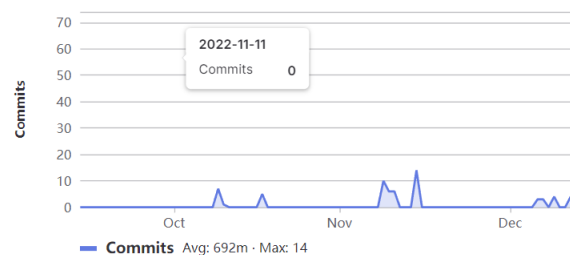
Slika 6.2: Aktivnost develop grane 2

laradjakovic229

75 commits (laradjakovic00@gmail.com)

**karloboros**

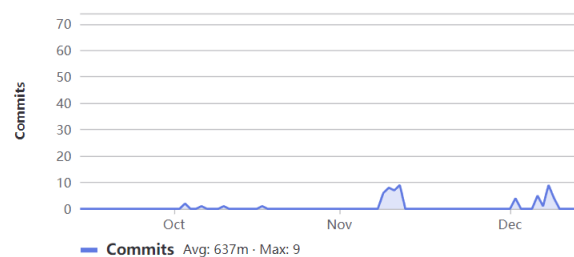
63 commits (karlo.boros1@gmail.com)



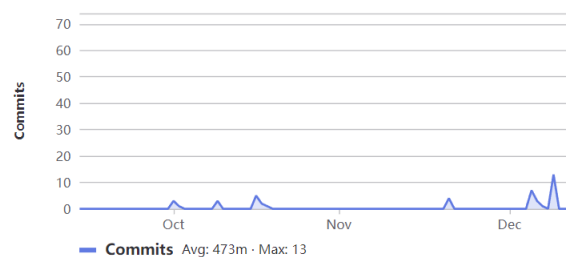
Slika 6.3: Aktivnost develop grane 3

LDomic05

58 commits (lucijadomicsplit@gmail.com)

**ninaduric7**

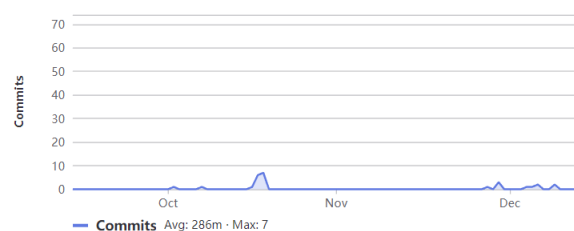
43 commits (ninaduric83@gmail.com)



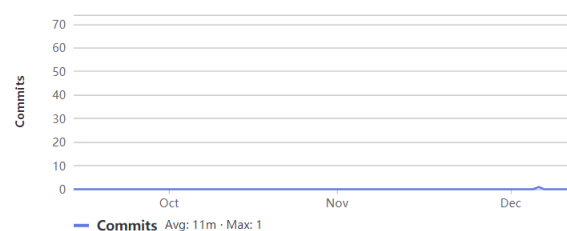
Slika 6.4: Aktivnost develop grane 4

mGolec73

26 commits (mateja.golec@fer.hr)

**mGolec73**

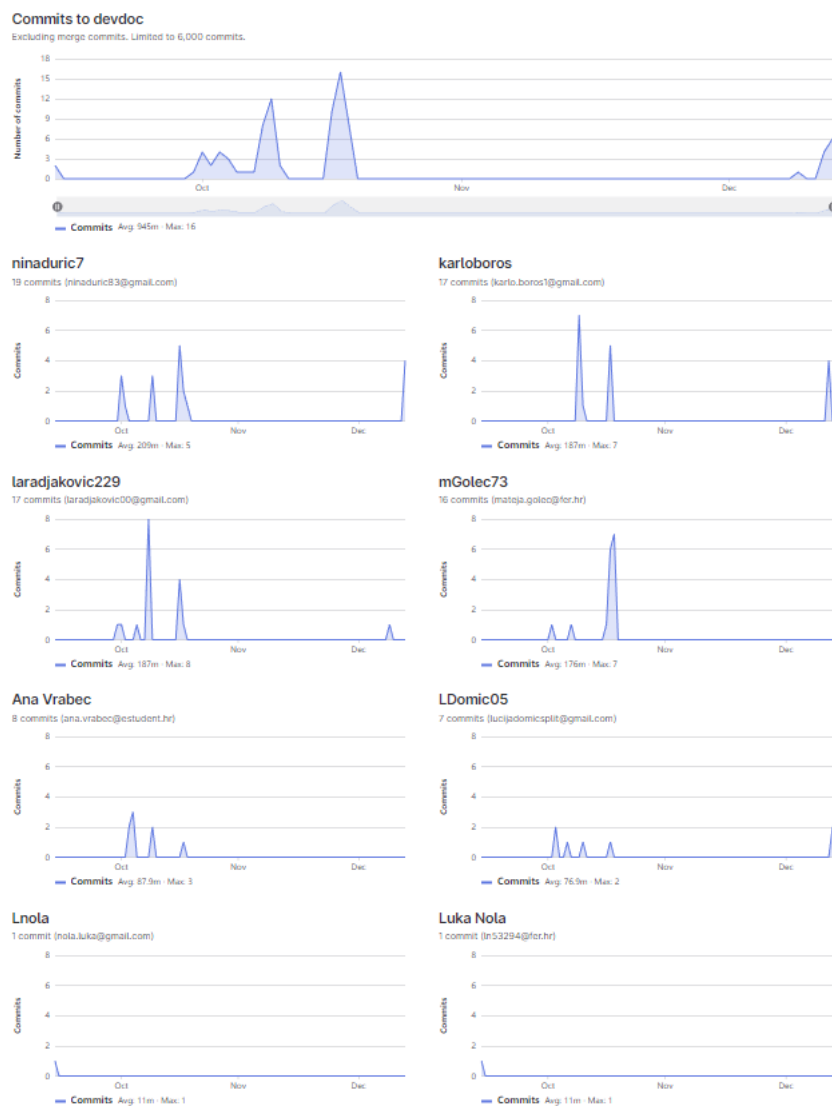
1 commit (mg53384@fer.hr)

**Luka Nola**

1 commit (ln53294@fer.hr)



Slika 6.5: Aktivnostde develop grane 5



Slika 6.6: Aktivnost devdoc grane