eugen-vucelic / FaksFit

Code Issues 9 Pull requests Actions Projects 1 Wiki Secur

Home

Edit New page

Jump to bottom

Lucija Burić edited this page on Nov 15, 2024 · 4 revisions

Dobrodošli na FaksFit wiki!

Programsko inženjerstvo ak. god. 2024./2025.

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

FaksFit

Grupa: <G04.1>

Tim:

- Brebrić Vitomir
- Burić Lucija
- Gračner Dominik (voditelj tima)
- Rajković Stjepan
- Rakić Dean
- Šarotar Adrian
- Vucelić Eugen

Nastavnik: Vlado Sruk

FaksFit

Pages 13

Find a page...

- Home
  - Programsko inženjerstvo ak. god. 2024./2025.
    - Sveučilište u Zagrebu
    - Fakultet elektrotehnike i računarstva
  - FaksFit
      - Grupa: <G04.1>
      - Tim:
      - Nastavnik: Vlado Sruk
- 1. Opis projektnog zadatka
- 2. Analiza zahtjeva
- 3. Specifikacija zahtjeva sustava
- 4. Arhitektura i dizajn sustava
- 5. Arhitektura komponenata i razmještaja
- 6. Ispitivanje programskog rješenja
- 7. Tehnologije za implementaciju aplikacije
- 8. Upute za puštanje u pogon
- 9. Zaključak i budući rad
- A. Dnevnik promjena dokumentacije
- A. Popis literature
- B. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```

1. Opis projektnog zadatka

dean-rakic edited this page on Nov 14, 2024 · 21 revisions

Ovaj projekt je rezultat timskog rada u sklopu projektnog zadatka kolegija Programsko inženjerstvo na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu.

Mnogi fakulteti Sveučilišta u Zagrebu u svojim programima imaju obavezne kolegije Tjelesne i zdravstvene kulture. Studentima se nude mnogi načini da polože ovaj kolegij, kroz članstva u sportskim sekcijama i dolaske na treninge ili organizirane aktivnosti. Zbog velike fleksibilnosti ovoga kolegija, često je njegovo praćenje zbunjujuće kako studentima, tako i nositeljima kolegija.

Naša jedinstvena web aplikacija će postaviti sve načine praćenja kolegija TZK na jedno mjesto i pojednostaviti njegovo praćenje svim studentima, voditeljima sekcija i nastavnicima. Voditelji aktivnosti/sportskih sekcija mogu putem aplikacije prijavljivati termine kada će se aktivnost održavati, a studenti mogu kroz aplikaciju prijaviti termin koji im odgovara. Nakon obavljenog termina, voditelj sekcije evidentira dolazak i upisuje bodove te na kraju semestra nastavnik može pregledati skupljene bodove i obavijestiti studenta o prolasku predmeta. Ideja je s ovom aplikacijom studentima značajno olakšati proces prolaska kolegija Tjelesne i zdravstvene kulture.

Jumbula je aplikacija sa sličnom premisom koja koristi class registration software. Opis aplikacije: "Jumbula helps organizations create flexible class schedules, onboard participants, collect payments, manage private and group classes, and communicate with participants and their families for online or in-person classes."

Slika: Izgled interface-a aplikacije Jumbula

Aplikacija bi bila korisna fakultetima diljem Hrvatske, njihovim profesorima Tjelesne i zdravstvene kulture te naravno studentima. Ovisno o tipu fakulteta, te državi u kojoj se koristi, aplikacija bi se mogla prilagoditi različitim aktivnostima i načinima prolaska kolegija. Dodatno bi se mogao implementirati i prijevod na druge jezike u regiji.
Cilj ovog projektnog zadatka je napraviti funkcionalni prototip aplikacije koji koristi bazu podataka gdje su spremljeni svi podaci profesora i studenata, omogućuje prijavu studenta na određeni termin aktivnosti koji stvara voditelj aktivnosti i evidenciju ostvarenih bodova. Student može vidjeti te ostvarene bodove, a profesor ih može potvrditi i na osnovu njih zaključiti prolaz ili pad direktno preko aplikacije.

FaksFit

Pages 13

Find a page...

- Home
- 1. Opis projektnog zadatka
- 2. Analiza zahtjeva
- 3. Specifikacija zahtjeva sustava
- 4. Arhitektura i dizajn sustava
- 5. Arhitektura komponenata i razmještaja
- 6. Ispitivanje programskog rješenja
- 7. Tehnologije za implementaciju aplikacije
- 8. Upute za puštanje u pogon
- 9. Zaključak i budući rad
- A. Dnevnik promjena dokumentacije
- A. Popis literature
- B. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```

2. Analiza zahtjeva

dean-rakic edited this page on Nov 14, 2024 · 1 revision

Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
FS-01	Sustav omogućuje registraciju unosom osobnih podataka i matičnog fakulteta	Visok	Zahtjev dionika	Korisnik može unijeti mail od fakulteta, ime, prezime, jmbag te odabrati matični fakultet i semestar u kojem polaže kolegij, nakon čega se prijavljuje u sustav ako već nije registriran
FS-02	Sustav omogućuje pregled i pristup aktivnostima svog fakulteta	Visok	Postojeći sustav	Korisnik može vidjeti ažurirani popis dostupnih aktivnosti za svoj fakultet ovisno o vrsti aktivnosti, vremenu i lokaciji održavanja (putem vanjskog servisa poput Google Maps-a) te broju preostalih mjesta
FS-03	Sustav omogućuje prijavu na nadolazeće termine do 24 sata prije početka	Visok	Zahtjev dionika	Korisnik može izabrati termine u kalendaru na kojima želi prisustvovati određenoj aktivnosti

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
FS-04	Sustav omogućuje pregled profila studenta	Visok	Postojeći sustav	Korisnik može pregledati svoje podatke i ostvarene bodove na kolegiju za ovaj semestar
FS-05	Sustav šalje obavijesti korisniku o dolasku ili nedolasku na termin, odnosno evidentiranim bodovima koje je skupio	Visok	Povratne informacije korisnika	Korisnik prima obavijest nakon što se evidentira njihov dolazak na aktivnost i bodovi se unesu u sustav
FV-01	Sustav omogućuje stvaranje termina za aktivnosti od strane voditelja sekcije	Visok	Zahtjev dionika	Korisnik može unijeti podatke o novom terminu, uključujući naziv, opis, vrijeme, lokaciju i kapacitet termina
FV-02	Sustav omogućuje evidentiranje bodova od strane voditelja sekcije	Visok	Zahtjev dionika	Korisnik može evidentirati bodove za određene studente koji su prijavili termin ovisno o tome jesu li došli na taj termin
FN-01	Sustav omogućuje određivanje bodova za pojedine aktivnosti i ukupnog broja bodova za prolazak kolegija te dodjeljivanje bodova nakon termina	Visok	Zahtjev dionika	Korisnik može odrediti broj bodova za određenu aktivnost nekog od voditelja sekcija te unijeti traženi broj bodova za prolazak kolegija. Isto tako može potvrditi ostvarene bodove studentu nakon evidentiranog termina
FN-02	Sustav omogućuje pregled svih studenata na njihovom kolegiju od strane nastavnika	Visok	Postojeći sustav	Korisnik može pregledati popis svih studenata koji su izabrali nastavnikov fakultet
FN-03	Sustav omogućuje pregled i pristup	Srednji	Postojeći sustav	Korisnik može pregledati podatke svojih studenata

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
	podatcima studenata od strane nastavnika			uključujući ostvarene bodova na kolegiju Tjelesne i zdravstvene kulture te ako je potrebno, manualno ih mijenjati
FN-04	Sustav omogućuje bilježenje prolaska kolegija na kraju semestra od strane nastavnika	Visok	Zahtjev dionika	Korisnik može na osnovu dotad ostvarenih bodova pojedinog studenta zaključiti prolaz odnosno pad kolegija

Aplikacija podržava tri tipa korisnika:

1. Student:

- Prilikom registracije unosi osobne podatke i matični fakultet.
- Nakon potvrde od nastavnika, može pristupiti aktivnostima svog fakulteta.
- Za prolaz kolegija, dužan je skupiti broj bodova koji definira nastavnik.
- Može vidjeti informacije o svim dostupnim aktivnostima (naziv, opis, vrijeme, lokacija, preostala mjesta).
- Može se prijavljivati na nadolazeće termine aktivnosti do 24 sata prije početka,
- Nakon termina dobiva obavijest o prisutnosti ili nedolasku.
- Nakon potvrđenog prolaza od nastavnika, više ne može birati termine.

2. Voditelj sekcije:

- Registrira detalje o sekciji i šalje ponude nastavnicima.
- Nakon potvrde, može stvarati termine za aktivnosti, unoseći naziv, opis, vrijeme, kapacitet i lokaciju.
- Nakon termina, evidentira prisutnost studenata i dodjeljuje bodove prema uputama nastavnika.

3. Nastavnik:

- Ima najviše ovlasti i radi na jednom fakultetu.
- Pregledava ponude voditelja sekcija i može ih prihvatiti ili odbiti, uz određivanje bodova za aktivnosti.
- Postavlja ukupan broj bodova koje studenti trebaju skupiti.
- Može voditi vlastitu aktivnost i nuditi je drugim nastavnicima.

- Prihvaća studente koji su odabrali matični fakultet pod njihovom nadležnosti, a na popisu su upisanih te može upravljati njihovim bodovima.
- Do kraja semestra, mora evidentirati prolaz ili pad predmeta za sve svoje studente.

Nefunkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet
NF-01	Sustav treba omogućiti učitavanje stranice unutar 3 sekunde.	Visok
NF-02	Sustav treba podržavati simultano korištenje 100 korisnika bez smanjenja performansi.	Visok
NF-03	Sustav treba omogućiti autentificiranu prijavu putem OAuth 2.0 protokola.	Visok
NF-03	Svi razvojni dokumenti sustava trebaju biti pohranjeni u verzijskom sustavu (Git) s jasno definiranim granama i pravilima za njihovo spajanje.	Visok
NF-03	Sustav mora biti usklađen s GDPR regulativama.	Visok

1. Zahtjevi programskog proizvoda:

- Sustav treba omogućiti učitavanje stranice unutar 3 sekunde.
- Sustav treba podržavati simultano korištenje 100 korisnika bez smanjenja performansi.
- Sustav treba omogućiti autentificiranu prijavu putem OAuth 2.0 protokola.

2. Organizacijski zahtjevi

- Svi razvojni dokumenti i kod trebaju biti pohranjeni u verzijskom sustavu (Git) s jasno definiranim granama i pravilima za njihovo spajanje.

3. Vanjski zahtjevi

- Sustav mora biti usklađen s GDPR regulativama.

Dionici

1. Student
2. Voditelj sekcije/aktivnosti
3. Nastavnik

4. Razvojni tim

Aktori i njihovi funkcionalni zahtjevi:

1. Svaki neregistrirani/neprijavljeni korisnik (inicijator) može:
   (a) registrirati/prijaviti se u sustav

2. Svaki prijavljeni korisnik (inicijator) može:
   (a) promijeniti neke od podataka u profilu

3. Student (inicijator) može:
   (a) pregledati kalendar termina
   (b) prijaviti određeni termin
   (c) pregledati svoj profil i ostvarene bodove

4. Voditelj sekcije (inicijator) može:
   (a) unijeti novi termin u kalendar
   (b) evidentirati dolazak pojedinog studenta na termin

5. Nastavnik (inicijator) može:
   (a) potvrditi bodove studentu nakon evidentiranog dolaska
   (b) pregledati studentove podatke i po potrebi mijenjati broj bodova
   (c) upisati prolaz odnosno pad pojedinom studentu na kraju semestra

FaksFit

Pages 13

Find a page...

Home

1. Opis projektnog zadatka

2. Analiza zahtjeva

Funkcionalni zahtjevi

1. Student:
2. Voditelj sekcije:
3. Nastavnik:

Nefunkcionalni zahtjevi

1. Zahtjevi programskog proizvoda:
2. Organizacijski zahtjevi
3. Vanjski zahtjevi

Dionici

- 3. Specifikacija zahtjeva sustava
- 4. Arhitektura i dizajn sustava
- 5. Arhitektura komponenata i razmještaja
- 6. Ispitivanje programskog rješenja
- 7. Tehnologije za implementaciju aplikacije
- 8. Upute za puštanje u pogon
- 9. Zaključak i budući rad
- A. Dnevnik promjena dokumentacije
- A. Popis literature
- B. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```

eugen-vucelic / FaksFit

Code Issues 9 Pull requests Actions Projects 1 Wiki Secu

3. Specifikacija zahtjeva sustava

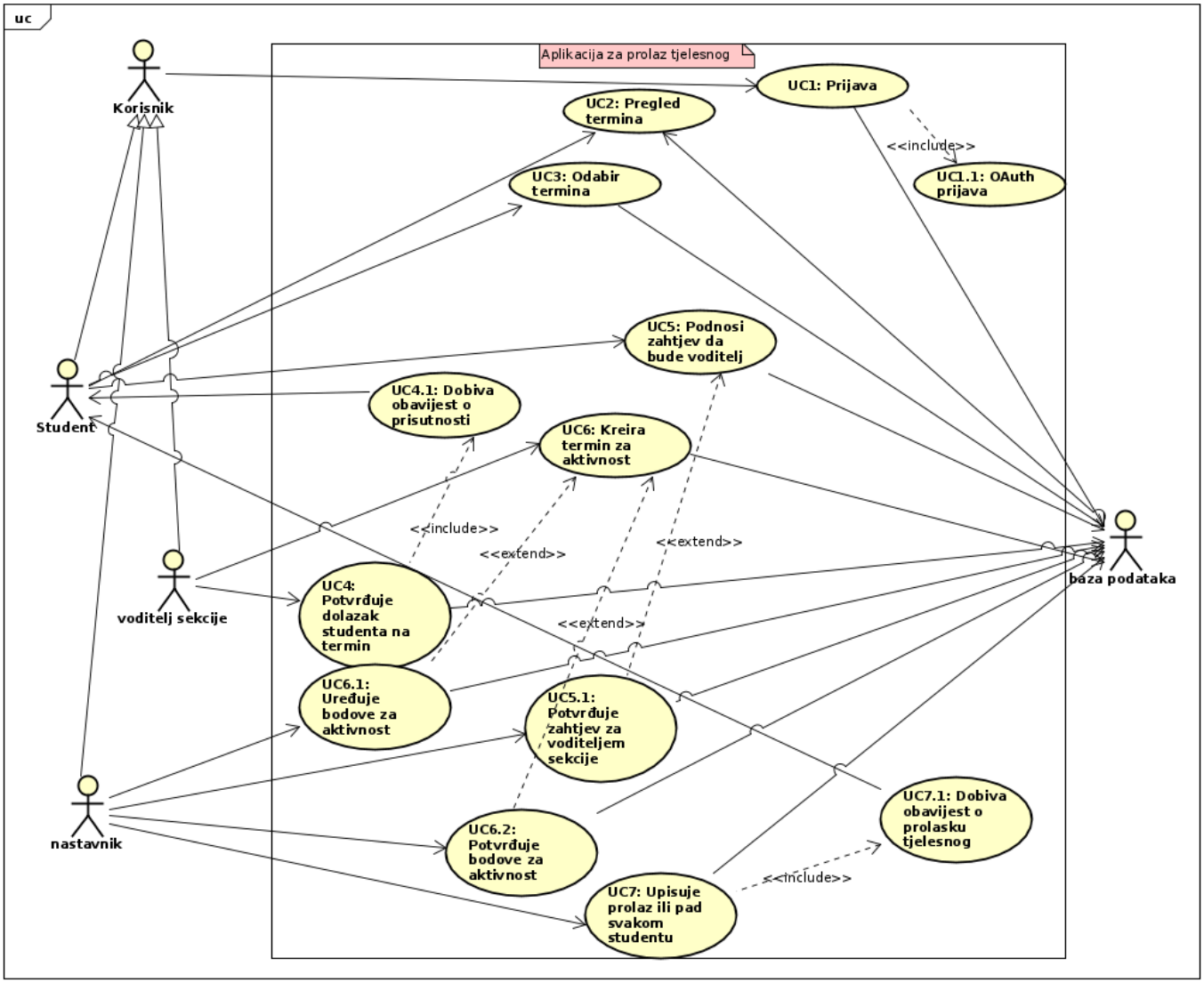
Edit New page

Jump to bottom

dean-rakic edited this page 5 hours ago · 2 revisions

Obrasci uporabe

Use case diagram 1



UC1 - Prijava

- **Glavni sudionik** - Korisnik (student, voditelj sekcije, nastavnik)
- **Cilj** - Ulazak u aplikaciju i pristup korisničkom sučelju
- **Sudionici** - Baza podataka
- **Preduvjet** - Posjedovanje vlastitog korisničkog računa
- **Opis osnovnog tijeka**:
  i. Korisnika se preusmjeruje na OAuth prijavu u sustav
- **Opis mogućih odstupanja**:

2. a U trenutku prijave, sustav OAuth ne radi
   i. Aplikacija obavještava korisnika da se pokuša prijaviti malo kasnije

UC1.1 - OAuth prijava

- **Glavni sudionik** - Korisnik (student, voditelj sekcije, nastavnik)
- **Cilj** - Ulazak u aplikaciju i pristup korisničkom sučelju
- **Sudionici** - Baza podataka
- **Preduvjet** - Posjedovanje vlastitog korisničkog računa
- **Opis osnovnog tijeka**:
  i. Korisnik upisuje OAuth korisničko ime i lozinku
  ii. Baza podataka provjerava postoji li račun s upisanim podacima
  iii. Ukoliko postoji, korisnik dobiva pristup korisničkom sučelju
- **Opis mogućih odstupanja**:

2. a Korisnik je upisao neispravno OAuth korisničko ime i/ili lozinku
   i. Aplikacija obavještava korisnika o neispravno unesenim podacima te te ga preusmjeruje na stranicu za prijavu

UC2 - Pregled termina

- **Glavni sudionik** - Student
- **Cilj** - Prikazati studentu postojeće termine u kalendaru
- **Sudionici** - Baza podataka
- **Preduvjet** - Student je prijavljen u aplikaciju
- **Opis osnovnog tijeka**:
  i. Korisnik odabire opciju pregleda termina
  ii. Baza podataka dohvaća postojeće podatke o terminima i prikazuje ih na sučelju

UC3 - Odabir termina

- **Glavni sudionik** - Student
- **Cilj** - Omogućiti studentu biranje termina i obavijestiti ga o statusu popunjenosti
- **Sudionici** - Baza podataka
- **Preduvjet** - Student je prijavljen u aplikaciju
- **Opis osnovnog tijeka**:
  i. Student izabire dostupan termin na kalendaru i pritišće na opciju za odabir termina
  ii. Baza podataka provjerava je li odabrani termin popunjen, ako nije, obavještava studenta da je uspješno odabrao termin.
  iii. Baza podataka sprema podatke o studentu koji se prijavio za odabrani termin i ažurira podatke o popunjenosti
- **Opis mogućih odstupanja**:

2. a Odabrani termin je popunjen
   i. Aplikacija šalje studentu obavijest da je odabrani termin popunjen i da treba izabrati drugi

UC4 - Potvrđuje dolazak studenta na termin

- **Glavni sudionik** - Voditelj sekcije
- **Cilj** - Voditelj sekcije pregledava popis studenata koji su odabrali termin za njegovu sekciju i čekaju potvrdu te dobiva mogućnost potvrde odnosno odbijanja određenih zahtjeva
- **Sudionici** - Baza podataka
- **Preduvjet** - Voditelj sekcije je prijavljen u aplikaciju te popis studenata koji čekaju potvrdu termina nije prazan
- **Opis osnovnog tijeka**:
  i. Voditelj sekcije pritišće na opciju pregleda popisa studenata
  ii. Baza podataka dohvaća podatke o studentima koji su odabrali termine za sekciju za koju je voditelj sekcije zadužan i prikazuje popis na sučelju
  iii. Voditelj sekcije bira između opcija potvrde i odbijanja termina koji je prijavio određeni student
  iv. Baza podataka sprema podatke o potvrđenim studentima i ažurira popunjenost tog termina

UC4.1 - Dobiva obavijest o prisutnosti

- **Glavni sudionik** - Student
- **Cilj** - Student dobiva obavijest da je prihvaćen na termin za aktivnost koju je prijavio
- **Sudionici** - Aplikacija ?
- **Preduvjet** - Student je prijavljen u aplikaciju i voditelj sekcije je potvrdio dolazak
- **Opis osnovnog tijeka**:
  i. Student pritišće na opciju obavijesti

ii. Iz baze podataka se dohvaća obavijest o potvrdi dolaska i prikazuje studentu

UC5 - Podnosi zahtjev da bude voditelj

- **Glavni sudionik** - Student
- **Cilj** - Studentu omogućiti da pritiskom na opciju slanja zahtjeva za postajanje voditeljem sekcije može napisati razlog i poslati novi
  zahtjev za postajanje voditeljem sekcije
- **Sudionici** - Baza podataka
- **Preduvjet** - Student je prijavljen u aplikaciju
- **Opis osnovnog tijeka**:
  i. Student pritišće na opciju slanja zahtjeva za postajanje voditelja sekcije na korisničkom sučelju
  ii. Studentu se prikazuje tekstualno polje u kojem može upisati razlog za slanje zahtjeva
  iii. Odabirom potvrde slanja zahtjeva, u bazu podataka se sprema novi studentov zahtjev za postajanje voditeljem sekcije
- **Opis mogućih odstupanja**:

2. a Korisnik odustaje od slanja zahtjeva
   i. Pritiskom na opciju otkazivanja pisanja zahtjeva, studenta se šalje nazad na glavnu stranicu korisničkog sučelja

UC5.1 - Potvrđuje zahtjev za voditeljem sekcije

- **Glavni sudionik** - Nastavnik
- **Cilj** - Nastavnik može potvrditi ili odbiti zahtjeva studenata za postajanje voditeljem sekcije
- **Sudionici** - Baza podataka
- **Preduvjet** - Student je poslao zahtjev za voditeljem sekcije i nastavnik je prijavljen u aplikaciju
- **Opis osnovnog tijeka**:
  i. Nastavnik pritišće na opciju pregleda zahtjeva
  ii. Iz baze podataka se dohvaćaju postojeći zahtjevi i prikazuju se nastavniku na sučelju
  iii. Nastavnik odabirom može potvrditi ili odbiti zahtjev
  iv. Nastavnikov odabir se sprema u bazu podataka te se studentu šalje obavijest o odluci

UC6 - Kreira termin za aktivnost

- **Glavni sudionik** - Voditelj sekcije
- **Cilj** - Voditelj sekcije pritiskom na opciju stvaranja novog termina za aktivnost sprema u kalendar novi termin
- **Sudionici** - Baza podataka
- **Preduvjet** - Voditelj sekcije je prijavljen u aplikaciju

- **Opis osnovnog tijeka**:
  i. Voditelj sekcije pritišće na opciju stvaranja novog termina za aktivnost
  ii. Voditelj sekcije unosi detalje novog termina i potvrđuje ih
  iii. Baza podataka sprema podatke novog termina
- **Opis mogućih odstupanja**:

2. a Voditelj sekcije odustaje od unosa novog termina
   i. Pritiskom na opciju odustani, voditelja sekcije se šalje na glavnu stranicu korisničkog sučelja

UC6.1 - Uređuje bodove za aktivnost

- **Glavni sudionik** - Nastavnik
- **Cilj** - Nastavnik ima mogućnost određivanja količine ostvarenih bodova za određenu aktivnost
- **Sudionici** - Baza podataka
- **Preduvjet** - Nastavnik je prijavljen u aplikaciju
- **Opis osnovnog tijeka**:
  i. Pritiskom na opciju uređivanja bodova, nastavnika se preusmjerava na stranicu na kojoj može urediti količinu bodova za određenu aktivnost
  ii. Odabirom količine bodova za postojeću aktivnost, nastavnik pritišće opciju potvrde
  iii. U bazu podataka se spremaju podaci o količini ostvarenih bodova za određenu aktivnost

UC6.2 - Potvrđuje bodove za aktivnost

- **Glavni sudionik** - Nastavnik
- **Cilj** - Nastavnik potvrđuje broj bodova za termin aktivnosti
- **Sudionici** - Baza podataka
- **Preduvjet** - Postoji termin za aktivnost i nastavnik je prijavljen u aplikaciju
- **Opis osnovnog tijeka**:
  i. Nastavnik pritišće na opciju potvrđivanja bodova za aktivnost
  ii. Iz baze podataka se dohvaćaju podaci o postojećim terminima i prikazuju se nastavniku
  iii. Nastavnik potvrđuje broj bodova za aktivnost
  iv. U bazu podataka se spremaju potvrđeni bodovi

UC7 - Upisuje prolaz ili pad svakom studentu

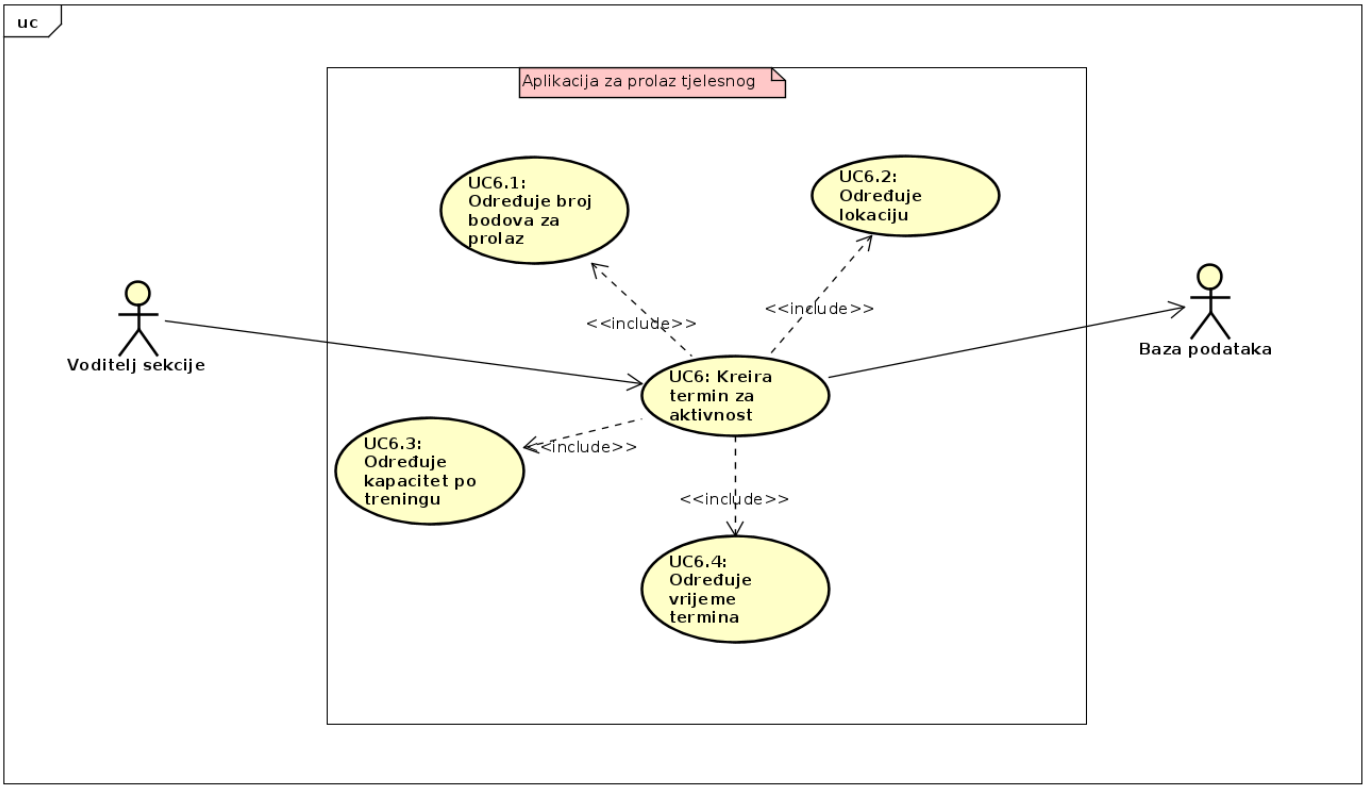
- **Glavni sudionik** - Nastavnik
- **Cilj** - Nastavnik ima mogućnost pregleda skupljenih bodova svakog studenta u ovom semestru te potvrdu prolaska odnosno pada predmeta tjelesne aktivnosti

- **Sudionici** - Baza podataka, Aplikacija ?
- **Preduvjet** - Nastavnik je prijavljen u aplikaciju i postoje podaci o skupljenim bodovima u ovom semestru
- **Opis osnovnog tijeka**:
  i. Nastavnik pritišće na opciju pregleda skupljenih bodova svakog studenta
  ii. Aplikacija dohvaća iz baze podataka podatke o skupljenim bodovima i prikazuje ih
  iii. Pritiskom na prolaz odnosno pad predmeta za određenog studenta, nastavnik izabire koji student je prošao predmet tjelesne aktivnosti a koji nije na osnovu skupljenih bodova
  iv. U bazu podataka se spremaju podaci o prolasku predmeta za svakog studenta

UC7.1 - Dobiva obavijest o prolasku tjelesnog

- **Glavni sudionik** - Student
- **Cilj** - Student dobiva obavijest o prolasku predmeta tjelesne aktivnosti, kojeg može vidjeti pritiskom na opciju obavijesti
- **Sudionici** - Aplikacija ?
- **Preduvjet** - Nastavnik je izabrao opciju prolaska odnosno pada predmeta za određenog studenta te je student prijavljen u aplikaciju
- **Opis osnovnog tijeka**:
  i. Nakon nastavnikove potvrde prolaska odnosno pada predmeta, aplikacija šalje studentu obavijest
  ii. Student pregledava obavijest pritiskom na opciju obavijesti

Use case diagram 2



UC6 - Kreira termin za aktivnost

- **Glavni sudionik** - Voditelj sekcije
- **Cilj** - Voditelj sekcije stvara novi termin za svoju aktivnost koja se sprema u bazu podataka
- **Sudionici** - Baza podataka
- **Preduvjet** - Voditelj sekcije je prijavljen u aplikaciju
- **Opis osnovnog tijeka**:
  i. Voditelj sekcije pritišće opciju kreiranja novog termina
  ii. Voditelj sekcije unosi detalje novog termina i potvrđuje ih
  iii. U bazu podataka se spremaju podaci o novom terminu i ažurira se kalendar

UC6.1 - Određuje broj bodova za prolaz

- **Glavni sudionik** - Voditelj sekcije
- **Cilj** - Voditelj sekcije ima opciju određivanja potrebnih bodova za prolaz njegove aktivnosti
- **Sudionici** - Baza podataka
- **Preduvjet** - Voditelj sekcije je prijavljen u aplikaciju
- **Opis osnovnog tijeka**:
  i. Voditelj sekcije pritišće opciju uređivanja broja bodova
  ii. Voditelj sekcije potvrđuje broj bodova za prolaz
  iii. U bazu podataka se sprema potrebni broj bodova za prolaz aktivnosti

UC6.2 - Određuje lokaciju

- **Glavni sudionik** - Voditelj sekcije
- **Cilj** - Voditelj sekcije ima opciju uređivanja lokacije gdje se održava termin aktivnosti
- **Sudionici** - Baza podataka
- **Preduvjet** - Voditelj sekcije je prijavljen u aplikaciju
- **Opis osnovnog tijeka**:
  i. Voditelj sekcije pritišće na opciju uređivanja lokacije
  ii. Voditelj sekcije potvrđuje lokaciju
  iii. U bazu podataka se sprema lokacije nove aktivnosti

UC6.3 - Određuje kapacitet po treningu

- **Glavni sudionik** - Voditelj sekcije
- **Cilj** - Voditelj sekcije ima opciju određivanja kapaciteta broja studenata na terminu
- **Sudionici** - Baza podataka
- **Preduvjet** - Voditelj sekcije je prijavljen u aplikaciju
- **Opis osnovnog tijeka**:
  i. Voditelj sekcije pritišće na opciju određivanja kapaciteta
  ii. Voditelj sekcije potvrđuje kapacitet
  iii. U bazu podataka se sprema podatak o kapacitetu novog termina

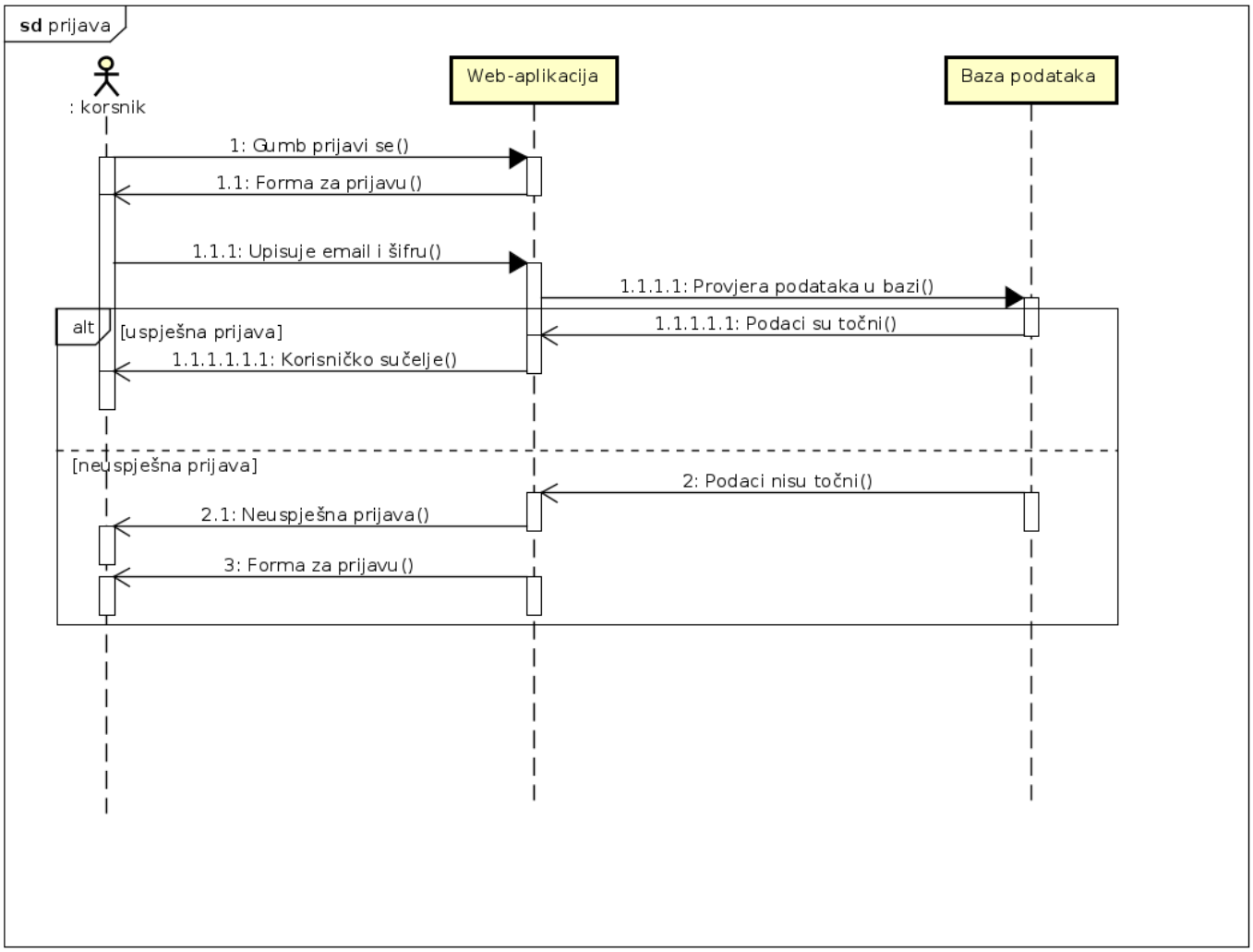
UC6.4 - Određuje vrijeme termina

- **Glavni sudionik** - Voditelj sekcije
- **Cilj** - Voditelj sekcije ima opciju određivanja vremena termina
- **Sudionici** - Baza podataka
- **Preduvjet** - Voditelj sekcije je prijavljen u aplikaciju
- **Opis osnovnog tijeka**:
  i. Voditelj sekcije pritišće na opciju određivanja vremena termina
  ii. Voditelj sekcije potvrđuje odabir
  iii. U bazu podataka se sprema podatak o vremenu termina

Sekvencijski dijagrami

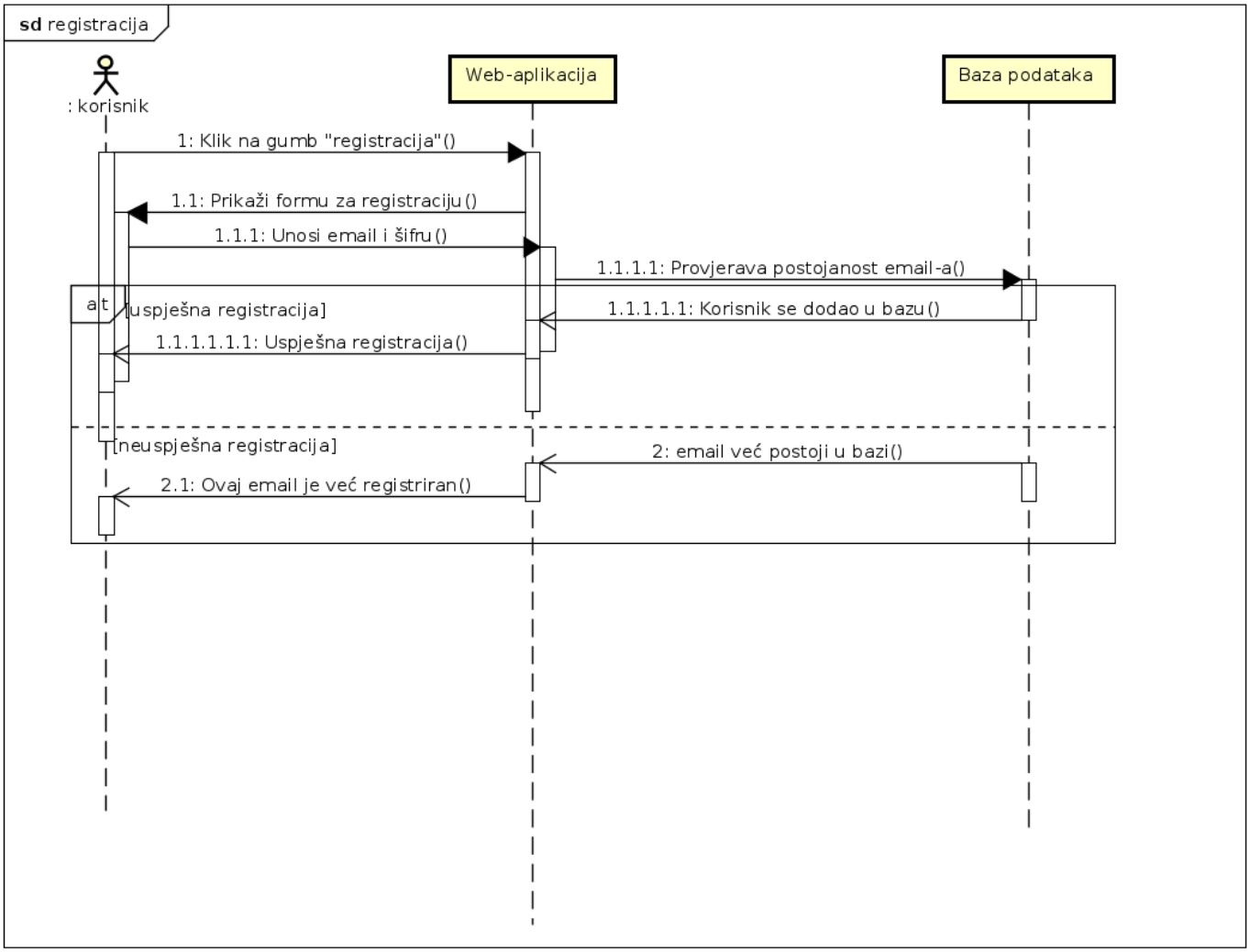
SD1 - prijava:

Korisnik pritišće gumb za prijavu, aplikacija kao odgovor šalje formu za prijavu u koju korisnik upisuje svoje korisničke podatke (email i šifru). Aplikacija provjerava postoji li račun s upisanim podacima spremljen u bazu podataka. Ukoliko postoji i podaci su točni, korisnik ulazi u aplikaciju i aplikacija mu prikazuje korisničko sučelje. U suprotnom scenariju u kojemu podaci nisu ispravni, aplikacija javlja korisniku da je došlo do neuspješne prijave i vraća ga na korisničko sučelje, gdje ponovo može upisati podatke za prijavu.



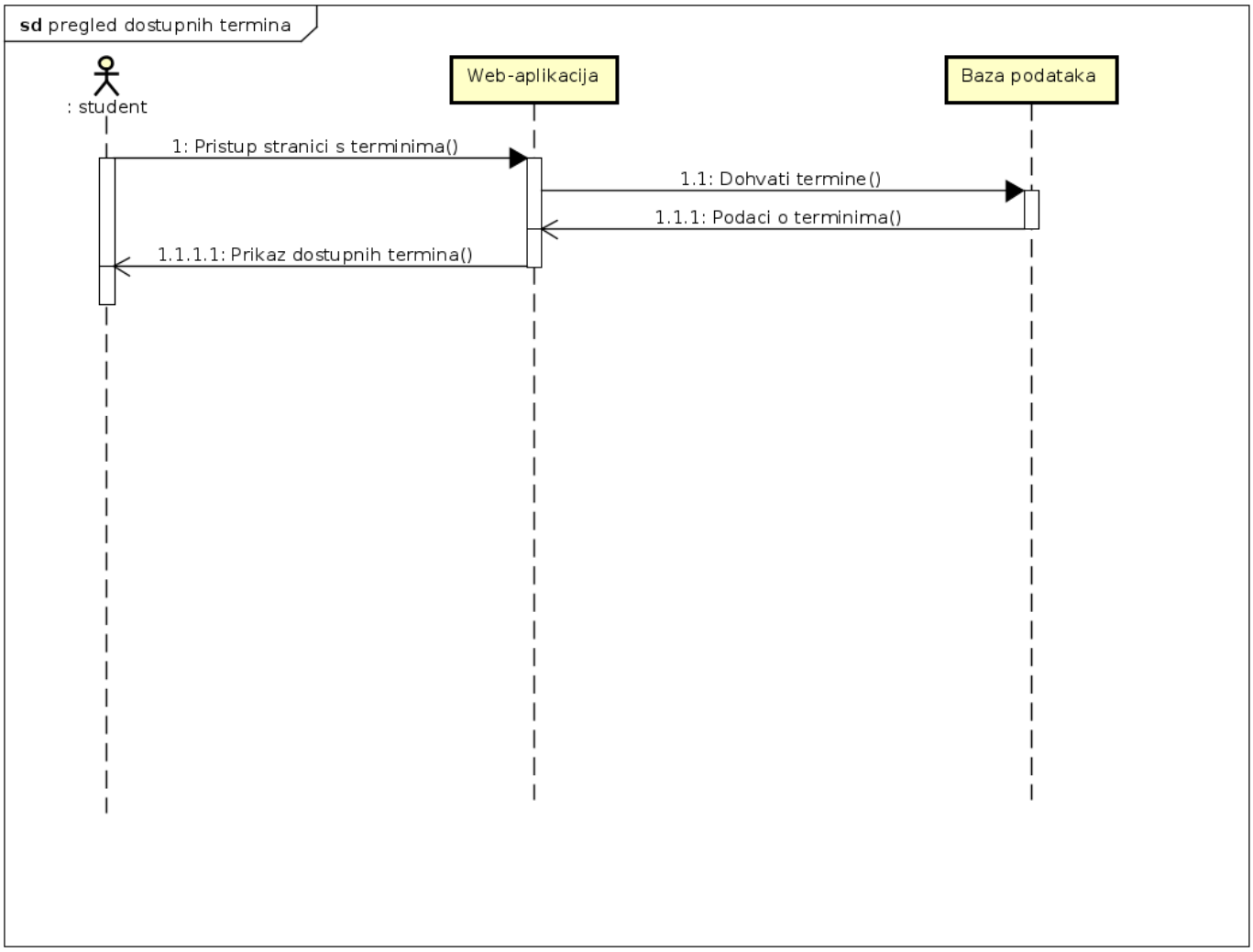
SD2 - registracija:

Korisnik pritišće gumb za registraciju, aplikacija kao odgovor šalje formu za registraciju u koju korisnik unosi email adresu i šifru za novi račun. Aplikacija provjerava postoji li u bazi podataka već račun s unesenom email adresom. Ukoliko ne postoji, aplikacija javlja korisniku da je uspješno registriran i podaci su ubačeni u bazu podataka. U suprotnom scenariju, email već postoji u bazi, te aplikacija javlja korisniku da je uneseni email već registriran u aplikaciju.



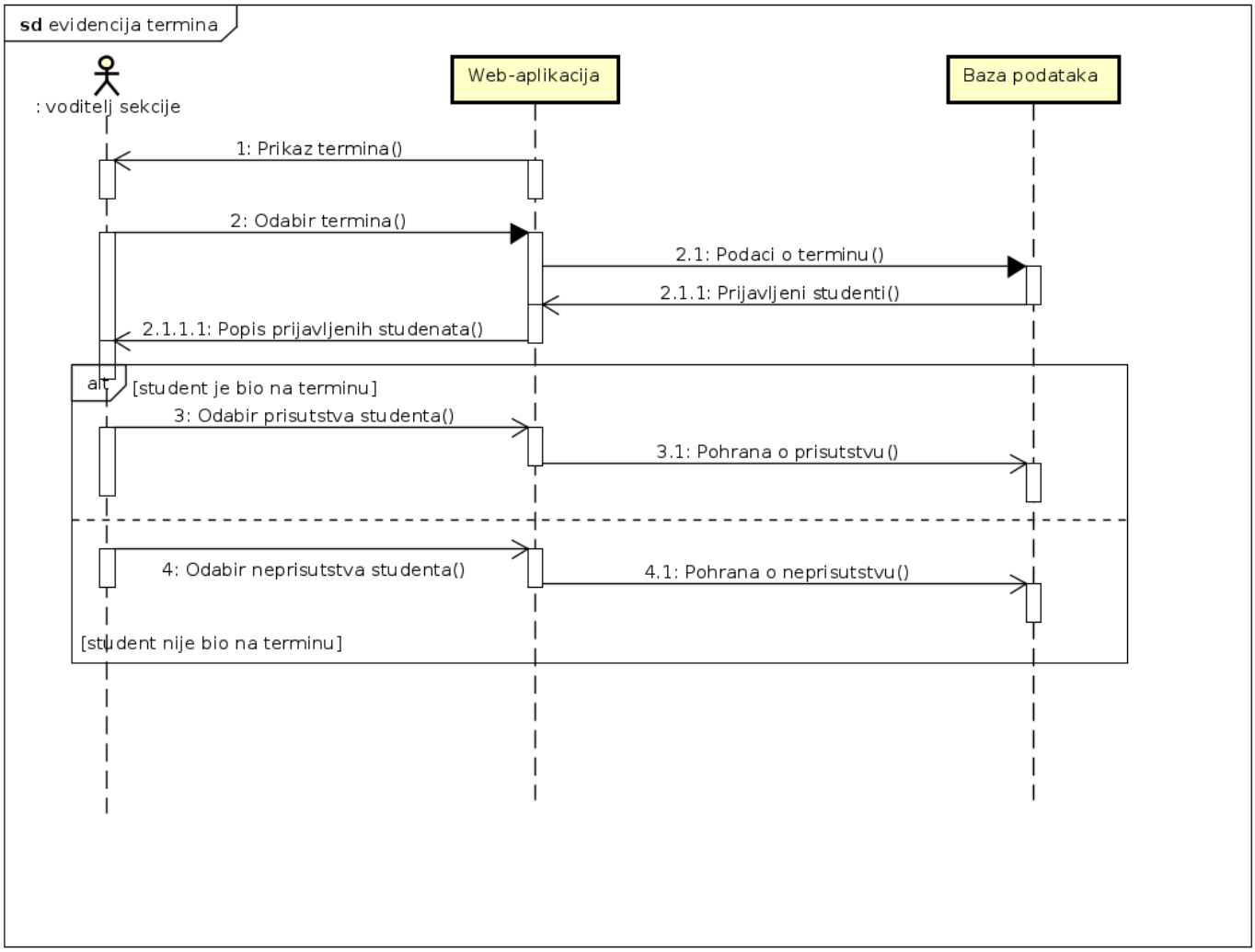
SD3 - pregled dostupnih termina:

Korisnik pristupa stranici s terminima, aplikacija u tom trenutku dohvaća podatke o terminima iz baze podataka te korisniku prikazuje dostupne termina na korisničkom sučelju.



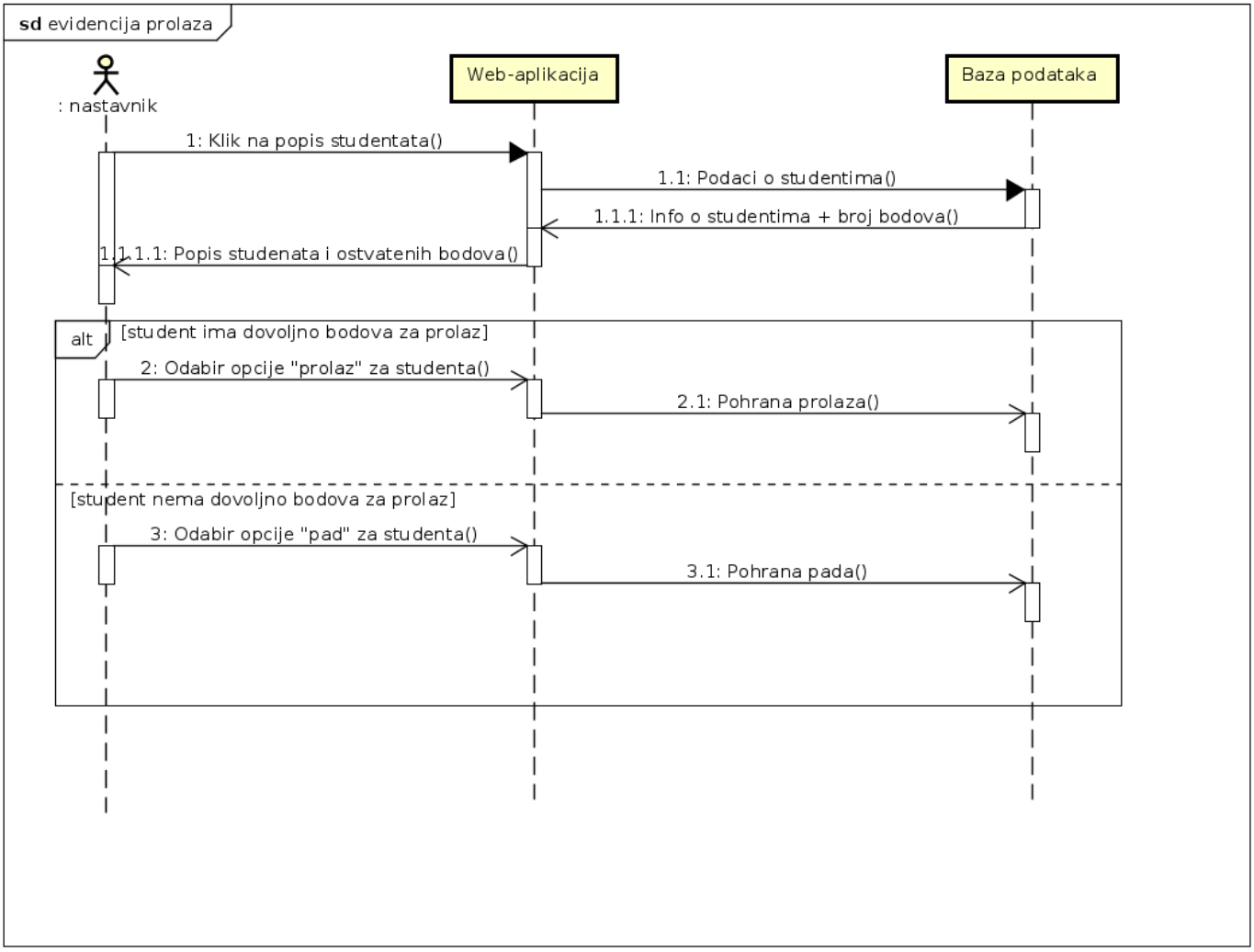
SD4 - evidencija termina:

Aplikacija prikazuje postojeće termine korisniku. Voditelj sekcije odabire jedan od termina te aplikacija dohvaća iz baze podataka podatke o terminu i prikazuje na korisničkom sučelju popis prijavljenih studenata. U slučaju da je student bio na terminu, voditelj sekcije potvrđuje prisustvo i aplikacija pohranjuje informaciju o prisustvu u bazu podataka. U protivnom, voditelj sekcije bilježi neprisustvo studenta na terminu te aplikacija bilježi neprisustvo u bazu podataka.



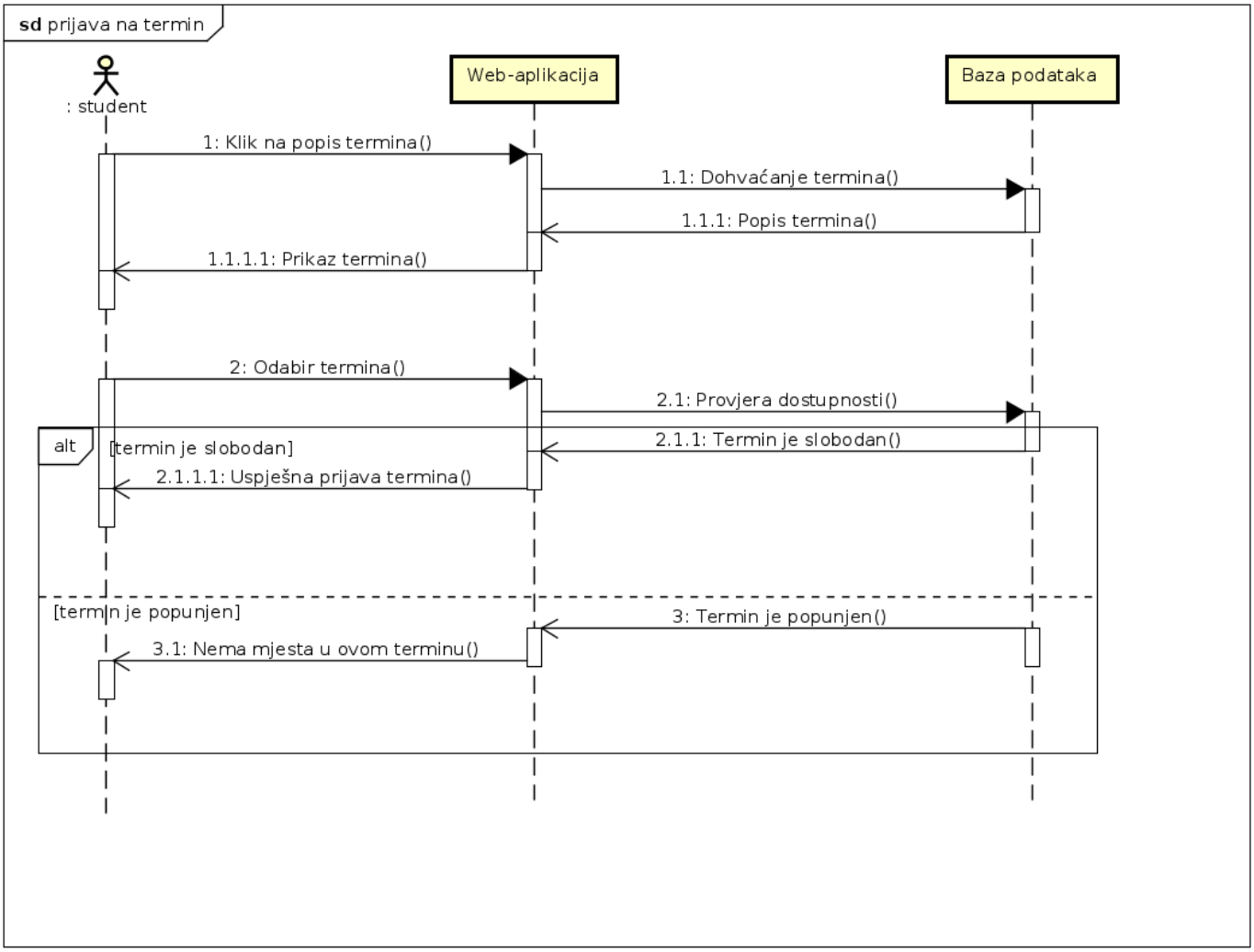
SD5 - evidencija prolaska:

Nastavnik pritišće opciju za prikaz popisa studenata. Aplikacija uzima iz baze podataka podatke o studentima i broju bodova te ih prikazuje na korisničkom sučelju. Ukoliko student ima dovoljno bodova za prolaz, Nastanik odabire opciju prolaska te aplikacija bilježi njegov odabir u bazu podataka. Ako student nema dovoljno bodova za prolaz kolegija, nastavnik bira opciju pada i aplikacija bilježi njegov odabir u bazu podataka.



SD6 - prijava termina:

Student bira opciju pregleda popisa termina. Aplikacija dohvaća podatke o postojećim terminima iz baze podataka i ispisuje termine na korisničko sučelje. Student bira jedan od termina, na što aplikacija provjerava s bazom podataka dostupnost odabranog termina. Ukoliko je termin slobodan, aplikacija sazna od baze podataka da ima mjesta te javlja studentu da se uspješno prijavio na termin. Ako je kapacitet popunjen, aplikacija javlja studentu da više nema mjestu u tom terminu.



Provjera uključenosti

ID zahtjeva	Opis	Prioritet	Izvor	Use Case
FS-01	Sustav omogućuje registraciju unosom osobnih podataka i matičnog fakulteta	Visok	Zahtjev dionika	UC1
FS-02	Sustav omogućuje pregled i pristup aktivnostima svog fakulteta	Visok	Postojeći sustav	UC2
FS-03	Sustav omogućuje prijavu na nadolazeće termine do 24 sata prije početka	Visok	Zahtjev dionika	UC3
FS-04	Sustav omogućuje pregled profila studenta	Visok	Postojeći sustav	X

ID zahtjeva	Opis	Prioritet	Izvor	Use Case
FS-05	Sustav šalje obavijesti korisniku o dolasku ili nedolasku na termin, odnosno evidentiranim bodovima koje je skupio	Visok	Povratne informacije korisnika	UC4.1
FV-01	Sustav omogućuje stvaranje termina za aktivnosti od strane voditelja sekcije	Visok	Zahtjev dionika	UC6
FV-02	Sustav omogućuje evidentiranje bodova od strane voditelja sekcije	Visok	Zahtjev dionika	UC4
FN-01	Sustav omogućuje određivanje bodova za pojedine aktivnosti i ukupnog broja bodova za prolazak kolegija te dodjeljivanje bodova nakon termina	Visok	Zahtjev dionika	UC6.1, UC6.2
FN-02	Sustav omogućuje pregled svih studenata na njihovom kolegiju od strane nastavnika	Visok	Postojeći sustav	UC7
FN-03	Sustav omogućuje pregled i pristup podatcima studenata od strane nastavnika	Srednji	Postojeći sustav	UC7
FN-04	Sustav omogućuje bilježenje prolaska kolegija na kraju semestra od strane nastavnika	Visok	Postojeći sustav	UC7.1

FaksFit

Pages 13

Find a page...

Home

1. Opis projektnog zadatka

2. Analiza zahtjeva

3. Specifikacija zahtjeva sustava

Obrasci uporabe
Use case diagram 1
UC1 - Prijava
UC1.1 - OAuth prijava
UC2 - Pregled termina
UC3 - Odabir termina
UC4 - Potvrđuje dolazak studenta na termin
UC4.1 - Dobiva obavijest o prisutnosti
UC5 - Podnosi zahtjev da bude voditelj
UC5.1 - Potvrđuje zahtjev za voditeljem sekcije
UC6 - Kreira termin za aktivnost
UC6.1 - Uređuje bodove za aktivnost
UC6.2 - Potvrđuje bodove za aktivnost
UC7 - Upisuje prolaz ili pad svakom studentu
UC7.1 - Dobiva obavijest o prolasku tjelesnog
Use case diagram 2
UC6 - Kreira termin za aktivnost
UC6.1 - Određuje broj bodova za prolaz
UC6.2 - Određuje lokaciju
UC6.3 - Određuje kapacitet po treningu
UC6.4 - Određuje vrijeme termina
Sekvencijski dijagrami
SD1 - prijava:
SD2 - registracija:
SD3 - pregled dostupnih termina:
SD4 - evidencija termina:
SD5 - evidencija prolaska:
SD6 - prijava termina:
Provjera uključenosti

4. Arhitektura i dizajn sustava

5. Arhitektura komponenata i razmještaja

6. Ispitivanje programskog rješenja

7. Tehnologije za implementaciju aplikacije

8. Upute za puštanje u pogon

9. Zaključak i budući rad

A. Dnevnik promjena dokumentacije

A. Popis literature

B. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```

4. Arhitektura i dizajn sustava

dean-rakic edited this page 5 hours ago · 6 revisions

Arhitektura sustava će se temeljiti na stilističkoj varijaciji ahitekture zasnovane na događajima – **MVC** (engl. Model View Controller). MVC obrazac naglašava ulogu događaja u upravljanju interakcijama korisnika i odgovorima sustava te promiče modularnost, održivost i skalabilnost u razvoju aplikacija jasnim razgraničenjem odgovornosti između komponenti. Odvaja korisničko sučalje od ostatka sustava i na taj način smanjuje međuovisnost (engl. Coupling).

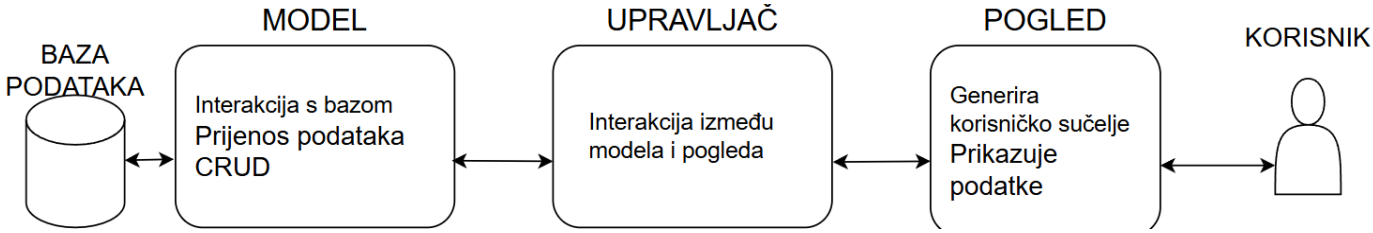
MVC obrazac je podijeljen na 3 komponente:

- **Model** - služi za interakciju s bazom podataka i prijenos potrebnih podataka upravljaču, koristi **CRUD** (engl. Create, Read, Update, Delete) operacije za upravljanje podacima, obavještava upravljača o promjenama stanja programa
- **Pogled** - koristi se za svu logiku korisničkog sučelja aplikacije, generira korisničko sučelje za korisnika I prikazuje podatke korisniku
- **Upravljač** - sadrži razrede koji upravljaju i rukuju korisničkom interakcijom s pogledom i modelom te služi kao poveznica između pogleda i modela, ne brine se o logici upravljanja podacima već samo šalje zahtjev modelu što treba učiniti

Arhitektura će biti podijeljena u 3 podsustava:

- **web poslužitelj**
- **web aplikacija**
- **baza podataka**

Web poslužitelj je komponenta koja upravlja **HTTP** (engl. Hyper Text Transfer Protocol) zahtjevima korisnika i poslužuje dostavljene podatke ili prosljeđuje zahtjeve prema web aplikaciji.
Web aplikacija obrađuje poslovnu logiku, obavlja autentifikaciju i autorizaciju korisnika, čita i modificira podatke te generira odgovore koje web posužitelj prikazuje korisniku.
Baza podataka se koristi za dugotrajnu pohranu podataka koje koristi web aplikacija.



Web aplikacija je izrađena koristeći sljedeće tehnologije:

- **Backend:** Razvijen u **Java** programskom jeziku koristeći **Spring Boot** framework, koji upravlja podacima i omogućuje definiranje modela podataka te poslovne logike aplikacije.
- **Frontend:** Izrađen pomoću **JavaScript** i **React** frameworka, omogućavajući brzo i efikasno kreiranje dinamičkih i interaktivnih korisničkih sučelja.

Ovakva podjela omogućuje da backend obrađuje logiku i podatke, dok frontend upravlja interakcijom s korisnikom.

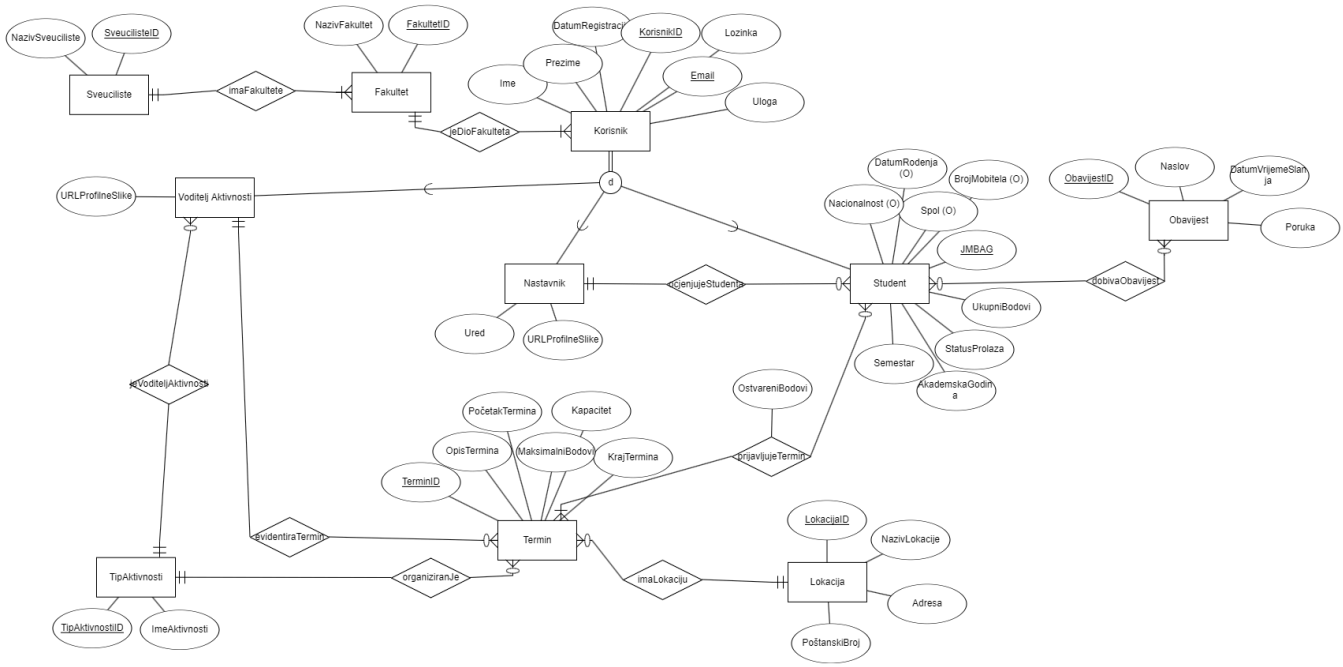
Kao razvojno okruženje koristimo **IntelliJ IDEA**.

Baza podataka

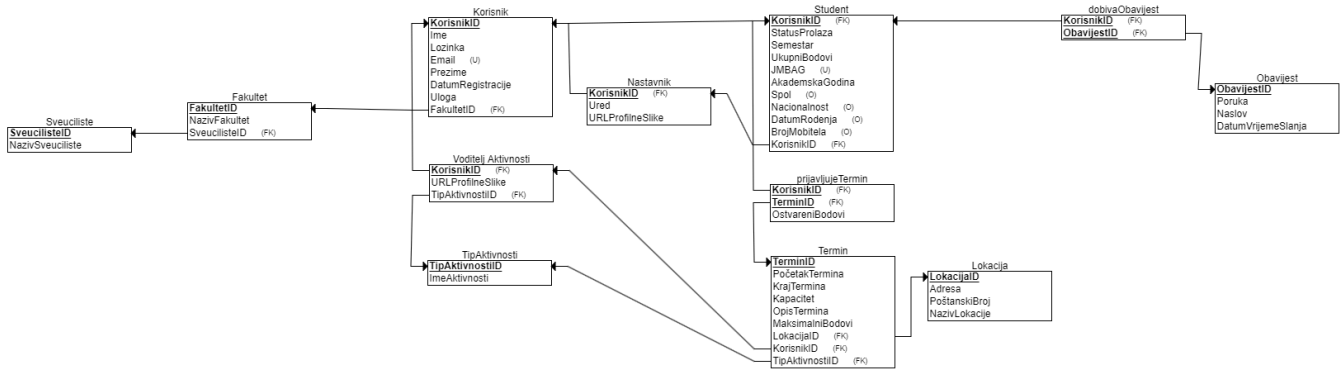
U našoj aplikaciji koristit ćemo relacijsku bazu podataka kako bismo olakšali modeliranje zadatka. Relacijska baza podataka organizirana je kao skup međusobno povezanih tablica. Svaka tablica predstavlja jednu entitetnu klasu, svaki redak predstavlja jedan objekt, a stupci predstavljaju njegove atribute. Svaki objekt ima jedinstveni identifikator nazvan **primarni ključ**, te može imati **strane ključeve** koji referenciraju primarne ključeve drugih objekata. Baza podataka omogućuje brzu i jednostavnu pohranu podataka koje možemo kasnije lako dohvatiti ili izmijeniti.

Baza podataka naše aplikacije sadrži sljedeće entitete:

- Sveučilište
- Fakultet
- Korisnik
- Uloga
- Obavijest
- Voditelj Aktivnosti
- Nastavnik
- Student
- Tip Aktivnosti
- Termin
- Lokacija



ER dijagram baze podataka



Relacijska shema baze podataka

Opis Tablica u Bazi Podataka

Tablica: `Sveuciliste`

Ovaj entitet sadrži informacije o sveučilištu. Sadrži atribute: SveucilisteID i NazivSveuciliste. Sveučilište može biti nadležno za više fakulteta.

Naziv Stupca	Tip Podatka	Opis
`SveucilisteID` (PK)	LONG	Jedinstveni identifikator sveučilišta
`NazivSveuciliste`	VARCHAR	Naziv sveučilišta

- **Veze**
  - 1:N veza s Fakultet

Tablica: `Fakultet`

Ovaj entitet sadrži informacije o fakultetu. Sadrži atribute: FakultetID, NazivFakultet i SveucilisteID. SveucilisteID je strani ključ koji predstavlja nadležno sveučilište fakulteta.

Naziv Stupca	Tip Podatka	Opis
`FakultetID` (PK)	LONG	Jedinstveni identifikator fakulteta
`NazivFakultet`	VARCHAR	Naziv fakulteta
`SveucilisteID` (FK)	VARCHAR	Identifikator nadležnog sveučilišta

- **Veze**
  - 1:N veza s Korisnik
  - N:1 veza sa Sveuciliste

Tablica: `Korisnik`

Ovaj entitet sadrži informacije o korisniku. Sadrži atribute: KorisnikID, Ime, Prezime, Email, Lozinka, DatumRegistracije, Uloga i FakultetID. FakultetID je strani ključ koji predstavlja fakultet za koji radi ili koji pohađa korisnik. Korisnik je *supertype* koji ima disjunktne podređene entitete Voditelj Aktivnosti, Nastavnik i Student.

Naziv Stupca	Tip Podatka	Opis
`KorisnikID` (PK)	LONG	Jedinstveni identifikator korisnika
`Ime`	VARCHAR	Ime korisnika
`Prezime`	VARCHAR	Prezime korisnika
`Email`	VARCHAR	Email adresa korisnika
`Lozinka`	VARCHAR	Zaštićena lozinka korisnika
`DatumRegistracije`	DATE	Datum registracije korisnika
`Uloga`	VARCHAR	Uloga korisnika na fakultetu (ENUM)
`FakultetID` (FK)	LONG	Identifikator fakulteta za koji radi ili kojeg pohađa korisnik

- **Veze**
  - N:1 veza s Fakultet
  - supertype:subtype veza s Voditelj Aktivnosti
  - supertype:subtype veza s Nastavnik

- supertype:subtype veza sa Student

Tablica: `dobivaObavijest`

Join tablica za N:N vezu između Student i Obavijest. KorisnikID i ObavijestID su strani ključevi.

Naziv Stupca	Tip Podatka	Opis
`KorisnikID` (PK) (FK)	LONG	Jedinstveni identifikator korisnika
`ObavijestID` (PK) (FK)	LONG	Jedinstveni identifikator obavijesti

- **Veze**
  - N:1 veza sa Student preko atributa KorisnikID
  - N:1 veza s Obavijest preko atributa ObavijestID

Tablica: `Obavijest`

Ovaj entitet sadrži sve informacije o obavijesti koju dobivaju korisnici. Sadrži atribute: ObavijestID, Naslov, Poruka i DatumVrijemeSlanja.

Naziv Stupca	Tip Podatka	Opis
`ObavijestID` (PK)	LONG	Jedinstveni identifikator obavijesti
`Poruka`	VARCHAR	Tekstni sadržaj obavijesti
`Naslov`	VARCHAR	Naslov obavijesti
`DatumVrijemeSlanja`	DATETIME	Datum i vrijeme slanja obavijesti

- **Veze**
  - N:N veza sa Student

Tablica: `Voditelj Aktivnosti`

Ovaj entitet sadrži sve informacije o voditelju aktivnosti. Sadrži atribute: KorisnikID, URLProfilneSlike i TipAktivnostiID. KorisnikID i TipAktivnostiID su strani ključevi. Voditelj aktivnosti voditelj je jedne aktivnosti, ali može postojati više voditelja jedne aktivnosti.

Naziv Stupca	Tip Podatka	Opis
`KorisnikID` (PK) (FK)	LONG	Jedinstveni identifikator korisnika
`URLProfilneSlike`	VARCHAR	URL na profilnu sliku voditelja aktivnosti s vanjskog servisa

Naziv Stupca	Tip Podatka	Opis
`TipAktivnostiID` (FK)	LONG	Identifikator tipa aktivnosti koje voditelj vodi

- **Veze**
  - 1:1 veza s Korisnik preko atributa KorisnikID
  - N:1 veza s TipAktivnosti
  - 1:N veza s Termin

Tablica: `Tip Aktivnosti`

Ovaj entitet sadrži informacije o aktivnosti kolegija TZK. Sadrži atribute: TipAktivnostiID i ImeAktivnosti. Jedan tip aktivnosti može imati više voditelja.

Naziv Stupca	Tip Podatka	Opis
`TipAktivnostiID` (PK)	LONG	Jedinstveni identifikator tipa aktivnosti
`ImeAktivnosti`	VARCHAR	Naziv aktivnosti

- **Veze**
  - 1:N veza s Voditelj Aktivnosti
  - 1:N veza s Termin

Tablica: `Nastavnik`

Ovaj entitet sadrži sve informacije o nastavniku. Sadrži atribute: KorisnikID, ProfilnaSlika i Ured. KorisnikID je strani ključ. Jedan nastavnik zadužen je za ocjenjivanje studenata (potvrđivanje statusa prolaza na temelju bodova) i nadgledanje voditelja aktivnosti.

Naziv Stupca	Tip Podatka	Opis
`KorisnikID` (PK) (FK)	LONG	Jedinstveni identifikator korisnika
`ProfilnaSlika`	VARCHAR	Profilna slika voditelja aktivnosti
`Ured`	VARCHAR	Lokacija ureda nastavnika

- **Veze**
  - 1:1 veza s Korisnik preko atributa KorisnikID
  - 1:N veza s Student

Tablica: `Student`

Ovaj entitet sadrži sve informacije o studentu. Sadrži atribute: KorisnikID, StatusProlaza, UkupniBodovi, JMBAG, Semestar, AkademskaGodina, Spol, Nacionalnost, DatumRodenja i BrojMobitela. KorisnikID je strani ključ. Više studenata ocjenjuje jedan nastavnik. Jedan student može upisati više termina, a svaki termin može imati više studenata.

Naziv Stupca	Tip Podatka	Opis
`KorisnikID` (PK)	LONG	Jedinstveni identifikator korisnika (studenta)
`StatusProlaza`	BOOLEAN	Zastavica prolaza/pada studenta
`UkupniBodovi`	INT	Ukupni ostvareni bodovi studenta u semestru
`JMBAG` (U)	VARCHAR	Jedinstveni identifikator studenta
`Semestar`	INT	Upisani semestar studenta
`AkademskaGodina`	VARCHAR	Akademska godina upisa tjelesnoga
`Spol`	VARCHAR	Spol studenta
`Nacionalnost`	VARCHAR	Nacionalnost studenta
`DatumRodenja`	VARCHAR	Datum rođenja studenta
`BrojMobitela`	VARCHAR	Broj mobitela studenta
`KorisnikID` (FK)	LONG	Jedinstveni identifikator korisnika (nastavnika)

- **Ograničenja**
  - JMBAG mora biti jedinstven i isključivo brojevi
  - Akademska godina mora biti valjana i ne smije biti u budućnosti
  - UkupniBodovi ne smiju premašiti prije definirani maksimum ili ići ispod nule
- **Veze**
  - 1:1 veza s Korisnik preko atributa KorisnikID
  - N:1 veza s Nastavnik
  - N:N veza s Termin
  - N:N veza s Obavijest

Tablica: `prijavljujeTermin`

Join tablica za N:N vezu između Student i Termin. KorisnikID i TerminID su strani ključevi.

Naziv Stupca	Tip Podatka	Opis
`KorisnikID` (PK) (FK)	LONG	Jedinstveni identifikator korisnika

Naziv Stupca	Tip Podatka	Opis
`TerminID` (PK) (FK)	LONG	Jedinstveni identifikator termina
`OstvareniBodovi`	INT	Broj ostvarenih bodova u terminu

- **Veze**
  - N:1 veza s Student preko atributa KorisnikID
  - N:1 veza s Termin preko atributa TerminID

Tablica: `Termin`

Ovaj entitet sadrži sve informacije o terminu. Sadrži atribute: TerminID, PocetakTermina, KrajTermina, Kapacitet, OpisTermina, MaksimalniBodovi, LokacijaID, KorisnikID i TipAktivnostiID. LokacijaID, KorisnikID i TipAktivnostiID su strani ključevi. Više studenata može prijaviti više termina. Jedan voditelj aktivnosti može evidentirati (stvarati i evdidentirati dolaznosti) više termina. Više termina mogu biti na istoj lokaciji.

Naziv Stupca	Tip Podatka	Opis
`TerminID` (PK)	LONG	Jedinstveni identifikator termina
`PocetakTermina`	DATETIME	Vrijeme početka termina
`KrajTermina`	DATETIME	Vrijeme kraja termina
`Kapacitet`	INT	Kapacitet termina
`OpisTermina`	VARCHAR	Tekstualni opis termina
`MaksimalniBodovi`	INT	Maksimalni broj bodova ostvariv na terminu
`LokacijaID` (FK)	LONG	Jedinstveni identifikator lokacije termina
`KorisnikID` (FK)	LONG	Jedinstveni identifikator korisnika
`TipAktivnostiID` (FK)	LONG	Jedinstveni identifikator tipa aktivnosti

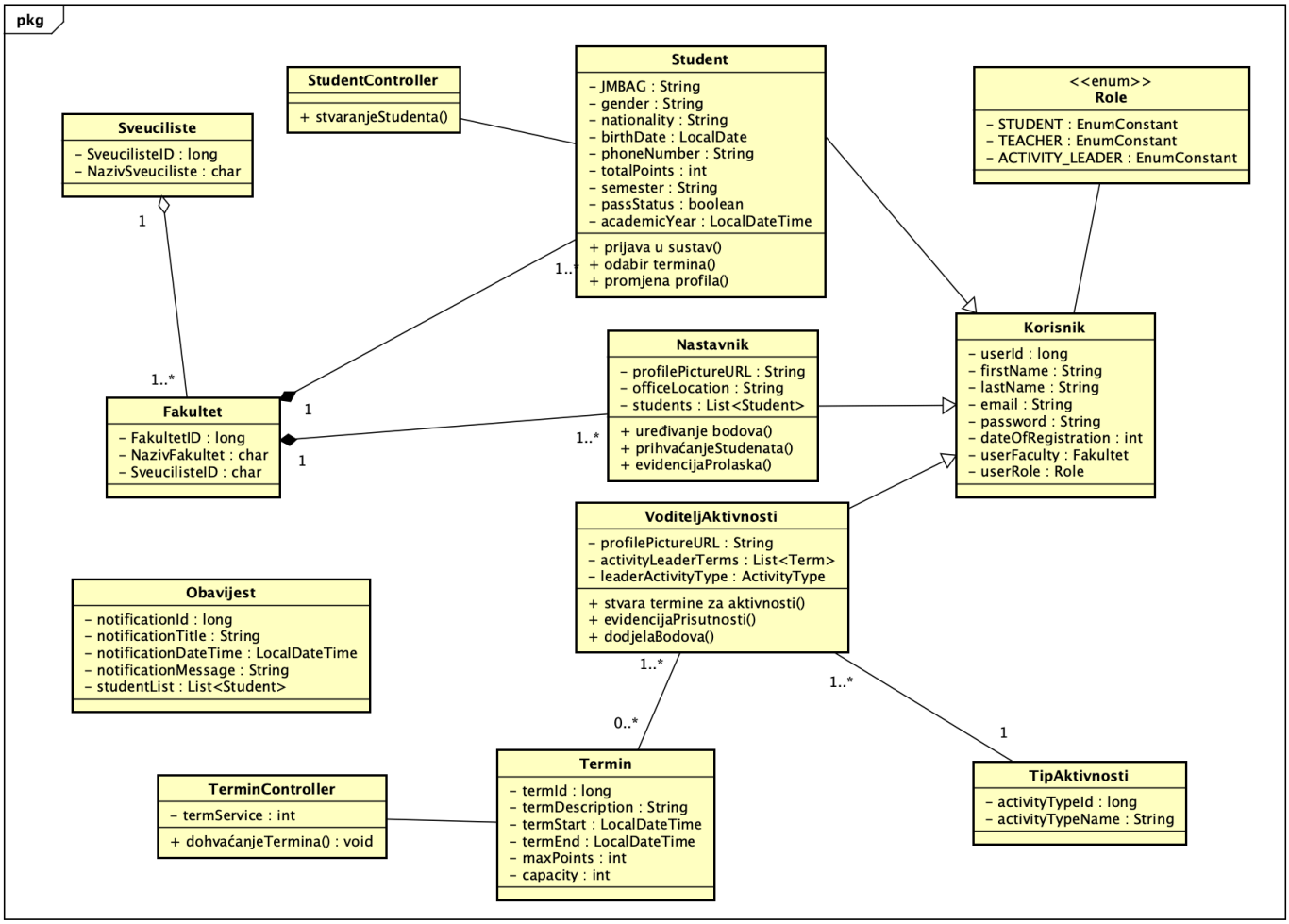
- **Veze**
  - N:1 veza s Voditelj Aktivnosti
  - N:1 veza s TipAktivnosti
  - N:1 veza s Lokacija
  - N:N veza s Student

Tablica: `Lokacija`

Ovaj entitet sadrži sve informacije o lokaciji. Sadrži atribute: LokacijaID, NazivLokacije, Adresa, PostanskiBroj. Na jednoj lokaciji može biti više termina.

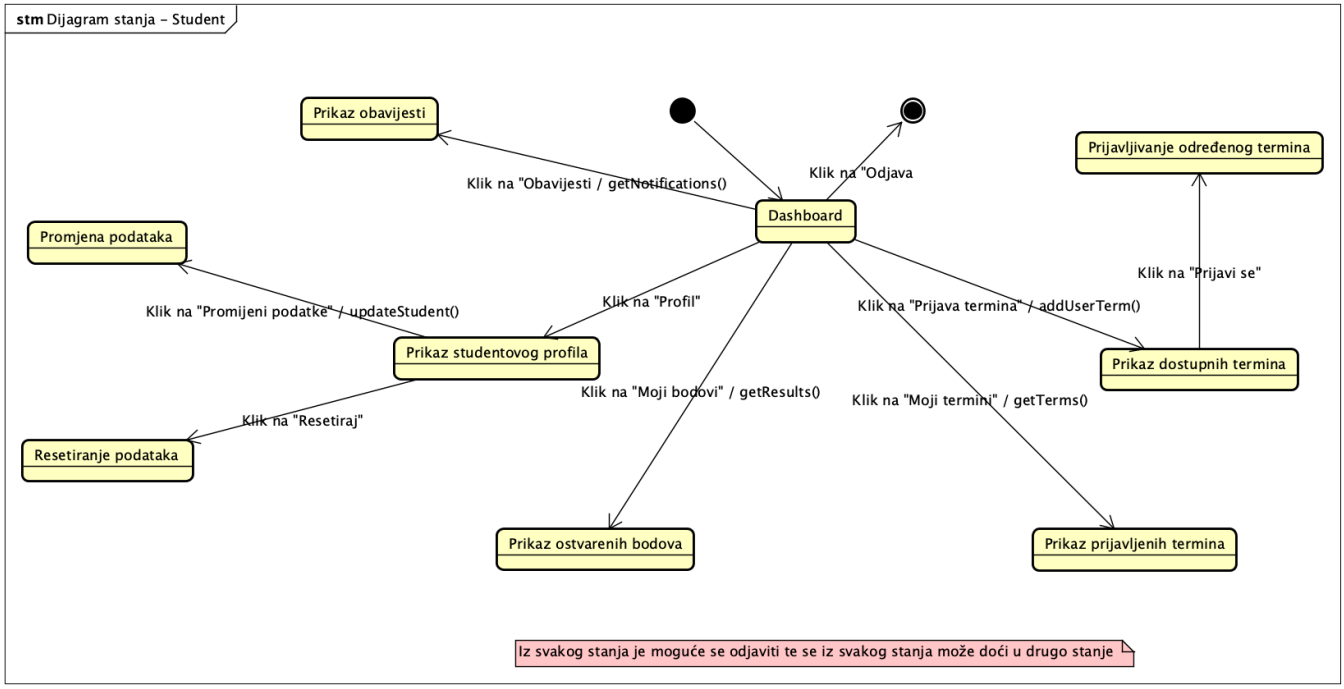
Naziv Stupca	Tip Podatka	Opis
`LokacijaID` (PK)	LONG	Jedinstveni identifikator lokacije
`NazivLokacije`	VARCHAR	Naziv lokacije
`Adresa`	VARCHAR	Adresa lokacije
`PostanskiBroj`	INT	Poštanski broj lokacije

- **Veze**
  - 1:N veza s Termin



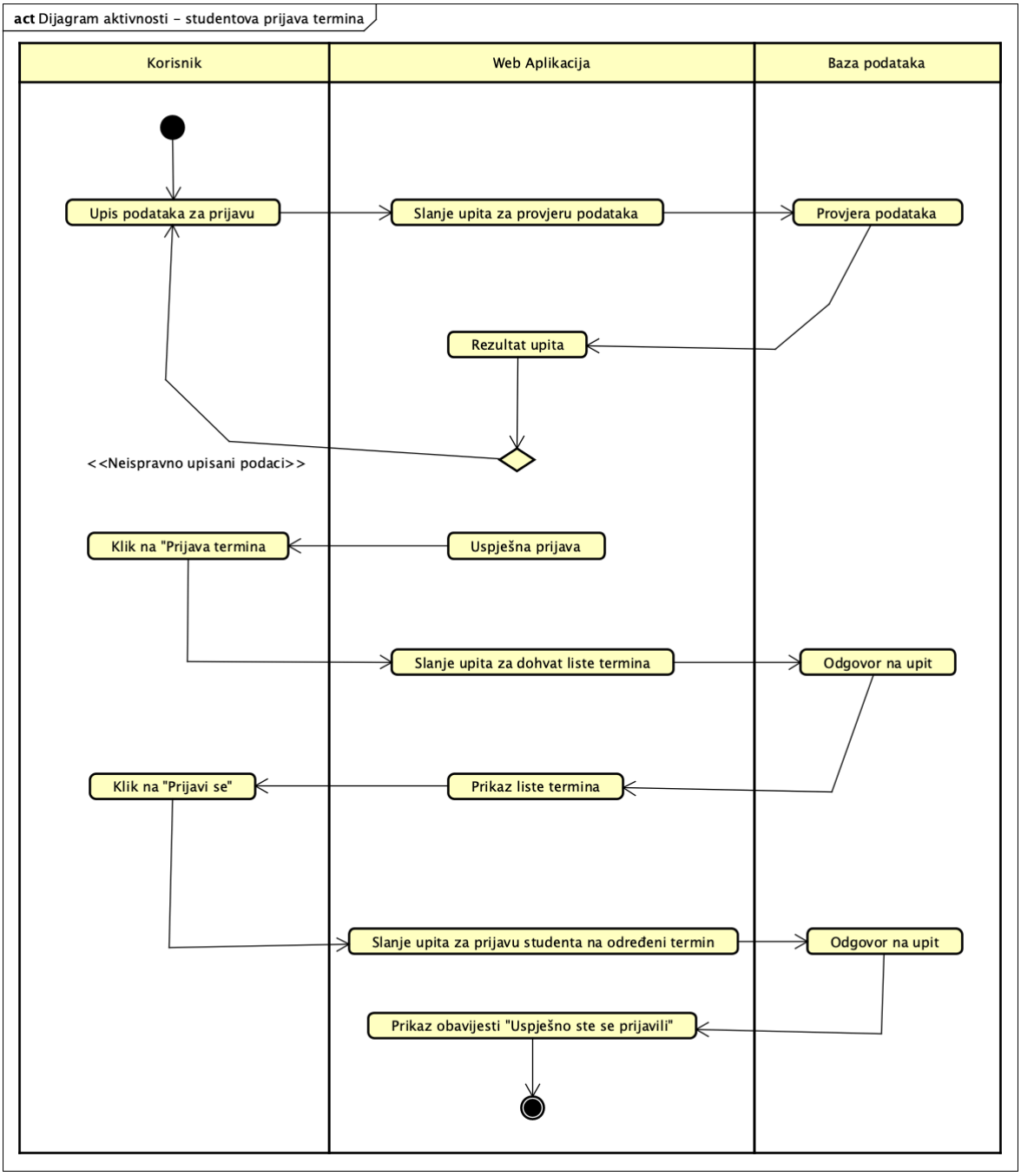
Dijagram razreda :

Dijagram razreda prikazuje osnovne klase projekta i njihove funkcionalnosti. U dijagramu se nalaze klase Student, Teacher i ActivityLeader koje služe kao različiti oblici korisnika u aplikaciji. U njihovim metodama se nalazi većina funkcionalnosti koju želimo omogućiti korisnicima. Ubačene su klase StudentController i TermController koje se trenutno nalaze u projektu. Prikazane su i veze između klasa i njihovi odnosi.



Dijagram stanja

Dijagram stanja prikazuje promjene stanja za studenta nakon uspješne prijave. Prvo što se pokaže studentu je prikaz Dashboard-a. S Dashboard-a se može ući u brojna druga stanja. Pritiskom na "Obavijesti", student može pregledati pristigle obavijesti relevantne njemu. Pritiskom na "Moji termini" odnosno "Moji bodovi", studentu su prikazani prijavljeni termini odnosno dosad ostvareni bodovi. Student može odabrati "Prijava termina", gdje dodatno može prijaviti neki od dostupnih termina. Može se ući i u pregled profila i podataka studenta, gdje isto tako postoji opcija resetiranja odnosno promjene podataka.



Dijagram aktivosti

Na dijagramu aktivnosti može se vidjeti kako izgleda proces prijave termina od strane studenta. Student upisuje korisničke podatke, koje aplikacija provjerava s bazom podataka i u slučaju uspješne prijave, mu daje mogućnost prijave termina. U protivnom ga šalje nazad na prikaz upisa korisničkih podataka. Odabirom prijava termina, aplikacija dobavlja iz baze podataka listu trenutno dostupnih termina i prikazuje ih studentu. Klik-om na "Prijavi se", student šalje aplikaciji upit da se želi prijaviti za određeni termin. Aplikacija u komunikaciji s bazom podataka sprema informaciju o studentovoj prijavi te mu javlja preko notifikacije da se uspješno prijavio.

FaksFit

Pages 13

Find a page...

- Home
- 1. Opis projektnog zadatka
- 2. Analiza zahtjeva
- 3. Specifikacija zahtjeva sustava
- 4. Arhitektura i dizajn sustava
  - Baza podataka
    - Opis Tablica u Bazi Podataka
      - Tablica: Sveuciliste
      - Tablica: Fakultet
      - Tablica: Korisnik
      - Tablica: dobivaObavijest
      - Tablica: Obavijest
      - Tablica: Voditelj Aktivnosti
      - Tablica: Tip Aktivnosti
      - Tablica: Nastavnik
      - Tablica: Student
      - Tablica: prijavljujeTermin
      - Tablica: Termin
      - Tablica: Lokacija
    - Dijagram razreda :
    - Dijagram stanja
    - Dijagram aktivosti
- 5. Arhitektura komponenata i razmještaja
- 6. Ispitivanje programskog rješenja
- 7. Tehnologije za implementaciju aplikacije
- 8. Upute za puštanje u pogon
- 9. Zaključak i budući rad
- A. Dnevnik promjena dokumentacije
- A. Popis literature

B. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```

eugen-vucelic / FaksFit

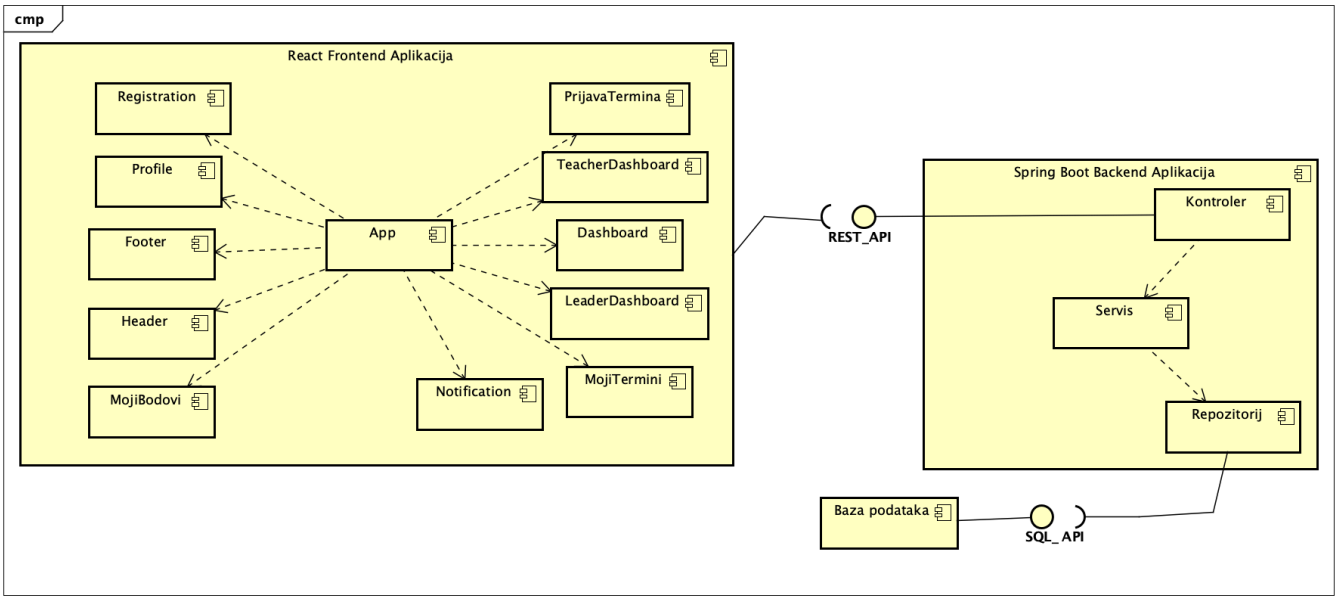
Code Issues 9 Pull requests Actions Projects 1 Wiki Secu

5. Arhitektura komponenata i razmještaja

Edit New page Jump to bottom

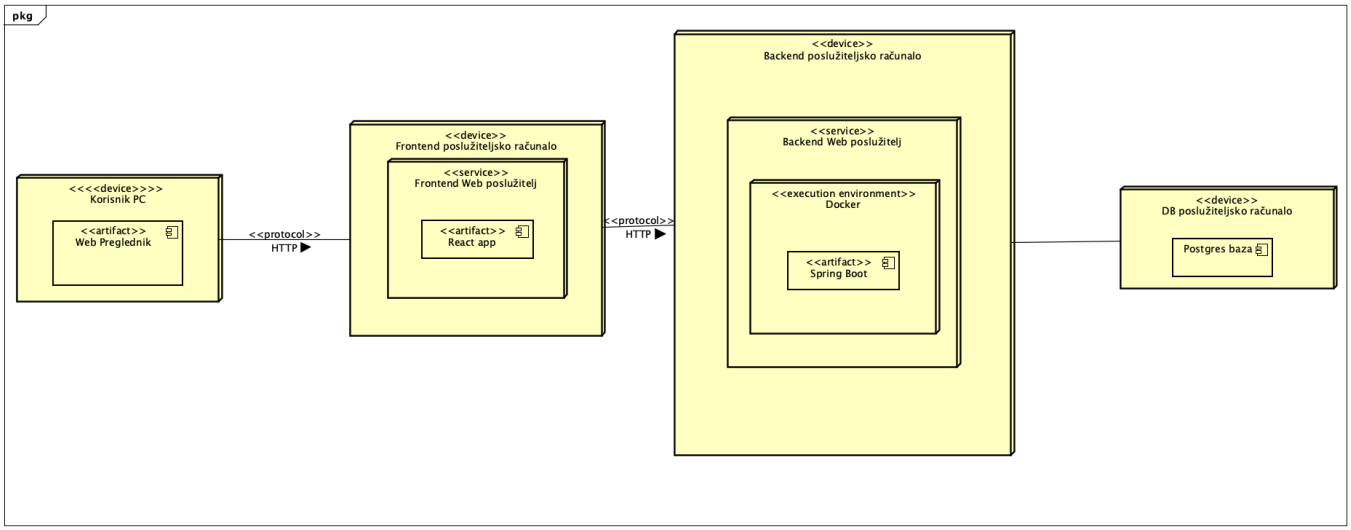
dean-rakic edited this page 5 hours ago · 6 revisions

Arhitektura komponenata i razmještaja



Dijagram komponenata

Dijagram komponenata prikazuje Spring Boot backend aplikaciji pomoću React frontend aplikacije i sučelja REST_API. Backend aplikacija je podijeljena na 3 komponente, Kontroler, Servis i Repozitorij. Kontroler zove Servis koji putem Repozitorija dobavlja podatke iz baze preko sučelja SQL_API. U React frontend aplikaciji, App se sastoji se od više JavaScript(.jsx) datoteka koje određuju prikaze određenih elemenata na stranici za prijavljenog korisnika.



Dijagram razmještaja

Dijagram razmještaja prikazuje topologiju sklopovlja i programsku potporu web-aplikacije. Komunikacija između korisnika i frontend poslužiteljskog računala te frontend poslužiteljskog računala i backend poslužiteljskog računala se događa preko HTTP veze. Kada korisnici pristupe aplikaciji, frontend Web poslužitelj pruža određeni prikaz, ovisno čemu se pristupi. Na backend poslužiteljskom računalu se nalazi Docker koji stavlja cijelu aplikaciju u kontejner, gdje onda izgradi Spring Boot aplikaciju i pokrene ju. Backend poslužiteljsko računalo pristupa Postgres bazi koja se nalazi na DB poslužiteljskom računalu.

FaksFit

Pages 13

Find a page...

Home

1. Opis projektnog zadatka

2. Analiza zahtjeva

3. Specifikacija zahtjeva sustava

4. Arhitektura i dizajn sustava

5. Arhitektura komponenata i razmještaja

Arhitektura komponenata i razmještaja

Dijagram komponenata

Dijagram razmještaja

- 6. Ispitivanje programskog rješenja
- 7. Tehnologije za implementaciju aplikacije
- 8. Upute za puštanje u pogon
- 9. Zaključak i budući rad
- A. Dnevnik promjena dokumentacije
- A. Popis literature
- B. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```

6. Ispitivanje programskog rješenja

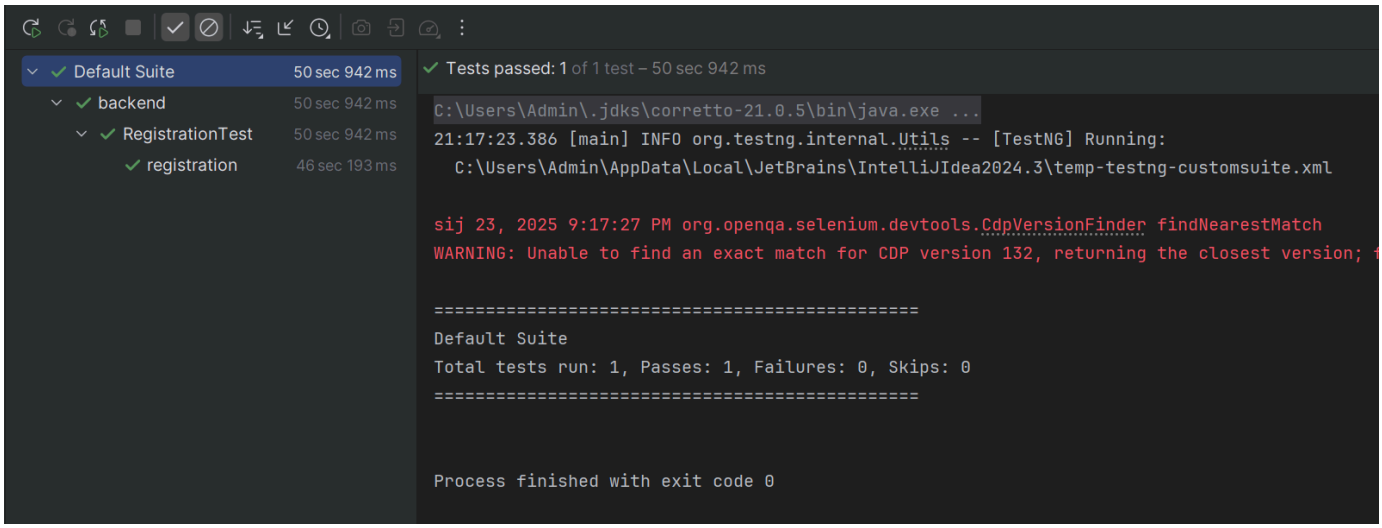
Lucija Burić edited this page yesterday · 2 revisions

Integracijski testovi

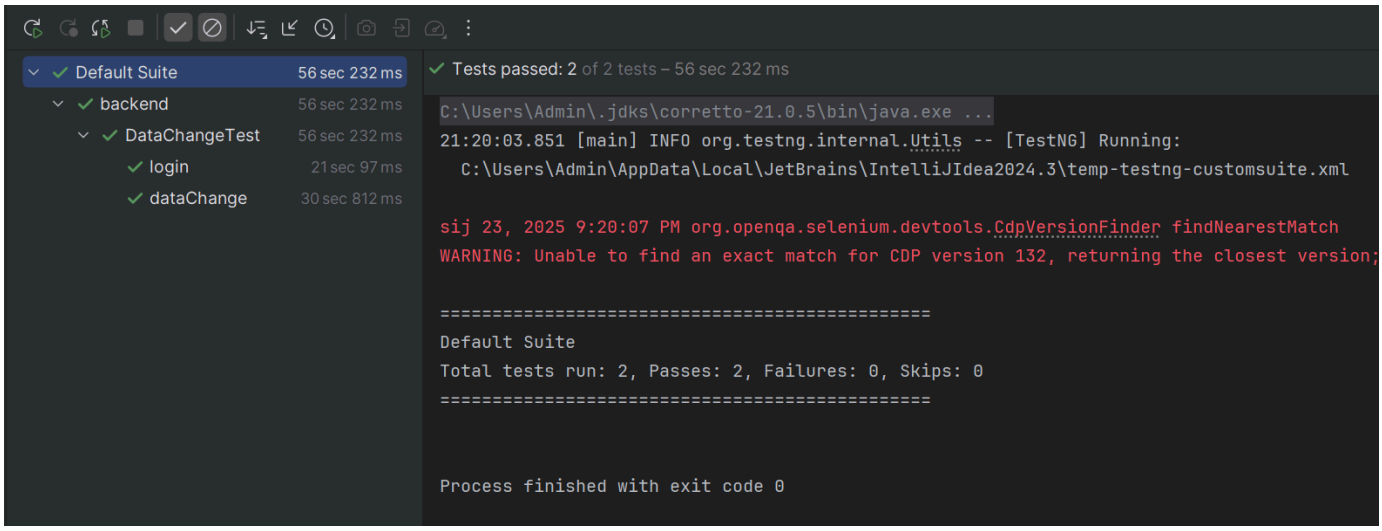
Integracijski testovi su napisani pomoću **Seleniuma**, alata otvorenog koda za automatizaciju web preglednika, kako bi se testiralo ponašanje aplikacije u stvarnim preglednicima. Cilj ovih testova je osigurati da različiti dijelovi aplikacije (frontend, backend, baze podataka itd.) ispravno rade zajedno kao cjelina.

Selenium omogućuje simulaciju stvarnih korisničkih interakcija s aplikacijom, kao što su klikanje gumba, unos teksta u obrasce, navigacija kroz stranice, odabir opcija iz izbornika, provođenje prijava i odjava.

Test `registration` provjerava da se registracija OAuth2 Google klijentom odvija ispravno.

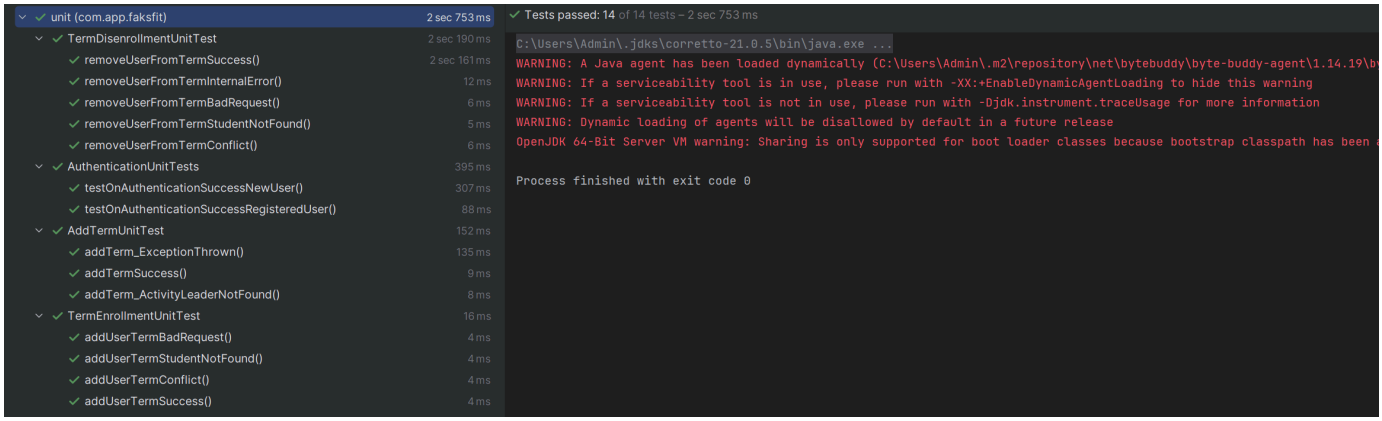


Test `login` provjerava da se prijava OAuth2 Google klijentom odvija ispravno. Test `dataChange` provjerava da prijavljeni korisnik tipa Student može unijeti izmjene u svoj profil.



Unit testovi

Unit testovi napisani su pomoću **JUnit-a**, frameworka za testiranje Java aplikacija. Unit testovi se koriste za automatsko testiranje koda kako bi se osiguralo da pojedine metode, klase ili funkcionalnosti rade ispravno.



`AddTermUnitTest`

Testovi u klasi `AddTermUnitTest` testiraju metodu `addTerm` iz klase `ActivityLeaderController` koja omogućuje voditelju aktivnosti da dodaje novi termin svoje aktivnosti na koje se studenti mogu prijaviti.

Test `addTermSuccess` provjerava uspješnost dodavanja novog termina kad su svi uvjeti zadovoljeni.

- stvara primjerak `NoviTerminDTO` s detaljima termina (početak, kraj, kapacitet itd.)
- postavlja ispravan JWT token i pripadajući email
- "mockira" korisnika (`User`) i voditelja aktivnosti (`ActivityLeader`)
- "mockira" metodu `jwtUtil.extractEmail` tako da vraća email iz tokena
- "mockira" metodu `userService.findByEmail` tako da vraća korisnika
- "mockira" metodu `activityLeaderService.getById` tako da vraća voditelja aktivnosti

- poziva metodu kontrolera `addTerm`
- provjerava da je odgovor `OK (200)`
- provjerava da je tijelo odgovora "Term added successfully"
- provjerava da su sve relevantne metode mockova pozvane točno onoliko puta koliko je očekivano

Test `addTermActivityLeaderNotFound` testira ponašanje metode kada voditelj aktivnosti ne postoji u sustavu.

- stvara primjerak `NoviTerminDTO` s detaljima termina (početak, kraj, kapacitet itd.)
- postavlja ispravan JWT token i pripadajući email
- "mockira" korisnika (`User`) i voditelja aktivnosti (`ActivityLeader`)
- "mockira" metodu `jwtUtil.extractEmail` tako da vraća email iz tokena
- "mockira" metodu `userService.findByEmail` tako da vraća korisnika
- "mockira" metodu `activityLeaderService.getById` tako da vraća *null*, što označava da voditelj aktivnosti nije pronađen
- poziva metodu kontrolera `addTerm`
- provjerava da je odgovor `NOT_FOUND (404)`
- provjerava da je tijelo odgovora `null`
- provjerava da metoda `termService.addTerm` nikad nije pozvana, jer voditelj aktivnosti nije pronađen

Test `addTermExceptionThrown` testira ponašanje metode kada se baci iznimka prilikom dodavanja termina.

- stvara primjerak `NoviTerminDTO` s detaljima termina (početak, kraj, kapacitet itd.)
- postavlja ispravan JWT token i pripadajući email
- "mockira" korisnika (`User`) i voditelja aktivnosti (`ActivityLeader`)
- "mockira" metodu `jwtUtil.extractEmail` tako da vraća email iz tokena
- "mockira" metodu `userService.findByEmail` tako da vraća korisnika
- "mockira" metodu `activityLeaderService.getById` tako da baca iznimku `IllegalArgumentException` s porukom "Invalid term details"
- poziva metodu kontrolera `addTerm`
- provjerava da je odgovor BAD_REQUEST (400)
- provjerava da je tijelo odgovora "Invalid term details"
- provjerava da su svi relevantni mockovi pozvani

`AuthenticationUnitTest`

Testovi u klasi `AuthenticationUnitTest` testiraju ispravnost metode `onAuthenticationSuccess` iz klase `CustomOAuth2SuccessHandler` .

Test `testOnAuthenticationSuccessNewUser` testira scenarij kada se novi korisnik (koji nije registriran u sustavu) autentificira putem OAuth2.

- "mockira" servise `StudentServiceImpl`, `UserServiceImpl` i `JWTUtil` tako da vraćaju `null`, što znači da korisnik nije pronađen u sustavu
- "mockira" HTTP zahtjev, odgovor i autentifikaciju (`HttpServletRequest`, `HttpServletResponse`, `Authentication`)
- "mockira" se `OAuth2User` koji predstavlja korisnika autentifikacijskog procesa tako da `oauth2User.getAttribute("email")` vraća email novog korisnika "newuser@example.com"
- poziva se metoda `onAuthenticationSuccess`
- provjerava da metoda `response.sendRedirect` preusmjerava korisnika na URL za registraciju s parametrom `email=newuser@example.com`

Test `testOnAuthenticationSuccessRegisteredUser` testira scenarij kada već registrirani korisnik autentificira putem OAuth2.

- "mockira" metodu `oauth2User.getAttribute("email")` tako da vraća email već registriranog korisnika "registereduser@example.com"
- "mockira" servis `studentService.findByEmail()` tako da vraća `null`, što znači da korisnik nije student
- "mockira" servis `userService.findByEmail()` tako da vraća korisnika s emailom i ulogom `STUDENT`
- "mockira" `jwtUtil.generateToken()` tako da generira mockirani JWT token "mockJwtToken"
- poziva metodu `onAuthenticationSuccess`
- provjerava se da je metoda `jwtUtil.generateToken()` pozvana za generiranje JWT tokena
- provjerava se da metoda `response.sendRedirect()` preusmjerava korisnika na početnu stranicu s parametrom `token=mockJwtToken`

`TermEnrollmentUnitTest`

Testovi u klasi `TermEnrollmentUnitTest` testiraju ispravnost metode `addUserTerm` iz klase `TermController`

Test `addUserTermSuccess` testira scenarij kada se korisnik uspješno prijavi na termin.

- "mockira" `oauthUser.getAttribute("email")` tako da vraća email korisnika "testuser@example.com"
- "mockira" `studentService.findByEmail(email)` tako da vraća instancu `Student` s odgovarajućim emailom
- poziva metodu `addUserTerm` sa zadanim `termId` i korisnikom
- provjerava da je metoda `termSignUpService.addUserToTerm(termId, student)` pozvana jednom

- provjerava da odgovor kontrolera ima status `OK (200)` i odgovarajuću poruku: "User successfully added to the term"

Test `addUserTermStudentNotFound` testira scenarij kada se korisnik nije pronađen u sustavu.

- "mockira" `oauthUser.getAttribute("email")` tako da vraća email korisnika "testuser@example.com"
- "mockira" `studentService.findByEmail(email)` tako da vraća `null`, što znači da korisnik nije pronađen
- poziva metodu `addUserTerm` sa zadanim `termId` i korisnikom
- provjerava da odgovor kontrolera ima status `NOT_FOUND (404)`

Test `addUserTermBadRequest` testira scenarij kada dođe do greške zbog nevažećih podataka o terminu.

- "mockira" `oauthUser.getAttribute("email")` tako da vraća email korisnika "testuser@example.com"
- "mockira" `studentService.findByEmail(email)` tako da vraća instancu `Student` s odgovarajućim emailom
- "mockira" `termSignUpService.addUserToTerm(termId, student)` tako da baca iznimku `IllegalArgumentException` s porukom "Invalid term"
- poziva metodu `addUserTerm` sa zadanim `termId` i korisnikom
- provjerava da je metoda `termSignUpService.addUserToTerm(termId, student)` pozvana jednom
- provjerava da odgovor kontrolera ima status `BAD_REQUEST (400)` i odgovarajuću poruku: "Invalid term"

Test `addUserTermConflict` testira scenarij kada postoji sukob pri prijavi korisnika na termin.

- "mockira" `oauthUser.getAttribute("email")` tako da vraća email korisnika "testuser@example.com"
- "mockira" `studentService.findByEmail(email)` tako da vraća instancu `Student` s odgovarajućim emailom
- "mockira" `termSignUpService.addUserToTerm(termId, student)` tako da baca iznimku `IllegalStateException` s porukom "Conflict"
- poziva metodu `addUserTerm` sa zadanim `termId` i korisnikom
- provjerava da je metoda `termSignUpService.addUserToTerm(termId, student)` pozvana jednom
- provjerava da odgovor kontrolera ima status `BAD_REQUEST (400)`

`TermDisenrollmentUnitTest`

Testovi u klasi `TermDisenrollmentUnitTest` testiraju ispravnost metode `removeUserFromTerm` iz klase `TermController`.

Test `removeUserFromTermSuccess` testira scenarij kada se korisnik uspješno odjavljuje s termina.

- "mockira" `oauthUser.getAttribute("email")` tako da vraća email korisnika "testuser@example.com"
- "mockira" `studentService.findByEmail(email)` tako da vraća instancu `Student`
- poziva metodu `removeUserFromTerm`
- provjerava da je metoda `termSignUpService.removeUserFromTerm` pozvana jednom
- provjerava da odgovor kontrolera ima status `OK (200)` i odgovarajuću poruku: "User successfully removed from the term"

Test `removeUserFromTermStudentNotFound` testira scenarij kada korisnik nije pronađen u sustavu.

- "mockira" `oauthUser.getAttribute("email")` tako da vraća email korisnika "testuser@example.com"
- "mockira" `studentService.findByEmail(email)` tako da vraća `null`
- poziva metodu `removeUserFromTerm`
- provjerava da odgovor kontrolera ima status `OK (200)` poruku: "Student not found"

Test `removeUserFromTermBadRequest` testira scenarij kada dođe do greške zbog nevažećeg termina.

- "mockira" `oauthUser.getAttribute("email")` tako da vraća email korisnika "testuser@example.com"
- "mockira" `studentService.findByEmail(email)` tako da vraća instancu `Student`
- "mockira" `termSignUpService.removeUserFromTerm` baca iznimku `IllegalArgumentException` s porukom "Invalid term"
- poziva metodu `removeUserFromTerm`
- provjerava da je metoda `termSignUpService.removeUserFromTerm` pozvana jednom
- provjerava da odgovor kontrolera ima status `BAD_REQUEST (400)`

Test `removeUserFromTermConflict` testira scenarij kada postoji sukob prilikom odjave s termina (npr. korisnik nije prijavljen na termin).

- "mockira" `oauthUser.getAttribute("email")` tako da vraća email korisnika "testuser@example.com"
- "mockira" `studentService.findByEmail(email)` tako da vraća instancu `Student`
- "mockira" `termSignUpService.removeUserFromTerm` baca iznimku `IllegalStateException` s porukom "Conflict"
- poziva metodu `removeUserFromTerm`
- provjerava da je metoda `termSignUpService.removeUserFromTerm` pozvana jednom

- provjerava da odgovor kontrolera ima status `CONFLICT (409)`

Test `removeUserFromTermInternalError` testira scenarij kada dolazi do neočekivane greške unutar sustava.

- "mockira" `oauthUser.getAttribute("email")` tako da vraća email korisnika "testuser@example.com"
- "mockira" `studentService.findByEmail(email)` tako da vraća instancu `Student`
- "mockira" `termSignUpService.removeUserFromTerm` baca iznimku `RuntimeException` s porukom "Unexpected error"
- poziva metodu `removeUserFromTerm`
- provjerava da je metoda `termSignUpService.removeUserFromTerm` pozvana jednom
- provjerava da odgovor kontrolera ima status `INTERNAL_SERVER_ERROR (500)`

FaksFit

Pages 13

Find a page...

Home

1. Opis projektnog zadatka

2. Analiza zahtjeva

3. Specifikacija zahtjeva sustava

4. Arhitektura i dizajn sustava

5. Arhitektura komponenata i razmještaja

6. Ispitivanje programskog rješenja

Integracijski testovi

Unit testovi

AddTermUnitTest

AuthenticationUnitTest

TermEnrollmentUnitTest

TermDisenrollmentUnitTest

7. Tehnologije za implementaciju aplikacije

8. Upute za puštanje u pogon

- 9. Zaključak i budući rad
- A. Dnevnik promjena dokumentacije
- A. Popis literature
- B. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```

7. Tehnologije za implementaciju aplikacije

Edit New page Jump to bottom

dean-rakic edited this page 3 days ago · 1 revision

Korištene tehnologije i alati

- Korišteni radni okvir za web na strani klijenta je **React** sa skriptnim jezikom **JavaScript**.
- Korišteni radni okvir za web na strani poslužitelja je **Spring Boot** s programskom jezikom **Java**.
- Dizajn baze podataka napravljen je u programu **Erdplus**.
- Baza podataka napravljena je u **H2**.
- Autentifikacija korisnika ostvarena je putem **OAuth 2.0**.
- Dizajn aplikacije razrađena je u alatu **Figma**.
- Razvoj aplikacije napravljen je u razvojnim okruženjima **IntelliJ Idea** i **WebStorm**.
- Za sustav kontrole verzija korišten je **Github** repozitorij.
- Za učinkovito upravljanje projektima, praćenje zadatka i vođenje razvoja softvera korištena je **Atlassian Jira**.

FaksFit

Pages 13

Find a page...

Home

1. Opis projektnog zadatka

2. Analiza zahtjeva

3. Specifikacija zahtjeva sustava

4. Arhitektura i dizajn sustava

5. Arhitektura komponenata i razmještaja

- 6. Ispitivanje programskog rješenja
- 7. Tehnologije za implementaciju aplikacije
  - Korištene tehnologije i alati
- 8. Upute za puštanje u pogon
- 9. Zaključak i budući rad
- A. Dnevnik promjena dokumentacije
- A. Popis literature
- B. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```

8. Upute za puštanje u pogon

Edit New page Jump to bottom

dean-rakic edited this page 4 hours ago · 7 revisions

Puštanje u pogon se izvodi kroz dva glavne faze, lokalno i preko Render-a

LOKALNO:
Klonirati repozitorij lokalno na svoju mašinu sa `git clone <link>` i prebaciti se na dev-branch pomoću `git checkout dev` . Nakon toga, treba pokrenuti individualno frontend i backend servere. Pokretanje frontenda se odvija u koracima. Prvo se pomakne u folder po imenu frontend te se pomoću Terminal-a instaliraju svi dependencies za frontend pomoću naredbe `npm install` . Potom naredbom `npm run dev` se pokrene frontend server na adresi "*localhost:5173*". Za backend server se treba pomaknuti u backend folder. Treba izgraditi Maven projekt i onda pokrenuti "main" aplikaciju.

RENDER:
Prvi korak je napraviti Postgres bazu na Renderu. Nakon stvaranja Postgres baze, treba uzeti external link baze i postaviti ga u Application properties na backendu. Frontend i backend se hostaju zasebno, svaki na svom serveru. Na Renderu se naprave dva servera, te se pomoću Docker file-a koji se nalazi u frontend i backend folderu hostaju oba. Docker file stvori kontejner za frontend i backend s zadnjeg commita na main branchu (ne dev branchu). Render na taj način hosta servere. Dok se hosta na serveru, treba promijeniti sve reference na **API_URL** na stvarni link gdje se poslužuje, ne smije ostati "*localhost*".

FaksFit

Pages 13

Find a page...

Home

1. Opis projektnog zadatka

2. Analiza zahtjeva

- 3. Specifikacija zahtjeva sustava
- 4. Arhitektura i dizajn sustava
- 5. Arhitektura komponenata i razmještaja
- 6. Ispitivanje programskog rješenja
- 7. Tehnologije za implementaciju aplikacije
- 8. Upute za puštanje u pogon
  - Puštanje u pogon se izvodi kroz dva glavne faze, lokalno i preko Render-a
- 9. Zaključak i budući rad
- A. Dnevnik promjena dokumentacije
- A. Popis literature
- B. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```

9. Zaključak i budući rad

Edit New page Jump to bottom

dean-rakic edited this page 4 hours ago · 5 revisions

Zaključak i budući rad

Tehnički izazovi:

- CORS error - riješen u konfiguraciji na backendu
- Session error - pamćenje koja vrsta korisnika je ulogirana - riješeno implementacijom JWT tokena

Stečena znanja

- korištenje Spring Boota i React frameworka
- Korištenje Docker i Render platformi za deployment aplikacije
- Uporaba Google OAutha za autentifikaciju i JWT tokena za praćenje sesija
- pravilno korištenje Git-a pomoću grana

Potrebna znanja za brže ostvarenje

- poznavanje Spring Boota i Reacta
- poznavanje nekog deploy providera
- temeljno poznavanje Java i JavaScript programskih jezika
- iskustvo programskog inženjerstva u svim koracima implementacije

Za nastavak rada

- specifičnija i točnija podjela poslova
- praćenje nekog od popularnih modela za razvoj (npr. spiralni model)
- bolji plan implementacije - što se radi kojim redoslijedom i zašto

Nije implementirano:

- sustav šalje obavijest korisniku o dolasku ili ne dolasku na termin

- sustav omogućuje evidentiranje bodova od strane voditelja
- sustav omogućuje određivanje bodova za pojedine aktivnosti
- sustav omogućuje bilježenje prolaska kolegija na kraju semestra

FaksFit

Pages 13

Find a page...

Home

1. Opis projektnog zadatka

2. Analiza zahtjeva

3. Specifikacija zahtjeva sustava

4. Arhitektura i dizajn sustava

5. Arhitektura komponenata i razmještaja

6. Ispitivanje programskog rješenja

7. Tehnologije za implementaciju aplikacije

8. Upute za puštanje u pogon

9. Zaključak i budući rad

Zaključak i budući rad

Tehnički izazovi:

Stečena znanja

Potrebna znanja za brže ostvarenje

Za nastavak rada

Nije implementirano:

A. Dnevnik promjena dokumentacije

A. Popis literature

B. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```

A. Dnevnik promjena dokumentacije

Edit New page

Jump to bottom

dean-rakic edited this page 4 hours ago · 12 revisions

Rev.	Opis promjene/dodatka	Autori	Datum
0.1	Napravljena wiki dokumentacija za plan izrade aplikacije	Burić Lucija	20.10.2024.
0.2	Dodana stranica `5. Arhitektura i dizajn sustava`	Burić Lucija Rajković Stjepan Vucelić Eugen	3.11.2024.
0.3	Dodani opisi obrazaca uporabe u `4. Specifikacija zahtjeva sustava`	Rakić Dean	3.11.2024.
0.4	Dodana 6. i 7. podsekcija Wiki-ja vezano za literaturu i prikaz aktivnosti grupe	Rakić Dean	12.11.2024.
0.5	Upotpunjeni dijelovi dokumentacije	Rakić Dean	14.11.2024.
0.6	Uređena dokumentacija	Burić Lucija	15.11.2024.
0.7	Uređena dokumentacija	Rakić Dean	22.1.2025.
0.8	Uređena dokumentacija	Rakić Dean	23.1.2025.
0.9	Dodani opisi testova u `6. Ispitivanje programskog rješenja`	Burić Lucija	23.1.2025.
1.0	Uređena dokumentacija	Rakić Dean	24.1.2025.

- Home
- 1. Opis projektnog zadatka
- 2. Analiza zahtjeva
- 3. Specifikacija zahtjeva sustava
- 4. Arhitektura i dizajn sustava
- 5. Arhitektura komponenata i razmještaja
- 6. Ispitivanje programskog rješenja
- 7. Tehnologije za implementaciju aplikacije
- 8. Upute za puštanje u pogon
- 9. Zaključak i budući rad
- A. Dnevnik promjena dokumentacije
- A. Popis literature
- B. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```

A. Popis literature

Edit New page Jump to bottom

dean-rakic edited this page 4 hours ago · 5 revisions

Korištena literatura :

- React tutorial : https://youtu.be/SqcY0GlETPk?si=D_8l9uDdVTDuGihx
- OAuth Security with Springboot and React: https://www.youtube.com/watch?v=ouE3NuTzf20
- Springboot tutorial : https://youtu.be/QuvS_VLbGko?si=U2fYheUiMxgJ28Fn
- Spring, React i Deployment materijali s tehničkog predavanja "Razvoj web aplikacije korištenjem tehnologija Spring Boot i React":
  i. https://gitlab.com/hrvojesimic/progi-project-teams-backend
  ii. https://gitlab.com/jtomic/opp-project-teams-frontend
  iii. https://github.com/progi-devops/progi-monorepo
- Korisna stranica za Spring: https://www.baeldung.com/
- Ostalo:
  Spring.io
  Geeksforgeeks.org
  stackoverflow.com
  medium.com
  https://mvnrepository.com/

FaksFit

Pages 13

Find a page...

Home

1. Opis projektnog zadatka

- 2. Analiza zahtjeva
- 3. Specifikacija zahtjeva sustava
- 4. Arhitektura i dizajn sustava
- 5. Arhitektura komponenata i razmještaja
- 6. Ispitivanje programskog rješenja
- 7. Tehnologije za implementaciju aplikacije
- 8. Upute za puštanje u pogon
- 9. Zaključak i budući rad
- A. Dnevnik promjena dokumentacije
- A. Popis literature
- B. Prikaz aktivnosti grupe

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```

B. Prikaz aktivnosti grupe

AdrianSarotar edited this page 31 minutes ago · 10 revisions

Dnevnik sastajanja

1. sastanak:

- Datum: 14.10.2024., uživo
- Prisustvovali: Stjepan Rajković, Lucija Burić, Dominik Gračner, Vitomir Brebrić, Eugen Vucelić, Adrian Šarotar, Dean Rakić
- Teme sastanka:
- Upoznavanje članova tima
- Dogovor teme projekta
- Raspodjela posla između članova na frontend development, backend development, rad na bazi podataka i dopunjavanje dokumentacije

2. sastanak:

- Datum : 28.10.2024, online
- Prisustvovali: Stjepan Rajković, Lucija Burić, Dominik Gračner, Vitomir Brebrić, Eugen Vucelić, Adrian Šarotar
- Teme sastanka:
- Demonstracija baze podataka
- Prokomentirane i definirane funkcionalnosti aplikacije, uključujući prijavu u sustav, odgovornosti studenta, voditelja sekcije i nastavnika

3. sastanak:

- Datum: 4.11.2024., uživo i online preko Teams-a
- Prisustvovali: Stjepan Rajković, Lucija Burić, Dominik Gračner, Vitomir Brebrić, Eugen Vucelić, Adrian Šarotar, Dean Rakić
- Teme sastanka:
- Fokus na razvoj stranice i korisničkog sučelja i povezivanja s backend-om
- Daljnja raspodjela poslova

4. sastanak:

- Datum: 10.12.2024., online preko Discord-a
- Prisustvovali: Stjepan Rajković, Lucija Burić, Dominik Gračner, Vitomir Brebrić, Eugen Vucelić, Adrian Šarotar, Dean Rakić
- Teme sastanka:
- Komentiranje pronađenih problema
- Provjera stanja frontend-a i backenda

5. sastanak:

- Datum: 13.1.2025., online preko Discord-a
- Prisustvovali: Stjepan Rajković, Lucija Burić, Dominik Gračner, Vitomir Brebrić, Eugen Vucelić, Adrian Šarotar, Dean Rakić
- Teme sastanka:
- Provjera stanja poslova nakon praznika

6. sastanak:

- Datum: 20.1.2025., online preko Discord-a
- Prisustvovali: Stjepan Rajković, Lucija Burić, Dominik Gračner, Vitomir Brebrić, Eugen Vucelić, Adrian Šarotar
- Teme sastanka:
- Dogovor za zadnji tjedan
- Dovršavanje implementacije

Plan rada

Pod "Projects" se nalazi Gantt dijagram u obliku Roadmapa vezan za dosadašnji i budući plan rada

Tablica aktivnosti (u satima)

Dominik Gračner

Frontend - 5
Backend - 25
Povezivanje Frontenda s Backendom - 3
Implementacija baze - 1
Povezivanje na vanjski API za Google OAuth - 3
Deployment - 14

Stjepan Rajković

Frontend - 30
Backend - 5
Povezivanje Frontenda s Backendom - 5
Povezivanje na vanjski API za Google Maps - 7

Lucija Burić

Frontend - 2
Backend - 25
Povezivanje Frontenda s Backendom - 10
Izrada opisa baze - 10
Implementacija baze - 5
Testovi - 10
Dokumentacija - 5

Eugen Vucelić

Frontend - 10
Backend - 50
Povezivanje Frontenda s Backendom - 10
Izrada opisa baze - 10
Implementacija baze - 10
Povezivanje na vanjski API za Google OAuth - 1
Deployment - 30

Vitomir Brebrić

frontend 35
backend 3
struktura i plan stranica 8
dizajn 8

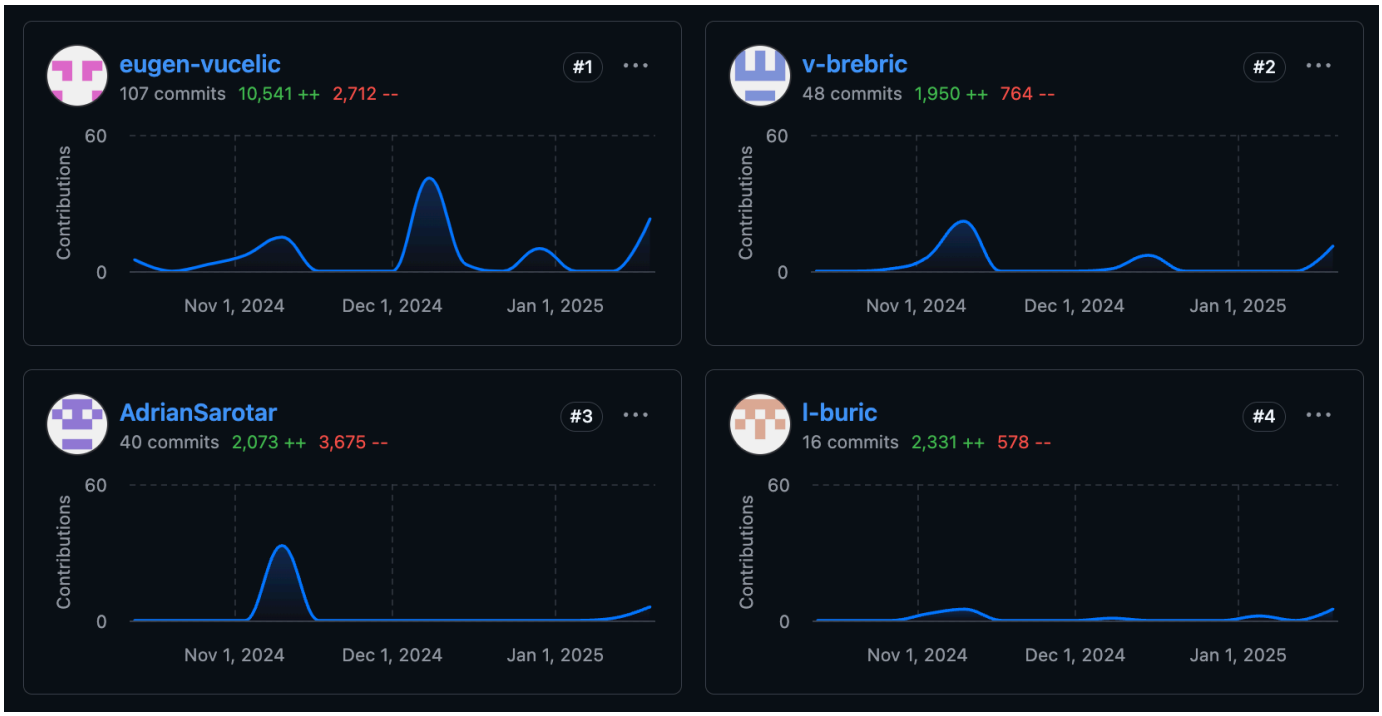
Adrian Šarotar

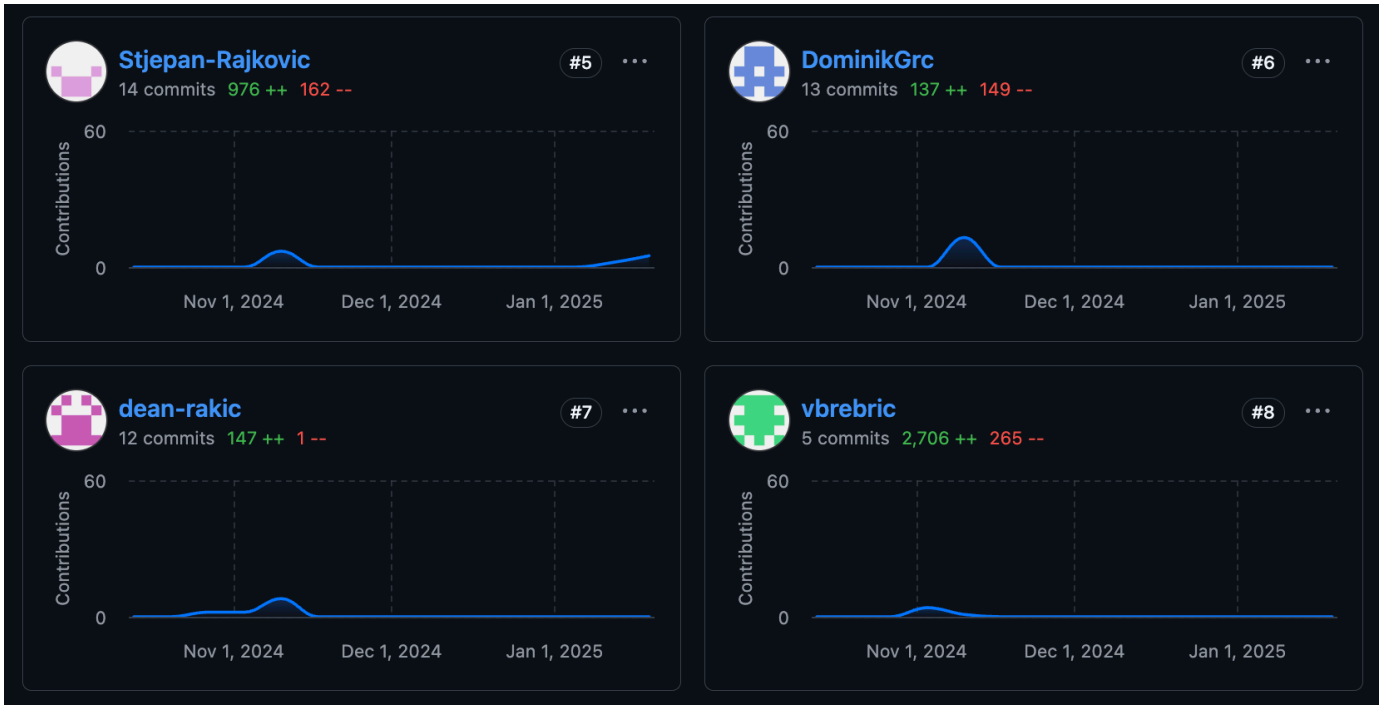
frontend 25
backend 2
povezivanje 5
deployment 17

Dean Rakić

Frontend - 3
Dokumentacija - 25
Deployment - 3

Dijagrami pregleda promjena





FaksFit

Pages 13

Find a page...

- Home
- 1. Opis projektnog zadatka
- 2. Analiza zahtjeva
- 3. Specifikacija zahtjeva sustava
- 4. Arhitektura i dizajn sustava
- 5. Arhitektura komponenata i razmještaja
- 6. Ispitivanje programskog rješenja
- 7. Tehnologije za implementaciju aplikacije
- 8. Upute za puštanje u pogon
- 9. Zaključak i budući rad
- A. Dnevnik promjena dokumentacije
- A. Popis literature
- B. Prikaz aktivnosti grupe
  - Dnevnik sastajanja
  - Plan rada
  - Tablica aktivnosti (u satima)
    - Dominik Gračner
    - Stjepan Rajković
    - Lucija Burić
    - Eugen Vucelić
    - Vitomir Brebrić
    - Adrian Šarotar
    - Dean Rakić
  - Dijagrami pregleda promjena

+ Add a custom sidebar

Clone this wiki locally

```
https://github.com/eugen-vucelic/FaksFit.wiki.git
```