Univerzitet u Beogradu

Matematički fakultet

Simić Aleksandra

Projekat i implementacija informacionog sistema za zadavanje, pregledanje i rešavanje programerskih zadataka

MASTER RAD

Beograd, Oktobar 2015

Univerzitet u Beogradu - Matematički fakultet
MASTER RAD

Autor:	Simić Aleksandra
Naslov:	Projekat i implementacija informacionog sistema za zadavanje, pregledanje i rešavanje programerskih zadataka
Mentor:	dr Saša Malkov, Matematički fakultet
Članovi komisije:	dr Vladimir Filipović, Matematički fakultet dr Mladen Nikolić, Matematički fakultet
Datum:	x.10.2015

Rezime

Informacioni sistemi predstavljaju jednu od najvažnijih oblasti računarstva. Sa druge strane, zadavanje, pregledanje i rešavanje programerskih zadataka imaju veliki značaj u procesu školovanja informatičara. Značaj teme ovog rada je u povezivanju ovih oblasti.
Cilj rada je da se projektuje i implementira veb zasnovani informacioni sistem za zadavanje, rešavanje i pregledanje programerskih zadataka. Sistem sadrži dve namenski različite celine:

- Prva celina pruža interfejs preko kog će nastavnici moći da zadaju domaće zadatke u okviru postojećih kurseva, a studenti će moći da postavljaju rešenja ovih zadataka i da odmah dobiju povratne informacije o tačnosti rešenja i potencijalnim greškama. Nastavnici će biti obavešteni o aktivnostima studenata i moći će da ispravljaju ocene koje je sistem dodelio.
- Druga celina omogućava studentima da u svoje slobodno vreme usavršavaju programerska znanja na pažljivo odabranim i razvrstanim zadacima za vežbu. Predlaganjem i dodavanjem zadataka će se zajedno baviti nastavno osoblje i studenti.

Sam postupak ocenjivanja rešenja nije deo ovog rada, već predstavlja poseban projekat.

Sadržaj

1 UVOD

Rad opisuje projekat i implementaciju informacionog sistema za zadavanje, pregledanje i rešavanje programerskih zadataka - Dr. Webgrade. Sistem je osmišljen sa ciljem da pruži pomoć pri usavršavanju programerskog znanja studenata kroz mnoštvo različitih, kako domaćih zadataka, tako i zadataka namenjenih za vežbu. Dodatno, plan je da sistem bude korišćen u svakodnevnoj nastavi na svim nivoima.

2 ANALIZA

2.1 Analiza procesa nastave programerskih predmeta

Kursevi

Svake godine studenti se upisuju na predviđene kurseve. Kurs se smatra uspešno položenim ako je student ispunio sve svoje obaveze na tom kursu, propisane od strane zaduženog lica. Zaduženo lice je koordinator kursa. Ukoliko kurs ima jednog predmetnog nastavnika on je ujedno i koordinator. Ako usled velikog broja studenata na kursu postoji više predmetnih nastavnika, jedan od njih je najčešće koordinator. Propisane obaveze su: predispitne obaveze i ispit. Predispitne obaveze su sačinjene od neke kombinacije teorijskih i praktičnih testova, kolokvijuma, seminarskih radova i domaćih zadataka. Ideja predispitnih obaveza je da studenti redovnije rade predmet i kao posledica toga, da ga bolje ga savladaju. Uprkos tome, veliki broj kurseva od predispitnih obaveza ima samo jedan kolokvijum, čime se ne postiže željeni efekat predispitnih obaveza.

Nastava programerskih predmeta se realizije kroz predavanja i vežbe, koji ne moraju biti usko povezani. Na predavanjima se rade teorijski koncepti bitni za taj kurs, dok se na vežbama rešavaju programerski problemi. Nastavnici predaju korišćenjem prezentacija, koje su napravljene na osnovu stručne literature ili direktno pišu sadržaj na tabli i pružaju dodatna objašnjenja. Predavanja retko obuhvataju veći broj programskih kodova, a čak i onda kad ih obuhvataju, studenti ih ne kucaju, već oni samo bivaju prikazani kao dodatni primeri.

U slučaju vežbi, koncept je značajno drugačiji, međutim pokazuje se da i na tom polju postoji problem. Kako se praktično znanje (programiranje) retko opsežnije obrađuje na profesorskim predavanjima, vežbe postaju opterećene količinom gradiva koja treba se pređe. Posledica je da količina usvojenog gradiva i to šta je student sposoban samostalno da isprogramira, najčešće nisu na željenom nivou.

Problemi

Glavne probleme predstavljaju mala prolaznost kurseva, koja vodi ka tome da

previše studenata sluša taj kurs, premali broj programerskih kodova koje prosečan student uspešno iskuca za vreme kursa i slično. Da bi ove tvrdnje bile potkrepljene primerima, ovo su konkretni rezultati u okviru nekih predmeta:

1. Programiranje 2, I smer, školska 2012/2013 godina:
    - kurs je pohađalo 260 studenata
    - kurs je položio 61 student (23.5%)
2. Prevođenje programskih jezika, I smer, školska 2012/2013 godina
    - kurs je pohađalo 94 studenata
    - kurs je položilo 28 studenata (42%)
3. Operativni sistemi, I smer, školska 2012/2013
    - kurs je pohađalo 94 studenata
    - kurs je položilo 20 studenata (21.3%)

Što se navedenih predmeta tiče, rezultati su slični i ako se pogledaju druge školske godine. Bitno je navesti još neke predmete čija prolaznost nije loša, a čiji koncept može značajno da se promeni tako da se podigne nivo znanja kod studenata. Na algoritamskim predmetima nema kodiranja, nego se sve objašnjava u pseudo-kodu. Kao posledica toga, studenti nemaju predstavu o implementacionim aspektima algoritama koje uče. Ne znaju na koje sve probleme mogu da naiđu, pa samim tim ni kako da ih reše.

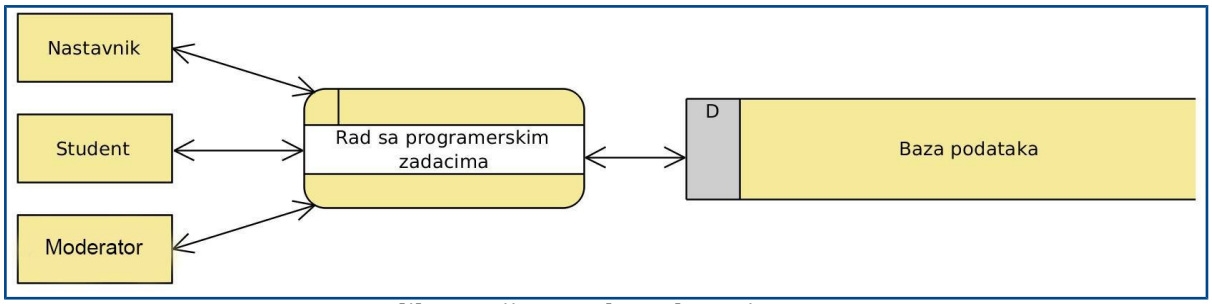
Sa druge strane, asistenti često stavljaju akcenat na određene delove gradiva koje bi sudenti trebalo da usavrše samostalnim radom. Kako je to samo neformalna sugestija, čije se ispunjenje ne proverava, studenti obično zanemare predloženo na svoju štetu.

Ideja je da se korišćenjem sistema *Dr. Webgrade* uvede formalna obaveza čije se ispunjenje lako prati. Dodatno, studenti će se navikavati da moraju da zadovolje forme ulaza i izlaza, sa kojima će se sretati na kolokvijumima i ispitima, kao i da ispoštuju zadata memorijska i vremenska ograničenja. Navedene prednosti koje sistem pruža bi trebalo da u velikoj meri povećaju znanje, pa samim tim i prolaznost studenata na programerskim kursevima.

Nastavnici će takođe imati dosta koristi od samog sistema. Prvenstveno, oslobodjeni su pregledanja radova - sistem to radi za njih. Dodatno, pružen im je jako dobar interfejs za pregledanje studentskih kodova i eventualnu promenu broja poena, ukoliko smatraju da student zaslužuje manje ili više poena. Pored toga, na raspolaganju su im statistike za svaki predmet, odnosno statistike za svaki domaći zadatak, iz kojih mogu da vide koliko studenata je rešilo koji zadatak i iz kog broja pokušaja. Na osnovu toga mogu da zaključuju koje gradivo su studenti bolje, odnosno lošije usvojili i da na vežbama stave akcenat na to lošije usvojeno gradivo.

2.2 Dijagrami toka podataka

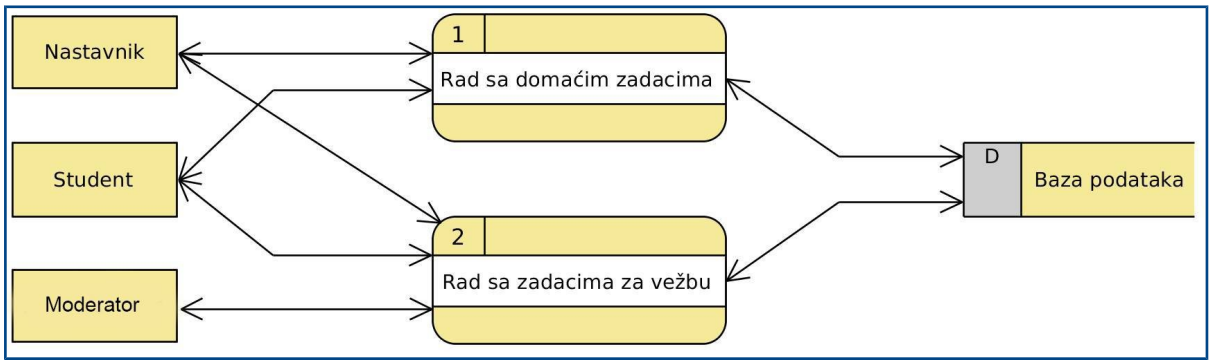
Na dijagramu konteksta svi procesi sistema su predstavljeni jednim procesom – *Rad sa programerskim zadacima*. Prikazano je sa kojim entitetima (*Nastavnik, Student, Moderator*) i skladištima podataka (*Baza podataka*) sistem interaguje.



Slika 1: Dijagram konteksta sistema

Na dijagramu prvog nivoa glavni proces delimo na dva podprocesa:

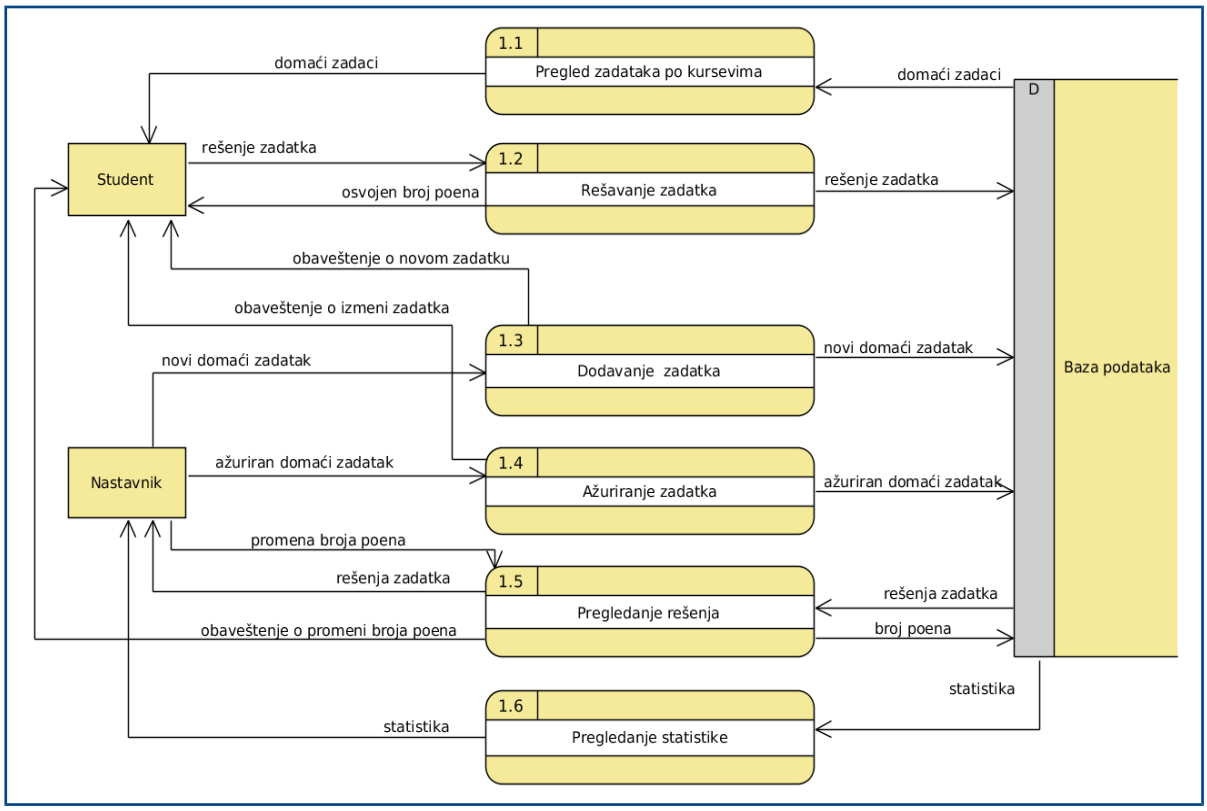
1. Rad sa domaćim zadacima
2. Rad sa zadacima za vežbu



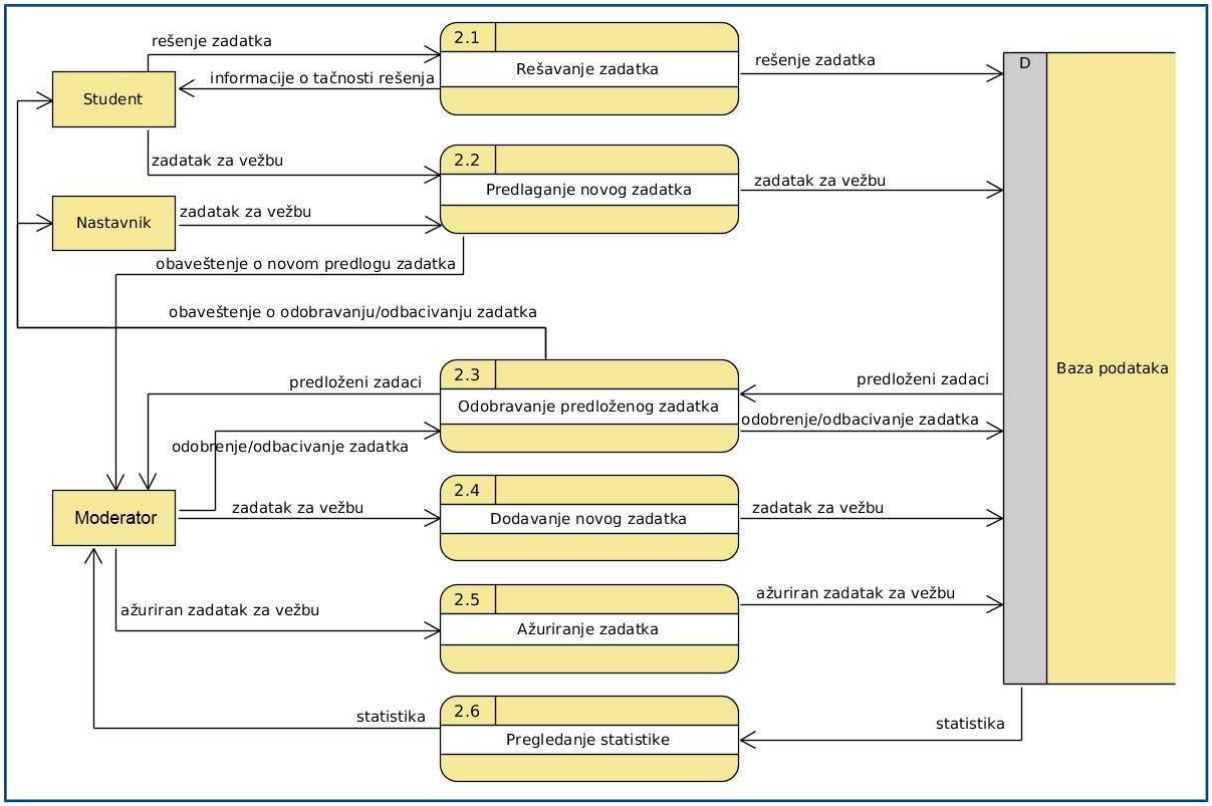
Slika 2: Dijagram toka podataka, nivo 1

Što se tiče dijagrama drugog nivoa, tu su detaljnije opisani procesi „Rad sa domaćim zadacima“ i „Rad sa zadacima za vežbu“. Pored toga, precizno su prikazani tokovi podataka između entiteta, procesa i skladišta podataka.

Proces „Rad sa domaćim zadacima“ se sastoji od 6 potprocesa: „Pregled zadataka po kursevima“, „Rešavanje zadatka“, „Dodavanje zadatka“, „Ažuriranje zadatka“, „Pregledanje rešenja“ i „Pregledanje statistike“ (slika 3).



Slika 3: Dijagram toka podataka, nivo 2, modul „Rad sa domaćim zadacima“



Slika 4: Dijagram toka podataka, nivo 2, modul „Rad sa zadacima za vežbu“

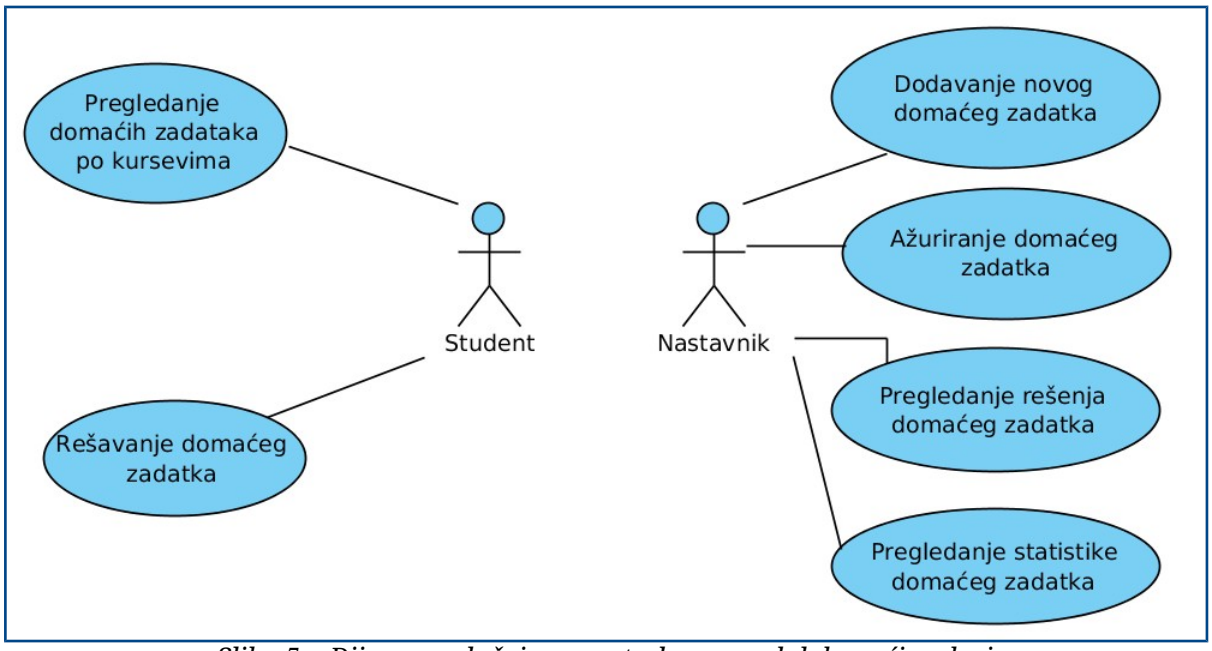
Proces „Rad sa zadacima za vežbu“ se takođe sastoji iz 6 potprocesa i oni su: „Rešavanje zadatka“, „Predlaganje novog zadatka“, „Odobravanje predloženog zadatka“, „Dodavanje novog zadatka“, „Ažuriranje zadatka“ i „Pregledanje statistike“ (slika 4).

2.3 Slučajevi upotrebe

2.3.1. Modul domaći

U okviru modula za domaće zadatke prepoznati su sledeći slučajevi upotrebe:

1. Pregledanje domaćih zadataka po kursevima
2. Rešavanje domaćeg zadatka
3. Dodavanje novog domaćeg zadatka
4. Ažuriranje domaćeg zadatka
5. Pregledanje rešenja domaćeg zadatka
6. Pregledanje statistike domaćeg zadatka



Slika 5a: Dijagram slučajeva upotrebe za modul domaći zadaci

2.3.1.1 Pregledanje domaćih zadataka po kursevima

1. **Kratak opis:** Student bira kurs za koji želi da vidi domaće zadatke. Prikazuje mu se lista domaćih zadataka za izabrani kurs.
2. **Učesnik:** Student
3. **Preduslovi:** Student je upisan na neki kurs u tekućoj školskoj godini.
4. **Postuslovi:** Otvorena je stranica sa domaćim zadacima za izabrani kurs.
5. **Osnovni tok:**

   Student bira kurs za koji želi da vidi domaće zadatke.

   Prikazuju se dve liste zadataka:

1. Zadaci koje student može da rešava
2. Zadaci kojima je prošao rok za izradu

Za svaki zadatak se prikazuje naziv, period trajanja domaćeg zadatka, broj poena koje zadatak nosi i maksimalan broj poena koje je student ostvario na tom zadatku (ako ga do tog trenutka nije rešavao, prikazaće se 0)

Student može da izabere neki od zadataka iz prve liste i time prelazi na slučaj upotrebe „Rešavanje domaćeg zadatka“.

6. **Alternativni tok:**

   Nema domaćih zadataka: Ukoliko na izabranom kursu nema domaćih zadataka, studentu se prikazuje odgovarajuća poruka.

2.3.1.2 Rešavanje domaćeg zadatka

1. **Kratak opis:** Student postavlja rešenje za izabrani domaći zadatak. Sistem ocenjuje okačeno rešenje na osnovu test primera i obaveštava studenta o broju osvojenih poena.
2. **Učesnik:** Student
3. **Preduslovi:** Student je upisan na bar jedan kurs na kom postoji bar jedan domaći zadatak.
4. **Postuslovi:** Student je obavešten o tačnosti rešenja koje je okačio.
5. **Osnovni tok:**

   Student bira neki od ponuđenih domaćih zadataka.

   Za izabrani zadatak se prikazuje: naziv, tekst, vremensko ograničenje, memorijsko ograničenje, lista ponudjenih programskih jezika u kojima je moguće rešiti zadatak, rok za izradu, dosadašnji broj pokušaja tog studenta da reši zadatak kao i najveći broj poena osvojen pri tim pokušajima.

   Student označava programski jezik koji je koristio za rešavanje zadatka.

   Student šalje fajl sa rešenjem.

   Student bira opciju „Pošalji“.

   Sistem vrši ocenjivanje zadatka na zadatim test primerima.

   Kako koji test primer prođe ili ne prođe, sistem o tome obaveštava studenta.

   Po završetku testiranja, sistem ispisuje konačan broj poena, kao i poruku za svaki od test primera.

   Poruke mogu biti:

   1. Tačno rešenje
   2. Netačno rešenje
   3. Prekoračenje vremenskog ograničenja
   4. Prekoračenje memorijskog ograničenja
   5. Greška pri kompajliranju (sa porukom o grešci koji je kompajler vratio)

6. Greška pri izvršavanju (sa porukom o grešci koju je ocenjivač vratio)

6. **Alternativni tokovi:**

   Izabrani programski jezik i vrsta datoteke u kojoj je smešteno rešenje se ne poklapaju: Rešenje neće biti poslato na obradu. Sistem prikazuje odgovarajuću poruku.

   Napuštanje stranice dok je ocenjivanje još uvek u toku: Sistem boduje okačeno rešenje sa 0 poena i broj pokušaja rešavanja zadatka se uvećava za 1.

2.3.1.3 Dodavanje novog domaćeg zadatka

1. **Kratak opis:** Nastavnik zadaje novi domaći zadatak u okviru nekog od kurseva koje predaje.
2. **Učesnik:** Nastavnik
3. **Preduslovi:** Nastavnik predaje bar jedan kurs u tekućoj školskoj godini
4. **Postuslovi:** Postavljen je novi domaći zadatak i svi studenti koji su upisani na odgovarajući kurs (kod nastavnika koji je zadao zadatak za taj kurs) su obavešteni o tome.
5. **Osnovni tok:**

   Nastavnik bira kurs za koji želi da zada novi domaći zadatak.

   Nastavnik bira opciju „Dodaj novi domaći“.

   Sistem prikazuje formular za dodavanje novog domaćeg zadatka na kom se nalaze polja za unos:

   1. Naziva zadatka
   2. Teksta zadatka
   3. Test primera
   4. Programskih jezika u kojima student može da reši taj zadatak
   5. Vremenskog ograničenja
   6. Memorijskog ograničenja
   7. Perioda trajanja domaćeg zadatka
   8. Broja poena koje on nosi

   Nastavnik unosi podatke u formular.

   Nastavnik bira opciju „Dodaj“.

   Sistem čuva domaći zadatak.

   Sistem obaveštava sve studente koji slušaju odgovarajući kurs kod ovog nastavnika o tome da je postavljen novi domaći zadatak.

6. **Alternativni tokovi:**

   Neispravan unos: Ukoliko nastavnik propusti da unese neki od podataka ili unese neispravan podatak, sistem ispisuje odgovarajuću poruku.

Prekid unosa: Ukoliko nastavnik napusti trenutnu stranicu, unos novog zadatka se obustavlja i uneti podaci neće biti zapamćeni.

2.3.1.4 Ažuriranje domaćeg zadatka

1. **Kratak opis:** Nastavnik ažurira neki od domaćih zadataka zadataka koje je prethodno zadao.
2. **Učesnik:** Nastavnik
3. **Preduslovi:** Nastavnik je prethodno zadao bar jedan domaći zadatak.
4. **Postuslovi:** Zadatak je izmenjen i studenti su obavešteni o izmeni.
5. **Osnovni tok:**

   Nastavnik bira kurs.

   Sistem ispisuje listu svih domaćih zadataka koje je nastavnik zadao na izabranom kursu.

   Nastavnik bira da izmeni domaći zadatak klikom na opciju „Izmeni domaći“.

   Sistem prikazuje formular popunjen prethodno unetim podacima.

   Nastavnik unosi izmene.

   Nastavnik bira opciju „Potvrdi izmene“.

   Sistem snima izmene.

   Sistem obaveštava sve studente koji slušaju odgovarajući kurs kod ovog nastavnika o izmenama domaćeg zadatka.
6. **Alternativni tokovi:**

   Neispravan unos: Ukoliko nastavnik izostavi da unese neki od podataka ili unese neispravan podatak, sistem ispisuje odgovarajuću poruku.

   Prekid unosa: Ukoliko nastavnik napusti trenutnu stranicu, uneti podaci neće biti zapamćeni.

2.3.1.5 Pregledanje rešenja domaćeg zadatka

1. **Kratak opis:** Nastavnik pregleda kodove koje su studenti okačili kao rešenja domaćeg zadatka i ukoliko misli da je to potrebno, vrši promenu broja poena koje je sistem dodelio nekom studentu na tom zadatku.
2. **Učesnik:** Nastavnik
3. **Preduslovi:** Nastavnik je zadao bar jedan domaći zadatak.
4. **Postuslovi:** Ukoliko je nastavnik promenio broj poena za nekog studenta, ta promena je sačuvana u bazi i student je obavešten o tome.
5. **Osnovni tok:**

   Nastavnik bira kurs.

   Sistem ispisuje listu svih domaćih zadataka koje je nastavnik zadao na izabranom kursu.

   Nastavnik bira da pregleda studentska rešenja, izborom opcije „Pregledaj rešenja“ .

Sistem prikazuje listu studenata koji su pokušali da reše zadatak. Uz svakog studenta stoji maksimalan broj poena koji su osvojili pri tim pokušajima, kao i dve opcije: *„Pregledaj najbolje rešenje“* i *„Pregledaj sva rešenja“*.

1. Nastavnik bira opciju „Pregledaj najbolje rešenje“:

   Sistem prikazuje rešenje kojem je dodeljeno najviše poena, a koje je okačio taj student.

   Nastavnik pregleda kod.

   1.1 Nastavnik vrši promenu broja poena za tog studenta.

   Nastavnik potvrđuje promenu klikom na dugme „Potvrdi“.

   Sistem snima promenu.

   Sistem obaveštava studenta o promeni.

   1.2 Nastavnik prelazi na sledećeg studenta klikom na dugme „Sledeći“.

   1.3 Nastavnik prelazi na prethodnog studenta klikom na dugme „Prethodni“.

   Nastavnik ponavlja prethodna tri koraka dokle god smatra da je to potrebno.

2. Nastavnik bira opciju „Pregledaj sva rešenja“

   Sistem prikazuje listu svih rešenja koje je okačio jedan student. Za svako rešenje prikazuje broj poena koji mu je dodeljen i datum kada je to rešenje postavljeno. Datu listu je moguće sortirati kako po broju poena, tako i po datumu.

   Nastavnik bira da pregleda neko od rešenja izborom opcije „Pregledaj rešenje“.

   Nastavnik pregleda kod.

   1.1 Nastavnik vrši promenu broja poena za tog studenta.

   Nastavnik potvrđuje promenu klikom na dugme „Potvrdi“.

   Sistem snima promenu.

   Sistem obaveštava studenta o promeni.

   1.2 Nastavnik prelazi na sledeće rešenje izborom opcije „Sledeće rešenje“.

   1.3 Nastvanik prelazi na prethodno rešenje izborom opcije „Prethodno rešenje“.

   Nastavnik ponavlja prethodna tri koraka dokle god smatra da je to potrebno.

6. **Alternativni tokovi:**

   Neispravan unos: Ukoliko nastavnik unese veći broj poena nego što taj zadatak nosi, sistem ispisuje odgovarajuću poruku.

   Prekid unosa: Ukoliko nastavnik napusti stranicu pre nego što potvrdi unos broja poena, uneti podaci neće biti zapamćeni.

2.3.1.6 Pregledanje statistike domaćih zadataka

1. **Kratak opis:** Nastavnik pregleda statistike domaćeg zadatka.
2. **Učesnik:** Nastavnik
3. **Preduslovi:** Nastavnik je zadao bar jedan domaći zadatak.
4. **Postuslovi:** Nastavnik ima uvid u sve statistike vezane za domaći zadatak.
5. **Osnovni tok:**

   Nastavnik bira kurs.

   Sistem prikazuje:

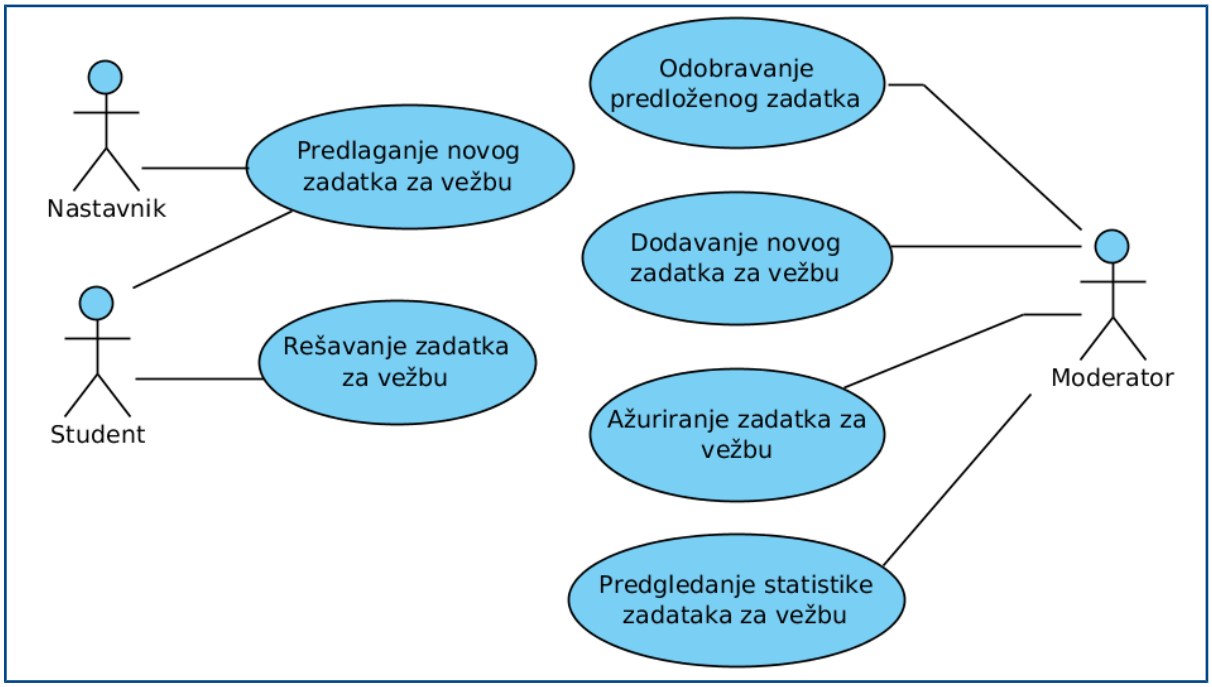
   1. Listu domaćih zadataka
   2. Listu studenata koji slušaju taj kurs
   3. Statističke podatke za svaki od domaćih zadataka.

   Nastavnik pregleda statistike.
6. **Alternativni tokovi:** /

2.3.2. Modul vežbanje

U okviru modula vežbanje prepoznati su sledeći slučajevi upotrebe:

1. Rešavanje zadatka za vežbu
2. Predlaganje novog zadatka za vežbu
3. Odobravanje predloženog zadatka
4. Dodavanje novog zadatka za vežbu
5. Ažuriranje zadatka za vežbu
6. Pregledanje statistike zadatka za vežbu



Slika 5b: Dijagram slučajeva upotrebe za modul vežbanje

2.3.2.1 Rešavanje zadatka za vežbu

1. **Kratak opis:** Student postavlja rešenje za izabrani zadatak za vežbu. Sistem ocenjuje okačeno rešenje na osnovu test primera i obaveštava studenta o tačnosti rešenja.
2. **Učesnik:** Student
3. **Preduslovi:** Na sistemu postoji bar jedan zadatak za vežbu.
4. **Postuslovi:** Rešenje je sačuvano i student je obavešten o tačnosti tog rešenja.
5. **Osnovni tok:**

   Student bira neki od ponuđenih zadataka za vežbu.

   Za izabrani zadatak se prikazuje: naziv, tekst, vremensko ograničenje, memorijsko ograničenje, lista ponudjenih programskih jezika u kojima je moguće rešiti zadatak, dosadašnji broj pokušaja tog studenta da reši zadatak kao i najveći broj poena osvojen pri tim pokušajima.

   Student bira programski jezik koji je koristio za rešavanje zadatka.

   Student bira fajl sa rešenjem.

   Student bira opciju „Pošalji“.

   Sistem vrši ocenjivanje zadatka na zadatim test primerima.

   Kako koji test primer prođe ili ne prođe, sistem o tome obaveštava studenta.

   Po završetku testiranja, sistem ispisuje konačan broj poena, kao i poruku za svaki od test primera.

   Poruke mogu biti:

   1. Tačno rešenje
   2. Netačno rešenje
   3. Prekorašenje vremenskog ograničenja
   4. Prekoračenje memorijskog ograničenja
   5. Greška pri kompajliranju (sa porukom o grešci koji je kompajler vratio)
   6. Greška pri izvršavanju (sa porukom o grešci koju je ocenjivač vratio)
6. **Alternativni tokovi:**

   Izabrani programski jezik i ekstenzija rešenja se ne poklapaju: Rešenje neće biti poslato na obradu. Sistem prikazuje odgovarajuću poruku.

   Napuštanje stranice dok je ocenjivanje još uvek u toku: Sistem boduje okačeno rešenje sa 0 poena i broj pokušaja rešavanja zadatka se uvećava za 1.

2.3.2.2 Predlaganje novog zadatka za vežbu

1. **Kratak opis:** Nastavnik ili student (u daljem tekstu „Akter“), predlažu da neki zadatak bude svrstan među zadatke za vežbu. Ovakvi zadaci, da bi

bili prikazani, moraju biti odobreni od strane moderatora sistema.

2. **Učesnici:** Nastavnik, Student
3. **Preduslovi:** /
4. **Postuslovi:** Predloženi zadatak je snimljen i moderator je obavešten o tome.
5. **Osnovni tok:**

   Akter bira da predloži novi zadatak izborom opcije „Dodaj novi zadatak“.

   Sistem prikazuje formular za dodavanje novog zadatka na kom se nalaze polja za unos:

   1. Naziva zadatka
   2. Teksta zadatka
   3. Test primera
   4. Vremenskog ograničenja
   5. Memorijskog ograničenja
   6. Kategorije

   Akter unosi podatke u formular.

   Akter bira opciju „Dodaj“.

   Sistem pamti predlog zadatka za vežbu.

   Sistem obaveštava moderatora o ovom predlogu.
6. **Alternativni tokovi:**

   Neispravan unos: Ukoliko akter izostavi da unese neki od obaveznih podataka ili unese neispravan podatak, sistem ispisuje odgovarajuću poruku.

   Prekid unosa: Ukoliko akter napusti trenutnu stranicu, uneti podaci neće biti zapamćeni.

2.3.2.3 Odobravanje predloženog zadatka

1. **Kratak opis:** Moderator vrši proveru nekog od predloženih zadataka za vežbu i odlučuje da odobri ili odbije taj zadatak. Osoba koja je predložila taj zadatak biva obaveštena o odluci moderatora
2. **Učesnik:** Moderator
3. **Preduslovi:** Postoji bar jedan predložen zadatak za vežbu.
4. **Postuslovi:** Zadatak je usvojen ili odbijen i osoba koja ga je predložila je obaveštena o tome.
5. **Osnovni tok:**

   Moderator pregleda sve predloge izborom opcije „Lista predloženih zadataka“.

   Sistem prikazuje listu zadataka. Uz svaki od zadataka stoji i ko ga je i kada predložio i opcija „Vidi detalje“.

   Moderator bira opciju „Vidi detalje“.

Sistem prikazuje popunjen formular predloga zadatka za vežbu.

Moderator proverava sve unete podatke.

Moderator po potrebi vrši izmene nekih podataka.

1.1 Moderator odobrava predlog zadatka izborom opcije „Odobri“.

1.2 Moderator odbija predlog zadatka izborom opcije „Odbij“.

Sistem obaveštava korisnika koji je predložio zadatak o odluci moderatora.

6. **Alternativni tokovi:**

   Neispravan unos: Ukoliko moderator izostavi da unese neki od podataka ili unese neispravan podatak, sistem ispisuje odgovarajuću poruku.

   Prekid unosa: Ukoliko moderator napusti stranicu, promene neće biti zapamćene.

2.3.2.4 Dodavanje novog zadatka za vežbu

1. **Kratak opis:** Moderator zadaje novi zadatak za vežbu.
2. **Učesnik:** Moderator
3. **Preduslovi:** /
4. **Postuslovi:** Postavljen je novi zadatak za vežbu.
5. **Osnovni tok:**

   Moderator bira opciju „Dodaj novi zadatak“.

   Sistem prikazuje formular za dodavanje novog zadatka za vežbu na kom se nalaze polja za unos:

   1. Naziva zadatka
   2. Teksta zadatka
   3. Test primera
   4. Vremenskog ograničenja
   5. Memorijskog ograničenja
   6. Kategorija

   Moderator unosi podatke u formular.

   Moderator bira opciju „Dodaj“.

   Sistem potvrđuje da je zadatak uspešno dodat.

6. **Alternativni tokovi:**

   Neispravan unos: Ukoliko moderator izostavi da unese neki od podataka ili unese neispravan podatak, sistem ispisuje odgovarajuću poruku.

   Prekid unosa: Ukoliko moderator napusti trenutnu stranicu, unos novog zadatka se obustavlja i uneti podaci neće biti zapamćeni.

2.3.2.5 Ažuriranje zadatka za vežbu

1. **Kratak opis:** Moderator ažurira neki od zadataka koje je prethodno

dodao.

2. **Učesnik:** Moderator
3. **Preduslovi:** Moderator je prethodno dodao bar jedan zadatak za vežbu.
4. **Postuslovi:** Zadatak za vežbu je uspešno izmenjen.
5. **Osnovni tok:**

   Moderator otvara listu svih zadataka izborom opcije „Zadaci za vežbu“.

   Sistem prikazuje zadatke i uz svaki zadatak nalazi se opcija „Izmeni“.

   Moderator bira da izmeni zadatak klikom na ovu opciju.

   Sistem prikazuje formular popunjen prethodno unetim podacima.

   Moderator unosi izmene.

   Moderator bira opciju „Potvrdi izmene“.

   Sistem potvrđuje da je zadatak uspešno promenjen.
6. **Alternativni tokovi:**

   Neispravan unos: Ukoliko moderator izostavi neki od podataka ili unese neispravan podatak, sistem ispisuje odgovarajuću poruku.

   Prekid unosa: Ukoliko moderator napusti trenutnu stranicu, uneti podaci neće biti zapamćeni.

2.3.2.6 Pregledanje statistike zadataka za vežbu

1. **Kratak opis:** Nastavnik, Student ili Moderator (u daljem tekstu korisnik sistema) pregleda statističke podatke nekog zadatka za vežbu.
2. **Učesnici:** Nastavnik, Student, Moderator
3. **Preduslovi:** /
4. **Postuslovi:** Korisnik sistema ima uvid u statističe podatke vezane za izabrani zadatak.
5. **Osnovni tok:**

   Korisnik sistema otvara listu svih zadataka izborom opcije „Zadaci za vežbu“.

   Sistem prikazuje listu zadataka, pri čemu uz svaki od zadataka postoji opcija „Statistike“.

   Korisnik sistema bira pomenutu opciju.

   Sistem prikazuje statističke podatke vezane za izabrani zadatak.
6. **Alternativni tokovi:** /

3 PROJEKTOVANJE

3.1 Projektovanje baze podataka

Radi preglednosti i boljeg razumevanja, baza podataka će takođe biti opisana kroz dva glavna modula: domaći zadaci i zadaci za vežbu. Neki od entiteta su bitni za oba modula, pa se iz tih razloga navode po dva puta (iz različitog ugla),

ali oni se odnose na jedan isti entitet (npr. ProgrammingLanguage ili Assignment).

3.1.1. Modul domaći zadaci

Detaljnom analizom slučajeva upotrebe, prepoznati su sledeći nezavisni entiteti, kao i podaci koji opisuju taj entitet:

1. ***Zadatak (Assignment)*** – Predstavlja jedan zadatak (kako domaći zadatak tako i zadatak za vežbu). Može se opisati pomoću sledećih podataka:
    - Naziv zadatka
    - Tekst zadatka
    - Vremensko ograničenje
    - Memorijsko ograničenje
    - Primeri za testiranje
    - Kategorija – u slučaju domaćeg zadatka ovo polje će imati vrednost 0
    - Autor zadatka – u slučaju domaćeg zadatka ovo polje će imati vrednost NULL
2. ***Domaći zadatak (Homework)*** – predstavlja jedan domaći zadatak. Može se opisati pomoću sledećih podataka:
    - Profesor koji je zadao zadatak
    - Kurs u okviru kog je zadat zadatak
    - Od kada do kada je moguće poslati rešenje zadatka (period izrade)
    - Broj poena koje zadatak nosi
    - Svi podaci o zadatku koji su navedeni u okviru entiteta Assignment
3. ***Rešenje domaćeg zadatka (HomeworkSolution)*** – predstavlja jedan pokušaj rešavanja domaćeg zadatka. Može se opisati pomoću sledećih podataka:
    - Domaći zadatak za koji je poslato rešenje
    - Student koji je pokušao da reši zadatak
    - Vreme kada je poslao rešenje
    - Rešenje zadatka (programski kod koji je student poslao)
    - Broj poena koje je sistem dodelio tom rešenju
4. ***Programski jezik (ProgrammingLanguage)*** – Predstavlja jedan programski jezik u kom je moguće slati rešenja zadataka. Može se opisati pomoću sledećih podataka:
    - Naziv programskog jezika

5. ***Kurs (Course)*** – Predstavlja jedan predmet (kurs) koji se drži u tekućoj školskoj godini. Može se opisati pomoću sledećih podataka:
    - Naziv predmeta
    - Oznaka predmeta
    - Semestar u kom se predmet sluša (1 ili 2)
6. ***Korisnik (User)*** – Predstavlja jednog korisnika sistema. Može se opisati pomoću sledećih podataka:
    - Korisničko ime
    - Šifra
    - Ime
    - Prezime
    - Tip korisnika (Nastavnik, Student, Moderator)
    - E-mail adresa
    - Slika
7. ***Nastavnik (Professor)*** – Predstavlja jednog nastavnika. Može se opisati pomoću svih podataka koji su već navedeni u tabeli User.
8. ***Student (Student)*** – Predstavlja jednog studenta. Može se opisati pomoću sledećih podataka:
    - Broj indeksa
    - Svi podaci koji su već navedeni u tabeli User

Primetimo da jednom domaćem zadatku može biti pridruženo više programskih jezika u kojima je moguće rešiti taj zadatak, kao i da jedan programski jezik može biti pridružen većem broju domaćih zadataka. Zbog toga nastaje asocijativni entitet ***HomeworkProgrammingLanguage.***

Takođe, jedan nastavnik može držati više predmeta (kurseva), ali i jedan predmet može držati više nastavnika. Kao posledica, nastaje asocijativni entitet ***ProfessorCourse.*** Slično, jedan student može slušati više predmeta i na jednom predmetu može biti upisano više studenata, odakle nastaje asocijativni entitet ***StudentCourse.***

Dobijeni dijagram entiteta i odnosa prikazan je na slici 6.

3.1.2. Modul vežbanje

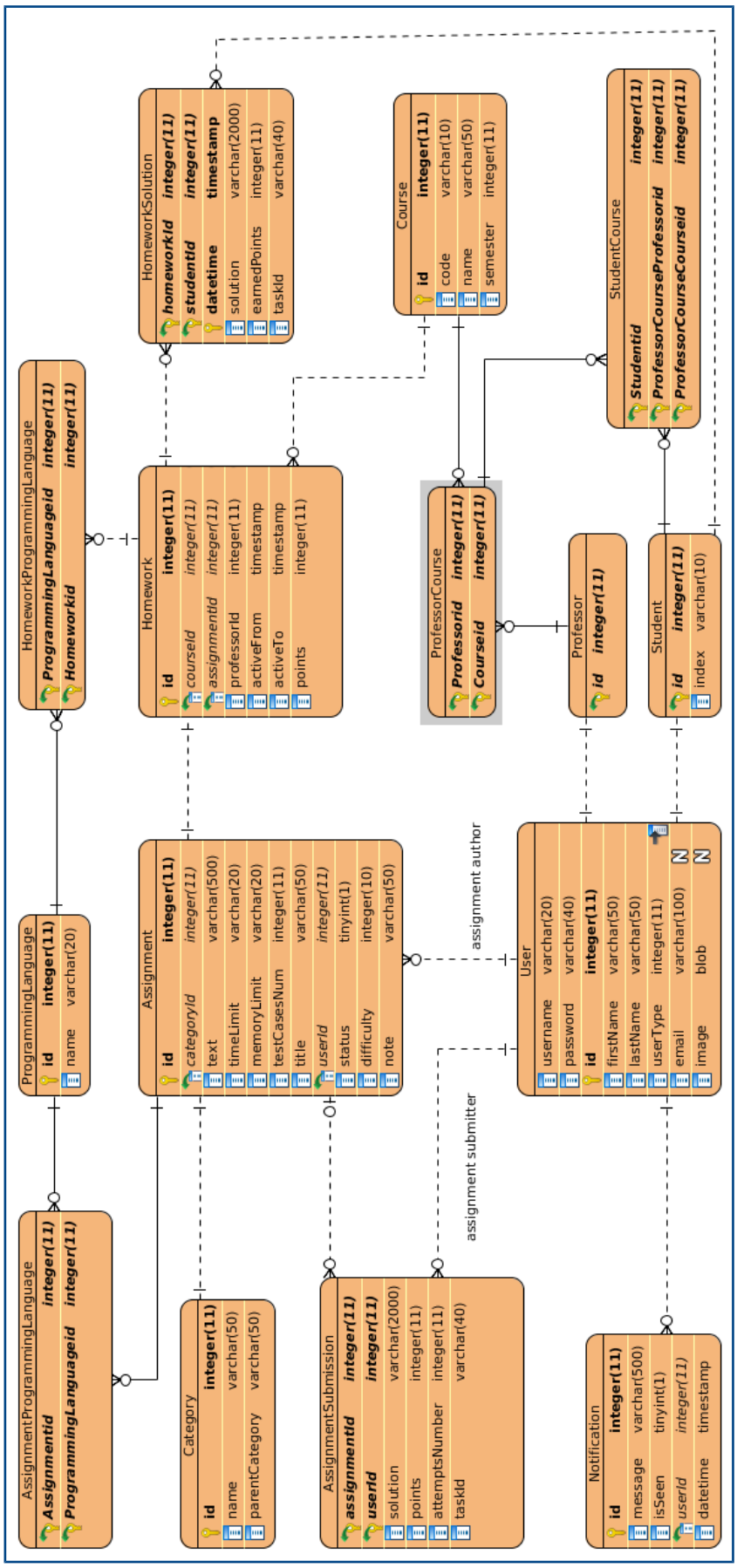
Detaljnom analizom slučajeva upotrebe, prepoznati su sledeći nezavisni entiteti, kao i podaci koji opisuju taj entitet:

1. ***Zadatak (Assignment)*** – predstavlja jedan zadatak za vežbu koji je predložio bilo koji korisnik sistema, a koji je odobren od strane moderatora sistema. Može se opisati pomoću sledećih podataka:
    - Naziv
    - Tekst zadatka

- Vremensko ograničenje za rešavanje
- Memorijsko ograničenje za rešavanje
- Kategorija u kojoj se zadatak nalazi
- Programski jezici u kojima je moguće rešiti zadatak
- Težina zadatka (vrednost 1-5)
- Test primeri
- Korisnik sistema koji je autor zadatka
- Polje koje će označavati status zadaka (da li je u pitanju predlog zadatka, zadatak u pripremi ili kompletiran zadatak)

2. ***Programski jezik (ProgrammingLanguage)*** – predstavlja jedan programski jezik u kom je moguće rešiti neki od zadataka. Može se opisati pomoću sledećih podataka:
   - Naziv programskog jezika
3. ***Kategorija (Category)*** – predstavlja jednu kategoriju zadatka za vežbu. Radi finije podele i bolje organizacije, svaka kategorija ima tačno jednu roditeljsku kategoriju. Može se opisati pomoću sledećih podataka:
   - Naziv kategorije
   - Naziv roditeljske kategorije
4. ***Rešenje zadatka (AssignmentSubmission)*** – predstavlja jedno rešenje nekog od zadataka za vežbu. Ovde se čuva samo najbolje rešenje za svakog korisnika sistema. Može se opisati pomoću sledećih podataka:
   - Korisnik koji je pokušao da reši zadatak
   - Zadatak koji je pokušao da reši
   - Rešenje koje je korisnik poslao
   - Procenat tačnosti rešenja
   - Broj pokušaja da se reši zadatak

Primetimo da jednom zadatku može biti pridruženo više programskih jezika u kojima je moguće rešiti taj zadatak, kao i da jedan programski jezik može biti pridružen većem broju zadataka. Odavde nastaje asocijativni entitet ***AssignmentProgrammingLanguage.*** Dobijeni dijagram entiteta i odnosa prikazan je na slici 6.



Slika 6: Dijagram entiteta i odnosa

3.2 Shema baze podataka

Prevođenjem entiteta i njihovih međusobnih odnosa u tabele dobija se shema baze podataka prikazana u tabeli 1.

Tabela	Opis
Assignment	Podaci o zadacima
Homework	Podaci o domaćim zadacima
ProgrammingLanguage	Podaci o programskim jezicima
HomeworkSolution	Podaci o rešenjima domaćih zadataka
User	Podaci o korisnicima
Professor	Podaci o nastavnicima
Student	Podaci o studentima
Course	Podaci o predmetima
ProfessorCourse	Podaci koji opisuju koji nastavnik drži koji predmet
StudentCourse	Podaci koji opisuju koji student sluša koji predmet kod kog nastavnika
HomeworkProgrammingLanguage	Podaci koji opisuju koji domaći zadatak je moguće rešiti u kom programskom jeziku
Category	Podaci o kategorijama zadataka za vežbu
AssignmentProgrammingLanguage	Podaci koji opisuju koji zadatak za vežbu je moguće rešiti u kom programskom jeziku
AssignmentSubmission	Podaci o rešenjima zadataka za vežbu
Notification	Podaci o obaveštenjima

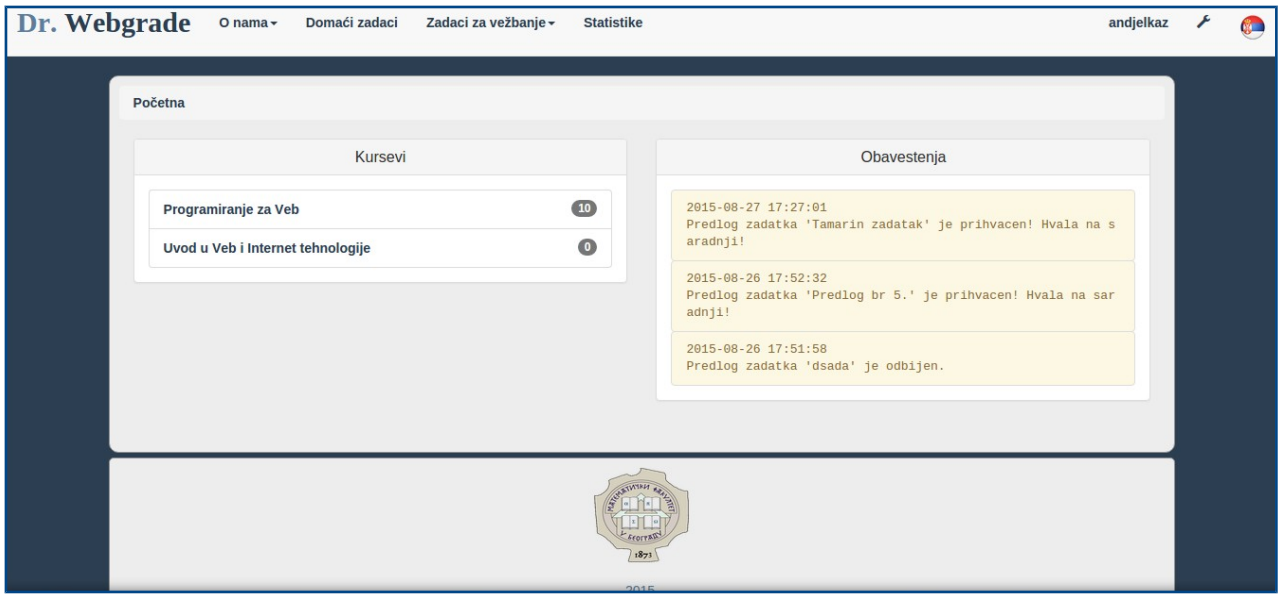
Tabela 1: Shema baze podataka

U dodatku A je dat detaljan opis svih tabela sheme baze podataka.

3.3 Korisnicki interfejs

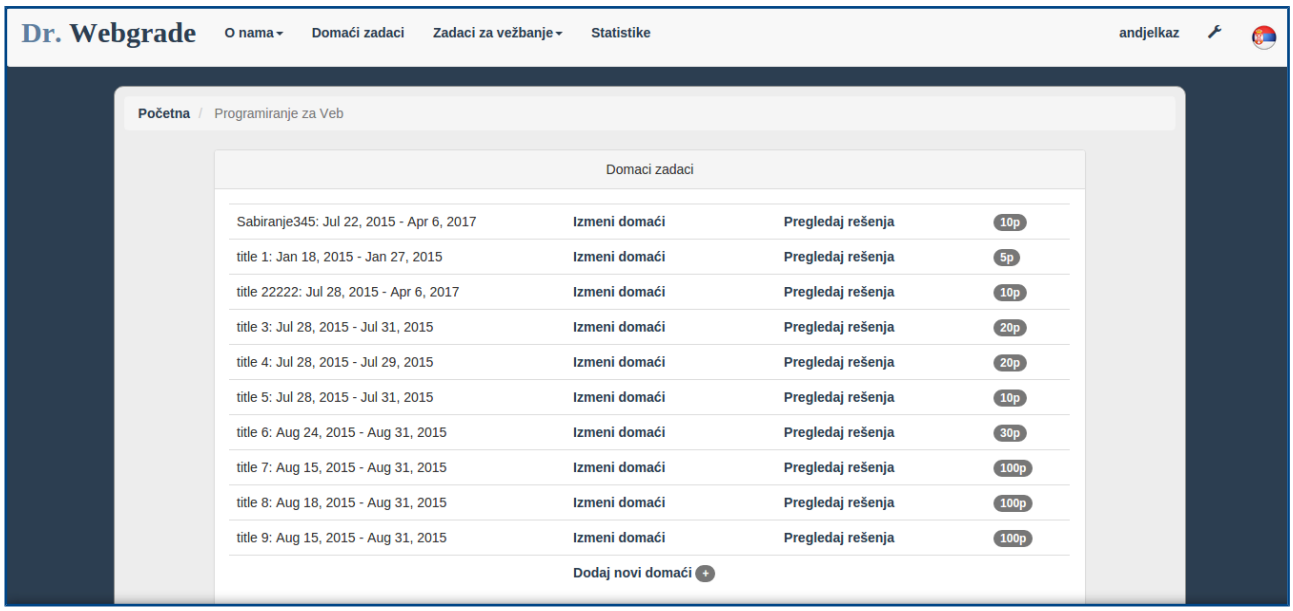
3.3.1 Deo aplikacije namenjen nastavnicima

Kada se uloguju na sistem, nastavnicima se prikazuje njihova početna stranica. Na njoj imaju pregled svih obaveštenja, kao i listu kurseva koje drže (slika 7). U nastavku naziva predmeta stoji broj koji predstavlja broj domaćih zadataka koji je zadat u okviru tog predmeta.



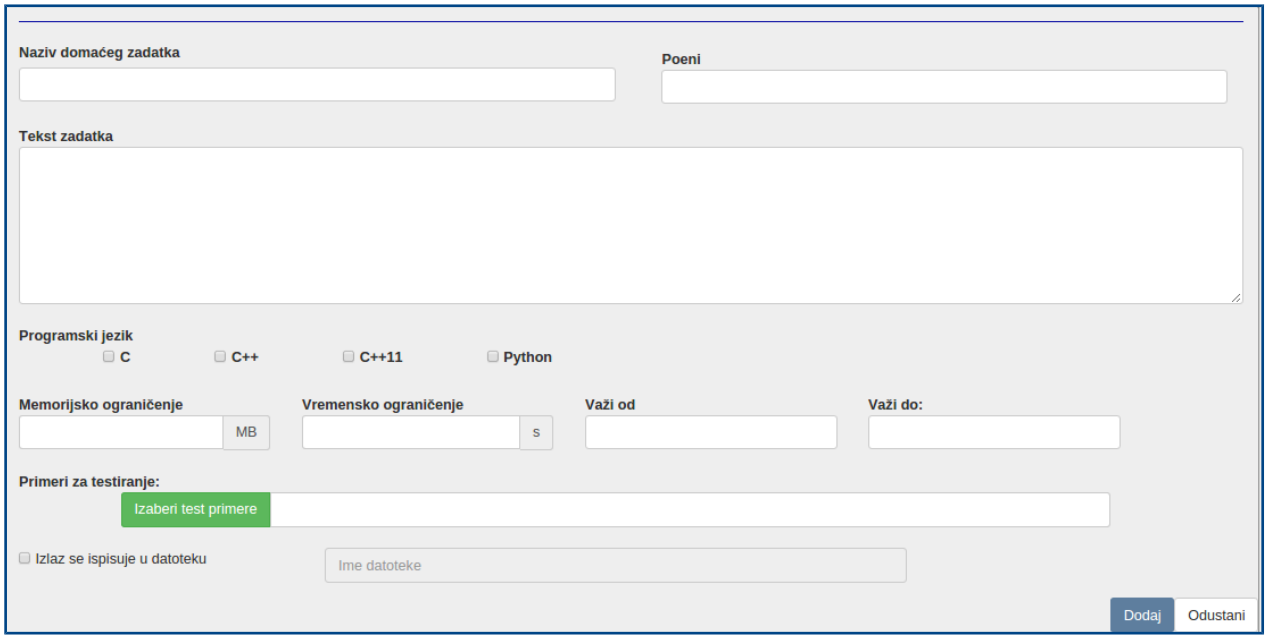
Slika 7: Početna stranica za nastavnike

Izborom nekog od predmeta, prikazuje se stranica sa listom svih domaćih zadataka koje je nastavnik zadao u okviru tog predmeta ili odgovarajuća poruka, ukoliko nastavnik nije zadao nijedan zadatak (slika 8). Primetimo da pored opcije za dodavanje novog domaćeg zadatka, za svaki od postojećih zadataka postoje dve opcije: „Izmeni domaći“ i „Pregledaj rešenja“.



Slika 8: Lista domaćih zadataka za izabrani kurs

Izborom opcije „Dodaj novi domaći“ nastavniku se otvara formular za unos novog domaćeg zadatka. (slika 9)



Slika 9: Formular za dodavanje novog domaćeg zadatka

Ukoliko nastavnik:

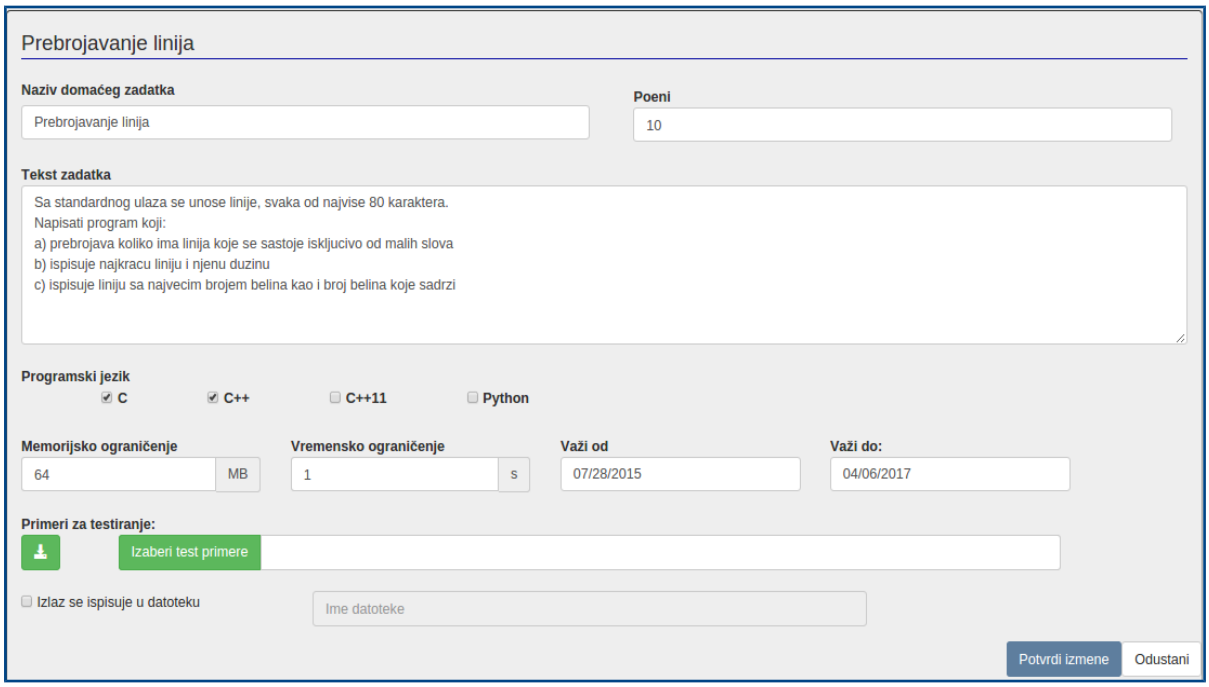
- unese broj poena koji je manji ili jednak 0
- ne izabere ni jedan programski jezik u kom je moguće rešiti zadatak
- unese vremensko ili memorijsko ograničenje koje je manje ili jednako 0
- ne izabere test primere

sistem će ispisati odgovarajuću poruku i zadatak neće biti sačuvan.

Ovde je bitno napomenuti na koji način se ispravno zadaju test primeri. Kako postoji više načina čitanja ulaza (standardni ulaz, argumenti komandne linije, datoteke), kao i ispisivanja izlaza (standardni izlaz, datoteka), a sam sistem omogućava korišćenje svih mogućih kombinacija ulaza, kao i izbor jednog od dva načina izlaza, format za zadavanje test primera je sledeći:

Ulaz koji student treba da očekuje sa standardnog ulaza je potrebno smestiti u fajlove sa nazivom *0.input*, *1.input*, ..., *n.input*, gde je n+1 ukupan broj test primera. Ulaz koji student treba da očekuje kao argumente komandne linije je potrebno smestiti u fajlove sa nazivom *0.args*, *1.args*, ..., *n.args*. Ulaz koji student treba da očekuje iz datoteka je potrebno smestiti u arhive sa nazivom *0.zip*, *1.zip*,..., *n.zip*. Vrlo je bitno napomenuti da je u ove arhive moguće smestiti više datoteka koji će se nalaziti u proizvoljnoj hijerarhiji direktorijuma.

Očekivani izlaz je potrebno smestiti u fajlove sa nazivom *0.output*, *1.output*, ..., *n.output*. Ukoliko je potrebno izlaz upisati u neku datoteku, naziv te datoteke je potrebno upisati u formular (slika 9), u suprotnom će se izlazom iz studentskog programa smatrati standardni izlaz.



Slika 10: Formular za izmenu domaćeg zadatka

Izborom opcije „Izmeni domaći“ (sa slike 8), prikazuje se formular prikazan na slici 9, popunjen podacima koje je nastavnik prethodno uneo za taj domaći (slika 10).

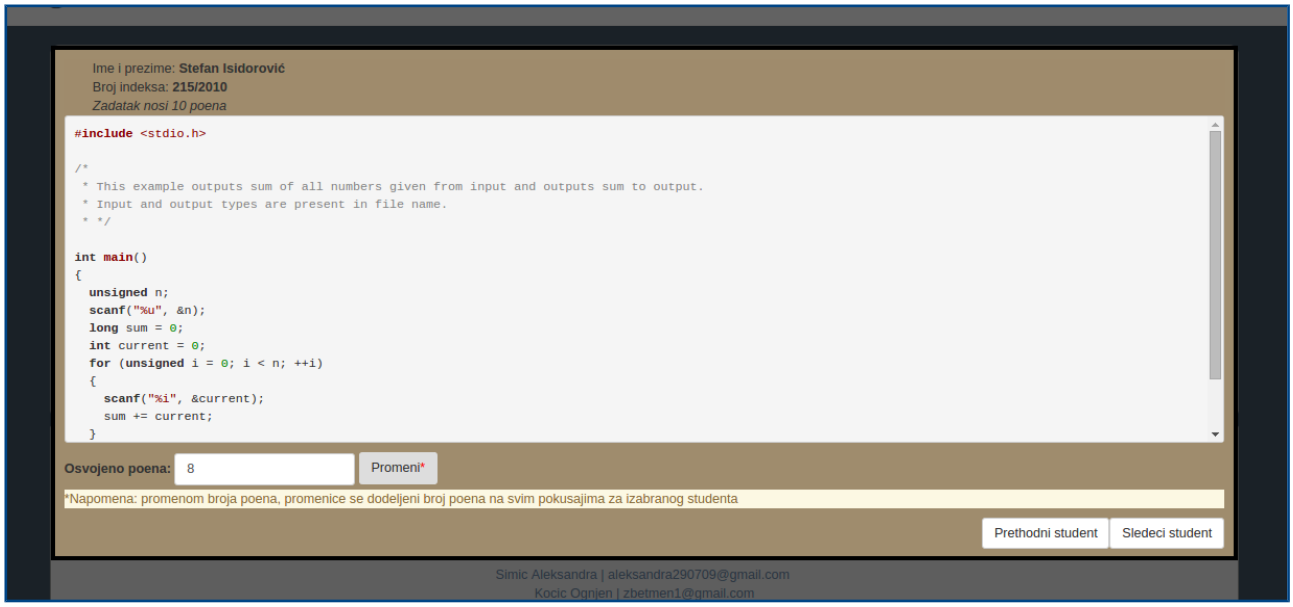
Ukoliko nastavnik želi da pogleda prethodno zadate test primere, sistem mu pruža tu mogućnost. Na ovom formularu postoji malo zeleno dugme (levo od opcije „Izaberi test primere“) koje služi za preuzimanje prethodno postavljenih test primera. Nakon preuzimanja ovih test primera, ako nastavnik ne zada nove, stari test primeri će ostati važeći.

Izborom opcije „Pregledaj rešenja“ (sa slike 8) prikazuje se lista studenata koji su pokušali da reše domaći zadatak. (slika 11)

Slika 11: Lista studenata koji su pokušali da reše domaći zadatak

Ovde se za svakog studenta prikazuju dve opcije: „Rešenje sa najviše poena“ i „Sva rešenja“.

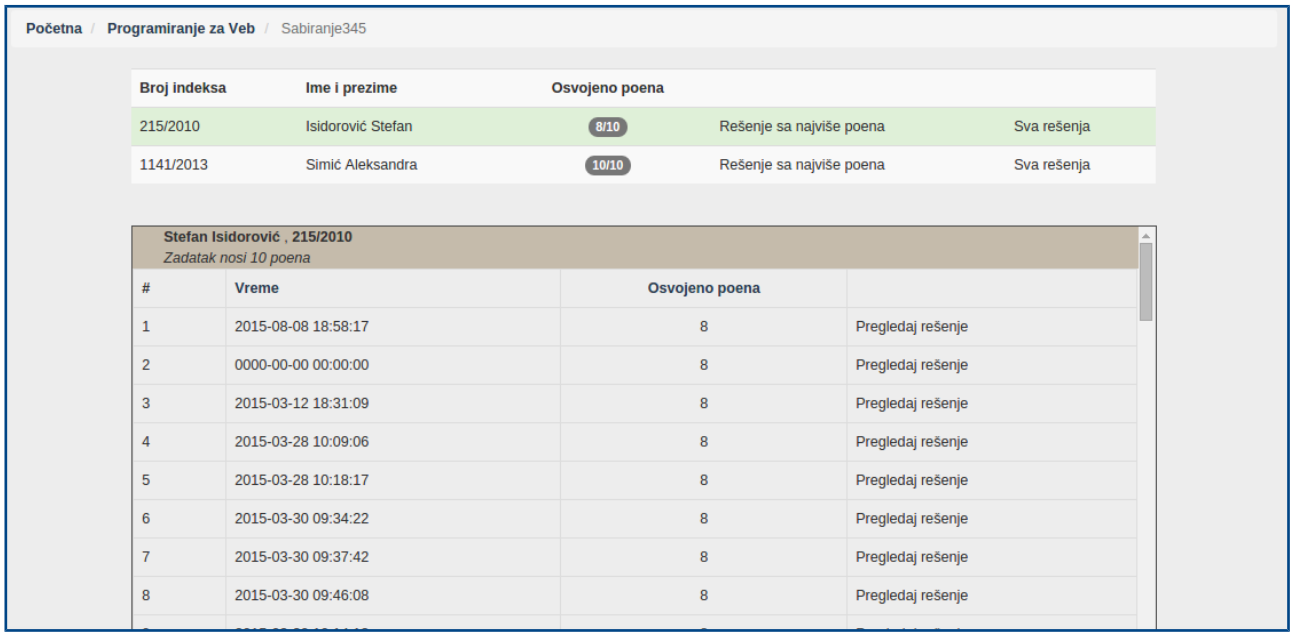
Izborom prve opcije nastavniku se otvara prozor u kome se prikazuje kod koji je okačio izabrani student. Ovo predstavlja najuspešniji pokušaj rešavanja zadatka za tog studenta.



Slika 12: Pregled studentskog koda

Ovde nastavnik ima opciju da promeni broj poena koje je sistem dodelio studentu, unosom broja poena i izborom opcije „Promeni“. Pored ovoga nastavnik može da prolazi kroz listu najboljih rešenja svih studenata izborom opcija „Prethodni student“, odnosno „Sledeći student“.

Izborom opcije „Sva rešenja“ sa slike 11, nastavniku se prikazuju svi pokušaji izabranog studenta da reši taj domaći zadatak. (slika 13)



Slika 13: Pregled svih pokušaja rešavanja zadatka za jednog studenta

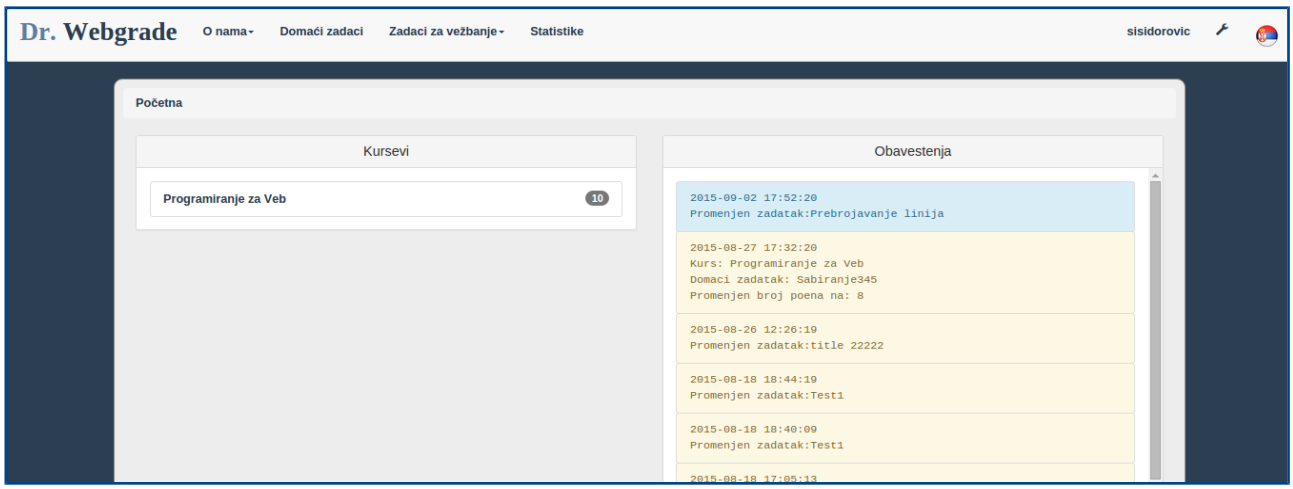
Primetimo da za svaki od pokušaja postoji prikazano vreme, broj osvojenih poena i opcija da se pregleda rešenje. Izborom opcije „Pregledaj rešenje“ prikazuje se prozor identičan onome na slici 12, samo što ovde postoje opcije „Sledeće rešenje“, odnosno „Prethodno rešenje“, koje služe za listanje okačenih

rešenja izabranog studenta.

3.3.2 Deo aplikacije namenjen studentima

Nakon što se uloguje na sistem, studentu se prikazuje njegova početna stranica na kojoj se nalazi lista kurseva koje sluša i sva njegova obaveštenja (slika 14).

Primetimo da su nova obaveštenja (ona koje student nije prethodno video) obojena plavom bojom.



Slika 14: Početna stranica za studenta

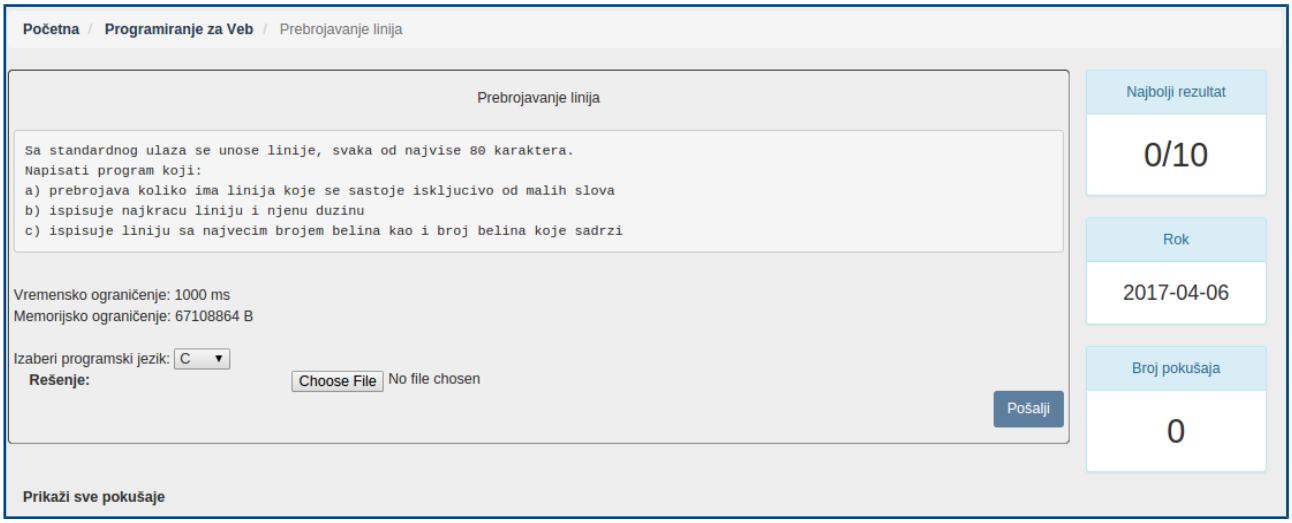
Izborom kursa, studentu se zatim prikazuje lista svih domaćih zadataka koji su zadani u okviru tog kursa. Ovi zadaci su podeljeni u dve grupe: aktivni zadaci i zadaci kojima je prošao rok za izradu. Za zadatke iz druge grupe student ne može da šalje rešenja.

Slika 15: Lista domaćih zadataka

Uz svaki zadatak je prikazano trajanje, kao i maksimalan broj poena koje je student osvojio na tom zadatku.

Klikom na naziv nekog od aktivnih domaćih zadataka studentu se prikazuje

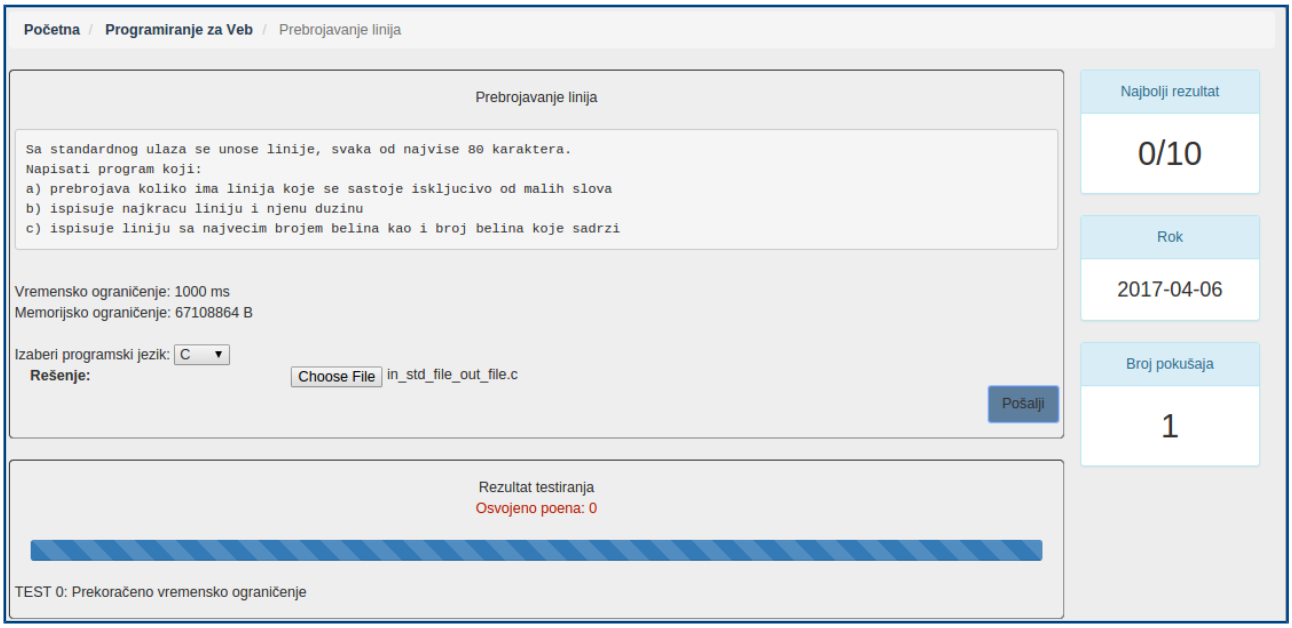
formular sa detaljima zadatka i opcijom za kačenje rešenja



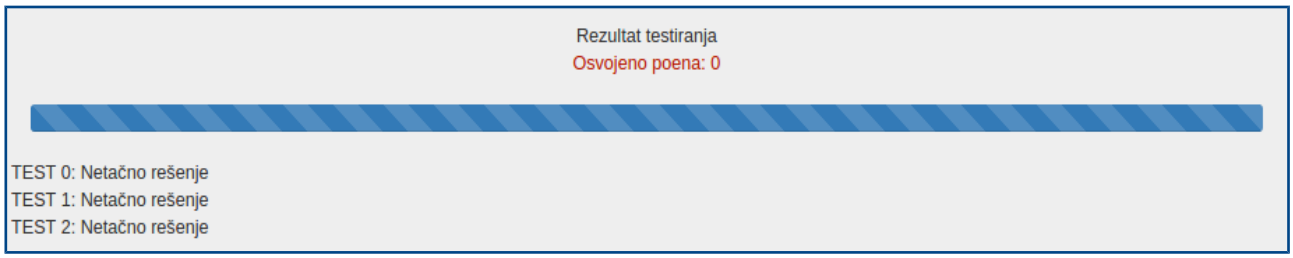
Slika 16: Formular za kačenje rešenja zadatka

Da bi se rešenje poslalo na pregledanje, neophodno je izabrati neki od ponudjenih programskih jezika i obezbediti da se ekstenzija okačene datoteke poklapa sa izabranim programskim jezikom.

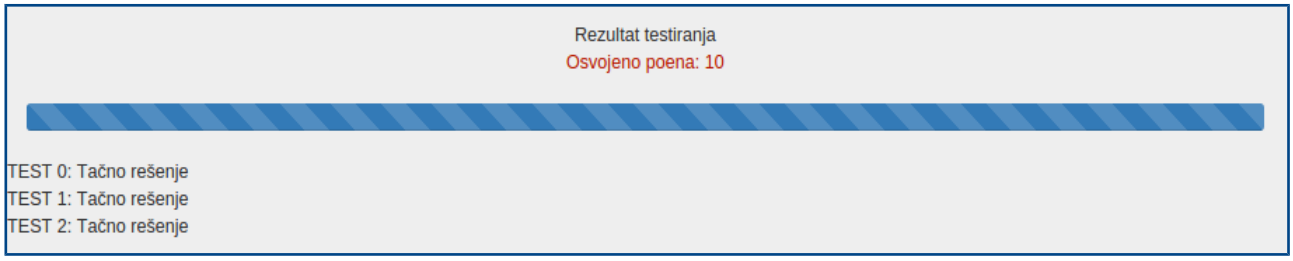
Izborom opcije „Pošalji“ rešenje se šalje na pregledanje. Rezultati testiranja se prikazuju odmah ispod. Kako koji test primer prođe ili ne prođe testove, tako se ti podaci prikazuju studentu (slike 17, 18, 19).



Slika 17: Primer kačenja rešenja



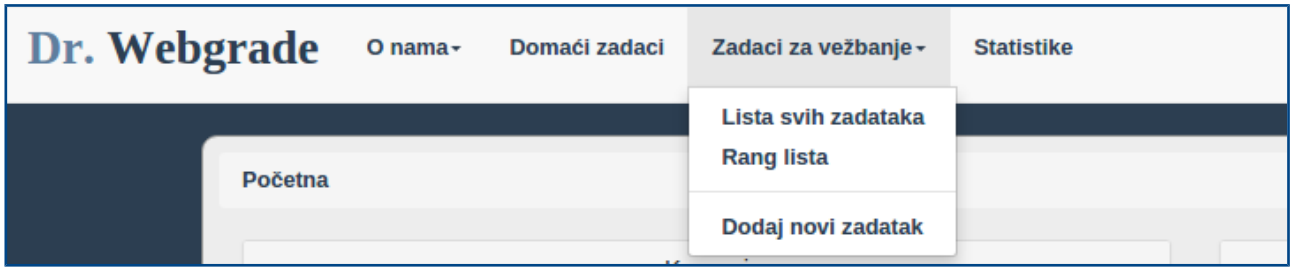
Slika 18: Primer rezultata testiranja za netačno rešenje



Slika 19: Primer rezultata testiranja za tačno rešenje

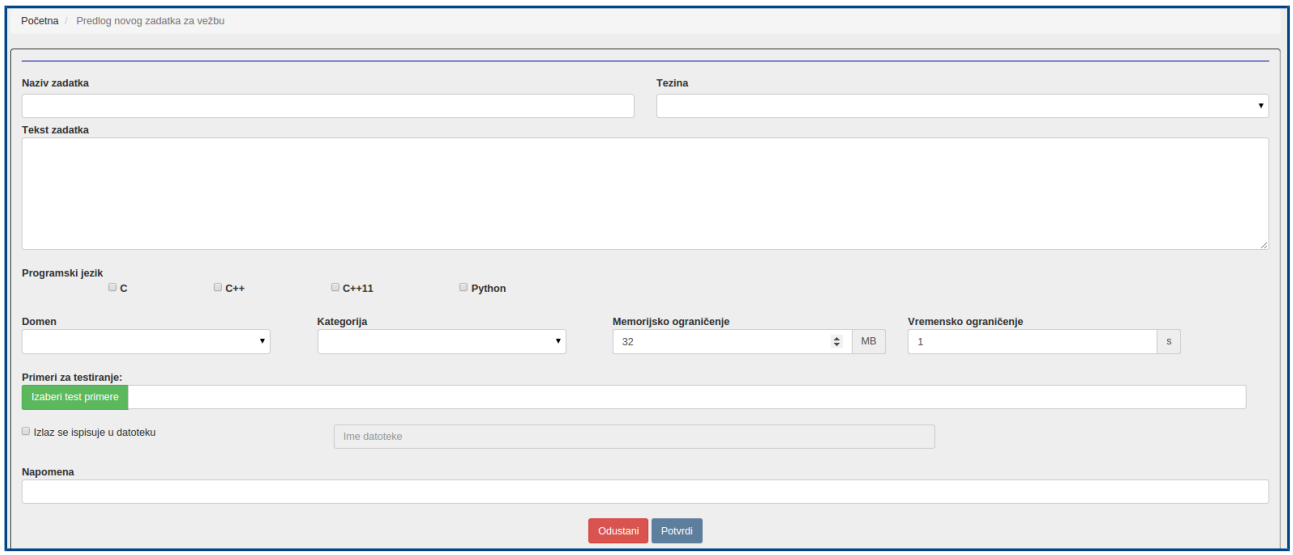
3.3.3 Deo aplikacije koji je zajednički za nastavnike i studente

I nastavnici i studenti (u daljem tekstu korisnici) u glavnom meniju imaju opciju „Zadaci za vežbu“. U okviru tog padajućeg menija, nalaze se 3 opcije: „Lista svih zadataka“, „Rang lista“ i „Dodaj novi zadatak“ (slika 20).



Slika 20: Deo menija u kom se nalaze opcije vezane za zadatke za vežbu

Izborom opcije „Dodaj novi zadatak“ korisnik može da predloži moderatoru novi zadatak za vežbu. Ovaj formular se razlikuje od formulara za zadavanje novog domaćeg zadatka po tome što je zadacima za vežbu pridružena težina, koja predstavlja ceo broj od 1 do 5 i služi da se na taj način „boduju“ ovi zadaci. Pored toga, svaki od zadataka za vežbu pripada nekoj kategoriji koja se nalazi u okviru nekog domena (roditeljske kategorije). Zbog toga je prvo potrebno izabrati domen, a zatim u okviru tog domena neku od ponuđenih kategorija. (slika 21)

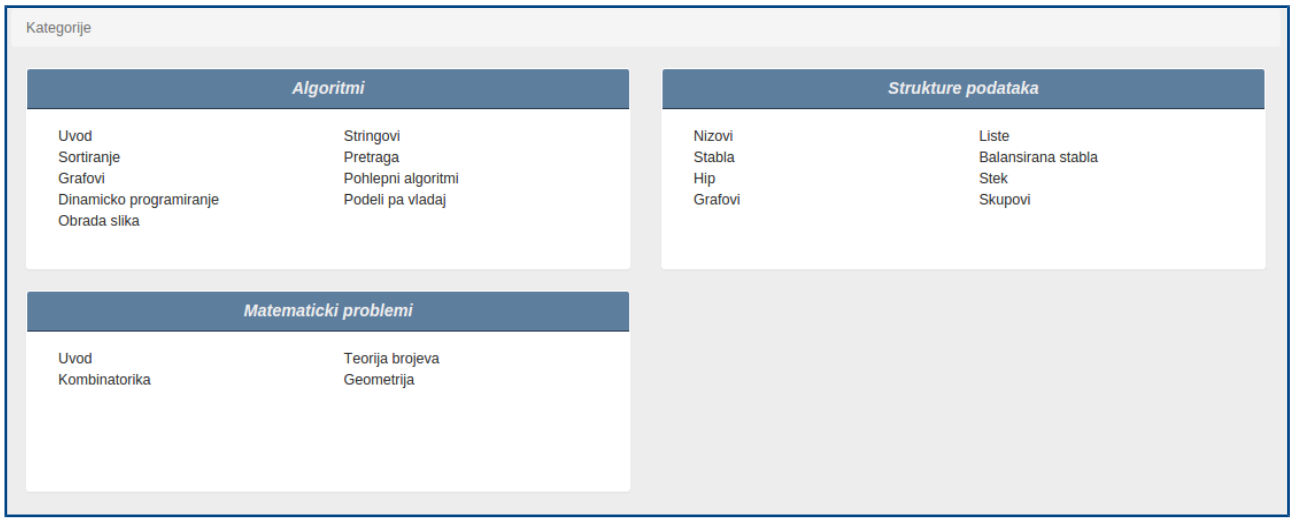


Slika 21: Formular za slanje predloga novom zadatka za vežbu

Na primer, jedan od postojećih domena je „Algoritmi“, a zatim se unutar njega nalaze kategorije kao što su „Grafovi“, „Niske“, „Sortiranje“... Takođe, ovde većina polja nije obavezna za unos, tj. ako korisnik nije siguran na koji način treba da popuni neka od polja formulara, može da ih ostavi praznim, a to će zatim moderator koji bude obrađivao predlog zadatka ispravno popuniti.

Iz ovih razloga, dodatno postoji i polje „Napomena“, gde korisnik koji predlaže zadatak može da napiše bilo kakve instrukcije za moderatora ukoliko smatra da je to potrebno.

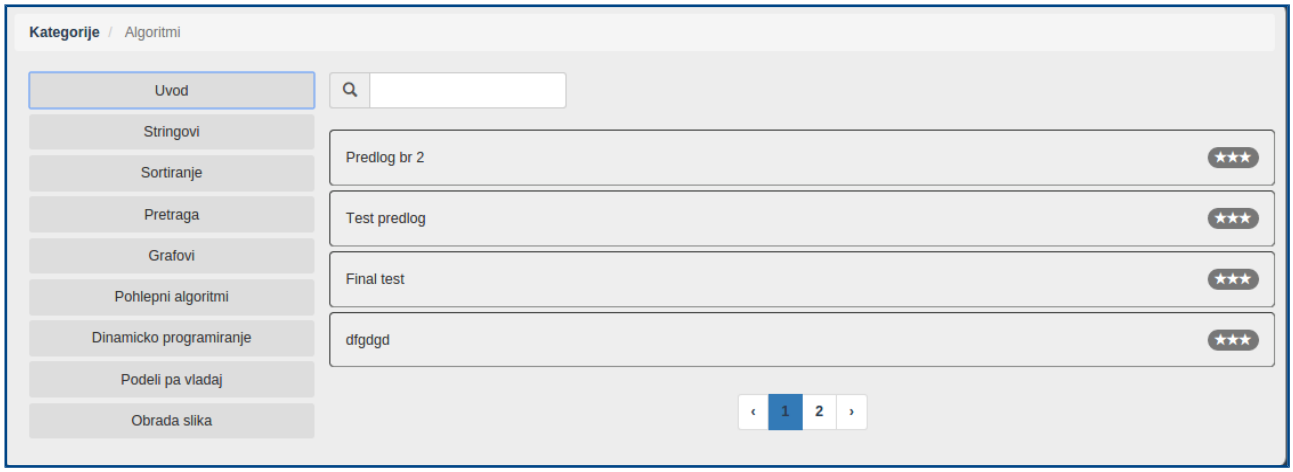
Izborom opcije „Lista svih zadataka“ iz glavnog menija (slika 20), korisniku se prikazuje lista svih domena sa kategorijama koje im pripadaju. (slika 22)



Slika 22: Lista domena i kategorija

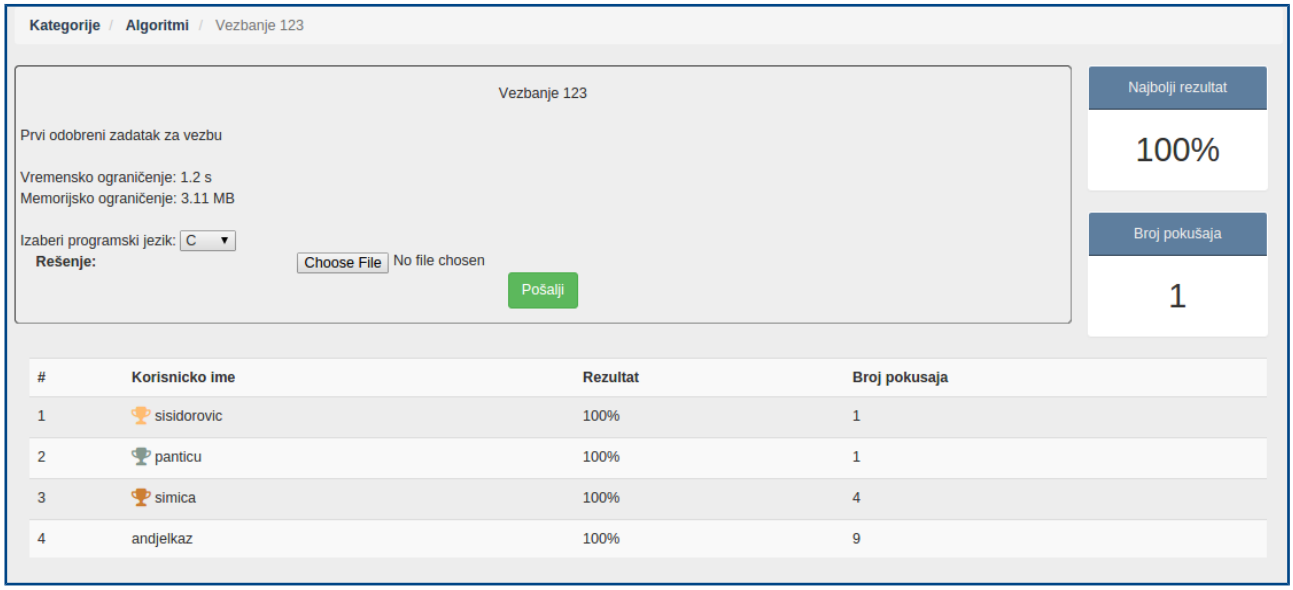
Izborom nekog od domena, prikazuje se stranica sa svim zadacima iz tog domena, razvrstanim po kategorijama (slika 23). Izborom kategorija sa leve strane, menja se spisak zadataka u središnjem delu. Primetimo da postoji polje za pretragu, koje se koristi za pretragu zadataka po naslovu. Pored ovoga,

zadaci su, radi bolje preglednosti, podeljeni na stranice (eng. pagination). Na slici su, radi ilustracije, postavljena 4 zadatka na jednu stranicu. U praksi će to biti 20 ili 30 zadataka. Zvezdice koje stoje uz svaki zadatak označavaju njegovu težinu.



Slika 23: Lista zadataka za vežbu, razvrstanih po kategorijama

Klikom na naslov zadatka, otvara se stranica za njegovo rešavanje, vrlo slična onoj za rešavanje domaćeg zadatka. Proces ocenjivanja zadatka je identičan, a samim tim i prikaz rezultata, samo što ne postoji broj osvojenih poena, već procent tačnosti rešenja. Na ovoj stranici se nalazi i rang lista rešenja za taj zadatak, sortirana prvo po rezultatu testiranja, a zatim po broju pokušaja (slika 24).



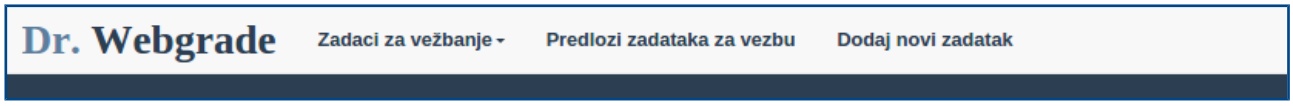
Slika 24: Rešavanje zadatka za vežbu

Izborom opcije „Rang lista“ iz glavnog menija prikazuje se globalna rang lista svih zadataka za vežbu. Skor se računa kao suma svih težina rešenih zadataka pomnožena sa osvojenim procentom na tim zadacima (slika 25).

Slika 25: Prikaz rang liste

3.3.4 Moderatorski deo aplikacije

Nakon što se uloguje, moderatoru se prikazuje njegova početna stranica. Glavni meni je prikazan na slici 26.

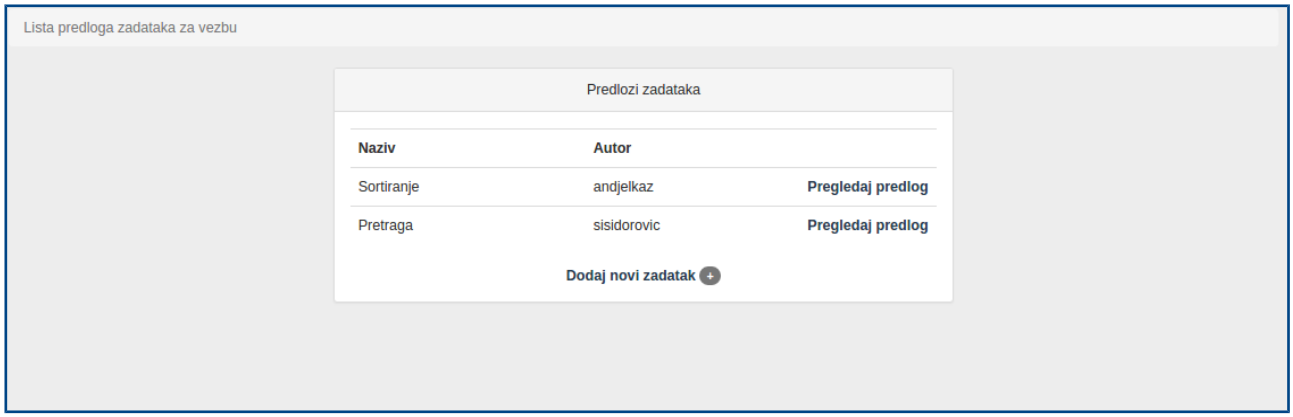


Slika 26: Glavni meni na moderetorskoj stranici

Opcija „Zadaci za vežbanje“ izgleda isto kao i kod Studenta i Nastavnika.

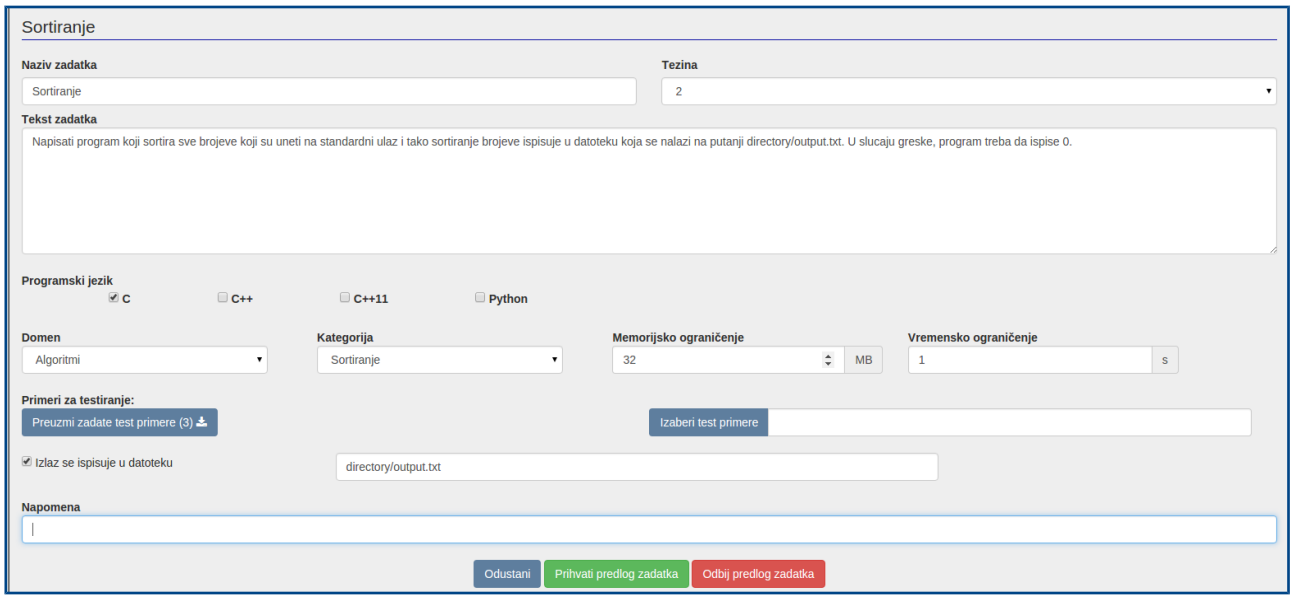
Opcija „Dodaj novi zadatak“ predstavlja prikaz formulara za zadavanje novog zadatka za vežbu. Ovaj formular je identičan kao kod opcije „Predlaganje zadatka za vežbu“ kod ostalih korisnika, samo što se u slučaju moderatora ovaj zadatak direktno snima u bazu među zadatke za vežbu, jer ga nije potrebno dodatno odobriti.

Opcija „Predlozi zadataka za vežbu“ predstavlja prikaz liste svih predloženih zadataka za vežbu koji do sada nisu odobreni ili odbijeni.



Slika 27: Lista predloga zadataka za vežbu

Izborom opcije „Pregledaj predlog“ moderatoru se prikazuje popunjen formular za zadavanje novog zadatka za vežbu (slika 28).

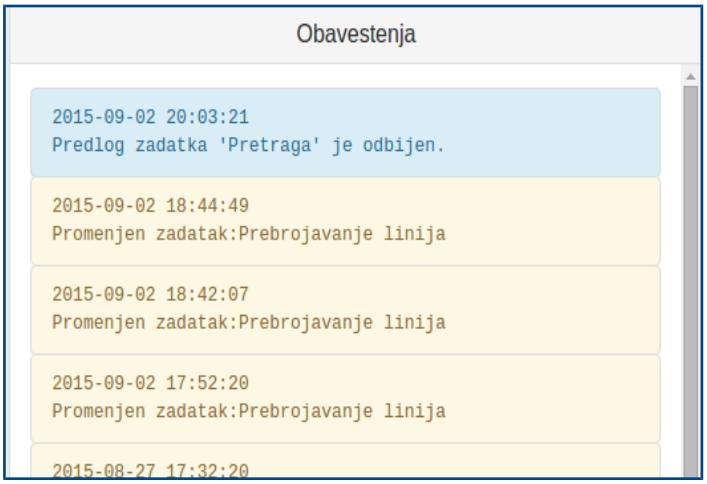


Slika 28: Obrada predloga zadatka za vežbu

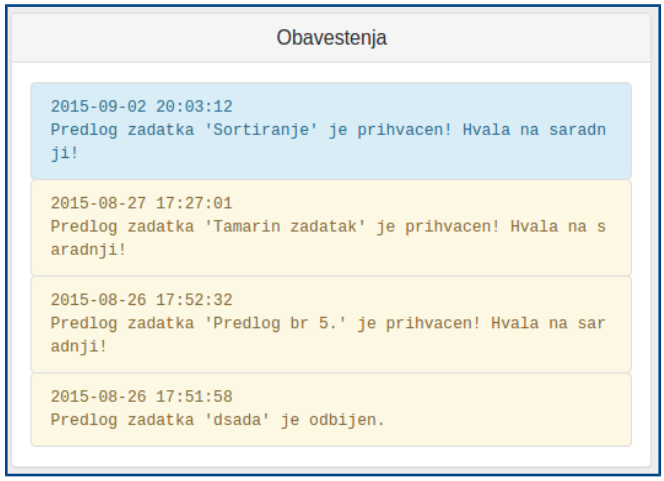
Moderator može da promeni sve podatke na formularu koje smatra pogrešnim. Takodje, ima opciju da preuzme postavljene test primere, da ih proveri i da okači izmenjene ukoliko smatra da treba. Ako ne okači neke druge test primere, prethodno okačeni test primeri će se smatrati ispravnim.

Moderator ovde ima 3 opcije:

1. Odustani – da napusti obradu predloga zadatka
2. Prihvati predlog zadatka – zadatak se smešta među sve ostale zadatke za vežbu i postaje dostupan za rešavanje i korisnik koji ga je predložio biva obavešten o tome (slika 29).
3. Odbaci predlog zadatka – zadatak se trajno briše sa sistema i korisnik koji ga je predložio biva obavešten o tome (slika 30).



Slika 29: Obaveštenje o odbijanju predloga



Slika 30: Obaveštenje o prihvatanju predloga

4 IMPLEMENTACIJA

4.1 Baza podataka

Za implementaciju aplikacije korišćena je baza podataka MySQL. Radi lakšeg upravljanja podacima, korišćen je i PhpMyAdmin [3]. To je besplatan alat, pisan u jeziku PHP, koji podržava izvođenje raznih operacija nad MySQL bazama podataka. Osnovna razlika u odnosu na rad iz komandne linije je u tome što je korisniku ovog alata pružen jasan korisnički interfejs. Putem ovog interfejsa mnoge stvari se mogu brže i efikasnije uraditi (pogotovo pregledavanje struktura tabela i podataka u njima). Jedina mana se ogleda u tome što je nezgodno ažurirati bazu podataka sa većim količinama podataka. Međutim, za operacije ovog tipa najčešće se koriste *pomoćni programi* (eng. utilities), kao što je na primer *LOAD*, pa PhpMyAdmin predstavlja dobar izbor.

4.2 Serverski deo aplikacije

Kao što je prethodno rečeno, serverski deo aplikacije je implementiran u jeziku PHP. Ovaj sloj ima tri glavne uloge:

1. Komunikacija sa bazom podataka
2. Primanje zahteva i isporučivanje rezultata
3. Komunikacija sa ocenjivačem zadataka

4.2.1 Komunikacija sa bazom podataka

Ova komunikacija se vrši korišćenjem ekstezije „PHP Data Object“ (PDO) koja služi za pristupanje bazi podataka iz programskog jezika PHP. PDO pruža takozvani apstraktni sloj za *pristupanje* podacima (end. data-access abstraction layer) koji omogućava da, bez obzira na to koju bazu podataka koristimo, možemo da koristimo iste funkcije za izvršavanje upita i dohvatanje podataka. Ali da ne dođe do zabune, PDO ne omogućava apstrakciju baze podataka (eng. database abstraction), što znači da nema opciju da preformuliše napisane upite, tako da oni mogu da se koriste na nekoj drugoj bazi podataka.

```
self::$db->connection = new PDO( 'mysql:localhost;dbname=web4graderDb',
                                  'root',
                                  '');
```

Programski kod 1: Konekcija na bazu podataka korišćenjem PDO konstruktora

U programskom kodu 1 je prikazan način na koji se vrši povezivanje sa bazom podataka. Kao što vidimo, dovoljno je pozvati konstruktor PDO objekta, koji kao prvi argument prima informacije o bazi podataka, hostu i nazivu baze podataka. Naredna dva argumenta su korisničko ime i šifra.

Taj objekat se dalje koristi pri izvršavanju upita nad bazom i dohvatanja podataka. Neke od najčešće korišćenih funkcija su:

1. public PDOStatement **PDO::prepare**(string $statement [,array options])
2. public bool **PDOStatement::bindParam**(mixed $param, mixed &$var [, int $dataType = PDO::PARAM_STR [,int $length [,mixed $options]]])
3. public bool **PDOStatement::execute** ([array $input_params])
4. public array **PDOStatement::fetchAll**([int $fetchStyle [, mixed $fetch_arg [, array ctro_args]]])

U nastavku sledi kratak opis svake od njih.

Funkcija ***prepare*** priprema SQL upit za izvršavanje.

Kada su upiti oblika „select * from tabela where ime='Ana'“, ne postoji razlog za bilo kakvo pripremanje ovih upita. Ali, ako imamo upite sa nekim parametrom (npr. Ana se prosledjuje kao parametar smešten u promenljivu $ime), onda tu postoje 2 pristupa. Prvi pristup je naivan, konstruiše se upit jednostavnim nadovezivanjem niski (programski kod 2).

```
$upit = "select * from tabela where ime='".$ime."'";
```

Programski kod 2: Konstruisanje upita nadovezivanjem niski

U ovom jednostavnom primeru, izgleda kao da je ovaj pristup korektan. Ali treba imati u vidu da su parametri koji stižu za ove upite, najčešće parametri koji su stigli sa klijentske strane, unešeni od strane korisnika. Zlonameran korisnik bi umesto da unese željeno ime, mogao da unese „ ' or 1=1--“. Na ovaj način se, nadovezivanjem niski, dobija upit prikazan u programskom kodu 3.

```
$upit = "select * from tabela where ime=' ' or 1=1 --' ";
```

Programski kod 3: Upit dobijen nadovezivanjem niski

Na ovaj način će zlonameran korisnik doći do svih podataka iz neke tabele jer će uslov u *where* sekciji uvek biti tačan, a `--` je oznaka za komentar i sve što je navedeno nakon toga (u ovom slučaju samo jedan apostrof) će se ignorisati.

Ovo je poznat bezbednosni problem i zove se „Umetanje SQL upita“ (*eng. SQL injection*).

Kao rešenje za ovaj problem koriste se takozvani pripremljeni upiti. Ideja je da se na sva mesta u upitu gde treba da se nadje neki prosledjeni parametar stavi oznaka oblika :ime_parametra. Sam upit u sebi može imati više ovakvih oznaka.Njih je posle potrebno zameniti pravim vrednostima tih parametara.

Funkcija *prepare* se koristi baš u ovu svrhu – da pripremi upite, tako da se te oznake posle mogu povezati sa svojim parametrima i da sam upit može ispravno da se izvrši. Druga prednost korišćenja pripremljenih upita su performanse. U trenutku poziva *prepare* funkcije vrši se sintaksa provera upita. Ukoliko isti upit treba da se izvrši više puta sa različitim parametrima, dovoljno je samo jednom pripremiti taj upit, što predstavlja značajnu uštedu u odnosu na situaciju gde u petlji svaki put pravimo novi upit nadovezivanjem niski.

Kao rezultat izvršavanja *prepare* funkcije dobija se objekat tipa *PDOStatement*.

Funkcija ***bindParam*** se koristi za povezivanje oznaka oblika :ime_parametra i konkretnih prosleđenih parametara. Treba obratiti pažnju na to da je drugi argument referenca na prosleđeni parametar, što znači da će se taj parametar ubaciti u upit tek pri pozivu *execute* funkcije.

Funkcija **execute** izvršava upit koji je prethodno pripremljen pozivom *prepare* funkcije. Ukoliko upit sadrži parametre, neophodno je prethodno pozvati *bindParam* funkciju, inače se poziv *execute* funkcije neće ispravno izvršiti.

Funkcija ***fetchAll*** dohvata sve redove koje je upit vratio i smešta ih u niz.

Primer korišćenja svih ovih funkcija naveden u programskom kodu 4.

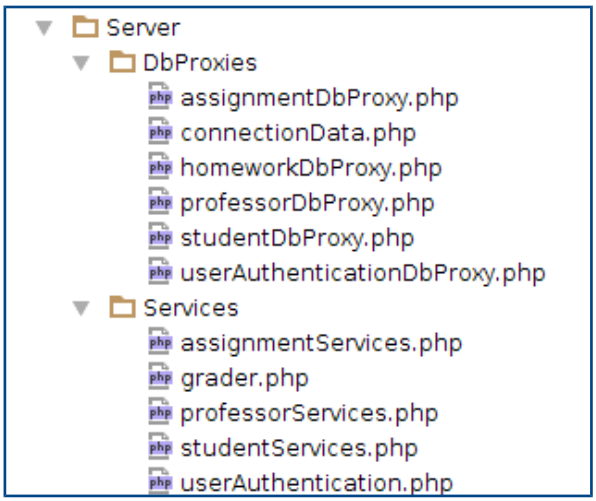
```
public function getAllStudentsForCourse($professorId, $courseId) {
    $query = "select *".
             "from web4graderDb.Student s ".
             "join web4graderDb.StudentCourse sc on sc.StudentId = s.Id".
             "where sc.CourseId = :cid and sc.ProfessorId = :pid";
    $stmt = self::$db->connection->prepare($query);
    $stmt->bindParam(":pid", $professorId, PDO::PARAM_INT);
    $stmt->bindParam(":cid", $courseId, PDO::PARAM_INT);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_OBJ);
}
```

Programski kod 4: Primer korišćenja PDO funkcija

Ovde je ilustrovano korišćenje nekih najosnovnijih funkcija. PDO ekstenzija je vrlo bogata i dobro dokumentovana i više informacija o tome se može naći na zvaničnom sajtu PHP-a [4].

4.2.2 Prihvatanje zahteva i isporučivanje rezultata

Serverski deo aplikacije je sa stanovišta organizacije podeljen na dva dela. Prvi deo čine php fajlovi koji služe za komunikaciju sa bazom podataka. U njima se nalaze samo funkcije koje vrše izvršavanje upita, sličnih onome koji je naveden u programskom kodu 4. Drugi deo aplikacije čine servisi. Oni primaju zahteve od klijenta, zatim pozivaju odredjene funkcije baze podataka i te rezultate prosleđuju dalje klijentu.



Slika 31: Organizacija datoteka na serveru

Pogledajmo organizaciju datoteka na serveru, prikazanu na slici 31. U direktorijumu *DbProxies* se nalaze prethodno pomenute datoteke sa funkcijama za komunikaciju sa bazom podataka, dok se u direktorijumu *Services* nalaze servisi koji koriste te funkcije.

Svi servisi su po organizaciji vrlo slični, pa će biti prikazan jedan od njih, npr. **professorServices**:

Na početku datoteke je potrebno uključiti php datoteke koje sadrže funkcije za komunikaciju sa bazom podataka, a koje će biti korišćene. U ovom slučaju to su homeworkDbProxy i professorDbProxy (Programski kod 5).

```
include_once("../DbProxies/homeworkDbProxy.php");
include_once("../DbProxies/professorDbProxy.php");
```

Programski kod 5: Uključivanje potrebnih datoteka

Dalje, zahteve upućene ka professorServices može slati samo nastavnik koji je ulogovan na sistem. Iz tog razloga se vrši provera da li je taj uslov ispunjen (programski kod 6).

```
session_start();
if(isset($_SESSION['userTypeName']) && $_SESSION['userTypeName'] !=
"professor")
    return;
```

Programski kod 6: Provera da li je zahtev došao od ulogovanog nastavnika

Zatim je potrebno instancirati objekte za komunikaciju sa bazom podataka (koji su implementirani kao Unikat) i ustanoviti koji metod je korišćen u zahtevu (GET, POST, PUT, DELETE, ...) (programski kod 7).

```
$method = $_SERVER['REQUEST_METHOD'];
$dbHw = HomeworkDbProxy::getInstance();
$dbProf = ProfessorDbProxy::getInstance();
```

Programski kod 7: Dohvatanje metoda i instanciranje objekata za komunikaciju sa bazom podataka

U nastavku datoteke se nalazi prvo grananje po tome koji je metod u pitanju, pa zatim grananje po traženoj akciji (programski kod 8).

```
if($method == "POST"){
        switch ($_POST['action']){
            case 'updateHomework':
                  ...
                break;
            case 'addHomework':
                  ...
                break;
            ...
        }
}
else if ($method == 'GET') {
    switch($_GET['action']){
        case 'initCourseInfo':
            ...
            break;
        case 'initHomeworkDetails':
            ...
            break;
      ...
    }
}
```

Programski kod 8: Prikaz grananja prvo po metodi, pa zatim po traženoj akciji

Na primer, kada profesor uđe na stranicu za neki kurs, potrebno je da mu se prikažu svi domaći zadaci na tom kursu i lista svih studenata koji slušaju taj kurs. U tom trenutku se vrši GET poziv sa akcijom `initCourseInfo`. Kod koji će se izvršiti prikazan je u programskom kodu 9.

```
case 'initCourseInfo':
    $courseId = $_GET['courseId'];
    $professorId = $_GET['professorId'];
    $res = new stdClass();
    $res->students = $dbProf->getAllStudentsForCourse($professorId,
$courseId);
    $res->homeworkList = $dbHw->getHomeworkList($professorId, $courseId);
    echo json_encode($res);
  break;
```

Programski kod 9: Izvršavanje akcije „initCourseInfo“

Prvo se, iz globalnog asocijativnog GET niza, pročitaju podaci koji su primljeni na server. Zatim se pozovu funkcije koje, na osnovu prosleđenih parametara, čitaju podatke iz baze podataka. Na kraju se ti podaci vraćaju klijentu.

Slično važi i za sve ostale zahteve i servise.

4.2.3 Komunikacija sa ocenjivačem

Iako mu se pristupa direktno iz PHP-a, ocenjivač je spoljna komponenta koja ne pripada ovom jeziku. Na početku datoteke u kojoj želimo da koristimo ocenjivač, potrebno je uključiti datoteku *grader.php* (programski kod 10).

```
include_once('grader.php');
```

Programski kod 10: Uključivanje interfejsa ocenjivača

Pored uključivanja navedene datoteke, potrebno je da ocenjivač bude dostupan PHP interpreteru. Interfejs ocenjivača je implementiran kao PHP

ekstenzija. Zbog toga je potrebno da se odgovarajuća dinamička biblioteka, koja ovu ekstenziju implementira, nalazi u podrazumevanom direktorijumu za PHP ekstenzije. Dodatno, potrebno je u PHP konfiguraciji (php.ini datoteka) naznačiti da želimo da koristimo ovu ekstenziju, navođenjem *extension= libgrader_php.so*, pri čemu je *libgrader_php.so* pomenuta dinamička biblioteka.

Sledeće funkcije čine interfejs ocenjivača:

1. grader::save_tests($relativePathToTestDir,$zippedData,$len)
2. grader::submit_task($testsDir, $fileName, $fileCont);
3. grader::get_task_status($taskId)

Funkcija save_tests je uvedena zato što postoji mogućnost da se ocenjivač i Veb server nalaze na različitim mašinama. U tom slučaju, potrebno je snimiti test primere na udaljenu mašinu gde se ocenjivač nalazi, što je i svrha ove funkcije. Ovo se radi da bi se izbeglo ponovljeno slanje istih test primera, koji će biti korišćeni prilikom ocenjivanja više različitih rešenja jednog zadatka. Prilikom snimanja test primera navodi se isključivo relativna putanja.

Funkcija submit_task vrši slanje studentskog rešenja na obradu ocenjivaču. Kao parametre prima putanju do direktorijuma gde se nalaze test primeri, ime datoteke sa rešenjem i njen sadržaj, a kao povratnu vrednost vraća identifikator pomoću koga se mogu dohvatati informacije o izvršavanju studentskog rešenja. Ispravan poziv ove funkcije je dat u programskom kodu 11.

```
$taskId = grader::submit_task($path, $fileName, $fileContent);
```

Programski kod 11: Primer ispravnog poziva funkcije submit_task

Ono što je bitno za performanse Veb aplikacije je da ovaj poziv nije blokirajući. Ocenjivač će odmah vratiti pomenuti identifikator, tj. neće se čekati na kraj obrade rešenja.

Funkcija get_task_status služi za proveru trenutnog stanja obrade rešenja zadatka. Prima identifikator koji je vratila funkcija submit_task i kao povratnu vrednost vraća status. Ovaj status se nalazi u JSON formatu. Ispravan poziv ove funkcije dat je u programskom kodu 12.

```
$response = grader::get_task_status($taskId);
```

Programski kod 12: Primer ispravnog poziva funkcije get_task_status

Kada se pozove funkcija json_decode nad objektom $response, dobija se objekat koji ima polja *state, test0, test1, ..., testn*. Polje *state* može imati vrednosti:

- *waiting* – zadatak nije stigao na red za ocenjivanje
- *finished* – ocenjivanje je završeno
- *running* – ocenjivanje je u toku
- *compiling* – kompajliranje programa je u toku
- *internal*_error – interna greška ocenjivača

- *malformed_request* – greška pri pozivu ocenjivača
- *compile_error* – greška pri kompilaciji

Ocenjivač ocenjuje rešenje tako što proverava njegovu tačnost na zadatim test primerima. Kako ocenjivač završi sa kojim test primerom, tako taj rezultat testiranja postaje dostupan. Ti rezultati se nalaze u poljima sa nazivom *test0, ..., testn*. Ova polja mogu imati vrednost:

- *correct_result* – tačan rezultat
- *incorrect_result* – neispravan rezultat
- *memory_limit* – prekoračeno memorijsko ograničenje
- *time_limit* – prekoračeno vremensko ograničenje
- *runtime_error* – greška pri izvršavanju

Proces ocenjivanja studentskog rešenja je sledeći:

1. Student šalje rešenje na server
2. Na serveru se poziva funkcija submit_task
3. Server vraća klijentu poruku da je rešenje uspešno poslato na obradu
4. Klijent šalje zahtev za proveru stanja
5. Server poziva funkciju get_task_status i vraća klijentu status
6. Klijent prikazuje rezultate do tada obrađenih test primera

   Koraci 4-6 se ponavljaju dok god se za status ne dobije neko od završnih stanja („*finished*“ ili neka od grešaka).

Nekada je zgodno videti konkretan izlaz koji je studentski program proizveo. Ocenjivač pruža mogućnost da se za prosleđeni taskId, dobije kompresovani direktorijum u kome se nalaze izlazi iz studentskog programa za svaki test primer. Ovakav pristup problemu omogućava Veb aplikaciji da smanji mrežni transport time što će se ove informacije prosleđivati korisnicima tek na eksplicitni zahtev.

4.3 Klijentski deo aplikacije

Klijentski deo aplikacije predstavlja skup html stranica, njima pridruženih stilova, kao i skup skriptova, napisanih u programskom jeziku *JavaScript*.

Postoje tri glavne html stranice. Sve ostale stranice su parcijalne, tj. umeću se u neku od glavnih stranica.

Prva od njih je *index.html* koja služi isključivo za logovanje na sistem. Ova stranica sadrži formular na kom se nalaze polja za unos korisničkog imena i šifre i dugme za logovanje na sistem. Ukoliko neko ko nije ulogovan na sistem pokuša da uđe na bilo koju drugu stranicu sistema, biće preusmeren na pomenutu stranicu za logovanje.

Druga stranica je *startPage.php*. Ovo je osnovna stranica za celokupnu kako studentsku, tako i nastavničku aplikaciju. Na njoj se nalaze samo navigacija,

donje zaglavlje (eng. footer) i mesto na koje će biti umetane parcijalne html stranice. Prethodno je pomenuto da je pri izradi aplikacije intenzivno korišćena biblioteka *Angular.js*. Za umetanje parcijalnih stranica upotrebljava se direktiva *ng-view* ove biblioteke, koja obezbeđuje da se na mestu na kom je direktiva navedena ubaci odgovarajuća parcijalna stranica. Sledeće pitanje koje se postavlja jeste na koji način se postiže da se u pravom trenutku učita prava parcijalna stranica. Ovo se postiže takozvanim rutiranjem (*eng. routing*). To je još jedna funkcionalnost koju nam obezbeđuje *Angular.js* biblioteka. Sve što je potrebno uraditi je navesti kojoj putanji u URI-ju odgovara koja parcijalna stranica (programski kod 13).

```
drWebgradeApp.config(['$routeProvider', function ($routerProvider){
    $routerProvider.
        when('/professor/:professorId', {
            templateUrl: 'Partial/p_homePage.html'
        }).
        when('/professor/:professorId/course/:courseId', {
            templateUrl: 'Partial/p_homeworkList.html'
        }).
        when('/professor/:professorId/course/:courseId/hw/:homeworkId', {
            templateUrl: 'Partial/p_homeworkDetails.html'
        }).
        when('/professor/:professorId/course/:courseId/hws/:homeworkId',
{
            templateUrl: 'Partial/p_homeworkSolutions.html'
        }).
        when('/student/:studentId', {
            templateUrl: 'Partial/s_homePage.html'
        }).
        when('/course/:courseId', {
            templateUrl: 'Partial/p_homeworkList.html'
        }).
        when('/s_homeworkList/:studentId/:courseId', {
            templateUrl: 'Partial/s_homeworkList.html'
        }).
        when('/s_homeworkDetails/:studentId/:courseId/:homeworkId', {
            templateUrl: 'Partial/s_homeworkDetails.html'
        }).
        when('/assignmentList', {
            templateUrl: 'Partial/a_assignmentDomainList.html'
        }).
        when('/assignmentList/:domain', {
            templateUrl: 'Partial/a_assignmentCategoryList.html'
        }).
        when('/assignmentList/:domain/:assignmentId', {
            templateUrl: 'Partial/a_assignmentDetails.html'
        }).
        when('/assignmentRangList', {
            templateUrl: 'Partial/a_rangList.html'
        }).
        when('/addNewAssignment', {
            templateUrl: 'Partial/a_addNewAssignment.html'
        })
        .otherwise({
```

```
            redirectTo:'/'
    });
```

Programski kod 13: Rutiranje korišćenjem ngRoute modula

Pored toga što obezbeđuje učitavanje odgovarajuće parcijalne stranice, ona nam omogućava i prosledjivanje parametara u putanji URI-ja. Na primer, ukoliko se unese putanja "*.../student/1*" treba da se učita početna stranica za studenta čiji je identifikator 1. Kao što može da se vidi u programskom kodu 13, ovaj identifikator je imenovan sa ***:studentId.*** U odgovarajućoj *JavaScript* datoteci koja učitava sve informacije o tom studentu, moguće je pristupiti unešenom parametru navođenjem: ***$routeParams.studentId***.

Dalje, potrebno je povezati svako od pomenutih parcijalnih stranica sa nekom JavaScript datotekom, kako bi se za svaku stranicu obezbedili odgovarajući podaci. Ovo se postiže korišćenjem *ng-controller* direktive. Prvo je potrebno definisati kontroler u okvoru JavaScript datoteke (programski kod 14).

```
angular.module('studentModule').controller('StudentInfoController',
        ['$http', '$routeParams', 'configConstants',
        function($http, $routeParams, $configConstants){

    //Authorisation check
    $http({
        method: 'POST',
        async: false,
        url: $configConstants.SERVER_URL+
             $configConstants.USER_AUTHENTICATION_SERVICE,
        data: { action:'getLoggedUser'}
    }).success(function (data) {
        if (data.userTypeName != "student" ||
            data.userId != $routeParams.studentId) {
            location.hash = "forbiddenPage";
        }
    });

    //Fields
    this.courses=[];
    this.studentId = $routeParams.studentId;
    var self = this;
    //Init
    $http({
        url: $configConstants.SERVER_URL+
$configConstants.STUDENT_SERVICE,
        method: 'GET',
        params: {action: 'getStudentInfo', studentId:this.studentId }
    }).success(function(data){
        self.courses=data.courses;
        if(data.image != "")
            $('#studentProfileImage').attr('src', ''+
data.image);
        else
            $('#studentProfileImage').attr('src',
'../Images/noimage.png');
        self.firstName = data.firstName;
        self.lastName= data.lastName;
```

```
        self.notifications = data.notifications;
        $http({
            url: $configConstants.SERVER_URL+
$configConstants.STUDENT_SERVICE,
            method: 'POST',
            data: {action: 'seeNotifications'}
        });
    });



    //Redirect to chosen homework
    this.showHomeworkList = function(courseId, studentId){
      window.location = "../Views/startPage.php#/s_homeworkList/" +
                        studentId + "/" + courseId;
    }
}]);
```

Programski kod 14: Kompletan kontroler za studentsku početnu stranicu.

Kontroler mora biti definisan u okviru nekog modula (u ovom slučaju to je *studentModule*). Pri definiciji, prvi argument predstavlja naziv kontrolera. Ovo je bitno, jer će se taj naziv koristiti u okviru html stranica za povezivanje stranice i kontrolera. Sledeći argument je niz koji sadrži listu komponenata (koju mogu činiti provajderi kao što je *$routeProvider*, servisi kao što je *$http* i slično), a zatim funkciju koja sadrži svu logiku nekog dela klijentske aplikacije (u ovom slučaju studentske početne stranice) i koja koristi prethodno navedene komponente.

Kao što je prethodno pomenuto, svakoj parcijalnoj stranici je potrebno pridružiti odgovarajući kontroler. Ovo se postiže *ng-controller* direktivom, prikazanom u programskom kodu 15. Kada se u ovoj direktivi navede, na primer *StrudentInfoController as si*, to označava da u nastavku dokumenta umesto punog imena kontrolera može da se piše samo *si*.

Primetimo da ***si.studentId*** iz programskog koda 15 zapravo predstavlja ***self.studentId*** iz programskog koda 14. Na ovaj način se vrši povezivanje svih podataka iz kontrolera sa poljima u html datotekama. Treba imati na umu da je ovo povezivanje dvostruko – promene u kontroleru menjaju podatke na html stranici, ali i promene u okviru html stranice dovode do promene podataka u kontroleru (nema potrebe za eksplicitnim dohvatanjem podataka).

Bitno je napomenuti da objekat ***si*** ima domen nivo bloka html elementa u kom je naveden.

```
<div ng-controller="StudentInfoController as si" class="...">
    <ol class="breadcrumb">
      <li class="active">
            <a href="#/student/{{si.studentId}}"> Početna </a>
      </li>
    </ol>
    <div class="col-md-6">
        <div class="panel panel-default">
            <div class="panel-heading">
                  <h3 class="panel-title"> Kursevi </h3>
```

```
            </div>
            <div class="panel-body">
                <div ng-repeat="course in si.courses" class="...">
                    <a ng-click= "si.showHomeworkList(course.id,
                                                      si.studentId)">
                        {{course.name}}</a>
                    <span class="badge">{{course.homeworkCount}}</span>
                </div>
            </div>
        </div>
    </div>

    <div class="col-md-6">
        <div class="panel panel-default">
            <div class="panel-heading">
                 <h3 class="panel-title">
                       Obavestenja
                 </h3>
            </div>
            <div class="panel-body" style="...">
                <div >
                    <pre ng-repeat="notification in si.notifications"
            ng-class="{'list-group-item-info': notification.isSeen==0,
                       'list-group-item-warning': notification.isSeen!=0}"
                        class="list-group-item">
                              {{notification.datetime}}<br/>
                              {{notification.message}}
                     </pre>
                </div>
            </div>
        </div>
    </div>
</div>
```

Programski kod 15: Kompletna stranica s_homePage.html

Ovim je u potpunosti opisan jedan deo studentske aplikacije. Svi ostali delovi su implementirani na analogan način (odgovarajuća paricijalna stranica se umeće u startPage.php, njoj se pridružuje odgovarajući kontroler koji vrši komunikaciju sa serverom, prikuplja potrebne podatke i time obezbeđuje da oni mogu biti prikazani u okviru stranice).

U svakom od kontrolera su primenjivane neke uobičajene prakse. Jedna od njih je da se prvo vrši provera autorizacije (da li korisnik koji je posetio stranicu ima pravo na to). Zatim se prikupljaju parametri iz putanje URI-ja (ukoliko ih ima). Nakon toga se vrši *http get* zahtev koji prikuplja podatke relevantne za tu stranicu. Ono što sledi posle toga varira od kontrolera do kontrolera, a zavisi od složenosti posla za koji je određeni kontroler zadužen. U radu je akcenat stavljen na to da jedan kontroler izvršava jedan celovit posao i da se poslovi koje izvršavaju dva kontrolera međusobno ne preklapaju. Takođe, svaki kontroler ovog dela aplikacije se nalazi u posebnoj datoteci i ukoliko je deo studentske aplikacije, ima naziv ***s_<ime>Controller*** (npr. *s_homePageController.js*), u slučaju nastavničke aplikacije ***p_<ime>Controller*** i u slučaju dela aplikacije gde se nalaze zadaci za vežbu ***a_<ime>Controller***.

Pored ovoga, i odgovarajuće parcijalne stranice imaju identična imena (npr. *s_homePage.html*), tako da je jednoznačno određeno koji kontroler odgovara kojoj stranici.

Treći veliki deo aplikacije je onaj namenjen moderatoru. Ovo je poseban deo jer moderator ima potpuno drugaciju ulogu od ostalih korisnika sistema. On je taj koji je zadužen da odobrava zadatke, da prati rad sistema i ostalih korisnika i slično. Glavna stranica je *adminPage.php*. Strukturno je ista kao i *startPage.php*. Parcijalne stranice i kontroleri koji odgovaraju njima su smešteni u datoteke:

admin_assignmentProposalList.html, admin_assignmentProposalDetails.html i admin_assignmentProposalController.js.

4.4 Arhitektura aplikacije

Dr. Webgrade je implementiran kao troslojna veb aplikacija. Slojevi i tehnologije koje se koriste u njima su:

1. Prezentacioni sloj (HTML5, CSS, JavaScript uz intenzivno korišćenje biblioteke Angular.js)
2. Središnji sloj – zadužen za komunikaciju između prezentacionog sloja i sloja podataka, kao i za povezivanje sa ocenjivačem zadataka (koji nije deo ovog projekta). (PHP)
3. Sloj podataka (MySQL baza podataka)

Pošto serverski sloj samo prima klijentske zahteve i isporučuje podatke, a sva obrada tih podataka se vrši unutar klijentskih skriptova, može se reći i da je u pitanju arhitektura bogatih (teških) klijenata.

4.5 Korišćeni alati

Pri izradi rada i aplikacije, korišćeni su sledeći alati:

1. PHPStorm [1]
2. Visual Paradigm Community Edition, Verzija 12.1 [2]

5 DISKUSIJA

Rad opisuje osnovne elemente ovog sistema. Planirano je da se u budućnosti radi na proširivanju opisanog sistema.

Neki od planova su sledeći:

1. Bilo bi dobro da sistem omogući da nastavnik napiše sam svoju funkciju za poređenje rezultata (koju će zatim snimiti i nuditi kao opciju, da nastavnik ne bi morao svaki put iznova da je zadaje). Trenutna verzija vrši poređenje na jednakost (izuzimajući beline). U nekim slučajevima je zgodno drugačije porediti rezultate (npr. za poređenje decimalnih brojeva bi bilo zgodno porediti sa određenom tačnošću).

2. Bilo bi dobro da sistem omogući da nastavnik može da okači neki obrazac koda (eng. template code) koji bi bio dostupan studentu uz tekst zadatka i student bi morao da rešava zadatak modifikujući taj obrazac. Ovo je posebno zgodno u situacijama kada je cilj da student implementira samo neke funkcije, a da ostatak koda izgleda onako kako to nastavnik želi.
3. Bilo bi dobro da sistem omogući da nastavnik može da vidi izlaz iz studentskog programa. U trenutnoj verziji sistema, nastavnik ima uvid u to koliko poena je sistem dodelio studentskom rešenju, ali ne i detaljan opis ocenjivanja tog rešenja (koje test primere je program prošao, a koje ne i koja je razlika izmedju očekivanog izlaza i izlaza koji je proizveo studentski program).
4. Bilo bi dobro da sistem omogući ocenjivanje zadataka čije rešenje može da čini više datoteka (sa potencijalno različitim ekstenzijama).
5. Bilo bi dobro da sistem omogući pravljenje i ocenjivanje nekih verzija teorijskih testova.
6. Bilo bi dobro da sistem omogući pravljenje i slušanje lekcija na neku temu. Na primer, tema može da bude: „Programski jezik C“. Sistem treba da omogući da neko lice (nastavnik, moderator ili čak i student) može da formira lekcije (eng. tutorial) na tu temu tako što će navoditi naizmenično tekst koji opisuje neki segment jezika C i zadatke koji testiraju da li je slušalac dobro savladao prethodno opisano. Ideja je da slušalac ne može da pređe na sledeći deo lekcija dok ne reši uspešno sve zadatke iz prethodnog dela.
7. Bilo bi dobro da sistem omogući organizovanje takmičenja iz programiranja. Ukoliko neki korisnik sistema želi da organizuje takmičenje, dovoljno je da zada sve zadatke i da postavi datum i trajanje takmičenja. Studenti treba da imaju pristup listi postavljenih takmičenja i ukoliko su zainteresovani, treba da mogu da se prijave za bilo koje od njih. U trenutku početka takmičenja, zadaci treba da postanu vidljivi za sve studente koji su se prijavili. Tokom trajanja takmičenja, studenti bi mogli da vide trenutnu rang listu, čima bi imali uvid u to koliko su bolji ili lošiji od ostalih takmičara. Rana verzija ovog informacionog sistema je već korišćena na takmičenju u programiranju na Matermatičkom fakultetu - *„MATF Challenge 2015“* i ova ideja proizlazi iz uspešne realizacije tog takmičenja.
8. Bilo bi dobro da sistem omogući da grupa studenata može zajednički da rešava neki domaći zadatak.

6 ZAKLJUČAK

U radu su opisani projekat i implementacija veb zasnovanog informacionog sistema *Dr.Webgrade* koji bi trebalo da pomogne studentima programiranja da brže i lakše usvoje gradivo koje se odnosi na programerske predmete.

Sistem omogućava nastavnicima da zadaju kako domaće zadatke, tako i zadatke za vežbu. Sistem vrši ocenjivanje tih zadataka, na osnovu test primera koje je nastavnik zadao. Ukoliko želi, nastavnik može da pregleda kodove koje su studenti poslali i da eventualno koriguje broj poena koje im je sistem dodelio.

Studenti imaju mogućnost da pokušaju da reše sve pomenute zadatke, kao i da odmah dobiju informacije o tačnosti svog rešenja. Pored toga, mogu i da predlažu svoje zadatke. Obradu tih predloga vrši moderator sistema. On je jedini korisnik koji može direktno da dodaje nove zadatke za vežbu.

Za implementaciju sistema korišćeni su programski jezici PHP i JavaScript, baza podataka MySQL, jezik za obeležavanje HTML i jezik za uređivanje stila CSS. U okviru programskog jezika JavaScript, dosta je korišćena Angular.js biblioteka. Prednost ove biblioteke u odnosu na druge je to što je jako dobro dizajnirana, ima širok interfejs, jednostavna je za korišćenje, ima vrlo detaljnu dokumentaciju, itd.

Kao što se može zaključiti iz prethodnog poglavlja, mesta za proširenje i poboljšanje sistema ima dosta i na tome će se raditi u budućnosti. Nadam se da će ovaj sistem biti od koristiti i studentima i nastavnicima i da će se uspešno upotrebljavati na fakultetu u okviru što većeg broja kurseva.

DODATAK

A – Sheme svih tabela baze podataka

Primarni ključevi su obeleženi **<u>podebljanim slovima</u>,** dok su spoljni ključevi obeleženi *iskošenim slovima.*

Assignment

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
<u>Id</u>	Int	Ne	Nema	Identifikator zadatka
Text	Text	Ne	Nema	Tekst zadatka
TimeLimit	Varchar	Da	NULL	Vremensko ograničenje
MemoryLimit	Varchar	Da	NULL	Memorijsko ograničenje
TestCasesNum	Int	Ne	0	Broj postavljenih test primera
Title	Varchar	Ne	Nema	Naslov zadatka
IsHomework	TinyInt	Ne	1	Polje koje označava da li je zadatak domaći zadatak ili zadatak za vežbu
CategoryId	Int	Ne	0	Identifikator kategorije koja je pridružena zadatku za vežbu. Ako je u pitanju domaći zadatak, ovo polje ima vrednost 0.
Difficulty	Int	Ne	3	Težina zadatka (ceo broj između 1 i 5)
UserId	Int	Ne	Nema	Identifikator korisnika koji je autor zadatka.
State	Int	Ne	Nema	Stanje zadatka: 0 – neodobren predlog zadatka 1 – zadatak u pripremi 2 – kompletiran zadatak
Note	Varchar	Da	Nema	Napomena upućena moderatoru u slučaju da je u pitanju predlog zadatka za vežbu.

Homework

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
Id	Int	Ne	Nema	Identifikator domaćeg zadatka
ProfessorId	Int	Ne	Nema	Identifikator nastavnika koji je zadao domaći zadatak
CourseId	Int	Ne	Nema	Identifikator kursa u okviru kog je zadat domaći zadatak
AssignmentId	Int	Ne	Nema	Identifikator odgovarajućeg zadatka.
ActiveFrom	Date	Ne	Nema	Datum od kog domaći zadatak postaje aktivan
ActiveTo	Date	Ne	Nema	Datum do kog je domaći zadatak akvivan
Points	Int	Ne	Nema	Broj poena koje domaći zadatak nosi

ProgrammingLanguage

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
Id	Int	Ne	Nema	Identifikator programskog jezika
Name	Varchar	Ne	Nema	Naziv programskog jezika

HomeworkSolution

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
HomeworkId	Int	Ne	Nema	Identifikator domaćeg zadatka koji student rešava
StudentId	Int	Ne	Nema	Identifikator studenta koji rešava domaći zadatak

DateTime	Datetime	Ne	Nema	Datum i vreme slanja rešenja zadatka
Solution	Text	Da	NULL	Programski kod koji predstavlja rešenje zadatka
EarnedPoints	Int	Da	NULL	Poeni koje je sistem dodelio rešenju
TestResults	Text	Da	NULL	Rezultat testiranja rešenja na svim test primerima. Ovde se čuva JSON reprezentacija rezultata testiranja.Npr: {"state":"finished", "test0":"correct_result", "test1":"correct_result", "test2":"correct_result"}
TaskId	Varchar	Da	NULL	Identifikator procesa koji je testirao zadatak. Ova kolona nema značaj za funkcionalnost, već služi za testiranje i otkrivanje i korekciju grešaka.

User

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
Id	Int	Ne	Nema	Identifikator korisnika
Username	Varchar	Ne	Nema	Korisničko ime
Password	Varchar	Ne	Nema	Šifra
FirstName	Varchar	Ne	Nema	Ime korisnika
LastName	Varchar	Ne	Nema	Prezime korisnika
UserTypeId	Int	Ne	Nema	Identifikator tipa korisnika
Email	Varchar	Ne	Nema	Adresa elektronske pošte
Image	Mediumtext	Ne	Nema	Slika

Professor

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
Id	Int	Ne	Nema	Identifikator profesora

Student

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
Id	Int	Ne	Nema	Identifikator studenta
Index	Varchar	Ne	Nema	Broj indeksa

Course

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
Id	Int	Ne	Nema	Identifikator predmeta
Code	Varchar	Ne	Nema	Oznaka predmeta
Name	Varchar	Ne	Nema	Naziv predmeta
Semester	Int	Ne	Nema	Semestar slušanja

ProfessorCourse

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
ProfessorId	Int	Ne	Nema	Identifikator nastavnika
CourseId	Int	Ne	Nema	Identifikator predmeta

StudentCourse

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
StudentId	Int	Ne	Nema	Identifikator studenta
ProfessorId	Int	Ne	Nema	Identifikator nastavnika
CourseId	Int	Ne	Nema	Identifikator predmeta

HomeworkProgrammingLanguage

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
HomeworkId	Int	Ne	Nema	Identifikator domaćeg zadatka
ProgrammingLanguageId	Int	Ne	Nema	Identifikator programskog jezika

Category

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
Id	Int	Ne	Nema	Identifikator kategorije
Name	Varchar	Ne	Nema	Naziv kategorije
ParentCategory	Varchar	Ne	Nema	Naziv roditeljske kategorije - domena

AssignmentProgrammingLanguage

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
AssignmentId	Int	Ne	Nema	Identifikator zadatka
ProgrammingLanguageId	Int	Ne	Nema	Identifikator programskog jezika

AssignmentSubmission

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
AssignmentId	Int	Ne	Nema	Identifikator zadatka za vežbu
UserId	Int	Ne	Nema	Identifikator korisnika koji je poslao rešenje zadatka za vežbu
Solution	Longtext	Ne	Nema	Rešenje koje je student poslao
Points	Int	Ne	0	Procenat tačnosti rešenja
AttemptsNumber	Int	Ne	1	Ukupan broj pokušaja

				jednog studenta da reši zadatak
TaskId	Varchar	Ne	Nema	Identifikator procesa koji je testirao zadatak. Ova kolona nema značaj za funkcionalnost, već služi za testiranje i otkrivanje i korekciju grešaka.

Notification

Kolona	Tip	Null vrednost	Podrazumevana vrednost	Opis
Id	Int	Ne	Nema	Identifikator obaveštenja
Message	Varchar	Ne	Nema	Tekst obaveštenja
Datetime	Datetime	Ne	Nema	Datum i vreme slanja obaveštenja
IsSeen	TinyInt	Ne	Nema	Polje koje označava da li je korisnik do sada imao prilike da vidi obaveštenje
UserId	Int	Ne	Nema	Korisnik kome je obaveštenje namenjeno

REFERENCE

[1] PHP Storm, Zvanični sajt, *https://www.jetbrains.com/phpstorm/* (17.09.2015.)

[2] Visual Paradigm Community Edition, Verzija 12.1, *http://www.visual-paradigm.com/download/community.jsp* (17.09.2015.)

[3] phpMyAdmin, Zvanični sajt, *https://www.phpmyadmin.net/* (17.09.2015)

[4] PDO, Zvanična dokumentacija, *http://php.net/manual/en/book.pdo.php* (17.09.2015)

[5] Angular.js, Zvanična dokumentacija, https://docs.angularjs.org/api (18.09.2015)